Wikipedia
The Free Encyclopedia

C++

C++	
Logo endorsed by the C++ standards committee	
Paradigms	Multi-paradigm: procedural, imperative, functional, object-oriented, generic, modular
Family	C
Designed by	Bjarne Stroustrup
Developer	ISO/IEC JTC 1 (Joint Technical Committee 1) / SC 22 (Subcommittee 22) / WG 21 (Working Group 21)
First appeared	1985
Stable release	C++23 (ISO/IEC 14882:2024) / 19 October 2024
Preview release	C++26 / 5 August 2025
Typing discipline	Static, strong, nominative, partially inferred
OS	Cross-platform
Filename extensions	.C, .cc, .cpp, .cxx, .c++, .h, .H, .hh, .hpp, .hxx, .h++ .cppm, .ixx[1]
Website	isocpp.org (https://isocpp.org/)
Major implementations	
GCC, LLVM Clang, Microsoft Visual C++, Embarcadero C++Builder, Intel C++ Compiler, IBM XL C++, EDG	

C++[b] is a high-level, general-purpose programming language created by Danish computer scientist Bjarne Stroustrup. First released in 1985 as an extension of the C programming language, adding object-oriented (OOP) features, it has since expanded significantly over time adding more OOP and other features; as of 1997/C++98 standardization, C++ has added functional features, in addition to facilities for low-level memory manipulation for systems like microcomputers or to make operating systems like Linux or Windows, and even later came features like generic programming (through the use of templates). C++ is usually implemented as a compiled language, and many vendors provide C++ compilers, including the Free Software Foundation, LLVM, Microsoft, Intel, Embarcadero, Oracle, and IBM.[14]

C++ was designed with systems programming and embedded, resource-constrained software and large systems in mind, with performance, efficiency, and flexibility of use as its design highlights.[15] C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications,[15] including desktop applications, video games, servers (e.g., e-commerce, web search, or databases), and performance-critical applications (e.g., telephone switches or space probes).[16]

C++ is standardized by the International Organization for Standardization (ISO), with the latest standard version ratified and published by ISO in October 2024 as *ISO/IEC 14882:2024* (informally known as C++23).[17] The C++ programming language was initially standardized in 1998 as *ISO/IEC 14882:1998*, which was then amended by the C++03, C++11, C++14, C++17, and C++20 standards. The current C++23 standard supersedes these with new features and an enlarged standard library. Before the initial standardization in 1998, C++ was developed by Stroustrup at Bell Labs since 1979 as an extension of the C language; he wanted an efficient and flexible language similar to C that also provided high-level features for program organization.[18] Since 2012, C++ has been on a three-year release schedule[19] with C++26 as the next planned standard.[20]

Influenced by

Ada, ALGOL 68,[2] BCPL,[3] C, CLU,[2] F#,[4][a] ML, Mesa,[2] Modula-2,[2] Simula, Smalltalk[2]

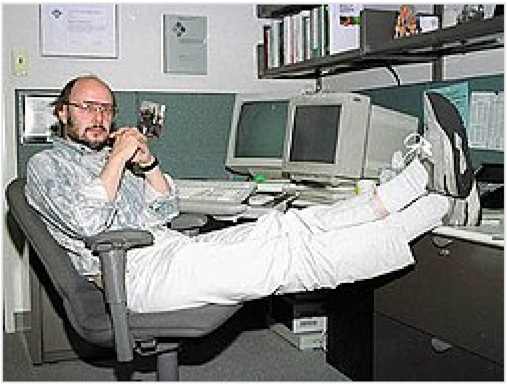
Influenced

Ada 95, C#,[5] C99, Carbon, Chapel,[6] Clojure,[7] D, Java,[8] JS++,[9] Lua,[10] Nim,[11] Objective-C++, Perl, PHP, Python,[12] Rust,[13] Seed7

C++ Programming at Wikibooks

History

In 1979, Bjarne Stroustrup, a Danish computer scientist, began work on "C with Classes", the predecessor to C++.[21] The motivation for creating a new language originated from Stroustrup's experience in programming for his PhD thesis. Stroustrup found that Simula had features that were very helpful for large software development, but the language was too slow for practical use, while BCPL was fast but too low-level to be suitable for large software development. When Stroustrup started working in AT&T Bell Labs, he had the problem of analyzing the UNIX kernel with respect to distributed computing. Remembering his PhD experience, Stroustrup set out to enhance the C language with Simula-like features.[22] C was chosen because it was general-purpose, fast, portable, and widely used. In addition to C and Simula's influences, other languages influenced this new language, including ALGOL 68, Ada, CLU, and ML.



Bjarne Stroustrup, the creator of C++, in his AT&T New Jersey office, c. 2000

Initially, Stroustrup's "C with Classes" added features to the C compiler, Cpre, including classes, derived classes, strong typing, inlining, and default arguments.[23]

A quiz on C++11 features being given in Paris in 2015

In 1982, Stroustrup started to develop a successor to C with Classes, which he named "C++" (`++` being the increment operator in C) after going through several other names. New features were added, including virtual functions, function and operator overloading, references, constants, type-safe free-store memory allocation (new/delete), improved type checking, and BCPL-style single-line comments with two forward slashes (`//`). Furthermore, Stroustrup developed a new, standalone compiler for C++, Cfront.

In 1984, Stroustrup implemented the first stream input/output library. The idea of providing an output operator rather than a named output function was suggested by Doug McIlroy[2] (who had previously suggested Unix pipes).

In 1985, the first edition of *The C++ Programming Language* was released, which became the definitive reference for the language, as there was not yet an official standard.[24] The first commercial implementation of C++ was released in October of the same year.[21]

In 1989, C++ 2.0 was released, followed by the updated second edition of *The C++ Programming Language* in 1991.[25] New features in 2.0 included multiple inheritance, abstract classes, static member functions, const member functions, and protected members. In 1990, *The Annotated C++ Reference Manual* was published. This work became the basis for the future standard. Later feature additions included templates, exceptions, namespaces, new casts, and a Boolean type.

In 1998, C++98 was released, standardizing the language, and a minor update (C++03) was released in 2003.

