# Introduction of Modern C++ Basics

## Early History of C++

## C++ History Review

* C with Classes, created by Bjarne Stroustrup in 1980s
* C front compiler, which was similar to a traditional preprocessor, converted C++ to C and use C compiler for assembly
* Core language:
* Object Oriented: class, derived class, abstract classes, virtual function
* Generic programming: template
* Operator overloading, references, name spaces, exception handling
* Standard library: Original STL implementation by Stepanov and Lee. 1994
* Containers, algorithms, iterators and functions6

## C++ 98

* The first international standard for C++ ISO
* Core language: cast operator, template instantiation, member templates
* Standard library: STL, auto\_ptr, I/O stream (std::cout, std::cin)

## C++ 03

* Minor revision of the C++98 standard, primarily bug fix release to address core language as well as library defects.

## C++ 11

* Modern C++ standard, bringing many major extensions and improvements
* Core language: auto, decltype, move constructors and move assignment operator, range-for, uniform initialization, lambda functions and expressions
* Standard library:
* Containers: <array>, <unordered\_map>, <unordered\_set>,
* Multi-threading: <thread>, <mutex>, <future>, <conditional\_variable>, <atomic>
* Time: <chrono>
* Functional: <functional>
* Smart pointers: <memory>, std::unique\_ptr, std::shared\_ptr cial Engineering (QF633) 5 / 50Evolution of C++

## C++ 14

* Minor revision after major C++11, mainly for minor improvements and defect fixes
* Core language: variable templates, generic lambdas, lambda init-capture
* Standard library: std::make\_unique

## C++ 17

* Major revision after major C++11
* Core language: constexpr if, if init-statement, structured bindings
* Standard library: Type erasure: std::any, std::variant
* Containers: std::string\_view +

## C++ 20

* Support from compilers are not yet completed
* The Big Four
* Concepts: write requirements for your templates to get improved error message instead of pages of cryptic error messages
* Range Library
* Co-routines: no C++20 concreate co-routines but only a framework for writing own co-routines Modules

## Object-Oriented

C++ is an [object-oriented programming language](https://www.simplilearn.com/tutorials/cpp-tutorial/oops-concepts-in-cpp) which means that the main focus is on objects and manipulations around these objects. This makes it much easier to manipulate code, unlike procedural or structured programming which requires a series of computational steps to be carried out.

## Speed

When speed is a critical metric, C++ is still the most preferred choice. The compilation and execution time of a[C++ program](https://www.simplilearn.com/tutorials/cpp-tutorial/cpp-basics) is much faster than most general-purpose[programming languages.](https://www.simplilearn.com/best-programming-languages-start-learning-today-article)

## C++ Advantages

Compiled

Unlike other programming languages where no compilation is required, every C++ code has to be first compiled to a low-level language and then executed.

Rich Library Support

The C++ Standard Template Library (STL) has many functions available to help write code quickly. For example, there are STLs for various containers like hash tables, maps, sets, etc.

## Pointer Support

C++ also supports pointers which are often not available in other programming languages.

## Closer to Hardware

C++ is closer to hardware than most general-purpose programming languages. This makes it very useful in those areas where hardware and software are closely coupled together, and low-level support is needed at the software level.

Almost all the programs and systems that you use or a part of their codebase is written [either in C or C++](https://www.simplilearn.com/tutorials/cpp-tutorial/difference-between-c-and-cpp). Be it Windows, a web browser, your favorite game, or video editing software, the uses of C++ can be found in almost all applications today. Here are some interesting areas where C++ is popularly used.

## Usage of C++ in Real World Application

## Operating Systems

C++ is a fast and strongly typed programming language which makes it an ideal choice for developing operating systems. Mac OS X has large amounts written in C++. Most of the software from Microsoft like Windows, Microsoft Office, IDE Visual Studio, and Internet Explorer are also written in C++.

## Games

Since C++ is closer to hardware, game development companies use it as their primary choice to develop gaming systems. It can easily manipulate resources and can override the complexities of 3D games and multiplayer networking.

