
UNIT–I: Introduction to Computers: System Software, Program Developing Steps, Algorithms, Flow charts. **Introduction to C:** Structure of C Program**,** Variable Names, Data Types, Constants, Operators, Type Conversions, Expressions, Precedence and Order of Evaluation.
Managing I/O: Input-Output Statements formatted I/O.

1. List and explain the functions of various parts of computer hardware and software.

Ans : Computer System :

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

I. Computer Hardware.

II.Computer Software.

The following figure shows a computer system.

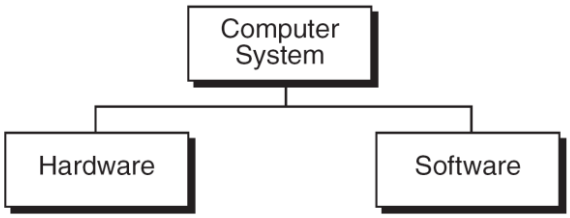


Fig : A Computer System

I. Computer Hardware:

The computer hardware is the physical equipment. The hardware component of computer system consists of 5 parts

A. Input Devices.
B. Central Processing Unit (CPU).
C. Primary Storage.
D. Output Devices.
E. Auxiliary Storage Devices.

The following figure shows the *hardware* components.

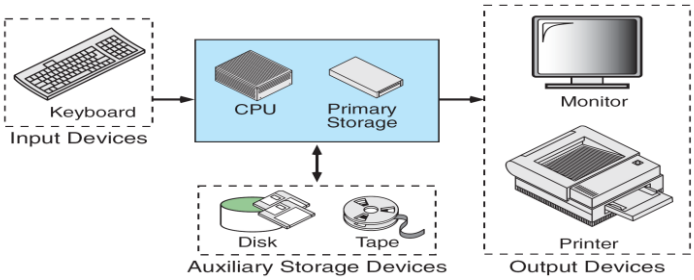


Fig: Hardware Components

A. Input Devices: The input device is usually a keyboard where programs and data are entered in to the computer. Other Input Devices : a touch screen , a mouse , a pen , an audio input unit.

B. Central Processing Unit (CPU): It 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 or more CPU's

C. Primary Storage: It is also known as main memory . It is a place where the programs and data is stored temporarily during processing. The Data in primary storage is erased , when we turn off a personal computer or when we log off from a time-sharing computer (volatile).

D. Output Devices: The output device is usually a monitor or a printer to show output. If the output is shown on the monitor, it is a soft copy and if the output is printed on the printer, it is a hard copy.

E. Auxiliary Storage Devices (secondary storage devices): It is used for both input and output. It is also known as secondary storage. It is a place where the programs and data are stored permanently. when we turn off the computer the programs and data remain in the secondary storage, ready for the next time when we need them.

II. Computer Software : Software is the collection of Programs (instructions) that allow the hardware to do its job.

They are two types of Computer Software.

A. System Software
B. Application Software

The following figure shows the Computer Software.

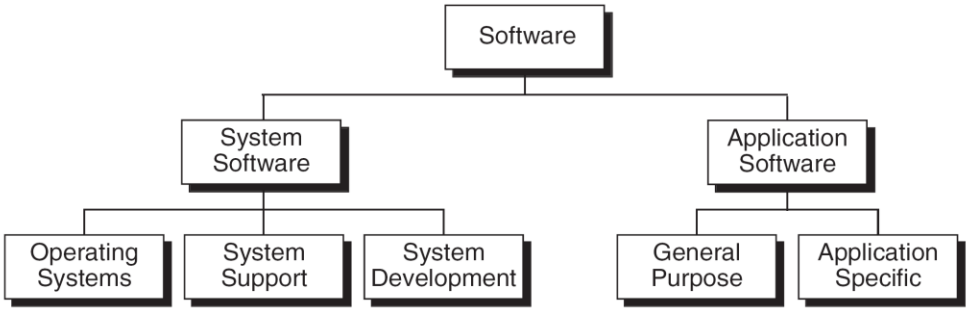


Fig: Computer Software

A. 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 in to three classes.

i. Operating System Software.
ii. System Support Software.
iii. System Development Software.

i. Operating System Software: It provides services such as a user interface, files and data base 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.

ii. System Support Software : System Support Software provides system utilities and other operating services. Examples of system utilities are sort programs and disk format programs. Operating services consist of programs that provide performance statistics for the operational staff and security monitors to protect the system and data.

iii. System Development Software: It includes language translators that convert programs in to machine language for execution , debugging tools to ensure that programs are error - free and computer -assisted software engineering (CASE) systems.

B. Application Software: It is directly responsible for helping users to solve their problems. Application software is broken in to two classes.

i. General - Purpose Software
ii. Application - Specific Software

i. General Purpose Software: It is purchased from a software developer and can be used for more than one application. Examples: word processors, database management systems, computer – aided design systems. They are labeled general purpose because they can solve a variety of user computing problems.

ii. Application Specific Software: It can be used only for its intended purpose.
Example: A general ledger system used by accountants.

They can be used only for the task for which they were designed.They can not be used for other generalized tasks.

