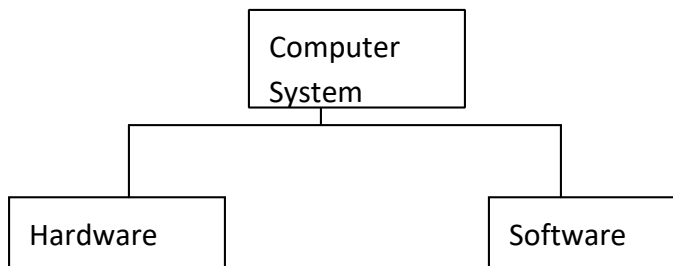


# UNIT - I Introduction to Computing:

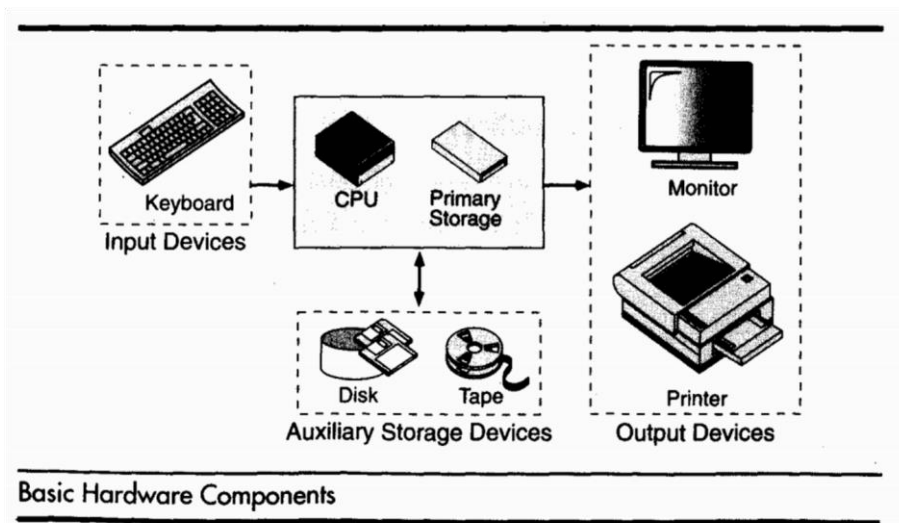
## Computer Systems:

A computer is a system made of two major components: hardware and software. The computer hardware is the physical equipment. The software is the collection of programs (instructions) that allow the hardware to do its job.



## Computer Hardware

The hardware component of the computer system consists of five parts: input devices, central processing unit (CPU), primary storage, output devices, and auxiliary storage devices.



The **input device** is usually a keyboard where programs and data are entered into the computers. Examples of other input devices include a mouse, a pen or stylus, a touch screen, or an audio input unit.

The **central processing unit (CPU)** is responsible for executing instructions such as arithmetic calculations, comparisons among data, and movement of data inside the system. Today's computers may have one, two, or more CPUs. **Primary storage**, also known as main memory, is a place where the programs and data are stored temporarily during processing. The data in primary storage are erased when we turn off a personal computer or when we log off from a time-sharing system.

The **output device** is usually a monitor or a printer to show output. If the output is shown on the monitor, we say we have a **soft copy**. If it is printed on the printer, we say we have a hard copy.

**Auxiliary storage**, also known as **secondary storage**, is used for both input and output. It is the place where the programs and data are stored permanently. When we turn off the computer, or programs and data remain in the secondary storage, ready for the next time we need them.

## Computer Software

Computer software is divided into two broad categories: system software and application software. System software manages the computer resources. It provides the interface between the hardware and the users. Application software, on the other hand, is directly responsible for helping users solve their problems.

