

Module 1

Introduction to Programming

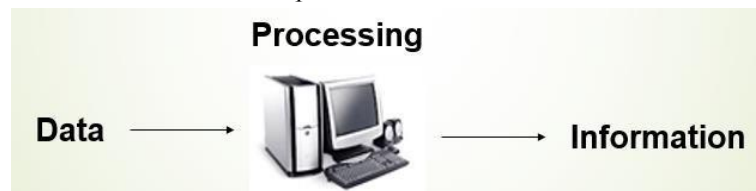
1.1 Introduction to Components of a Computer System

Computer: A computer is an electronic device, operating under the control of instructions stored in its own memory that can accept data (input), process the data according to specified rules, produce information (output), and store the information for future use.

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.

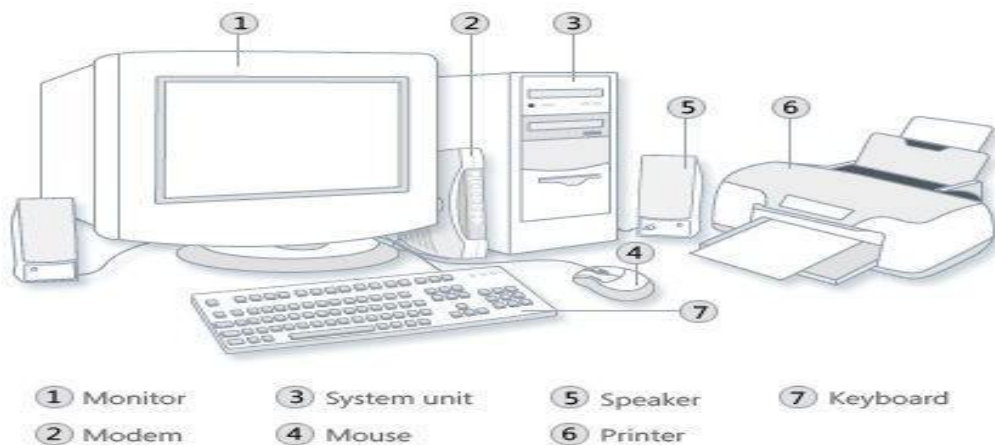


Computer Components

Any kind of computers consists of HARDWARE AND SOFTWARE.

Hardware:













Computer hardware is the collection of physical elements that constitutes a computer system. Computer hardware refers to the physical parts or components of a computer such as the monitor, mouse, keyboard, computer data storage, hard drive disk (HDD), system unit (graphic cards, sound cards, memory, motherboard and chips), etc. all of which are physical objects that can be touched.



Input Devices

Input device is any peripheral (piece of computer hardware equipment to provide data and control signals to an information processing system such as a computer or other information appliance.

Input device Translate data from form that humans understand to one that the computer can work with. Most common are keyboard and mouse

Examples of Manual Input Devices			
Keyboard 	Numeric Keypad 	Pointing Device 	Remote Control 
Joystick 	Touch Screen 	Scanner 	Graphics Tablet 
Microphone 	Digital Camera 	Webcams 	Light Pens 

Central Processing Unit (CPU)

A CPU is brain of a computer. It is responsible for all functions and processes. Regarding computing power, the CPU is the most important element of a computer system.

The CPU is comprised of three main parts :

Arithmetic Logic Unit (ALU): Executes all arithmetic and logical operations. Arithmetic calculations like as addition, subtraction, multiplication and division. Logical operation like compare numbers, letters, or special characters

Control Unit (CU): controls and co-ordinates computer components.

- 1.Read the code for the next instruction to be executed.
- 2.Increment the program counter so it points to the next instruction.
- 3.Read whatever data the instruction requires from cells in memory.
- 4.Provide the necessary data to an ALU or register.
- 5.If the instruction requires an ALU or specialized hardware to complete, instruct the hardware to perform the requested operation.

Registers :Stores the data that is to be executed next, "very fast storage area".

Primary Memory:-

1.RAM: Random Access Memory (RAM) is a memory scheme within the computer system responsible for storing data on a temporary basis, so that it can be promptly accessed by the processor as and when needed. It is volatile in nature, which means that data will be erased once supply to the storage device is turned off. RAM stores data randomly and the processor accesses these data randomly from the RAM storage. RAM is considered "random access" because you can access any memory cell directly if you know the row and column that intersect at that cell.

2.ROM (Read Only Memory): ROM is a permanent form of storage. ROM stays active regardless of whether power supply to it is turned on or off. ROM devices do not allow data stored on them to be modified.

Secondary Memory:-

Stores data and programs permanently :its retained after the power is turned off

1.Hard drive (HD): A hard disk is part of a unit, often called a "disk drive," "hard drive," or "hard disk drive," that store and provides relatively quick access to large amounts of data on an electromagnetically charged surface or set of surfaces.

2.Optical Disk: an optical disc drive (ODD) is a disk drive that uses laser light as part of the process of reading or writing data to or from optical discs. Some drives can only read from discs, but recent drives are commonly both readers and recorders, also called burners or writers. Compact

discs, DVDs, and Blu-ray discs are common types of optical media which can be read and recorded by such drives. Optical drive is the generic name; drives are usually described as "CD" "DVD", or "Bluray", followed by "drive", "writer", etc. There are three main types of optical media: CD, DVD, and Blu-ray disc. CDs can store up to 700 megabytes (MB) of data and DVDs can store up to 8.4 GB of data. Blu-ray discs, which are the newest type of optical media, can store up to 50 GB of data. This storage capacity is a clear advantage over the floppy disk storage media (a magnetic media), which only has a capacity of 1.44 MB.

3.Flash Disk

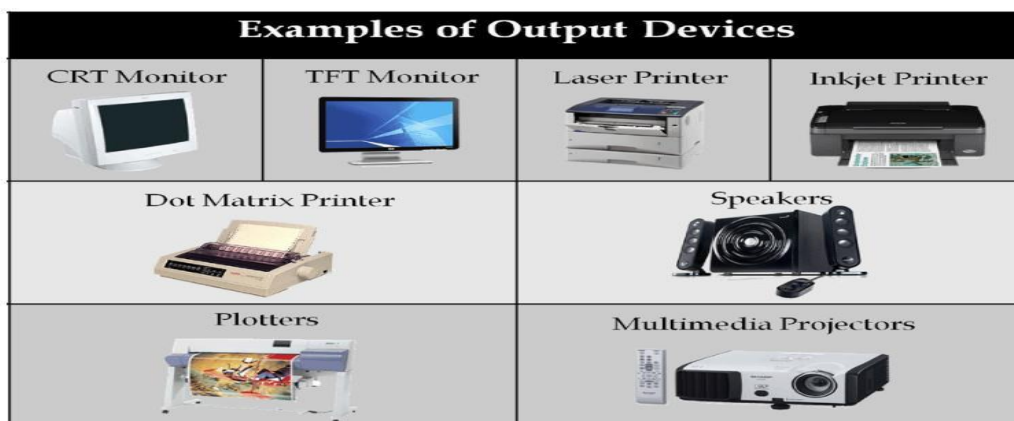
A storage module made of flash memory chips. A Flash disks have no mechanical platters or access arms, but the term "disk" is used because the data are accessed as if they were on a hard drive. The disk storage structure is emulated.

