COMPUTER:

A Computer is an electronic device which performs operations such as accepts data as an input, store the data, manipulate or process the data and produce the result as an output.
Main task performed by a computer

- Accept the data
- Process or manipulate the data
- Display or store the result in the form of human understanding
- Store the data, instructions and results.

Advantages of Using Computers:

- **Speed:** Computers can carry out instructions in less than a millionth of a second.
- **Accuracy:** Computers can do the calculations without errors and very accurately.
- **Diligence:** Computers are capable of performing any task given to them repetitively.
- **Storage Capacity:** Computers can store large volume of data and information on magnetic media.

A Computer is a system made up of two major components:

1. Hardware
2. Software

The computer hardware is the physical equipment.
The software is the collection of programs that allow the hardware to do its job.

Hardware:

The hardware component of the computer system consists of five parts.

- Input devices
- Output devices
- Central processing unit
- Primary storage
- Auxiliary storage devices

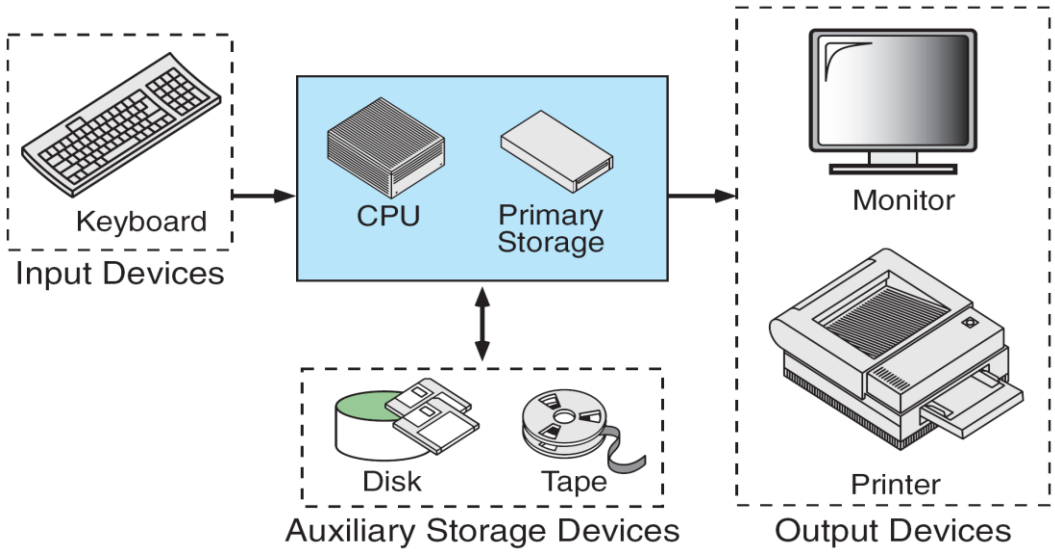


Fig: Computer

Input Devices

- These are used to enter data and programs into the computer
- These are for man to machine communication

  Ex: Keyboard, mouse, scanner, touch screen, audio input.

Output Devices

- These are used to get the desired output from the computer
- These are for machine to man communication

  Ex: Printer, Monitor, Speakers
- If the output is shown on monitor then it is called "**Soft copy**"
- If the output is printed on a paper using printer then it is called "**hard copy**"

CPU (Central Processing Unit)

- It is responsible for processing the instructions
- It consists of 3 parts

  1) ALU – Arithmetic & Logic Unit
  2) CU- Control Unit
  3) Memory
- ALU performs arithmetic operations like addition, subtraction, multiplication, division and logical operations like comparisons among data
- CU is responsible for movement of data inside the system
- Memory is used for storage of data and programs. It is divided into 2

  1) Primary Memory / Main Memory
  2) Secondary Memory/ Auxiliary Memory

1) Primary Memory

- It is also called main memory
- Data is stored temporarily i.e. data gets erased when computers is turned off

Ex: RAM

2) Secondary Memory

- It is also called as auxiliary memory
- Data is stored permanently so that user can reuse the data even after power loss.

Ex: Hard disk, CD, DVD, Floppy etc.

Software:

Software is a set of program or instructions given to the computer to carry out certain jobs or tasks. It is set of programs which actually makes the computer useful. The software can be classified into three categories:

System Software: System software is responsible for managing a variety of independent hardware components, so that they can work together. Its purpose is to unburden the application software programmer from the often complex details of the particular computer being used, including such accessories as communications devices, printers, device readers, displays and keyboards, and also to partition the computer's resources such as memory and processor time in a safe and stable manner.

Ex: Device drivers, Operating systems, Servers

Programming Software: Programming Software usually provides tools to assist a programmer in writing computer programs, and software using different programming languages in a more convenient way. The tools include:

- Compilers

- Debuggers
- Interpreters
- Linkers
- Text editors

Application Software:

Application software is broken into two classes: general-purpose and application-specific software.

General purpose software

These are purchased from software developers

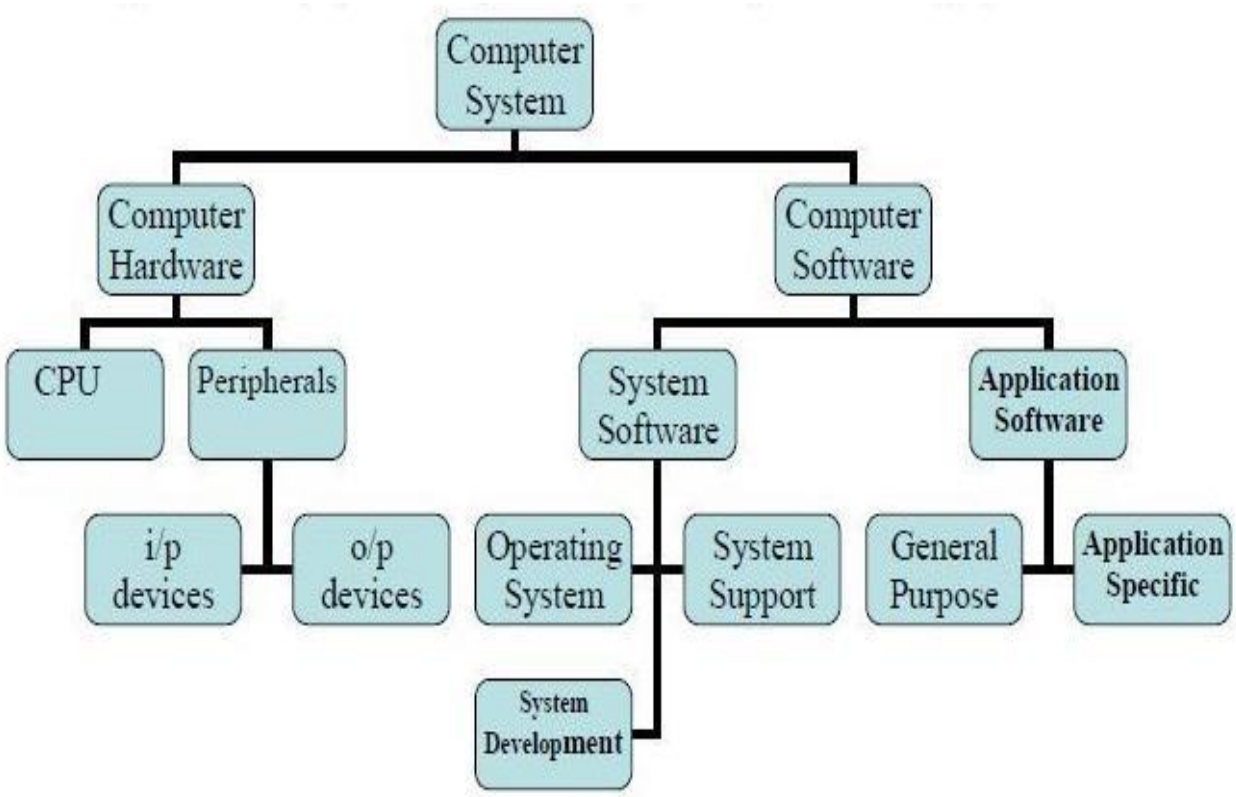
They can be used for more than one application

Ex: word processors, DBMS etc…

Application Specific Software

Application software is developed to aid in any task that benefits from computation. It is a broad category, and encompasses Software of many kinds, including the internet browser being used to display this page. This category includes:

- Computer aided design
- Databases
- Educational software
- Image editing
- Medical software
- Video games
- Word processing



COMPUTER ENVIRONMENTS

- In early days, there was only one environment i.e. the mainframe computer. It occupies much space and consumed large amount of power.
- With the invention of transistor and IC (Integrated Circuits) technology, mini computers were introduced in which many components are fused together.
- After invention of microprocessor technology, the environment changed resulting in minicomputers and personal computers
- The computing environments are classified into **4 types**. They are
    1) Personal computing environment
    2) Time sharing environment
    3) Client – server environment
    4) Distributed computing environment

1) Personal computing environment

- In a personal computer, all the basic elements of the CPU are combined into the Microprocessor and all the hardware components are tied together.
- The whole computer can be used by the user to perform his task independently.