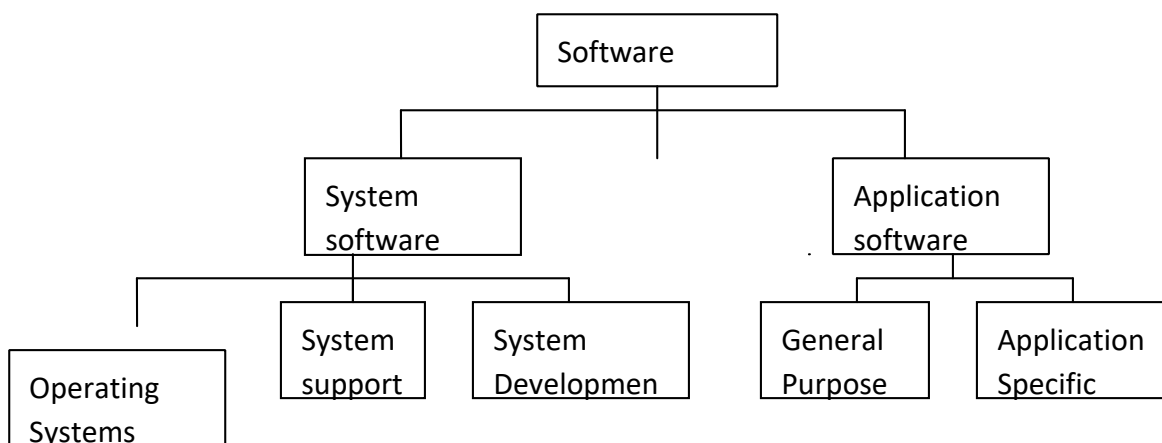


Fig: Types of software

### System Software:

**System software** consists of programs that manage the hardware resources of a computer and perform required information processing tasks. These programs are divided into three classes: the operating system, system support, and system development.

The **operating system** provides services such as a user interface, file and database access, and interfaces to communication systems such as Internet protocols. The primary purpose of this software is to keep the system operating in an efficient manner while allowing the users access to the system.

**System support software** provides system utilities and other operating services. Examples of system utilities are sort programs and disk format programs. Operating services consists of programs that provide performance statistics for the operational staff and security monitors to protect the system and data.

The last system software category, **system development software**, includes the language translators that convert programs into machine language for execution, debugging tools to ensure that the programs are error free and computer –assisted software engineering(CASE) systems.

### **Application software**

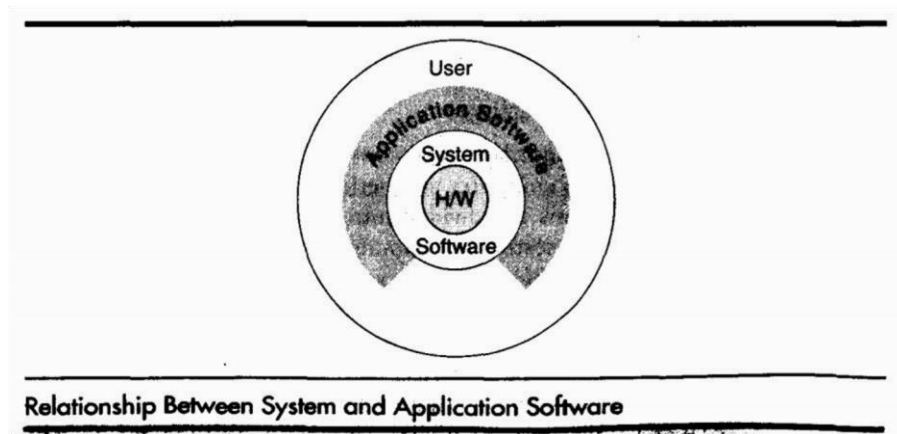
**Application software** is broken in to two classes :general-purpose software and application – specific software. **General purpose software** is purchased from a software developer and can be used for more than one application. Examples of general purpose software include word processors ,database management systems ,and computer aided design systems. They are labeled general purpose because they can solve a variety of user computing problems.

**Application –specific software** can be used only for its intended purpose.

A general ledger system used by accountants and a material requirements planning system used by a manufacturing organization are examples of application-specific software. They can be used only for the task for which they were designed they cannot be used for other generalized tasks.