Relationship between system and application software is shown in the figure:

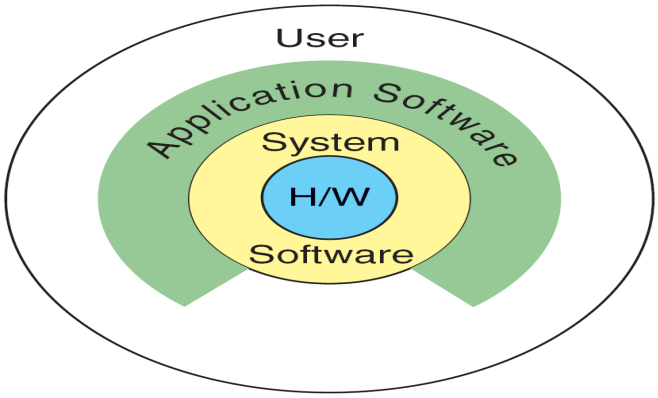


Fig: Relationship between System and Application Software

Each circle is an interface point. The inner core is hardware. The user is represented by the outer layer. To work with the system , the typical user uses some form of application software. The application software in turn interacts with operating system (OS), which is part of system software layer. The System software provides the direct interaction with the hardware. The opening at the bottom of the figure is the path followed by the user who interacts directly with the Operating System when necessary.

2. Explain Creation and Running of Programs ?

or

Describe how the Developers will Write the Programs ?

Ans: CREATING AND RUNNING PROGRAMS :

It is the job of programmer to write and test the program.The following are four steps for creating and running programs:

A. Writing and Editing the Program.
B. Compiling the Program.
C. Linking the Program with the required library modules.
D. Executing the Program.

A. Writing and Editing Program : The Software to write programs is known as text editor. A text editor helps us enter , change and store character data. Depending on the editor on our system, it could be used to write letters, create reports or write programs. Example : word processor.

The text editor could be generalized word processor, but every compiler comes with associated text editor. Some of the features of editors are

Search : To locate and replace statements.

Copy , Paste : To copy and move statements.
Format : To set tabs to align text.
After the program is completed the program is saved in a file to disk. This file will be input to the compiler, it is known as source file. The following figure shows the various steps in building a *C – program.*

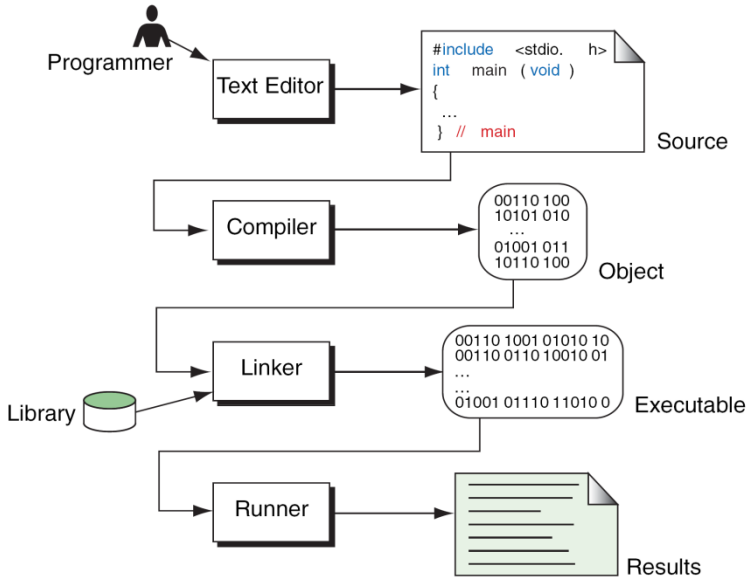


Fig: Building a C - program

B. Compiling Programs: The code in a source file on the disk must be translated in to machine language. This is the job of compiler which translates code in source file stored on disk in to machine language. The C compiler is actually two separate programs: the preprocessor and the translator. The preprocessor reads the source code and prepares it for the compiler. It scans special instructions known as preprocessor commands. These commands tell the preprocessor to take for special code libraries, make substitutions in the code. The result of preprocessing is called translation unit. The translator reads the translation unit and writes resulting object module to a file that can be combined with other precompiled units to form the final program. An object module is the code in machine language. This module is not ready for execution because it does not have the required C and other functions included.

C. Linking Programs: C programs are made up of many functions.
Example: printf ( ) , cos ( )… etc
Their codes exists else where , they must be attached to our program. The *linker* assembles all these functions , ours and the systems, in to a final executable program.
D. Executing Programs : Once our program has been linked, it is ready for execution. To execute a program, we use operating system command, such as *run* to load the program in to main memory and execute it. Getting program in to memory is the function of an Operating System programs called *loader*. *Loader* locates the executable program and reads it in to memory. In a typical program execution,

the program reads data for processing, either from user or from file. After the program processes the data, it prepares output. Data output can be to user's monitor or to a file. When program has executed, *Operating System* removes the program from memory.

3. List and Explain about the Program Development?

or

Explain how the Developers will Test the Program?

or

What do you mean by 'program development cycle'? Explain the steps involved in it?

0r

What are different steps followed in program development ?

