UNIT-I: History and Hardware - Computer Hardware, Bits and Bytes, Components, Programming Languages - Machine Language, Assembly Language, Low- and High-Level Languages, Procedural and Object-Oriented Languages, Application and System Software, The Development of C Algorithms,The Software Development Process.

Introduction to Computers:

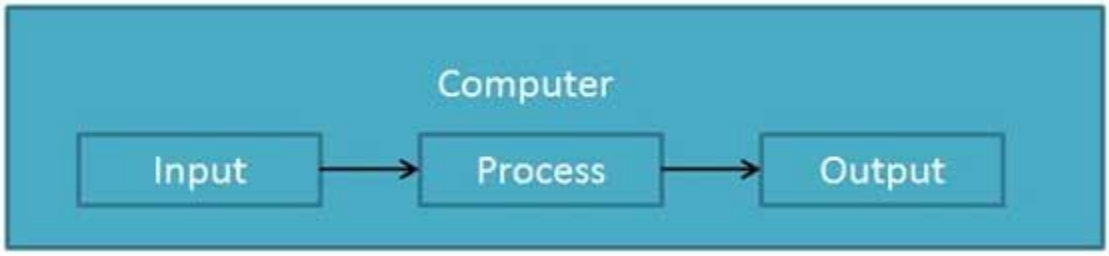
Computer is an advanced electronic device that takes raw data as input from the user and processes it under the control of set of instructions (called program), gives the result (output), and saves it for the future use.

Today's world is an information-rich world and it has become a necessity for everyone to know about computers. Purpose of this tutorial is to introduce you about computer and its fundamentals.

Functionalities of a computer:

Any digital computer carries out five functions in gross terms:

- Takes data as input.
- Stores the data/instructions in its memory and use them when required.
- Processes the data and converts it into useful information.
- Generates the output
- Controls all the above four steps.



Definition

Computer is an electronic data processing device which

- accepts and stores data input,
- processes the data input, and
- generates the output in a required format.

Advantages:

Following list demonstrates the advantages of computers in today's arena.

High Speed

- Computer is a very fast device.
- It is capable of performing calculation of very large amount of data.
- The computer has units of speed in microsecond, nanosecond, and even the picosecond.
- It can perform millions of calculations in a few seconds as compared to man who will spend many months for doing the same task.

Accuracy

- In addition to being very fast, computers are very accurate.
- The calculations are 100% error free.
- Computers perform all jobs with 100% accuracy provided that correct input has been given.

Storage Capability

- Memory is a very important characteristic of computers.
- A computer has much more storage capacity than human beings.
- It can store large amount of data.
- It can store any type of data such as images, videos, text, audio and many others.

Versatility

- A computer is a very versatile machine.
- A computer is very flexible in performing the jobs to be done.
- This machine can be used to solve the problems related to various fields.
- At one instance, it may be solving a complex scientific problem and the very next moment it may be playing a card game.

Reliability

- A computer is a reliable machine.
- Modern electronic components have long lives.
- Computers are designed to make maintenance easy.

Automation

- Computer is an automatic machine.
- Automation means ability to perform the given task automatically.
- Once a program is given to computer i.e., stored in computer memory, the program and instruction can control the program execution without human interaction.

Reduction in Paper Work

- The use of computers for data processing in an organization leads to reduction in paper work and results in speeding up a process.
- As data in electronic files can be retrieved as and when required, the problem of maintenance of large number of paper files gets reduced.

Reduction in Cost

- Though the initial investment for installing a computer is high but it substantially reduces the cost of each of its transaction.

Disadvantages:

Following list demonstrates the disadvantages of computers in today's arena

No I.Q

- A computer is a machine that has no intelligence to perform any task.
- Each instruction has to be given to computer.
- A computer cannot take any decision on its own.

Dependency

- It functions as per a user's instruction, so it is fully dependent on human being

Environment

- The operating environment of computer should be dust free and suitable.

No Feeling

- Computers have no feelings or emotions.
- It cannot make judgement based on feeling, taste, experience, and knowledge unlike a human being.

Hardware represents the physical and tangible components of a computer i.e. the components that can be seen and touched.

Computer Generations:

Generation in computer terminology is a change in technology a computer is/was being used. Initially, the generation term was used to distinguish between varying hardware technologies. But nowadays, generation includes both hardware and software, which together make up an entire computer system.

There are totally five computer generations known till date. Each generation has been discussed in detail along with their time period and characteristics. Here approximate dates against each generations have been mentioned which are normally accepted.

Following are the main five generations of computers

S.N.	Generation & Description
1	**First Generation** The period of first generation: 1946-1959. Vacuum tube based.
2	**Second Generation** The period of second generation: 1959-1965. Transistor based.
3	**Third Generation** The period of third generation: 1965-1971. Integrated Circuit based.
4	**Fourth Generation** The period of fourth generation: 1971-1980. VLSI microprocessor based.
5	**Fifth Generation** The period of fifth generation: 1980-onwards. ULSI microprocessor based

Computer – Types

Computers can be broadly classified by their speed and computing power.

Sr.No.	Type	Specifications
1	PC (Personal Computer)	It is a single user computer system having moderately powerful microprocessor
2	WorkStation	It is also a single user computer system which is similar to personal computer but have more powerful microprocessor.
3	Mini Computer	It is a multi-user computer system which is capable of supporting hundreds of users simultaneously.
4	Main Frame	It is a multi-user computer system which is capable of supporting hundreds of users simultaneously. Software technology is different from minicomputer.
5	Supercomputer	It is an extremely fast computer which can execute hundreds of millions of instructions per second.