## GUI Based Applications

C++ is also used to develop [GUI-based](https://www.simplilearn.com/tutorials/cpp-tutorial/cpp-gui) and desktop applications. Most of the applications from Adobe such as Photoshop, Illustrator, etc. are developed using C++.

## Computational intensive or speed matters Application

Mathematics computation, physics simulation, high frequency trading system, quant library etc.

|  |  |  |
| --- | --- | --- |
| **Job role** | **Salary (Average)** | **Top companies hiring** |
| C++ Developer | ~$100,000 (USD) | BOSCH Group, Capgemini, Amazon, Microsoft, Accenture, IBM, Meta, Adobe, Apple, Mozilla |
| Quant / Quant developer | ~$180,000 (USD) | Banks, hedge funds, brokers and trading companies etc |
| Full Stack Developer | ~$180,000 (USD) | Meta, Netflix, Airbnb, Uber, Infosys, Wipro, Zomato, Swiggy, Ola, Paytm, Amazon, Microsoft |

## Job prospect

## VS Code by Microsoft

## IDE - integrated development environment

* Code editing.
* Redefined.
* Free. Built on open source.
* Runs everywhere

More details to setup vs code on mac can be found in below link:

<https://code.visualstudio.com/docs/setup/mac>

## Microsoft Visual Studio for Windows

Microsoft Visual Studio is a powerful tool for C++, including text editing, compiling and linking, debug, source code repository etc.

* One stop solution and powerful.
* Steep learning curve.
* Recommend “community” version for study purpose

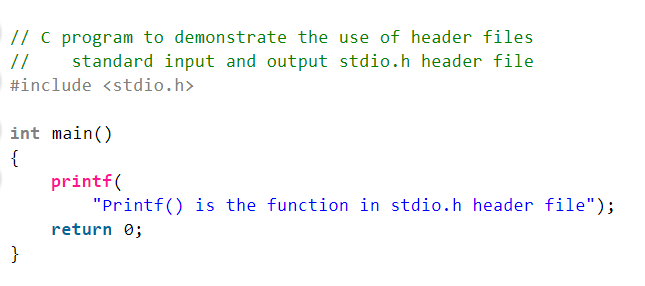
More details to setup visual studio can be found in below link:

<https://visualstudio.microsoft.com/downloads/>

## Structure of C++ Program: Components

The structure of a C++ program is made up of multiple source code files that cater to different components such as main function, member functions, class definition, headers/ standard headers, comments, variables, data types, namespaces, input/ output statements, etc.

Let’s see one example: suppose we have a file named **progl.cpp** as below



### Header file

In C language, header files contain a set of predefined standard library functions. The .h is the extension of the header files in C and we request to use a header file in our program by including it with the C pre-processing directive “#include”.

C Header files offer the features like library functions, data types, macros, etc by importing them into the program with the help of a **pre-processor directive** “#include”. Syntax of Header Files in C: We can include header files in C by using one of the given two syntax whether it is a pre-defined or user-defined header file.

**#include <filename.h> // for files in system/default director**

**#include "filename.h" // for files in same directory as source file**

### Cpp file

Files with CPP file extension are source code files for applications written in C++ programming language. A single C++ project may contain more than one CPP files as application source code.

### Namespace Declarations

Namespace declarations are used in defining named scopes which help in organizing and grouping related code. It’s a way to group a set of identifiers under a common name, to avoid naming conflicts and improve readability. They are part of the actual C++ code and are processed by the C++ compiler during the compilation phase. Different users can create separate namespaces thus facilitating the use of similar names for entities. This further helps in avoiding possible compile-time errors which might arise due to identical-name conflicts.

### Functions in C++

Functions are a group of code statements also referred to as blocks of code/ source code. They are used to group a set of instructions that perform a specific task. Functions can be called from other parts of the program, making it easier to reuse code, implement modularity and improve the readability of the program. A coder can either use any of the many built-in functions already available or create specific user-defined functions, as they deem fit.

