
Programming languages — C++

Langages de programmation — C++





COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2020

All rights reserved. Unless otherwise specified, or required in the context of its implementation, 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
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Foreword	x
1 Scope	1
2 Normative references	2
3 Terms and definitions	3
4 General principles	9
4.1 Implementation compliance	9
4.2 Structure of this document	10
4.3 Syntax notation	11
5 Lexical conventions	12
5.1 Separate translation	12
5.2 Phases of translation	12
5.3 Character sets	13
5.4 Preprocessing tokens	14
5.5 Alternative tokens	15
5.6 Tokens	15
5.7 Comments	15
5.8 Header names	16
5.9 Preprocessing numbers	16
5.10 Identifiers	16
5.11 Keywords	17
5.12 Operators and punctuators	18
5.13 Literals	19
6 Basics	28
6.1 Preamble	28
6.2 Declarations and definitions	28
6.3 One-definition rule	30
6.4 Scope	34
6.5 Name lookup	40
6.6 Program and linkage	53
6.7 Memory and objects	57
6.8 Types	71
6.9 Program execution	78
7 Expressions	90
7.1 Preamble	90
7.2 Properties of expressions	91
7.3 Standard conversions	94
7.4 Usual arithmetic conversions	99
7.5 Primary expressions	99
7.6 Compound expressions	116
7.7 Constant expressions	147
8 Statements	153
8.1 Preamble	153
8.2 Labeled statement	154
8.3 Expression statement	154
8.4 Compound statement or block	154
8.5 Selection statements	154

8.6	Iteration statements	156
8.7	Jump statements	159
8.8	Declaration statement	161
8.9	Ambiguity resolution	161
9	Declarations	163
9.1	Preamble	163
9.2	Specifiers	165
9.3	Declarators	182
9.4	Initializers	197
9.5	Function definitions	213
9.6	Structured binding declarations	219
9.7	Enumerations	220
9.8	Namespaces	224
9.9	The <code>using</code> declaration	231
9.10	The <code>asm</code> declaration	236
9.11	Linkage specifications	237
9.12	Attributes	239
10	Modules	247
10.1	Module units and purviews	247
10.2	Export declaration	248
10.3	Import declaration	251
10.4	Global module fragment	252
10.5	Private module fragment	254
10.6	Instantiation context	255
10.7	Reachability	256
11	Classes	258
11.1	Preamble	258
11.2	Properties of classes	259
11.3	Class names	260
11.4	Class members	262
11.5	Unions	284
11.6	Local class declarations	287
11.7	Derived classes	287
11.8	Member name lookup	295
11.9	Member access control	298
11.10	Initialization	308
11.11	Comparisons	319
11.12	Free store	322
12	Overloading	324
12.1	Preamble	324
12.2	Overloadable declarations	324
12.3	Declaration matching	326
12.4	Overload resolution	327
12.5	Address of overloaded function	351
12.6	Overloaded operators	352
12.7	Built-in operators	356
12.8	User-defined literals	358
13	Templates	360
13.1	Preamble	360
13.2	Template parameters	361
13.3	Names of template specializations	365
13.4	Template arguments	368
13.5	Template constraints	373
13.6	Type equivalence	379

13.7	Template declarations	380
13.8	Name resolution	401
13.9	Template instantiation and specialization	417
13.10	Function template specializations	431
14	Exception handling	451
14.1	Preamble	451
14.2	Throwing an exception	452
14.3	Constructors and destructors	453
14.4	Handling an exception	454
14.5	Exception specifications	456
14.6	Special functions	458
15	Preprocessing directives	460
15.1	Preamble	460
15.2	Conditional inclusion	462
15.3	Source file inclusion	464
15.4	Module directive	465
15.5	Header unit importation	466
15.6	Macro replacement	467
15.7	Line control	472
15.8	Error directive	473
15.9	Pragma directive	473
15.10	Null directive	473
15.11	Predefined macro names	473
15.12	Pragma operator	475
16	Library introduction	477
16.1	General	477
16.2	The C standard library	478
16.3	Method of description	478
16.4	Library-wide requirements	484
17	Language support library	505
17.1	General	505
17.2	Common definitions	505
17.3	Implementation properties	509
17.4	Integer types	519
17.5	Startup and termination	520
17.6	Dynamic memory management	522
17.7	Type identification	529
17.8	Source location	530
17.9	Exception handling	532
17.10	Initializer lists	536
17.11	Comparisons	537
17.12	Coroutines	545
17.13	Other runtime support	549
18	Concepts library	552
18.1	General	552
18.2	Equality preservation	552
18.3	Header <concepts> synopsis	553
18.4	Language-related concepts	555
18.5	Comparison concepts	560
18.6	Object concepts	563
18.7	Callable concepts	563