Computer Hardware:

Hardware represents the physical and tangible components of a computer i.e. the components that can be seen and touched.

Examples of Hardware are following:

- **Input devices --** keyboard, mouse etc.
- **Output devices --** printer, monitor etc.
- **Secondary storage devices --** Hard disk, CD, DVD etc.
- **Internal components --** CPU, motherboard, RAM etc.

Computer – Memory:

A memory is just like a human brain. It is used to store data and instructions. Computer memory is the storage space in computer where data is to be processed and instructions required for processing are stored. The memory is divided into large number of small parts called cells. Each location or cell has a unique address which varies from zero to memory size minus one. For example if computer has 64k words, then this memory unit has 64 * 1024=65536 memory locations. The address of these locations varies from 0 to 65535.

Memory is primarily of three types

- Cache Memory
- Primary Memory/Main Memory
- Secondary Memory

Cache Memory:

Cache memory is a very high speed semiconductor memory which can speed up CPU. It acts as a buffer between the CPU and main memory. It is used to hold those parts of data and program which are most frequently used by CPU. The parts of data and programs are transferred from disk to cache memory by operating system, from where CPU can access them.

Advantages

The advantages of cache memory are as follows:

- Cache memory is faster than main memory.
- It consumes less access time as compared to main memory.
- It stores the program that can be executed within a short period of time.
- It stores data for temporary use.

Disadvantages

The disadvantages of cache memory are as follows:

- Cache memory has limited capacity.
- It is very expensive.

Primary Memory (Main Memory):

Primary memory holds only those data and instructions on which computer is currently working. It has limited capacity and data is lost when power is switched off. It is generally made up of semiconductor device. These memories are not as fast as registers. The data and instruction required to be processed reside in main memory. It is divided into two subcategories RAM and ROM.

Characteristics of Main Memory

- These are semiconductor memories
- It is known as main memory.
- Usually volatile memory.

- Data is lost in case power is switched off.
- It is working memory of the computer.
- Faster than secondary memories.
- A computer cannot run without primary memory.

Secondary Memory:

This type of memory is also known as external memory or non-volatile. It is slower than main memory. These are used for storing data/Information permanently. CPU directly does not access these memories instead they are accessed via input-output routines. Contents of secondary memories are first transferred to main memory, and then CPU can access it. For example : disk, CD-ROM, DVD etc.

Characteristic of Secondary Memory

- These are magnetic and optical memories
- It is known as backup memory.
- It is non-volatile memory.
- Data is permanently stored even if power is switched off.
- It is used for storage of data in a computer.
- Computer may run without secondary memory.
- Slower than primary memories.

Computer - Memory Units:

Memory unit is:

- the amount of data that can be stored in the storage unit.
- that in which storage capacity is expressed in terms of Bytes.

Following are the main memory storage units:

Sr.No.	Unit	Description
1	Bit (Binary Digit)	A binary digit is logical 0 and 1 representing a passive or an active state of a component in an electric circuit.
2	Nibble	A group of 4 bits is called nibble.
3	Byte	A group of 8 bits is called byte. A byte is the smallest unit which can represent a data item or a character.
4	Word	A computer word, like a byte, is a group of fixed number of bits processed as a unit which varies from computer to computer but is fixed for each computer.The length of a computer word is called word-size or word length and it may be as small as 8 bits or may be as long as 96 bits. A computer stores the information in the form of computer words.

Few higher storage units are following

Sr.No.	Unit	Description
1	Kilobyte (KB)	1 KB = 1024 Bytes
2	Megabyte (MB)	1 MB = 1024 KB
3	GigaByte (GB)	1 GB = 1024 MB
4	TeraByte (TB	1 TB = 1024 GB
5	PetaByte (PB)	1 PB = 1024 TB

Memory is measure in the following units:

1 byte = 8 bits (Each 1 or 0 is called a *bit* (i.e. binary digit). Each character (i.e. a letter, a number, a space, or a punctuation mark) has its own arrangements of 8 bits, e.g. 01000001 = "A",
01000010 = "B".

- 1 KB (kilobyte) = 1024 (2^{10}) bytes
- 1 MB (megabyte) = 1024 (2^{10}) KB
- 1 GB (gigabyte) = 1024 (2^{10}) MB

Computer – Components:

All types of computers follow a same basic logical structure and perform the following five basic operations for converting raw input data into information useful to their users.

Sr.No.	Operation	Description
1	Take Input	The process of entering data and instructions into the computer system
2	Store Data	Saving data and instructions so that they are available for processing as and when required.
3	Processing Data	Performing arithmetic, and logical operations on data in order to convert them into useful information.
4	Output Information	The process of producing useful information or results for the user, such as a printed report or visual display.

5	Control the workflow	Directs the manner and sequence in which all of the above operations are performed.

Input Unit

This unit contains devices with the help of which we enter data into computer. This unit makes link between user and computer. The input devices translate the information into the form understandable by computer.

CPU (Central Processing Unit)

CPU is considered as the brain of the computer. CPU performs all types of data processing operations. It stores data, intermediate results and instructions(program). It controls the operation of all parts of computer.

CPU itself has following three components

- ALU(Arithmetic Logic Unit)
- Memory Unit
- Control Unit

Output Unit

Output unit consists of devices with the help of which we get the information from computer. This unit is a link between computer and users. Output devices translate the computer's output into the form understandable by users.