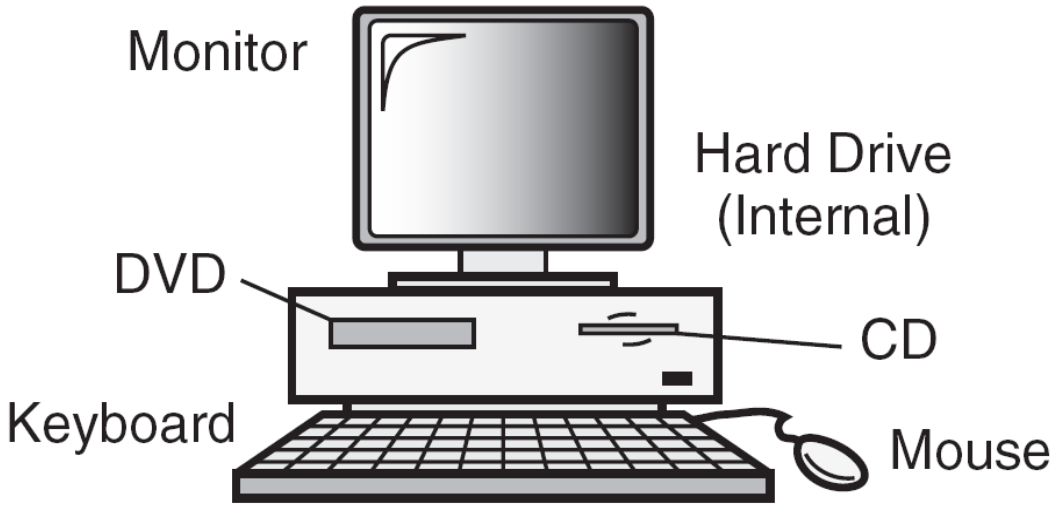


Fig: personal computing environment

Disadvantage:

- Much of the CPU time is wasted

2) Time- Sharing environment

- For effective usage of CPU time, many users are connected to one (or) more computers to form a time- sharing environment.
- There will be a central computer which performs the following:
    - controls the shared resources
    - manages shared data & printing
    - does the computing

Advantage:

- Computer is busy always reducing CPU idle time

Disadvantage:

- Response becomes slow as the number of users connected to the central computer increases

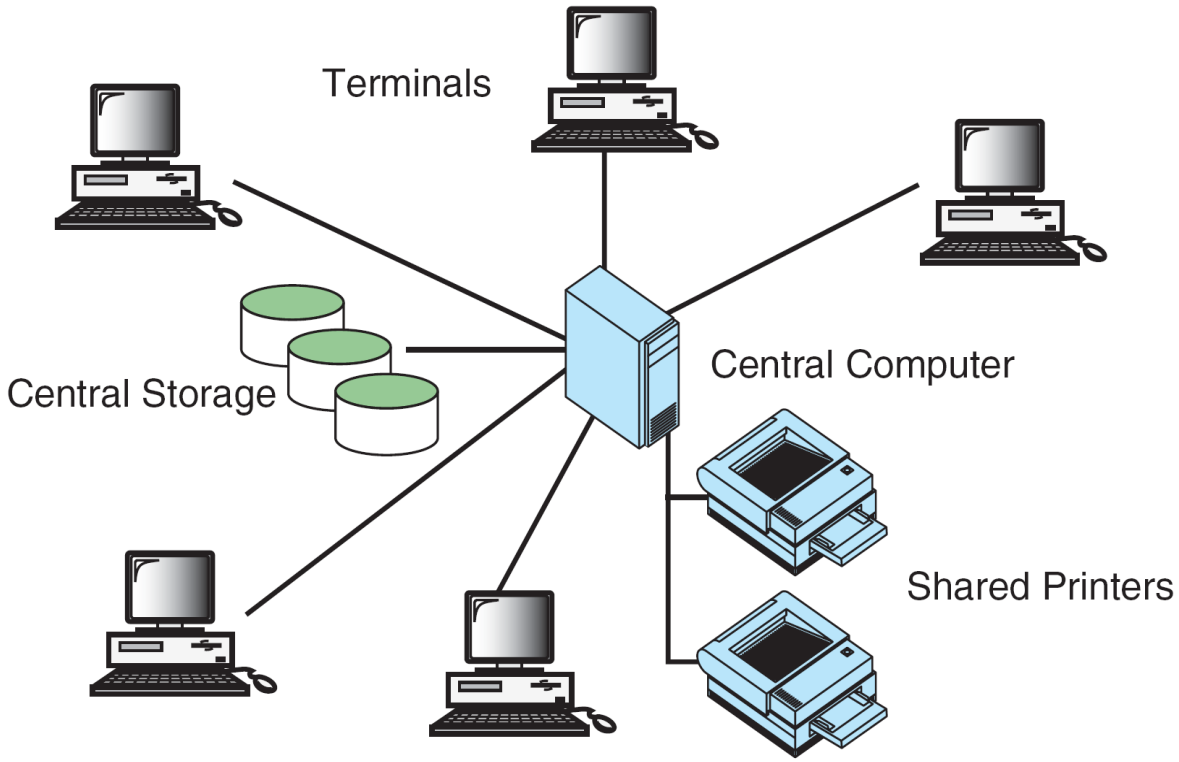


Fig: Time-sharing environment

3) Client – Server environment

- This environment splits the computing function between a central computer and user computer
- Users are given personal computer (PC's) on workstations so that some of computation responsibility can be moved from central computer and assigned to the workstation
- In the client/server environment, the user's microcomputers or workstations are called client. The central computer, which may be a powerful microcomputer, minicomputer, or central mainframe systems, is known as server.

Advantages

- Response time is faster
- Users are more productive

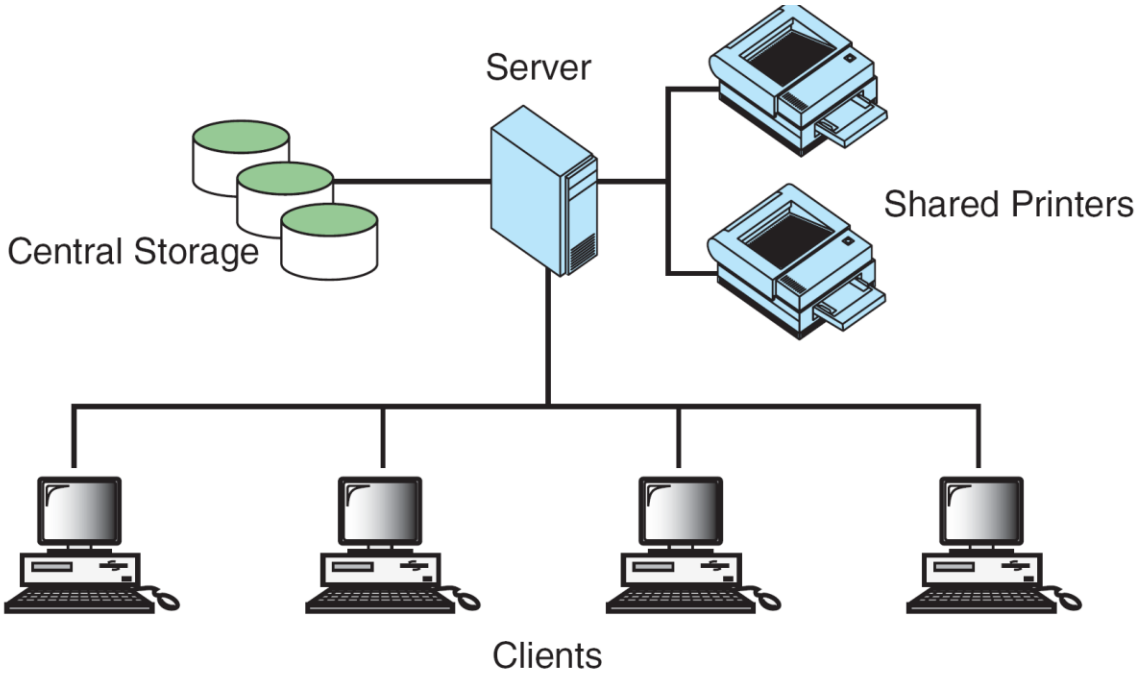


Fig: Client-server environment

4) Distributed computing environment

- It Provides integration of computing functions between different servers and clients
- "Internet" provides connectivity to different servers throughout the world.
- It provides
  -- reliable
  -- scalable
  -- highly available network

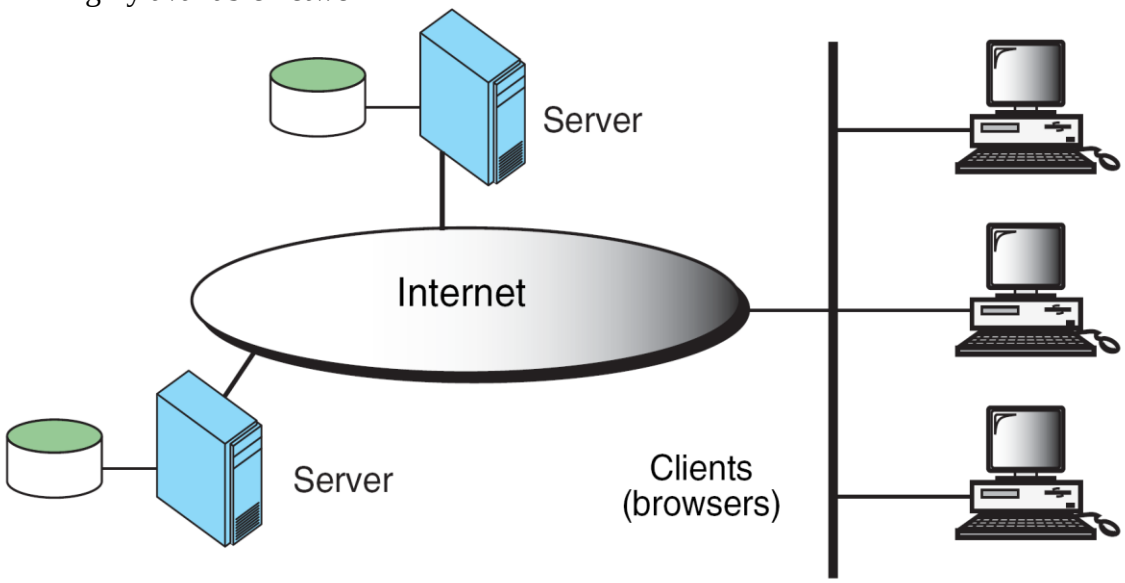


Fig: Distributed computing environment

COMPUTER LANGUAGES

- Computer programming languages are used to give instructions to the computer in a language which computer understands
- Computer languages are classified into 3 types
  1) Machine languages
  2) Symbolic languages
  3) High level languages

1) Machine languages

- Computer is a machine and since its memory can store only 1's and 0's, instructions must be given to the computer in streams of 1's and 0's i.e. binary code.
- These are easily understandable by the machine
- Programs written in binary code can be directly fed to computer for execution and it is known as machine language.

Advantage:

- Execution is very fast since there is no need of conversion

Disadvantage:

- Writing and reading programs in machine language is very difficult
- Machine instructions are difficult to remember

2) Symbolic Languages

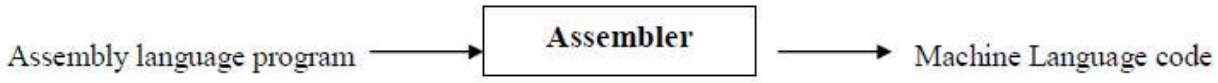
- It is also called as assembly language
- An assembly program contains "Mnemonics"
- "Mnemonic" means information can be memorized easily and is generally in the form of abbreviations.

Advantage:

- Easy to read and write programs in assembly language when compared to machine language
- Mnemonics are easy to remember

Disadvantage:

- Assembly programs are machine dependent
- Execution is slow since it needs conversion into machine language
- "Assembler" is used to convert assembly language program into machine language.



3) High level languages

- A set of languages which are very close to our native languages are called "high-level languages".
- High level languages have control structures, I/o facilities, hardware independence

Ex: FORTRAN, COBOL, PASCAL, C, C++ etc.

