**C++ Notes**

**Last Modified: 2023 September 6**

**Chapter 1: Theory**

**This section includes some theory questions, such as algorithms and understanding of the basic theory. But, some practical understandings (i.e., what is the use of comments?) are discussed in Chapter 3.**

* What is C++?
  + C++ is a cross-platform language that can be used to create high-performance applications.
  + C++ gives programmers a high level of control over system resources and memory.
  + The language was updated 4 major times in 2011, 2014, 2017, and 2020 to C++11, C++14, C++17, C++20.
* Why use C++?
  + C++ is an **object-oriented programming language** which gives a clear structure to programs and allows code to be reused, lowering development costs.
* Difference between C and C++
  + C++ was developed as an extension of C, and both languages have almost the same syntax.
  + The main difference between C and C++ is that C++ support classes and objects, while C does not.
* Why use visual code as an editor or an IDE (Integrated Development Environment)?
  + It is easy to install, and it is cross-platform. It runs well on Windows, Mac, and Linux.
* What is a compiler?
  + A compiler is a software that compiles our codes into code that can directly run on the hardware or binary executable format.
  + Testing our code on multiple compilers can make our code more portable.
  + Example: Mingw (GCC), Msvc, Clang llvm
* What is compile error and runtime error?
  + **Compile Error:** Occurs when you violate the rules of writing syntax, such as missing a semicolon, a parenthesis or using a reserved word. The compiler detects these errors and prevents the code from running.
  + **Runtime Error:** Occurs during the execution of the code, such as dividing by zero, null pointer or logic error. These errors are usually not detected by the compiler but cause the program to produce wrong results or terminate abnormally.
* How is a cpp file executed?
  + When we compile our program, the compiler converts our code into binary format, which is unreadable by humans but readable by machines. It is converted into binary format because it is easily understandable by the CPU.

A computer screen with a computer code

Description automatically generated

* When we run our program, the binary file is loaded up in memory (RAM), which is the program area in the following graph. CPU starts running the program statements by statements. It starts at the top (in our example, allocating space to store our integer ***a***). If a variable value is not specified, some junk value will be stored instead (e.g., ***c***). The CPU keeps a return address for a function to know where it should return after executing the function.

A computer screen shot of a computer

Description automatically generated

* What is Statement?
  + A statement is a basic unit of computation in a C++ program. Every C++ program is a collection of statements organized in a certain way to achieve some goal.
  + Statements end with a semicolon in C++ (***;***).
  + Statements are executed in order from top to bottom when the program is run.
  + Execution keeps going until there is a statement causing the program to terminate or run another sequence of statements.
* What are Core Features?
  + Core features are a basic building block of the C++ programming language. For example, How to define variables, How to use a function in C++, etc.
* What is Standard Library?
  + A set of highly specialized components that we can easily used in C++ programs.
  + E.g., ***#include <iostream>***, ***#include <string>***
* What is STL (Standard Template Library)?
  + A part of C++ standard library, but it is collection of container and algorithm types. It is used to implement common data structures and algorithms in C++ programs.
  + Advantage: It provides a way to write generic, reusable code that can be applied to different data types.
  + E.g., ***List***, ***Vector***, ***Map***, ***Sort***, ***Find***, ***Transform***
* What is an Escape Sequence?
  + Escape sequence forces the cursor to change its position to the beginning of the next line on the screen.

|  |  |
| --- | --- |
| **Escape Sequence** | **Description** |
| **\t** | Creates a horizontal tab. |
| **\\** | Inserts a backslash character (\) |
| **\"** | Inserts a double quote character. |

**Chapter 2: Configuration Setting**

* How to download C++ compilers
  + Download Mingw and Clang llvm in one place: <https://winlibs.com/>
* How to test whether the compiler GCC has been installed and added to the path (Windows)
  + *g++ --version* & *clang++ --version* in cmd

A screenshot of a computer program

Description automatically generated

**2.1 Visual Studio**

* How to check whether Visual Studio has been appropriately downloaded in Windows
  + Open Developer Command Prompt for VS and type cl.exe

A screenshot of a computer program

Description automatically generated

**2.2 Visual Studio Code**

* How to **run a cpp file** in Visual Studio Code?