Comparison between Main memory (RAM) and Secondary Memory (Hard disk)

RAM	Hard Disk (Hard Drive)
Memory	Storage
Smaller amount (typically 500 MB-6 GB)	Much larger amount (typically 80GB to 1000 GB)
Temporary storage of files and programs	Permanent storage of files and programs
A little like your real desktop - has only your current work on it (which could be ruined by a spill of Coke or coffee!)	Like a file cabinet - has long-term storage of work (it's safe from spills!)
Contents disappear when you turn off power to the computer and when the computer crashes	Contents remain when you turn off the power to the computer (they don't disappear unless you purposely delete them), and when the computer crashes
Consists of chips (microprocessors)	Consists of hard disks (platters)
When you want to use a program, a temporary copy is put into RAM and that's the copy you use	Holds the original copy of the program permanently

Output devices

An output device is any piece of computer hardware equipment used to communicate the results of data processing carried out by an information processing system (such as a computer) which converts the electronically generated information into human- readable form.



Software

Software is a generic term for organized collections of computer data and instructions, often broken into two major categories: system software that provides the basic non- task-specific functions of the computer, and application software which is used by users to accomplish specific tasks.

Software Types

1. **System software** is responsible for controlling, integrating, and managing the individual hardware components of a computer system so that other software and the users of the system see it as a functional unit without having to be concerned with the low-level details such as transferring data from memory to disk, or rendering text onto a display. Generally, system software consists of an operating system and some fundamental utilities such as disk formatters, file managers, display managers, text editors, user authentication (login) and management tools, and networking and device control software.
2. **Application software** is used to accomplish specific tasks other than just running the computer system. Application software may consist of a single program, such as an image viewer; a small collection of programs (often called a software package) that work closely together to accomplish a task, such as a spreadsheet or text processing system; a larger collection (often called a software suite) of related but independent programs and packages that have a common user interface or shared data format, such as Microsoft Office, which consists of closely integrated word processor, spreadsheet, database, etc.; or a software system, such as a database management system, which is a collection of fundamental programs that may provide some service to a variety of other independent applications.

Comparison between Application Software and System Software

	System Software	Application Software
	Computer software, or just software is a general term primarily used for digitally stored data such as computer programs and other kinds of information read and written by computers. App comes under computer software though it has a wide scope now.	Application software, also known as an application or an "app", is computer software designed to help the user to perform specific tasks.
Example:	Microsoft Windows Linux Unix Mac OSX DOS	Opera (Web Browser) Microsoft Word (Word Processing) Microsoft Excel (Spreadsheet software) MySQL (Database Software) Microsoft PowerPoint (Presentation Software) Adobe Photoshop (Graphics Software)
Interaction:	Generally, users do not interact with system software as it works in the background.	Users always interact with application software while doing different activities.
Dependancy:	System software can run independently of the application software.	Application software cannot run without the presence of the system software.

Unit of Measurements

Storage measurements: The basic unit used in computer data storage is called a bit (binary digit). Computers use these little bits, which are composed of ones and zeros, to do things and talk to other computers. All your files, for instance, are kept in the computer as binary files and translated into words and pictures by the software (which is also ones and zeros). This two number system, is called a “binary number system” since it has only two numbers in it. The decimal number system in contrast has ten unique digits, zero through nine.

Computer Storage units

Bit	BIT	0 or 1
Kilobyte	KB	1024 bytes
Megabyte	MB	1024 kilobytes
Gigabyte	GB	1024 megabytes
Terabyte	TB	1024 gigabytes

Size example

1 bit - answer to an yes/no question

1 byte - a number from 0 to 255.

90 bytes: enough to store a typical line of text from a book.

4 KB: about one page of text.

120 KB: the text of a typical pocket book.

3 MB - a three minute song (128k bitrate)

650-900 MB - an CD-ROM

1 GB -114 minutes of uncompressed CD-quality audio at 1.4 Mbit/s

8-16 GB - size of a normal flash drive

Speed measurement: The speed of Central Processing Unit (CPU) is measured by Hertz (Hz), Which represent a CPU cycle. The speed of CPU is known as Computer Speed.

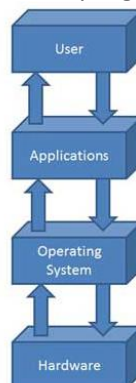
CPU SPEED MEASURES	
1 hertz or Hz	1 cycle per second
1 MHz	1 million cycles per second or 1000 Hz
1 GHz	1 billion cycles per second or 1000 MHz

Where a program is stored and executed: Programs are stored on secondary storage devices such as hard disks. When you install a program on your computer, the program is actually copied to your hard disk. But when you execute a program, the program is copied (loaded) from your hard disk to the main memory, and that copy of the program is executed.

Operating System:

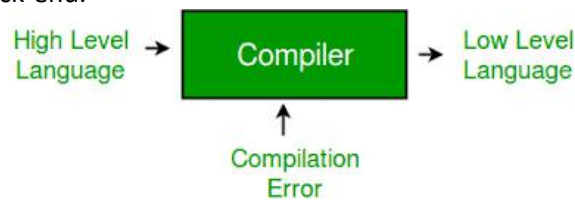
The Operating System is a program with the following features –

- 1) An operating system is a program that acts as an interface between the software and the computer hardware.
- 2) It is an integrated set of specialized programs used to manage overall resources and operations of the computer.
- 3) It is a specialized software that controls and monitors the execution of all other programs that reside in the computer, including application programs and other system software.



Compiler: A compiler is a computer program which helps you transform source code written in a high-level language into low-level machine language. It translates the code written in one programming language to some other language without changing the meaning of the code. The compiler also makes the end code efficient which is optimized for execution time and memory space.

The compiling process includes basic translation mechanisms and error detection. Compiler process goes through lexical, syntax, and semantic analysis at the front end, and code generation and optimization at a back-end.



1.2 Idea of Algorithm

An algorithm is a step by step method of solving a problem. It is commonly used for data processing, calculation and other related computer and mathematical operations.

An algorithm is also used to manipulate data in various ways, such as inserting a new data item, searching for a particular item or sorting an item.

Algorithms are generally created independent of underlying languages, i.e. an algorithm can be implemented in more than one programming language.

Steps to solve logical and numerical problems:

A computer cannot solve a problem on its own. One has to provide step by step solutions of the problem to the computer. In fact, the task of problem solving is not that of the computer. It is the programmer who has to write down the solution to the problem in terms of simple operations which the computer can understand and execute.