Advantage:

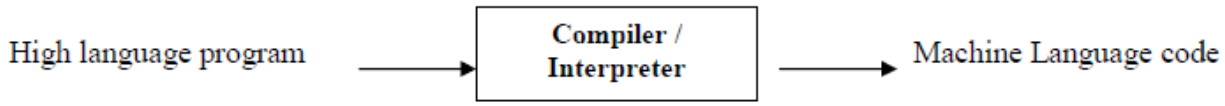
i) Machine independence i.e. programs are "Portable" i.e. programs can be moved from one system to another

ii) Easy to learn and understand

iii) Takes less time to write programs

Disadvantage:

- High level language programs needs a translator for conversion into machine language
- 'Compilers' (or) 'Interpreters' are used for converting high level language program into machine language.



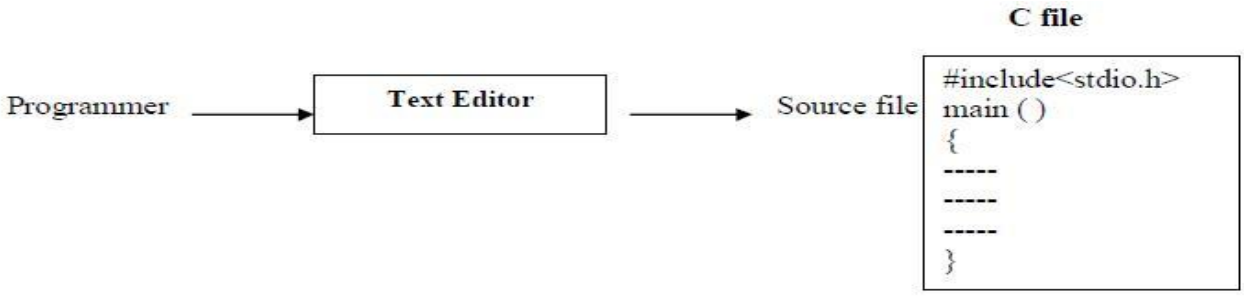
- Compiler converts entire statements in the program at a time.
- Interpreter converts one statement at a time.

CREATING AND RUNNING PROGRAMS

- Program consists of set of instructions written in a programming language
- The job of a programmer is to write and test the program.
- There are **4 steps** for converting a 'C' program into machine language.
    1) Writing and editing the program
    2) Compiling the program
    3) Linking the program
    4) Executing the program

1) Writing and editing the program

- '**Text editors**' are used to write programs.
- Users can enter, change and store character data using text editors
- Special text editor is often included with a compiler
- After completing writing program, the file is saved to disk. It is known as **'source file'**
- This file is input to the compiler



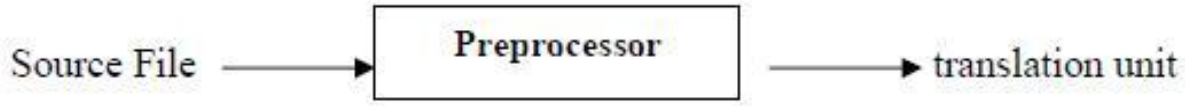
2) Compiling the program

- "**Compiler**" is a software that translates the source file into machine language
- The 'C' compiler is actually 2 separate programs
    A) Preprocessor
    B) Translator

A) Preprocessor

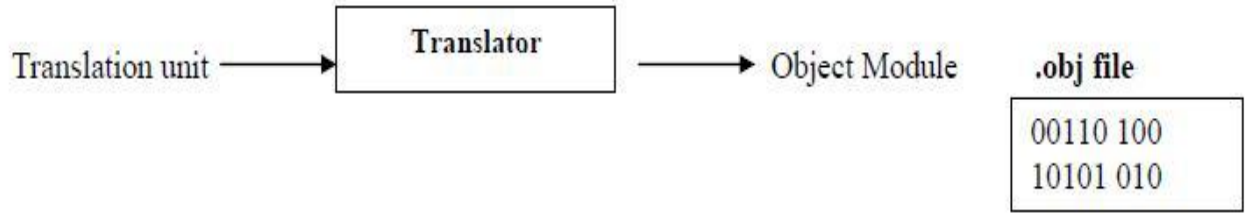
- It reads the source code and prepares it for the translator
- It scans for special instructions known as 'preprocessor' commands

- These commands tell the preprocessor to look for special code libraries and make substitutions
- The result of preprocessing is called 'translation' unit



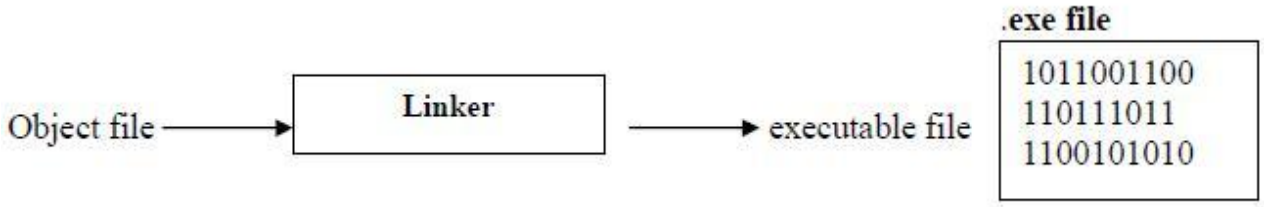
B) Translator

- It does the actual work of converting the program into machine language
- It reads the translation unit and results in '**object module'** i.e., code in machine language
- But it is not yet executable because it does not have the 'C' and other functions included.



3) Linking programs

- '**Linker**' assembles input /output functions, mathematical library functions and some of the functions that are part of source program into final executable program
- It is called as executable file that it is ready for execution



4) Executing Programs

- '**Loader**' is the software that gets the program that is ready for execution into the memory when everything is loaded, the program takes control and the '**Runner**' begins its execution.
- In the process of execution, the program reads the data from the user, processes the data and prepares the output

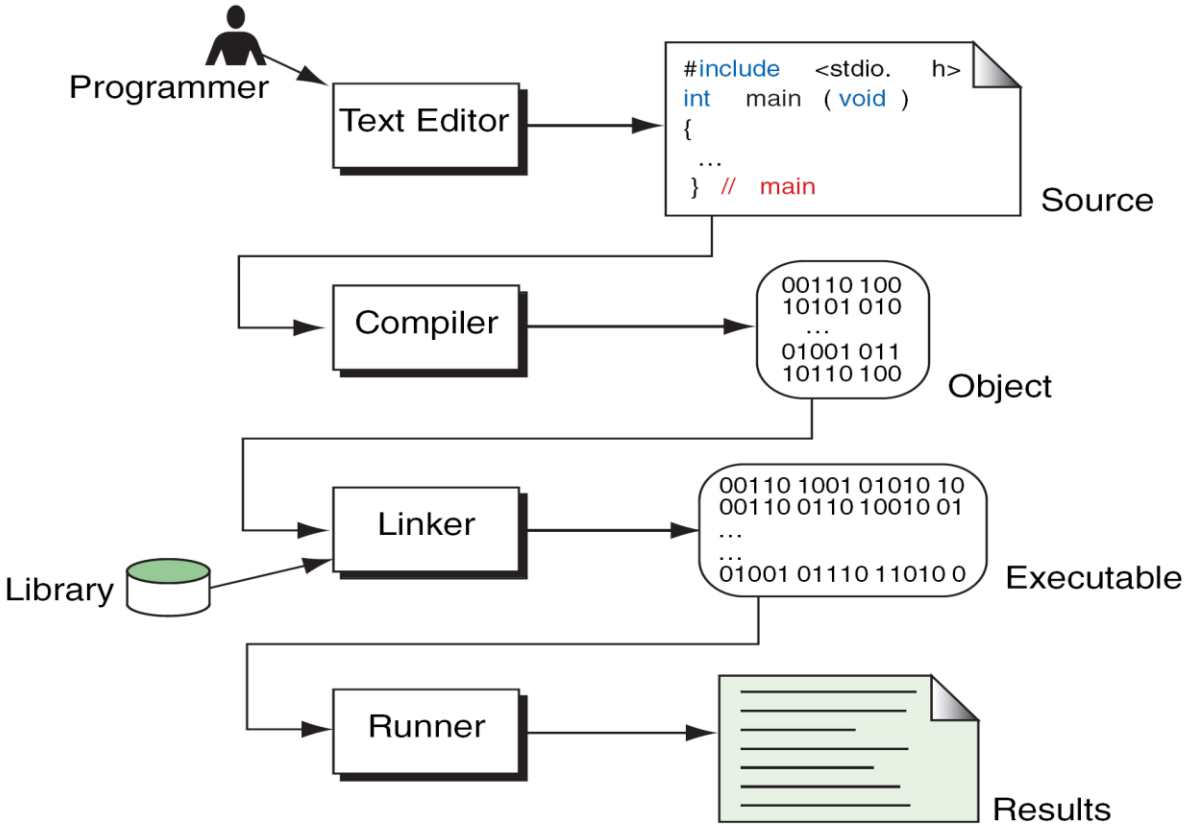


FIG: Building a C Program

INTRODUCTION TO 'C' LANGUAGE

- 'C' is a high level programming language developed in **1972 by Dennis Ritchie at AT & T Bell Laboratories**
- It was primarily developed for systems programming i.e. for designing operating systems compilers etc

IMPORTANCE OF 'C' LANGUAGE

1. It is a robust language, whose rich set of built-in functions and operations can be used to write any complex program.
2. It is a middle level language because the 'C' compiler combines the capabilities of an assembly language with the features of a high-level language and therefore it is well suited for writing both system software and business packages.
3. 'C' Programs are efficient and fast
4. C is highly portable, that is c programs written for one computer can be run on another with little (or) no modification.
5. 'C' Language is well suited for structured programming, thus requiring the user to think of a problem in terms of function modules (or) blocks.
6. 'C' program has the ability to extend itself.

Program Development Steps:

1. Statement of Problem
a) Working with existing system and using proper questionnaire, the problem should be explained clearly.
b) What in puts are available, what outputs are required and what is needed for creating workable solution, should be understood clearly.
2. Analysis
a) The method of solutions to solve the problem can be identified.
b) We also judge that which method gives best results among different methods of solution.
3. Design
a) Algorithms and flow charts will be prepared.
b) Focus on data, architecture, user interfaces and program components.
4. Implementation
The algorithms and flow charts developed in the previous steps are converted into actual programs in the high level languages like C.
a. Compilation
The process of translating the program into machine code is called as Compilation. Syntactic errors are found quickly at the time of compiling the program. These errors occur due to the usage of wrong syntaxes for the statements.
Eg: x=a*y+b
There is a syntax error in this statement, since, each and every statement in C language ends with a semicolon (;).
b. Execution
The next step is Program execution. In this phase, we may encounter two types of errors.
Runtime Errors: these errors occur during the execution of the program and terminate the program abnormally.
Logical Errors: these errors occur due to incorrect usage of the instructions in the program. These errors are neither detected during compilation or execution nor cause any stoppage to the program execution but produces incorrect output.

