COSC1076 | Semester 1 2020 | Advanced Programming Techniques

Lab | Week 01

Overview

The Week 01 Lab is about getting yourself set-up and ready-to-go for class.

Lab Questions

This lab contains a series of exercises. You may have done many of these exercises in Java already, however, it is good to practice these same exercises to become familiar with writing and compiling C++ programs.

It is a good idea to attempt the lab questions before coming to class. The lab might also be longer than you can complete in 2 hours. It is a good to finish the lab at home.

You should demonstrate your work to your tutor.

! If you are using the School Lab Computers, then you don't need to do the setup parts. Use the information on Canvas to use VSCode which is setup there.

Unix Proficiency

If you found working with the terminal in the above C++ program confusing, or didn't really know what was going on, then you should do the Unix Lab Induction as linked on Canvas.

Setup (Editor)

To work in C++ you could use literally any editor or IDE that you wish. To make things consistent though (including other course) we are going to use the VSCode editor. It runs on Windows, Mac and Linux. It is also already installed on the University Computers, so we can just use it from there.

Activity: If you are using your own computer, Download and install VSCode for your laptop.

Activity: After this you need to install the C++ Extension for VSCode, so VSCode will know how to recognise C++ programs.

Setup (C++ Compiler)

As shown in the simple diagram below, to write and run C++ programs, requires a *compilation stage*. This turn the C++ text program into *machine code* that a computer can actually run[1].

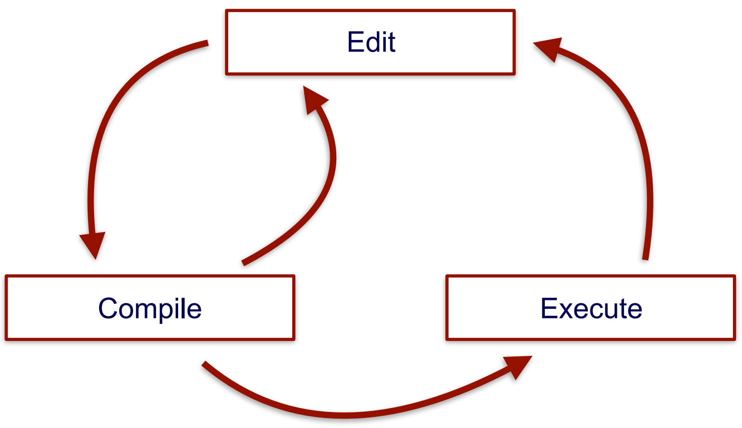


Figure 1: C++ Development Cycle

The C++ compiler you need is specific to your operating system:

1. For MacOS: Install XCode from the Mac App Store.

[1]Aside: When you wrote a Java Program and "ran" it, the Java Virtual Machine which is the program used to run Java programs would do this compilation process behind the scenes for you.

2. For Windows: Use the Windows Subsystem for Linux.
3. For Linux: Install the `build-essentials` package.

Activity: Install the C++ compiler for your operating system.

Connect your Compiler to VSCode

To make your lives easier we would like to invoke the C++ compiler from within VSCode. To do this we need to link VSCode and the compiler we just isntalled.

Activity: For your operating system, follow the instructions to link VSCode and C++. Scroll down to the section titles: ***Configure the compiler path***

- Mac with XCode
- GCC on Windows (with Linux subsystem)

(Alternative) Use the School Teaching Servers

If you wish to use the Lab Computers and/or work on the CSIT Teaching server(s), then you will need two additional pieces of software:

- SCP software (WinSCP on Windows)
- SSH Software (PuTTy on Windows)

For information on how to use these, please review the Unix Induction (as linked on Canvas).

Once you upload your code and log-in to the School Teaching Servers, (via SSH), ***make sure you remember to enable the C++ toolchain***, via the command:

```
scl enable devtoolset-8 bash
```

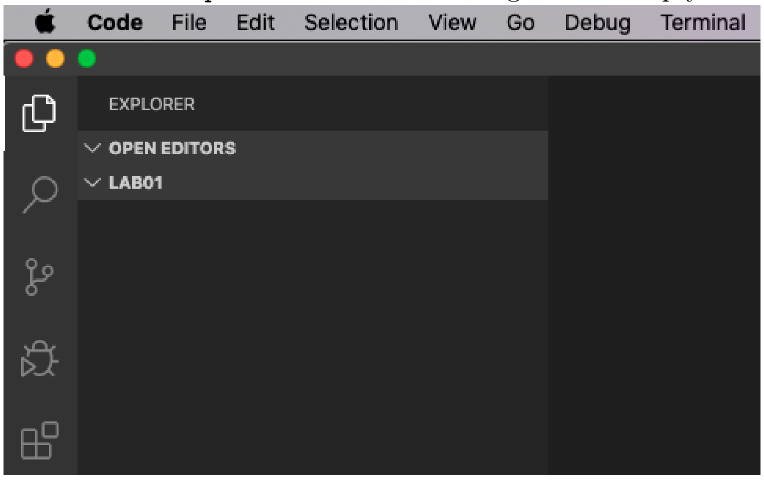
! If you can't run the compilation command (in the exercise below), then you most likely have forgotten to "scl-enable".

Try out your first C++ Program

The last bit of this lab is to create and run your first C++ program. Complete the following exercises. Once you have done this, show it to your tutor so we know you are on track.

Exercises:

1. On your compile create a new folder for `lab01`.
2. In VSCode, open this folder `File->Open...`. You should then get a new empty *workspace*.



3. Create a new C++ file called `helloworld.cpp`. In this new file, copy-and-paste the following very basic Hello World program[2]. We don't expect you to fully understand what is happening in this code...yet. For now we want to make sure you get familiar with creating, and running new C++ files.

