PROGRAMMING IN C

Dr.P.Anuradha, Assistant Professor, Department of Physics.

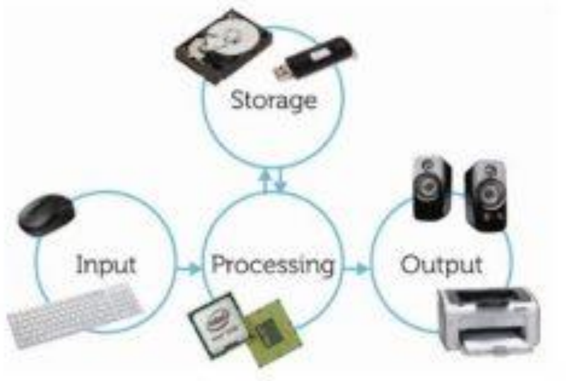
UNIT II

Components of a Computer System

Every computer system has the following three basic components:

1. Input unit
2. Central processing unit
3. Output unit

Elements of a computer system



While there are other components as well, these three are primarily responsible for making a computer function. They must work in complete synergy because that will ensure smooth overall functioning. Hence, we can even call them building blocks of a computer system.

Input Unit

These components help users enter data and commands into a computer system. Data can be in the form of numbers, words, actions, commands, etc. The main function of input devices is to direct commands and data into computers. Computers then use their CPU to process this data and produce output.

For example, a laptop's keyboard is an input unit that enters numbers and characters. Similarly, even a mouse can be an input unit for entering directions and commands. Other examples include barcode readers, Magnetic Ink Character Readers (MICR), Optical Character Readers (OCR), etc.

Another example of input devices is touch-screens. Users can simply touch these screens without using any other device to enter commands. From smartphones to ATM machines, these input devices are becoming very popular these days.

Central Processing Unit (CPU)

After receiving data and commands from users, a computer system now has to process it according to the instructions provided. Here, it has to rely on a component called the central processing unit. The CPU further uses these three elements:

a) Memory Unit

Once a user enters data using input devices, the computer system stores this data in its memory unit. This data will now remain here until other components of CPU process it. The memory unit uses a set of pre-programmed instructions to further transmit this data to other parts of the CPU.

b) Arithmetic and Logic Unit

This part of the CPU performs arithmetic operations. It does basic mathematical calculations like addition, subtraction, division, multiplication, etc. Further, it can even perform logical functions like the comparison of data.

c) Control Unit

This unit is the backbone of computers. It is responsible for coordinating tasks between all components of a computer system. The control unit collects data from input units and sends it to processing units depending on its nature. Finally, it also further transmits processed data to output units for users.

Output Unit

The third and final component of a computer system is the output unit. After processing of data, it is converted into a format which humans can understand. After conversion, the output units displays this data to users. Examples of output devices include monitors, screens, printers and speakers. Thus, output units basically reproduce the data formatted by the computer for users' benefit.