General Structure of a C program:

```
/* Documentation section */

/* Link section */

/* Global declaration section */

main() function
{
Local Declaration part
Executable part (statements)
}

/*User Defined function definitions */
```

Documentation:

The documentation section is used for displaying any information about the program like the purpose of the program, name of the author, date and time written etc, and this section should be enclosed within comment lines. The statements in the documentation section are ignored by the compiler.
C uses 2 different formats:

Line commenting: it uses two slashes (//) to identify the comment and it is used to comment entire line.

Block Commenting: it uses two pair of tokens /* and */ to comment entire block of code.

Link section:

The link section consists of the inclusion of header files.
Ex: #include <stdio.h>,#include <conio.h>

Global declaration section:

Anything declared in the global declaration section is accessible throughout the program, i.e. accessible to all the functions in the program.

Main() function:

main() function is mandatory for any program and it includes two parts, the declaration part and the executable part.

User Defined function definitions:
This section is optional and used when we require including user defined functions in the program.

C Language Components

The four main components of C language are

1) The Character Set.
2) Variables
3) Data Types
4) Tokens

1) **The Character Set:** Character set is a set of valid characters that a language cab recognizes. A character represents any letter, digit or any other sign.
   - Letters - A,B,C……Z or a,b,c ……z.
   - Digits - 0,1……9
   - Special Symbols - ~!@#$%^&.....
   - White Spaces - Blank space, horizontal tab, carriage return, new line, form feed.

2) Variables:

It is the name given to a memory location that may be used to store a data value

A variable may take different values at different times during execution

A variable name may be chosen by the programmer in a meaningful way so as to reflect its Function (or) nature in the program

Eg: sum, avg , total etc.

Rules for naming a variable:

1) They must begin with a letter
2) The length of the variable must not exceed 31 characters in ANSI standard. But first eight characters are treated as significant by many compilers
3) Upper and lowercase characters are different Eg:
   total, TOTAL, Total are 3 different variables
4) The variable name should not be a keyword
5) White space is not allowed

2) Data Types:

Data type specifies the set of values and the type of data that can be stored in a variable. They allow the programmer to select the type appropriate to the needs of application.

Types :

1) Primary data types
2) Derived data types

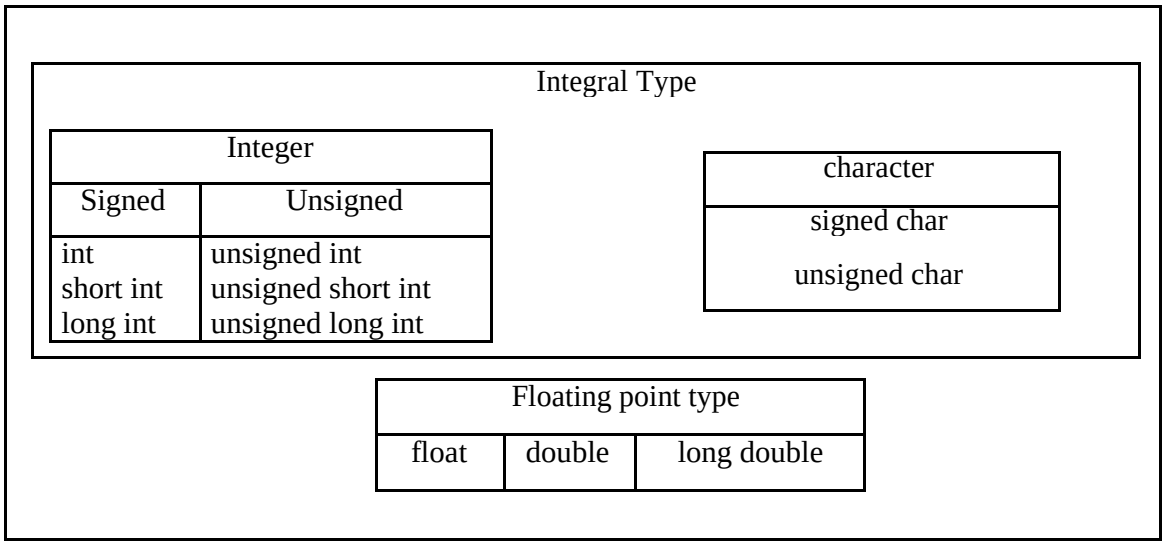
1. Primary data types

'C' compilers support 4 fundamental data types

They are

1) integer
2) character
3) Floating – point
4) Double precision floating point

PRIMRARY DATA TYPES



1. Integral data type

Integral data types are used to store whole numbers and characters It is further classified into

a) integer data type

b) character data type

a) Integer data type

This data type is used to store whole numbers

It has 3 classes of integer storage namely, short int, int and long int in both signed and unsigned forms

Type	size (in bytes)	Range	Control string
Short int (or) signed short int	1	-128 to 127	% h
Unsigned short int	1	0 to 255	% h
int (or) signed int	2	-32768 to 32767	% d
unsigned int	2	0 to 65535	% u
Long int (or) signed long int	4	-2147483648 to 2147483647	%ld
Unsigned long int	4	0 to 4294967295	%lu

b) character data type

This data type is used to store characters

These characters are internally stored as integers

Each character has an equivalent ASCII value

eg: 'A' has ASCII value 65

Character data type

Type	Size (in bytes)	Range	Control string
Char (or) signed char	1	- 128 to +127	%c
Unsigned char	1	0 to 255	%c

2. Floating – point Data types

It is used to store real numbers (i.e., decimal point numbers). For 6 digits of accuracy, 'float' is used.

For 12 digits of accuracy, 'double' is used.

For more than 12 digits of accuracy, 'long double' is used..

Floating point data type

Type	Size (in bytes)	Range	Control string
float	4	3.4 E – 38 to 3.4 E + 38	%f
double	8	1.7 E – 308 to 1.7 E +308	%lf
long double	10	3.4 E – 4932 to 1.1 E +4932	%Lf

Variable declaration

Syntax:

Datatype v1,v2,… vn;

Where v1, v2,...vn are names of variables

eg: int sum;

float a,b;

Variable can be declared in 2 ways

1. local declaration
2. global declaration

'local declaration' is declaring a variable inside the main block and its value is available within that block

'global declaration' is declaring a variable outside the main block and its value is available through out the program.

Eg :

```
int a, b;        /* global declaration*/
main ( )
{
```

```
int c; /* local declaration*/
-    -    -
-    -    -
}
```

Tokens: The smallest individual unit in a program is known as a token. C has five tokens

i. Keywords
ii. Identifiers
iii. Constants
iv. Punctuations
v. Operators

i. Keywords: Keywords are reserved word in C. They have predefined meaning cannot changed. All keywords must be written in lowercase. Eg:- auto, long, char, short etc.

ii. Identifiers: - Identifiers refer to the names of variable, functions and arrays. These are user-defined names. An identifier in C can be made up of letters, digits and underscore. Identifiers may start with either alphabets or underscore. The underscore is used to make the identifiers easy to read and mark functions or library members.

iii. Constants: - Constants in C refers to fixed values that do not change during the execution of a program. C support several types of constants.

a. Numerical Constants
    i. Integer Constant
        1. Decimal Constant
        2. Octal Constant
        3. Hexadecimal Constant
    ii. Float Constant
b. Character Constants
    i. Single Character Constant
    ii. String Constant

Integer Constant: - An integer constant is a whole number without any fractional part. C has three types of integer constants.

Decimal Constant: - Decimal integers consists of digits from 0 through 9
Eg.: 34,900,3457,-978

Octal Constant: - An Octal integer constant can be any combination of digits from 0 through 7. In C the first digit of an octal number must be a zero(0) so as to differentiate it from a decimal number.
Eg.: 06,034,-07564

Hexadecimal Constant: Hexadecimal integer constants can be any combination of digits

0 through 9 and alphabets from „a through „f or „A through „F. In C, a hexadecimal constant must begin with 0x or 0X (zero x) so as to differentiate it from a decimal number. Eg:-0x50, 0XAC2 etc

Floating Constants (Real): Real or floating point numbers can contain both an integer part and a fractional part in the number. Floating point numbers may be represented in two forms, either in the fractional form or in the exponent form.

A float point number in fractional form consists of signed or unsigned digits including decimal point between digits. E.g:- 18.5, .18 etc.

Very large and very small numbers are represented using the exponent form. The exponent notation use the „E or „e symbol in the representation of the number. The number before the „E is called as mantissa and the number after forms the exponent.

Eg.:-$5.3E^{-5}$,$-6.79E^{3}$,$78e^{05}$

Single Character Constant: - A character constant in usually a single character or any symbol enclosed by apostrophes or single quotes. Eg.: ch='a'

String Constant: - A sequence of character enclosed between double quotes is called string constant. Eg.: "Hello Good Morning"

Backslash Character Constants [Escape Sequences]:

Backslash character constants are special characters used in output functions. Although they contain two characters they represent only one character. Given below is the table of escape sequence and their meanings.

Constant	Meaning
'**\a**'	.Audible Alert (Bell)
'**\b**'	.Backspace
'**\f**'	.Formfeed
'**\n**'	.New Line
'**\r**'	.Carriage Return
'**\t**'	.Horizontal tab
'**\v**'	.Vertical Tab
'\''	.Single Quote
'\"'	.Double Quote
'**\?**'	.Question Mark
'\\'	.Back Slash
'**\0**'	.Null

iv) Punctuations: - 23 characters are used as punctuations in C. eg: + _ / ; : > ! etc

OPERATORS AND EXPRESSIONS

Operator performs an operation on data Operators are classified into

1. Arithmetic operators.
2. Relational operators.
3. Logical operators.
4. Assignment operators.
5. Increment and decrement operators.
6. Bitwise operators.
7. Conditional operators.
8. Special operators.

1). Arithmetic operator

These operators are used for numerical calculations (or) to perform arithmetic operations like addition, subtraction etc.