1. Input Devices

Following are few of the important input devices which are used in a computer:

- Keyboard
- Mouse
- Joy Stick
- Light pen
- Track Ball
- Scanner
- Graphic Tablet
- Microphone
- Magnetic Ink Card Reader(MICR)
- Optical Character Reader(OCR)
- Bar Code Reader
- Optical Mark Reader(OMR)

2. CPU (Central Processing Unit)

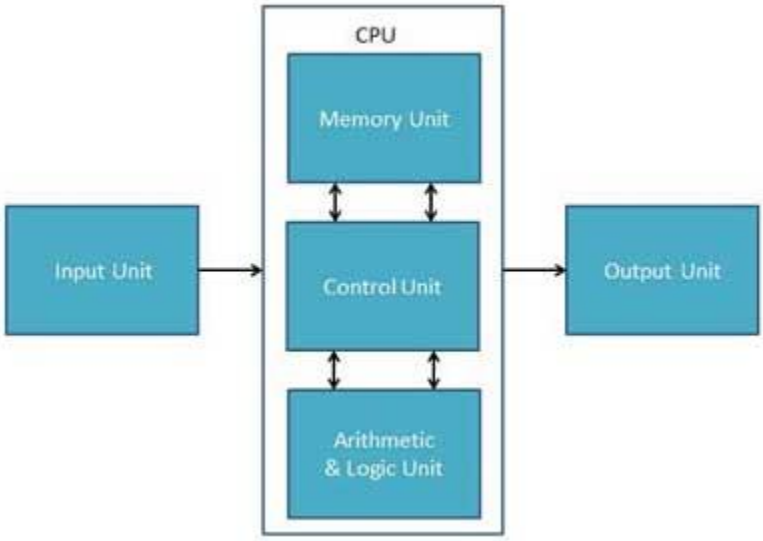
CPU consists of the following features:

- CPU is considered as the brain of the computer.
- CPU performs all types of data processing operations.

- It stores data, intermediate results and instructions(program).
- It controls the operation of all parts of computer.

CPU itself has following three components.

- Memory or Storage Unit
- Control Unit
- ALU(Arithmetic Logic Unit)



Memory or Storage Unit

This unit can store instructions, data and intermediate results. This unit supplies information to the other units of the computer when needed. It is also known as internal storage unit or main memory or primary storage or Random access memory(RAM).

Its size affects speed, power and capability. Primary memory and secondary memory are two types of memories in the computer. Functions of memory unit are:

- It stores all the data and the instructions required for processing.
- It stores intermediate results of processing.
- It stores final results of processing before these results are released to an output device.
- All inputs and outputs are transmitted through main memory.

Control Unit

This unit controls the operations of all parts of computer but does not carry out any actual data processing operations.

Functions of this unit are:

- It is responsible for controlling the transfer of data and instructions among other units of a computer.
- It manages and coordinates all the units of the computer.
- It obtains the instructions from the memory, interprets them, and directs the operation of the computer.
- It communicates with Input/Output devices for transfer of data or results from storage.

ALU(Arithmetic Logic Unit)

This unit consists of two subsections namely

- Arithmetic section
- Logic Section

Arithmetic Section

Function of arithmetic section is to perform arithmetic operations like addition, subtraction, multiplication and division. All complex operations are done by making repetitive use of above operations.

Logic Section

Function of logic section is to perform logic operations such as comparing, selecting, matching and merging of data.

3. Output Devices

Following are few of the important output devices which are used in a computer.

- Monitors
- Graphic Plotter
- Printer

Programming language:

Just as humans use language to communicate, and different regions have different languages, computers also have their own languages that are specific to them.

Different kinds of languages have been developed to perform different types of work on the computer. Basically, languages can be divided into two categories according to how the computer understands them.

Two Basic Types of Computer Language

- **Low-Level Languages:** A language that corresponds directly to a specific machine
- **High-Level Languages:** Any language that is independent of the machine

There are also other types of languages, which include

- **System languages:** These are designed for low-level tasks, like memory and process management
- **Scripting languages:** These tend to be high-level and very powerful
- **Domain-specific languages:** These are only used in very specific contexts

These languages are not mutually exclusive, and some languages can belong to multiple categories. The terms low-level and high-level are also open to interpretation, and some languages that were once considered high-level are now considered low-level as languages have continued to develop.

Low-Level Languages

Low-level computer languages are either machine codes or are very close them. A computer cannot understand instructions given to it in high-level languages or in English. It can only understand and execute instructions given in the form of machine language i.e. binary.

There are two types of low-level languages:

- **Machine Language:** a language that is directly interpreted into the hardware
- **Assembly Language:** a slightly more user-friendly language that directly corresponds to machine language

Machine Language

Machine language is the lowest and most elementary level of programming language and was the first type of programming language to be developed. Machine language is basically the only language that a computer can understand and it is usually written in hex.

In fact, a manufacturer designs a computer to obey just one language, its machine code, which is represented inside the computer by a string of binary digits (bits) 0 and 1. The symbol 0 stands for the absence of an electric pulse and the 1 stands for the presence of an electric pulse. Since a computer is capable of recognizing electric signals, it understands machine language.

Advantages	Disadvantages
Machine language makes fast and efficient use of the computer.	All operation codes have to be remembered
It requires no translator to translate the code. It is directly understood by the computer.	All memory addresses have to be remembered.
	It is hard to amend or find errors in a program written in the machine language.

Assembly Language

Assembly language was developed to overcome some of the many inconveniences of machine language. This is another low-level but very important language in which operation codes and operands are given in the form of alphanumeric symbols instead of 0's and l's.

These alphanumeric symbols are known as mnemonic codes and can combine in a maximum of five-letter combinations e.g. ADD for addition, SUB for subtraction, START, LABEL etc. Because of this feature, assembly language is also known as 'Symbolic Programming Language.'

