WIKIPEDIA

Microsoft Visual C++

Microsoft Visual C++ (**MSVC**) is a compiler for the C, C++ and C++/CX programming languages by Microsoft. MSVC is proprietary software; it was originally a standalone product but later became a part of Visual Studio and made available in both trialware and freeware forms. It features tools for developing and debugging C++ code, especially code written for the Windows API, DirectX and .NET.

Many applications require redistributable Visual C++ runtime library packages to function correctly. These packages are often installed independently of applications, allowing multiple applications to make use of the package while only having to install it once. These Visual C++ redistributable and runtime packages are mostly installed for standard libraries that many applications use.[3]

Visual C++

Developer(s)	Microsoft
Initial release	February 1993[1]
Stable release	14.29.30133 / July 27, 2021
Written in	C++[2]
Operating system	Windows
Platform	IA-32, x86-64 and ARM
Available in	English, Chinese (Simplified & Traditional), Czech, French, German, Italian, Japanese, Korean, Polish, Portuguese (Brazilian), Russian, Spanish, Turkish
Type	IDE
License	Trialware and freeware
Website	docs.microsoft.com/en-us/cpp/ (https://docs.microsoft.com/en-us/cpp/)

Contents

History

The predecessor to Visual C++ was called *Microsoft C/C++*. There was also a *Microsoft QuickC* 2.5 and a *Microsoft QuickC for Windows* 1.0. The Visual C++ compiler is still known as *Microsoft C/C++* and as of the release of Visual C++ 2015 Update 2, is on version 14.0.23918.0.

16-bit versions

- Microsoft C 1.0, based on Lattice C, was Microsoft's first C product in 1983. It was not K&R C.

- C 2.0 added large model support.
- C 3.0 was the first version developed inside Microsoft.[4] This version intended compatibility with K&R and the later ANSI standard. It was being used inside Microsoft (for Windows and Xenix development) in early 1984. It shipped as a product in 1985.
- C 4.0 added optimizations and CodeView, a source-level debugger.
- C 5.0 added loop optimizations and 'huge memory model' (arrays bigger than 64 KB) support. Microsoft Fortran and the first 32-bit compiler for 80386 were also part of this project.
- C 5.1 released in 1988 allowed compiling programs for OS/2 1.x.
- C 6.0 released in 1989. It added the *Programmer's Workbench* IDE, global flow analysis, a source browser, and a new debugger, and included an optional C++ front end.[5]
- C/C++ 7.0 was released in 1992. Added built-in support for C++ and MFC (Microsoft Foundation Class Library) 1.0.[6]
- Visual C++ 1.0, which included MFC 2.0, was the first version of 'Visual' C++, released in February 1993. It was Cfront 2.1 compliant[7] and available in two editions:[1]
  - Standard: replaced QuickC for Windows.
  - Professional: replaced C/C++ 7.0. Included the ability to build both DOS and Windows applications, an optimizing compiler, a source profiler, and the Windows 3.1 SDK.[7] The Phar Lap 286 DOS Extender Lite was also included.[8]
- Visual C++ 1.5 was released in December 1993, included MFC 2.5, and added OLE 2.0 and ODBC support to MFC.[9] It was the first version of Visual C++ that came only on CD-ROM.
  - Visual C++ 1.51 and 1.52 were available as part of a subscription service.
  - Visual C++ 1.52b is similar to 1.52, but does not include the Control Development Kit.
  - Visual C++ 1.52c was a patched version of 1.5. It is the last, and arguably most popular, development platform for Microsoft Windows 3.x. It is available through Microsoft Developer Network.