19	Diagnostics library	565
19.1	General	565
19.2	Exception classes	565
19.3	Assertions	568
19.4	Error numbers	568
19.5	System error support	570
20	General utilities library	579
20.1	General	579
20.2	Utility components	579
20.3	Compile-time integer sequences	584
20.4	Pairs	585
20.5	Tuples	589
20.6	Optional objects	599
20.7	Variants	611
20.8	Storage for any type	622
20.9	Bitsets	627
20.10	Memory	632
20.11	Smart pointers	648
20.12	Memory resources	671
20.13	Class template <code>scoped_allocator_adaptor</code>	680
20.14	Function objects	684
20.15	Metaprogramming and type traits	707
20.16	Compile-time rational arithmetic	732
20.17	Class <code>type_index</code>	734
20.18	Execution policies	736
20.19	Primitive numeric conversions	738
20.20	Formatting	740
21	Strings library	759
21.1	General	759
21.2	Character traits	759
21.3	String classes	764
21.4	String view classes	790
21.5	Null-terminated sequence utilities	800
22	Containers library	805
22.1	General	805
22.2	Container requirements	805
22.3	Sequence containers	839
22.4	Associative containers	867
22.5	Unordered associative containers	885
22.6	Container adaptors	906
22.7	Views	914
23	Iterators library	921
23.1	General	921
23.2	Header <code><iterator></code> synopsis	921
23.3	Iterator requirements	928
23.4	Iterator primitives	948
23.5	Iterator adaptors	951
23.6	Stream iterators	972
23.7	Range access	978
24	Ranges library	980
24.1	General	980
24.2	Header <code><ranges></code> synopsis	980
24.3	Range access	985
24.4	Range requirements	989

24.5	Range utilities	992
24.6	Range factories	997
24.7	Range adaptors	1007
25	Algorithms library	1044
25.1	General	1044
25.2	Algorithms requirements	1044
25.3	Parallel algorithms	1046
25.4	Header <code><algorithm></code> synopsis	1049
25.5	Algorithm result types	1084
25.6	Non-modifying sequence operations	1087
25.7	Mutating sequence operations	1099
25.8	Sorting and related operations	1115
25.9	Header <code><numeric></code> synopsis	1142
25.10	Generalized numeric operations	1145
25.11	Specialized <code><memory></code> algorithms	1154
25.12	C library algorithms	1160
26	Numerics library	1161
26.1	General	1161
26.2	Numeric type requirements	1161
26.3	The floating-point environment	1161
26.4	Complex numbers	1162
26.5	Bit manipulation	1170
26.6	Random number generation	1173
26.7	Numeric arrays	1210
26.8	Mathematical functions for floating-point types	1229
26.9	Numbers	1244
27	Time library	1245
27.1	General	1245
27.2	Header <code><chrono></code> synopsis	1245
27.3	<i>Cpp17Clock</i> requirements	1259
27.4	Time-related traits	1259
27.5	Class template <code>duration</code>	1261
27.6	Class template <code>time_point</code>	1268
27.7	Clocks	1271
27.8	The civil calendar	1282
27.9	Class template <code>hh_mm_ss</code>	1311
27.10	12/24 hours functions	1313
27.11	Time zones	1313
27.12	Formatting	1326
27.13	Parsing	1330
27.14	Header <code><ctime></code> synopsis	1334
28	Localization library	1335
28.1	General	1335
28.2	Header <code><locale></code> synopsis	1335
28.3	Locales	1336
28.4	Standard <code>locale</code> categories	1342
28.5	C library locales	1374
29	Input/output library	1375
29.1	General	1375
29.2	Iostreams requirements	1375
29.3	Forward declarations	1376
29.4	Standard <code>iostream</code> objects	1378
29.5	Iostreams base classes	1380
29.6	Stream buffers	1395