This language is also very difficult and needs a lot of practice to master it because there is only a little English support in this language. Mostly assembly language is used to help in compiler orientations. The instructions of the assembly language are converted to machine codes by a language translator and then they are executed by the computer.

Advantages	Disadvantages
Assembly language is easier to understand and use as compared to machine language.	Like machine language, it is also machine dependent/specific.
It is easy to locate and correct errors.	Since it is machine dependent, the programmer also needs to understand the hardware.
It is easily modified.	

High-Level Languages

High-level computer languages use formats that are similar to English. The purpose of developing high-level languages was to enable people to write programs easily, in their own native language environment (English).

High-level languages are basically symbolic languages that use English words and/or mathematical symbols rather than mnemonic codes. Each instruction in the high-level language is translated into many machine language instructions that the computer can understand.

Advantages	Disadvantages
High-level languages are user-friendly	A high-level language has to be translated into the machine language by a translator, which takes up time
They are similar to English and use English vocabulary and well-known symbols	The object code generated by a translator might be inefficient compared to an equivalent assembly language program
They are easier to learn	
They are easier to maintain	
They are problem-oriented rather than 'machine'-based	
A program written in a high-level language can be translated into many machine languages and can run on any computer for which there exists an appropriate translator	
The language is independent of the machine on which it is used i.e. programs developed in a high-level language can be run on any computer text	

Types of High-Level Languages

Many languages have been developed for achieving a variety of different tasks. Some are fairly specialized, and others are quite general.

These languages, categorized according to their use, are:

1) Algebraic Formula-Type Processing

These languages are oriented towards the computational procedures for solving mathematical and statistical problems.

Examples include:

- BASIC (Beginners All Purpose Symbolic Instruction Code)
- FORTRAN (Formula Translation)
- PL/I (Programming Language, Version 1)
- ALGOL (Algorithmic Language)
- APL (A Programming Language)

2. Business Data Processing

These languages are best able to maintain data processing procedures and problems involved in handling files. Some examples include:

- COBOL (Common Business Oriented Language)
- RPG (Report Program Generator)

3. String and List Processing

These are used for string manipulation, including search patterns and inserting and deleting characters. Examples are:

- LISP (List Processing)
- Prolog (Program in Logic)

4. Object-Oriented Programming Language

In OOP, the computer program is divided into objects. Examples are:

- C++
- Java

5. Visual Programming Language

These programming languages are designed for building Windows-based applications.Examples are:

- Visual Basic
- Visual Java
- Visual C

Procedural Programming:

Procedural programming is the standard approach used in traditional computer language such as C, Pascal, FORTRAN & BASIC. The basic idea is to have a program specify the sequence of steps that implements a particular algorithm . Procedural programming is a term used to denote the way in which a computer programmer writes a program. This method of developing software, which also is called an application, revolves around keeping code as concise as possible. It also focuses on a very specific end result to be achieved.

Procedural programming creates a step by step program that guides the application through a sequence of instructions. Each instruction is executed in order. Procedural programming focuses on processes. In procedural programming data and functions are stored in separate memory location, while in OOP data and functions are stored in same memory location. Programs are made up of modules, which are parts of a program that can be coded and tested separately, and then assembled to form a complete program.

In procedural languages (i.e. C) these modules are procedures, where a procedure is a sequence of statements. In C for example, procedures are a sequence of imperative statements, such as assignments, tests, loops and invocations of sub procedures. These procedures are functions, which map arguments to return statements.

The design method used in procedural programming is called Top Down Design. This is where you start with a problem (procedure) and then systematically break the problem down into sub problems (sub procedures). This is called functional decomposition, which continues until a sub problem is straightforward enough to be solved by the corresponding sub procedure. When changes are made to the main procedure (top), those changes can cascade to the sub procedures of main, and the sub-sub procedures and so on, where the change may impact all procedures in the pyramid. The problem with PP approach is its handling of data. PP approach gives no importance to data. By 'data' we mean the information collected from user, the new results obtained after calculations etc.

Advantages of Procedural Programming:

- Its relative simplicity, and ease of implementation of compilers and interpreters.
- The ability to re-use the same code at different places in the program without copying it.
- An easier way to keep track of program flow.
- The ability to be strongly modular or structured.
- Needs only less memory.

Disadvantages of Procedural Programming

- Data is exposed to whole program, so no security for data.
- Difficult to relate with real world objects.
- Difficult to create new data types reduces extensibility.
- Importance is given to the operation on data rather than the data.

Object Oriented Programming:

Object-oriented programming (OOP) refers to a type of computer programming (software design) in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure.

In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects.

The Basics: Object Oriented Programming Concepts

If you are new to object-oriented programming languages, you will need to know a few basics before you can get started with code. The following Webopedia definitions will help you better understand object-oriented programming:

Abstraction: The process of picking out (abstracting) common features of objects and procedures.
Class: A category of objects. The class defines all the common properties of the different objects that belong to it.
Encapsulation: The process of combining elements to create a new entity. A procedure is a type of encapsulation because it combines a series of computer instructions.
Information hiding: The process of hiding details of an object or function. Information hiding is a powerful programming technique because it reduces complexity.
Inheritance: a feature that represents the "is a" relationship between different classes.
Interface: the languages and codes that the applications use to communicate with each other and with the hardware.
Messaging: Message passing is a form of communication used in parallel programming and object-oriented programming.
Object: a self-contained entity that consists of both data and procedures to manipulate the data.
Polymorphism: A programming language's ability to process objects differently depending on their data type or class.
Procedure: a section of a program that performs a specific task.

