**ITCS 1212L**

**Lab lessons 1**

Computer is a programmable device. The **central processing unit** (**CPU**) of a computer is a piece of hardware that carries out the instructions of a computer program. The main tasks of CPU are Data transfer from memory to CPU registers (Accumulator) and vice versa and processing data inside ALU. Generally it performs the basic arithmetical, logical, and input/output operations of a computer system. (as shown in Figure.1)

**CPU**

ALU

Output

Input

Registers

Control Unit

Storage (Memory)

Figure 1. the flow of data in the computer

As mentioned above a typical **CPU** has a number of **components**. Including arithmetic logic unit (ALU) which performs simple arithmetic and logical operations. The other components are: Registers such as Instruction Register (IR), Accumulator, Program counter and Decoder(as shown in figure2). In addition to these parts, CPU also includes a Control Unit (CU) which controls the operations inside the CPU as well as operations related to other devices outside CPU such as memory. (as shown in Figure1)



Figure2. Sample of binary codes in CPU

In the CPU, the Programming is in machine language which is binary. Assembly is corresponding to machine language in symbolic form. (figure.2)



Figure.3 components of CPU and sample of Assembly language in RAM

It is going to be extremely difficult to use machine language for programming and It is easier to use assembly language as shown in figure 2. In these assembly instructions the CPU calculates (Z+Y)\* X .However, assembly is still hard to use because it is close to hardware. Therefore we need a programming language which is close to our language. We call it a High Level Language (HLL). C++ is a HLL language, but, the question is that is it possible to execute the C++ code in the processor (cpu)? No, it is not since it has to be translated into machine language.

Compiler is a tool (a program or set of programs) that translate a HLL into machine language. Figure 3, bellow illustrates the role of compiler:



Figure4. The Role of compiler

**Integrated Environments:** An integrated development environment (IDE) is a software package that bundles an editor (used to write programs), a compiler (that translates programs) and a run time component into one system. For example, the figure 4 below shows a screen from the Code::Blocks integrated environment.