Operator	Description	Example	a =20, b=10	output
+	Addition	a+b	20+10	30
-	Subtraction	a-b	20-10	10
*	multiplication	a*b	20*10	200
/	Division	a/b	20/10	2 (quotient)
%	Modular division	a%b	20%10	0 (remainder)

Integer Arithmetic:
When an arithmetic operation is performed on two whole numbers or integers than such an operation is called as integer arithmetic. It always gives an integer as the result. Let x = 27 and y = 5 be 2 integer numbers. Then the integer operation leads to the following results.
x + y = 32 , x – y = 22 , x * y = 115 , x % y = 2 , x / y = 5
In integer division the fractional part is truncated.

Floating point arithmetic:
When an arithmetic operation is preformed on two real numbers or fraction numbers such an operation is called floating point arithmetic. The floating point results can be truncated according to the properties requirement. The remainder operator is not applicable for floating point arithmetic operands.
Let x = 14.0 and y = 4.0 then
x + y = 18.0, x – y = 10.0, x * y = 56.0, x / y = 3.50

Mixed mode arithmetic:
When one of the operand is real and other is an integer and if the arithmetic operation is carried out on these 2 operands then it is called as mixed mode arithmetic. If any one operand is of real type then the result will always be real thus 15/10.0 = 1.5

2). Relational operators:

These are used for comparing two expressions.

. C supports the following relational operators.

Operator	Meaning
<	is less than
<=	is less than or equal to
>	is greater than
>=	is greater than or equal to
==	is equal to
!=	is not equal to

It is required to compare the marks of 2 students, salary of 2 persons, we can compare them using relational operators.

Operator	Description	Examble	a =10, b=20	output
<	less than	a<b	10<20	1
<=	less than (or) equal to	a<=b	10< = 20	1
>	greater than	a>b	10>20	0
>=	greater than (or) equal to	a>=b	10> =20	0
= =	equal to	a= =b	10 = = 20	0
! =	not equal to	a! = b	10 ! =20	1

The output of a relational expression is either true (1) (or) false (0)

3. Logical Operators

These are used to combine 2 (or) more expressions logically

They are logical AND (&&) logical OR (||) and logical NOT (!)

Logical AND (&&)

exp1	exp2	exp1&&exp2
T	T	T
T	F	F
F	T	F
F	F	F

Logical OR(||)

exp1	exp2	exp1||exp2
T	T	T
T	F	T
F	T	T
F	F	F

Logical NOT (!)

exp	!(exp)
T	F
F	T

Operator	Description	Example	a =10, b=20,c=30	output
&&	logical AND	(a>b) && (a<c)	(10>20) & & (10<30)	0
||	logical OR	(a>b) | | (a<c)	(10>20) ||(10<30)	1
!	logical NOT	! (a>b)	! (10>20)	1

4. Assignment operators

It is used to assign a value to a variable

Types

1) simple assignment
2) compound assignment

Operator	Description	Examble
=	Simple assignment	a=10
+ =, - =, * =, / =, %=	Compound assignment	a+=10 ⇨ a=a+10 a-=10 ⇨ a=a-10

5. Increment and decrement operator:

a) Increment operator (++):

It is used to increment the value of a variable by 1 2 types : i) pre increment

ii) post increment

increment operator is placed before the operand in pre increment and the value is first incremented and then operation is performed on it.

eg: z = ++a; a= a+1
z=a

increment operator is placed after the operand in post increment and the value is incremented after the operation is performed

eg: z = a++; z=a
a= a+1

b) Decrement operator : (- -)

It is used to decrement the values of a variable by 1 2 types : i) pre decrement

ii) post decrement

decrement operator is placed before the operand in predecrement and the value is first decremented and then operation is performed on it.

eg: z = - - a;  a= a-1
z=a

decrement operator is placed after the operand in post decrement and the value is decremented after the operation is performed

eg: z = a--;  z=a
a= a-1

6. Bitwise Operators:

Unlike other operators, bitwise operators operate on bits (i.e. on binary values of on operand)

Operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Left shift
>>	Right shift

Bitwise AND		
a	b	a &b
0	0	0
0	1	0
1	0	0
1	1	1

Bitwise OR		
a	b	a | b
0	0	0
0	1	1
1	0	1
1	1	1

Bitwise XOR		
a	b	a ^b
0	0	0
0	1	1
1	0	1
1	1	0

eg: let a= 12, b=10

a&b

	8	4	2	1
a =12	1	1	0	0
b =10	1	0	1	0
a &b	1	0	0	0

a&b = 8

a | b

	8	4	2	1
a =12	1	1	0	0
b =10	1	0	1	0
a | b	1	0	1	0

a | b = 14

a ^ b

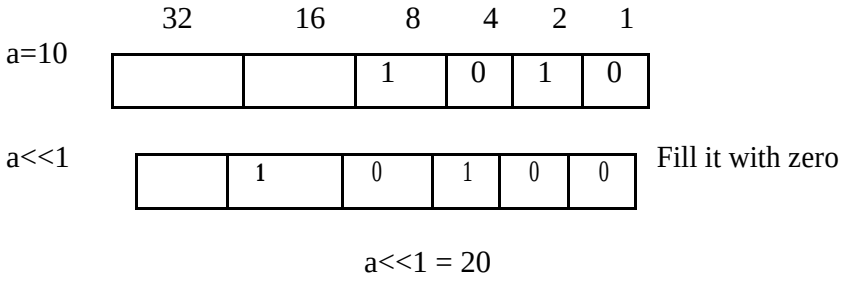
	8	4	2	1
a =12	1	1	0	0
b =10	1	0	1	0
a ^b	0	1	1	0

a ^ b = 6

Left Shift

If the value of a variable is left shifted one time, then its value gets doubled

eg: a = 10 then a<<1 = 20

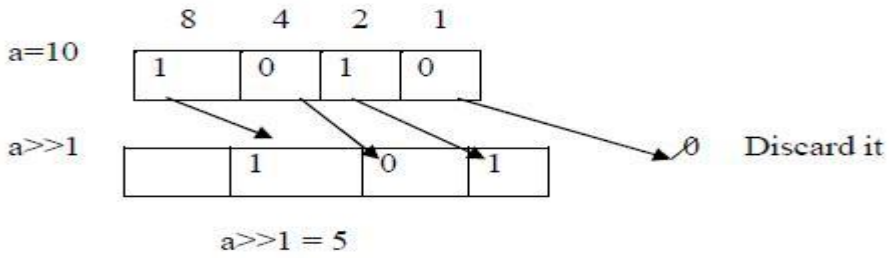


a<<1 = 20

Right shift

If the value of a variable is right shifted one time, then its value becomes half the original value

eg: a = 10 then a>>1 = 5



a>>1 = 5

7. Conditional operator (? :)

It is also called ternary operator

Syntax:

exp1 ? exp2 : exp3

if exp1 is true epx2 is evaluated, otherwise exp3 is evaluated it can also be represented in if – else form

```
if( exp1)
exp2;
else
exp3;
```

8) special operations

Some of the special operations are coma, ampersand (&), size of operators.

a) coma: (,)

It is used as separator for variables

eg; a=10, b=20

b) Address:(&)

It is used to get the address of a variables.

C) Size of ( ) :

It is used to get the size of a data type of a variable in bytes.

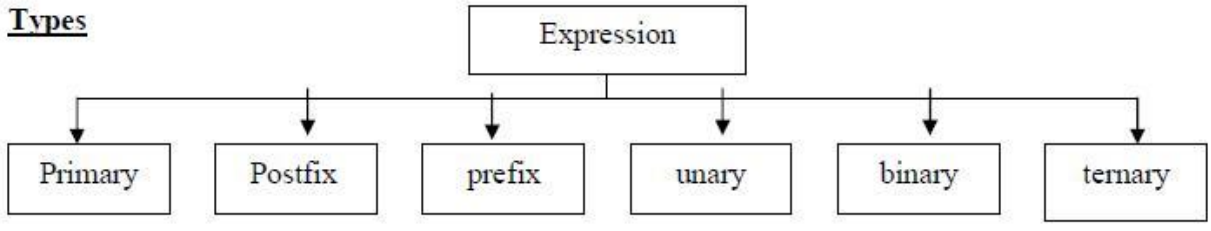
EXPRESSIONS

Def: An expression is a combination of operators and operands which reduces to a single value

An operand is a data item on which an operation is performed.

An operator indicates an operation to be performed on data

eg; z = 3+2*1
z = 5



1. Primary expressions

The operand in the primary expression can be a name, a constant or any parenthesized expression E.g.: c = a+ (5*b);

2. Postifix expressions:

The operator will be after the operands in a postfix expression

Eg:

z=a++ ===> z= a

a = a+1

3. Prefix expressions

The operator is before the operand in a prefix expression.

Eg: z=++a ===> a = a+1

z= a

4. **unary expression:**

It contains one operator and one operand eg: a++, --b

5. Binary expression

It contains 2 operands and one operator

Eg: a=b, c*d

6. Ternary expression

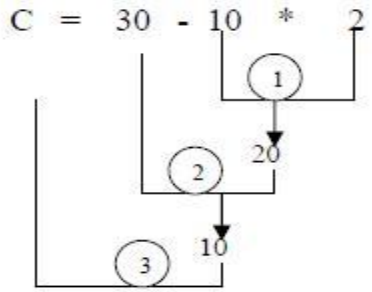
It contains 3 operands and one operator

Eg ; Exp1? Exp2 :Exp3

If Exp1 is true ,Exp2 is executed. otherwise Exp3 is executed.

EXPRESSION EVALUATION, PRECEDENCE AND ASSOCIATIVITY:

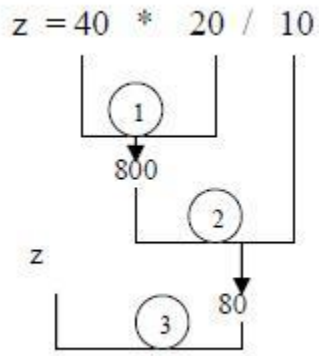
Expressions are evaluated by the 'C' compiler based on precedence and associativity rules. If an expression contains different priority operators then precedence rules are considered.



Here, 10*2 is evaluated first since '*' has more priority than '-' and '='

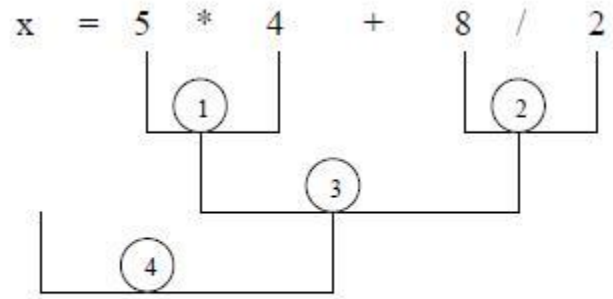
If an expression contains same priority then assoiciativity rules are considered i.e. left right (or right to left)

eg: z= a*b/c



Here '*' and '/' have same priority so, left to right associativity is considered

Eg: x =5 *4 + 8/2



x = 24

Parenthesis has highest priority and comma has least priority among operators

Type Conversions

Converting one data type into another is the concept of type conversion.