Advantages of Object Oriented Programming

One of the principal advantages of object-oriented programming techniques over procedural programming techniques is that they enable programmers to create modules that do not need to be changed when a new type of object is added. A programmer can simply create a new object that inherits many of its features from existing objects. This makes object-oriented programs easier to modify.

OOPL - Object Oriented Programming Languages

An *object-oriented programming language* (OOPL) is a high-level programming language based on the object-oriented model. To perform object-oriented programming, one needs an object-oriented programming language. Many modern programming languages are object-oriented, however some older programming languages, such as Pascal, do offer object-oriented versions. Examples of object-oriented programming languages include Java, C++ and Smalltalk.

Procedural	**Object-oriented**
procedure	method
record	object
module	class
procedure call	message

Computer – Software

Software is a set of programs, which is designed to perform a well-defined function. A program is a sequence of instructions written to solve a particular problem.

There are two types of software

- System Software
- Application Software

System Software

The system software is collection of programs designed to operate, control, and extend the processing capabilities of the computer itself. System software are generally prepared by computer manufactures. These software products comprise of programs written in low-level languages which interact with the hardware at a very basic level. System software serves as the interface between hardware and the end users.

Some examples of system software are Operating System, Compilers, Interpreter, Assemblers etc.

Features of system software are as follows:

- Close to system
- Fast in speed
- Difficult to design
- Difficult to understand
- Less interactive
- Smaller in size
- Difficult to manipulate
- Generally written in low-level language

Application Software

Application software products are designed to satisfy a particular need of a particular environment. All software applications prepared in the computer lab can come under the category of Application software.

Application software may consist of a single program, such as a Microsoft's notepad for writing and editing simple text. It may also consist of a collection of programs, often called a software package, which work together to accomplish a task, such as a spreadsheet package.

Examples of Application software are following:

- Payroll Software
- Student Record Software
- Inventory Management Software
- Income Tax Software
- Railways Reservation Software
- Microsoft Office Suite Software

- Microsoft Word
- Microsoft Excel
- Microsoft Powerpoint

Features of application software are as follows:

- Close to user
- Easy to design
- More interactive
- Slow in speed
- Generally written in high-level language
- Easy to understand
- Easy to manipulate and use
- Bigger in size and requires large storage space

Relationship between Hardware and Software

- Hardware and software are mutually dependent on each other. Both of them must work together to make a computer produce a useful output.
- Software cannot be utilized without supporting hardware.
- Hardware without set of programs to operate upon cannot be utilized and is useless.
- To get a particular job done on the computer, relevant software should be loaded into the hardware
- Hardware is a one-time expense.
- Software development is very expensive and is a continuing expense.
- Different software applications can be loaded on a hardware to run different jobs.
- A software acts as an interface between the user and the hardware.
- If hardware is the 'heart' of a computer system, then software is its 'soul'. Both are complimentary to each other.

What is a computer algorithm?

To make a computer do anything, you have to write a computer program. To write a computer program, you have to tell the computer, step by step, exactly what you want it to do. The computer then "executes" the program, following each step mechanically, to accomplish the end goal.

When you are telling the computer *what* to do, you also get to choose *how* it's going to do it. That's where**computer algorithms** come in. The algorithm is the basic technique used to get the job done. Let's follow an example to help get an understanding of the algorithm concept.

An Algorithm Development Process

There are many ways to write an algorithm. Some are very informal, some are quite formal and mathematical in nature, and some are quite graphical. The instructions for connecting a DVD player to a television are an algorithm.

The development of an algorithm (a plan) is a key step in solving a problem. Once we have an algorithm, we can translate it into a computer program in some programming language. Our algorithm development process consists of five major steps.

Step 1: Obtain a description of the problem.
Step 2: Analyze the problem.
Step 3: Develop a high-level algorithm.
Step 4: Refine the algorithm by adding more detail.
Step 5: Review the algorithm.

Classifying Programming Languages

Different languages have different purposes, so it makes sense to talk about different kinds, or types, of languages. Some types are:

- Machine languages — interpreted directly in hardware
- Assembly languages — thin wrappers over a corresponding machine language
- High-level languages — anything machine-independent
- System languages — designed for writing low-level tasks, like memory and process management

Machine Code

Most computers work by executing stored programs in a fetch-execute cycle. Machine code generally features

- Registers to store values and intermediate results
- Very low-level machine instructions (add, sub, div, sqrt)
- Labels and conditional jumps to express control flow
- A lack of memory management support — programmers do that themselves

Machine code is usually written in hex. Example for the Intel 64 architecture:

```
89 F8 A9 01 00 00 00 75 06 6B C0
03 FF C0 C3 C1 E0 02 83 E8 03 C3
```

Assembly Language

An assembly language is basically just a simplistic encoding of machine code into something more readable. It does add labeled storage locations and jump targets and subroutine starting addresses, but not much more. Here's the function on the Intel 64 architecture using the GAS assembly language:

```
        .globl  f
        .text
f:
        mov     %edi, %eax      # Put first parameter into eax register
        test    $1, %eax        # Isloate least significant bit
        jnz     odd             # If it's not a zero, jump to odd
        imul    $3, %eax        # It's even, so multiply it by 3
        inc     %eax            # and add 4
        ret                     # and return it
even:
        shl     $2, %eax        # It's odd, so multiply by 4
        sub     $3, %eax        # and subtract 3
        ret                     # and return it
```

Programming Languages. Steps in Developing a C Program

Unstructured programming is a characterized by all the program code being present as a single continuous block Whereas in Structured Programming, programmatic tasks are split into smaller sections (known as functions or subroutines) that can be called whenever they are required.