Ans: Program Development :

Program Development is a *multi step process* as follows

A. Understand the Problem
B. Develop a Solution - Algorithm - Pseudo code - Flowchart
C. Write a Program
D. Test it.

For program development , developers are given

Program requirement statement
Design of program interfaces
Overview of the complete project

A. Understand the problem :The first step in solving any problem is *understand* it. We begin by reading the requirements statements carefully.When we think that we fully understand it, we review our understanding with the user and the system analyst.

Example: Calculate the square footage of house.

The questions that arise are

1. What is definition of square footage ?
2. How is it useful ?

B. System requirement: We do planning of functional requirements - Software's Version, Non-functional requirements - Resources - Internet , table , A/cs , Coffee. User functional requirements - Problem Oriented and Language used is in simple i.e. English.

C. Analysis: User requirements are transformed to technical terms like Blueprint and also look for different alternatives considering all risks.

D. Design: In UML software users will develop flowcharts, structure charts, data flow diagrams

1. Choose the best way to find solution among different alternatives.
2. Choose in analysis , design files , databases and functions.

E. Code:

Unimportant in Software development life cycle

1. Uses software for code generation (forward energy).
2. Otherwise go for manual coding.
3. Basic testing is done (compilation).

F. System Test :Test if user requirements are met Blackbox Testing: Test programs together, make sure system works as a whole. Whitebox Testing: Release ά-version to customer, according to his/her feed back we develop β-Version: β - version is released to selected group of people.

G.Maintenance :

Keeps the system working once it has been put in to production ?

4. Explain about Algorithm ?

or

Define Algorithm and State Properties of it ?

Ans: Algorithm :

Algorithm is a finite set of instructions that , if followed accomplishes a particular task.

Algorithm should satisfy the following criteria

1. *Input :* Zero or more quantities are externally supplied.
2. *Output :* At least one quantity is produced.
3. *Definiteness :* Each instruction is clear and unambiguous. Ex Add B or C to A
4. *Finiteness :* Algorithm should terminate after finite number of steps when traced in all cases

Ex: Go on adding elements to an array

5. *Effectivenes:* Every instruction must be basic i.e., it can be carried out, by a person using pencil and paper.

Algorithm must also be general to deal with any situation.

5. List out the advantages and disadvantages of algorithm.

Ans.

Advantages of Algorithms:

- It provides the core solution to a given problem.the solution can be implemented on a computer system using any programming language of user's choice.
- It facilitates program development by acting as a design document or a blue print of a given problem solution.
- It ensures easy comprehension of a problem solution as compared to an equilavent computer program.
- It eases identification and removel of logical errors in a program.
- It facilitates algorithm analysis to find out the most efficient solution to a given problem.

Dis advantages of Algorithms:

- In large algorithms the flow of program control becomes difficult to track.
- Algorithms lack visual representation of programming constructs like flowcharts; thus understanding the logic becomes relatively difficult.

6. Explain the Algorithm with Examples ?

Ans: Algorithm:

The same problem can be solved with different methods. So, to solve a problem different algorithms, may be accomplished. Algorithm may vary in time, space utilized. User writes algorithm in his / her own language. So, it can not be executed on computer. Algorithm should be in sufficient detail that it can be easily translated in to any of the languages.

Example1 : Add two numbers.

Step 1: Start
Step 2: Read 2 numbers as A and B
Step 3: Add numbers A and B and store result in C
Step 4 : Display C
Step 5: Stop

Example2: Average of 3 numbers.

1. Start
2. Read the numbers a , b , c
3. Compute the sum and divide by 3
4. Store the result in variable d
5. Print value of d
6. End

Example3: Average of n inputted numbers.

1. Start
2. Read the number n
3. Initialize i to zero
4. Initialize sum to zero
5. If i is greater than n
6. Read a
7. Add a to sum
8. Go to step 5
9. Divide sum by n & store the result in avg
10. Print value of avg
11. End

7. Explain about a Flowchart ?

Ans: *Flowchart :*

A *flowchart* is a visual representation of the sequence of steps for solving a problem . A *flowchart* is a set of symbols that indicate various operations in the *program.* For every process , there is a corresponding *symbol* in the *flowchart* . Once an *algorithm* is written , its pictorial representation can be done using *flowchart symbols*. In other words, a pictorial representation of a *textual algorithm* is done using a *flowchart.*

A *flowchart* gives a pictorial representation of an *algorithm.*

- The first *flowchart* is made by John Von Neumann in 1945.
- It is a symbolic diagram of operations sequence, dataflow, control flow and processing logic in information processing.
- The symbols used are simple and easy to learn.
- It is a very helpful tool for programmers and beginners.

Purpose of a Flowchart :

- Provides Communication.
- Provides an Overview.
- Shows all elements and their relationships.
- Quick method of showing Program flow.
- Checks Program logic.
- Facilitates Coding.
- Provides Program revision.
- Provides Program documentation.

Advantages of a Flowchart :

- *Flowchart is* an important aid in the development of an algorithm itself.
- E asier to Understand than a Program itself.
- Independent of any particular programming language.
- Proper documentation.
- Proper debugging.
- Easy and Clear presentation.

Limitations of a Flowchart :

- Complex logic.
- Drawing is time consuming.
- Difficult to draw and remember.
- Technical detail.

