
Programming languages — C++

Langages de programmation — C++



This document is a preview generated by EASS



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2017, 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	xi
1 Scope	1
2 Normative references	2
3 Terms and definitions	3
4 General principles	7
4.1 Implementation compliance	7
4.2 Structure of this document	8
4.3 Syntax notation	8
4.4 The C++ memory model	8
4.5 The C++ object model	9
4.6 Program execution	11
4.7 Multi-threaded executions and data races	15
4.8 Acknowledgments	20
5 Lexical conventions	22
5.1 Separate translation	22
5.2 Phases of translation	22
5.3 Character sets	23
5.4 Preprocessing tokens	24
5.5 Alternative tokens	25
5.6 Tokens	25
5.7 Comments	26
5.8 Header names	26
5.9 Preprocessing numbers	26
5.10 Identifiers	27
5.11 Keywords	28
5.12 Operators and punctuators	29
5.13 Literals	29
6 Basic concepts	39
6.1 Declarations and definitions	39
6.2 One-definition rule	41
6.3 Scope	44
6.4 Name lookup	50
6.5 Program and linkage	63
6.6 Start and termination	66
6.7 Storage duration	70
6.8 Object lifetime	74
6.9 Types	77
6.10 Lvalues and rvalues	83
6.11 Alignment	84

7	Standard conversions	86
7.1	Lvalue-to-rvalue conversion	87
7.2	Array-to-pointer conversion	87
7.3	Function-to-pointer conversion	88
7.4	Temporary materialization conversion	88
7.5	Qualification conversions	88
7.6	Integral promotions	89
7.7	Floating-point promotion	89
7.8	Integral conversions	89
7.9	Floating-point conversions	90
7.10	Floating-integral conversions	90
7.11	Pointer conversions	90
7.12	Pointer to member conversions	90
7.13	Function pointer conversions	91
7.14	Boolean conversions	91
7.15	Integer conversion rank	91
8	Expressions	93
8.1	Primary expressions	96
8.2	Postfix expressions	109
8.3	Unary expressions	120
8.4	Explicit type conversion (cast notation)	129
8.5	Pointer-to-member operators	130
8.6	Multiplicative operators	131
8.7	Additive operators	131
8.8	Shift operators	132
8.9	Relational operators	133
8.10	Equality operators	133
8.11	Bitwise AND operator	135
8.12	Bitwise exclusive OR operator	135
8.13	Bitwise inclusive OR operator	135
8.14	Logical AND operator	135
8.15	Logical OR operator	135
8.16	Conditional operator	136
8.17	Throwing an exception	137
8.18	Assignment and compound assignment operators	137
8.19	Comma operator	138
8.20	Constant expressions	139
9	Statements	144
9.1	Labeled statement	145
9.2	Expression statement	145
9.3	Compound statement or block	145
9.4	Selection statements	145
9.5	Iteration statements	148
9.6	Jump statements	150
9.7	Declaration statement	152
9.8	Ambiguity resolution	153

10	Declarations	155
10.1	Specifiers	157
10.2	Enumeration declarations	174
10.3	Namespaces	178
10.4	The <code>asm</code> declaration	191
10.5	Linkage specifications	191
10.6	Attributes	194
11	Declarators	201
11.1	Type names	202
11.2	Ambiguity resolution	203
11.3	Meaning of declarators	204
11.4	Function definitions	216
11.5	Structured binding declarations	219
11.6	Initializers	220
12	Classes	237
12.1	Class names	239
12.2	Class members	241
12.3	Unions	251
12.4	Local class declarations	254
13	Derived classes	255
13.1	Multiple base classes	256
13.2	Member name lookup	258
13.3	Virtual functions	261
13.4	Abstract classes	265
14	Member access control	267
14.1	Access specifiers	268
14.2	Accessibility of base classes and base class members	269
14.3	Friends	272
14.4	Protected member access	275
14.5	Access to virtual functions	276
14.6	Multiple access	276
14.7	Nested classes	276
15	Special member functions	278
15.1	Constructors	278
15.2	Temporary objects	281
15.3	Conversions	283
15.4	Destructors	286
15.5	Free store	289
15.6	Initialization	291
15.7	Construction and destruction	298
15.8	Copying and moving class objects	301
16	Overloading	309
16.1	Overloadable declarations	309
16.2	Declaration matching	311
16.3	Overload resolution	312