After C++98, C++ evolved relatively slowly until, in 2011, the C++11 standard was released, adding numerous new features, enlarging the standard library further, and providing more facilities to C++ programmers. After a minor C++14 update released in December 2014, various new additions were introduced in C++17.[26] After becoming finalized in February 2020,[27] a draft of the C++20 standard was approved on 4 September 2020, and officially published on 15 December 2020.[28][29]

On January 3, 2018, Stroustrup was announced as the 2018 winner of the Charles Stark Draper Prize for Engineering, "for conceptualizing and developing the C++ programming language".[30]

In December 2022, C++ ranked third on the TIOBE index, surpassing Java for the first time in the history of the index. As of November 2024, the language ranks second after Python, with Java being in third.[31]

In March 2025, Stroustrup issued a call for the language community to defend it. Since the language allows manual memory management, bugs that represent security risks such as buffer overflow may be introduced in programs when inadvertently misused by the programmer.[32]

Etymology

According to Stroustrup, "the name signifies the evolutionary nature of the changes from C."[33] This name is credited to Rick Mascitti (mid-1983)[23] and was first used in December 1983. When Mascitti was questioned informally in 1992 about the naming, he indicated that it was given in a tongue-in-cheek spirit. The name comes from C's `++` operator (which increments the value of a variable) and a common naming convention of using "+" to indicate an enhanced computer program.

During C++'s development period, the language had been referred to as "new C" and "C with Classes"[23][34] before acquiring its final name.

Philosophy

Throughout C++'s life, its development and evolution has been guided by a set of principles:[22]

- It must be driven by actual problems and its features should be immediately useful in real world programs.
- Every feature should be implementable (with a reasonably obvious way to do so).
- Programmers should be free to pick their own programming style, and that style should be fully supported by C++.
- Allowing a useful feature is more important than preventing every possible misuse of C++.
- It should provide facilities for organizing programs into separate, well-defined parts, and provide facilities for combining separately developed parts.

- No implicit violations of the type system (but allow explicit violations; that is, those explicitly requested by the programmer).
- User-created types need to have the same support and performance as built-in types.
- Unused features should not negatively impact created executables (e.g. in lower performance).
- There should be no language beneath C++ (except assembly language).
- C++ should work alongside other existing programming languages, rather than fostering its own separate and incompatible programming environment.
- If the programmer's intent is unknown, allow the programmer to specify it by providing manual control.

Standardization

C++ is standardized by an ISO working group known as JTC1/SC22/WG21. The working group holds three week-long meetings each year.[41] So far, it has published seven revisions of the C++ standard and is currently working on the next revision, C++26.

C++ standards

Year	ISO/IEC Standard	Informal name
1998	14882:1998[35]	C++98
2003	14882:2003[36]	C++03
2011	14882:2011[37]	C++11, C++0x
2014	14882:2014[38]	C++14, C++1y
2017	14882:2017[39]	C++17, C++1z
2020	14882:2020[40]	C++20, C++2a
2024	14882:2024[17]	C++23, C++2b
TBA		C++26, C++2c

In 1998, the ISO working group standardized C++ for the first time as *ISO/IEC 14882:1998*, which is informally known as *C++98*. In 2003, it published a new version of the C++ standard called *ISO/IEC 14882:2003*, which fixed problems identified in C++98.

The next major revision of the standard was informally referred to as "C++0x", but it was not released until 2011.[42] C++11 (14882:2011) included many additions to both the core language and the standard library.[37]

Scene during the C++ standards committee meeting in Stockholm in 1996

In 2014, C++14 (also known as C++1y) was released as a small extension to C++11, featuring mainly bug fixes and small improvements.[43] The Draft International Standard ballot procedures completed in mid-August 2014.[44]

After C++14, a major revision C++17, informally known as C++1z, was completed by the ISO C++ committee in mid July 2017 and was approved and published in December 2017.[45]

As part of the standardization process, ISO also publishes technical reports and specifications:

- ISO/IEC TR 18015:2006[46] on the use of C++ in embedded systems and on performance implications of C++ language and library features,
- ISO/IEC TR 19768:2007[47] (also known as the C++ Technical Report 1) on library extensions mostly integrated into C++11,
- ISO/IEC TR 29124:2010[48] on special mathematical functions, integrated into C++17,
- ISO/IEC TR 24733:2011[49] on decimal floating-point arithmetic,

- ISO/IEC TS 18822:2015[50] on the standard filesystem library, integrated into C++17,
- ISO/IEC TS 19570:2015[51] on parallel versions of the standard library algorithms, integrated into C++17,
- ISO/IEC TS 19841:2015[52] on software transactional memory,
- ISO/IEC TS 19568:2015[53] on a new set of library extensions, some of which are already integrated into C++17,
- ISO/IEC TS 19217:2015[54] on the C++ concepts, integrated into C++20,
- ISO/IEC TS 19571:2016[55] on the library extensions for concurrency, some of which are already integrated into C++20,
- ISO/IEC TS 19568:2017[56] on a new set of general-purpose library extensions,
- ISO/IEC TS 21425:2017[57] on the library extensions for ranges, integrated into C++20,
- ISO/IEC TS 22277:2017[58] on coroutines, integrated into C++20,
- ISO/IEC TS 19216:2018[59] on the networking library,
- ISO/IEC TS 21544:2018[60] on modules, integrated into C++20,
- ISO/IEC TS 19570:2018[61] on a new set of library extensions for parallelism
- ISO/IEC TS 23619:2021[62] on new extensions for reflective programming (reflection),
- ISO/IEC TS 9922:2024[63] on new set of concurrency extensions, and
- ISO/IEC TS 19568:2024[64] on another new set of library extensions.

More technical specifications are in development and pending approval.

Language

The C++ language has two main components: a direct mapping of hardware features provided primarily by the C subset, and zero-overhead abstractions based on those mappings. Stroustrup describes C++ as "a light-weight abstraction programming language [designed] for building and using efficient and elegant abstractions";[15] and "offering both hardware access and abstraction is the basis of C++. Doing it efficiently is what distinguishes it from other languages."[65]

C++ inherits most of C's syntax. A hello world program that conforms to the C standard is also a valid C++ hello world program. The following is Bjarne Stroustrup's version of the Hello world program that uses the C++ Standard Library stream facility to write a message to standard output:[66][67][c]

```cpp
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";
}
```

Standard library

The C++ standard consists of two parts: the core language and the standard library. C++ programmers expect the latter on every major implementation of C++; it includes aggregate types (vectors, lists, maps, sets, queues, stacks, arrays, tuples), algorithms (find, for_each, binary_search, random_shuffle, etc.), input/output facilities (iostream, for reading from and writing to the console and files), filesystem library, localisation support, smart pointers for automatic memory management, regular expression support, multi-threading library, atomics support (allowing a variable to be read or written to by at most one thread at a time without any external synchronisation), time utilities (measurement, getting current time, etc.), a system for converting error reporting that does not use C++ exceptions into C++ exceptions, a random number generator, and a slightly modified version of the C standard library (to make it comply with the C++ type system).