Strictly 32-bit versions

- Visual C++ 1.0 (original name: Visual C++ 32-bit Edition) was the first version for 32-bit development for the Intel 386 architecture.[10] Although released when 16-bit version 1.5 was available, it did not include support for OLE2 and ODBC. It was also available in a bundle called Visual C++ 16/32-bit Suite, which included Visual C++ 1.5.[11]
- Visual C++ 2.0, which included MFC 3.0, was the first version to be 32-bit only. In many ways, this version was ahead of its time, since Windows 95, then codenamed "Chicago", was not yet released, and Windows NT had only a small market share. Microsoft included and updated Visual C++ 1.5 as part of the 2.x releases up to 2.1, which included Visual C++ 1.52, and both 16-bit and 32-bit version of the Control Development Kit (CDK) were included. Visual C++ 2.x also supported Win32s development. It is available through Microsoft Developer Network. There was a Visual C++ 2.0 RISC Edition for MIPS and Alpha processors, as well as a cross-platform edition for the Macintosh (68000 instruction set).[12]
  - Visual C++ 2.1 and 2.2 were updates for 2.0 available through subscription.
- Visual C++ 4.0, released on 1995-12-11,[13] introduced the Developer Studio IDE. Its then-novel tiled layout of non-overlapping panels—navigation panel, combination editor/source level debugger panel, and console output panel[14]—continues through the Visual Studio product line (as of 2019). Visual C++ 4.0 included MFC 4.0, was designed for Windows 95 and Windows NT. To allow support of legacy (Windows 3.x/DOS) projects, 4.0 came bundled with the Visual C++ 1.52 installation CD. Updates available through subscription included Visual C++ 4.1, which came with the Microsoft Game SDK (later released separately as the DirectX SDK), and Visual C++ 4.2. Version number 3.0 was skipped to achieve version number parity between Visual C++ 4.0 and MFC 4.0.[15]

- Visual C++ 4.2 did not support Windows 3.x (Win32s) development.[16] This was the final version with a cross-platform edition for the Macintosh available and it differed from the 2.x version in that it also allowed compilation for the PowerPC instruction set.
- Visual C++ 5.0, which included MFC 4.21 and was released 1997-04-28,[13] was a major upgrade from 4.2.[17] Available in four editions: Learning,[18] Professional,[19] Enterprise,[20] and RISC.[21]
- Visual C++ 6.0 (commonly known as VC6), which included MFC 6.0, was released in 1998.[22][23] The release was somewhat controversial since it did not include an expected update to MFC. Visual C++ 6.0 is still quite popular and often used to maintain legacy projects. There are, however, issues with this version under Windows XP, especially under the debugging mode (for example, the values of static variables do not display). The debugging issues can be solved with a patch called the "Visual C++ 6.0 Processor Pack".[24] Version number: 12.00.8804
- Visual C++ .NET 2002 (also known as Visual C++ 7.0), which included MFC 7.0, was released in 2002 with support for link time code generation and debugging runtime checks, .NET 1.0, and Visual C# and Managed C++. The new user interface used many of the hot keys and conventions of Visual Basic, which accounted for some of its unpopularity among C++ developers. Version number: 13.00.9466
- Visual C++ .NET 2003 (also known as Visual C++ 7.1), which included MFC 7.1, was released in 2003 along with .NET 1.1 and was a major upgrade to Visual C++ .NET 2002. It was considered a patch to Visual C++ .NET 2002. Accordingly, the English language upgrade version of Visual Studio .NET 2003 shipped for minimal cost to owners of the English-language version of Visual Studio .NET 2002. This was the last version to support Windows 95 and NT 4.0 as a target. Version number: 13.10.3077
- eMbedded Visual C++[25] in various versions was used to develop for some versions of the Windows CE operating system. Initially it replaced a development environment consisting of tools added onto Visual C++ 6.0. eMbedded Visual C++ was replaced as a separate development environment by Microsoft Visual Studio 2005.