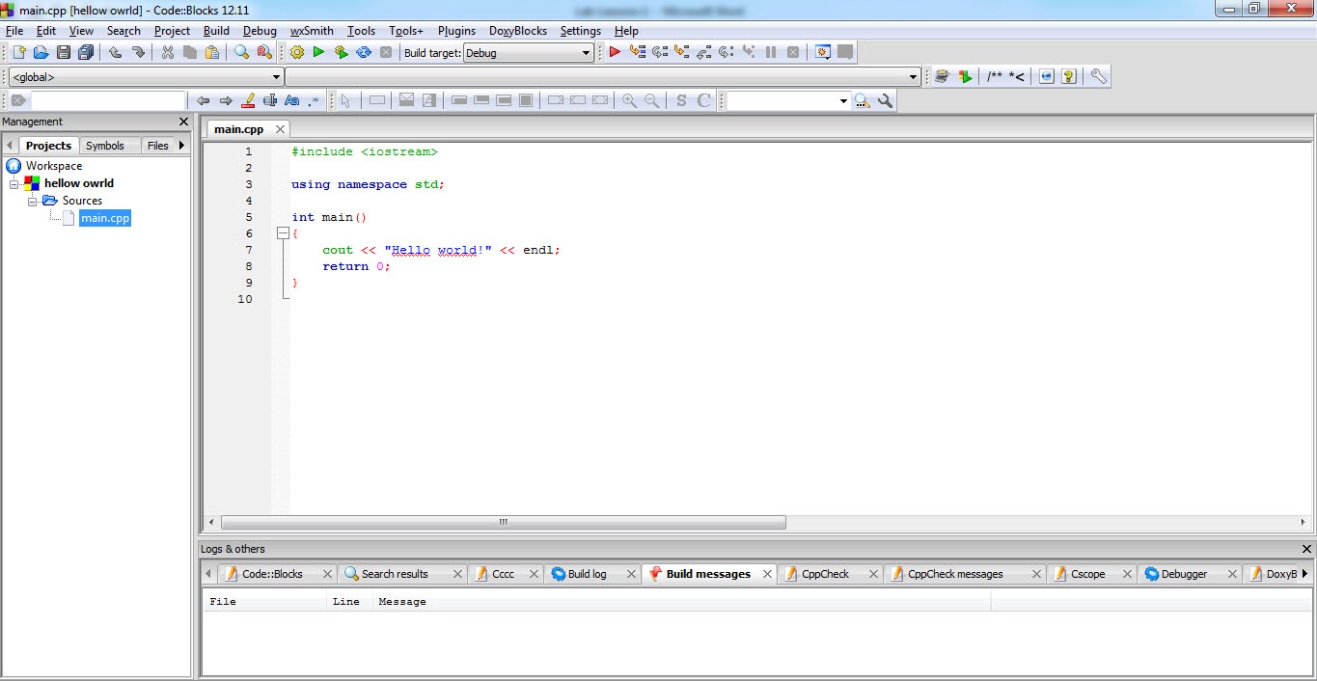


Figure 5. Code::Blocks environment

In lab1 you will enter source code using an IDE called Code::Blocks then compile, link and run the executable. You also learn about syntax and logic errors.

The programmer types his/her source code using a text editor. We are using a “suped up” text editor that is part of an IDE (Integrated Development Environment) called Code::Blocks. The text editor has tools built in to help the programmer, such as curly- brace matching and color-coding parts of the program code.

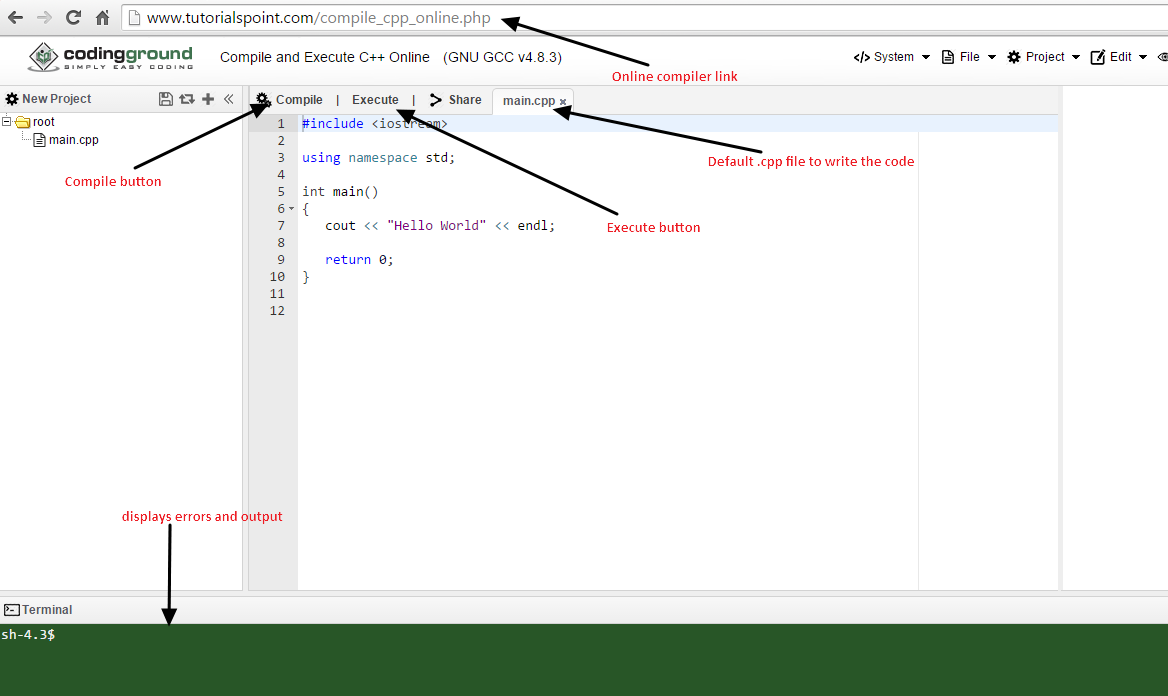
Your source code is saved with the extension of .cpp, for c++. This source code file serves as input to the compiler, a program which checks for syntax errors and if there are none, converts the source code to machine code. The output of the compiler program is a file with the extension .o. The linker program takes as input the object code from your compile and object code from pre-written and pre-compiled library routines. The output from the linker program is an executable file (.exe), which will run on your computer. Figure 5 bellow summarizes the translation process.



Figure6. translation process

**If in case you don’t have code blocks installed in the system, you can use the online compiler environment just by clicking the below link.**

[**http://www.tutorialspoint.com/compile\_cpp\_online.php**](http://www.tutorialspoint.com/compile_cpp_online.php)



**Figure 7. online compiler environment**

In order to use this online compiler environment, you should be aware of the below mentioned options available on it.