The draft "Working Paper" standard that became approved as C++98; half of its size was devoted to the C++ Standard Library.

The design of the C++ standard library, much like the C standard library, is minimalistic, and contains only core features for programming, lacking most of the more specialised features offered by the Java standard library or C# standard library. For more features, some third-party libraries such as Boost libraries and POCO C++ Libraries, which offer additional features, may be used to supplement the standard library.

A large part of the C++ library is based on the Standard Template Library (STL). Useful tools provided by the STL include containers as the collections of objects (such as vectors and lists), iterators that provide array-like access to containers, and algorithms that perform operations such as searching and sorting.

Furthermore, (multi)maps (associative arrays) and (multi)sets are provided, all of which export compatible interfaces. Therefore, using templates it is possible to write generic algorithms that work with any container or on any sequence defined by iterators.

As in C, the features of the library may be accessed by using the `#include` directive to include a standard header. The C++ Standard Library provides 105 standard headers, of which 27 are deprecated. With the introduction of modules in C++20, these headers may be accessed with `import`, and in C++23, the entire standard library can now be directly imported as module itself, with `import std;`. Currently, the C++ standard library provides two modules, `std` and `std.compat` (a compatibility module for `std` which exports C standard library facilities into the global namespace).

The standard incorporates the STL that was originally designed by Alexander Stepanov, who experimented with generic algorithms and containers for many years. When he started with C++, he finally found a language where it was possible to create generic algorithms (e.g., STL sort) that perform even better than, for example, the C standard library qsort, thanks to C++ features like using inlining and compile-time binding instead of function pointers. The standard does not refer to it as "STL", as it is merely a part of the standard library, but the term is still widely used to distinguish it from the rest of the standard library (input/output streams, internationalization, diagnostics, the C library subset, etc.).[68]

Most C++ compilers, and all major ones, provide a standards-conforming implementation of the C++ standard library.

C++ Core Guidelines

The C++ Core Guidelines[69] are an initiative led by Bjarne Stroustrup, the inventor of C++, and Herb Sutter, the convener and chair of the C++ ISO Working Group, to help programmers write 'Modern C++' by using best practices for the language standards C++11 and newer, and to help developers of compilers and static checking tools to create rules for catching bad programming practices.

The main aim is to efficiently and consistently write type and resource safe C++.

The Core Guidelines were announced[70] in the opening keynote at CPPCon 2015.

The Guidelines are accompanied by the Guideline Support Library (GSL),[71] a header only library of types and functions to implement the Core Guidelines and static checker tools for enforcing Guideline rules.[72]

Compatibility

To give compiler vendors greater freedom, the C++ standards committee decided not to dictate the implementation of name mangling, exception handling, and other implementation-specific features. The downside of this decision is that object code produced by different compilers is expected to be incompatible. There are, however, attempts to standardize compilers for particular machines or operating systems. For example, the Itanium C++ ABI is processor-independent (despite its name) and is implemented by GCC and Clang.[73]

With C

C++ is often considered to be a superset of C but this is not strictly true.[74] Most C code can easily be made to compile correctly in C++ but there are a few differences that cause some valid C code to be invalid or behave differently in C++. For example, C allows implicit conversion from `void*` to other pointer types but C++ does not (for type safety reasons). Also, C++ defines many new keywords, such as `new` and `class`, which may be used as identifiers (for example, variable names) in a C program.

Some incompatibilities have been removed by the 1999 revision of the C standard (C99), which now supports C++ features such as line comments (`//`) and declarations mixed with code. On the other hand, C99 introduced a number of new features that C++ did not support that were incompatible or redundant in C++, such as variable-length arrays, native complex-number types (however, the `std::complex` class in the C++ standard library provides similar functionality, although not code-compatible), designated initializers, compound literals, and the `restrict` keyword.[75] Some of the C99-introduced features were included in the subsequent version of the C++ standard, C++11 (out of those which were not redundant).[76][77][78] However, the C++11 standard introduces new incompatibilities, such as disallowing assignment of a string literal to a character pointer, which remains valid C.

To intermix C and C++ code, any function declaration or definition that is to be called from/used both in C and C++ must be declared with C linkage by placing it within an `extern "C" {/*...*/}` block. Such a function may not rely on features depending on name mangling (i.e., function overloading).

Inline assembly

Programs developed in C or C++ often utilize inline assembly to take advantage of its low-level functionalities, greater speed, and enhanced control compared to high-level programming languages[79][80] when optimizing for performance is essential. C++ provides support for embedding assembly language using asm declarations,[81] but the compatibility of inline assembly varies significantly between compilers and architectures. Unlike high-level language features such as Python or Java, assembly code is highly dependent on the underlying processor and compiler implementation.

Variations across compilers

Different C++ compilers implement inline assembly in distinct ways.

- GCC (GNU Compiler Collection) and Clang:[82] Use the GCC extended inline assembly syntax. Using `__asm__` keyword instead of `asm` when writing code that can be compiled with `-ansi` and `-std` options, which allows specifying input/output operands and clobbered registers. This approach is widely adopted, including by Intel[83] and IBM[84] compilers.
- MSVC (Microsoft Visual C++): The inline assembler is built into the compiler. Previously supported inline assembly via the `__asm` keyword, but this support has been removed in 64-bit mode, requiring separate .asm modules instead.[85]
- TI ARM Clang and Embedded Compilers:[86] Some embedded system compilers, like Texas Instruments' TI Arm Clang, allow inline assembly but impose stricter rules to avoid conflicts with register conventions and calling conventions.

Interoperability between C++ and Assembly

C++ provides two primary methods of integrating ASM code.