16.4	Address of overloaded function	333
16.5	Overloaded operators	334
16.6	Built-in operators	339
17	Templates	342
17.1	Template parameters	343
17.2	Names of template specializations	347
17.3	Template arguments	348
17.4	Type equivalence	354
17.5	Template declarations	355
17.6	Name resolution	373
17.7	Template instantiation and specialization	388
17.8	Function template specializations	400
17.9	Deduction guides	421
18	Exception handling	422
18.1	Throwing an exception	423
18.2	Constructors and destructors	425
18.3	Handling an exception	425
18.4	Exception specifications	427
18.5	Special functions	430
19	Preprocessing directives	432
19.1	Conditional inclusion	433
19.2	Source file inclusion	435
19.3	Macro replacement	436
19.4	Line control	441
19.5	Error directive	442
19.6	Pragma directive	442
19.7	Null directive	442
19.8	Predefined macro names	442
19.9	Pragma operator	444
20	Library introduction	445
20.1	General	445
20.2	The C standard library	446
20.3	Definitions	446
20.4	Method of description (Informative)	449
20.5	Library-wide requirements	454
21	Language support library	476
21.1	General	476
21.2	Common definitions	476
21.3	Implementation properties	481
21.4	Integer types	490
21.5	Start and termination	491
21.6	Dynamic memory management	492
21.7	Type identification	500
21.8	Exception handling	502
21.9	Initializer lists	507
21.10	Other runtime support	508

22	Diagnostics library	511
22.1	General	511
22.2	Exception classes	511
22.3	Assertions	515
22.4	Error numbers	515
22.5	System error support	517
23	General utilities library	528
23.1	General	528
23.2	Utility components	528
23.3	Compile-time integer sequences	536
23.4	Pairs	537
23.5	Tuples	541
23.6	Optional objects	553
23.7	Variants	567
23.8	Storage for any type	580
23.9	Bitsets	586
23.10	Memory	592
23.11	Smart pointers	607
23.12	Memory resources	634
23.13	Class template <code>scoped_allocator_adaptor</code>	645
23.14	Function objects	651
23.15	Metaprogramming and type traits	675
23.16	Compile-time rational arithmetic	699
23.17	Time utilities	702
23.18	Class <code>type_index</code>	719
23.19	Execution policies	720
24	Strings library	723
24.1	General	723
24.2	Character traits	723
24.3	String classes	729
24.4	String view classes	762
24.5	Null-terminated sequence utilities	772
25	Localization library	778
25.1	General	778
25.2	Header <code><locale></code> synopsis	778
25.3	Locales	780
25.4	Standard <code>locale</code> categories	787
25.5	C library locales	825
26	Containers library	826
26.1	General	826
26.2	Container requirements	826
26.3	Sequence containers	864
26.4	Associative containers	896
26.5	Unordered associative containers	918
26.6	Container adaptors	942

27 Iterators library	952
27.1 General	952
27.2 Iterator requirements	952
27.3 Header <code><iterator></code> synopsis	958
27.4 Iterator primitives	961
27.5 Iterator adaptors	964
27.6 Stream iterators	977
27.7 Range access	984
27.8 Container access	985
28 Algorithms library	986
28.1 General	986
28.2 Header <code><algorithm></code> synopsis	986
28.3 Algorithms requirements	1005
28.4 Parallel algorithms	1006
28.5 Non-modifying sequence operations	1009
28.6 Mutating sequence operations	1017
28.7 Sorting and related operations	1027
28.8 C library algorithms	1046
29 Numerics library	1047
29.1 General	1047
29.2 Definitions	1047
29.3 Numeric type requirements	1047
29.4 The floating-point environment	1048
29.5 Complex numbers	1049
29.6 Random number generation	1059
29.7 Numeric arrays	1102
29.8 Generalized numeric operations	1122
29.9 Mathematical functions for floating-point types	1136
30 Input/output library	1153
30.1 General	1153
30.2 Iostreams requirements	1154
30.3 Forward declarations	1154
30.4 Standard istream objects	1156
30.5 Iostreams base classes	1158
30.6 Stream buffers	1175
30.7 Formatting and manipulators	1184
30.8 String-based streams	1211
30.9 File-based streams	1221
30.10 File systems	1235
30.11 C library files	1288
31 Regular expressions library	1292
31.1 General	1292
31.2 Definitions	1292
31.3 Requirements	1293
31.4 Header <code><regex></code> synopsis	1295
31.5 Namespace <code>std::regex_constants</code>	1301
31.6 Class <code>regex_error</code>	1304

31.7	Class template <code>regex_traits</code>	1305
31.8	Class template <code>basic_regex</code>	1307
31.9	Class template <code>sub_match</code>	1313
31.10	Class template <code>match_results</code>	1318
31.11	Regular expression algorithms	1324
31.12	Regular expression iterators	1329
31.13	Modified ECMAScript regular expression grammar	1335
32	Atomic operations library	1338
32.1	General	1338
32.2	Header <code><atomic></code> synopsis	1338
32.3	Type aliases	1342
32.4	Order and consistency	1342
32.5	Lock-free property	1344
32.6	Class template <code>atomic</code>	1345
32.7	Non-member functions	1352
32.8	Flag type and operations	1352
32.9	Fences	1353
33	Thread support library	1355
33.1	General	1355
33.2	Requirements	1355
33.3	Threads	1358
33.4	Mutual exclusion	1363
33.5	Condition variables	1384
33.6	Futures	1391
A	Grammar summary	1408
A.1	Keywords	1408
A.2	Lexical conventions	1408
A.3	Basic concepts	1413
A.4	Expressions	1413
A.5	Statements	1417
A.6	Declarations	1418
A.7	Declarators	1422
A.8	Classes	1424
A.9	Derived classes	1425
A.10	Special member functions	1426
A.11	Overloading	1426
A.12	Templates	1426
A.13	Exception handling	1427
A.14	Preprocessing directives	1428
B	Implementation quantities	1430
C	Compatibility	1432
C.1	C++ and ISO C	1432
C.2	C++ and ISO C++ 2003	1441
C.3	C++ and ISO C++ 2011	1447
C.4	C++ and ISO C++ 2014	1449
C.5	C standard library	1453