### Main Function

The main function is the entry point or the startup function for every C++ program, i.e. this section defines the main() function of the program. This is where the program starts executing, hence this section is a mandatory part of the structure of the C++ program.

Besides the function declaration and program execution, the return type of the main function is also mentioned in this section. All in all, the main function coordinates the flow of the entire program by calling other functions, handles user input and output, and performs various other tasks.

### Comments

Comments are used to provide explanations and documentation for the code only, that is, they are not executable statements. It’s considered good practice to always write comments for your code to improve the readability of the program. There are two types of comment styles supported in C++, namely- single-line comments and multiline comments. The symbol // is used for single-line comments (or single comments) while /\* and \*/ are used for writing multiline comments.

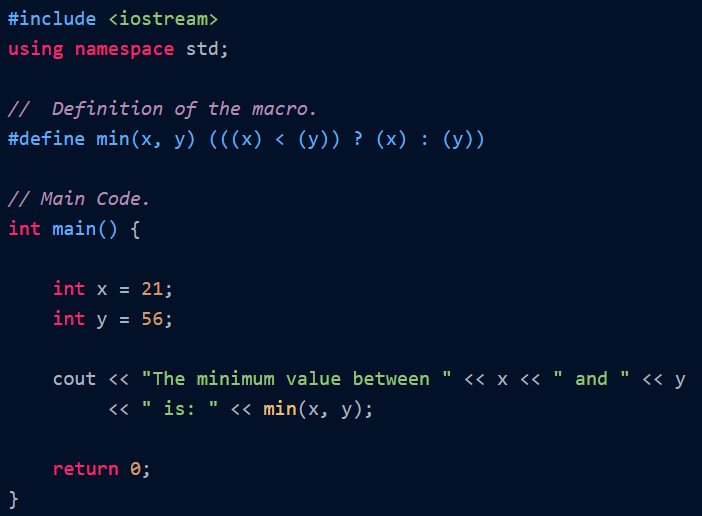


### C++ Macros

A C++ macro is defined as a section of code that that particular macro value can replace. We can define the macro by using a #define directive. When the compiler goes to the macros while program compilation, the macro's name is replaced by the definition of the macro. The termination of the C++ Macros does not need a semi-colon (;).

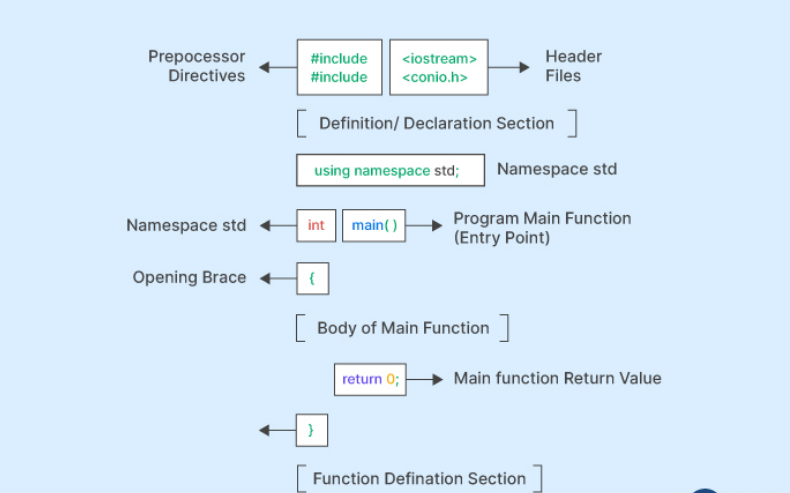






More details can be found in the link below:

<https://www.scaler.com/topics/cpp-macro/>



# Step-by-Step Guide to Compile and Execute C++ Program

Compiling and executing a C++ program involves several steps, each of which is crucial to ensure the smooth and successful execution of the program. They are:

### Writing the code

The first step in the execution of a C++ program is to write the code/ lines of code for the specified purpose. This involves utilizing all the components we learned above as required in a logical manner.