1. Standalone assembly files – Assembly code is written separately and linked with C++ code.[87]

2. Inline assembly – Assembly code is embedded within C++ code using compiler-specific extensions.

See also

Computer programming portal

- Carbon (programming language) — development at Google to potentially be a successor to C++
- Comparison of programming languages
- List of C++ compilers
- List of C++ software and tools
- List of C++ programming books
- Outline of C++
- Category:C++ libraries

Notes

a. For the idea of the C++20 stackless coroutines.

b. Pronounced /ˈsiː plʌs plʌs/ *SEE PLUSS PLUSS* and sometimes abbreviated as **CPP** or **CXX**.

c. This code is copied directly from Bjarne Stroustrup's errata page (p. 633). He addresses the use of `'\n'` rather than `std::endl`. Also see Can I write "void main()"? (http://www.stroustrup.com/bs_faq2.html#void-main) Archived (https://web.archive.org/web/20200702224848/http://www.stroustrup.com/bs_faq2.html#void-main) 2 July 2020 at the Wayback Machine for an explanation of the implicit `return 0;` in the `main` function. This implicit return is *not* available in other functions.

References

1. "Overview of modules in C++" (https://learn.microsoft.com/en-us/cpp/cpp/modules-cpp?view=msvc-170). Microsoft. 24 April 2023.
2. Stroustrup, Bjarne (1996). "A history of C++: 1979-1991". *History of programming languages---II*. ACM. pp. 699–769. doi:10.1145/234286.1057836 (https://doi.org/10.1145%2F234286.1057836).
3. Stroustrup, Bjarne (16 December 2021). "C++20: Reaching for the Aims of C++ - Bjarne Stroustrup - CppCon 2021" (https://www.youtube.com/watch?v=15QF2q66NhU). CppCon. Archived (https://web.archive.org/web/20211230092718/https://www.youtube.com/watch?v=15QF2q66NhU) from the original on 30 December 2021. Retrieved 30 December 2021.
4. Stroustrup, Bjarne (12 June 2020). "Thriving in a crowded and changing world: C++ 2006–2020" (https://doi.org/10.1145%2F3386320). *Proceedings of the ACM on Programming Languages*. **4** (HOPL). Association for Computing Machinery (ACM): 1–168. doi:10.1145/3386320 (https://doi.org/10.1145%2F3386320). ISSN 2475-1421 (https://search.worldcat.org/issn/2475-1421). S2CID 219603741 (https://api.semanticscholar.org/CorpusID:219603741).
5. Naugler, David (May 2007). "C# 2.0 for C++ and Java programmer: conference workshop". *Journal of Computing Sciences in Colleges*. **22** (5). "Although C# has been strongly influenced by Java it has also been strongly influenced by C++ and is best viewed as a descendant of both C++ and Java."
6. "Chapel spec (Acknowledgements)" (https://chapel-lang.org/spec/spec-0.98.pdf) (PDF). Cray Inc. 1 October 2015. Archived (https://web.archive.org/web/20180624150422/https://chapel-lang.org/spec/spec-0.98.pdf) (PDF) from the original on 24 June 2018. Retrieved 14 January 2016.
7. Fogus, Michael. "Rich Hickey Q&A" (https://web.archive.org/web/20170111184835/http://www.codequarterly.com/2011/rich-hickey/). *Code Quarterly*. Archived from the original (https://www.codequarterly.com/2011/rich-hickey/) on 11 January 2017. Retrieved 11 January 2017.
8. Harry. H. Chaudhary (28 July 2014). "Cracking The Java Programming Interview :: 2000+ Java Interview Que/Ans" (https://books.google.com/books?id=0rUtBAAAQBAJ&pg=PA133). Archived (https://web.archive.org/web/20210527025512/https://books.google.com/books?id=0rUtBAAAQBAJ&pg=PA133) from the original on 27 May 2021. Retrieved 29 May 2016.
9. Roger Poon (1 May 2017). "Scaling JS++: Abstraction, Performance, and Readability" (https://www.onux.com/jspp/blog/scaling-jspp-abstraction-performance-and-readability/). Archived (https://web.archive.org/web/20200511095442/https://www.onux.com/jspp/blog/scaling-jspp-abstraction-performance-and-readability/) from the original on 11 May 2020. Retrieved 21 April 2020.

10. "The evolution of an extension language: a history of Lua" (https://www.lua.org/history.html). *www.lua.org*. Retrieved 4 January 2023.
11. "FAQ Nim Programming Language" (https://nim-lang.org/faq.html). Archived (https://web.archive.org/web/20170711004631/https://nim-lang.org/faq.html) from the original on 11 July 2017. Retrieved 21 April 2020.
12. "9. Classes — Python 3.6.4 documentation" (https://docs.python.org/tutorial/classes.html). *docs.python.org*. Archived (https://web.archive.org/web/20121023030209/http://docs.python.org/tutorial/classes.html) from the original on 23 October 2012. Retrieved 9 January 2018.
13. "Influences - The Rust Reference" (https://doc.rust-lang.org/reference/influences.html). *doc.rust-lang.org*. Retrieved 4 January 2023.
14. Stroustrup, Bjarne (1997). "1". *The C++ Programming Language* (https://archive.org/details/cprogramminglang00stro_0) (Third ed.). Addison-Wesley. ISBN 0-201-88954-4. OCLC 59193992 (https://search.worldcat.org/oclc/59193992).
15. Stroustrup, B. (6 May 2014). "Lecture:The essence of C++. University of Edinburgh" (https://www.youtube.com/watch?v=86xWVb4XIyE). *YouTube*. Archived (https://web.archive.org/web/20150428003608/https://www.youtube.com/watch?v=86xWVb4XIyE) from the original on 28 April 2015. Retrieved 12 June 2015.
16. Stroustrup, Bjarne (17 February 2014). "C++ Applications" (http://www.stroustrup.com/applications.html). *stroustrup.com*. Archived (https://web.archive.org/web/20210404065717/https://www.stroustrup.com/applications.html) from the original on 4 April 2021. Retrieved 5 May 2014.
17. "ISO/IEC 14882:2024" (https://www.iso.org/standard/83626.html). International Organization for Standardization. Retrieved 21 October 2020.
18. "Bjarne Stroustrup's Homepage" (http://www.stroustrup.com). *www.stroustrup.com*. Archived (https://web.archive.org/web/20190514123147/http://www.stroustrup.com/) from the original on 14 May 2019. Retrieved 15 May 2013.
19. "C++ IS schedule" (http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2020/p1000r4.pdf) (PDF). Archived (https://web.archive.org/web/20200810105609/http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2020/p1000r4.pdf) (PDF) from the original on 10 August 2020. Retrieved 9 August 2020.
20. "C++; Where it's heading" (https://dzone.com/articles/c-where-is-it-heading-and-what-are-the-new-feature). Archived (https://web.archive.org/web/20181203104022/https://dzone.com/articles/c-where-is-it-heading-and-what-are-the-new-feature) from the original on 3 December 2018. Retrieved 3 December 2018.
21. Stroustrup, Bjarne (7 March 2010). "Bjarne Stroustrup's FAQ: When was C++ invented?" (http://www.stroustrup.com/bs_faq.html#invention). *stroustrup.com*. Archived (https://web.archive.org/web/20160206214150/http://www.stroustrup.com/bs_faq.html#invention) from the original on 6 February 2016. Retrieved 16 September 2010.
22. Stroustrup, Bjarne. "Evolving a language in and for the real world: C++ 1991-2006" (http://stroustrup.com/hopl-almost-final.pdf) (PDF). Archived (https://web.archive.org/web/20071120015600/http://www.research.att.com/~bs/hopl-almost-final.pdf) (PDF) from the original on 20 November 2007. Retrieved 14 August 2013.
23. Stroustrup, Bjarne. "A History of C ++ : 1979− 1991" (http://www.stroustrup.com/hopl2.pdf) (PDF). Archived (https://web.archive.org/web/20190202050609/http://www.stroustrup.com/hopl2.pdf) (PDF) from the original on 2 February 2019. Retrieved 18 July 2013.
24. Stroustrup, Bjarne. "The C++ Programming Language" (http://www.stroustrup.com/1st.html) (First ed.). Archived (https://web.archive.org/web/20120809032136/http://www.stroustrup.com/1st.html) from the original on 9 August 2012. Retrieved 16 September 2010.
25. Stroustrup, Bjarne. "The C++ Programming Language" (http://www.stroustrup.com/2nd.html) (Second ed.). Archived (https://web.archive.org/web/20120809032141/http://www.stroustrup.com/2nd.html) from the original on 9 August 2012. Retrieved 16 September 2010.