In order to solve a problem by the computer, one has to pass through certain stages or steps. They are:

1. Understanding the problem
2. Analyzing the problem
3. Developing the solution
4. Coding and implementation.

1. Understanding the problem: Here we try to understand the problem to be solved in totally. Before with the next stage or step, we should be absolutely sure about the objectives of the given problem.

2. Analyzing the problem: After understanding thoroughly the problem to be solved, we look different ways of solving the problem and evaluate each of these methods. The idea here is to search an appropriate solution to the problem under consideration. The end result of this stage is a broad overview of the sequence of operations that are to be carried out to solve the given problem.

3. Developing the solution: Here the overview of the sequence of operations that was the result of analysis stage is expanded to form a detailed step by step solution to the problem under consideration.

4. Coding and implementation: The last stage of the problem solving is the conversion of the detailed sequence of operations into a language that the computer can understand. Here each step is converted to its equivalent instruction or instructions in the computer language that has been chosen for the implementation.

Representation of algorithms:

A set of sequential steps usually written in Ordinary Language to solve a given problem is called Algorithm.

It may be possible to solve to problem in more than one ways, resulting in more than one algorithm. The choice of various algorithms depends on the factors like reliability, accuracy and easy to modify. The most important factor in the choice of algorithm is the time requirement to execute it, after writing code in High-level language with the help of a computer. The algorithm which will need the least time when executed is considered the best.

Steps involved in algorithm development

An algorithm can be defined as “a complete, unambiguous, finite number of logical steps for solving a specific problem “

Step1. Identification of input: For an algorithm, there are quantities to be supplied called input and these are fed externally. The input is to be identified first for any specified problem.

Step2: Identification of output: From an algorithm, at least one quantity is produced, called for any specified problem.

Step3 : Identification the processing operations : All the calculations to be performed in order to lead to output from the input are to be identified in an orderly manner.

Step4 : Processing Definiteness : The instructions composing the algorithm must be clear and there should not be any ambiguity in them.

Step5 : Processing Finiteness : If we go through the algorithm, then for all cases, the algorithm should terminate after a finite number of steps.

Step6 : Possessing Effectiveness : The instructions in the algorithm must be sufficiently basic and in practice they can be carries out easily.

An algorithm must possess the following properties

1.Finiteness: An algorithm must terminate in a finite number of steps

2.Definiteness: Each step of the algorithm must be precisely and unambiguously stated

3.Effectiveness: Each step must be effective, in the sense that it should be primitive easily convert able into program statement) can be performed exactly in a finite amount of time.

4.Generalality: The algorithm must be complete in itself so that it can be used to solve problems of a specific type for any input data.

5.Input/output: Each algorithm must take zero, one or more quantities as input data produce one or more output values. An algorithm can be written in English like sentences or in any standard representation sometimes, algorithm written in English like languages are called Pseudo Code

Example

1. Suppose we want to find the average of three numbers, the algorithm is as follows

Step 1 Read the numbers a, b, c

Step 2 Compute the sum of a, b and c

Step 3 Divide the sum by 3

Step 4 Store the result in variable d

Step 5 Print the value of d

Step 6 End of the program

Algorithms for Simple Problem

Write an algorithm for the following

1. Write an algorithm to calculate the simple interest using the formula. Simple interest = $P * N * R / 100$.

Where P is principle Amount, N is the number of years and R is the rate of interest.

Step 1: Read the three input quantities' P, N and R.

Step 2 : Calculate simple interest as

Simple interest = $P * N * R / 100$

Step 3: Print simple interest.

Step 4: Stop.

2. Area of Triangle: Write an algorithm to find the area of the triangle.

Let b, c be the sides of the triangle ABC and A the included angle between the given sides.

Step 1: Input the given elements of the triangle namely sides b, c and angle between the sides A.

Step 2: Area = $(1/2) * b * c * \sin A$

Step 3: Output the Area

Step 4: Stop.

3. Write an algorithm to find the largest of three numbers X, Y, Z.

Step 1: Read the numbers X, Y, Z.

Step 2: if $(X > Y)$

Big = X

else BIG = Y

Step 3 : if $(BIG < Z)$

Step 4: Big = Z

Step 5: Print the largest number i.e. Big

Step 6: Stop.

4. Write down an algorithm to find the largest data value of a set of given data values

Algorithm largest of all data values:

Step 1: LARGE = 0

Step 2: read NUM

Step 3: While NUM >= 0 do

3.1 if NUM > LARGE

3.1.1 then

3.1.1.1 LARGE = NUM

3.2. read NUM

Step 4: Write "largest data value is", LARGE Step 5: end.

5. Write an algorithm which will test whether a given integer value is prime or not.

Algorithm prime testing:

Step 1: M = 2

Step 2: read N

Step 3: MAX= SQRT (N)

Step 4: While M <= MAX do

4.1 if $(M * (N/M) = N)$

4.1.1 then

4.1.1.1 go to step 7

4.2. M =M + 1

Step 5: Write "number is prime"
Step 6: go to step 8
Step 7: Write "number is not a prime"
Step 8: end.

6. Write algorithm to find the factorial of a given number N

Step 1: PROD 1
Step 2: I 0
Step 3: read N
Step 4: While I < N do
4.1 I I + 1
4.2. PROD PROD * I
Step 5: Write "Factorial of", N, "is", PROD
Step 6: end.

7. Write an algorithm to find sum of given data values until negative value is entered.

Algorithm Find – Sum

Step 1: SUM=0
Step 2: I =0
Step 3: read NEW VALUE
Step 4: While NEW VALUE < = 0 do
4.1 SUM= SUM + NEW VALUE
4.2 I=I + 1
4.3 read NEW VALUE
Step 5: Write "Sum of", I, "data value is, "SUM
Step 6: END

8. Write an algorithm to calculate the perimeter and area of rectangle. Given its length and width.

Step 1: Read length of the rectangle.
Step 2: Read width of the rectangle.
Step 3: Calculate perimeter of the rectangle using the formula $\text{perimeter} = 2 * (\text{length} + \text{width})$
Step 4: Calculate area of the rectangle using the formula $\text{area} = \text{length} * \text{width}$.
Step 5: Print perimeter.
Step 6: Print area.
Step 7: Stop.

Flowchart

A flow chart is a step by step diagrammatic representation of the logic paths to solve a given problem. Or A flowchart is visual or graphical representation of an algorithm.

The flowcharts are pictorial representation of the methods to be used to solve a given problem and help a great deal to analyze the problem and plan its solution in a systematic and orderly manner. A flowchart when translated in to a proper computer language, results in a complete program.

Advantages of Flowcharts

1. The flowchart shows the logic of a problem displayed in pictorial fashion which facilitates easier checking of an algorithm.
2. The Flowchart is good means of communication to other users. It is also a compact means of recording an algorithm solution to a problem.
3. The flowchart allows the problem solver to break the problem into parts. These parts can be connected to make master chart.
4. The flowchart is a permanent record of the solution which can be consulted at a later time.