The relation ship between system and application software is shown in fig-2.In this figure, each circle represents an interface point .The inner core is hard ware. The user is represented by the out layer. To work with the system,the typical user uses some form of application software. The application software in turn interacts with the operating system ,which is apart of the system software layer. The system software provides the direct interaction with the hard ware. The opening

at the bottom of the figure is the path followed by the user who interacts directly with the operating system when necessary.



## Computer Languages:

To write a program for a computer, we must use a **computer language**. Over the years computer languages have evolved from machine languages to natural languages.

1940's                      Machine level Languages

1950's                      Symbolic Languages

1960's                      High-Level Languages

## Machine Languages

In the earliest days of computers, the only programming languages available were machine languages. Each computer has its own machine language, which is made of streams of 0's and 1's.

Instructions in machine language must be in streams of 0's and 1's because the internal circuits of a computer are made of switches transistors and other electronic devices that can be in one of two states: off or on. The off state is represented by 0 , the on state is represented by 1.

The only language understood by computer hardware is machine language.

## **Symbolic Languages:**

In early 1950's Admiral Grace Hopper, A mathematician and naval officer developed the concept of a special computer program that would convert programs into machine language. The early programming languages simply mirror to the machine languages using symbols of mnemonics to represent the various machine language instructions because they used symbols, these languages were known as symbolic languages.

Computer does not understand symbolic language it must be translated to the machine language. A special program called assembler translates symbolic code into machine language. Because symbolic languages had to be assembled into machine language they soon became known as assembly languages.

Symbolic language uses symbols or mnemonics to represent the various ,machine language instructions.

## **High Level Languages:**

Symbolic languages greatly improved programming efficiency; they still required programmers to concentrate on the hardware that they were using. Working with symbolic languages was also very tedious because each machine instruction has to be individually coded. The desire to improve programmer efficiency and to change the focus from the computer to the problem being solved led to the development of high-level language.

High level languages are portable to many different computers, allowing the programmer to concentrate on the application problem at hand rather than the intricacies of the computer. High-level languages are designed to relieve the programmer from the details of the assembly language. High level languages share one thing with symbolic languages, They must be converted into machine language. The process of converting them is known as compilation.

The first widely used high-level languages, FORTRAN (FORmula TRANslation) was created by John Backus and an IBM team in 1957; it is still widely used today in scientific and engineering applications. After FORTRAN was COBOL (Common Business-Oriented Language). Admiral Hopper played a key role in the development of the COBOL Business language.

C is a high-level language used for system software and new application code.

## ALGORITHM:

Algorithms was developed by an Arab mathematician. It is chalked out step-by-step approach to solve a given problem. It is represented in an English like language and has some mathematical symbols like  $\rightarrow$ ,  $>$ ,  $<$ ,  $=$  etc. To solve a given problem or to write a program you approach towards solution of the problem in a systematic, disciplined, non-adhoc, step-by-step way is called Algorithmic approach. Algorithm is a penned strategy(to write) to find a solution.

### Example: Algorithm/pseudo code to add two numbers

Step 1: Start  
Step 2: Read the two numbers in to a,b  
Step 3:  $c=a+b$   
Step 4: write/print c    Step  
5: Stop.


## FLOW CHART :

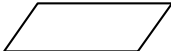
A Flow chart is a Graphical representation of an Algorithm or a portion of an Algorithm. Flow charts are drawn using certain special purpose symbols such as Rectangles, Diamonds, Ovals and small circles. These symbols are connected by arrows called flow lines.

(or)

The diagrammatic representation of way to solve the given problem is called flow chart.

**The following are the most common symbols used in Drawing flowcharts:**

Oval  Terminal start/stop/begin/end.

Parallelogram  Input/output Making data available

For processing(input) or recording of the process information(output).