26. Sutter, Herb (30 June 2016). "Trip report: Summer ISO C++ standards meeting (Oulu)" (https://web.archive.org/web/20161008031743/https://herbsutter.com/2016/06/30/trip-report-summer-iso-c-standards-meeting-oulu/). *herbsutter.com*. Archived from the original (https://herbsutter.com/2016/06/30/trip-report-summer-iso-c-standards-meeting-oulu/) on 8 October 2016. "the next standard after C++17 will be C++20"
27. Dusíková, Hana (6 November 2019). "N4817: 2020 Prague Meeting Invitation and Information" (http://open-std.org/JTC1/SC22/WG21/docs/papers/2019/n4817.pdf) (PDF). Archived (https://web.archive.org/web/20191229102449/http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/n4817.pdf) (PDF) from the original on 29 December 2019. Retrieved 13 February 2020.
28. "Current Status" (https://isocpp.org/std/status). *isocpp.org*. Archived (https://web.archive.org/web/20200908083135/https://isocpp.org/std/status) from the original on 8 September 2020. Retrieved 7 September 2020.
29. "C++20 Approved -- Herb Sutter" (https://isocpp.org/blog/2020/09/cpp20-approved-herb-sutter). *isocpp.org*. Archived (https://web.archive.org/web/20200911150359/https://isocpp.org/blog/2020/09/cpp20-approved-herb-sutter) from the original on 11 September 2020. Retrieved 8 September 2020.
30. "Computer Science Pioneer Bjarne Stroustrup to Receive the 2018 Charles Stark Draper Prize for Engineering" (https://web.archive.org/web/20180103190112/https://www.nae.edu/177355.aspx) (Press release). National Academy of Engineering. 3 January 2018. Archived from the original (https://www.nae.edu/177355.aspx) on 3 January 2018. Retrieved 14 December 2021.
31. TIOBE (November 2024). "TIOBE Index for November 2024" (https://www.tiobe.com/tiobe-index/). *TIOBE.com*. TIOBE Company. Archived (https://web.archive.org/web/20241118090936/https://www.tiobe.com/tiobe-index/) from the original on 18 November 2024. Retrieved 18 November 2024.
32. Claburn, Thomas (2 March 2025). "C++ creator calls for help to defend programming language from 'serious attacks' " (https://www.theregister.com/2025/03/02/c_creator_calls_for_action/). *The Register*. Retrieved 5 March 2025.
33. "Bjarne Stroustrup's FAQ – Where did the name "C++" come from?" (http://www.stroustrup.com/bs_faq.html#name). Archived (https://web.archive.org/web/20160206214150/http://www.stroustrup.com/bs_faq.html#name) from the original on 6 February 2016. Retrieved 16 January 2008.
34. "C For C++ Programmers" (https://web.archive.org/web/20101117003419/http://www.ccs.neu.edu/course/com3620/parent/C-for-Java-C++/c-for-c++-alt.html). Northeastern University. Archived from the original (https://www.ccs.neu.edu/course/com3620/parent/C-for-Java-C++/c-for-c++-alt.html) on 17 November 2010. Retrieved 7 September 2015.
35. "ISO/IEC 14882:1998" (https://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?ics1=35&ics2=60&ics3=&csnumber=25845). International Organization for Standardization. Archived (https://web.archive.org/web/20170115080045/http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?ics1=35&ics2=60&ics3=&csnumber=25845) from the original on 15 January 2017. Retrieved 23 November 2018.
36. "ISO/IEC 14882:2003" (https://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?ics1=35&ics2=60&ics3=&csnumber=38110). International Organization for Standardization. Archived (https://web.archive.org/web/20210813193332/https://www.iso.org/standard/38110.html) from the original on 13 August 2021. Retrieved 23 November 2018.
37. "ISO/IEC 14882:2011" (https://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?ics1=35&ics2=60&ics3=&csnumber=50372). International Organization for Standardization. Archived (https://web.archive.org/web/20160527084921/http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?ics1=35&ics2=60&ics3=&csnumber=50372) from the original on 27 May 2016. Retrieved 23 November 2018.

