Introduction To Programming

- Computer programming is the act of writing computer programs, which are a sequence of instructions written using a Computer Programming Language to perform a specified task by the computer.

- Computer Programming is fun and easy to learn provided you adopt a proper approach.

- Before getting into computer programming, let us first understand computer programs and what they do.
- *A computer program is a sequence of instructions written using a Computer Programming Language to perform a specified task by the computer.*
- The two important terms that we have used in the above definition are −
- Sequence of instructions
- Computer Programming Language

- To understand these terms, consider a situation when someone asks you about how to go to a nearby KFC. What exactly do you do to tell him the way to go to KFC?
- You will use Human Language to tell the way to go to KFC, something as follows −
- First go straight, after half kilometer, take left from the red light and then drive around one kilometer and you will find KFC at the right.

- Here, you have used English Language to give several steps to be taken to reach KFC. If they are followed in the following sequence, then you will reach KFC –
- 1. Go straight
- 2. Drive half kilometer
- 3. Take left
- 4. Drive around one kilometer
- 5. Search for KFC at your right side

- Now, try to map the situation with a computer program. The above sequence of instructions is actually a **Human Program** written in **English Language**, which instructs on how to reach KFC from a given starting point. This same sequence could have been given in Georgian, Spanish, Hindi, Arabic, or any other human language, provided the person seeking direction knows any of these languages.
- Now, let's go back and try to understand a computer program, which is a sequence of instructions written in a Computer Language to perform a specified task by the computer.

- Output in C++ programming language – cout << “Hello World” << endl;

- The above computer program instructs the computer to print "Hello, World!" on the computer screen.
- A computer program is also called a **computer software**, which can range from two lines to millions of lines of instructions.
- Computer program instructions are also called program source code and **computer programming** is also called **program coding**.
- A computer without a computer program is just a dump box; it is programs that make computers active.

- As we have developed so many languages to communicate among ourselves, computer scientists have developed several computer-programming languages to provide instructions to the computer (i.e., to write computer programs).

- if you understood what a **computer program** is, then we will say: *the act of writing computer programs is called computer programming.*
- As we mentioned earlier, there are hundreds of programming languages, which can be used to write computer programs and following are a few of them −
- Java
- C
- C++
- Python
- PHP
- Perl
- Ruby

Uses of Computer Programs

- Today computer programs are being used in almost every field, household, agriculture, medical, entertainment, defense, communication, etc. Listed below are a few applications of computer programs −
- MS Word, MS Excel, Adobe Photoshop, Internet Explorer, Chrome, etc., are examples of computer programs.
- Computer programs are being used to develop graphics and special effects in movie making.
- Computer programs are being used to perform Ultrasounds, X-Rays, and other medical examinations.
- Computer programs are being used in our mobile phones for SMS, Chat, and voice communication.
-

Computer Programmer

- Someone who can write computer programs or in other words, someone who can do computer programming is called a Computer Programmer.
- Based on computer programming language expertise, we can name a computer programmers as follows −
- C Programmer
- C++ Programmer
- Java Programmer
- Python Programmer
- PHP Programmer
- Perl Programmer
- Ruby Programmer

Algorithm

- Algorithm
- From programming point of view, an **algorithm** is a step-by-step procedure to resolve any problem. An algorithm is an effective method expressed as a finite set of well-defined instructions.

- Thus, a computer programmer lists down all the steps required to resolve a problem before writing the actual code. Following is a simple example of an algorithm to find out the largest number from a given list of numbers −

- 1. Get a list of numbers L_1, L_2, L_3....L_N
- 2. Assume L_1 is the largest, Largest = L_1
- 3. Take next number L_i from the list and do the following
- 4. If Largest is less than L_i
- 5. Largest = L_i
- 6. If L_i is last number from the list then
- 7. Print value stored in Largest and come out 8. Else repeat same process starting from step 3

- The above algorithm has been written in a crude way to help beginners understand the concept. You will come across more standardized ways of writing computer algorithms as you move on to advanced levels of computer programming.

- We assume you are well aware of English Language, which is a well-known **Human Interface Language**. English has a predefined grammar, which needs to be followed to write English statements in a correct way. Likewise, most of the Human Interface Languages (Hindi, English, Spanish, French, etc.) are made of several elements like verbs, nouns, adjectives, adverbs, propositions, and conjunctions, etc.

- Similar to Human Interface Languages, Computer Programming Languages are also made of several elements. We will take you through the basics of those elements and make you comfortable to use them in various programming languages. These basic elements include −
-

- Programming Environment
- Basic Syntax
- Data Types
- Variables
- Keywords
- Basic Operators
- Decision Making
- Loops
- Numbers
- Characters
- Arrays
- Strings
- Functions
- File I/O

- We will explain all these elements in subsequent chapters with examples using different programming languages. First, we will try to understand the meaning of all these terms in general and then, we will see how these terms can be used in C++ programming languages.

