Computer Programming Fundamentals

Handouts No. 3: Computer Programming Languages

Computer Program

A program is a sequence of instructions (called programming statements), executing one after another - usually in a sequential manner.

Programming

Programming is telling the computer what to do using the computer own language.

Programming Language

Language has been our primary mean of communication and human interaction for thousands of years. For a community, the language contained the words that the people need to communicate, words themselves are abstract, but they indicate the meaning, they point to objects or actions, etc.

The programming language is pretty much like writing a paragraph of instruction or creating a to-do list to computers. Unlike us humans, the to-do list and instructions you write for the computer has to be extremely detailed and written in some logic.

High-Level and Low-Level Languages

Both high-level language and low-level language are the programming languages' types. The main difference between high-level language and low-level language is that, Programmers can easily understand or interpret or compile the high-level language in comparison of machine. On the other hand, Machine can easily understand the low-level language in comparison of human beings.

Examples of low-level languages are assembly and machine language.

Examples of high-level languages are C, C++, Java, Python, etc.

Machine Language

Machine language is a low-level language comprised of binary digits (ones and zeros). Since computers are digital devices, they only recognize binary data. Every program, video, image, and character of text is represented in binary. This binary data, or machine code, is processed as input by the CPU. The resulting output is sent to the operating system or an application, which displays the data visually. For example, the ASCII value for the letter "A" is 01000001 in machine code, but this data is displayed as "A" on the screen. An image may have thousands or even millions of binary values that determine the color of each pixel.

Assembly Language

An assembly language is a low-level programming language designed for a specific type of processor. While assembly languages differ between processor architectures, they often include similar instructions and operators. Below are some examples of instructions supported by x86 processors.

MOV - move data from one location to another

ADD - add two values

SUB - subtract a value from another value

PUSH - push data onto a stack

POP - pop data from a stack

JMP - jump to another location

Source Code, Object Code, and Executable Code

- **Source code** is the set of instructions and statements written by a programmer using a computer programming language. This code is later translated into machine language.
- **Object Code** is a set of instructions in machine language translated from a source code.
- **Executable Code** is a set of instruction in machine language generated form one or multi object code files linked together.

Assembler, Complier, Interpreter, and Linker

- **Assembler** is a program that translates code from assembly language to machine language.
- **Compiler** is a program that translates the whole source code file from high-level language to machine code.
- **Interpreter** is a program that translates source code file from high-level language to machine language also. However, the interpreter translates a statement and executes it before it goes to the next statement. On the other hand, the compiler translates the whole file before it runs it.
- **Linker** is a program that combines multiple object files into one executable program file.

C \ C++ Programming Languages

The C programming language is a general-purpose, middle-level language originally developed by Dennis M. Ritchie at Bell Labs. It was created over the period 1969 through 1973 for the development of the UNIX operating system, which had previously been written in assembly language. The name C was chosen because many of its features derived from an earlier language called B. Whereas the B language is no longer in common use, C became and still remains one of the most popular and influential programming languages in use today.

Although C is a general-purpose language it is most often used for systems programming. This includes software that controls the computer hardware directly, such as drivers, operating systems, and software for embedded microprocessors. C can also be used for writing applications, which run on top of system software.

C++ was developed by Bjarne Stroustrup of AT&T Bell Laboratories in the early 1980's, and is based on the C language. The name is a pun - "++" is a syntactic construct used in C (to increment a variable), and C++ is intended as an incremental improvement of C. Most of C is a subset of C++, so that most C programs can be compiled (i.e. converted into a series of low-level instructions that the computer can execute directly) using a C++ compiler.

A computer isn’t smart. Believe it or not, on your worst days, you are still light-years ahead of your computer in intelligence. You can think, and you can tell a computer what to do. Here is where the computer shines: It will obey your instructions. Your computer will sit for days processing the data you supply, without getting bored or wanting overtime pay.

The computer can’t decide what to do on its own. Computers can’t think for themselves, so programmers (people who tell computers what to do) must give computers extremely detailed instructions. Without instructions, a computer is useless; with incorrect instructions, a computer will not successfully execute your desired task. A computer can no more process your payroll without detailed instructions than an automobile can start by itself and drive around the block independently. The collection of detailed expressions that you supply when you want your computer to perform a specific task is known as a program.

Note:

Word processors, apps, spreadsheets, and computer games are nothing more than computer programs. Facebook is a collection of programs. Without such programs, the computer would just sit there, not knowing what to do next. A word-processing program contains a list of detailed instructions, written in a computer language such as C++, that tells your computer exactly how to be a word processor. When you program, you are telling the computer to follow the instructions in the program you have supplied.