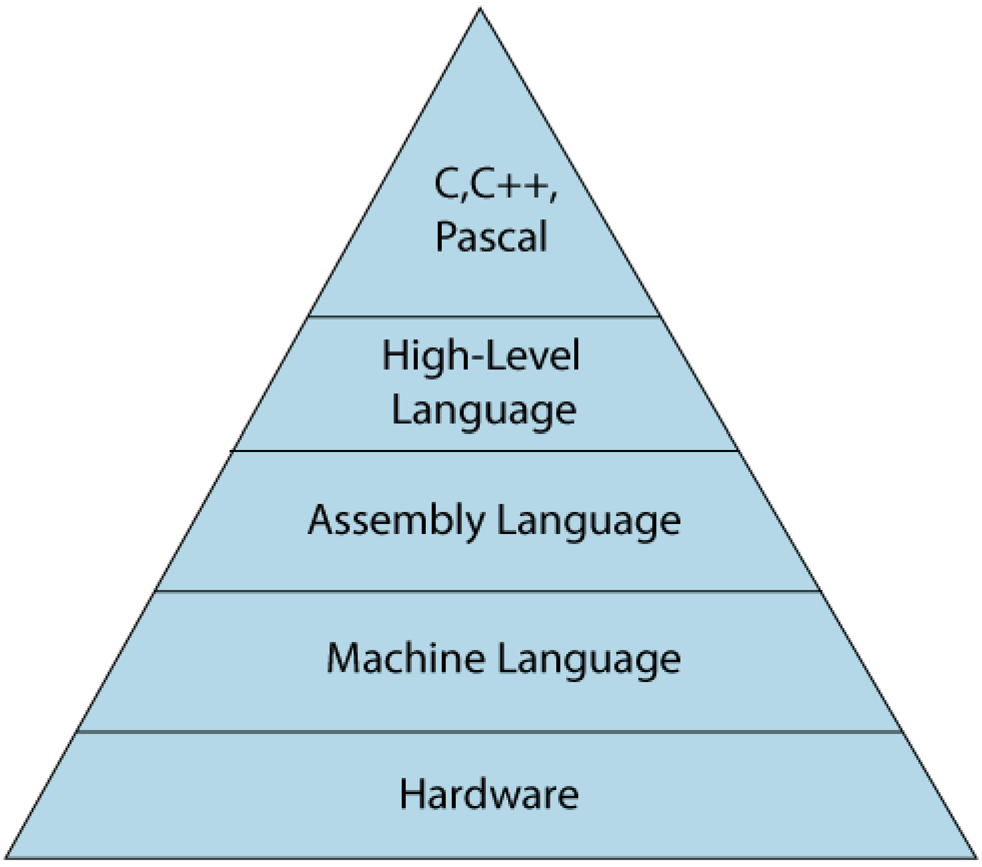
Programming languages

A programming language defines a set of instructions that are compiled together to perform a specific task by the CPU (Central Processing Unit). The programming language mainly refers to high-level languages such as C, C++, Pascal, Ada, COBOL, etc.

Each programming language contains a unique set of keywords and syntax, which are used to create a set of instructions. Thousands of programming languages have been developed till now, but each language has its specific purpose. These languages vary in the level of abstraction they provide from the hardware. Some programming languages provide less or no abstraction while some provide higher abstraction. Based on the levels of abstraction, they can be classified into two categories:

- Low-level language
- High-level language

The image which is given below describes the abstraction level from hardware. As we can observe from the below image that the machine language provides no abstraction, assembly language provides less abstraction whereas high-level language provides a higher level of abstraction.



Low-level language

The low-level language is a programming language that provides no abstraction from the hardware, and it is represented in 0 or 1 forms, which are the machine instructions. The languages that come under this category are the Machine level language and Assembly language.

Machine-level language

The machine-level language is a language that consists of a set of instructions that are in the binary form 0 or 1. As we know that computers can understand only machine instructions, which are in binary digits, i.e., 0 and 1, so the instructions given to the computer can be only in binary codes. Creating a program in a machine-level language is a very difficult task as it is not easy for the programmers to write the program in machine instructions. It is error-prone as it is not easy to understand, and its maintenance is also very high. A machine-level language is not portable as each computer has its machine instructions, so if we write a program in one computer will no longer be valid in another computer.

The different processor architectures use different machine codes, for example, a PowerPC processor contains RISC architecture, which requires different code than intel x86 processor, which has a CISC architecture.

Assembly Language

The assembly language contains some human-readable commands such as mov, add, sub, etc. The problems which we were facing in machine-level language are reduced to some extent by using an extended form of machine-level language known as assembly language. Since assembly language instructions are written in English words like mov, add, sub, so it is easier to write and understand.

As we know that computers can only understand the machine-level instructions, so we require a translator that converts the assembly code into machine code. The translator used for translating the code is known as an assembler.

The assembly language code is not portable because the data is stored in computer registers, and the computer has to know the different sets of registers.

The assembly code is not faster than machine code because the assembly language comes above the machine language in the hierarchy, so it means that assembly language has some abstraction from the hardware while machine language has zero abstraction.