Rectangle

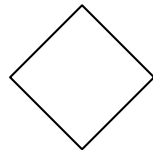


Process

Any processing to be

Done .A process changes or moves data.An assignment operation.

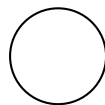
Diamond



Decision

Decision or switching type of operations.

Circle



Connector

Used to connect

Different parts of flowchart.

Arrow →

Flow

Joins two symbols

and also represents flow of execution.

## INTRODUCTION TO 'C' LANGUAGE:

C language facilitates a very efficient approach to the development and implementation of computer programs. The History of C started in 1972 at the Bell Laboratories, USA where Dennis M. Ritchie proposed this language. In 1983 the American National Standards Institute (ANSI) established committee whose goal was to produce “an unambiguous and machine independent definition of the language C “ while still retaining it’s spirit .

C is the programming language most frequently associated with UNIX. Since the 1970s, the bulk of the UNIX operating system and its applications have been written in C. Because the C language does not directly rely on any specific hardware architecture, UNIX was one of the first portable operating systems. In other words, the majority of the code that makes up UNIX does not know and does not care which computer it is actually running on. Machine-specific features are isolated in a few modules within the UNIX kernel, which makes it easy for you to modify them when you are porting to a different hardware architecture.

C was first designed by Dennis Ritchie for use with UNIX on DEC PDP-11 computers. The language evolved from Martin Richard's BCPL, and one of its earlier forms was the B language, which was written by Ken Thompson for the DEC PDP-7. The first book on C was *The C Programming Language* by Brian Kernighan and Dennis Ritchie, published in 1978.

In 1983, the American National Standards Institute (ANSI) established a committee to standardize the definition of C. The resulting standard is known as *ANSI C*, and it is the recognized standard for the language, grammar, and a core set of libraries. The syntax is slightly different from the original C language, which is frequently called K&R for Kernighan and Ritchie. There is also an ISO (International Standards Organization) standard that is very similar to the ANSI standard.

It appears that there will be yet another ANSI C standard officially dated 1999 or in the early 2000 years; it is currently known as "C9X."

## BASIC STRUCTURE OF C LANGUAGE:

The program written in C language follows this basic structure. The sequence of sections should be as they are in the basic structure. A C program should have one or more sections but the sequence of sections is to be followed.

1. Documentation section



2. Linking section
3. Definition section
4. Global declaration section
5. Main function section

```
{  
  
    Declaration section  
  
    Executable section  
  
}
```

6. Sub program or function section

**1. DOCUMENTATION SECTION :** comes first and is used to document the use of logic or reasons in your program. It can be used to write the program's objective, developer and logic details. The documentation is done in C language with `/*` and `*/`. Whatever is written between these two are called comments.

**2. LINKING SECTION :** This section tells the compiler to link the certain occurrences of keywords or functions in your program to the header files specified in this section.

e.g. `#include <stdio.h>`

**3. DEFINITION SECTION :** It is used to declare some constants and assign them some value.

e.g. `#define MAX 25`

Here `#define` is a compiler directive which tells the compiler whenever `MAX` is found in the program replace it with `25`.

**4. GLOBAL DECLARATION SECTION :** Here the variables which are used through out the program (including main and other functions) are declared so as to make them global(i.e accessible to all parts of program)

e.g. int i; (before main())

**5. MAIN FUNCTION SECTION** : It tells the compiler where to start the execution from

```
main()

{

    point from execution starts

}
```

main function has two sections

1. declaration section : In this the variables and their data types are declared.
2. Executable section : This has the part of program which actually performs the task we need.