Unstructured program code is difficult to read and debug. Program structures can always be implemented in any programming language by a combination of conditional statements and Goto statements. Though structured programming is not essential it is preferable. Unstructured programming is still used in some scripting languages such as MS-DOS batch files, older programming languages such as BASIC or FORTRAN. Assembly language is mostly an unstructured language, because the underlying machine code never has structure.

Procedural and Declarative (Non-Procedural) Programming Languages

Procedural programming is a programming model based upon the concept of the procedure call. Procedures contain a series of computational steps to be carried out. Procedures are also known as routines, subroutines, methods, or functions. Any given procedure might be called at any point during a program's execution, including by other procedures or itself.

BASIC, C, C++, COBOL, Fortran, Java, Pascal, Perl, VBScript, Visual Basic are some examples of procedural programming languages.

It is a high-level language that describes a problem rather than defining a solution.

What type of language is C?

C is a general purpose, structured programming language. Its instructions consist of terms that resemble algebraic expressions, augmented by English Keywords such as if, else, for, do and while. In this respect C resembles other high-level structured programming languages such as Pascal and Fortran. C also contains certain additional features, that allow it to be used at a lower level, thus bridging the gap between machine language and the more conventional high-level languages. This flexibility allows C to be used for systems programming (Eg: writing operating systems) as well as for applications programming.

Steps/Programs in the Development of a "C" Program

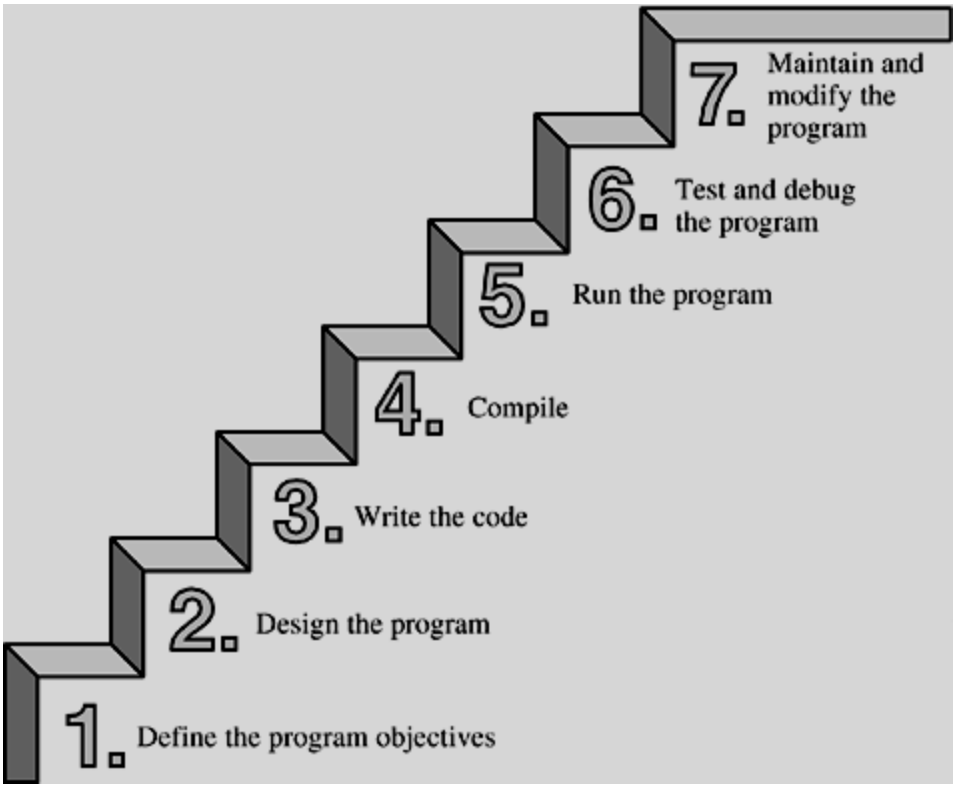
The development of a "C" program involves the use of the following programs in the order of their usage

- **Editor:**This program is used for Writing the Source Code, the first thing that any programmer writing a program in any language would be doing.
- **Debugger:**This program helps us identify syntax errors in the source code.
- **Pre Processer:**There are certain special instructions within the source code identified by the # symbol that are carried on by a special program called a preprocessor.
- **Compiler:**The process of converting the C source code to machine code and is done by a program called Compiler.
- **Linker:**The machine code relating to the source code you have written is combined with some other machine code to derive the complete program in an executable file. This is done by a program called the linker.

Using C: Seven Steps

C is a compiled language. First, to give you an overview of programming, let's break down the act of writing a C program into seven steps .Note that this is an idealization. In practice, particularly for larger projects, you would go back and forth, using what you learned at a later step to refine an earlier step.

The seven steps of programming.



Step 1: Define the Program Objectives

Naturally enough, you should start with a clear idea of what you want the program to do. Think in terms of the information your program needs, the calculation and manipulation the program needs to do, and the information the program should report back to you. At this level of planning, you should be thinking in general terms, not in terms of some specific computer language.

Step 2: Design the Program

After you have a abstract picture of what your program have to to do, you should decide how the program will go about it. What should the user interface be like? How should the program be organized? Who will the target user be? How much time do you have to complete the program?

You also need to decide how to represent the data in the program and, possibly, as well as which methods to use to process the data.