29.7	Formatting and manipulators	1403
29.8	String-based streams	1427
29.9	File-based streams	1441
29.10	Synchronized output streams	1453
29.11	File systems	1458
29.12	C library files	1503
30	Regular expressions library	1506
30.1	General	1506
30.2	Definitions	1506
30.3	Requirements	1507
30.4	Header <code><regex></code> synopsis	1508
30.5	Namespace <code>std::regex_constants</code>	1512
30.6	Class <code>regex_error</code>	1515
30.7	Class template <code>regex_traits</code>	1515
30.8	Class template <code>basic_regex</code>	1517
30.9	Class template <code>sub_match</code>	1522
30.10	Class template <code>match_results</code>	1523
30.11	Regular expression algorithms	1528
30.12	Regular expression iterators	1532
30.13	Modified ECMAScript regular expression grammar	1537
31	Atomic operations library	1540
31.1	General	1540
31.2	Header <code><atomic></code> synopsis	1540
31.3	Type aliases	1544
31.4	Order and consistency	1544
31.5	Lock-free property	1546
31.6	Waiting and notifying	1546
31.7	Class template <code>atomic_ref</code>	1547
31.8	Class template <code>atomic</code>	1553
31.9	Non-member functions	1568
31.10	Flag type and operations	1568
31.11	Fences	1570
32	Thread support library	1572
32.1	General	1572
32.2	Requirements	1572
32.3	Stop tokens	1574
32.4	Threads	1579
32.5	Mutual exclusion	1586
32.6	Condition variables	1605
32.7	Semaphore	1612
32.8	Coordination types	1614
32.9	Futures	1617
Annex A	Grammar summary	1632
A.1	General	1632
A.2	Keywords	1632
A.3	Lexical conventions	1632
A.4	Basics	1636
A.5	Expressions	1637
A.6	Statements	1641
A.7	Declarations	1641
A.8	Modules	1647
A.9	Classes	1648
A.10	Overloading	1649
A.11	Templates	1649
A.12	Exception handling	1651

A.13	Preprocessing directives	1651
Annex B Implementation quantities		1653
Annex C Compatibility		1655
C.1	C++ and ISO C++ 2017	1655
C.2	C++ and ISO C++ 2014	1662
C.3	C++ and ISO C++ 2011	1666
C.4	C++ and ISO C++ 2003	1667
C.5	C++ and ISO C	1673
C.6	C standard library	1681
Annex D Compatibility features		1683
D.1	General	1683
D.2	Arithmetic conversion on enumerations	1683
D.3	Implicit capture of *this by reference	1683
D.4	Comma operator in subscript expressions	1683
D.5	Array comparisons	1683
D.6	Deprecated volatile types	1684
D.7	Redeclaration of static constexpr data members	1684
D.8	Non-local use of TU-local entities	1685
D.9	Implicit declaration of copy functions	1685
D.10	C headers	1685
D.11	Requires paragraph	1686
D.12	Relational operators	1686
D.13	char* streams	1687
D.14	Deprecated type traits	1694
D.15	Tuple	1695
D.16	Variant	1695
D.17	Deprecated iterator class template	1696
D.18	Deprecated move_iterator access	1696
D.19	Deprecated shared_ptr atomic access	1696
D.20	Deprecated basic_string capacity	1698
D.21	Deprecated standard code conversion facets	1698
D.22	Deprecated convenience conversion interfaces	1700
D.23	Deprecated locale category facets	1703
D.24	Deprecated filesystem path factory functions	1704
D.25	Deprecated atomic operations	1704
Bibliography		1706
Cross references		1707
Cross references from ISO C++ 2017		1731
Index		1734
Index of grammar productions		1768
Index of library headers		1773
Index of library names		1775
Index of library concepts		1847
Index of implementation-defined behavior		1850

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.

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) or the IEC list of patent declarations received (see <http://patents.iec.ch>).

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 voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

This sixth edition cancels and replaces the fifth edition (ISO/IEC 14882:2017), which has been technically revised.

The main changes compared to the previous edition are as follows:

- inclusion of the provisions of ISO/IEC TS 19217:2015, ISO/IEC TS 21425:2017, ISO/IEC TS 22277:2017, ISO/IEC TS 21544:2018, portions of ISO/IEC TS 19571:2016, and portions of ISO/IEC TS 19568:2017.
- addition of concepts, *requires-clauses*, *requires-expressions*, and `<concepts>` (18.3) header
- addition of coroutines, including `co_yield`, `co_await`, and `co_return` keywords and `<coroutine>` (17.12.2) header
- addition of modules, *import-declarations*, and *export-declarations*
- addition of three-way comparison, defaulted comparisons, rewriting of comparison operator expressions, and `<compare>` (17.11.1) header
- addition of designated initializers
- support for class types and floating-point types as the type of a non-type template parameter
- new attributes `[[no_unique_address]]`, `[[likely]]`, `[[unlikely]]`
- support for optional reason string in `[[nodiscard]]` attribute
- ability to require constant initialization with `constexpr` keyword
- ability to require constant evaluation with `constexpr` keyword
- extensions to constant evaluation
- support for controlling destruction in a class-specific operator delete function
- addition of `using enum` declaration
- addition of `char8_t` type
- support for an initializer statement in range-based for loops