Differences between Machine-Level language and Assembly language

High-Level Language

The high-level language is a programming language that allows a programmer to write the programs which are independent of a particular type of computer. The high-level languages are considered as high-level because they are closer to human languages than machine-level languages.

When writing a program in a high-level language, then the whole attention needs to be paid to the logic of the problem.

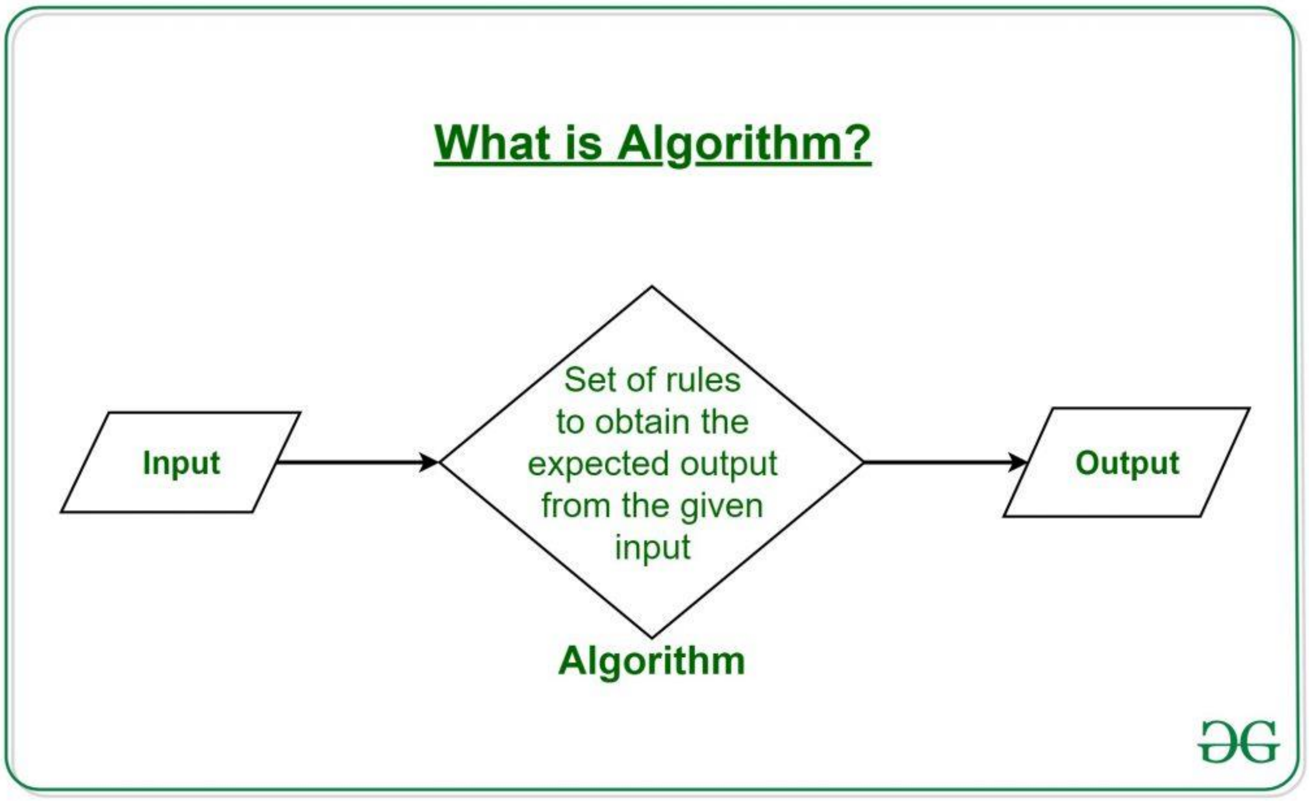
A compiler is required to translate a high-level language into a low-level language.

Advantages of a high-level language

- The high-level language is easy to read, write, and maintain as it is written in English like words.
- The high-level languages are designed to overcome the limitation of low-level language, i.e., portability. The high-level language is portable; i.e., these languages are machine-independent.