1. Install C/C++ Extension.
2. Terminal -> Configure Tasks -> Choose the preferred compiler.

A screenshot of a computer

Description automatically generated

1. A JSON file will be created, which is our configuration to configure which compilers VS code uses to compiler our *main.cpp* file.
2. To run the cpp file in a particular C++ version: google GCC C++ <version>, e.g., GCC C++ 20
3. From <https://stackoverflow.com/questions/66975491/how-to-use-c-20-in-g>, we know we should add *-std=c++20* in JSON file.

A screen shot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

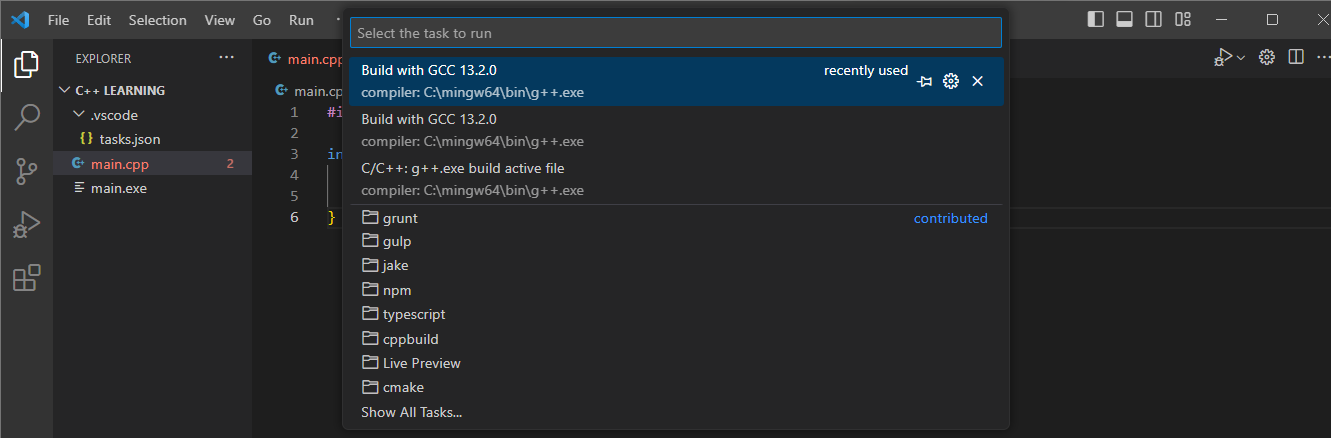
We can also check the tutorial from the C++ extension.

1. Open a new terminal -> Run task -> *main.exe* is created.

A screenshot of a computer

Description automatically generated

1. Select the compiler. (We can change the display name by modifying labels in the JSON file.)



1. If the IDE has been connected to our gcc compiler, we can type *.\main.exe* to run the program.

A black background with white text

Description automatically generated

* It is always good to have stored args so that we can directly copy and paste our configuration into each project's json. *(${workspaceFolder}\\\*.cpp* builds multiple files in the same project.)

"args": [

                "-fdiagnostics-color=always",

                "-g",

                "-std=c++20",

                "${workspaceFolder}\\\*.cpp",

                "-o",

                "${fileDirname}\\${fileBasenameNoExtension}.exe"

         ]

* How can we configure our project to compile it in another compiler?
  1. Terminal -> Configure Task -> Select the new compiler

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* 1. The JSON file will have a new configuration for the new compiler.
  2. The args section can be the same as GCC.
* How to use the compiler from Microsoft (Msvc)
  1. Open Developer Command Prompt for VS
  2. *cd <project dir>*
  3. Open the project in VS code by command *code .*

A screenshot of a computer program

Description automatically generated

* Why do we open our project in this way?
  + Because the terminal can only identify Msvc in this situation as the compiler has not been added into PATH.

A screenshot of a computer

Description automatically generated

1. Terminal -> Configure Task -> C/C++: cl.exe build active file

A screenshot of a computer

Description automatically generated

1. Default args for Msvc json file:

"args": [

                "/Zi",

                "/std:c++latest",

                "/EHsc",

                "/Fe:",

                "${workspaceFolder}\\\*.cpp",

                "${fileDirname}\\${fileBasenameNoExtension}.exe"

         ]

* How to configure C/C++ extension in VS code?
* View -> Command Palette -> C/C++ Edit Configurations (UI)

A screenshot of a computer

Description automatically generated

**Chapter 3: Code Examples**

**3.1 First Program**

#include <iostream>

// using namespace std;

consteval int get\_value(){

    return 3;

}

int main(){

    constexpr int value = get\_value();

    std::cout << "value : " << value << std::endl;

    return 0;

}

***#include <iostream>***: To include a standard library header file, which provides input and output functionality, such as ***std::cout*** and ***std::endl***.