- support for default member initializers for bit-fields
- support for parenthesized aggregate initialization
- extensions to lambda expressions
- extensions to structured bindings
- support for inline namespaces in nested namespace definitions
- support for conditionally-explicit member functions
- extensions to class template argument deduction
- reduced cases in which **typename** is required
- support for calling an undeclared *template-id* via argument-dependent name lookup
- revised memory model
- extended support for variadic macros with `__VA_OPT__`
- feature test macros and `<version>` (17.3.2) header
- addition of ranges and `<ranges>` (24.2) header
- addition of calendar and time zone support
- addition of text formatting library and `<format>` (20.20.1) header
- addition of `<barrier>` (32.8.3.2), `<latch>` (32.8.2.2), and `<semaphore>` (32.7.2) headers
- addition of mathematical constants library and `<numbers>` (26.9.1) header
- support for representing source locations and `<source_location>` (17.8.1) header
- addition of `span` view and `<span>` (22.7.2) header
- addition of joining thread class and `<stop_token>` (32.3.2) header
- extensions to atomic types and operations
- addition of `unsequenced` execution policy
- new utility functions, types, and templates in the standard library
- addition of bit manipulation library and `<bit>` (26.5.2) header
- addition of a synchronized buffered output stream and `<syncstream>` (29.10.1) header
- support for heterogeneous lookup for unordered containers
- support for element existence detection in associative containers
- support for move semantics in `<numeric>` (25.9) algorithms
- support for efficient access to the buffer of a `basic_stringbuf`
- extended constant expression evaluation support in the standard library

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

1 Scope

[intro.scope]

- ¹ This document specifies requirements for implementations of the C++ programming language. The first such requirement is that they implement the language, so this document also defines C++. Other requirements and relaxations of the first requirement appear at various places within this document.
- ² C++ is a general purpose programming language based on the C programming language as described in ISO/IEC 9899:2018 *Programming languages — C* (hereinafter referred to as the *C standard*). C++ provides many facilities beyond those provided by C, including additional data types, classes, templates, exceptions, namespaces, operator overloading, function name overloading, references, free store management operators, and additional library facilities.

2 Normative references

[intro.refs]

¹ The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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.

- (1.1) — ISO/IEC 2382, *Information technology — Vocabulary*
- (1.2) — ISO 8601:2004, *Data elements and interchange formats — Information interchange — Representation of dates and times*
- (1.3) — ISO/IEC 9899:2018, *Programming languages — C*
- (1.4) — ISO/IEC 9945:2003, *Information Technology — Portable Operating System Interface (POSIX¹)*
- (1.5) — ISO/IEC 10646, *Information technology — Universal Coded Character Set (UCS)*
- (1.6) — ISO/IEC 10646:2003,² *Information technology — Universal Multiple-Octet Coded Character Set (UCS)*
- (1.7) — ISO 80000-2:2009, *Quantities and units — Part 2: Mathematical signs and symbols to be used in the natural sciences and technology*
- (1.8) — Ecma International, *ECMAScript³ Language Specification*, Standard Ecma-262, third edition, 1999.

² The library described in ISO/IEC 9899:2018, Clause 7, is hereinafter called the *C standard library*.⁴

³ The operating system interface described in ISO/IEC 9945:2003 is hereinafter called *POSIX*.

⁴ The ECMAScript Language Specification described in Standard Ecma-262 is hereinafter called *ECMA-262*.

⁵ [Note 1: References to ISO/IEC 10646:2003 are used only to support deprecated features (D.21). — end note]

1) POSIX® is a registered trademark of the Institute of Electrical and Electronic Engineers, Inc. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of this product.

2) Cancelled and replaced by ISO/IEC 10646:2017.

3) ECMAScript® is a registered trademark of Ecma International. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of this product.

4) With the qualifications noted in [Clause 17](#) through [Clause 32](#) and in [C.6](#), the C standard library is a subset of the C++ standard library.