32-bit and 64-bit versions

- Visual C++ 2005 (also known as Visual C++ 8.0), which included MFC 8.0, was released in November 2005. This version supports .NET 2.0 and includes a new version of C++ targeted to the .NET framework (C++/CLI) with the purpose of replacing the previous version (Managed C++). Managed C++ for CLI is still available via compiler options, though. It also introduced OpenMP. With Visual C++ 2005, Microsoft also introduced Team Foundation Server. Visual C++ 8.0 has problems compiling MFC AppWizard projects that were created using Visual Studio 6.0, so maintenance of legacy projects can be continued with the original IDE if rewriting is not feasible. Visual C++ 2005 is the last version able to target Windows 98 and Windows Me.[26][27] SP1 version (14.00.50727.762) is also available in Microsoft Windows SDK Update for Windows Vista.
- Visual C++ 2008 (also known as Visual C++ 9.0) was released in November 2007. This version supports .NET 3.5. Managed C++ for CLI is still available via compiler options. By default, all applications compiled against the Visual C++ 2008 Runtimes (static and dynamic linking) will only work under Windows 2000 and later.[28][29] A feature pack released for VC9, later included in SP1, added support for C++ TR1 library extensions. SP1 version (15.00.30729.01) is also available in Microsoft Windows SDK for Windows 7.
- Some versions of Visual C++ supported Itanium 2.
- Visual C++ 2010 (also known as Visual C++ 10.0) was released on April 12, 2010. It uses a SQL Server Compact database to store information about the source code, including IntelliSense information, for better IntelliSense and code-completion support.[30] However, Visual C++ 2010 does not support Intellisense for C++/CLI.[31] This version adds a C++ parallel computing library called the Parallel Patterns Library, partial support for C++11, significantly improved IntelliSense based on the Edison Design Group front end,[32] and performance

improvements to both the compiler and generated code.[33] This version is built on .NET 4.0, but supports compiling to machine code. The partial C++11 support mainly consists of six compiler features:[34] lambdas, rvalue references, auto, decltype, static_assert, and nullptr. C++11 also supports library features (e.g., moving the TR1 components from std::tr1 namespace directly to std namespace). Variadic templates were also considered, but delayed until some future version due to having a lower priority, which stemmed from the fact that, unlike other costly-to-implement features (lambda, rvalue references), variadic templates would benefit only a minority of library writers rather than the majority of compiler end users.[35] By default, all applications compiled against Visual C++ 2010 Runtimes only work on Windows XP SP2 and later. The RTM version (16.00.30319) is also available in Windows SDK for Windows 7 and .NET Framework 4 (WinSDK v7.1).[36] SP1 version (16.00.40219) is available as part of Visual Studio 2010 Service Pack 1 or through the Microsoft Visual C++ 2010 Service Pack 1 Compiler Update for the Windows SDK 7.1.[37]

- Visual C++ 2012 (also known as Visual C++ 11.0) was released on August 15, 2012. It features improved C++11 support, and support for Windows Runtime development.[38]
- Visual C++ 2013 (also known as Visual C++ 12.0) was released on October 17, 2013. It features further C++11 and C99 support, and introduces a REST SDK.[39]
- Visual C++ 2015 (also known as Visual C++ 14.0) was released on July 20, 2015.[40] It features improved C++11/14/17 support.[41] Without any announcement from Microsoft, Visual Studio 2015 Update 2 started generating telemetry calls in compiled binaries. After some users contacted Microsoft about this problem, Microsoft said they would remove these telemetry calls when compiling with the future Visual Studio 2015 Update 3.[42][43] The function in question was removed from the Visual C++ CRT static libraries in Visual Studio 2015 Update 3.
- Visual C++ 2017 (also known as Visual C++ 14.1) was released on March 7, 2017.
- Visual C++ 2019 (also known as Visual C++ 14.2) was released on April 2, 2019.

Internal version numbering

The predefined macro `_MSC_VER` indicates the major and minor version numbers of the Visual C++ compiler. The macro's value is an integer literal in which the last two digits indicate the minor version number and the preceding digits indicate the major version number.

From Visual Studio 2017, `_MSC_VER` is incremented monotonically at every Visual C++ toolset update (https://blogs.msdn.microsoft.com/vcblog/2016/10/05/visual-c-compiler-version/). Thus, for example, the version of | 14.11 that ships with Visual Studio 2017 version 15.3.0 sets `_MSC_VER` to `1911`. Microsoft recommends using the >= operator to test the value of `_MSC_VER`.

