
**Programming Languages — C++
Extensions for Library Fundamentals**

*Langages de programmation — Extensions C++ pour les
fondamentaux de bibliothèque*

Withdrawn

Withdrawn



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2015, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Foreword	6
1 General	7
1.1 Scope	7
1.2 Normative references	7
1.3 Namespaces, headers, and modifications to standard classes	7
1.4 Terms and definitions	8
1.5 Future plans (Informative)	8
1.6 Feature-testing recommendations (Informative)	8
2 Modifications to the C++ Standard Library	10
2.1 Uses-allocator construction	10
3 General utilities library	11
3.1 Utility components	11
3.1.1 Header <experimental/utility> synopsis	11
3.1.2 Class erased_type	11
3.2 Tuples	11
3.2.1 Header <experimental/tuple> synopsis	11
3.2.2 Calling a function with a tuple of arguments	12
3.3 Metaprogramming and type traits	12
3.3.1 Header <experimental/type_traits> synopsis	12
3.3.2 Other type transformations	15
3.4 Compile-time rational arithmetic	16
3.4.1 Header <experimental/ratio> synopsis	16
3.5 Time utilities	17
3.5.1 Header <experimental/chrono> synopsis	17
3.6 System error support	17
3.6.1 Header <experimental/system_error> synopsis	17
4 Function objects	18
4.1 Header <experimental/functional> synopsis	18
4.2 Class template function	19
4.2.1 function construct/copy/destroy	21
4.2.2 function modifiers	21
4.3 Searchers	22
4.3.1 Class template default_searcher	22
4.3.1.1 default_searcher creation functions	23
4.3.2 Class template boyer_moore_searcher	23
4.3.2.1 boyer_moore_searcher creation functions	24
4.3.3 Class template boyer_moore_horspool_searcher	24
4.3.3.1 boyer_moore_horspool_searcher creation functions	25
5 Optional objects	26
5.1 In general	26
5.2 Header <experimental/optional> synopsis	26
5.3 optional for object types	27
5.3.1 Constructors	29
5.3.2 Destructor	30
5.3.3 Assignment	31
5.3.4 Swap	33
5.3.5 Observers	33
5.4 In-place construction	34

5.5	No-value state indicator	34
5.6	Class <code>bad_optional_access</code>	35
5.7	Relational operators	35
5.8	Comparison with <code>nullopt</code>	35
5.9	Comparison with <code>T</code>	36
5.10	Specialized algorithms	37
5.11	Hash support	37
6	Class <code>any</code>	38
6.1	Header <code><experimental/any></code> synopsis	38
6.2	Class <code>bad_any_cast</code>	39
6.3	Class <code>any</code>	39
6.3.1	<code>any</code> construct/destroy	39
6.3.2	<code>any</code> assignments	40
6.3.3	<code>any</code> modifiers	41
6.3.4	<code>any</code> observers	41
6.4	Non-member functions	41
7	<code>string_view</code>	43
7.1	Header <code><experimental/string_view></code> synopsis	43
7.2	Class template <code>basic_string_view</code>	44
7.3	<code>basic_string_view</code> constructors and assignment operators	46
7.4	<code>basic_string_view</code> iterator support	47
7.5	<code>basic_string_view</code> capacity	47
7.6	<code>basic_string_view</code> element access	48
7.7	<code>basic_string_view</code> modifiers	48
7.8	<code>basic_string_view</code> string operations	49
7.8.1	Searching <code>basic_string_view</code>	50
7.9	<code>basic_string_view</code> non-member comparison functions	52
7.10	Inserters and extractors	53
7.11	Hash support	53
8	Memory	54
8.1	Header <code><experimental/memory></code> synopsis	54
8.2	Shared-ownership pointers	56
8.2.1	Class template <code>shared_ptr</code>	56
8.2.1.1	<code>shared_ptr</code> constructors	60
8.2.1.2	<code>shared_ptr</code> observers	61
8.2.1.3	<code>shared_ptr</code> casts	62
8.2.2	Class template <code>weak_ptr</code>	63
8.2.2.1	<code>weak_ptr</code> constructors	64
8.3	Type-erased allocator	64
8.4	Header <code><experimental/memory_resource></code> synopsis	64
8.5	Class <code>memory_resource</code>	65
8.5.1	Class <code>memory_resource</code> overview	65
8.5.2	<code>memory_resource</code> public member functions	66
8.5.3	<code>memory_resource</code> protected virtual member functions	66
8.5.4	<code>memory_resource</code> equality	67
8.6	Class template <code>polymorphic_allocator</code>	67
8.6.1	Class template <code>polymorphic_allocator</code> overview	67
8.6.2	<code>polymorphic_allocator</code> constructors	68
8.6.3	<code>polymorphic_allocator</code> member functions	68
8.6.4	<code>polymorphic_allocator</code> equality	70
8.7	template alias <code>resource_adaptor</code>	70
8.7.1	<code>resource_adaptor</code>	70

