University of Colombo School of Computing

Orientation Program – 2022 Batch

Basics of Programming & Logical Thinking

***Outline*:**

- *Basics of Programming*
  - *What is Computer Programming*
  - *Basic elements of programming*
    - *Programming Environment*
    - *Keywords*
    - *Data Types*
    - *Variables*
    - *Operators*
    - *If else conditions*
    - *Loops*
- *Logical Thinking*

What is Computer Programming?

Humans use their native language or another human language to communicate with each other. This communication can be of written mode or spoken mode. But these general human languages are not suitable for a human to communicate with a computer since the computers cannot understand these languages. They need a special language to communicate with humans and get instructions from humans. Programming can be mentioned as this language humans use **to give instructions to the computers**.

Computer programming is defined as a process of developing and implementing various set of instructions given to the computer to perform a certain predefined task. Computer Programming is easy if it is appropriately managed. There are many computer programming languages available so finalizing the right language is not an easy task. Some examples for popular programming languages are Java, Python, C, C++, C# and PHP.

Same as any other Human Languages are made of nouns, adjectives, adverbs, propositions, and conjunctions, etc programming languages are also made of different elements. For example, similar to human languages, programming languages also follow grammar called syntax.

Basic Elements of Programming

Most important basic elements for programming languages are:

- Programming Environment
- Keywords
- Data Types
- Variables
- Operators
- If else conditions
- Loops
- Numbers, Characters and Arrays
- Functions
- Input and Output Operations

Programming Environment

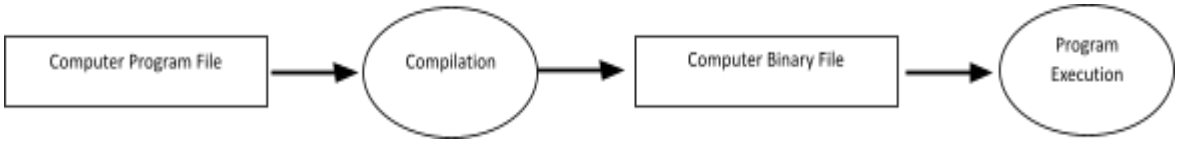
Programming environment is the base on top where we do our programming. So, the software that we need to do programming can be identified as the programming environment.

In programming using any computer language, the following are the basic requirements.

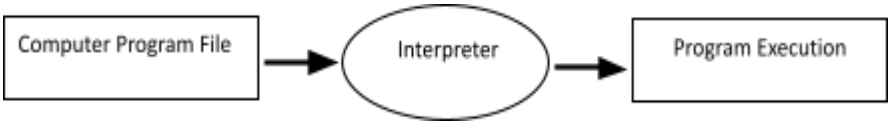
- Text Editor – This is a software where we can write the programs and convert to any format based on the programming language we are using. The most basic text editor in the Windows environment is Notepad. However text editors are not directly related to programming; in fact, they are designed to work with framework or language of your choosing. An IDE, which stands for Integrated Development Environment, is something a bit more powerful than text editors and it is intended as a set of tools that all work together: text editor, compiler, build or make integration, debugging, etc. Unlike text editors, IDE provide advanced features such as methods suggestions, color differentiation, inbuilt Compiler etc. You can read more about the difference between a text editor and an IDE from the following link.

  Difference between Text editor and IDE

- A compiler – This is where the computer programs are converted to the binary format. Computers cannot understand the general languages we are using. Hence, they need to be converted to the general format the computer can understand, which is binary format.



- An interpreter – An interpreter and a Compiler has alternative tasks when referring to the execution of the program. However, an interpreter is needed in places where it is not required to convert to binary format, but instead, **execute the program code line by line**. Some examples of programing languages that need interpreters except compilers are; Python, PHP, Perl etc.



Basic Syntax

```
#include <stdio.h>

int main() {

  /* printf() function to write Hello, World! */

  printf( "Hello, World!" );

}
```

This code component shows how we can display the message '*Hello, World!*' using the C programming language.

In explaining the code component, **#include <stdio.h>** statement is a compulsory component in C programming statements.

- Every C program starts with main(), which is called the main function, and followed by a left curly brace. The rest of the program instruction is written in between and finally a right curly brace ends the program. Hence the code lines in between the curly braces are called the **program body**.

- **Functions** are small units of programs and they are used to carry out a specific task. For example, the above program makes use of two functions: **main()** and **printf().** Here, the function **main()** provides the entry point for the program execution and the other function **printf()** is being used to print an information on the computer screen.