8. Explain the Symbols in a Flowchart ?

Ans: *Symbols : Symbols are divided in to the following two parts.*

l. Auxiliary *Symbols.*

ll. *Primary Symbols.*

Some of the *common symbols used in flowcharts* are shown below :

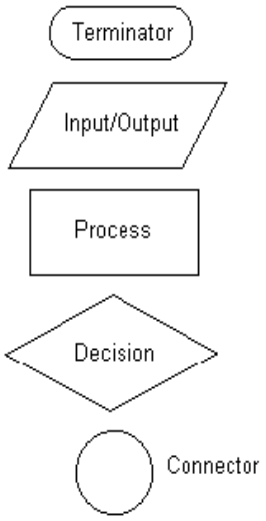


Fig: Flow chart Symbols

with flowchart , essential steps of an algorithm are shown using the shapes above. The flow of data between steps is indicated by arrows or flow lines.

9. Write an algorithm and flow chart for swapping two numbers

Ans: To Swap two integer numbers:

Algorithm : a. using third variable

Step 1 : Start
Step 2 : Input num1 , num2
Step 3 : temp = num1
Step 4 : num1 = num2
Step 5 : num2 = temp
Step 6 : Output num1 , num2
Step 7 : Stop

Algorithm : b. with out using third variable

Step 1 : Start
Step 2 : Input num1 , num2
Step 3 : calculate num1 = num1 + num2
Step 4 : calculate num2 = num1 – num2
Step 5 : calculate num1 = num1 – num2

Step 6 : Output num1 , num2
Step 7 : Stop

Flowcharts :

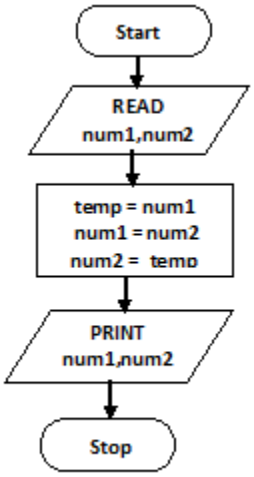


Fig a: With using third variable

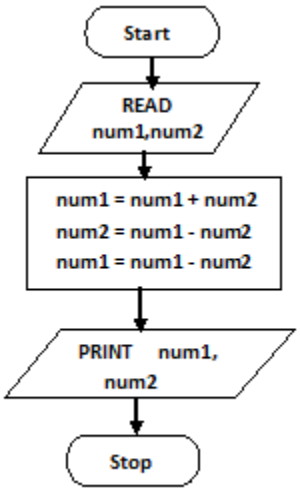


Fig b: Without using third variable

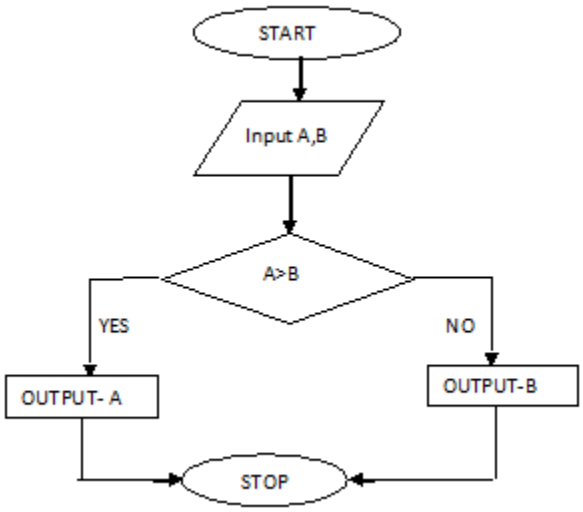
10. Write an algorithm and flowchart to find the largest among two numbers.

Ans:

Algorithm:

Step 1 : Start
Step 2 : Input A , B
Step 3 : if A > B then output A
else output B
Step 4 : Stop

Flowchart :

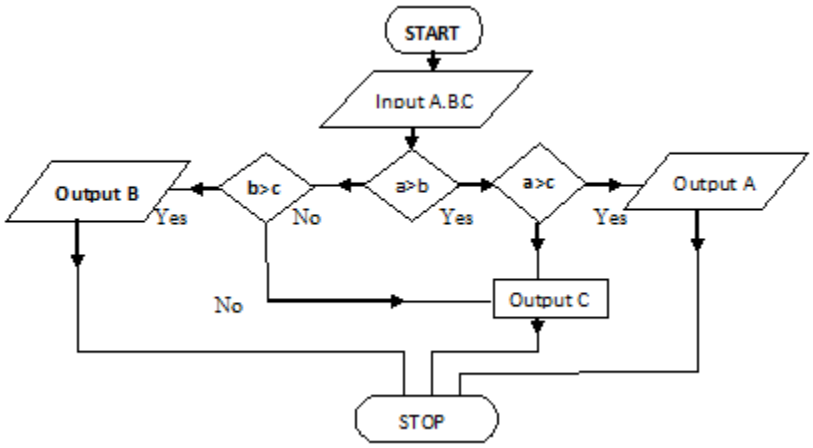


11. Write an algorithm and flowchart to find the largest among three numbers.

Algorithm: (method 1)

