What is Programming

What is Programming

A computer is composed of many devices:

Storage device (RAM, HDD)

Processing device(CPU)

Display device(MONITOR)

why do we need computer programs?

If we only had devices without human intervention, there would be no way for them to know how to perform tasks. For example, if we wanted to add numbers together, we'd need to instruct the CPU on how to do it. These instructions must be communicated in a language the CPU understands, called a programming language. The instructions we write using this language are known as computer programs or code. So, programming languages are like the means by which we tell the CPU what tasks to carry out, such as adding numbers.

Computer programs are essential because they provide precise instructions to devices, such as the CPU, on how to accomplish tasks efficiently. Consider the task of sorting a collection of numbers. While it's feasible for humans to sort small collections manually, it becomes impractical for larger datasets containing millions or billions of numbers. This is where computer programs shine.

For instance, let's take the collection of numbers:

A = [7, 9, 1, 3, 5, 2]

To sort these numbers, we need to provide structured instructions to the computer. One way to do this is by creating an empty list, say B, where we'll arrange the numbers in ascending order:

B = [ ]

The instruction for sorting can be expressed as follows: "Continuously select the smallest number from A and append it to B until A is empty."

In simpler terms, we instruct the computer to repeatedly find the smallest number in A and move it to B until A is devoid of elements. This systematic approach ensures that the numbers in B are arranged in ascending order, thus achieving the desired outcome of sorting the collection.

Running a Program

What is a Compiler?

A compiler is a special type of software that translates programs written in high-level programming languages into machine code or executable code that a computer can understand and execute directly.

The compiler takes the entire program written in a high-level language (like C, C++, Java, etc.) and translates it into machine code, which is a low-level binary language understood by the computer's hardware.

What do we mean by “running a program” ?

To "run a program" means to execute the instructions written in a computer program using a computer's CPU. When you run a program, you're essentially instructing the computer to interpret and execute the commands and logic outlined in the program's code.

Consider the following simple C program that prints "Hello, World!" to the console:

```
#include <stdio.h>

int main() {

 printf("Hello, World!\n");

 return 0;

}
```

Here's what happens when you run this program:

Compilation: Before running the program, it needs to be translated into machine code that the computer can understand. This process is called compilation. You use a C compiler (like GCC or Clang) to compile the C source code into an executable file.
Execution: Once the program is compiled, you can execute it. To do so, you typically double-click the executable file or use a command-line interface to run it. When you initiate the execution, the operating system loads the program into memory and hands control over to the CPU.

Understanding a small Program

Understanding a small program

Let us write a small piece of code:

```
#include <stdio.h>

int main() {

 printf("Hello, World!\n");

 return 0;

}
```

Let's break down each line of the C program:

#include <stdio.h> :

#include: This is a preprocessor directive in C. Preprocessor directives are commands to the compiler that are executed before the compilation process begins. The #include directive tells the compiler to include the contents of a specified header file in the current source file.

<stdio.h>: This part specifies the name of the header file to be included. In this case, stdio.h stands for "standard input-output header.

This line is a preprocessor directive. It tells the compiler to include the contents of the standard input-output library (stdio.h) in the program. This library contains functions like printf() and scanf() for input and output operations.

int main() :

This line defines the main function, which is the entry point of every C program. The int before main indicates that the function returns an integer value to the operating system, typically used to indicate the status of program execution (0 for success, non-zero for failure).

printf("Hello, World!\n");

This line is a function call to printf(), which is used to print formatted output to the standard output (usually the console or terminal). In this case, it prints the string "Hello, World!\n", where \n represents a newline character that moves the cursor to the next line after printing.

return 0;

This line terminates the main function and returns the value 0 to the operating system, indicating the successful execution of the program. It's a common convention in C programs to use return 0; at the end of the main to signify that the program was completed without errors.

Orientation