**6. SUB PROGRAM OR FUNCTION SECTION** : This has all the sub programs or the functions which our program needs.

**SIMPLE 'C' PROGRAM:**

```
/* simple program in c */

#include<stdio.h>

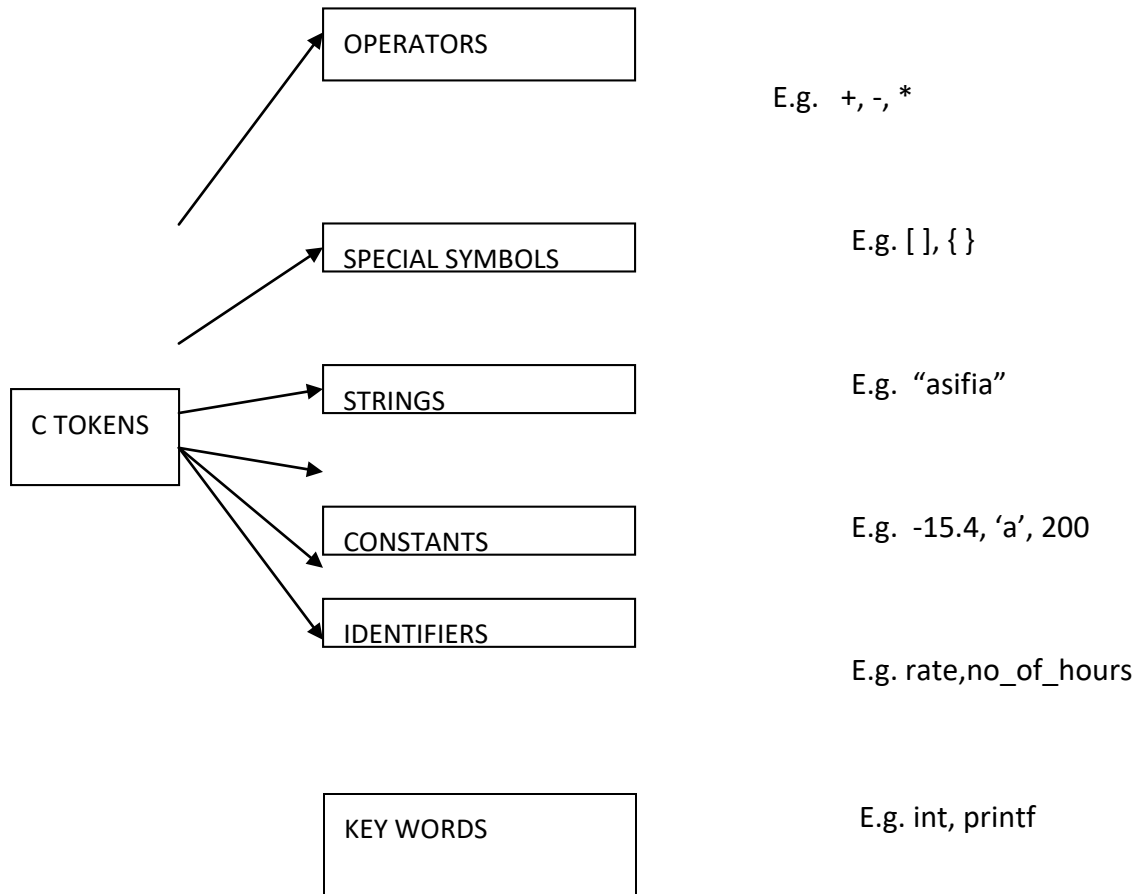
main()

{

    printf("welcome to c programming");
```

```
}/* End of main */ C-TOKENS :
```

Tokens are individual words and punctuations marks in English language sentence. The smallest individual units are known as C tokens.



A C program can be divided into these tokens. A C program contains minimum 3 c tokens no matter what the size of the program is.

## KEYWORDS :

There are certain words, called keywords (reserved words) that have a predefined meaning in 'C' language. These keywords are only to be used for their intended purpose and not as identifiers.

The following table shows the standard 'C' keywords

Auto	Break	Case	Char	Const	Continue
Default	Do	Double	Else	Enum	Extern

Float	For	Goto	If	Int	Long
Register	Return	Short	Signed	Sizeof	Static
Struct	Switch	Typedef	Union	Unsigned	void
Volatile	While				

## IDENTIFIERS :

Names of the variables and other program elements such as functions, array, etc, are known as identifiers.