Here are the values of `_MSC_VER` for various versions of Visual C++:

MSC versions

MSC version	_MSC_VER
1.0	100
2.0	200
3.0	300
4.0	400
5.0	500
6.0	600
7.0	700

MSVC++ versions

MSVC++ version	_MSC_VER
1.0	800
2.0	900
4.0	1000 (Developer Studio 4.0)
4.2	1020 (Developer Studio 4.2)
5.0	1100 (Visual Studio 5.0)
6.0	1200 (Visual Studio 6.0)
7.0	1300 (Visual Studio 2002 7.0)
7.1	1310 (Visual Studio 2003 7.1)
8.0	1400 (Visual Studio 2005 8.0)
9.0	1500 (Visual Studio 2008 9.0)
10.0	1600 (Visual Studio 2010 10.0)
11.0	1700 (Visual Studio 2012 11.0)
12.0	1800 (Visual Studio 2013 12.0)
14.0	1900 (Visual Studio 2015 14.0)
14.1	1910 (Visual Studio 2017 15.0)
14.11	1911 (Visual Studio 2017 15.3)
14.12	1912 (Visual Studio 2017 15.5)
14.13	1913 (Visual Studio 2017 version 15.6)
14.14	1914 (Visual Studio 2017 version 15.7)
14.15	1915 (Visual Studio 2017 version 15.8)
14.16	1916 (Visual Studio 2017 version 15.9)
14.2	1920 (Visual Studio 2019 Version 16.0)
14.21	1921 (Visual Studio 2019 Version 16.1)
14.22	1922 (Visual Studio 2019 Version 16.2)
14.23	1923 (Visual Studio 2019 Version 16.3)
14.24	1924 (Visual Studio 2019 Version 16.4)
14.25	1925 (Visual Studio 2019 Version 16.5)
14.26	1926 (Visual Studio 2019 Version 16.6)
14.27	1927 (Visual Studio 2019 Version 16.7)
14.28	1928 (Visual Studio 2019 Version 16.8 + 16.9)
14.29	1929 (Visual Studio 2019 Version 16.10 + 16.11)

These version numbers refer to the major version number of the Visual C++ compilers and libraries, as can be seen from the installation directories. It does not refer to the year in the name of the Visual Studio release. A thorough list is available.[44]

Note that the C++ compiler executable version matches `_MSC_VER` and is different from the version of the Visual C++ product as a whole. For example the cl.exe included in 14.22 (Visual Studio 2019 16.2.5) reports its version as 19.22.27905 if run without arguments.

There is also a `_MSC_FULL_VER` value, defined since 1200, for extra information about the build number.

Compatibility

ABI

The Visual C++ compiler ABI have historically changed between major compiler releases.[45] This is especially the case for STL containers, where container sizes have varied a lot between compiler releases.[46] Microsoft therefore recommends against using C++ interfaces at module boundaries when one wants to enable client code compiled using a different compiler version. Instead of C++, Microsoft recommends using C[47] or COM[48] interfaces, which are designed to have a stable ABI between compiler releases.

All 14.x MSVC releases have a stable ABI,[49] and binaries built with these versions can be mixed in a forwards-compatible manner, noting the following restrictions:

- The toolset version used must be equal to or higher than the highest toolset version used to build any linked binaries.
- The MSVC Redistributable version must be equal to or higher than the toolset version used by any application component.
- Static libraries or object files compiled with /GL (Whole program optimisation) aren't binary compatible between versions and must use the exact same toolset.

C runtime libraries

Visual C++ ships with different versions of C runtime libraries.[50] This means users can compile their code with any of the available libraries. However, this can cause some problems when using different components (DLLs, EXEs) in the same program. A typical example is a program using different libraries. The user should use the same C Run-Time for all the program's components unless the implications are understood. Microsoft recommends using the multithreaded, dynamic link library (/MD or /MDd compiler option) to avoid possible problems.[50]

POSIX

Although Microsoft's CRT implements a large subset of POSIX interfaces, the Visual C++ compiler will emit a warning on *every* use of such functions by default. The rationale is that C and C++ standards require an underscore prefix before implementation-defined interfaces, so the use of these functions are non-standard.[51] However, systems that are actually POSIX-compliant would not accept these underscored names, and it is more portable to just turn off the warning instead.