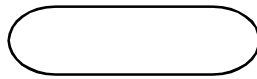
Differences between Algorithm and Flowchart

Algorithm	Flowchart
1.A method of representing the step-by-step logical procedure for solving a problem.	1.Flowchart is diagrammatic representation of an algorithm constructed using different types of boxes and symbols.
2..It contains step-by-step English descriptions, each step representing particular operation leading to a solution of problem	2.The flowchart employs a series of blocks and arrows, each of which represents particular step in an algorithm
3.These are particularly useful for small problems	3.These are useful for detailed representations of complicated programs
4.For complex programs, algorithms prove to be Inadequate	4.For complex programs, Flowcharts prove to be adequate

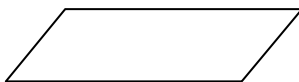
Symbols used in Flow-Charts

The symbols that we make use while drawing flowcharts as given below are as per conventions followed by International Standard Organization (ISO).

a. Oval: Rectangle with rounded sides is used to indicate either START/ STOP of the program



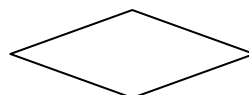
b. Input and output indicators: Parallelograms are used to represent input and output operations. Statements like INPUT, READ and PRINT are represented in these Parallelograms.



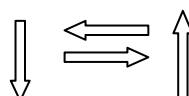
c. Process Indicators: - Rectangle is used to indicate any set of processing operation such as for storing arithmetic operations.



d. Decision Makers: The diamond is used for indicating the step of decision making and therefore known as decision box. Decision boxes are used to test the conditions or ask questions and depending upon the answers, the appropriate actions are taken by the computer. The decision box symbol is



e. Flow Lines: Flow lines indicate the direction being followed in the flowchart. In a Flowchart, every line must have an arrow on it to indicate the direction. The arrows may be in any direction



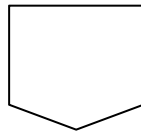
f. On- Page connectors: Circles are used to join the different parts of a flowchart and these circles are called on-page connectors. The uses of these connectors give a neat shape to the

flowcharts. In a complicated problems, a flowchart may run in to several pages. The parts of the flowchart on different pages are to be joined with each other. The parts to be joined are indicated by



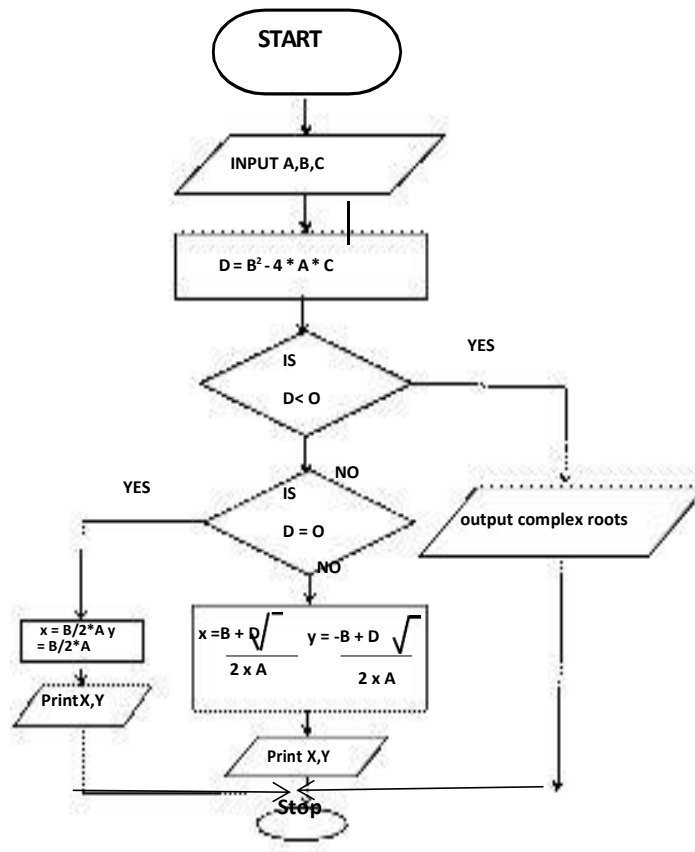
circle

g. Off-page connectors: This connector represents a break in the path of flowchart which is too large to fit on a single page. It is similar to on-page connector. The connector symbol marks where the algorithm ends on the first page and where it continues on the second.

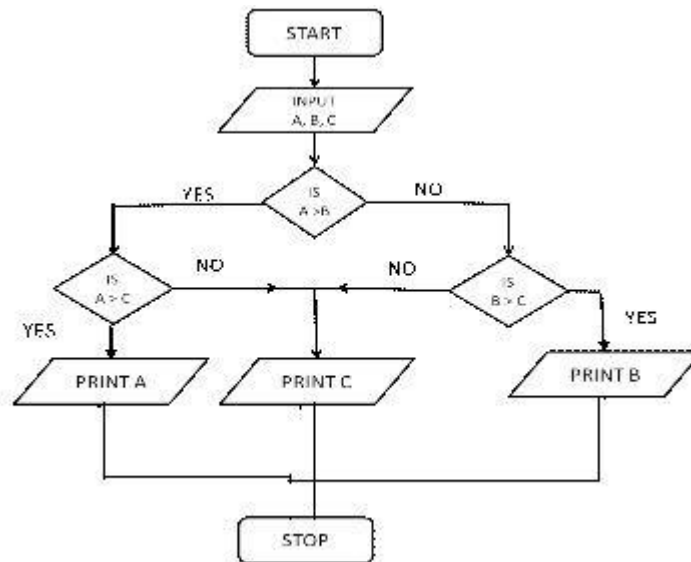


Simple Problems using Flow Chart Draw the Flowchart for the following

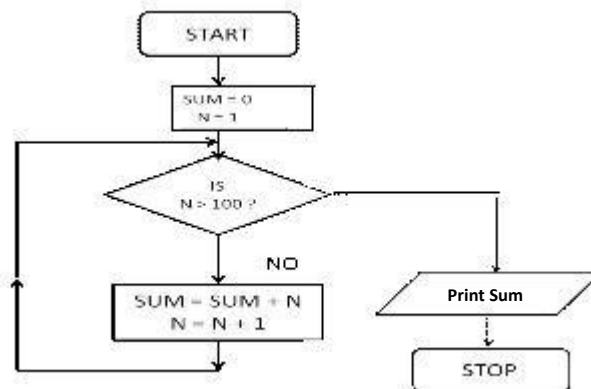
1. Draw the flowchart to find roots of quadratic equation $ax^2+bx+c=0$. The coefficients a, b, c are the input data