- A C program can have statements enclosed inside **/*.....*/.** Such statements are called **comments** and these comments are used to make the programs user friendly and easy to understand. The good thing about comments is that they are completely ignored by compilers and interpreters. Therefore, comments can also be used to randomly stop the function of a certain code component.

- Another specific rule in C Programming is that, every individual statement in a C Program must be ended with a **semicolon (;).**

- If you do not follow the rules defined by the programing language, then at the time of compilation, you will get **syntax errors** and the program will not be compiled. When speaking of syntax errors, even the missing of a single semicolon or dot may give a syntax error according to the programming language used. You can refer to the following link to get to know more about syntax errors.
  https://woz-u.com/blog/common-programming-syntax-errors-and-how-to-fix-them/amp/

This code component gives the code lines print the statement 'Hello, World!' in Java programming language.

```
public class HelloWorld {
  public static void main(String []args) {
    /* println() function to write Hello, World! */
    System.out.println("Hello, World!");
  }
}
```

```
# print function to write Hello, World! */

print "Hello, World!"
```

This code component gives the code lines print the statement 'Hello, World!' in Python programming language. In Python program, we are directly executing it and it does not need an intermediate step called compilation. As well, Python does not require a semicolon (;) to terminate a statement, rather a new line always means termination of the statement.

Data Types

As the name itself suggests, Data Types refer to the type of data that is relevant to the data we are using in computer programming. Consider the following cases;

Program	Data	Data type
Program 1 – Addition of two numbers	14 + 53	Numbers
Program 2 – Addition of two numbers with decimal points	24.50 + 89.75	Decimal Numbers
Program 3 – Displaying the name of a thing / person	UCSC	Letters
Program 4 – Display the Student Registration number	2019 / IS / 001	Letters, Numbers and Characters
Program 5 – Displaying the date	2020-01-31	Date

In programming, these different types of data are treated differently. For example, in the above cases, the way the data of Program 2 is executed is different from the way the data of Program 1 is executed, depending on the data types. The inner processing changes accordingly. Therefore, it is important to define and specify the type of data at the beginning of the program itself. The names given to these data types in different programming languages are different.

You can read about the C programming data types from the following link.

https://www.programiz.com/c-programming/c-data-types

Variables

A variable is simply a way to store some sort of information for later use, and we can retrieve this information by referring to a "name" that will describe this information.

Creating variables is named as *Declaring Variables*. Different programming languages have different ways of declaring variables inside a program.

```
#include <stdio.h>

int main() {

  int a;

  int b;

}
```

This code component is an example of declaring a variable in C language. It creates two variables to reserve two memory locations with names **a** and **b**. We created these variables using **int** keyword to specify variable data type which means we want to store integer values in these two variables. We can store different variables with different data types and variable names.

```
#include <stdio.h>

int main() {

  int a, b;

}
```

This code component explains an instance where different variables with the same data type can be stored using a single statement, separated by commas.

- A variable name can hold only a single type of value. For example, if variable **a** has been defined as **int** type, then it can store only integer values.

- You can use the same variable name only once inside the program. For example, if a variable **a** has been defined to store an integer value, then you cannot define **a** again to store any other type of value

```
#include <stdio.h>
int main() {
  int a;
  int b;
  a = 10;
  b = 20;
}
```

This code component explains how we can store values in variables after they have been declared. This program stores **10** in variable **a** and **20** is being stored in variable **b**. Almost all the programming languages have similar way of storing values in variable where we keep variable name in the left hand side of an equal sign "=" and whatever value we want to store in the variable, we keep that value in the right hand side.

Once the variables are declared and values are stored in them, they can be accessed by methods specific to the programming language.

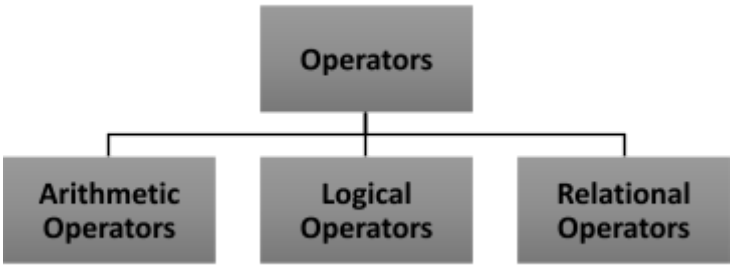
Keywords

Reserved keywords are a part of basic programming syntax. Different programming languages provide different sets of reserved keywords, but there is one important and common rule in all the programming languages that we cannot use a reserved keyword to name our variables, which means we cannot name our variable like int or float, rather these keywords can only be used to specify a variable data type. However, reserved keywords are not only data types. There are more keywords.