#### Saving the code

Once the code is written, we have to save the code in a file with a .cpp extension. This is the extension for source code files in C++. This file of source code contains all the code that will be compiled and then executed in the steps ahead.

### Preprocessing

Before the code is compiled, it goes through a preprocessor. At this stage, the preprocessor checks the code for any preprocessor directives such as include statements, #define statements, etc., and replaces them with the appropriate code. This ensures that all the necessary libraries, standard libraries, and header files are included in the code.

### Compiling

During the compilation process, the C++ compiler reads the code within the curly braces and checks for any syntax errors. It then compiles the code into object code (i.e. binary code or machine-readable instructions).

### Linking

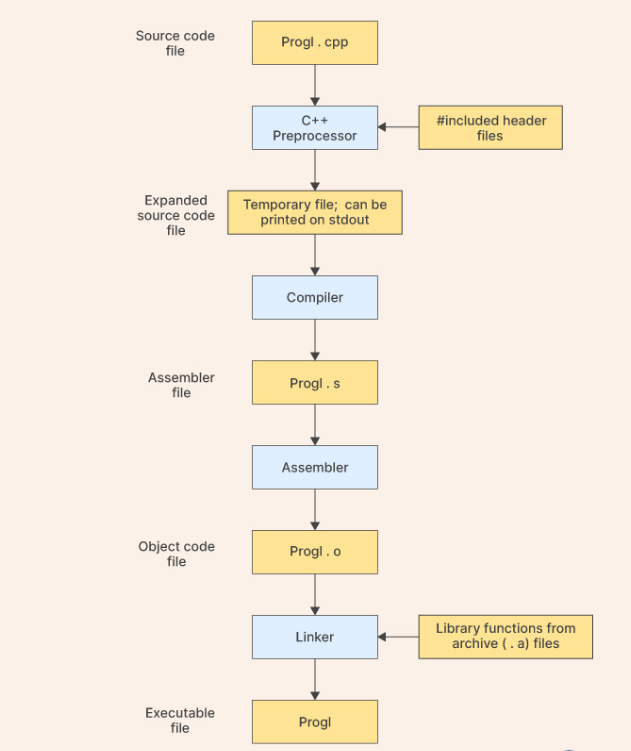
After the compilation of the code, it needs to be linked to the necessary libraries and header files that we have included in the program. This ensures that the program has access to all the functions and variables defined in the program. The linker also checks for any unresolved symbols in the object code and links the code to the appropriate libraries.

### Generating Executable

After the code is successfully compiled and linked, it is ready to be executed. This file is called an executable file. This file contains all the machine-readable instructions that will be executed by the computer.

### Execution

The final step is to execute the program. The executable file is loaded into memory, and the program is executed.



## Recap Preprocessing

Preprocessor directives are invoked to perform various pre-processing tasks, such as importing header files, declaring namespaces, defining constants, and other such operations that need to be done before the program starts executing.

* Header files usually contain function prototypes, class definitions, macro definitions, and other declarations that are necessary for the program to use a particular library or API.
* They are also used to define Macros and constants. Macros are defined using the #define directive and can take arguments that are replaced by their values when macros are expanded.

## Recap Compiling

Compilation The translation of C++ instructions into machine readable files is called compilation. This is done by a program called compiler.

###### The binary files produced by the compiler are object files (.obj in Windows and .o in Linux/Unix).

###### Syntax errors in C++ code are detected by the compiler — compilation error.

###### Compilation errors are pleasant (!) Easy to correct. The compiler messages are usually helpful and link you directly to the place of error.