D Compatibility features	1456
D.1 Redclaration of <code>static constexpr</code> data members	1456
D.2 Implicit declaration of copy functions	1456
D.3 Deprecated exception specifications	1456
D.4 C++ standard library headers	1456
D.5 C standard library headers	1457
D.6 <code>char*</code> streams	1457
D.7 <code>uncaught_exception</code>	1466
D.8 Old adaptable function bindings	1466
D.9 The default allocator	1471
D.10 Raw storage iterator	1472
D.11 Temporary buffers	1473
D.12 Deprecated type traits	1474
D.13 Deprecated iterator primitives	1475
D.14 Deprecated <code>shared_ptr</code> observers	1475
D.15 Deprecated standard code conversion facets	1475
D.16 Deprecated convenience conversion interfaces	1477
Bibliography	1482
Cross references	1483
Cross references from ISO C++ 2014	1504
Index	1506
Index of grammar productions	1539
Index of library names	1543
Index of implementation-defined behavior	1601

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

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 ISO documents 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 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 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 the following URL: www.iso.org/iso/foreword.html.

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

This fifth edition cancels and replaces the fourth edition (ISO/IEC 14882:2014), which has been technically revised.

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

- expression evaluation order is specified in more cases
- removal of trigraphs
- adjustments to value categories resulting in copy elision being mandatory
- additional character and floating point literal syntaxes
- lambda expressions extended to permit capture of `*this` and use in constant expressions
- initializer statements for `if` and `switch` statements
- addition of `constexpr` if statements
- range-based `for` statement generalized to support heterogeneous `begin` and `end` types
- addition of structured bindings
- addition of inline variables
- list initialization extended to support enumerations and aggregates with base classes
- message in `static_assert` is now optional
- addition of nested namespace definition syntax

- extended support for attributes
- exception specifications are now part of function types
- template argument deduction is now supported for class templates
- addition of fold expressions
- pack expansion can be performed on using declarations
- permitted forms of template parameters and template arguments have been generalized
- dynamic allocation is supported for over-aligned types
- preprocessor can detect presence of header files with `__has_include`
- new utility functions, types, and templates in the standard library, including
 - an `any` type
 - an `optional` class template
 - a `variant` class template
 - a `clamp` function
 - a `std::byte` type
 - a `not_fn` function
 - a `void_t` alias template
 - `conjunction`, `disjunction`, and `negation` templates
 - an `invoke` function, and `is_invocable` and `invoke_result` type traits
 - an `is_swappable` type trait
- extended constant expression evaluation support in the standard library
- elementary conversion functions between strings and numeric types added
- constructors for `pair` and `tuple` are conditionally-explicit
- `shared_ptr`s of array types now supported
- additional algorithms for managing uninitialized memory
- addition of polymorphic memory resources
- addition of substring search facilities providing the Boyer-Moore and Boyer-Moore-Horspool search algorithms
- addition of variable templates for type traits
- addition of a non-owning string view template
- ability to splice elements between containers for maps and sets
- better support for element insertion in unique-key maps
- support for incomplete types in containers
- addition of parallel algorithms
- addition of `sample` algorithm
- addition of mathematical special functions, and `gcd`, `lcm`, and three-argument `hypot` functions
- addition of support for operations on file systems
- addition of shared mutexes and variadic lock guards
- removal of deprecated features

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 9899:2011 *Programming languages — C* (hereinafter referred to as the *C standard*). In addition to the facilities provided by C, C++ provides 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) — Ecma International, *ECMAScript Language Specification*, Standard Ecma-262, third edition, 1999.
- (1.2) — ISO/IEC 2382 (all parts), *Information technology — Vocabulary*
- (1.3) — ISO/IEC 9899:2011, *Programming languages — C*
- (1.4) — ISO/IEC 9945:2003, *Information Technology — Portable Operating System Interface (POSIX)*
- (1.5) — ISO/IEC 10646-1:1993, *Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane*
- (1.6) — ISO/IEC/IEEE 60559:2011, *Information technology — Microprocessor Systems — Floating-Point arithmetic*
- (1.7) — ISO 80000-2:2009, *Quantities and units — Part 2: Mathematical signs and symbols to be used in the natural sciences and technology*

² The library described in Clause 7 of ISO/IEC 9899:2011 is hereinafter called the *C standard library*.¹

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*.

¹ With the qualifications noted in Clauses 21 through 33 and in C.5, the C standard library is a subset of the C++ standard library.