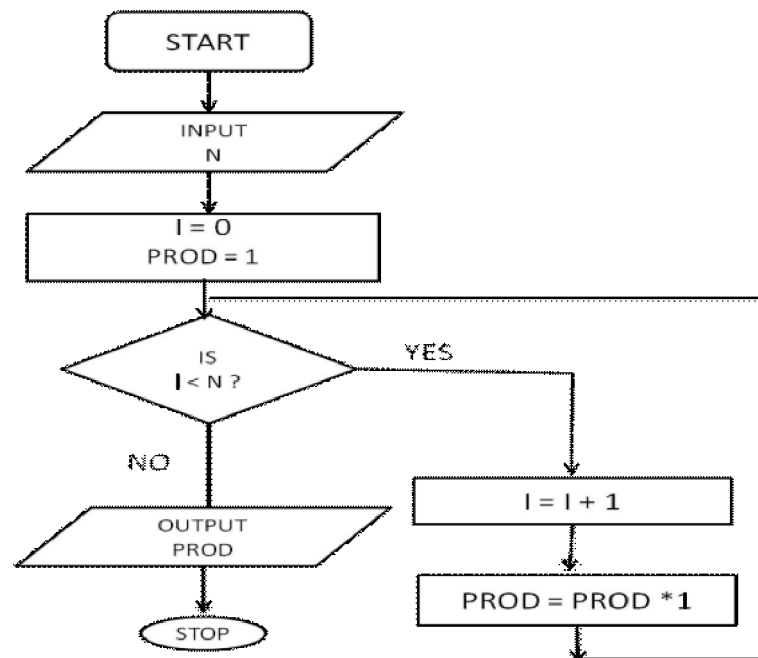
2. Draw a flowchart to find out the biggest of the three unequal positive numbers.



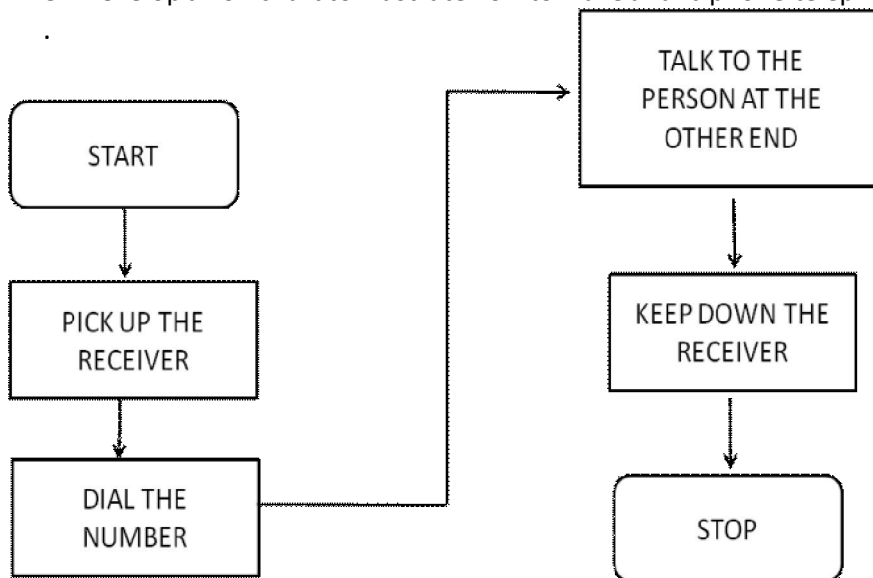
3. Draw a flowchart for adding the integers from 1 to 100 and to print the sum.



4. Draw a flowchart to find the factorial of given positive integer N.

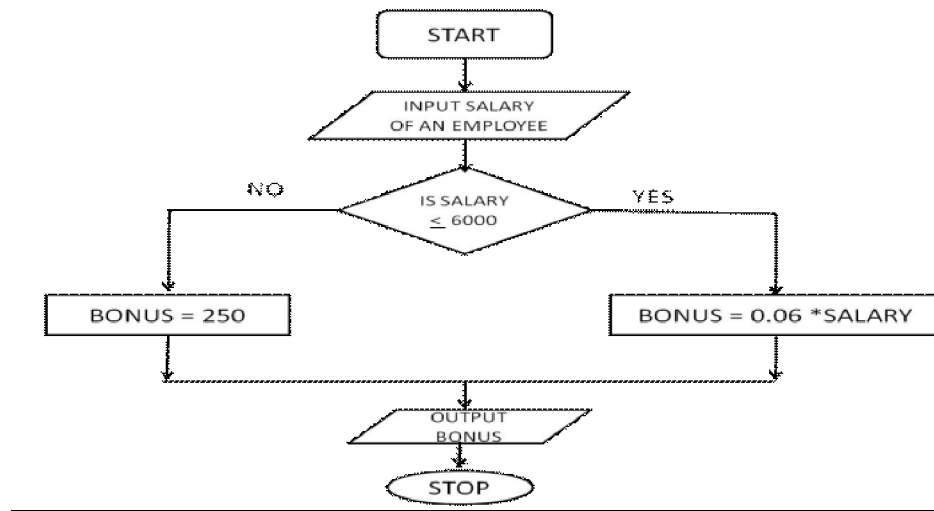


5. Develop a flowchart to illustrate how to make a Land phone telephone call



Flowchart for Telephone call

6. ABC company plans to give a 6% year-end bonus to each of its employees earning Rs 6,000 or more per month, and a fixed Rs 250/-- bonus to the remaining employees. Draw a flowchart for calculating the bonus for an employee



Pseudo code

The Pseudo code is neither an algorithm nor a program. It is an abstract form of a program. It consists of English like statements which perform the specific operations. It is defined for an algorithm. It does not use any graphical representation. In pseudo code, the program is represented in terms of words and phrases, but the syntax of program is not strictly followed.

Advantages: * Easy to read, * Easy to understand, * Easy to modify.

Example: Write a pseudo code to perform the basic arithmetic operations.

Read n1, n2

Sum = n1 + n2 Diff = n1 – n2 Mult = n1 * n2 Quot = n1/n2

Print sum, diff, mult, quot End.

1.3 From Algorithms to Program

'C' is high level language and is the upgraded version of another language (Basic Combined Program Language). C language was designed at Bell laboratories in the early 1970's by Dennis Ritchie. C being popular in the modern computer world can be used in Mathematical Scientific, Engineering and Commercial applications.

The most popular Operating system UNIX is written in C language. This language also has the features of low level languages and hence called as "System Programming Language"

Features of C language

- 1 Simple, versatile, general purpose language
- 2 It has rich set of Operators
- 3 Program execution are fast and efficient
- 4 Can easily manipulate with bits, bytes and addresses
- 5 Varieties of data types are available
- 6 Separate compilation of functions is possible and such functions can be called by any C program
- 7 Block- structured language

Character Set

The character set is the fundamental raw-material for any language. Like natural languages, computer languages will also have well defined character-set, which is useful to build the programs.