Environment

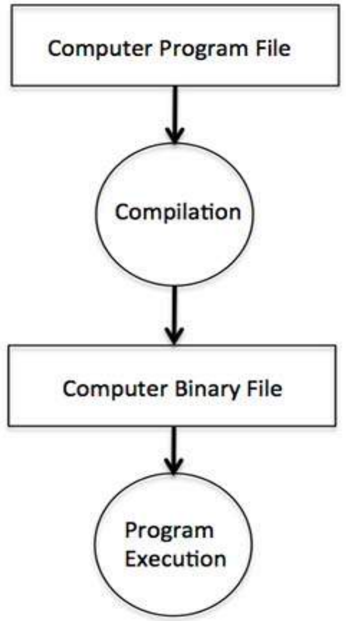
- Though Environment Setup is not an element of any Programming Language, it is the first step to be followed before setting on to write a program.

Compiler

- You write your computer program using your favorite programming language and save it in a text file called the program file.
- Now let us try to get a little more detail on how the computer understands a program written by you using a programming language. Actually, the computer cannot understand your program directly given in the text format, so we need to convert this program in a binary format, which can be understood by the computer.

- The conversion from text program to binary file is done by another software called Compiler and this process of conversion from text formatted program to binary format file is called program compilation. Finally, you can execute binary file to perform the programmed task.

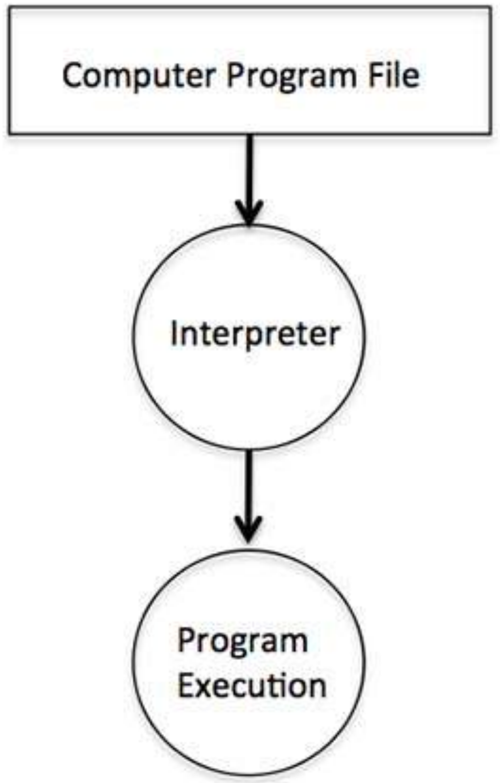
- We are not going into the details of a compiler and the different phases of compilation.
- The following flow diagram gives an illustration of the process



- So, if you are going to write your program in any such language, which needs compilation like C, C++, Java and C#, etc., then you will need to install their compilers before you start programming.

Interpreter

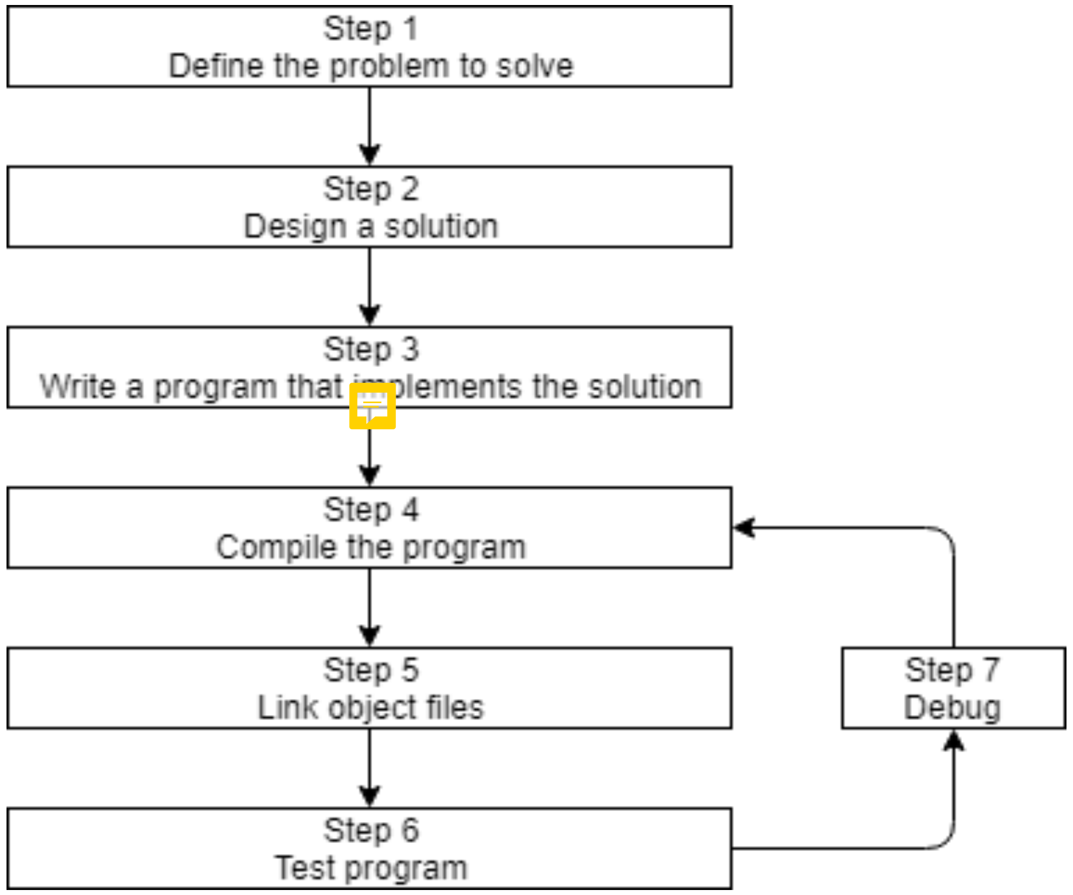
- We just discussed about compilers and the compilation process. Compilers are required in case you are going to write your program in a programming language that needs to be compiled into binary format before its execution.
- There are other programming languages such as Python, PHP, and Perl, which do not need any compilation into binary format, rather an interpreter can be used to read such programs line by line and execute them directly without any further conversion.