You can buy or download thousands of programs for your computer, tablet, or phone, but when a business needs a computer to perform a specific task, that business hires programmers and developers to create software that follows the specifications the business needs. You can make your computer or mobile device do many things, but you might not be able to find a program that does exactly what you want. This book rescues you from that dilemma. After you learn C++, you will be able to write programs that contain instructions that tell the computer how to behave.

Programs produce output when you run or execute them. The prepared dish is a recipe's output, and the word processor or app is the output produced by a running program.

Tip:

A computer program tells your computer how to do what you want. Just as a chef needs a recipe to make a dish, a program needs instructions to produce results. A recipe is nothing more than a set of detailed instructions that, if properly written, describes that proper sequence and the contents of the steps needed to prepare a certain dish. That's exactly what a computer program is to your computer.

Warning:

Just as when a chef gets an ingredient wrong or misses a step in a recipe, the resulting dish can be inedible; if you mistype code or skip a step, your program will not work.

The Programming Process

Most people follow these basic steps when writing a program:

1. Decide exactly what the program should do.
2. Use a text editor to write and save your programming language instructions. An editor is a lot like a word processor (although not usually as fancy) that lets you create and edit text. All the popular C compilers include an integrated editor along with the programming language compiler. All C program filenames end in the .c file extension.
3. Compile the program.
4. Check for program errors. If any appear, fix them and go back to step 3.
5. Execute the program.

Just in case you still don't fully understand the need for a compiler, your source code is like the raw materials that your computer needs. The compiler is like a machine that converts those raw materials to a final product, a compiled program that the computer can understand.

Note:

An error in a computer program is called a **bug**. Getting rid of errors is called **debugging** a program.

Warning:

If you have never programmed, this all might seem confusing. Relax. Most of today's C++ compilers come with a handy tutorial you can use to learn the basics of the compiler's editor and compiling commands.

Setup C++ Programming Environment

To begin programming in C++ on Windows operating system, you need a **text editor** and a **C++ compiler**. You can get both at the same time by installing an **Integrated Development Environment (IDE)** that includes support for C++. There are many IDEs that support C++ language such as Microsoft's Visual Studio Community Edition, which is a free version of Visual Studio, Eclipse, and Code::Blocks.

However, in this course we will use light-weight text editor, which is **Visual Studio Code** with **cpptools** extension which support c/c++ languages. As C++ compiler, we will use **MinGW**, which is a native Windows port of the **GNU Compiler Collection (GCC)**, with freely distributable import libraries and header files for building native Windows applications.

There is only one step left to start writing C++ programs. We have to add the GCC compiler directory to a Windows system variable called "Path".

Setting the "path" variable in Windows

- Press the Windows key+X to access the **Power User Task Menu**.
- In the **Power User Task Menu**, select the **System** option.
- In the About window, click the Advanced system settings link under Related settings on the far-right side.
- In the **System** Properties window, click the **Advanced** tab, then click the **Environment Variables** button near the bottom of that tab.
- In the **Environment Variables** window, highlight the **Path** variable in the **System** variables section and click the **Edit** button.
- Add or modify the path lines with the paths you want the computer to access. Each directory path is separated with a semicolon. In **Windows 10**, each line is added separately using the **New** button.

OR,

- from the start icon, search for "**edit the system environment variables**".
- Then, open the program **Environment Variables**.
- on the lower box "**system variables**", look for "**Path**" variable. When you find it, select it.
- choose Edit ➔ New ➔ write the directory of the bin folder, where you have installed the MinGW package.

  In my case, I installed MinGW on C:\ drive, therefore my new path will be **C:\MinGW\bin**.
- Restart your computer and start writing C++ programs.

Command Prompt

A lot of Windows users have never even touched the Command Prompt. With today's advanced operating systems, it's easy to use a computer without ever worrying about entering text commands in the command line.

The Command Prompt, often abbreviated to CMD, is the **c**ommand **l**ine **i**nterface (CLI) for Windows operating systems. A command line interface is a way of interacting with a computer directly using text commands.

Early PC operating systems, like MS-DOS, operated exclusively through command-line interfaces. There was no mouse cursor, window management, or other **g**raphical **u**ser **i**nterface (GUI) elements like we take for granted today.

However, in modern versions of Windows, you can still use the Command Prompt to interface with your computer directly instead of clicking through various menus. Power users prefer the Command Prompt for some tasks, as you can take actions with a few simple keystrokes that would require dozens of clicks in the GUI.