8.7.2	resource_adaptor_imp constructors	71
8.7.3	resource_adaptor_imp member functions	71
8.8	Access to program-wide memory_resource objects	72
8.9	Pool resource classes	72
8.9.1	Classes synchronized_pool_resource and unsynchronized_pool_resource	72
8.9.2	pool_options data members	74
8.9.3	pool resource constructors and destructors	75
8.9.4	pool resource members	75
8.10	Class monotonic_buffer_resource	76
8.10.1	Class monotonic_buffer_resource overview	76
8.10.2	monotonic_buffer_resource constructor and destructor	77
8.10.3	monotonic_buffer_resource members	78
8.11	Alias templates using polymorphic memory resources	78
8.11.1	Header <experimental/string> synopsis	78
8.11.2	Header <experimental/deque> synopsis	79
8.11.3	Header <experimental/forward_list> synopsis	79
8.11.4	Header <experimental/list> synopsis	79
8.11.5	Header <experimental/vector> synopsis	80
8.11.6	Header <experimental/map> synopsis	80
8.11.7	Header <experimental/set> synopsis	81
8.11.8	Header <experimental/unordered_map> synopsis	81
8.11.9	Header <experimental/unordered_set> synopsis	82
8.11.10	Header <experimental/regex> synopsis	82
9	Futures	83
9.1	Header <experimental/future> synopsis	83
9.2	Class template promise	83
9.3	Class template packaged_task	84
10	Algorithms library	86
10.1	Header <experimental/algorithm> synopsis	86
10.2	Search	86
10.3	Shuffling and sampling	87

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT), see the following URL: [Foreword — Supplementary information](#).

The committee responsible for this document is ISO/IEC JTC 1.

1 General

[general]

1.1 Scope

[general.scope]

- ¹ This technical specification describes extensions to the C++ Standard Library (1.2). These extensions are classes and functions that are likely to be used widely within a program and/or on the interface boundaries between libraries written by different organizations.
- ² This technical specification is non-normative. Some of the library components in this technical specification may be considered for standardization in a future version of C++, but they are not currently part of any C++ standard. Some of the components in this technical specification may never be standardized, and others may be standardized in a substantially changed form.
- ³ The goal of this technical specification is to build more widespread existing practice for an expanded C++ standard library. It gives advice on extensions to those vendors who wish to provide them.

1.2 Normative references

[general.references]

- ¹ The following referenced document is indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.
 - ISO/IEC 14882:2014, *Programming Language — C++*
- ² ISO/IEC 14882:— is herein called the *C++ Standard*. References to clauses within the C++ Standard are written as "C++14 §3.2". The library described in ISO/IEC 14882:— clauses 17–30 is herein called the *C++ Standard Library*.
- ³ Unless otherwise specified, the whole of the C++ Standard's Library introduction (C++14 §17) is included into this Technical Specification by reference.