Step 3: Write the Code

Now that you have a clear design for your program, you can begin to implement it by writing the code. That is, you translate your program design into the C language. Here is where you really have to put your

knowledge of C to work. You can draw your ideas on paper, but eventually you have to get your code into the computer. The mechanics of this process depend on your programming environment. In general, you use a text editor to create what is called a ***source code*** file. This file contains the C rendition of your program design. An example of C source code.

Example of C Source Code

```
#include <stdio.h>
int main(void)
{
    int dogs;
    printf("How many dogs do you have?\n");
    scanf("%d", &dogs);
    printf("So you have %d dog(s)!\n", dogs);
    return 0;
}
```

As part of this step, you should document your work. The simplest way is to use C's comment facility to incorporate explanations into your source code.

Step 4: Compile

The next step is to compile the source code. Recall that the compiler is a program whose job is to convert source code into executable code. ***Executable code*** is code in the native language, or ***machine language***, of your computer. This language consists of detailed instructions expressed in a numeric code.

As you read earlier, different computers have different machine languages, and a C compiler translates C into a particular machine language. C compilers also include code from C libraries into the final program; the libraries contain a fund of standard routines, such as printf() and scanf(), for your use. (More accurately, a program called a *linker* brings in the library routines, but the compiler runs the linker for you on most systems.) The end result is an executable file containing code that the computer understands and that you can run.

The compiler also checks that your program is valid C. If the compiler finds errors, it reports them to you and doesn't produce an executable file. Understanding a particular compiler's complaints is another skill you will pick up.

Step 5: Run the Program

Traditionally, the executable file is a program you can run. To run the program in many common environments, including MS-DOS, Unix, Linux consoles, just type the name of the executable file. Other environments, such as VMS on a VAX, might require a run command or some other mechanism.

Integrated development environments (IDEs), such as those provided for Windows and Macintosh environments, allow you to edit and execute your C program from within the IDE by selecting choices from a menu or by pressing special keys. The resulting program also can be run directly from the operating system by clicking or double-clicking the filename or icon.

Step 6: Test and Debug the Program

The fact that your program runs is a good sign, but it's possible that it could run incorrectly. Consequently, you should check to see that your program does what it is supposed to do. You'll find that some of your programs have mistakes—*bugs*. ***Debugging*** is the process of finding and fixing program errors. Making mistakes is a natural part of learning. It seems inherent to programming, so when you combine learning and programming. As you become a more powerful and subtle programmer, your errors, too, will become more powerful and subtle.

Step 7: Maintain and Modify the Program

When you create a program for yourself or for someone else, that program could see extensive use. If it does, you'll probably find reasons to make changes in it. Perhaps there is a minor bug that shows up only when someone enters a name beginning with *Zz*, or you might think of a better way to do something in the program. You could add a clever new feature. You might adapt the program so that it runs on a different computer system. All these tasks are greatly simplified if you document the program clearly and if you follow sound design practices.

C – Basic Program

BASIC COMMANDS IN C PROGRAMMING TO WRITE BASIC C PROGRAM:

Below are few commands and syntax used in C programming to write a simple C program. Let's see all the sections of a simple C program line by line.

S.no	Command	Explanation
1	#include <stdio.h>	This is a preprocessor command that includes standard input output header file(stdio.h) from the C library before compiling a C program
2	int main()	This is the main function from where execution of any C program begins.
3	{	This indicates the beginning of the main function.
4	/*_some_comments_*/	whatever is given inside the command "/* */" in any C program, won't be considered for compilation and execution.
5	printf("Hello_World! ");	printf command prints the output onto the screen.
6	getch();	This command waits for any character input from keyboard.
7	return 0;	This command terminates C program (main function) and returns 0.
8	}	This indicates the end of the main function.

BASIC STRUCTURE OF C PROGRAM:

Structure of C program is defined by set of rules called protocol, to be followed by programmer while writing C program. All C programs are having sections/parts which are mentioned below.

1. Documentation section
2. Link Section
3. Definition Section
4. Global declaration section
5. Function prototype declaration section
6. Main function
7. User defined function definition section

S.No	Sections	Description
1	Documentation section	We can give comments about the program, creation or modified date, author name etc in this section. The characters or words or anything which are given between "/*" and "*/", won't be considered by C compiler for compilation process.These will be ignored by C compiler during compilation.
2	Link Section	Header files that are required to execute a C program are included in this section.
3	Definition Section	In this section, variables are defined and values are set to these variables.
4	Global declaration section	Global variables are defined in this section. When a variable is to be used throughout the program, can be defined in this section.
5	Function prototype	Function prototype gives many information about a function like return type, parameter names used inside the function.
6	Main function	Every C program is started from main function and this function contains two major sections called declaration
7	User defined function section	User can define their own functions in this section which perform particular task as per the user requirement.

C – printf and scanf

- printf() and scanf() functions are inbuilt library functions in C which are available in C library by default. These functions are declared and related macros are defined in "stdio.h" which is a header file.
- We have to include "stdio.h" file as shown in below C program to make use of these printf() and scanf() library functions.

C PRINTF() FUNCTION:

- printf() function is used to print the "character, string, float, integer, octal and hexadecimal values" onto the output screen.
- We use printf() function with %d format specifier to display the value of an integer variable.
- Similarly %c is used to display character, %f for float variable, %s for string variable, %lf for double and %x for hexadecimal variable.
- To generate a newline, we use "\n" in C printf() statement.

Note: C language is case sensitive. For example, printf() and scanf() are different from Printf() and Scanf(). All characters in printf() and scanf() functions must be in lower case.