Step 1 : Start
Step 2 : Input A, B, C
Step 3 : if A > B go to step 4 , otherwise go to step 5
Step 4 : if A > C go to step 6 , otherwise go to step 8
Step 5 : if B > C go to step 7, otherwise go to step 8
Step 6 : print " A is largest " , go to step 9
Step 7 : print " B is largest ", go to step 9
Step 8 : print " C is largest " , go to step 9
Step 9 : Stop

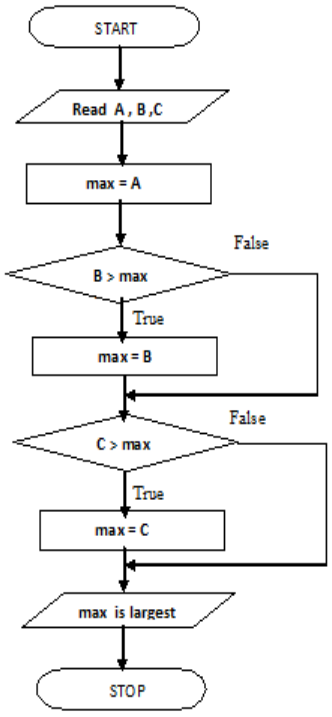
Flowchart:



Algorithm: b.

Step 1 : Start
Step 2 : Input A, B, C
Step 3 : Let max = A
Step 4 : if B > max then max = B
Step 5 : if C > max then max = C
Step 6 : output max is largest
Step 7 : Stop

Flowchart:



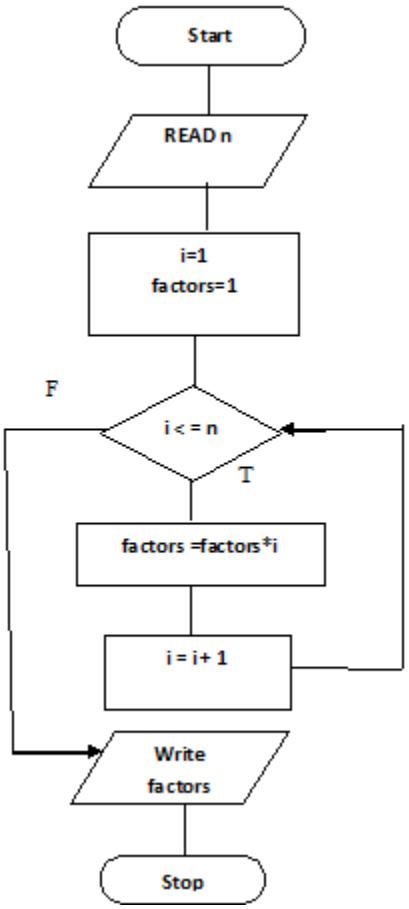
12. Write an algorithm and flowchart to find factorial of a number

Hint: fact (4) = 1 * 2* 3* 4

Sol. Algorithm:

Step 1: Start
Step 2: Input n
Step 3: Initialize counter variable, i , to 1 and factors = 1
Step 4: if i <= n go to step 5 otherwise go to step 7
Step 5: calculate factors = factors * i
Step 6: increment counter variable, i, and go to step 4
Step 7: output factors.
Step 8: stop

Flow chart:



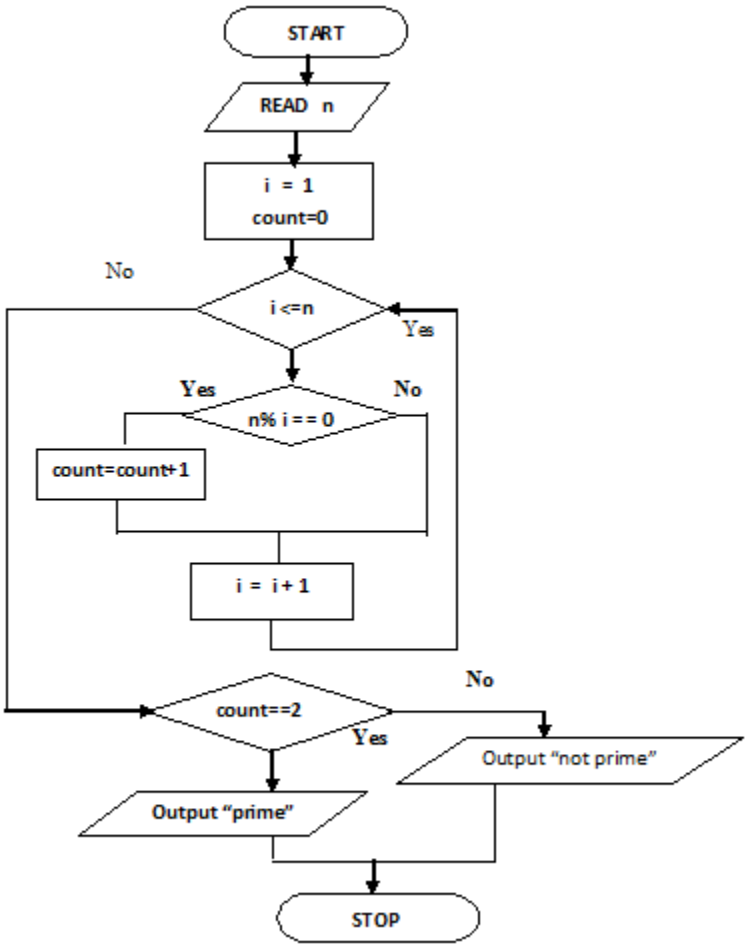
13. Write an algorithm and flowchart to find whether a number is prime or not.