The C language consists of two character sets namely – source character set execution character set. Source character set is useful to construct the statements in the source program. Execution character set is employed at the time of execution of program.

1. **Source character set** : This type of character set includes three types of characters namely alphabets, Decimals and special symbols.
Alphabets : A to Z, a to z and Underscore(_)
Decimal digits : 0 to 9
Special symbols: + - * / ^ % = & ! () { } [] " ' etc
2. **Execution character set** : This set of characters are also called as non-graphic characters because these are invisible and cannot be printed or displayed directly. These characters will have effect only when the program being executed. These characters are represented by a back slash (\) followed by a character.

Execution character	Meaning	Result at the time of execution
\n	End of a line	Transfers the active position of cursor to the initial position of next line
\0 (zero)	End of string	Null
\t	Horizontal Tab	Transfers the active position of cursor to the next Horizontal Tab
\v	Vertical Tab	Transfers the active position of cursor to the next Vertical Tab
\f	Form feed	Transfers the active position of cursor to the next logical page
\r	Carriage return	Transfers the active position of cursor to the initial position of current line

Source Code:

The Complete structure of C program is

The basic components of a C program are:

- main()
- pair of braces { }
- declarations and statements
- user defined functions

Preprocessor Statements: These statements begin with # symbol. They are called preprocessor directives. These statements direct the C preprocessor to include header files and also symbolic constants in to C program. Some of the preprocessor statements are

#include<stdio.h>: for the standard input/output functions

#include<test.h>: for file inclusion of header file Test.

#define NULL 0: for defining symbolic constant NULL = 0 etc.

Global Declarations: Variables or functions whose existence is known in the main() function and other user defined functions are called global variables (or functions) and their declarations is called global declaration. This declaration should be made before main().

main(): As the name itself indicates it is the main function of every C program. Execution of C program starts from main (). No C program is executed without main() function. It should be written in lowercase letters and should not be terminated by a semicolon. It calls other Library functions user defined functions. There must be one and only one main() function in every C program.

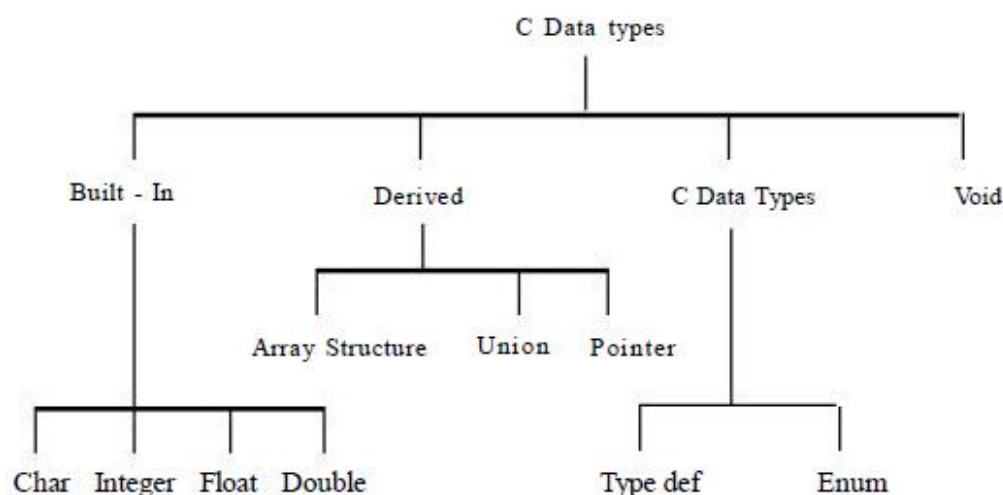
Braces: Every C program uses a pair of curly braces ({,}). The left brace indicates beginning of main() function. On the other hand, the right brace indicates end of the main() function. The braces can also be used to indicate the beginning and end of user-defined functions and compound statements.

Declarations: It is part of C program where all the variables, arrays, functions etc., used in the C program are declared and may be initialized with their basic data types.

Statements: These are instructions to the specific operations. They may be input-output statements, arithmetic statements, control statements and other statements. They are also including comments.

User-defined functions: These are subprograms. Generally, a subprogram is a function, and they contain a set of statements to perform a specific task. These are written by the user; hence the name is user-defined functions. They may be written before or after the main() function.

Variables:



The built-in data types and their extensions is the subject of this chapter. Derived data types such as arrays, structures, union and pointers and user defined data types such as typedef and enum will be discussed later.

Basic Data Types

There are four basic data types in C language. They are Integer data, character data, floating point data and double data types.

a. Character data: Any character of the ASCII character set can be considered as a character data types and its maximum size can be 1 byte or 8 byte long. 'Char' is the keyword used to represent character data type in C.

Char - a single byte size, capable of holding one character.

b. Integer data: The keyword 'int' stands for the integer data type in C and its size is either 16 or 32 bits. The integer data type can again be classified as

1. Long int - long integer with more digits
2. Short int - short integer with fewer digits.
3. Unsigned int - Unsigned integer

4. Unsigned short int – Unsigned short integer

5. Unsigned long int – Unsigned long integer

As above, the qualifiers like short, long, signed or unsigned can be applied to basic data types to derive new data types.

int - an Integer with the natural size of the host machine.

c. Floating point data: - The numbers which are stored in floating point representation with mantissa and exponent are called floating point (real) numbers. These numbers can be declared as 'float' in C.

float – Single – precision floating point number value.

d. Double data : - Double is a keyword in C to represent double precision floating point numbers.

double - Double – precision floating point number value.

Data Kinds in C

Various data kinds that can be included in any C program can fall in to the following.

a.Constants/Literals

b.Reserve Words Keywords

c.Delimiters

d.Variables/Identifiers

a. Constans/Literals: Constants are those, which do not change, during the execution of the program. Constants may be categorized in to:

- Numeric Constants
- Character Constants
- String Constants

1.Numeric Constants

Numeric constants, as the name itself indicates, are those which consist of numerals, an optional sign and an optional period. They are further divided into two types:

(a)Integer Constants (b) Real Constants

a.Integer Constants

A whole number is an integer constant Integer constants do not have a decimal point. These are further divided into three types depending on the number systems they belong to. They are:

i.Decimal integer constants

ii.Octal integer constants

iii.Hexadecimal integer constants

i.A decimal integer constant is characterized by the following properties

- It is a sequence of one or more digits ([0...9], the symbols of decimal number system).
- It may have an optional + or – sign. In the absence of sign, the constant is assumed to be positive.

- Commas and blank spaces are not permitted.

•It should not have a period as part of it. Some examples of valid decimal integer constants: 456

-123

Some examples of invalid decimal integer constants:

4.56 - Decimal point is not permissible

1,23 - Commas are not permitted

ii.An octal integer constant is characterized by the following properties

- It is a sequence of one or more digits ([0...7], symbols of octal number system).
- It may have an optional + or – sign. In the absence of sign, the constant is assumed to be positive.