EXAMPLE PROGRAM FOR C PRINTF() FUNCTION:

```
#include <stdio.h>
int main()
{
char ch = 'A';
char str[20] = "fresh2refresh.com";
float flt = 10.234;
int no = 150;
double dbl = 20.123456;
printf("Character is %c \n", ch);
printf("String is %s \n" , str);
printf("Float value is %f \n", flt);
printf("Integer value is %d\n" , no);
printf("Double value is %lf \n", dbl);
printf("Octal value is %o \n", no);
printf("Hexadecimal value is %x \n", no);
return 0;
}
```

Output: Character is A
String is fresh2refresh.com
Float value is 10.234000
Integer value is 150
Double value is 20.123456

Octal value is 226
Hexadecimal value is 96

You can see the output with the same data which are placed within the double quotes of printf statement in the program except

- %d got replaced by value of an integer variable (no),
- %c got replaced by value of a character variable (ch),
- %f got replaced by value of a float variable (flt),
- %lf got replaced by value of a double variable (dbl),
- %s got replaced by value of a string variable (str),
- %o got replaced by a octal value corresponding to integer variable (no),
- %x got replaced by a hexadecimal value corresponding to integer variable
- \n got replaced by a newline.

C SCANF() FUNCTION:

- scanf() function is used to read character, string, numeric data from keyboard
- Consider below example program where user enters a character. This value is assigned to the variable "ch" and then displayed.
- Then, user enters a string and this value is assigned to the variable "str" and then displayed.

EXAMPLE PROGRAM FOR PRINTF() AND SCANF() FUNCTIONS IN C:

```
#include <stdio.h>
int main()
{
char ch;
char str[100];
printf("Enter any character \n");
scanf("%c", &ch);
printf("Entered character is %c \n", ch);
printf("Enter any string ( upto 100 character ) \n");
scanf("%s", &str);
printf("Entered string is %s \n", str);
}
```

Output :

```
Enter any character:  a
Entered character is:  a
Enter any string ( upto 100 character )  :  hai
Entered string is :hai
```

The format specifier %d is used in scanf() statement. So that, the value entered is received as an integer and %s for string.

C – Tokens,Keywords

C TOKENS:

C tokens are the basic buildings blocks in C language which are constructed together to write a C program. Each and every smallest individual units in a C program are known as C tokens. C tokens are of six types. They are,

1. Keywords (eg: int, while),
2. Identifiers (eg: main, total),
3. Constants (eg: 10, 20),
4. Strings (eg: "total", "hello"),
5. Special symbols (eg: (), {}),
6. Operators (eg: +, /,-,*)

C TOKENS EXAMPLE PROGRAM:

```
int main()
{
int x, y, total;
x = 10, y = 20;
total = x + y;
Printf ("Total = %d \n", total);
}
```

where,

- main – identifier
- {,}, (,) – delimiter
- int – keyword
- x, y, total – identifier
- main, {, }, (,), int, x, y, total – tokens

IDENTIFIERS IN C LANGUAGE:

- Each program elements in a C program are given a name called identifiers.
- Names given to identify Variables, functions and arrays are examples for identifiers. eg. x is a name given to integer variable in above program.

RULES FOR CONSTRUCTING IDENTIFIER NAME IN C:

1. First character should be an alphabet or underscore.
2. Succeeding characters might be digits or letter.
3. Punctuation and special characters aren't allowed except underscore.
4. Identifiers should not be keywords.

KEYWORDS IN C LANGUAGE:

- Keywords are pre-defined words in a C compiler.
- Each keyword is meant to perform a specific function in a C program.
- Since keywords are referred names for compiler, they can't be used as variable name.

C language supports 32 keywords which are given below. Click on each keywords below for detail

AUTO	DOUBLE	INT	STRUCT	CONST	FLOAT	SHORT	UNSIGNED
BREAK	ELSE	LONG	SWITCH	CONTINUE	FOR	SIGNED	VOID
CASE	ENUM	REGISTER	TYPEDE	DEFAULT	GOTO	SIZEOF	VOLATILE
CHAR	EXTERN	RETURN	UNIOnN	DO	IF	STATIC	WHILE

description and example programs.

C – CONSTANTS

- C Constants are also like normal variables. But, only difference is, their values can not be modified by the program once they are defined.
- Constants refer to fixed values. They are also called as literals
- Constants may be belonging to any of the data type.

Syntax:

const data_type variable_name; (or) const data_type *variable_name;

VARIABLE:

- C variable is a named location in a memory where a program can manipulate the data. This location is used to hold the value of the variable.
- The value of the C variable may get change in the program.
- C variable might be belonging to any of the data type like int, float, char etc.

RULES FOR NAMING C VARIABLE:

1. Variable name must begin with letter or underscore.
2. Variables are case sensitive
3. They can be constructed with digits, letters.
4. No special symbols are allowed other than underscore.
5. sum, height, _value are some examples for variable name

DECLARING & INITIALIZING C VARIABLE:

- Variables should be declared in the C program before to use.

- Memory space is not allocated for a variable while declaration. It happens only on variable definition.
- Variable initialization means assigning a value to the variable.

S.No	Type	Syntax	Example
1	Variable declaration	data_type variable_name;	int x, y, z; char flat, ch;
2	Variable initialization	data_type variable_name = value;	int x = 50, y = 30; char flag = 'x', ch='l';

DIFFERENCE BETWEEN VARIABLE DECLARATION & DEFINITION IN C:

S.No	Variable declaration	Variable definition
1	Declaration tells the compiler about data type and size of the variable.	Definition allocates memory for the variable.
2	Variable can be declared many times in a program.	It can happen only one time for a variable in a program.
3	The assignment of properties and identification to a variable.	Assignments of storage space to a variable.