2 types

1. Implicit type conversion
2. Explicit type conversion

- Implicit type conversion is automatically done by the compiler by converting smaller data type into a larger data type.

Implicit arithmetic conversions work much as expected. In general, if an operator like + or * that takes two operands (a binary operator) has operands of different types, the ``lower" type is *promoted* to the ``higher" type before the operation proceeds. The result is of the integer type. If there are no unsigned operands, however, the following informal set of rules will suffice:

- If either operand is long double, convert the other to long double.
- Otherwise, if either operand is double, convert the other to double.
- Otherwise, if either operand is float, convert the other to float.
- Otherwise, convert char and short to int.
- Then, if either operand is long, convert the other to long.

- Explicit type conversion is done by the user by using (type)

Eg:
```
int a, b,c;
float avg;
avg=(float)(a+b+c)/3;
```

EXECUTABLE STATEMENTS:

- A 'C' program contains executable statements
- The 'C' compiler translates the executable statements into machine language
- The machine language versions of these statements are executed by the compiler when a user runs program

Types

1) Input – output statements
2) Assignment statements

1. Input – output statements

- Storing a value into memory is called ' input operation'.
- After executing the computations, the results are stored in memory and the results can be displayed to the user by 'output operation'
- All input / output operations in 'C' are performed using input / output functions
- The most common I/o functions are supplied as part of the 'C' standard I/O library through the preprocessor directive # include<stdio.h>

Most commonly used I/o functions are
a) printf ( )
b) scanf ( )

a) printf ( ) function:
Syntax:
printf("format string", print list);
e.g.: printf ("average of 3 numbers = %f",avg);

* The printf ( ) function displays the value of its format string after substituting the values of the expressions in the print list.
* It also replaces the escape sequences such as '\n' by their meanings.

b) scanf ( ) function
Syntax:
scanf ("format string", input list);
e.g.: scanf ("%d %f", &a, &b);

The scanf ( ) function copies into memory data typed at the keyboard by the program user during program execution.

The input list must be preceded by ampersand (&)

2) **Assignment statements**

- The assignment statement store a value (or) a computational result in a variable and is used to perform most arithmetic operations in a program.

Syntax: variable=expression

e.g.:

1. c = a+b;
2. avg = sum/3;
3. r1 = (b*b – 4 * a*c);

- The variable before the assignment operator is assigned the value of the expression after it.
- The previous value of variable is destroyed

Decision control statements:

These are used to make a decision among the alternative paths

They are

1. simple – if statement
2. if – else statement
3. nested – if else statement
4. Else – if ladder

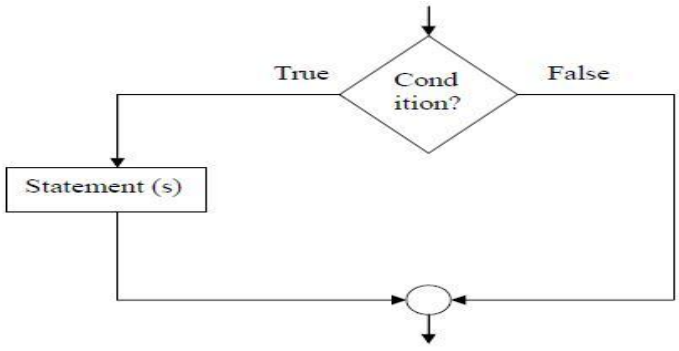
Simple – if statement

'if' keyword is used to execute a set of statement when the logical condition is true

Syntax :

```
if (condition)
{
        Statement (s)
}
```

Flow chart



2. if else statement

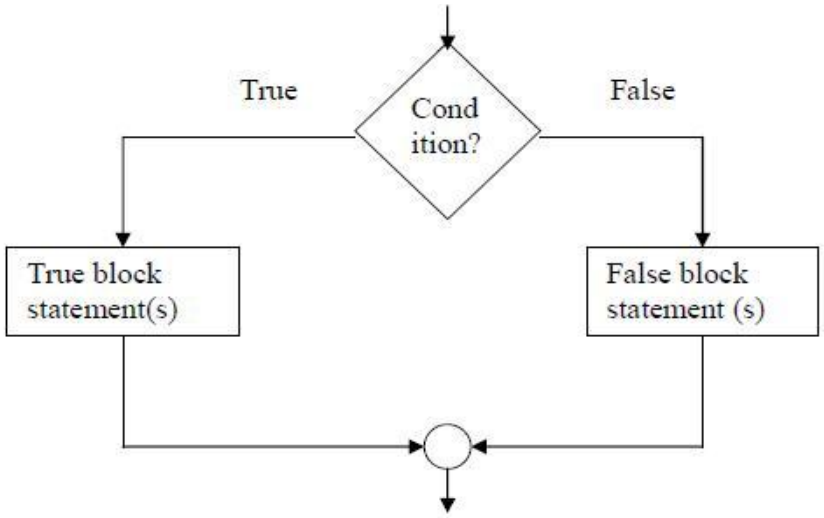
If –else statement takes care of true as well as false conditions

'true block' is executed when the condition is true and 'false block' (or) 'else block' is executed when the condition is false.

Syntax:

```
if (condition)
{
        True block statement(s)
}
else
{
        False block statement(s)
}
```

Flow chart



3. Nested if - else statement

A 'nested if' is an if statement that is the object of either if (or) an else
'if' is placed inside another if (or) else

Syntax:

```
if (condition1)
{
        if (condition2)
        statement1;
        else
         statementt2;
}
else
```

{

if (condition3)

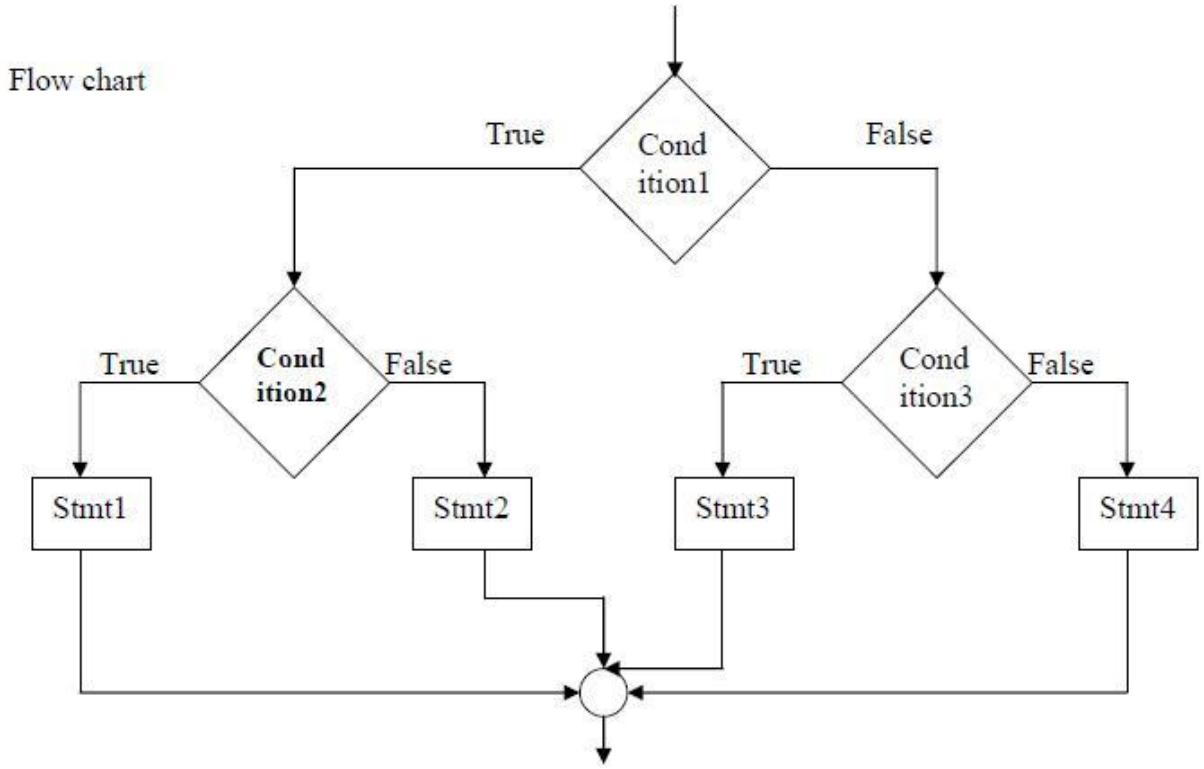
statement3;

else

statement4;

}

Flow chart



4. <u>Else – if ladder</u>

This is the most general way of writing a multi-way decision

Syntax

if (condition1)

statementmt1;

else if (condition2)

statement2;

- - - -

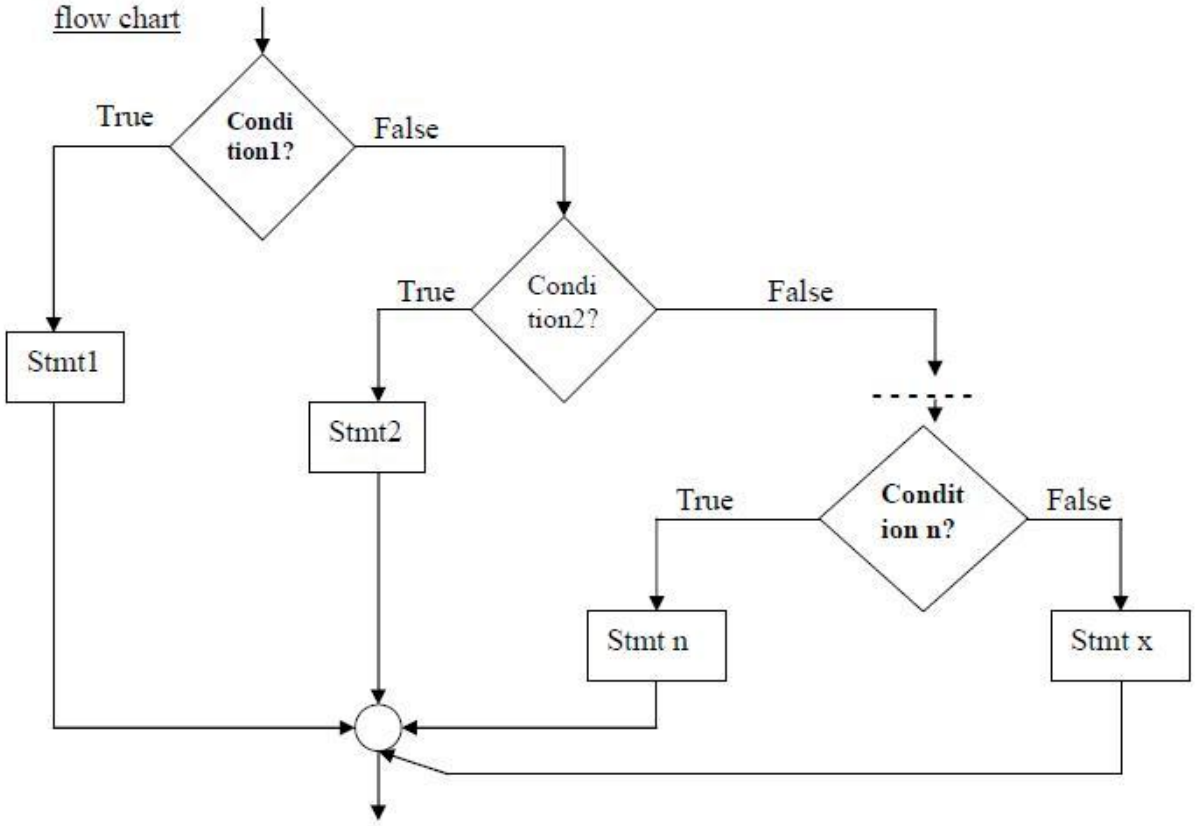
-- - - -

else if (condition n)