- It should start with the digit 0.

- Commas and blank spaces are not permitted.
- It should not have a period as part of it. Some examples of valid octal integer constants:

0456
-0123
+0123

Some examples of invalid octal integer constants:

04.56 - Decimal point is not permissible 04,56 - Commas are not permitted
x34 - x is not permissible symbol 568 - 8 is not a permissible symbol

iii. An hexadecimal integer constant is characterized by the following properties

- It is a sequence of one or more symbols ([0...9][A....Z], the symbols of Hexadecimal number system).
- It may have an optional + or - sign. In the absence of sign, the constant is assumed to be positive.
- It should start with the symbols 0X or 0x.
- Commas and blank spaces are not permitted.
- It should not have a period as part of it.

Some examples of valid hexadecimal integer constants: 0x456

-0x123 0x56A
0XB78

Some examples of invalid hexadecimal integer constants: 0x4.56 - Decimal point is not permissible

0x4,56 - Commas are not permitted.

b. Real Constants

The real constants also known as floating point constants are written in two forms:

(i) Fractional form, (ii) Exponential form.

i. Fractional Form

The real constants in Fractional form are characterized by the following characteristics:

- Must have at least one digit.
- Must have a decimal point.
- May be positive or negative and in the absence of sign taken as positive.
- Must not contain blanks or commas in between digits.
- May be represented in exponential form, if the value is too higher or too low.

Some examples of valid real constants: 456.78

-123.56

Some examples of invalid real constants:

4.56 - Blank spaces are not permitted 4,56 - Commas are not permitted
456 - Decimal point missing

ii. Exponential Form

The exponential form offers a convenient way for writing very large and small real constant. For example, 56000000.00, which can be written as 0.56

*, 108 is written as 0.56E8 or 0.56e8 in exponential form. 0.000000234, which can be written as 0.234 * 10⁻⁶ is written as 0.234E-6 or 0.234e-6 in exponential form. The letter E or e stand for exponential form.

A real constant expressed in exponential form has two parts: (i) Mantissa part, (ii) Exponent part. Mantissa is the part of the real constant to the left of E or e, and the Exponent of a real constant is to the right of E or e. Mantissa and Exponent of the above two number are shown below.

Mantissa ↓	Exponent ↓	Mantissa ↓	Exponent ↓
0.56	E 8	0.234	E -6

In the above examples, 0.56 and 0.234 are the mantissa parts of the first and second numbers, respectively, and 8 and -6 are the exponent parts of the first and second number, respectively.

The real constants in exponential form and characterized by the following characteristics:

- The mantissa must have at least one digit.
- The mantissa is followed by the letter E or e and the exponent.
- The exponent must have at least one digit and must be an integer.
- A sign for the exponent is optional. Some examples of valid real constants: 3E4
23e-6
0.34E6

Some examples of invalid real constants: 23E - No digit specified for exponent

23e4.5 - Exponent should not be a fraction

23,4e5 - Commas are not allowed

256*e8- * not allowed

2.Character Constants

Any character enclosed with in single quotes (') is called character constant.

A character constant:

•May be a single alphabet, single digit or single special character placed with in single quotes.

- Has a maximum length of 1 character.

Here are some examples,

- 'C'
- 'c'
- '.'
- '*

3.String Constants

A string constant is a sequence of alphanumeric characters enclosed in double quotes whose maximum length is 255 characters.

Following are the examples of valid string constants:

- "My name is Krishna"
- "Bible"
- "Salary is 18000.00"

Following are the examples of invalid string constants:

My name is Krishna - Character are not enclosed in double quotation marks.

"My name is Krishna - Closing double quotation mark is missing.

'My name is Krishna' - Characters are not enclosed in double quotation marks

b. Reserve Words/Keywords

In C language, some words are reserved to do specific tasks intended for them and are called Keywords or Reserve words. The list reserve words are

auto	do	goto
break	double	if
case	else	int
char	extern	long
continue	float	register
default	for	return
short	sizeof	static
struct	switch	typedef
union	unsigned	void
while	const	enum
volatile	enum	noalias

C. Delimiters

This is symbol that has syntactic meaning and has got significance. These will not specify any operation to result in a value. C language delimiters list is given below

Symbol	Name	Meaning
#	Hash	Pre-processor directive
,	comma	Variable delimiter to separate variable
:	colon	label delimiter
;	Semicolon	statement delimiter
()	parenthesis	used for expressions
{ }	curly braces	used for blocking of statements
[]	square braces	used along with arrays

Variables / Identifiers

These are the names of the objects, whose values can be changed during the program execution. Variables are named with description that transmits the value it holds.

[A quantity of an item, which can change its value during the execution of program is called variable. It is also known as Identifier].

Rules for naming a variable:-

It can be of letters, digits and underscore(_)

First letter should be a letter or an underscore, but it should not be a digit.

Reserve words cannot be used as variable names. Example: basic, root, rate, roll-no etc are valid names.

Declaration of variables:

Syntax	type	Variable list
int	i, j	i, j are declared as integers
float	salary	salary is declared as floating point variable
Char	sex	sex is declared as character variable

Variables and Memory Locations

A variable is the name of a memory cell. It is "variable" because the value in the cell can change. Each memory cell has an address. High-level languages use a symbol table to map a variable name to the address it represents.

Here's a simplified example of what happens when a program is executed:

Program

$x = 2$

$y = x + 6$

Main Memory

Address	Value
0	2
4	8
8	

Symbol Table

x	0
y	4

The program puts data in contiguous memory. So the variable 'x' is at address 0, 'y' at address 4.

The symbol table maps variable names to addresses. So x is 0 to the computer, and y is the address 4.

Type Conversion in C

A type cast is basically a conversion from one type to another. There are two types of type conversion:

1. Implicit Type Conversion

Also known as 'automatic type conversion'.

Done by the compiler on its own, without any external trigger from the user.

Generally takes place when in an expression more than one data type is present. In such condition type conversion (type promotion) takes place to avoid loss of data.