ALGORITHMS

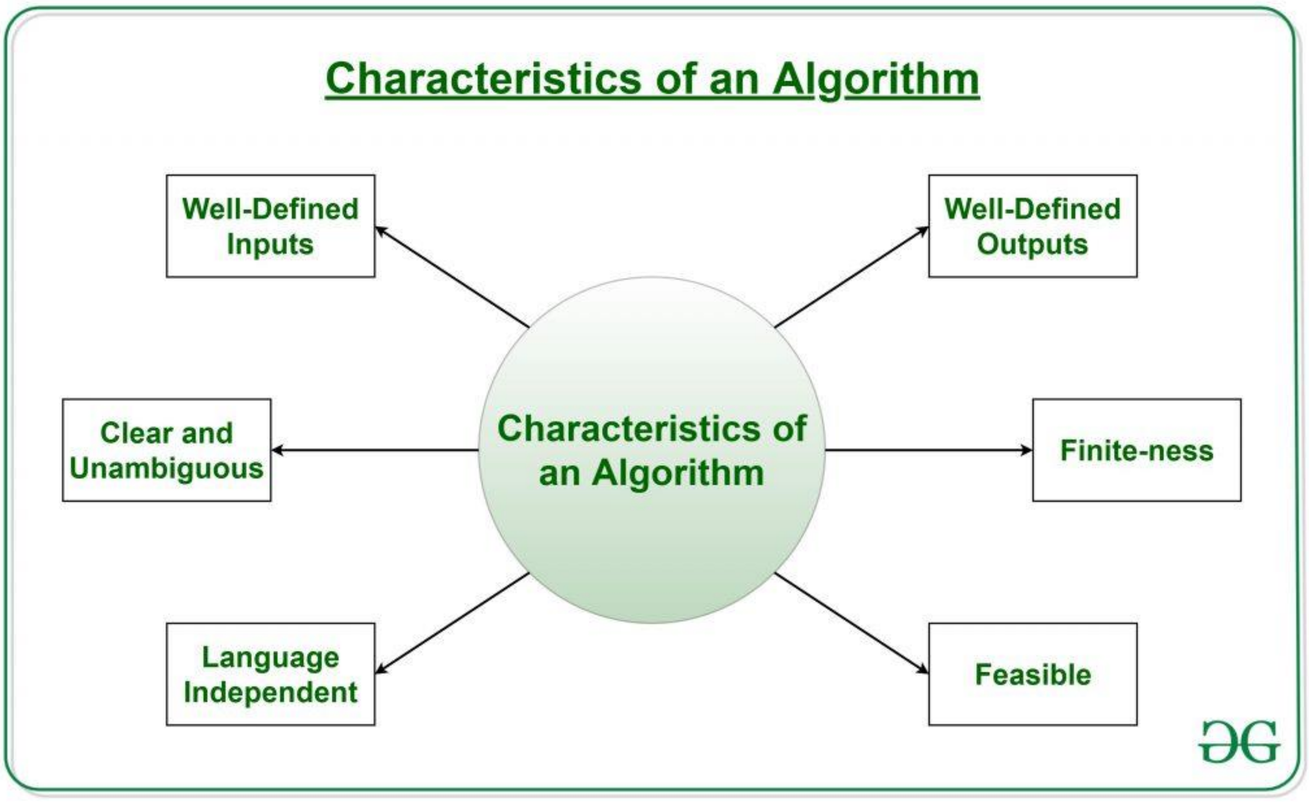
The word Algorithm means “a process or set of rules to be followed in calculations or other problem-solving operations”. Therefore Algorithm refers to a set of rules/instructions that step-by-step define how a work is to be executed upon in order to get the expected results.


What is Algorithm?
Input
Set of rules to obtain the expected output from the given input
Output
Algorithm

It can be understood by taking an example of cooking a new recipe. To cook a new recipe, one reads the instructions and steps and execute them one by one, in the given sequence. The result thus obtained is the new dish cooked perfectly. Similarly, algorithms help to do a task in programming to get the expected output.