There are few rules that govern the way variable are named (identifiers).

1. Identifiers can be named from the combination of A-Z, a-z, 0-9, \_(Underscore).
2. The first alphabet of the identifier should be either an alphabet or an underscore. digit are not allowed.
3. It should not be a keyword.

Eg: name, ptr, sum

After naming a variable we need to declare it to compiler of what data type it is .

The format of declaring a variable is

Data-type id1, id2, ..., idn; where data type could be float, int, char or any of the data types.

id1, id2, id3 are the names of variable we use. In case of single variable no commas are required.

eg float a, b, c;

int e, f, grand total;

```
char present_or_absent;
```

### **ASSIGNING VALUES :**

When we name and declare variables we need to assign value to the variable. In some cases we assign value to the variable directly like

```
a=10;
```

in our program.

In some cases we need to assign values to variable after the user has given input for that.

eg we ask user to enter any no and input it

```
/* write a program to show assigning of values to variables */
```

```
#include<stdio.h> main()
```

```
{
```

```
int a; float b;
```

```
printf("Enter any number\n");
```

```
b=190.5;
```

```
scanf("%d",&a);    printf("user
```

```
entered %d", a); printf("B's values is
```

```
%f", b);
```

```
}
```

## CONSTANTS :

A quantity that does not vary during the execution of a program is known as a constant supports two types of constants namely Numeric constants and character constants.

### NUMERIC CONSTANTS:

1. Example for an integer constant is 786,-127
2. Long constant is written with a terminal 'l' or 'L', for example 1234567899L is a Long constant.
3. Unsigned constants are written with a terminal 'u' or 'U', and the suffix 'ul' and 'UL' indicates unsigned long. for example 123456789u is a Unsigned constant and 1234567891ul is an unsigned long constant.
4. The advantage of declaring an unsigned constant is to increase the range of storage.
5. Floating point constants contain a decimal point or an exponent or both. For Eg : 123.4, 1e-2, 1.4E-4, etc. The suffixes f or F indicate a float constant while the absence of f or F indicate the double, l or L indicate long double.

### CHARACTER CONSTANTS:

A character constant is written as one character with in single quotes such as 'a'. The value of a character constant is the numerical value of the character in the machines character set. certain character constants can be represented by escape sequences like '\n'. These sequences look like two characters but represent only one.

The following are the some of the examples of escape sequences:

Escape sequence	Description
-----------------	-------------

<code>\a</code>	Alert
<code>\b</code>	Backspace
<code>\f</code>	Form feed
<code>\n</code>	New Line
<code>\r</code>	Carriage return
<code>\t</code>	Horizontal Tab
<code>\v</code>	Vertical Tab

String constants or string literal is a sequence of zero or more characters surrounded by a double quote. Example , “ I am a little boy”. quotes are not a part of the string.

To distinguish between a character constant and a string that contains a single character ex: ‘a’ is not same as “a”. ‘a’ is an integer used to produce the numeric value of letter a in the machine character set, while “a” is an array of characters containing one character and a ‘\0’ as a string in C is an array of characters terminated by NULL.

There is one another kind of constant i.e Enumeration constant , it is a list of constant integer values.

Ex.: `enum color { RED, Green, BLUE }`

The first name in the enum has the value 0 and the next 1 and so on unless explicit values are specified.

If not all values specified , unspecified values continue the progression from the last specified value. For example

`Enum months { JAN=1, FEB,MAR, ..., DEC -`

Where the value of FEB is 2 and MAR is 3 and so on.

Enumerations provide a convenient way to associate constant values with names.

## **VARIABLES :**

A quantity that can vary during the execution of a program is known as a variable. To identify a quantity we name the variable for example if we are calculating a sum of two numbers we will name the variable that will hold the value of sum of two numbers as 'sum'.