C

Although the product originated as an IDE for the C programming language, for many years the compiler's support for that language conformed only to the original edition of the C standard, dating from 1989, but not the C99 revision of the standard. There had been no plans to support C99 even in 2011, more than a decade after its publication.[52]

Visual C++ 2013 finally added support for various C99 features in its C mode (including designated initializers, compound literals, and the `_Bool` type),[53] though it was still not complete.[54] Visual C++ 2015 further improved the C99 support, with full support of the C99 Standard Library, except for features that require C99 language features not yet supported by the compiler.[55]

Most of the changes from the C11 revision of the standard are still not supported by Visual C++ 2017.[56] For example, generic selections via the `_Generic` keyword are not supported by the compiler and result in a syntax error.[57]

The preprocessor was overhauled in 2018, with C11 in sight:[58]

> Full C11 conformance is on our roadmap, and updating the preprocessor is just the first step in that process. The C11 `_Generic` feature is not actually part of the preprocessor, so it has not yet been implemented. When implemented I expect the feature to work independently of if the traditional or updated preprocessor logic is used.

`_Generic` support has been committed to MSVC as of February 2020, not clear on when it will ship.[59]

In September 2020, Microsoft announced C11 and C17 standards support in MSVC.[60]

References

1. "Visual C++ adds Windows support" (https://books.google.com/books?id=vjsEAAAAMBAJ). *InfoWorld*. February 22, 1993. p. 17.
2. Lextrait, Vincent (January 2010). "The Programming Languages Beacon, v10.0" (https://archive.today/20120530/http://www.lextrait.com/Vincent/implementations.html). Archived from the original (http://www.lextrait.com/Vincent/implementations.html) on 30 May 2012. Retrieved 14 March 2010.
3. "Do I need these Microsoft Visual C++ redistributables?" (http://ask-leo.com/do_i_need_these_microsoft_visual_c_redistributables.html). Ask Leo!. Retrieved 2012-11-18.
4. Leibson, Steve (1985-02-01). "Software Reviews: Expert team analyzes 21 C compilers" (https://archive.org/details/Computer_Language_Issue_06_1985-02_CL_Publications_US/page/n85/mode/2up?q=microsoft+c+3.0). *Computer Language*. Retrieved 2020-06-05.
5. Ladd, Scott Robert (August 1, 1990). "Optimizing With Microsoft C 6.0" (http://www.drdobbs.com/windows/optimizing-with-microsoft-c-60/184408398).
6. Retrieved from http://support.microsoft.com/kb/196831.
7. "Visual C++ is a strong development tool" (https://books.google.com/books?id=OjsEAAAAMBAJ). *InfoWorld*. June 21, 1993. p. 94.
8. "Rival DOS Extenders debut at show" (https://books.google.com/books?id=fzwEAAAAMBAJ). *InfoWorld*. March 1, 1993. p. 18.
9. "Visual C++ 1.5 integrates OLE, ODBC" (https://books.google.com/books?id=8ToEAAAAMBAJ). *InfoWorld*. November 8, 1993. p. 5.
10. "Microsoft set to prerelease 32-bit Visual C++" (https://books.google.com/books?id=mTsEAAAAMBAJ). *InfoWorld*. July 19, 1993. p. 12.
11. "C++ IDEs evolve" (https://books.google.com/books?id=FTsEAAAAMBAJ). *InfoWorld*. April 4, 1994. p. 79.
12. "Microsoft Visual C++ Strategy" (http://accu.org/index.php/journals/1771).
13. "Obsolete Products" (https://web.archive.org/web/20050814234847/http://support.microsoft.com/gp/lifeobsoleteproducts). Archived from the original (http://support.microsoft.com/gp/lifeobsoleteproducts) on 2005-08-14.
14. Toth, Viktor (1996). "1" (http://doc.sumy.ua/prog/unleash4/vcu01fi.htm). *Visual C++ 4.0 unleashed* (https://books.google.com/books?id=93o_AQAAIAAJ&q=visual+c%2B%2B+4.0+unleashed). Indianapolis: SAMS Publishing. ISBN 9780672308741. Retrieved 26 July 2013.