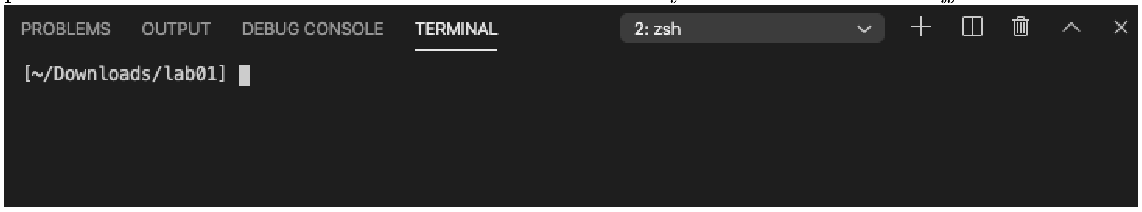
[2]Programming tradition dictates that the first program a person writes in any new programming language is "Hello World".

```cpp
#include <iostream>

int main(void) {
    std::cout << "Hello World!" << std::endl;

    return EXIT_SUCCESS;
}
```

4. Next we need to *compile* the program so that it can be run. We are going to use *command-line* compilation[3] where we will provide the explicit compilation command.
   (a) First we need to open a *Terminal*. From the Menu, select `Terminal -> New Terminal`. This should open a new terminal window as shown. The exact look of your terminal *will be different to mine.*

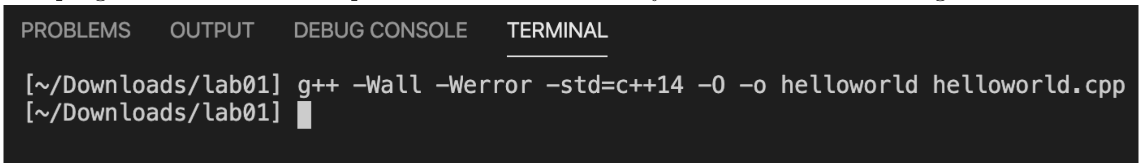


```
[~/Downloads/lab01] ▊
```

   (b) Next we compile the software with the command:

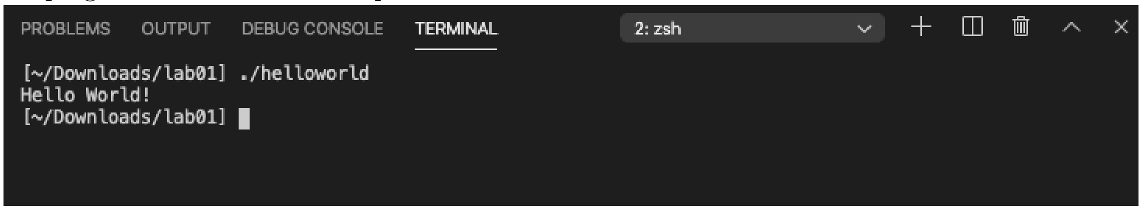
```
g++ -Wall -Werror -std=c++14 -O -o helloworld helloworld.cpp
```

   (c) Your program will now be compiled. If this is successful you should see something like below.



```
[~/Downloads/lab01] g++ -Wall -Werror -std=c++14 -O -o helloworld helloworld.cpp
[~/Downloads/lab01] ▊
```

   The exact text might differ, but the compilation should happen just the same.
   (d) What this does is compile your program. if you navigate to the folder on your computer where you saved the C++ file, you should see a couple of items appear:
      i. The C++ file: `helloworld.cpp`
      ii. A folder with a similar name. This is used for debugging and *we will be ignoring it for now.*
      iii. An *executable* with the name `helloworld`. (If you are on windows it might be `helloworld.exe`)
5. Now we can run the *executeable* we created in the above step. For this we are go back to the terminal:
   (a) To execute the program type `./helloworld` and hit enter
   (b) The program should run and output the text `Hello World!` to the terminal window.



```
[~/Downloads/lab01] ./helloworld
Hello World!
[~/Downloads/lab01] ▊
```

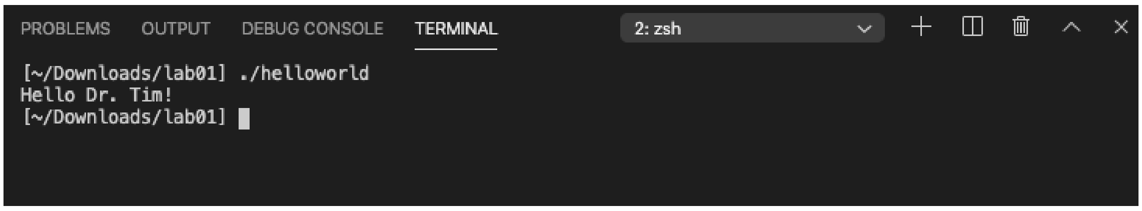
If you successfully complete all of these activities, well done! You are well on your way for this course and ready for the first lecture.

Bonus Exercises

! **Practice** is very important for this course. Practice includes trying this out for yourself. These are bonus exercises for you to experiment with the code.

1. Modify the hello-world program to instead print out your own name.
2. Once you have made the modifications, remember to *compile* and *execute* your program. For example, mine gives:

[3]We could use VSCode to help, but we want to learn more general skills in this course.



```
[~/Downloads/lab01] ./helloworld
Hello Dr. Tim!
[~/Downloads/lab01] █
```

3. Modify your program to print your name 3 times.
4. (Optional) Modify your program to print your name 10 times *using a for-loop*. Of course we haven't taught you this yet. However, your *might* be able to figure this out on your own. The syntax for Java and C++ are very similar[4]. Using your Java knowledge can you figure out how to use a for-loop?.

[4]In fact both are C-style programming languages.