***int main()***: The entry point of a C++ program.

***std::cout***: Print something.

***std::endl***: Print a new line. This can also be replaced by ***cout << "\n"***.

***using namespace std***: Can be used to omit ***std::***

***return 0***: Send message to the operation system if the program is finished successfully or if there is some kind of problem.

**3.2 Comments**

#include <iostream>

// Entry point main function

int main(int argc, char \*\*argv)

{

    //One line comment

    /\*

        Multi-line block comment

        Another line

        Oh! And another one!

    \*/

   //Print out some text

   std::cout << "Hello World in C++20!" << std::endl;

   return 0;

}

* What is the use of comments?
  + Comments can be used to explain C++ code, and to make it more readable.
  + It can also be used to prevent execution when testing alternative code.
  + The compiler will ignore comments.

**3.3 Functions**

#include <iostream>

// A function that adds two numbers and returns the result

int addNumbers(int first\_number, int second\_number){

    int sum = first\_number + second\_number;

    return sum;

}

int main(int argc, char \*\*argv){

    int firstNumber = 12;

    int secondNumber {9}; // Another way to initialize variables

    int sum = firstNumber + secondNumber;

    std::cout << "The sum of " << firstNumber << " and " << secondNumber <<" is " << sum << std::endl;

    std::cout << "The sum of " << firstNumber << " and " << secondNumber <<" is " << addNumbers(firstNumber, secondNumber) << std::endl;

    return 0;

}

***int addNumbers***: A function that adds two numbers and returns the result.

* A function must be defined before it is used.
* The advantage of function is reusable.

**3.4 Inputs and Outputs**

#include <iostream>

#include <string> // To use std::string

int main(int argc, char \*\*argv){

    // std::cout: Printing stuff to the console

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

    // Error

    std::cerr << "std::cerr output: Something went wrong." << std::endl;

    // Log message

    std::clog << "std::clog output: This is a log message." << std::endl;

    int age;

    std::string name;

    std::cout << "Please type in your Last Name: " << std::endl;

    // Input

    std::cin >> name;

    std::cout << "Please type in your age: " << std::endl;

    std::cin >> age;

    std::cout << "Hello " << name << "! You are " << age << " years old." << std::endl;

    // Chaining std::cin

    std::cout << "Please type in your Last Name and age, separated by spaces: " << std::endl;

    std::cin >> name >> age;

    std::cout << "Hello " << name << "! You are " << age << " years old." << std::endl;

    // Reading data with spaces

    std::string full\_name;

    std::cout << "Please type in your full name and age: " << std::endl;

    std::getline(std::cin, full\_name); // Enter then input age

    std::cin >> age;

    std::cout << "Hello " << full\_name << "! You are " << age << " years old." << std::endl;

    return 0;

}

***std::cout***: To print/output the variable to the console(terminal). The output is stored in a buffer until it is flushed. The buffer is typically flushed when a newline character (***\n***) is encountered or the program exits.

***std::cerr***: An unbuffered stream that is useful for printing error messages or other critical information that needs to be displayed immediately. The output is immediately sent to the console without being stored in a buffer.

***std::log***: Printing log messages to the console. It is buffered like ***cout***. This makes it useful for logging information that does not need to be displayed immediately, but still needs to be stored in a buffer for later retrieval.

***std::cin***: Reading data from the terminal. The example above shows three usages of ***cin***.

***#include <string>***: To include a standard library header file, which provides ***std::string***

**References**

* [**C++ Programming Course - Beginner to Advanced**](https://www.google.com/url?q=https://www.youtube.com/watch?v%3D8jLOx1hD3_o%26ab_channel%3DfreeCodeCamp.org&sa=D&source=calendar&usd=2&usg=AOvVaw32IHq1-uzixQnuCtxAm8fA)
* [**W3schools C++ Tutorial**](https://www.w3schools.com/cpp/cpp_intro.asp)