38. "ISO/IEC 14882:2014" (https://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=64029&ICS1=35&ICS2=60). International Organization for Standardization. Archived (https://web.archive.org/web/20160429201210/http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=64029&ICS1=35&ICS2=60) from the original on 29 April 2016. Retrieved 23 November 2018.
39. "ISO/IEC 14882:2017" (https://www.iso.org/standard/68564.html). International Organization for Standardization. Archived (https://web.archive.org/web/20130129110331/http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50372) from the original on 29 January 2013. Retrieved 2 December 2017.
40. "ISO/IEC 14882:2020" (https://www.iso.org/standard/79358.html). International Organization for Standardization. Archived (https://web.archive.org/web/20201216154357/https://www.iso.org/standard/79358.html) from the original on 16 December 2020. Retrieved 16 December 2020.
41. "Meetings and Participation" (https://isocpp.org/std/meetings-and-participation). *News, Status & Discussion About Standard C++*. The Standard C++ Foundation. Retrieved 6 September 2025.
42. "We have an international standard: C++0x is unanimously approved" (https://herbsutter.com/2011/08/12/we-have-an-international-standard-c0x-is-unanimously-approved/). *Sutter's Mill*. 12 August 2011. Archived (https://web.archive.org/web/20180628182816/https://herbsutter.com/2011/08/12/we-have-an-international-standard-c0x-is-unanimously-approved/) from the original on 28 June 2018. Retrieved 23 November 2018.
43. "The Future of C++" (https://channel9.msdn.com/Events/Build/2012/2-005). Archived (https://web.archive.org/web/20181023213741/https://channel9.msdn.com/Events/Build/2012/2-005) from the original on 23 October 2018. Retrieved 23 November 2018 – via channel9.msdn.com.
44. "We have C++14! : Standard C++" (https://isocpp.org/blog/2014/08/we-have-cpp14). *isocpp.org*. Archived (https://web.archive.org/web/20140819083101/https://isocpp.org/blog/2014/08/we-have-cpp14) from the original on 19 August 2014. Retrieved 19 August 2014.
45. Sutter, Herb (15 July 2017). "Trip report: Summer ISO C++ standards meeting (Toronto)" (https://herbsutter.com/2017/07/15/trip-report-summer-iso-c-standards-meeting-toronto/). Archived (https://web.archive.org/web/20170806182136/https://herbsutter.com/2017/07/15/trip-report-summer-iso-c-standards-meeting-toronto/) from the original on 6 August 2017. Retrieved 4 August 2017.
46. "ISO/IEC TR 18015:2006" (https://www.iso.org/standard/43351.html). International Organization for Standardization. Archived (https://web.archive.org/web/20190115203236/https://www.iso.org/standard/43351.html) from the original on 15 January 2019. Retrieved 15 February 2019.
47. "ISO/IEC TR 19768:2007" (https://www.iso.org/standard/43289.html). International Organization for Standardization. Archived (https://web.archive.org/web/20160304045148/http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?ics1=35&ics2=60&ics3=&csnumber=43289) from the original on 4 March 2016. Retrieved 15 February 2019.
48. "ISO/IEC TR 29124:2010" (https://www.iso.org/standard/50511.html). International Organization for Standardization. Archived (https://web.archive.org/web/20190112054620/https://www.iso.org/standard/50511.html) from the original on 12 January 2019. Retrieved 15 February 2019.
49. "ISO/IEC TR 24733:2011" (https://www.iso.org/standard/38843.html). International Organization for Standardization. Archived (https://web.archive.org/web/20190115203556/https://www.iso.org/standard/38843.html) from the original on 15 January 2019. Retrieved 15 February 2019.

50. "ISO/IEC TS 18822:2015" (https://www.iso.org/standard/63483.html). International Organization for Standardization. Archived (https://web.archive.org/web/20190115201441/https://www.iso.org/standard/63483.html) from the original on 15 January 2019. Retrieved 15 February 2019.
51. "ISO/IEC TS 19570:2015" (https://www.iso.org/standard/65241.html). International Organization for Standardization. Archived (https://web.archive.org/web/20190115201509/https://www.iso.org/standard/65241.html) from the original on 15 January 2019. Retrieved 15 February 2019.
52. "ISO/IEC TS 19841:2015" (https://www.iso.org/standard/66343.html). International Organization for Standardization. Archived (https://web.archive.org/web/20190115201300/https://www.iso.org/standard/66343.html) from the original on 15 January 2019. Retrieved 15 February 2019.
53. "ISO/IEC TS 19568:2015" (https://www.iso.org/standard/65238.html). International Organization for Standardization. Archived (https://web.archive.org/web/20190115202436/https://www.iso.org/standard/65238.html) from the original on 15 January 2019. Retrieved 15 February 2019.
54. "ISO/IEC TS 19217:2015" (https://www.iso.org/standard/64031.html). International Organization for Standardization. Archived (https://web.archive.org/web/20190115201732/https://www.iso.org/standard/64031.html) from the original on 15 January 2019. Retrieved 15 February 2019.
55. "ISO/IEC TS 19571:2016" (https://www.iso.org/standard/65242.html). International Organization for Standardization. Archived (https://web.archive.org/web/20190115201226/https://www.iso.org/standard/65242.html) from the original on 15 January 2019. Retrieved 15 February 2019.
56. "ISO/IEC TS 19568:2017" (https://www.iso.org/standard/70587.html). International Organization for Standardization. Archived (https://web.archive.org/web/20190115202428/https://www.iso.org/standard/70587.html) from the original on 15 January 2019. Retrieved 15 February 2019.
57. "ISO/IEC TS 21425:2017" (https://www.iso.org/standard/70910.html). International Organization for Standardization. Archived (https://web.archive.org/web/20190115201957/https://www.iso.org/standard/70910.html) from the original on 15 January 2019. Retrieved 15 February 2019.
58. "ISO/IEC TS 22277:2017" (https://www.iso.org/standard/73008.html). International Organization for Standardization. Archived (https://web.archive.org/web/20190115202004/https://www.iso.org/standard/73008.html) from the original on 15 January 2019. Retrieved 15 February 2019.
59. "ISO/IEC TS 19216:2018" (https://www.iso.org/standard/64030.html). International Organization for Standardization. Archived (https://web.archive.org/web/20190115201923/https://www.iso.org/standard/64030.html) from the original on 15 January 2019. Retrieved 15 February 2019.
60. "ISO/IEC TS 21544:2018" (https://www.iso.org/standard/71051.html). International Organization for Standardization. Archived (https://web.archive.org/web/20190115201858/https://www.iso.org/standard/71051.html) from the original on 15 January 2019. Retrieved 15 February 2019.
61. "ISO/IEC TS 19570:2018" (https://www.iso.org/standard/70588.html). International Organization for Standardization. Archived (https://web.archive.org/web/20190115201519/https://www.iso.org/standard/70588.html) from the original on 15 January 2019. Retrieved 15 February 2019.
62. "ISO/IEC TS 23619:2021" (https://www.iso.org/standard/76425.html). International Organization for Standardization. Archived (https://web.archive.org/web/20181215153205/https://www.iso.org/standard/76425.html) from the original on 15 December 2018. Retrieved 11 October 2021.