## **DATA TYPES :**

To represent different types of data in C program we need different data types. A data type is essential to identify the storage representation and the type of operations that can be performed on that data. C supports four different classes of data types namely

1. Basic Data types
2. Derives data types
3. User defined data types
4. Pointer data types

## **BASIC DATA TYPES:**

All arithmetic operations such as Addition , subtraction etc are possible on basic data types.

E.g.: `int a,b;`

`Char c;`

**The following table shows the Storage size and Range of basic data types:**

<b>TYPE</b>	<b>LENGTH</b>	<b>RANGE</b>
-------------	---------------	--------------



Unsigned char	8 bits	0 to 255	
Char	8 bits	-128 to 127	
Short int	16 bits	-32768 to 32767	
Unsigned int	32 bits	0 to 4,294,967,295	
Int	32 bits	-2,147,483,648 to 2,147,483,648	
Unsigned long	32 bits	0 to 4,294,967,295	
Enum	16 bits	-2,147,483,648 to 2,147,483,648	
Long	32 bits	-2,147,483,648 to 2,147,483,648	
Float	32 bits	3.4*10E-38 to 3.4*10E38	
Double	64 bits	1.7*10E-308 to 1.7*10E308	
Long double	80 bits	3.4*10E-4932 to 1.1*10E4932	

### DERIVED DATA TYPES:

Derived datatypes are used in 'C' to store a set of data values. Arrays and Structures are examples for derived data types.

Ex: `int a[10];`

`Char name[20];`

### USER DEFINED DATATYPES:

C Provides a facility called typedef for creating new data type names defined by the user. For Example ,the declaration ,

**`typedef int Integer;`**

makes the name Integer a synonym of int. Now the type Integer can be used in declarations, casts, etc, like,

**Integer num1, num2;**

Which will be treated by the C compiler as the declaration of num1, num2 as int variables.

“typedef” is more useful with structures and pointers.

### **POINTER DATA TYPES:**

Pointer data type is necessary to store the address of a variable.

### **INPUT AND OUTPUT STATEMENTS :**

The simplest of input operator is getchar to read a single character from the input device.

```
varname=getchar();
```

you need to declare varname.

The simplest of output operator is putchar to output a single character on the output device.

```
putchar(varname)
```

The getchar() is used only for one input and is not formatted. Formatted input refers to an input data that has been arranged in a particular format, for that we have scanf.

```
scanf("control string", arg1, arg2,...argn);
```

Control string specifies field format in which data is to be entered.

arg1, arg2... argn specifies address of location or variable where data is stored.

eg scanf("%d%d",&a,&b);  
%d used for integers

%f floats

%l long

%c character

for formatted output you use printf printf("control  
string", arg1, arg2,...argn);

```
/* program to exhibit i/o */
```

```
#include<stdio.h>
```

```
main()
```

```
{
```

```
    int a,b;    float c;
```

```
    printf("Enter any number");
```

```
    a=getchar();
```

```
    printf("the char is ");
```

```
    putchar(a);
```

```
    printf("Exhibiting the use of scanf");
```

```
    printf("Enter three numbers");
```

```
    scanf("%d%d%f",&a,&b,&c); printf("%d%d%f",a,b,c);
```

```
}
```

## **OPERATORS :**

An operator is a symbol that tells the compiler to perform certain mathematical or logical manipulations. They form expressions.

C operators can be classified as

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment or Decrement operators
6. Conditional operator
7. Bit wise operators
8. Special operators

**1. ARITHMETIC OPERATORS** : All basic arithmetic operators are present in C.

operator	meaning
+	add
-	subtract
*	multiplication
/	division
%	modulo division(remainder)

An arithmetic operation involving only real operands(or integer operands) is called real arithmetic(or integer arithmetic). If a combination of arithmetic and real is called mixed mode arithmetic.