## Compiler Recommended

###### VS Code: G++.exe build and debug active file

###### <https://code.visualstudio.com/docs/cpp/config-mingw>

###### MS VS: Embedded compiler

###### <https://learn.microsoft.com/en-us/cpp/get-started/?view=msvc-170>

# Introduction on CMake tools

## What is CMake

In software development, CMake is cross-platform free and open-source software for build automation, testing, packaging and installation of software by using a compiler-independent method. CMake is not a build system itself; it generates another system's build files. It supports directory hierarchies and applications that depend on multiple libraries. It can invoke native build environments such as Make, Qt Creator, Ninja, Android Studio, Apple's Xcode, and Microsoft Visual Studio. It has minimal dependencies, requiring only a C++ compiler on its own build system.

## Key Features

## **Single Source build on multiple platform**

CMake allows developers to describe how to build simple and very complicated software systems with a single set of input files. This can be used to build the software on multiple platforms, from Android to iOS to High-Performance Computing systems.

## **Accurate dependencies and minimal rebuilds**

CMake has accurate dependencies and allows for minimal rebuilds of only the code that has changed. It also supports Fortran and C++ modules.

## **Out-of-source builds**

CMake allows for out-of-source builds and gives the ability to have multiple build trees per source tree.

...

More details in <https://cmake.org/>features

## Example of cmakelist.txt

Suppose we have the code below in file hello\_world.cpp

#include <iostream>int main(int argc, char\*\* argv){

std::cout << "Hello World" << std::endl;

**return** 0;

}

In same project folder, we shall have below in CmakeList.txt

**project**(HelloWorld)

**cmake\_minimum\_required**(**VERSION** 3.0)

**add\_executable**(hello\_world main.cpp)

The structure is simple:

* Name of the project (basically, what you want).
* Minimum version of CMake to be used the CMakeLists.txt.
* Create an executable hello\_world which uses source file make.cpp

First, go in your project directory and create a build directory :

$ cd my/project

$ mkdir build

Then, the build directory is created:

$ ls

build CMakeLists.txt main.cpp

Let's go in the build directory and call CMake:

$ cd build

$ cmake ..

Then call **make** :

|  |  |
| --- | --- |
| 1  2  3  4  5 | $ makeScanning dependencies of target hello\_world[ 50%] Building CXX object CMakeFiles/hello\_world.dir/main.cpp.o[100%] Linking CXX executable hello\_world[100%] Built target hello\_world |

Finally, you can call you program:

$ ./hello\_world

Hello World

## First C++ Program – ‘Hello World’

## Type of program in Windows

###### Console application

A console application or command-line program is a computer program (applications or utilities) designed to be used via a text-only user interface, such as a text terminal, the command-line interface of some operating systems (Unix, DOS etc.) or the text-based interface included with most graphical user interface (GUI) operating systems, such as the Windows Console in Microsoft Windows, the Terminal in macOS, and xterm in Unix.

###### Dynamic Linked Library (dll)

In Windows, a dynamic-link library (DLL) is a kind of executable file that acts as a shared library of functions and resources. Dynamic linking is an operating system capability. It enables an executable to call functions or use resources stored in a separate file

## Create your first console application

* Start VS code or Visual studio (create a project with application type as console application)
* Create a new file named helloworld.cpp
* Type below syntax
* Compile and run