**main.cpp :** Is the file in which you can write the program

**Compiler button:** Lets you compile the program just by clicking it

**Execute button:** Lets you execute the program just by clicking it

**Terminal:** Output of each command run is displayed at the terminal

Note: you can also explore the File, Project buttons available here

This environment provides you with several options to run the program.

1. You can directly copy and paste the program that you have in your local text editor to the main.cpp file and compile and execute it.

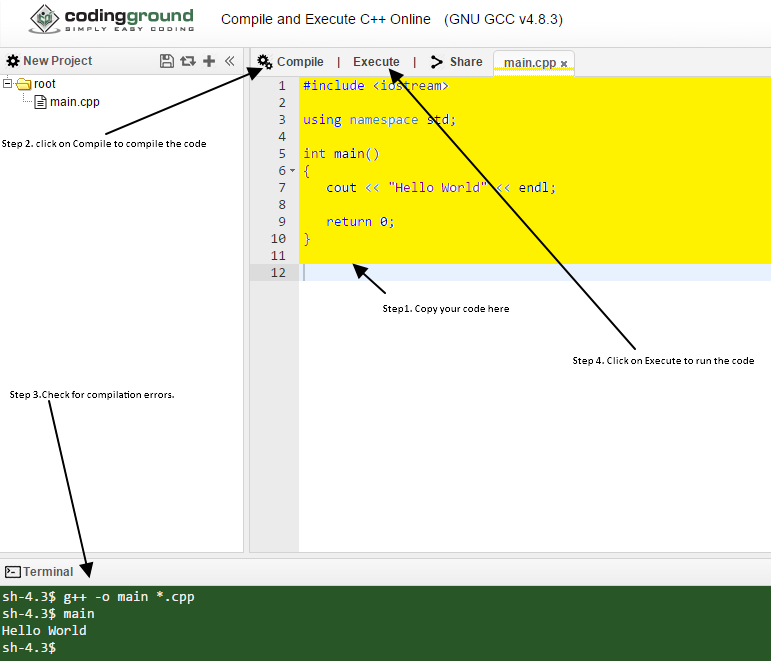


Figure 8. copy paste code to online compiler environment

1. You can upload your .cpp file here by following the below steps:

Make sure the root directory is not having the default main.cpp file(you can right click on main.cpp file and delete it)