15. "History of Visual Studio (Part 3)" (http://blogs.msdn.com/ricom/archive/2009/10/07/my-history-of-visual-studio-part-3.aspx).
16. "Major Changes from Visual C++ 4.0 to 4.2" (https://web.archive.org/web/20100228175137/http://msdn.microsoft.com/en-us/library/aa697418(VS.71).aspx). Archived from the original (http://msdn.microsoft.com/en-us/library/aa697418(VS.71).aspx) on 2010-02-28. Retrieved 2018-04-18.
17. "Major Changes from Visual C++ 4.2 to 5.0" (http://msdn.microsoft.com/en-us/library/aa697419(VS.71).aspx).
18. "Microsoft Visual C++ 5.0 Learning Edition" (https://web.archive.org/web/19990427114135/http://www.microsoft.com/products/prodref/199_ov.htm). Archived from the original (http://www.microsoft.com/products/prodref/199_ov.htm) on April 27, 1999.
19. "Microsoft Visual C++ 5.0 Professional Edition" (https://web.archive.org/web/19990427101205/http://www.microsoft.com/products/prodref/197_ov.htm). Archived from the original (http://www.microsoft.com/products/prodref/197_ov.htm) on April 27, 1999.
20. "Microsoft Visual C++ 5.0 Enterprise Edition" (https://web.archive.org/web/19990417134138/http://www.microsoft.com/products/prodref/198_ov.htm). Archived from the original (http://www.microsoft.com/products/prodref/198_ov.htm) on April 17, 1999.
21. "Microsoft Visual C++ 5.0 RISC Edition" (https://web.archive.org/web/19990429121236/http://www.microsoft.com/products/prodref/501_ov.htm). Archived from the original (http://www.microsoft.com/products/prodref/501_ov.htm) on April 29, 1999.
22. Shields, Nathan P. (June 8, 2018). "Criminal Complaint" (https://www.justice.gov/opa/press-release/file/1092091/download). United States Department of Justice. p. 128. "This alone is not a dispositive link, as Visual C++ 6.0, released in 1998, still has proponents mostly because it does not require the installation of Microsoft's .NET framework in order to run, as later versions of Visual C++ do."
23. "Major Changes from Visual C++ 5.0 to 6.0" (https://web.archive.org/web/20080914202003/http://msdn.microsoft.com/en-us/library/aa729389(VS.71).aspx). Archived from the original (http://msdn.microsoft.com/en-us/library/aa729389(VS.71).aspx) on September 14, 2008.
24. This page stresses that *Users must also be running Windows 98, Windows NT 4.0, or Windows 2000.* Retrieved from http://msdn2.microsoft.com/en-us/vstudio/aa718349.aspx.
25. Douglas Boling :*Programming Microsoft Windows CE .NET, Third Edition* Microsoft Press; 3rd edition (June 25, 2003) Paperback: 1264 pages ISBN 978-0735618848 - Companion CD with Microsoft eMbedded Visual C++ 4.0 Service Pack 2 (http://examples.oreilly.de/english_examples/9780735618848/cd_contents/Readme.txt) Archived (https://archive.today/20130211131045/http://examples.oreilly.de/english_examples/9780735618848/cd_contents/Readme.txt) 2013-02-11 at archive.today
26. How to: Modify WINVER and _WIN32_WINNT (http://msdn.microsoft.com/en-us/library/6sehtctf(v=VS.90).aspx)
27. Breaking Changes (http://msdn.microsoft.com/en-us/library/bb531344(v=VS.90).aspx)
28. Windows Platforms (CRT) (http://msdn.microsoft.com/en-us/library/ws0swas0(VS.100).aspx)
29. "Visual C++ 2008 Breaking Changes" (http://msdn.microsoft.com/en-us/library/bb531344.aspx?ppud=4).
30. Visual C++ Team Blog. "IntelliSense, part 2: The Future" (http://blogs.msdn.com/vcblog/archive/2008/02/29/intellisense-part-2-the-future.aspx). Retrieved March 12, 2008.
31. "Why IntelliSense is not supported for C++/CLI in Visual Studio 2010" (http://blogs.msdn.com/b/vcblog/archive/2011/03/03/10136696.aspx). Retrieved March 13, 2011.
32. Visual C++ Team Blog (27 May 2009). "Rebuilding Intellisense" (http://blogs.msdn.com/b/vcblog/archive/2009/05/27/rebuilding-intellisense.aspx).
33. Visual C++ Team Blog. "Visual C++ Code Generation in Visual Studio 2010" (http://blogs.msdn.com/vcblog/archive/2009/11/02/visual-c-code-generation-in-visual-studio-2010.aspx).
34. "C++0x Core Language Features In VC10: The Table" (http://blogs.msdn.com/vcblog/archive/2010/04/06/c-0x-core-language-features-in-vc10-the-table.aspx).