The Algorithm designed are language-independent, i.e. they are just plain instructions that can be implemented in any language, and yet the output will be the same, as expected.

What are the Characteristics of an Algorithm?



As one would not follow any written instructions to cook the recipe, but only the standard one. Similarly, not all written instructions for programming is an algorithm. In order for some instructions to be an algorithm, it must have the following characteristics:

- **Clear and Unambiguous**: Algorithm should be clear and unambiguous. Each of its steps should be clear in all aspects and must lead to only one meaning.
- **Well-Defined Inputs**: If an algorithm says to take inputs, it should be well-defined inputs.
- **Well-Defined Outputs:** The algorithm must clearly define what output will be yielded and it should be well-defined as well.
- **Finite-ness:** The algorithm must be finite, i.e. it should not end up in an infinite loops or similar.

- **Feasible:** The algorithm must be simple, generic and practical, such that it can be executed upon will the available resources. It must not contain some future technology, or anything.
- **Language Independent:** The Algorithm designed must be language-independent, i.e. it must be just plain instructions that can be implemented in any language, and yet the output will be same, as expected.

How to Design an Algorithm?

Inorder to write an algorithm, following things are needed as a pre-requisite:

1. The **problem** that is to be solved by this algorithm.
2. The **constraints** of the problem that must be considered while solving the problem.
3. The **input** to be taken to solve the problem.
4. The **output** to be expected when the problem the is solved.
5. The **solution** to this problem, in the given constraints.

Then the algorithm is written with the help of above parameters such that it solves the problem.

Example: Consider the example to add three numbers and print the sum.

- **Step 1: Fulfilling the pre-requisites**
  As discussed above, in order to write an algorithm, its pre-requisites must be fulfilled.
  1. **The problem that is to be solved by this algorithm**: Add 3 numbers and print their sum.
  2. **The constraints of the problem that must be considered while solving the problem**: The numbers must contain only digits and no other characters.
  3. **The input to be taken to solve the problem:** The three numbers to be added.
  4. **The output to be expected when the problem the is solved:** The sum of the three numbers taken as the input.

5. **The solution to this problem, in the given constraints:** The solution consists of adding the 3 numbers. It can be done with the help of '+' operator, or bit-wise, or any other method.

- **Step 2: Designing the algorithm**

  Now let's design the algorithm with the help of above pre-requisites:

  Algorithm to add 3 numbers and print their sum:

  1. START
  2. Declare 3 integer variables num1, num2 and num3.
  3. Take the three numbers, to be added, as inputs in variables num1, num2, and num3 respectively.
  4. Declare an integer variable sum to store the resultant sum of the 3 numbers.
  5. Add the 3 numbers and store the result in the variable sum.
  6. Print the value of variable sum
  7. END

- **Step 3: Testing the algorithm by implementing it.**

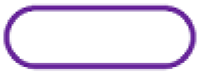
  Inorder to test the algorithm,

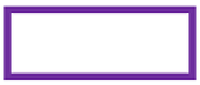
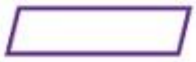
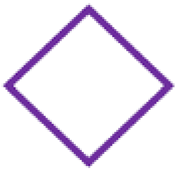
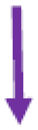
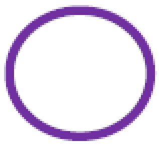
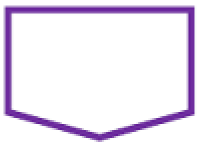
FLOW CHARTS

Flowchart is a diagrammatic representation of sequence of logical steps of a program. Flowcharts use simple geometric shapes to depict processes and arrows to show relationships and process/data flow.

Flowchart Symbols

Here is a chart for some of the common symbols used in drawing flowcharts.