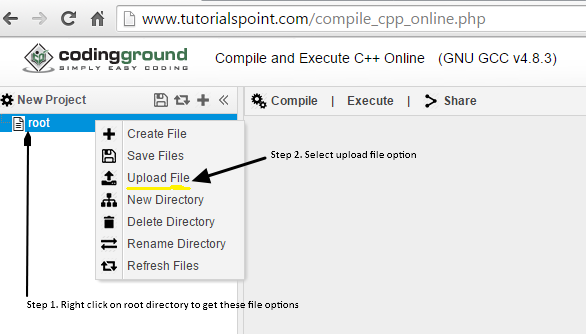


Figure 9. file options

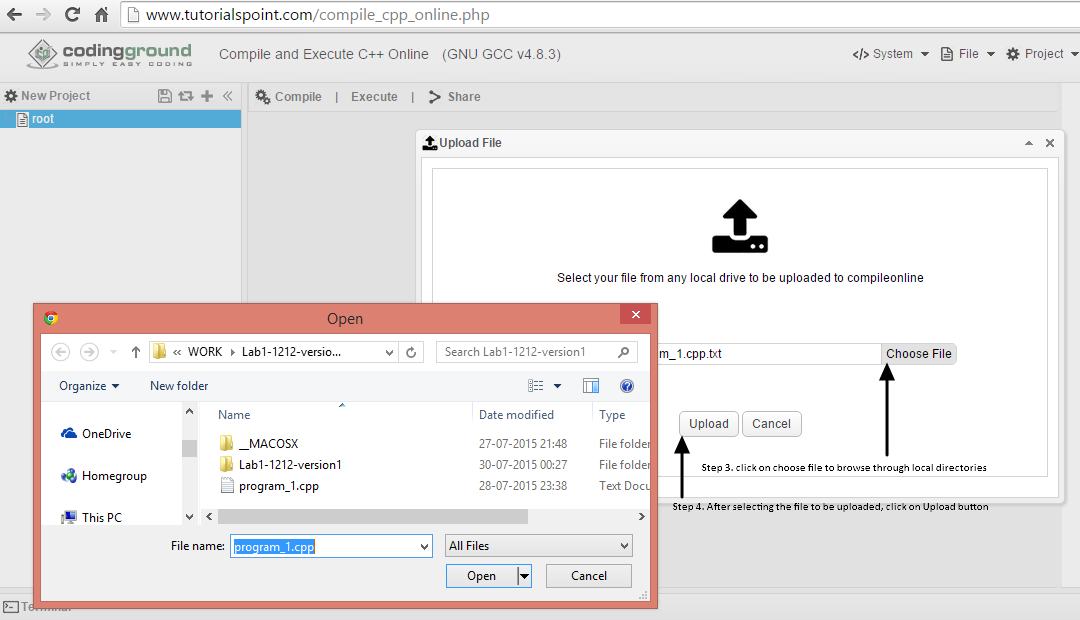


Figure 10. Uploading file option on online compiler environment

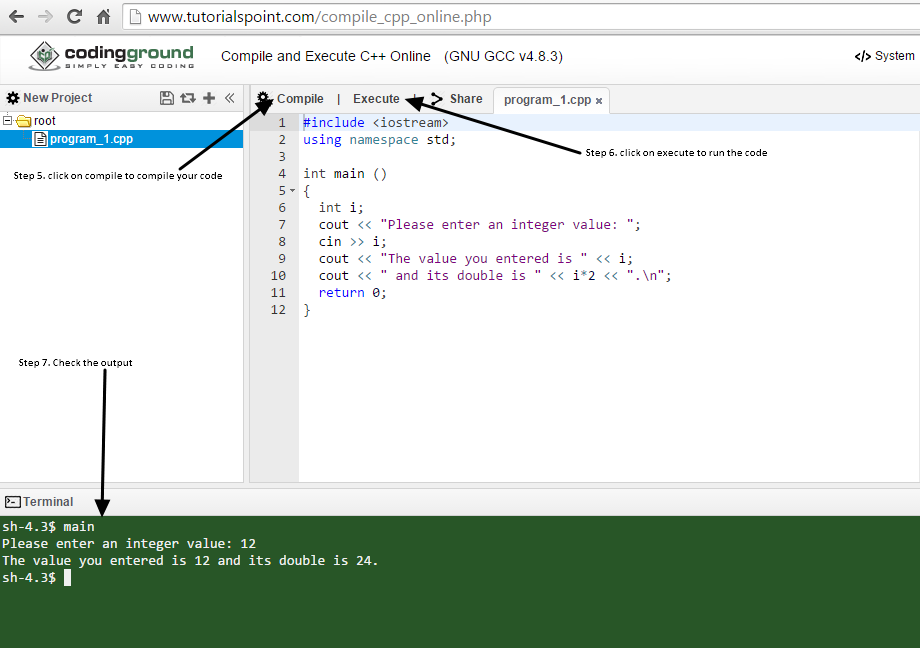


Figure 11. compile and execute code on online compiler environment

You might want to save your code back to local system either as a .cpp file or as a project. This online compiler environment provides you with options to download file as well as project.

Saving the code as a file to local system:

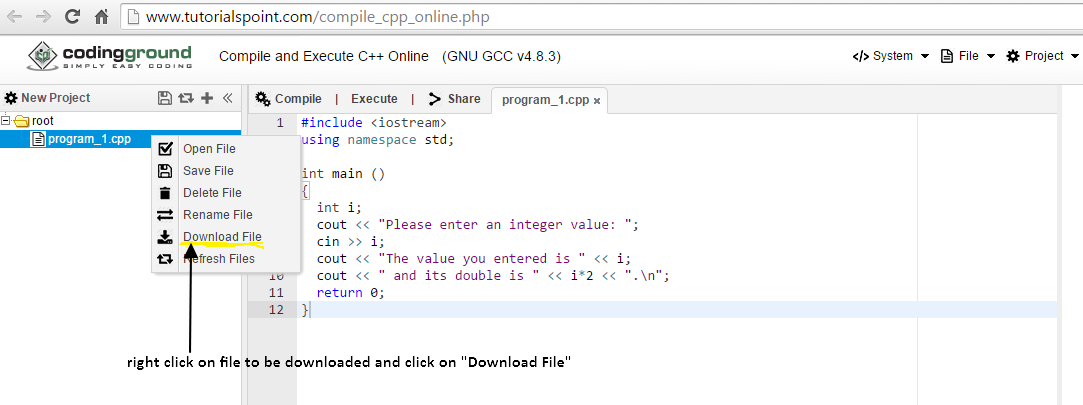


Figure 12. download .cpp file to local system

Saving the code as a project to local system:

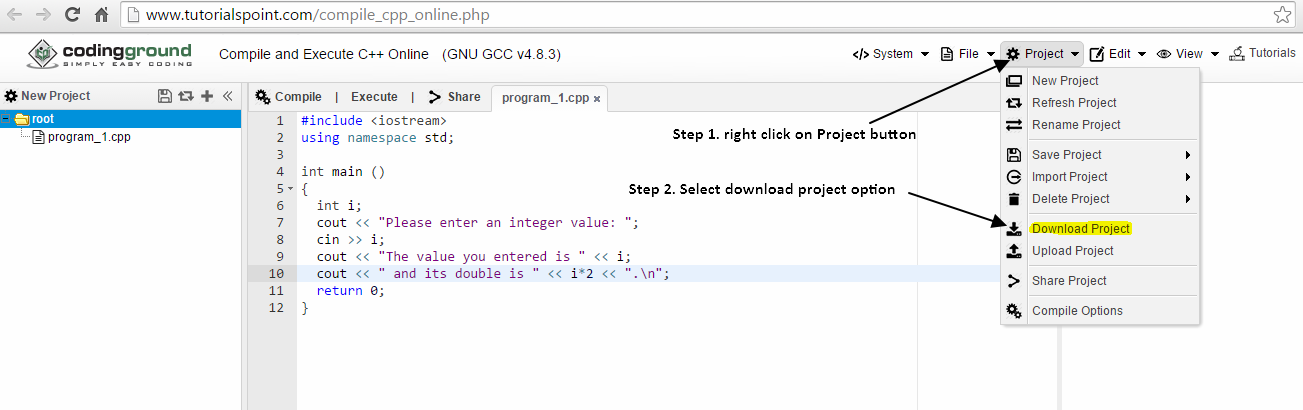


Figure 13. Download project on online compiler environment

*Note: Online compiler will download the project as a zip file. When you want to upload the project on this environment it expects the zip file.*

Also there is edit button which lets you with normal text editing options for editing the code.



Figure 14. Edit option on online compiler environment

**Logic error vs. Syntax error**: Syntax errors occur when a program does not conform to the grammar of a programming language, and the compiler cannot compile the source file. Logic errors occur when a program does not do what the programmer expects it to do.   
Syntax errors are usually easy to fix because the compiler will tell you where the error occurs and you simply fix the syntax error. For example you may miss a semicolon or a curly bracket where it’s supposed to be. Simply locate those errors and fix them. Logical errors indicate the logic used when coding the program failed to solve the problem. You do not get error messages with logic errors. Your only clue to the existence of logic errors is the production of wrong solutions.

**Algorithm**: Algorithm is a formula or set of steps for solving a particular problem. To be an algorithm, a set of rules must be unambiguous and have a clear stopping point. Algorithms can be expressed in any [language](http://www.webopedia.com/TERM/L/language.html), from [natural languages](http://www.webopedia.com/TERM/N/natural_language.html) like English or French to [programming languages](http://www.webopedia.com/TERM/P/programming_language.html) like [C++](http://www.webopedia.com/TERM/F/FORTRAN.html).  
We use algorithms every day. For example, a recipe for baking a cake is an algorithm. Most [programs](http://www.webopedia.com/TERM/P/program.html), consist of algorithms. Inventing elegant algorithms - algorithms that are simple and require the fewest steps possible - is one of the principal challenges in programming.

## Qualities of a good algorithm

1. Inputs and outputs should be defined precisely.
2. Each steps in algorithm should be clear and unambiguous.
3. Algorithm should be most effective among many different ways to solve a problem.
4. An algorithm shouldn't have computer code. Instead, the algorithm should be written in such a way that, it can be used in similar programming languages.

Example of an algorithm to **add two numbers entered by user.**

Step 1: Start

Step 2: Declare variables num1, num2 and sum.

Step 3: Read values num1 and num2.

Step 4: Add num1 and num2 and assign the result to sum.

Sum ← num1 + num2

Step 5: Display sum

Step 6: Stop