35. "Stephan T. Lavavej: Everything you ever wanted to know about nullptr" (http://channel9.msdn.com/shows/Going+Deep/Stephan-T-Lavavej-Everything-you-ever-wanted-to-know-about-nullptr/).
36. Microsoft Windows SDK Blog. "Released: Windows SDK for Windows 7 and .NET Framework 4" (http://blogs.msdn.com/b/windowssdk/archive/2010/05/25/released-windows-sdk-for-windows-7-and-net-framework-4.aspx).
37. FIX: Visual C++ compilers are removed when you upgrade Visual Studio 2010 Professional or Visual Studio 2010 Express to Visual Studio 2010 SP1 if Windows SDK v7.1 is installed (http://support.microsoft.com/kb/2519277/en-us)
38. "What's New for Visual C++ in Visual Studio 2012" (https://msdn.microsoft.com/en-us/library/vstudio/hh409293(v=vs.110).aspx). *Microsoft Developer Network*. Microsoft. Retrieved September 20, 2015.
39. "What's New for Visual C++ in Visual Studio 2013" (https://msdn.microsoft.com/en-us/library/vstudio/hh409293(v=vs.120).aspx). *Microsoft Developer Network*. Miicrosoft. Retrieved September 20, 2015.
40. Eric Battalio (July 20, 2015). "Visual Studio 2015 RTM Now Available" (http://blogs.msdn.com/b/vcblog/archive/2015/07/20/visual-studio-2015-rtm-now-available.aspx). *Visual C++ Team Blog*. Microsoft.
41. Stephan T. Lavavej (June 19, 2015). "C++11/14/17 Features In VS 2015 RTM" (http://blogs.msdn.com/b/vcblog/archive/2015/06/19/c-11-14-17-features-in-vs-2015-rtm.aspx). *Visual C++ Team Blog*. Microsoft.
42. Reviewing Microsoft's Automatic Insertion of Telemetry into C++ Binaries (https://www.infoq.com/news/2016/06/visual-cpp-telemetry)
43. "Visual Studio adding telemetry function calls to binary? • /r/cpp" (https://www.reddit.com/r/cpp/comments/4ibauu/visual_studio_adding_telemetry_function_calls_to/d30dmvu). *reddit*. 7 May 2016. Retrieved 2016-08-17.
44. "Pre-defined Compiler Macros / Wiki / Compilers" (https://sourceforge.net/p/predef/wiki/Compilers/#microsoft-visual-c). *sourceforge.net*. Retrieved 2016-02-11.
45. Microsoft MSDN: Breaking Changes in Visual C++ (http://msdn.microsoft.com/en-us/library/bb531344.aspx)
46. Microsoft MSDN: Containers (Modern C++) (http://msdn.microsoft.com/en-us/library/vstudio/hh438470.aspx)
47. Microsoft MSDN: Portability At ABI Boundaries (Modern C++) (http://msdn.microsoft.com/en-us/library/vstudio/hh438475.aspx)
48. Microsoft forum: Binary compatibility across Visual C++ versions (http://social.msdn.microsoft.com/Forums/en/vcgeneral/thread/86eda6a7-4d90-4e19-a9d4-6cbe22b661f4) Archived (https://archive.today/20130216110824/http://social.msdn.microsoft.com/Forums/en/vcgeneral/thread/86eda6a7-4d90-4e19-a9d4-6cbe22b661f4) 2013-02-16 at archive.today
49. docs.microsoft.com: C++ binary compatibility between Visual Studio 2015, 2017, and 2019 (https://docs.microsoft.com/en-us/cpp/porting/binary-compat-2015-2017?)
50. C Run-Time Libraries (http://msdn.microsoft.com/en-us/library/abx4dbyh.aspx)
51. "Compatibility" (https://docs.microsoft.com/en-us/cpp/c-runtime-library/compatibility). *Microsoft: CRT library features*.
52. "C99 support" (https://web.archive.org/web/20160304185822/http://connect.microsoft.com/VisualStudio/feedback/details/653336/c99-support). *Microsoft Connect*. Archived from the original (https://connect.microsoft.com/VisualStudio/feedback/details/653336/c99-support) on 2016-03-04. Retrieved 2015-09-21.
53. "What's New for Visual C++ in Visual Studio 2013" (https://msdn.microsoft.com/en-us/library/hh409293(v=vs.120).aspx). *Microsoft Developer Network*. Microsoft.
54. Pat Brenner (July 19, 2013). "C99 library support in Visual Studio 2013" (http://blogs.msdn.com/b/vcblog/archive/2013/07/19/c99-library-support-in-visual-studio-2013.aspx). *Visual C++ Team Blog*. Microsoft.