How to Open the Command Prompt in Windows

- Type "command prompt" into the Start menu to search for it. You can also type "cmd" (the short name of the executable that runs the Command Prompt) if you prefer.
- Press **Win + R** to open the Run box, then type "cmd" and hit Enter to open it.

Command Prompt Basics

When you open a Command Prompt window, you'll see some basic info about your current Windows version. You'll then see a line like the below:

C:\Users\Username>

This is your current location. Any commands you run that rely on location (such as deleting files) will take place in this folder. Other CMD commands are more general and don't rely on you being in a specific location.

It's important to know that when working in the Command Prompt, you must type commands exactly as they should be. Since you're issuing commands directly to your computer, it won't understand if you type something wrong.

If you type a command that your computer doesn't recognize, you'll see a message that says [Command] is not recognized... and Windows won't do anything.

The **dir** command, which is short for **directory**, will list the contents of the folder that you're currently in. As mentioned earlier, you can check this by looking at the folder that appears to the left of your current command.

To change your current location, use **cd** (short for **change directory**) followed by the folder you want to visit. Available folders are marked with **<DIR>** when you run the **dir** command. So for example, to move to your Desktop folder from your default user folder, you would type

```
cd Desktop
```

And to move up one folder, use:

```
cd..
```

or to return to the drive root, use:

```
cd\
```

cd command can be used to change directory to another one (only in **PowerShell** not in **Command Prompt**), for example to change the directory to drive E, use:

`cd E:` ← **in PowerShell**

`E:` ← **in Command Prompt (**without cd**)**

Use **md [new folder name]** to create a new folder. **md** is short of **make directory**. For instance, `md lab` will make a folder called **lab**.

CMD Management

If there's too much clutter on the Command Prompt's screen, type **cls** to clear the contents and start fresh. And if there's a command running that you want to cancel (maybe it's taking too long), hit **Ctrl + C** to end it.

Writing Your First C++ Program

You get to see your first C++ program in this chapter! Please don't try to understand every character of the C++ programs discussed here. Relax and just get familiar with the look and feel of C++. After a while, you will begin to recognize elements common to all C++ programs.

```
#include <iostream>

using namespace std;

int main()

  {

  cout<<"Hello Wolrd";

  return 0;

  }
```

Open your programming software (text editor) and type in the program as listed. Simple, right?

follow the following steps:

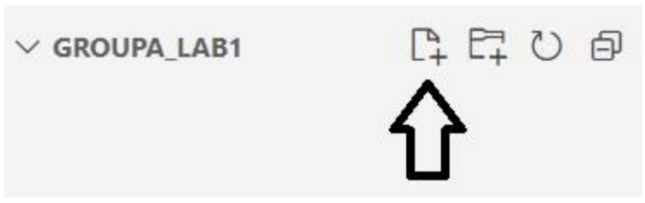
1. open the (Command Prompt)
    a. from start menu or search icon on the left bottom corner of your Windows machine, search for **cmd**
       Then select Command Prompt
    b. In the keyboard, press **Win + R**, then type **cmd** and press **Enter**
2. Using cd navigate to your user folder (you should be there by default)
3. Create a folder to hold your program files. Use md command.
4. navigate to the new created folder using cd command.

5. open that folder with Visual Studio Code. use the following command:

```
code .
```

where code is Visual Studio Code program and the dot (.) specify the current folder.

6. On the VS Code, In the Explorer panel and under your folder click on add file icon.



7. Write your file name followed by C++ file extension, which is cpp such as **program.cpp**
8. Start writing C++ programs.

Compiling C++ Program

Each C++ source file needs to be compiled into an object file. The object files resulting from the compilation of multiple source files are then linked into an executable file.

A C++ source file can include other files, known as header files, with the #include directive. Header files have extensions like .h, .hpp, or .hxx, or have no extension at all like in the C++ standard library and other libraries' header files (like Qt). The extension doesn't matter for the C++ preprocessor, which will literally replace the line containing the #include directive with the entire content of the included file.

After Writing your program on Visual Studio Code, save your changes. Either from **File → Save** or by pressing the Keyboard shortcut **Ctrl + S**

Open a Terminal window in Visual Studio Code from **View → Terminal** or by pressing the Keyboard shortcut **Ctrl + `** which is the same button that has the Arabic letter (ذ).

On the terminal make sure that you are in the same folder with your C++ file then execute the following command:

```
g++  filename.cpp
```

substitute **filename** with your file name such as **program.cpp**

The above command will generate an executable file called (**a.exe)**. In order to run it, write the following command:

```
./a
```

To generate an executable file with your preferred name write the command:

```
g++ program.cpp -o myname
```

substitute **myname** with your preferred name. However, to run the executable file you have to change

`./a` → `./myname`