Hint: A number is said to be prime number for which the only factors are 1 and itsel

Algorithm:

Step 1: Start
Step 2: Input n
Step 3: Let i = 1, count=0
Step 4: if i > n/2 go to step 7
Step 5: if (n % i = = 0) count = count + 1
Step 6: increment i and go to step 4
Step 7: if count=2 then print “prime number” else print “not prime number”
Step 8: Stop

Flow chart:



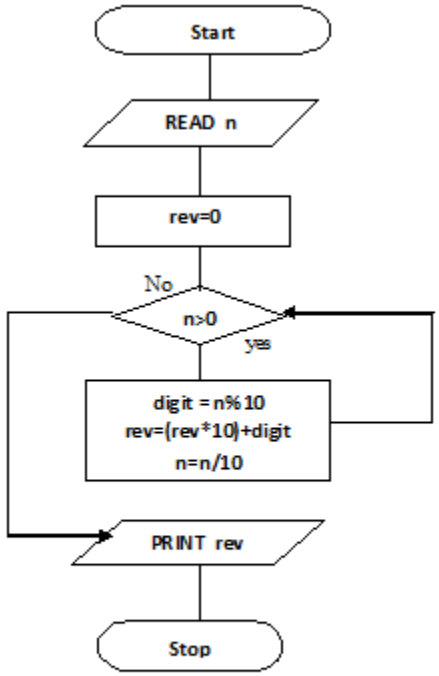
14. **Write an algorithm and flowchart to find reverse of a number.**

Algorithm:

Step 1: **Start**
Step 2: **Input n**
Step 3: **Let rev=0**
Step 4: **if n<=0 go to step 8**
Step 5: **digit = n % 10**

Step 6: **rev = (rev * 10) + digit**
Step 7: **n = n / 10 then go to step 4**
Step 8: **output rev**
Step 9: **Stop**

Flow chart:

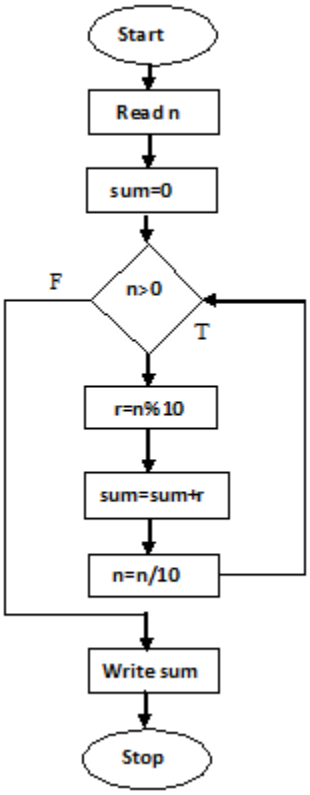


15. Write an algorithm and flowchart to find the Sum of individual digits if a given number

Algorithm :-

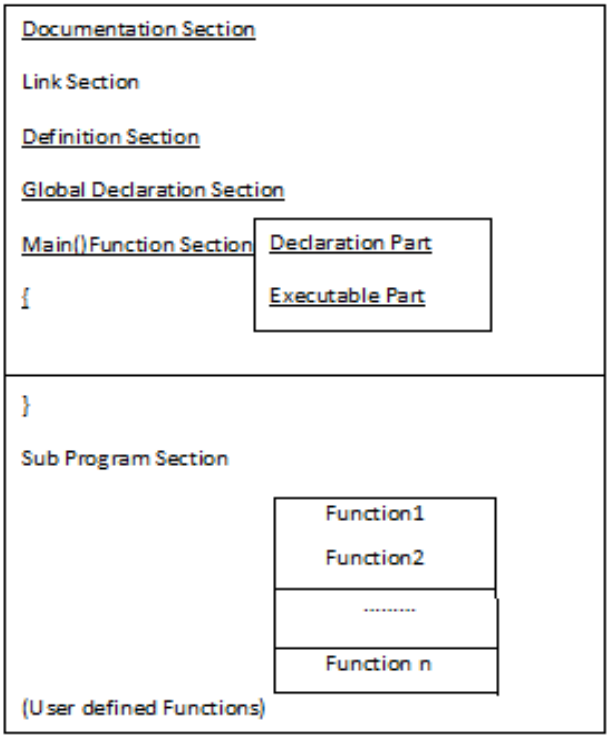
Step 1 : Start
Step 2 : read n
Step 3 : Sum = 0
Step 4 : While n > 0 do
4.1 : r = n % 10
4.2 : Sum = Sum + r
4.3 : n = n/10
Step 5 : End while
Step 6 : Write Sum
Step 7 : Stop

Flowchart :-



16. **what is the general structure of `C' program and explain with example?**

Ans: Structure of C program:



This section consists of a set of comment lines giving the name of the program, and other details. In which the programmer would like to user later.

Ex:- /*……*/

Link section: Link section provides instructions to the compiler to link functions from the system library.

Ex:- # include<stdio.h>

include<conio.h>

Definition section: Definition section defines all symbolic constants.

Ex:- # define A 10.