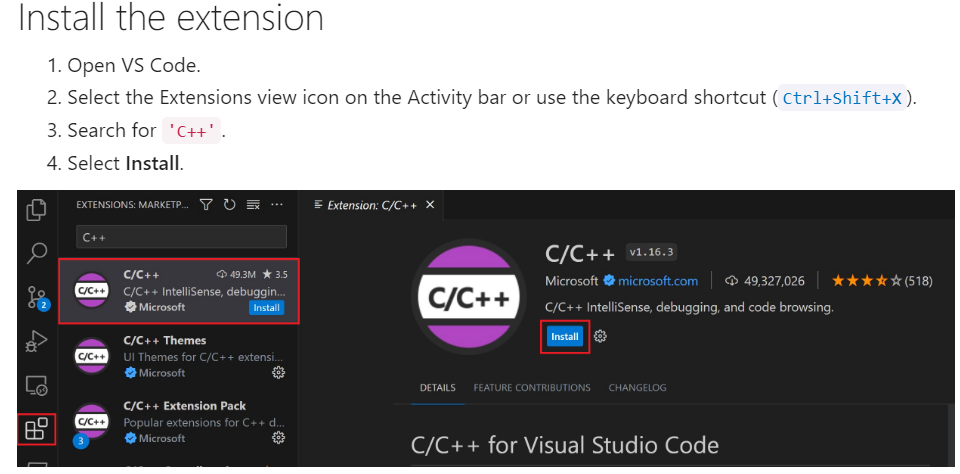
###### Steps

1. Setup the IDE on your computer, install the needed application and packages.
2. Following details in appendix and try to create “hello world” program.

# Appendix

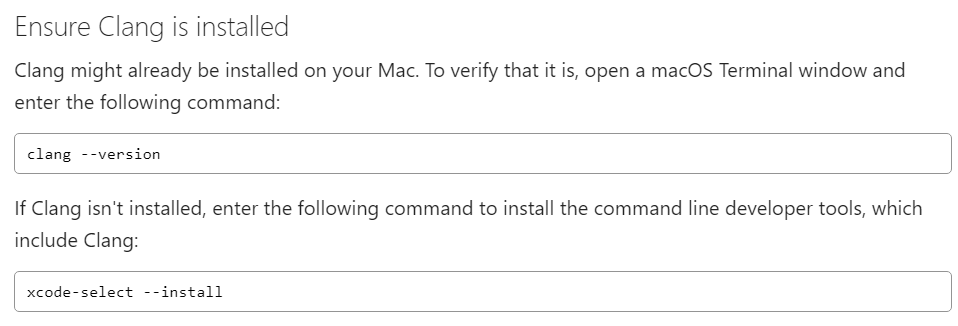
## Step by step for Setup VS Code on mac or windows

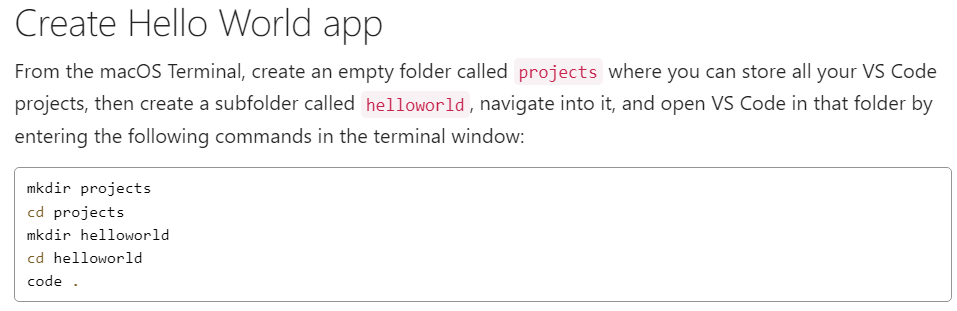
<https://code.visualstudio.com/docs/languages/cpp#_install-the-extension>