55. "What's New for Visual C++ in Visual Studio 2015" (https://msdn.microsoft.com/en-us/library/hh409293(v=vs.140).aspx). *Microsoft Developer Network*. Microsoft.
56. "Walkthrough: Compile a C program on the command line" (https://docs.microsoft.com/en-us/cpp/build/walkthrough-compile-a-c-program-on-the-command-line). *Visual C++ Documentation*. Microsoft.
57. "MSVC 2017 does not support _Generic (type generic macros) in C" (https://developercommunity.visualstudio.com/content/problem/250665/msvc-2017-does-not-support-generic-type-generic-ma.html).
58. Luvsanbat, Bat-Ulzii (July 6, 2018). "MSVC Preprocessor Progress towards Conformance" (https://web.archive.org/web/20190108124214/https://blogs.msdn.microsoft.com/vcblog/2018/07/06/msvc-preprocessor-progress-towards-conformance/). *Microsoft Developer Network*. Archived from the original (https://blogs.msdn.microsoft.com/vcblog/2018/07/06/msvc-preprocessor-progress-towards-conformance/) on 8 Jan 2019.
59. {{cite weblurl=https://www.reddit.com/r/cpp/comments/hjn4uv/c20_features_and_fixes_in_vs_2019_16context=3 ltitle=C++20 Features and Fixes ldate=July 3, 2020 website=reddit
60. "C11 and C17 Standard Support Arriving in MSVC" (https://devblogs.microsoft.com/cppblog/c11-and-c17-standard-support-arriving-in-msvc/). September 14, 2020.

Further reading

- Johnson, Brian (8 August 2004). "Building Win32 Applications Using Visual C++ 2005 Express" (https://docs.microsoft.com/en-us/archive/blogs/brianjo/building-win32-applications-using-visual-c-2005-express). *Brian Johnson's Startup Developer Blog*. Microsoft – via Microsoft Docs Archive.
- Springfield, Jim (25 September 2015). "Rejuvenating the Microsoft C/C++ Compiler" (https://devblogs.microsoft.com/cppblog/rejuvenating-the-microsoft-cc-compiler/). *C++ Team Blog*. Microsoft.

External links

- Official website (https://docs.microsoft.com/en-us/cpp/)
- C++ Tools and Features in Visual Studio Editions (https://docs.microsoft.com/en-us/cpp/overview/visual-cpp-tools-and-features-in-visual-studio-editions?redirectedfrom=MSDN&view=vs-2019)
- Microsoft C++ Build Tools (https://visualstudio.microsoft.com/visual-cpp-build-tools/)
- C9::GoingNative (http://channel9.msdn.com/Shows/C9-GoingNative) on Microsoft Channel 9