Global declaration section: Some of the variables that are used in more than one function throughout the program are called global variables and declared outside of all the functions. This section declares all the user-defined functions.

Every C program must have one main ( ) function section. This contains two parts.

i) Declaration part: This part declares all the variables used in the executable part.

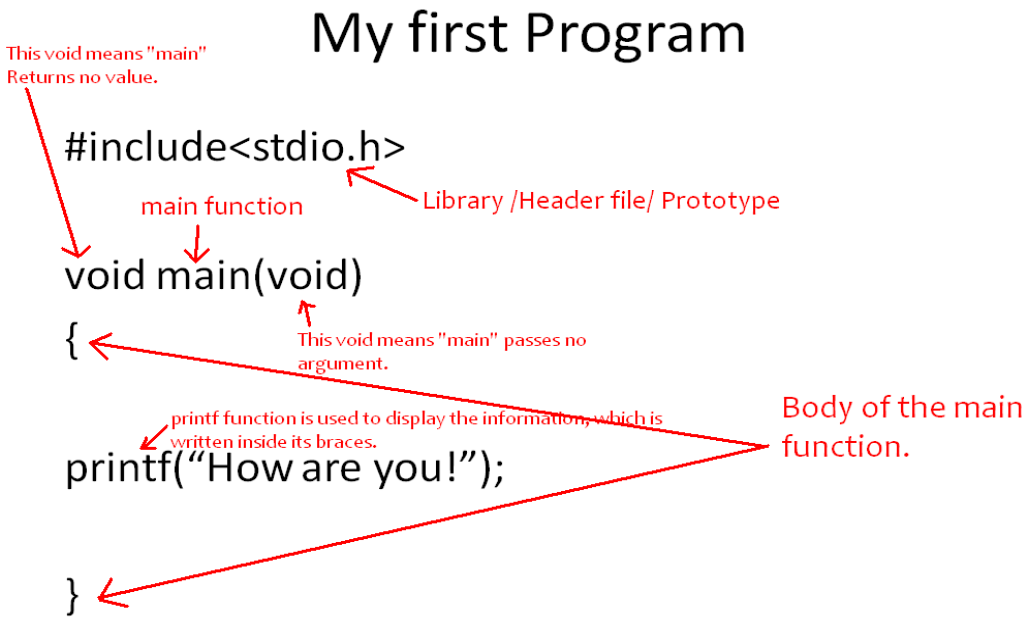
Ex:- int a,b;

ii) Executable part: This part contains at least one statement .These two parts must appear between the opening and closing braces. The program execution begins at the opening brace and ends at the closing brace. All the statements in the declaration and executable parts end with a semicolon (;).

Sub program section:

This section contains all the user-defined functions, that are called in the main () function. User-defined functions generally places immediately after the main() function, although they may appear in any order.

Ex:



17. **What is a variable? and write the rules for constructing variable?**

Variables: It is a data name that may be used to store a data value. It cont be changed during the execution of a program. A variable may taken different values at different times during execution.

Rules: Variable names may consist of letters, digits and under score(_) character.

- First char must be an alphabet or an '-'

- Length of the variable cont exceed upto 8 characters, some C compilers can be recognized upto 31 characters.
- No , and no white space, special symbols allowed.
- Variables name should not be a keyword.
- Both upper & lower case letters are used.

  Ex:- mark,sum1,tot-value,delhi→ valid

  Prics$, group one, char→ invalid

18. How to declare and initialize a variable?

Declaration does two things:

1. It tells the compiler what the variable name is.
2. It specifies what type of data the variable will hold.

The declaration of variables must be done before they are used in the program.

The syntax for declaring a variable is as follows:

Data-type v1,v2,…….,vn;

V1,v2,…vn are the names of variables. Variables are separated by commas. A declaration statement must end with a semicolon. For example , valid declarations are:

int count;

int number, total;

double ratio;

The simplest declaration of a variable is shown in the following code fragment:

Ex:

```
main()/*…….Program Name……..*/
{
/*……Declaration……..*/
float  x,y;
int code;
```

short int count;

long int amount;

double deviation;

unsigned n;

char c;

/*……**Computation**……*/

/*……**Program ends**……..*/

Initialization of variable :

Initialize a variable in c to assign it a starting value. Without this we can't get whatever happened to memory at that moment.

C does not initialize variables automatically. So if you do not initialize them properly, you can get unexpected results. Fortunately, C makes it easy to initialize variables when you declare them.

For Example :

int x=45;

int **month_lengths[]** = {23,34,43,56,32,12,24}**;**

struct **role** = **{** "Hamlet", 7, FALSE, "Prince of Denmark ", "Kenneth Branagh"};

Note : The initialization of variable is a good process in programming.

19. Explain data types in 'C'?

DATA TYPES

Data type is the type of the data that are going to access within the program. C supports different data types. Each data type may have pre-defined memory requirement and storage representation. C supports 4 classes of data types.

1. Primary or (fundamental) data type(int, char, float, double)
2. User-defined data type(type def)
3. Derived data type(arrays, pointers, structures, unions)
4. Empty data type(void)- void type has no value.

