

**“AZƏRBAYCAN HAVA YOLLARI” CJSC NATIONAL AVIATION ACADEMY”**

**Individual Work №: 3**

**Topic: C / C ++ compilers.**

**Subject: OS2**

**Teacher: Mamed Shahmaliyev**

**Group: 1459i Student: Yusifov Rəvan**

It is no secret that I love C++. One of the things I love about C++ is the fact it’s such an ‘open’ programming language that allows anyone to implement their own compiler. So naturally, there are a lot of C++ compilers out there. In the same way, there are many [C++ IDEs](https://www.incredibuild.com/blog/best-c-ides), but I discuss that in a different blog post.

Normally, we compile and run C++ code for a platform (or a host) which is known as the hosted environment. When a C++ program executes without the help of an Operating System, it is running under a freestanding environment. Lack of an Operating System makes a freestanding environment very restricted. There are many requirements to run under such an environment which you can check out [here](https://en.cppreference.com/w/cpp/freestanding). One day, I shall write a blog on development under freestanding environments for C++, but this blog is about the compilers I consider to be top-notch in their game. Let us enumerate.

**Top C++ compilers for hosted environments**

**1. Microsoft Visual C++ compiler**

This is the C and C++ compiler that Microsoft bundles with Visual Studio. The current compiler version, bundled with Visual Studio 2019 version 16.10, is 19.28.29914, which supports both the C++17 core language features as well as C++17 library features completely and the C++20 features [partially](https://docs.microsoft.com/en-us/cpp/overview/visual-cpp-language-conformance?view=msvc-160). It is expected that Visual Studio 2022 – which is currently in the preview version – will include a Visual C++ compiler that will fully support the C++20 features. Although the Visual C++ compiler is primarily used for Windows development, using the windows subsystem for Linux (WSL) integration, it can be used [to develop native Linux applications too](https://devblogs.microsoft.com/cppblog/c-with-visual-studio-2019-and-windows-subsystem-for-linux-wsl). Checkout this nice [video](https://www.youtube.com/watch?v=ijmZKRIfoOI) to learn more on this topic.

**2. The GNU compiler collection**

The GNU compiler collection, GCC, is one of the most famous open-source tools in existence. It is a tool that can be used to compile multiple languages and not just C or C++. The current version of GCC, GCC 11, has full support for C++17 core language features as well as C++17 library features. It also has experimental support for almost all of the C++20 language and library features, except for some minor features in Modules. Notably, GCC 11 also includes some features of the draft C++23 standard which is the next revision of the C++ standard after C++20.  
\*[Learn more about what is GCC](https://www.incredibuild.com/integrations/gcc)

**3. Clang/LLVM**

In one of my recent blog posts, I had compared [GCC vs Clang](https://www.incredibuild.com/blog/gcc-vs-clang-battle-of-the-behemoths). I had detailed the architecture of this compiler and described how the LLVM backend makes it easy to add new optimizations to the compiler. The current version of Clang/LLVM, version 12.0, currently supports C++17 fully and has experimental support for C++20.  As you proceed through this blog post you will understand why a lot of other C++ compilers want to base their code on this open-source platform.

**4. Intel C++ compiler**

I have used Intel C++ compiler (Intel® oneAPI DPC++/C++ Compiler to be precise) for computationally intensive applications and I have found its performance to be top-notch. Unlike Visual Studio which includes frameworks like MFC (Microsoft foundation classes) for desktop application development and WebView2 support for Web-based applications, Intel’s compiler includes support for Threading Building Blocks (currently open-sourced as oneAPI) and Data Parallel C++ (DPC++) clearly showing the difference in focus. Computationally intensive applications with data parallelism (with parallel STL), Field-programmable gate array (FPGA) support as well as support for Graphics Processing Unit (GPU) is where Intel Compiler shines. The latest version of the Intel C++ compiler supports the C++17 standard.

**5. IBM XLC++**

IBM XLC++ compiler is offered for platforms like z/OS, Linux on Power, AIX, and IBM I (with PASE). This compiler offers advanced optimization technologies thereby generating optimized code for developing complex C++ programs. Recently IBM has contributed code to Clang/LLVM project for Power, AIX, and IBM Z platforms. Last year (2020), IBM announced their [intention to adopt](https://community.ibm.com/community/user/ibmz-and-linuxone/blogs/blog-entry1/2020/02/23/ibm-cc-and-fortran-compilers-to-adopt-llvm-open-source-infrastructure) Clang/LLVM framework for its IBM XLC++ compiler toolchain. This should enable the IBM XLC++ compiler to support the latest C++ standards easily.

**Top C++ compilers for freestanding environments**

**1. Keil C++ compiler**

[µVision](https://www2.keil.com/mdk5/uvision/) is a windows-based software development platform from ARM to develop embedded applications in many different platforms (for a partial list of platforms that Keil µVision supports, see the below image). The ARM [compiler version 6](https://www2.keil.com/mdk5/compiler/6/) is based on the Clang/LLVM compiler toolchain with the full support of the C++14 language standard. This compiler fully supports functional safety standards like IEC 61508 and ISO 26262 for developing functional safety applications. This compiler has direct support for security-critical applications that employ Arm® TrustZone® for isolating security-critical components in a system. Typical application development on Keil µVision C++ compiler is on eMetering, Lighting, Industrial Networking, Alarm Systems, and Motor Control. For hobbyist programmers, Keil µVision is available free of cost with some limitations on the program size.

**2. Texas Instruments code generation tools for C/C++**

The original TI Arm C/C++ Compiler Tools (ARM-CGT-XX) were standalone C/C++ compilers that are currently in the maintenance phase. The newer version for programming Arm Cortex-M and Arm Cortex-R devices is TI Arm Clang Compiler Tools (ARM-CGT-CLANG-X). This is derived from open source Clang/LLVM compiler toolchain. The ARM Optimizing C/C++ Compiler v20 from Texas Instruments supports the C++14 language standard.

**3. MPLAB XC++ Compiler**

PIC and AVR microcontrollers are traditionally programmed using C, but the newer versions of hardware support 32 bits (from the traditional 8 bit) and can be compiled using C++. The MPLAB XC++ Compiler from Microchip is based on GCC. The compiler is distributed with the C++03 standard library and supports the C++03 standard. After the acquisition of Atmel, Microchip became the leader in embedded control solutions. Its MPLAB X development environment runs on Windows, Linux, and OS X environments and comes bundled with the MPLAB XC++ Compiler.

**Conclusion**

C++ language is rapidly evolving. As many of the compiler teams are realizing lately, it is not easy to keep up with the standard. Basing the compiler on a well established and open source framework like Clang/LLVM is a great way to quickly support the newest features of C++. In this blog post I have listed the mainstream compilers in both hosted and freestanding environments. I am aware of compilers  that don’t exactly fall under freestanding or hosted C++ environments, for example, EDG eccp is a fully-featured C++ frontend that is used by other compilers, is not included in this list. A [transpiler](https://en.wikipedia.org/wiki/Source-to-source_compiler" \t "_blank) like [Emscripten](https://emscripten.org/index.html" \t "_blank) that converts C++ code to Javascript is also not featured in this list. A cross-compiler like MinGW also does not feature in this list. Note: You can check out the various compilers’ support for C++ standards using this [link](https://en.cppreference.com/w/cpp/compiler_support).