63. "ISO/IEC TS 9922:2024" (https://www.iso.org/standard/83630.html). International Organization for Standardization. Archived (https://web.archive.org/web/20250401173108/https://www.iso.org/standard/83630.html) from the original on 1 April 2025. Retrieved 1 April 2025.
64. "ISO/IEC TS 19568:2024" (https://www.iso.org/standard/86293.html). International Organization for Standardization. Archived (https://web.archive.org/web/20250225094830/https://www.iso.org/standard/86293.html) from the original on 25 February 2025. Retrieved 1 April 2025.
65. B. Stroustrup (interviewed by Sergio De Simone) (30 April 2015). "Stroustrup: Thoughts on C++17 - An Interview" (https://www.infoq.com/news/2015/04/stroustrup-cpp17-interview). Archived (https://web.archive.org/web/20150708132155/http://www.infoq.com/news/2015/04/stroustrup-cpp17-interview) from the original on 8 July 2015. Retrieved 8 July 2015.
66. Stroustrup, Bjarne (2000). *The C++ Programming Language* (Special ed.). Addison-Wesley. p. 46. ISBN 0-201-70073-5.
67. Stroustrup, Bjarne. "Open issues for The C++ Programming Language (3rd Edition)" (http://www.stroustrup.com/3rd_issues.html). Archived (https://web.archive.org/web/20140505221112/http://www.stroustrup.com/3rd_issues.html) from the original on 5 May 2014. Retrieved 5 May 2014.
68. Graziano Lo Russo (2008). "An Interview with A. Stepanov" (http://www.stlport.org/resources/StepanovUSA.html). *stlport.org*. Archived (https://web.archive.org/web/20090304120628/http://www.stlport.org/resources/StepanovUSA.html) from the original on 4 March 2009. Retrieved 8 October 2015.
69. "C++ Core Guidelines" (https://isocpp.github.io/CppCoreGuidelines/). *isocpp.github.io*. Archived (https://web.archive.org/web/20200216200622/http://isocpp.github.io/CppCoreGuidelines/) from the original on 16 February 2020. Retrieved 9 February 2020.
70. "Bjarne Stroustrup announces C++ Core Guidelines : Standard C++" (https://isocpp.org/blog/2015/09/bjarne-stroustrup-announces-cpp-core-guidelines). *isocpp.org*. Archived (https://web.archive.org/web/20200511035635/https://isocpp.org/blog/2015/09/bjarne-stroustrup-announces-cpp-core-guidelines) from the original on 11 May 2020. Retrieved 31 March 2020.
71. "microsoft/GSL" (https://github.com/microsoft/GSL). 18 July 2021. Archived (https://web.archive.org/web/20210718130829/https://github.com/microsoft/GSL) from the original on 18 July 2021. Retrieved 18 July 2021 – via GitHub.
72. "Using the C++ Core Guidelines checkers" (https://docs.microsoft.com/en-us/cpp/code-quality/using-the-cpp-core-guidelines-checkers). *Microsoft Learn*. Archived (https://web.archive.org/web/20210813193329/https://docs.microsoft.com/en-us/cpp/code-quality/using-the-cpp-core-guidelines-checkers?view=msvc-160) from the original on 13 August 2021. Retrieved 31 March 2020.
73. "C++ ABI Summary" (https://mentorembedded.github.io/cxx-abi/). 20 March 2001. Archived (https://web.archive.org/web/20180710195559/https://mentorembedded.github.io/cxx-abi/) from the original on 10 July 2018. Retrieved 30 May 2006.
74. "Bjarne Stroustrup's FAQ – Is C a subset of C++?" (http://www.stroustrup.com/bs_faq.html#C-is-subset). Archived (https://web.archive.org/web/20160206214150/http://www.stroustrup.com/bs_faq.html#C-is-subset) from the original on 6 February 2016. Retrieved 5 May 2014.
75. "C9X – The New C Standard" (http://home.datacomm.ch/t_wolf/tw/c/c9x_changes.html). Archived (https://web.archive.org/web/20180621084656/http://home.datacomm.ch/t_wolf/tw/c/c9x_changes.html) from the original on 21 June 2018. Retrieved 27 December 2008.
76. "C++0x Support in GCC" (https://gcc.gnu.org/projects/cxx0x.html). Archived (https://web.archive.org/web/20100721215324/http://gcc.gnu.org/projects/cxx0x.html) from the original on 21 July 2010. Retrieved 12 October 2010.