For example, if you try to use any reserved keyword for the purpose of variable name, then you will get a syntax error.

Operators

An operator in a programming language is a symbol that tells the compiler or interpreter to perform specific mathematical, relational or logical operations and produce final results. There are three types of operators in programming languages.



Arithmetic Operators:

Computer programs are widely used for mathematical calculations. We can write a computer program which can do simple calculation like adding two numbers (2 + 3) and we can also write a program, which can solve a complex equation like $P(x) = x^4 + 7x^3 - 5x + 9$.

In the first equation, 2 and 3 are called operands and + is called an Operator. Similar concepts exist in computer programming as well. The following are some common arithmetic operators used in language C.

Operator	Purpose
+	Addition
-	Subtraction
*	Multiplication
/	Division

Relational Operators:

Relational operators are usually used for a comparison of two operands and finally they will produce a Boolean result of **True** or **False.** Some of them are listed in the following table.

Assume A= 20 and B=10,

Operator	Description	Example
= =	Checks if two operands are equal	A==B Returns False
! =	Checks if two operands are not equal	A! = B Returns True
>	Checks if left operand is greater than the right operand	A>B Returns True
<	Checks if left operand is lesser than right operand	A<B Returns False
>=	Checks if left operand is greater than **or** equal to the right operand	A>=B Returns True
< =	Checks if left operand is lesser than **or** equal to the right operand	A<=B Returns False

Logical Operators:

Logical Operators help us take decisions based on certain conditions. Hope you know what are logical AND, logical OR and logical NOT by now. These three operators are used in logical operators. You can read about more logical operators from the following link.

https://whatis.techtarget.com/definition/logic-gate-AND-OR-XOR-NOT-NAND-NOR-and-XNOR

For the explanation of logical operators, assume A=1 and B=0;

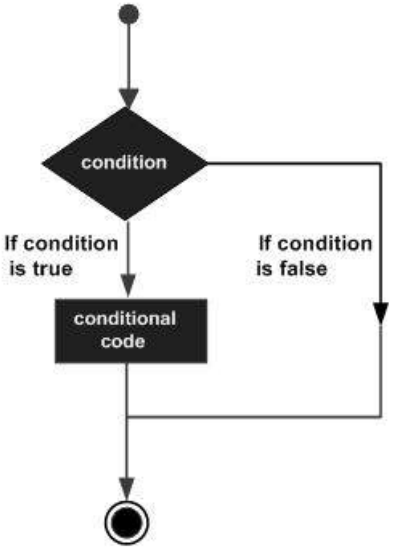
Operator	Explanation	Example
&&	Logical AND	A&&B Returns False
||	Logical OR	A||B Returns True
!	Logical NOT	!(A&&B) Returns True

Decisions by If Else Statements

There will be many cases in computer programming when you will be given two or more options and you will have to select an option based on the given conditions.

Consider the following scenario.

For example, assume “x” and “y” are two integer numbers. We need to check which number is the smallest and which number is the largest. An If condition will be used to compare the values of “x” and “y” to identify the smallest number. This can be represented by the flow chart below.

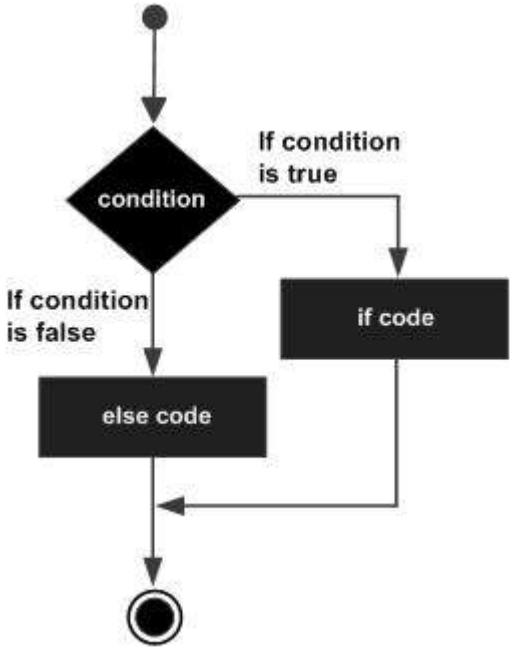


When it is programmed in C, the following is the code. Since the value of “x” is smaller than”y”, the If condition will evaluate to true.

```
#include <stdio.h>
int main() {
  int x = 15;
  int y = 40;
  if( x < y) {
    printf( "x is smaller than y");
  }
}
```

If..Else Statements

An If statement can be followed by an optional else statement, which executes when the output of the first if condition is false. It can be represented in the following flow chart.