So, if you are going to write your programs in PHP, Python, Perl, Ruby, etc., then you will need to install their interpreters before you start programming.

- Let’s start with a little coding, which will really make you a computer programmer. We are going to write a single-line computer program to write **Hello, World!** on your screen. Let’s see how it can be written using C++ programming languages.

```
#include <iostream>
int main() {
    cout << “Hello, World” << endl;
}
```


Step 1
Define the problem to solve
Step 2
Design a solution
Step 3
Write a program that implements the solution
Step 4
Compile the program
Step 5
Link object files
Step 6
Test program
Step 7
Debug

- **Step 1: Define the problem that you would like to solve**
- This is the “what” step, where you figure out what problem you are intending to solve. Coming up with the initial idea for what you would like to program can be the easiest step, or the hardest. But conceptually, it is the simplest. All you need is an idea that can be well defined, and you’re ready for the next step.
- Here are a few examples:
- “I want to write a program that will allow me to enter many numbers, then calculates the average.”
- “I want to write a program that generates a 2d maze and lets the user navigate through it. The user wins if they reach the end.”
- “I want to write a program that reads in a file of stock prices and predicts whether the stock will go up or down.”

- **Step 2: Determine how you are going to solve the problem**
- This is the “how” step, where you determine how you are going to solve the problem you came up with in step 1. It is also the step that is most neglected in software development. The crux of the issue is that there are many ways to solve a problem -- however, some of these solutions are good and some of them are bad. Too often, a programmer will get an idea, sit down, and immediately start coding a solution. This often generates a solution that falls into the bad category.
- Typically, good solutions have the following characteristics:
- They are straightforward (not overly complicated or confusing).
- They are well documented (especially around any assumptions being made or limitations).
- They are built modularly, so parts can be reused or changed later without impacting other parts of the program.
- They are robust, and can recover or give useful error messages when something unexpected happens.

- When you sit down and start coding right away, you’re typically thinking “I want to do <something>”, so you implement the solution that gets you there the fastest. This can lead to programs that are fragile, hard to change or extend later, or have lots of **bugs** (technical defects).

- **Step 3: Write the program**
- In order to write the program, we need two things: First, we need knowledge of a programming language -- that’s what these tutorials are for! Second, we need an editor. It’s possible to write a program using any editor you want, even something as simple as Window’s notepad or Unix’s vi or pico. However, we strongly urge you to use an editor that is designed for coding. Don’t worry if you don’t have one yet. We’ll cover how to install a code editor shortly.

- **Step 4: Compiling your source code**
- In order to compile a C++ program, we use a C++ compiler. The C++ compiler sequentially goes through each source code (.cpp) file in your program and does two important tasks:
- First, it checks your code to make sure it follows the rules of the C++ language. If it does not, the compiler will give you an error (and the corresponding line number) to help pinpoint what needs fixing. The compilation process will also be aborted until the error is fixed.
- Second, it translates your C++ source code into a machine language file called an **object file**. Object files are typically named *name.o* or *name.obj*, where *name* is the same name as the .cpp file it was produced from.
- If your program had 3 .cpp files, the compiler would generate 3 object files:

- **Step 5: Linking object files and libraries**
- After the compiler creates one or more object files, then another program called the **linker** kicks in. The job of the linker is three fold:
- First, to take all the object files generated by the compiler and combine them into a single executable program.

- **Steps 6 & 7: Testing and Debugging**
- This is the fun part (hopefully)! You are able to run your executable and see whether it produces the output you were expecting!
- If your program runs but doesn’t work correctly, then it’s time for some debugging to figure out what’s wrong. We will discuss how to test your programs and how to debug them in more detail soon.