statement n;

else

statement x;



Case Control statement:

Switch statement:

It is used to select one among multiple decisions

'switch' successively tests a value against a list of integer (or) character constant.

When a match is found, the statements (or) statements associated with that value are executed.

Syntax

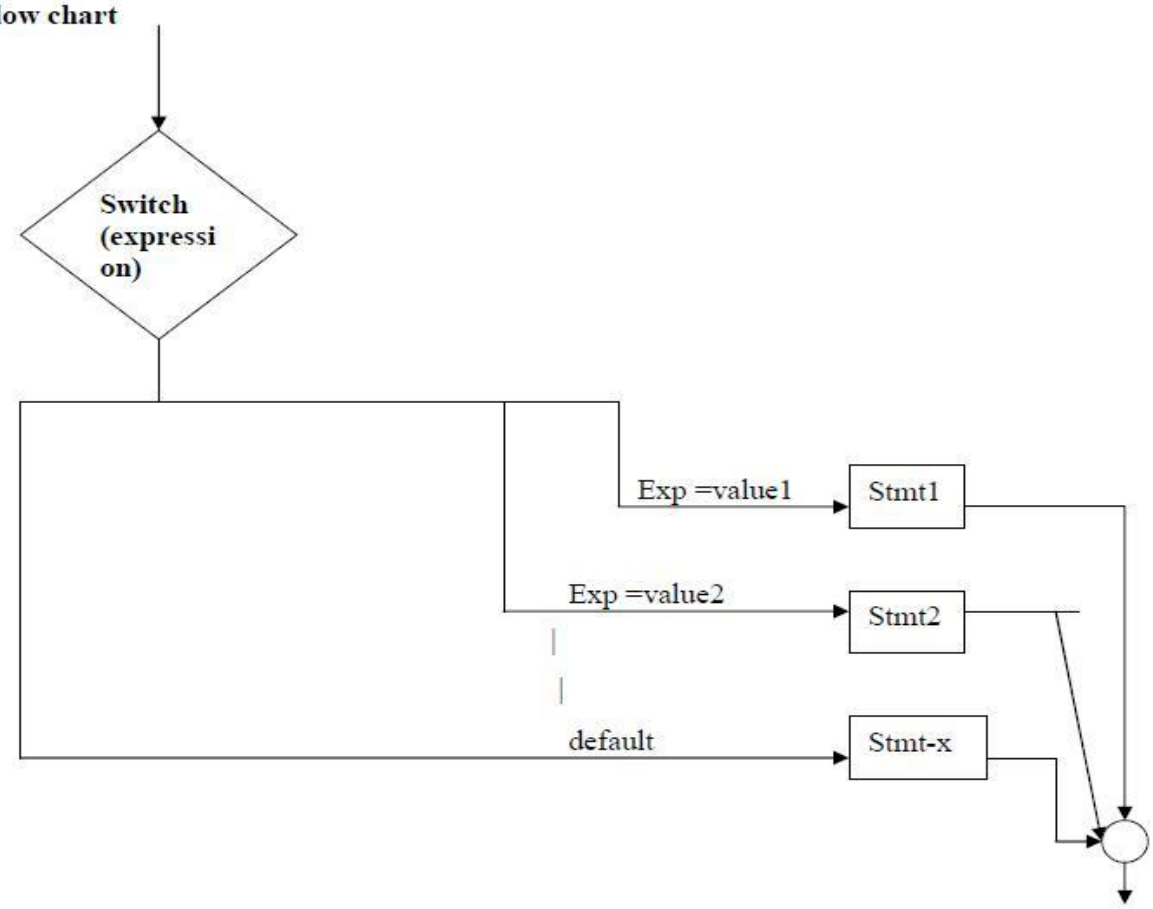
```
switch (expression)
{
  Case value1: stmt1;
               break;
  case value2 : stmt2;
               break;
```

- - - - - -

default: stmt – x;

}

Flow chart



Loop control statements

These are used to repeat set of statements

They are

1) for loop
2) while loop
3) do-while loop

1). for loop

Syntax

for (initialization ; condition ; increment / decrement)

{

body of the loop

}

for statement contains **3 parts**

i) **initialization** is usually an assignment statement that is used to set the loop control variable

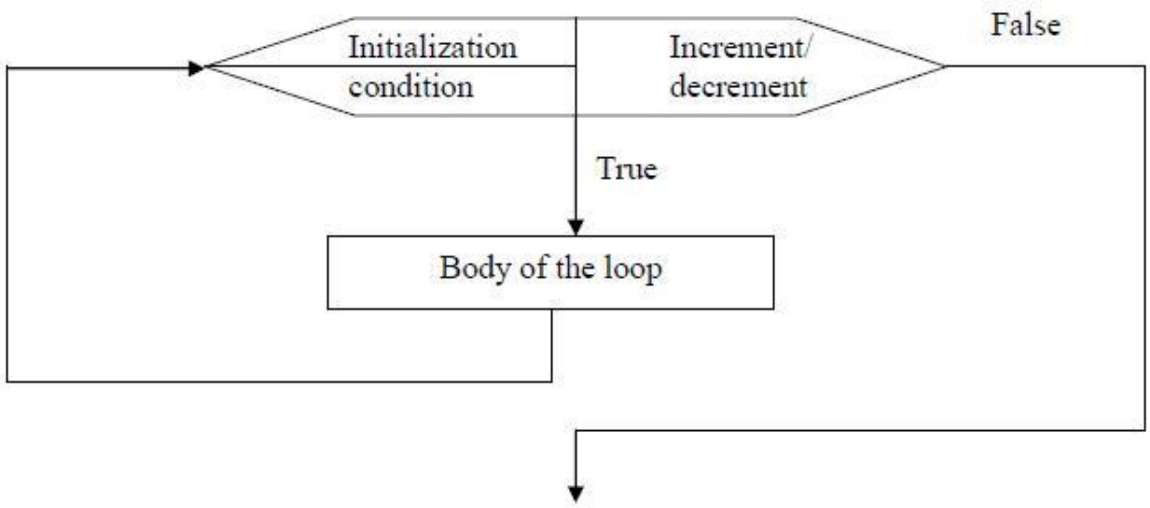
ii) The **condition** is a relational expression that determines when the loop will exit.

iii) The **increment/decrement** part defines how the loop control variable will change each time loop is repeated

iv) loop continues to execute as long as the condition is true

v) Once the condition is false, program continues with the next statement after for loop.

Flow chart



2) While loop

Syntax:

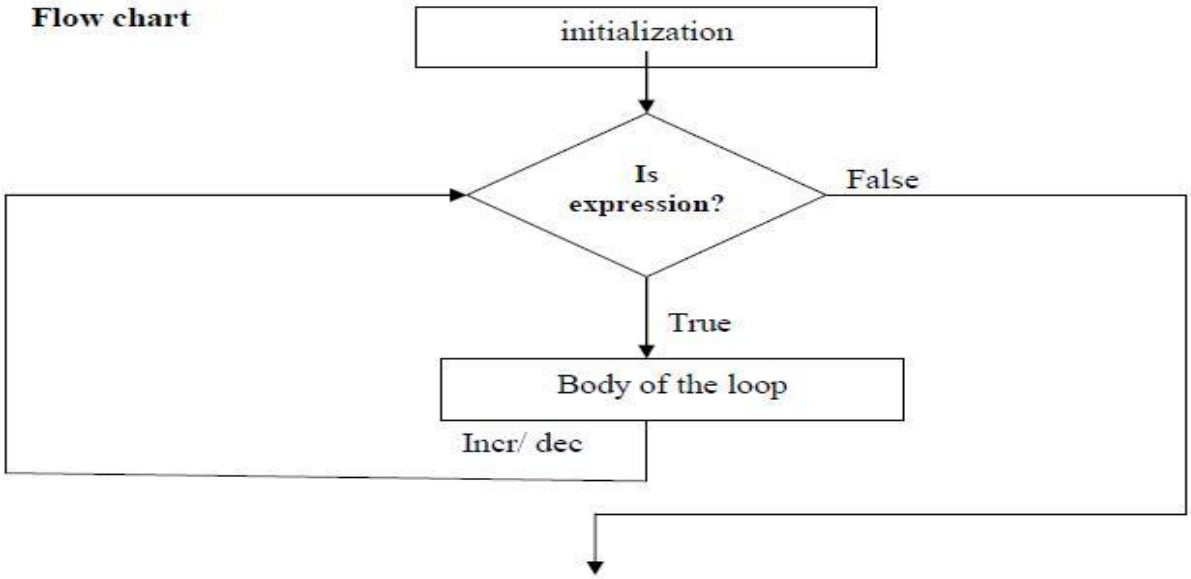
```
Initialization;
while (condition)
{
        body of the loop

        increment/decrement;
}
```