For example, assume "x" is a mark given for a student. If the mark is greater than 55, then the *student is brilliant*, otherwise *student is not brilliant*. This can be programmed in C as follows.

```
#include <stdio.h>
int main() {
  int x = 45;
  if( x > 55) {
    printf( "Student is brilliant\n");
  } else {
    printf( "Student is not brilliant\n");
  }
}
```

If..elseif..else Statements

Assume “a” and “b” are two integer numbers. If value is x is greater than y, “x is the maximum”. Else if, value of y is greater than x, then “y is the maximum”. Otherwise the two numbers are equal.

```
#include <stdio.h>
int main() {
  int x = 45;
  int y = 55;
  if( x > y) {
    printf( "x is the maximum\n");
  }
  else if( y > x) {
    printf( "y is the maximum\\n");
  }
  else {
    printf( "x and y are equal\n");
  }
}
```

Loops

Almost all the programming languages provide a concept called loop, which helps in executing one or more statements up to a desired number of times. All high-level programming languages provide various forms of loops, which can be used to execute one or more statements repeatedly.

Let's consider a situation when you want to print Hello, World! Five times. It is simple, but again, let's consider another situation when you want to write Hello, World! a thousand times. We can certainly not write printf() statements a thousand times. That is why the Loops in programming languages have been introduced. The C language code for the above scenario is as follows.

```
#include <stdio.h>
int main() {
  int i = 0;
  while ( i < 5 ) {
    printf( "Hello, World!\n");
    i = i + 1;
  }
}
```

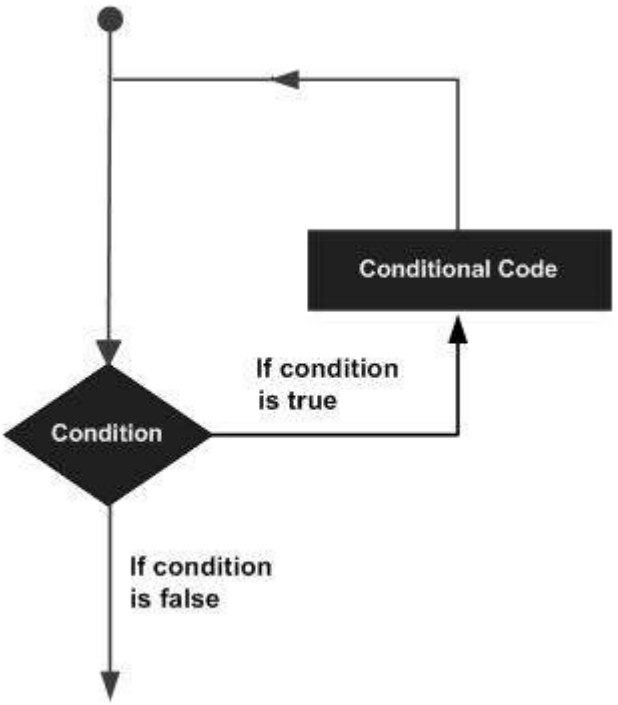
The above program makes use of a while loop, which is being used to execute a set of programming statements enclosed within {....}. Here, the computer first checks whether the given condition, i.e., variable "i" is less than 5 or not and if it finds the condition is true, then the loop

body is entered to execute the given statements. Here, we have the following two statements in the loop body −

- First statement is printf() function, which prints Hello World!
- Second statement is i = i + 1, which is used to increase the value of variable i

After executing all the statements given in the loop body, the computer goes back to while(i < 5) and the given condition, (i < 5), is checked again, and the loop is executed again if the condition holds true. This process repeats till the given condition remains true which means variable "i" has a value less than 5.

To be clearer, the following is the flow chart for a loop process.



There are several types of loops in programming languages as follows.

- The For loop
- The While loop
- The Do While loop

Exercises

With the help of the knowledge you gained by if else statements and different loops, try generating the following patterns.