**2. RELATIONAL OPERATORS :** We often compare two quantities and depending on their relation take certain decisions for that comparison we use relational operators.

operator	meaning
<	is less than
>	is greater than
<=	is less than or equal to
>=	is greater than or equal to
==	is equal to
!=	is not equal to

It is the form of

ae-1 relational operator ae-2

**3. LOGICAL OPERATORS :** An expression of this kind which combines two or more relational expressions is termed as a logical expressions or a compound relational expression. The operators and truth values are

op-1	op-2	op-1 && op-2	op-1    op-2
non-zero	non-zero	1	1
non-zero	0	0	1
0	non-zero	0	1
0	0	0	0

op-1	!op-1
------	-------

non-zero	zero
zero	non-zero

**5. ASSIGNMENT OPERATORS :** They are used to assign the result of an expression to a variable.  
The assignment operator is '='.

$v \text{ op} = \text{exp}$

$v$  is variable     $\text{op}$  binary operator

$\text{exp}$  expression     $\text{op} =$  short hand

assignment operator short hand assignment

operators use of simple assignment

operators use of short hand assignment

operators

$a = a + 1$     $a += 1$     $a = a - 1$     $a -= 1$     $a = a \% b$     $a \% = b$

**6. INCREMENT AND DECREMENT OPERATORS :**

$++$  and  $--$  are called increment and decrement operators used to add or subtract.

Both are unary and as follows

$++m$  or  $m++$

$--m$  or  $m--$

The difference between  $++m$  and  $m++$  is    if  $m=5$ ;

$y = ++m$  then it is equal to  $m=5; m++; y=m$ ;    if  $m=5$ ;

$y = m++$  then it is equal to  $m=5; y=m; m++$ ;

**7. CONDITIONAL OPERATOR :** A ternary operator pair "?:" is available in C to construct conditional expressions of the form

exp1 ? exp2 : exp3;

It work as if exp1 is true then

exp2 else exp3

**8. BIT WISE OPERATORS :** C supports special operators known as bit wise operators for manipulation of data at bit level. They are not applied to float or double.

operator	meaning
----------	---------

&	Bitwise AND
---	-------------

	Bitwise OR
--	------------

^	Bitwise exclusive OR
---	----------------------

<<	left shift
----	------------

>>	right shift
----	-------------

~	one's complement
---	------------------

**9. SPECIAL OPERATORS :** These operators which do not fit in any of the above classification are ,(comma), sizeof, Pointer operators(& and \*) and member selection operators (. and ->). The comma operator is used to link related expressions together. sizeof operator is used to know the sizeof operand.

/\* programs to exhibit the use of operators \*/

#include<stdio.h> main()

```

{

    int sum, mul, modu;

float sub, divi;


    int i,j;  float l, m;  printf("Enter
two integers ");

scanf("%d%d",&i,&j);

printf("Enter two real numbers");

scanf("%f%f",&l,&m);  sum=i+j;

mul=i*j;

    modu=i%j;  sub=l-m;  divi=l/m;

printf("sum is %d", sum);  printf("mul is
%d", mul);  printf("Remainder is %d",
modu);  printf("subtraction of float is
%f", sub);  printf("division of float is
%f", divi);

}

/* program to implement relational and logical */

#include<stdio.h> main()

{

    int i, j, k;    printf("Enter any
three numbers ");    scanf("%d%d%d",

```



```

&i, &j, &k);    if((i<j)&&(j<k))

printf("k is largest"); else if i<j || j>k

    {

        if i<j && j >k

printf("j is largest");

        else

printf("j is not largest of all");    }

}

/* program to implement increment and decrement operators */

#include<stdio.h> main()

{

    int i;

    printf("Enter a number");

scanf("%d", &i);

    i++;

    printf("after incrementing %d ", i);

    i--;

    printf("after decrement %d", i);

}

/* program using ternary operator and assignment */

```

```
#include<stdio.h> main()

{

    int i,j,large;    printf("Enter two
numbers ");    scanf("%d%d",&i,&j);
large=(i>j)?i:j;      printf("largest of two
is %d",large);

}
```