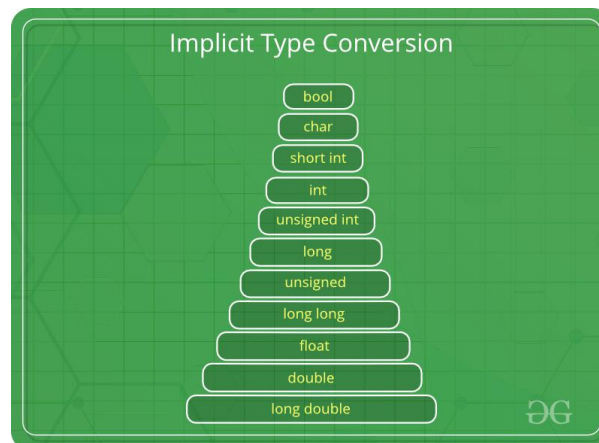
All the data types of the variables are upgraded to the data type of the variable with largest data type.

bool -> char -> short int -> int ->

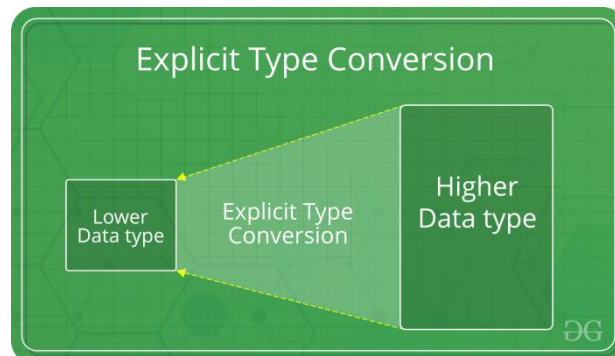
unsigned int -> long -> unsigned ->

long long -> float -> double -> long double

It is possible for implicit conversions to lose information, signs can be lost (when signed is implicitly converted to unsigned), and overflow can occur (when long long is implicitly converted to float).



2. Explicit Type Conversion



This process is also called type casting and it is user defined. Here the user can type cast the result to make it of a particular data type.

The syntax in C:

(type) expression

Type indicates the data type to which the final result is converted.

Runtime Environment:

By runtime, we mean a program in execution. Runtime environment is a state of the target machine, which may include software libraries, environment variables, etc., to provide services to the processes running in the system.

Storage Allocation

Runtime environment manages runtime memory requirements for the following entities:

Code : It is known as the text part of a program that does not change at runtime. Its memory requirements are known at the compile time.

Procedures : Their text part is static but they are called in a random manner. That is why, stack storage is used to manage procedure calls and activations.

Variables : Variables are known at the runtime only, unless they are global or constant. Heap memory allocation scheme is used for managing allocation and de-allocation of memory for variables in runtime.

Static Allocation

In this allocation scheme, the compilation data is bound to a fixed location in the memory and it does not change when the program executes. As the memory requirement and storage locations are known in advance, runtime support package for memory allocation and de-allocation is not required. When we declare a variable in C it follows static allocation i.e. Memory location is fixed by the compiler for declared variable.

Dynamic Allocation

The mechanism by which storage/memory/cells can be allocated to variables during the run time is called Dynamic Memory Allocation. Dynamic Memory Allocation in C is done using malloc(), calloc(), free() and realloc() functions.

Storage Classes:

A storage class defines the scope (visibility) and life-time of variables and/or functions within a C Program. They precede the type that they modify. We have four different storage classes in a C program –

Storage classes in C				
Storage Specifier	Storage	Initial value	Scope	Life
auto	stack	Garbage	Within block	End of block
extern	Data segment	Zero	global Multiple files	Till end of program
static	Data segment	Zero	Within block	Till end of program
register	CPU Register	Garbage	Within block	End of block

The auto Storage Class

The auto storage class is the default storage class for all local variables.

```
{  
    int mount;  
    auto int month;  
}
```

The example above defines two variables with in the same storage class. 'auto' can only be used within functions, i.e., local variables.

The register Storage Class

The register storage class is used to define local variables that should be stored in a register instead of RAM. This means that the variable has a maximum size equal to the register size (usually one word) and can't have the unary '&' operator applied to it (as it does not have a memory location).

```
{  
    register int miles;  
}
```

The register should only be used for variables that require quick access such as counters. It should also be noted that defining 'register' does not mean that the variable will be stored in a register. It means that it MIGHT be stored in a register depending on hardware and implementation restrictions.

The static Storage Class

The static storage class instructs the compiler to keep a local variable in existence during the life-time of the program instead of creating and destroying it each time it comes into and goes out of scope. Therefore, making local variables static allows them to maintain their values between function calls.

The static modifier may also be applied to global variables. When this is done, it causes that variable's scope to be restricted to the file in which it is declared.

In C programming, when static is used on a global variable, it causes only one copy of that member to be shared by all the objects of its class.

```
#include <stdio.h>

/* function declaration */
void func(void);

static int count = 5; /* global variable */

main() {
    while(count-->0) {
        func();
    }
    return 0;
}

/* function definition */
void func( void ) {

    static int i = 5; /* local static variable */
    i++;
    printf("i is %d and count is %d\n", i, count);
}
```

When the above code is compiled and executed, it produces the following result –

i is 6 and count is 4

i is 7 and count is 3

i is 8 and count is 2

i is 9 and count is 1

i is 10 and count is 0

The extern Storage Class

The extern storage class is used to give a reference of a global variable that is visible to ALL the program files. When you use 'extern', the variable cannot be initialized however, it points the variable name at a storage location that has been previously defined.

When you have multiple files and you define a global variable or function, which will also be used in other files, then extern will be used in another file to provide the reference of defined variable or function. Just for understanding, extern is used to declare a global variable or function in another file.

The extern modifier is most commonly used when there are two or more files sharing the same global variables or functions as explained below.

First File: main.c

```
#include <stdio.h>

int count ;

extern void write_extern();

main() {
    count = 5;
    write_extern();
}
```

Second File: support.c

```
#include <stdio.h>

extern int count;

void write_extern(void) {
    printf("count is %d\n", count);
}
```

Here, extern is being used to declare count in the second file, where as it has its definition in the first file, main.c. Now, compile these two files as follows –

```
$gcc main.c support.c
```

It will produce the executable program a.out. When this program is executed, it produces the following result –

```
count is 5
```

Syntax and Logical error in Compilation

Error is an illegal operation performed by the user which results in abnormal working of the program.

Programming errors often remain undetected until the program is compiled or executed. Some of the errors inhibit the program from getting compiled or executed. Thus errors should be removed before compiling and executing.

Syntax errors: Errors that occur when you violate the rules of writing C/C++ syntax are known as syntax errors. This compiler error indicates something that must be fixed before the code can be compiled. All these errors are detected by compiler and thus are known as compile-time errors.

Most frequent syntax errors are:

Missing Parenthesis (})

Printing the value of variable without declaring it

Missing semicolon

Logical Errors : On compilation and execution of a program, desired output is not obtained when certain input values are given. These types of errors which provide incorrect output but appears to be error free are called logical errors. These are one of the most common errors done by beginners of programming.

These errors solely depend on the logical thinking of the programmer and are easy to detect if we follow the line of execution and determine why the program takes that path of execution.

Object code: object code is a program or a file that is created after compiling the source code. A programmer writes a program using a programming language. The computer does not understand this program. Therefore, the compiler converts this source code into an object code. The object code is a binary file, and the computer understands this file

Executable code: executable code is a file or a program that indicates tasks according to encoded instructions the CPU can directly execute. The CPU can directly execute an executable code.