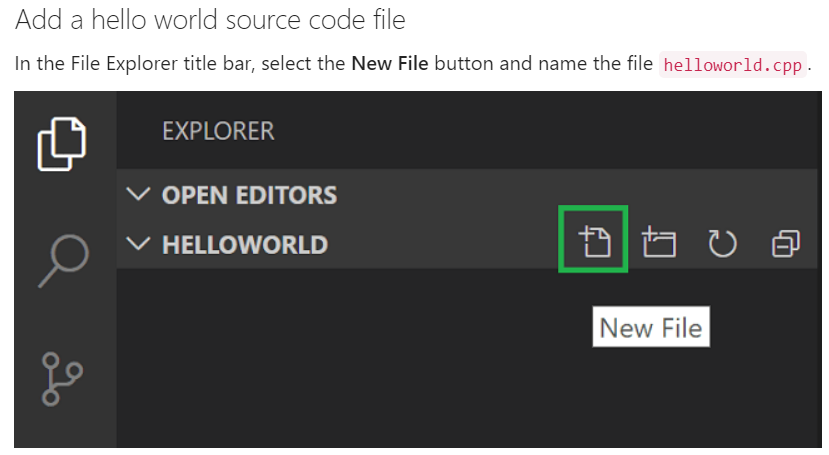


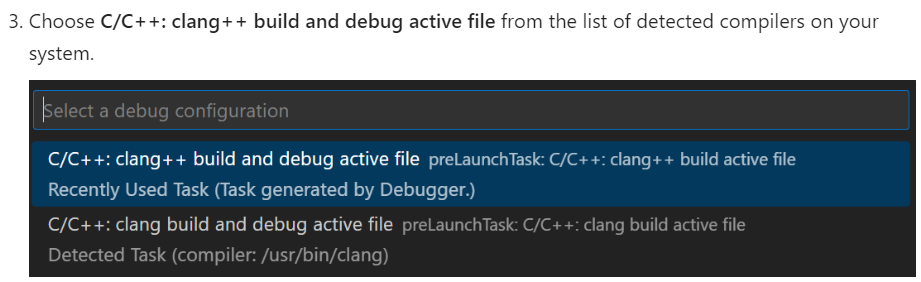
For Mac

<https://code.visualstudio.com/docs/cpp/config-clang-mac#_prerequisites>



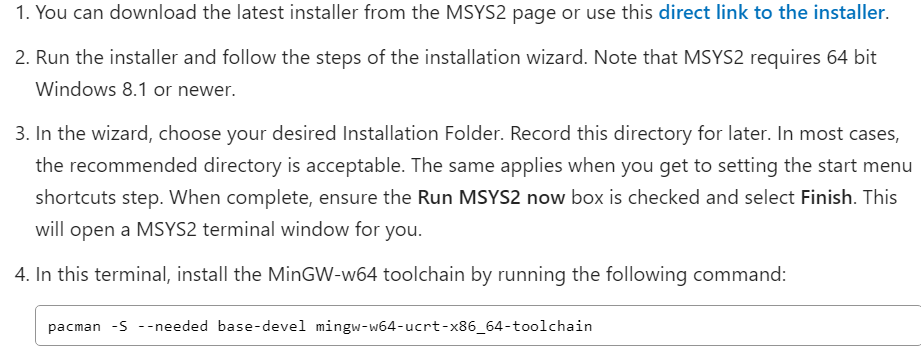


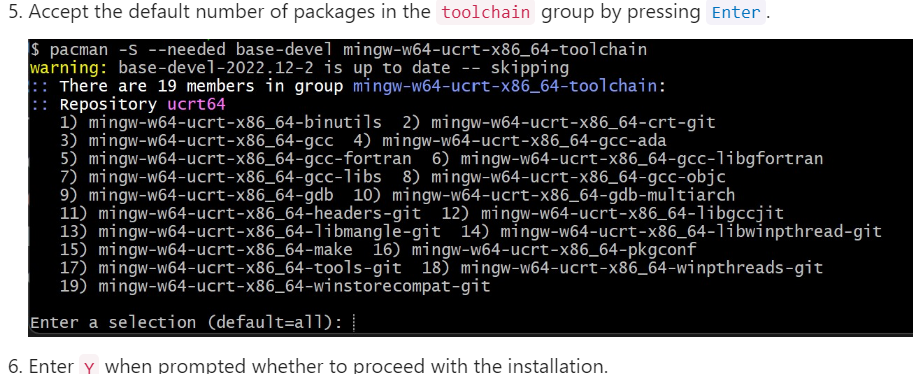


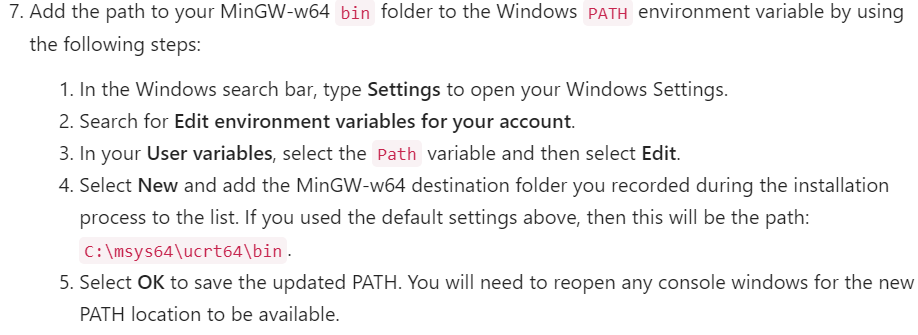


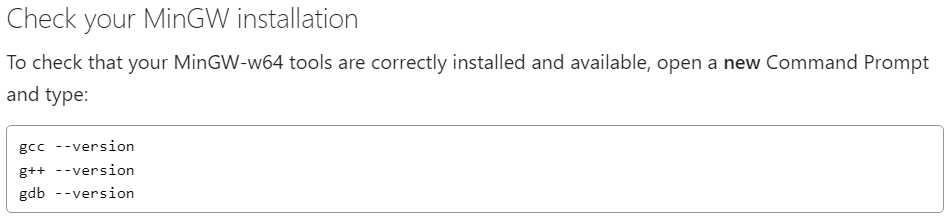
For windows:

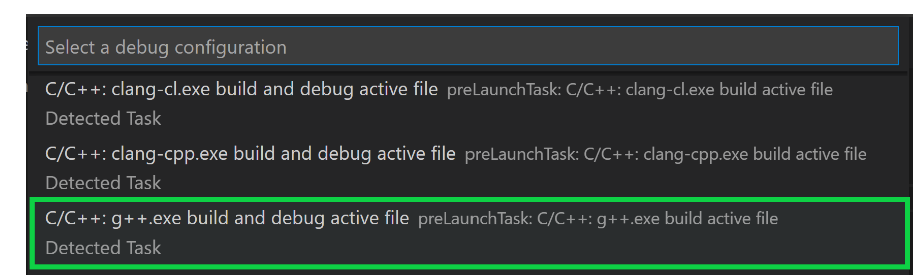
<https://code.visualstudio.com/docs/cpp/config-mingw#_prerequisites>











Source code

#include <iostream>

#include <vector>

#include <string>

using namespace std;

int main()

{

vector<string> msg {"Hello", "C++", "World", "from", "VS Code", "and the C++ extension!"};

for (const string& word : msg)

{

cout << word << " ";

}

cout << endl;

}

## Step by step for installing visual studio or windows

[Install Visual Studio and choose your preferred features | Microsoft Learn](https://learn.microsoft.com/en-us/visualstudio/install/install-visual-studio?view=vs-2022)

1. Download the Visual Studio Bootstrapper:

* Go to the Visual Studio downloads page.
* Select the version of Visual Studio you need (Community).
* Download the Visual Studio bootstrapper file (e.g., VisualStudioSetup.exe).

2. Install the Visual Studio Installer:

* Run the downloaded bootstrapper file.
* Follow the on-screen instructions to install the Visual Studio Installer.

3. Choose Workloads and Components:

* Once the installer is installed, open it.
* Select the workloads (feature sets) you need for your development tasks (e.g., ".NET desktop development", "ASP.NET and web development").
* Optionally, select individual components if you need specific features not included in the workloads.

4. Customize Installation (Optional):

* **Change the installation location:** You can choose a different drive or folder for the installation.
* **Install language packs:** Select the language packs you need.

5. Start the Installation:

* Click the "Install" button to start the installation process.
* Wait for the installation to complete.

### Create a new project using Visual Studio

1. A screenshot of a computer

   AI-generated content may be incorrect.Create a new project
2. choose windows console application as type

A screenshot of a computer

AI-generated content may be incorrect.

1. Give the name of project and location of the project file and solution file

A screenshot of a computer

AI-generated content may be incorrect.

1. Build solution and run the program

A screen shot of a computer

AI-generated content may be incorrect.