77. "C++0x Core Language Features In VC10: The Table" (https://blogs.msdn.com/b/vcblog/archive/2010/04/06/c-0x-core-language-features-in-vc10-the-table.aspx). Archived (https://web.archive.org/web/20100821114635/http://blogs.msdn.com/b/vcblog/archive/2010/04/06/c-0x-core-language-features-in-vc10-the-table.aspx) from the original on 21 August 2010. Retrieved 12 October 2010.
78. "Clang - C++98, C++11, and C++14 Status" (https://clang.llvm.org/cxx_status.html). Clang.llvm.org. 12 May 2013. Archived (https://web.archive.org/web/20130704124639/http://clang.llvm.org/cxx_status.html) from the original on 4 July 2013. Retrieved 10 June 2013.
79. Bokil, Milind A. (2021). "Writing Assembly Routines within C/C++ and Java Programs (https://www.researchgate.net/publication/354744729_Writing_Assembly_Routines_within_CC_and_Java_Programs)". ResearchGate. Retrieved 1 April 2025.
80. De Vilhena, Paulo Emílio; Lahav, Ori; Vafeiadis, Viktor; Raad, Azalea (2024). "Extending the C/C++ Memory Model with Inline Assembly" (https://doi.org/10.1145/3689749). *Proceedings of the ACM on Programming Languages*. **8**: 1081–1107. arXiv:2408.17208 (https://arxiv.org/abs/2408.17208). doi:10.1145/3689749 (https://doi.org/10.1145%2F3689749).
81. cppreference.com contributors. "asm declaration (https://en.cppreference.com/w/cpp/language/asm)". *cppreference.com*. Retrieved 1 April 2025.
82. "Extended Asm (Using the GNU Compiler Collection)" (https://gcc.gnu.org/onlinedocs/gcc/Extended-Asm.html). *GCC Online Documentation*. GNU Project. Retrieved 1 April 2025.
83. Intel Corporation. "Inline Assembly (https://www.intel.com/content/www/us/en/docs/cpp-compiler/developer-guide-reference/2021-9/inline-assembly.html)". *Intel® C++ Compiler Classic Developer Guide and Reference*, Version 2021.9. Retrieved 1 April 2025.
84. IBM. "Inline assembly statements (IBM extension) (https://www.ibm.com/docs/en/xl-c-aix/13.1.3?topic=statements-inline-assembly-extension)". *IBM Documentation*. Retrieved 1 April 2025.
85. "Inline Assembler Overview" (https://learn.microsoft.com/en-us/cpp/assembler/inline/inline-assembler-overview?view=msvc-170). *Microsoft Learn*. Microsoft. Retrieved 1 April 2025.
86. "Interfacing C and C++ With Assembly Language" (https://software-dl.ti.com/codegen/docs/tiarmclang/compiler_tools_user_guide/compiler_manual/runtime_environment/interfacing-c-and-c-with-assembly-language-stdz0544217.html#interfacing-c-and-c-with-assembly-language). *Texas Instruments*. Texas Instruments Incorporated. 23 February 2025. Retrieved 1 April 2025.
87. "C++ to ASM linkage in GCC" (https://wiki.osdev.org/C%2B%2B_to_ASM_linkage_in_GCC). *OSDev Wiki*. Retrieved 1 April 2025.

Further reading

- Abrahams, David; Gurtovoy, Aleksey (2005). *C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond*. Addison-Wesley. ISBN 0-321-22725-5.
- Alexandrescu, Andrei (2001). *Modern C++ Design: Generic Programming and Design Patterns Applied*. Addison-Wesley. ISBN 0-201-70431-5.
- Alexandrescu, Andrei; Sutter, Herb (2004). *C++ Design and Coding Standards: Rules and Guidelines for Writing Programs*. Addison-Wesley. ISBN 0-321-11358-6.
- Becker, Pete (2006). *The C++ Standard Library Extensions : A Tutorial and Reference*. Addison-Wesley. ISBN 0-321-41299-0.
- Brokken, Frank (2010). *C++ Annotations* (https://www.icce.rug.nl/documents/cplusplus/). University of Groningen. ISBN 978-90-367-0470-0. Archived (https://web.archive.org/web/20100428014354/http://www.icce.rug.nl/documents/cplusplus/) from the original on 28 April 2010. Retrieved 28 April 2010.

- Coplien, James O. (1994) [reprinted with corrections, original year of publication 1992]. *Advanced C++: Programming Styles and Idioms* (https://archive.org/details/advancedcbsprogr00copl). Addison-Wesley. ISBN 0-201-54855-0.
- Dewhurst, Stephen C. (2005). *C++ Common Knowledge: Essential Intermediate Programming*. Addison-Wesley. ISBN 0-321-32192-8.
- Information Technology Industry Council (15 October 2003). *Programming languages – C++* (Second ed.). Geneva: ISO/IEC. 14882:2003(E).
- Josuttis, Nicolai M. (2012). *The C++ Standard Library, A Tutorial and Reference* (Second ed.). Addison-Wesley. ISBN 978-0-321-62321-8.
- Koenig, Andrew; Moo, Barbara E. (2000). *Accelerated C++ – Practical Programming by Example* (https://archive.org/details/acceleratedcprac2000koen). Addison-Wesley. ISBN 0-201-70353-X.
- Lippman, Stanley B.; Lajoie, Josée; Moo, Barbara E. (2011). *C++ Primer* (https://archive.org/details/cprimer0000lipp_5thed) (Fifth ed.). Addison-Wesley. ISBN 978-0-321-71411-4.
- Lippman, Stanley B. (1996). *Inside the C++ Object Model*. Addison-Wesley. ISBN 0-201-83454-5.
- Meyers, Scott (2005). *Effective C++* (https://archive.org/details/effectivec55spec00meye) (Third ed.). Addison-Wesley. ISBN 0-321-33487-6.
- Stroustrup, Bjarne (2013). *The C++ Programming Language* (Fourth ed.). Addison-Wesley. ISBN 978-0-321-56384-2.
- Stroustrup, Bjarne (1994). *The Design and Evolution of C++*. Addison-Wesley. ISBN 0-201-54330-3.
- Stroustrup, Bjarne (2014). *Programming: Principles and Practice Using C++* (Second ed.). Addison-Wesley. ISBN 978-0-321-99278-9.
- Sutter, Herb (2001). *More Exceptional C++: 40 New Engineering Puzzles, Programming Problems, and Solutions*. Addison-Wesley. ISBN 0-201-70434-X.
- Sutter, Herb (2004). *Exceptional C++ Style*. Addison-Wesley. ISBN 0-201-76042-8.
- Vandevoorde, David; Josuttis, Nicolai M. (2003). *C++ Templates: The complete Guide*. Addison-Wesley. ISBN 0-201-73484-2.

External links

- JTC1/SC22/WG21 (https://www.open-std.org/jtc1/sc22/wg21/) – the ISO/IEC C++ Standard Working Group
- Standard C++ Foundation (https://isocpp.org/) – a non-profit organization that promotes the use and understanding of standard C++. Bjarne Stroustrup is a director of the organization.
- C++ Keywords (https://en.cppreference.com/w/cpp/keyword)
- C++ Expressions (https://en.cppreference.com/w/cpp/language/expressions)
- C++ Operator Precedence (https://en.cppreference.com/w/cpp/language/operator_precedence)