Intialization is done before the loop

Loop continues as long as the condition is true

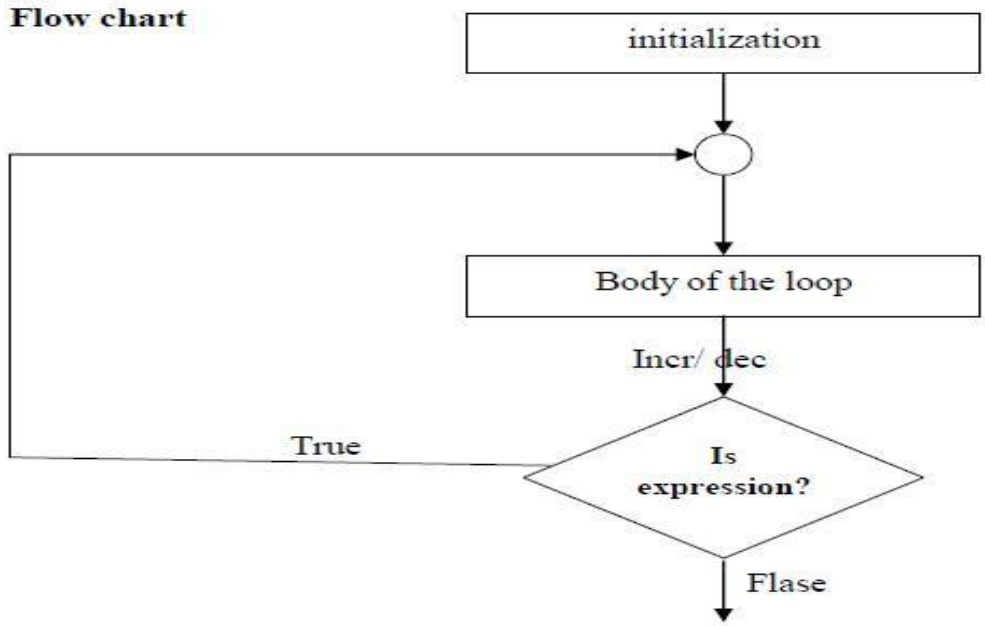
increment/decrement part is done within the loop



3) do-while loop

Syntax:

```
Initialization ;
do
{
        body of the loop
        inc/ dec
} while (condition);
```



Nested for loops

In nested for loops one (or) more for statements are included in the body of the loop

The number of iterations in this type of structure will be equal to number of iterations in the outer loop multiplied by the number of iterations in the inner loop

Other related statements

1) break
2) continue
3) goto

1) break

- It is a keyword used to terminate the loop (or) exit from the block
- The control jumps to next statement after the loop (or) block
- 'break is used with for, while, do-while and switch statement
- When break is used in nested loops then only the innermost loop is terminated

2) continue

- It is a keyword used for continuing the next iteration of the loop
- It skips the statements after the continue statement
- It is used with for, while and do-while

3) goto

It is used to after the normal sequence of program execution by transferring the control to some other part of program

C Operator Precedence and Associativity

Operator	*Description*	*Associativity*

`()`
`[]`
`.`
`->`
`++ --`
	Parentheses (function call) (see Note 1)
	Brackets (array subscript)
	Member selection via object name
	Member selection via pointer
	Postfix increment/decrement
		left-to-right

`++ --`
`+ -`
`! ~`
`(`*`type`*`)`
`*`
`&`
`sizeof`
	Prefix increment/decrement
	Unary plus/minus
	Logical negation/bitwise complement
	Cast (change *type*)
	Dereference
	Address
	Determine size in bytes
		right-to-left

`* / %`	Multiplication/division/modulus	left-to-right
`+ -`	Addition/subtraction	left-to-right
`<< >>`	Bitwise shift left, Bitwise shift right	left-to-right
`< <=` `> >=`	Relational less than/less than or equal to Relational greater than/greater than or equal to	left-to-right
`== !=`	Relational is equal to/is not equal to	left-to-right
`&`	Bitwise AND	left-to-right
`^`	Bitwise exclusive OR	left-to-right
`|`	Bitwise inclusive OR	left-to-right
`&&`	Logical AND	left-to-right
`||`	Logical OR	left-to-right
`?:`	Ternary conditional	right-to-left

`=`
`+= -=`
`*= /=`
`%= &=`
`^= |=`
`<<= >>=`
	Assignment
	Addition/subtraction assignment
	Multiplication/division assignment
	Modulus/bitwise AND assignment
	Bitwise exclusive/inclusive OR assignment
	Bitwise shift left/right assignment
		right-to-left

`,`	Comma (separate expressions)	left-to-right

Note 1: Parentheses are also used to group expressions to force a different order of evaluation; such parenthetical expressions can be nested and are evaluated from inner to outer.

COMPUTER PROGRAMMING UNIT-I

1. Which of the following is not a feature of C language? [d]
a) portability b) extensibility c) structured d) case insensitive
2. ____ is a program that copies programs from a storage device to main memory. [c]
a) compiler b) interpreter c) loader d) linker
3. BIOS is stored in _____. [b]
a) RAM b) ROM c) Hard Disk d) magnetic tape
4. Which of the following is not a data type in C? [d]
a) int b) float c) char d) string
5. The statement violates the rules of structured programming. [a]
a) goto b) break c) continue d) switch
6. The _____ statement is used to terminate the execution of the nearest enclosing loop in which it appears. [a]
a) break b) goto c) loop d) continue
7. Which of the following is an example of application software? [a]
a) DBMS b) OS c) Language Translator d) Security monitor

8. The series of interrelated phases that is used to develop computer software is known as [d]
a) Program development b) Software engineering
c) System design d) Software Development Life Cycle

9. Which of the following statements about the do-while loop is false? [c]
a) a do-while loop executed one or more iterations
b) any statement may be used as the action in a do-while
c) The do-while is best suited for use as an event-controlled loop
d) the do-while is the only loop that requires a semicolon
10. The _____ function reads data from the standard input file keyboard. [d]
a) write b) printf c) read d) scanf

11. What will be sum of the binary numbers 1111 and 11001 [d]
(A) 111100 (B) 100010 (C) 11110 (D) 101000
12. Which one of the following is known as the language of the computer [b]
(A) Programming language (B) Machine language
(C) High level language (D) Assembly level language
13. Find the output [b]

```
Void main()
{char a[]=”12345\0”;
int i=strlen(a);
printf(“here in 3 %d\n”,++i);}
```

(A) here in 3 (B) here in 3 6 (C) 6 (D) 3

14. Which of the following is syntactically correct [d]
(A) for(;); (B) for(); (C) for(,); (D) for(;;);
15. Find out the output for the following [c]
#include<stdio.h>
main() {
int c=--2;
printf("c=%d",c);}
(A) -2 (B) 0 (C) 2 (D) None
16. Identify the result [d]
Void main()
{ int i=5;
Printf("%d",i+++++i);}
(A) 5 (B) 6 (C) 10 (D) compiler error

17. which one of the following is not a translator program [c]
(A) Assembler (B) Interpreter (C) Linker (D) Compiler
18. What will be the ASCII Octal value of A [b]
(A) 100 (B) 101 (C) 110 (D) 111
19. a<<1 is equal to [a]
(A) multiplying by 2 (B) dividing by 2 (C) added 2 (D) None
20. Consider the following and find the output [c]
Main()
{ int a=0;int b=30;char x=1;
If (a,b,x)
Printf("Hello");
}
(A) compiler error (B) abxHello (C) Hello (D) None

21. What type of errors are checked during compilation [a]
(a) logical errors (b) divide by zero error (c) run - time errors (d) syntax errors
22. Which one of the following numeric value is used to represent base of the binary Number [c]
a) 8 b) 10 c) 2 d) 16
23. What will be the binary value of B [b]
a) 1001 b) 1011 c) 1100 d) 1101
24. The program fragment [d]
int a=5, b=2;
printf("%d",a+++++b);
a) prints 7 b) prints 8 c) prints 9 d) none of the above

25. Which one of the following is known as the language of the computer [c]
a) Programming Language b) High Level Language
c) Machine Language d) Assembly language

26. Consider the following program segment. i=6720; j=4; [c]

```
while((i%j)==0)
{
i=i/j; j=j+1;
}
```

On termination j will have the value

a) 4 b) 8 c) 9 d) 6720

27. #include<stdio.h> [d]

```
main()
{
int i=1,j=2;
switch(i)
{
case 1: printf("GOOD");break;
case j: printf("BAD"); break;
}
}
```

a) GOOD b) BAD c) GOOD BAD d) Compiler Error

1. _____ finds syntax error in the source program.
Ans: Compiler
2. _____ is a system software program that helps in coordinating system resources and allows other programs to execute.
Ans: Operating System
3. C language was developed by _____.
Ans: Dennis Ritchie
4. A variable can be made constant by declaring it with the qualifier _____ at the time of Initialization.
Ans: const
5. _____ problem occurs when there is no matching else.
Ans: Dangling else
6. English like statements that follow a loosely defined syntax and are used to convey the design of an algorithm is called _____.
Ans: pseudo code
7. The file created from the compiler is known as _____ file.
Ans: Object File
8. In a pre-test loop, if the body is executed n times, the limit test is executed _____ times.
Ans: n+1 times
9. The multiway selection is coded using switch statement which is similar to _____ construct.
Ans: else-if
10. The _____ operator is used to turn on mask bits in an operand.
Ans: bitwise inclusive or (|)

11. The process of repeating a group of statements in an algorithm is known as_____
Ans: Iteration
12. Extend the term CPU _____
Ans: Central Processing Unit
13. Monitor, keyboard, mouse and printers are _____devices
Ans: Hardware
14. _____is used to compile your c program
Ans: Alt+F9
15. Short Integer size is ____ bytes
Ans: 2 Bytes
16. The while loop repeats a statement until the test at the top proves _____
Ans: False
17. The _____statement transfers control to a statement within its body
Ans: switch
18. The _____is a unconditional branching statement used to transfer control of the program from one statement to another
Ans: goto
19. ANSI stands for _____
Ans: American National Standards Institute

20. ALU stands for _____
Ans: Arithmetic Logic Unit
21. _____ translates the high level language source code into low-level language
Ans: Compiler
22. The size of long double variable is _____
Ans: 10 bytes
23. The _____of an operator gives the order in which expressions involving operators of the same precedence are evaluated.
Ans: Associativity
24. The output of the assembler in the form of sequence of 0's and 1's is called _____
Ans: Object or machine code
25. The process of repeating a group of statements in an algorithm is known as _____
Ans: iteration
26. The # symbol is known as _____
Ans: Preprocessor derivative
27. _____ are identifiers reserved by the C language for special use
Ans: Key words
28. The _____of an operator gives the order in which operators are applied in expressions
Ans: Precedence
29. _____ is very similar to the while loop except that the test occurs at the end of the loop body
Ans: Do while loop