Symbol	Symbol Name	Purpose
	Start/Stop	Used at the beginning and end of the algorithm to show start and end of

		the program.
	Process	Indicates processes like mathematical operations.
	Input/ Output	Used for denoting program inputs and outputs.
	Decision	Stands for decision statements in a program, where answer is usually Yes or No.
	Arrow	Shows relationships between different shapes.
	On-page Connector	Connects two or more parts of a flowchart, which are on the same page.
	Off-page Connector	Connects two parts of a flowchart which are spread over different pages.

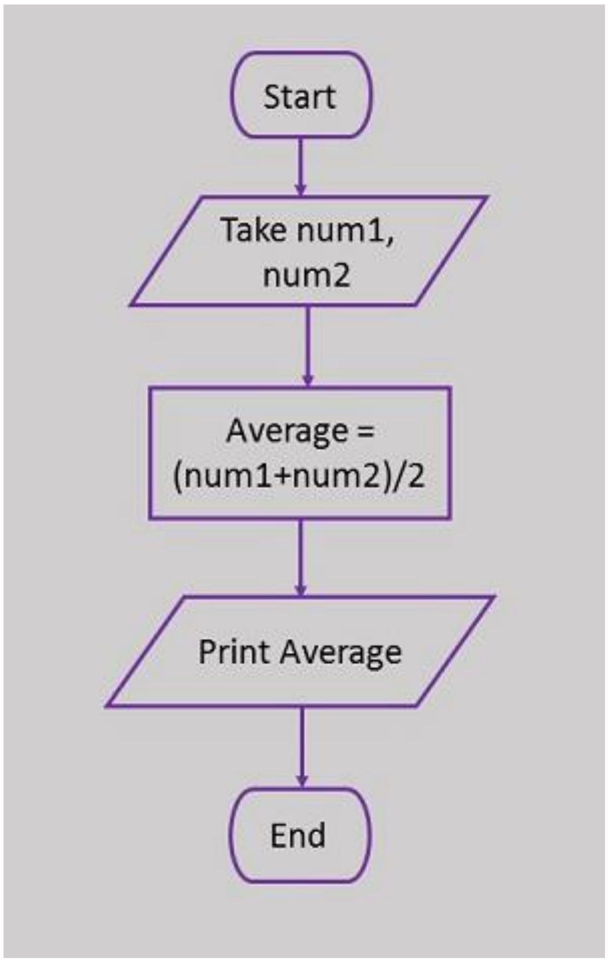
Guidelines for Developing Flowcharts

These are some points to keep in mind while developing a flowchart −

- Flowchart can have only one start and one stop symbol
- On-page connectors are referenced using numbers
- Off-page connectors are referenced using alphabets

- General flow of processes is top to bottom or left to right
- Arrows should not cross each other

EXAMPLE



Operating System (OS)

An operating system (OS) is a collection of software that manages computer hardware resources and provides common services for computer programs. The operating system is a vital component of the system software in a computer system.. An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

Some popular Operating Systems include Linux Operating System, Windows Operating System, VMS, OS/400, AIX, z/OS, etc.

Following are some of important functions of an operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Applications of Operating System

Following are some of the important activities that an Operating System performs −

- **Security** − By means of password and similar other techniques, it prevents unauthorized access to programs and data.
- **Control over system performance** − Recording delays between request for a service and response from the system.
- **Job accounting** − Keeping track of time and resources used by various jobs and users.
- **Error detecting aids** − Production of dumps, traces, error messages, and other debugging and error detecting aids.
- **Coordination between other softwares and users** − Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

.

REFERENCES

1. **https://www.toppr.com/guides/accountancy/application-of-computers-in-accounting/components-computer-system/**

2. **https://www.javatpoint.com/classification-of-programming-languages**

3. **https://www.geeksforgeeks.org/introduction-to-algorithms/#:~:text=Algorithm%20Basics,other%20problem%2Dsolving%20operations%E2%80%9D.&text=Similarly%2C%20algorithms%20help%20to%20do,to%20get%20the%20expected%20output.**

4. ht**tps://www.tutorialspoint.com/programming_methodologies/programming_methodologies_flowchart_elements.htm**

5. https://www.tutorialspoint.com/operating_system/index.htm