1 byte = 8 bits (0's and 1's)

1. Primary or (fundamental) data type

All C compilers support 4 fundamentals data types, they are

i) Integer (int)
ii) Character(char)
iii) Floating (float)
iv) Double – precision floating point(double)

1. Integer types:

Integers are whole numbers with a range of values supported by a particular machine. Integers occupy one word of storage and since the word size of the machine vary. If we use 16 bit word length the size of an integer value is -32768 to +32767. In order to control over the range of numbers and storage space, C has 3 classes of integer storage, namely short, long, and unsigned.

DATA TYPES	RANGE	Size	Control string
Int	-2^{15} to $2^{15}-1$ -32768 to +32767	2 bytes	%d (or) %i
Signed short int(or) short int	-128 to +127	1 byte	%d (or) %i
Unsigned short int	0 to 255	1 byte	%d (or) %i
unsigned int	0 to 65'535	2 bytes	%u
Un signed long int	0 to 4'294'967'295	4 bytes	
long int (or)signed long int	-2147483648 to +2147483647	4 bytes	%ld
long unsigned int	0 to 4294967295	4 bytes	%lu

2. Character type:-

Single character can be defined as a character (char) type data. Characters are usually stored in 8 bits of internal storage. Two classes of char types are there.

Signed char, unsigned char.

Signed char(or) char → 1 byte-→ -128 to +127→%c

Unsigned char→ 1 byte→ 0 to 255→%c

3. Floating point types:

Floating point (real) numbers are stored in 32 bits, with 6 digits of precision when accuracy provided by a float number is not sufficient

Float → 4 bytes→3.4e-38 to 3.4e+38→%f

4.Double precision type:

Double data type number uses 64 bits giving a precision of 14 digits. These are known as double precision no.s. Double type represents the same data type that float represents, but with a greater precision. To extend the precision further, we may we long double which uses 80 bits.

double→ 8 bytes → 1.7e-308 to 1.7e+308→%lf

long double→ 10 bytes → 3.4e-4932 to 1.1e+4932→ %lf

2. User defined data types:

C –supports a feature known as "type definition" that allows users define an identifier that would represents an existing type.

Ex:- typedef data-type identifier;

Where data-type indicates the existing datatype

Identifier indicates the new name that is given to the data type.

Ex:- typedef int marks;

Marks m1, m2, m3;

→typedef cont create a new data type ,it can rename the existing datatype. The main advantage of typedef is that we can create meaningful datatype names for increasing the readability of the program.

Another user-defined datatype is "enumerated data type(enum)"

Syntax: enum identifier{value1, value2,…….valuen};

Where identifier is user-defined datatype which is used to declare variables that can have one of the values enclosed within the braces. Value1 ,value2,.valuen all these are known as enumeration constants.

Ex:- enum identifier v1, v2,……vn

V1=value3;
V2=value1;……
Ex:- enum day {Monday,Tuesday…… sunday};
Enum day week-f,week-end

Week-f = Monday
-
-
-
-
(or)
enum day{Monday…Sunday}week-f, week-end;

20. (a) Describe the different types of constants in C with example?

(b) What are the rules for creating C constants explain with example?

TYPES OF C CONSTANTS

1. Integer constants
2. Real constants
3. Character constants
4. String constants

1. Integer constants: An integer constant refers to a sequence of digits. There are three types of integers, namely, decimal integer, octal integer and hexadecimal integer.

Examples of Integer Constant**:**
426 **,**+786 , -34(decimal integers)

037, 0345, 0661(octal integers)

0X2, 0X9F, 0X (hexadecimal integers)

2. Real constants: These quantities are represented by numbers containing fractional parts like 18.234. Such numbers are called real (or floating point) constants.

Examples of Real Constants:

+325.34

426.0

-32.67 etc.

The exponential form of representation of real constants is usually used if the value of the constant is either too small or too large. In exponential form of representation the Real Constant is represented in two parts. The first part present before 'e' is called **Mantissa** and the part following 'e' is called **Exponent**.

For ex. .000342 can be written in Exponential form as 3.42e-4.

3. Single Character constants: Single character constant contains a single character enclosed within a pair of single quote marks**.**

For ex. 'A','5','+',' '

Note that the character constant'5' is not same as the number 5. The last constant is a blank space.Character constant has integer values known as ASCII values. For example, the statement

Printf("%d",a); would print the number 97,the ASCII value of the letter a. Similarly, the statement **printf("%c",97);** would output the letter 'a'

String constants: A string constant is a sequence of character enclosed in double quotes. the characters may be letters, numbers, special characters and blank space.

Examples are:

"HELLO!"

"1979"

"welcome"

"?.....!"

"5+3"

"X"

RULES OF CONSTRUCTING INTEGER CONSTANTS

a. an integer constant must have at least one digit.

b. It must not have a decimal point.

c. It can be either positive or negative.

d. The allowable range for constants is -32768 to 32767

In fact the range of integer constants depends upon compiler.

For ex. 435

+786

-7000

RULES OF CONSTRUCTING REAL CONSTANTS

a. A real constants must have at least one digit

b. it must have a decimal point.

c. it could be either positive or negative.

d. default sign is positive.

For ex. +325.34

426.0

In exponential form of representation, the real constants is represented in two parts. The part appearing before 'e' is called mantissa where as the part following 'e' is called exponent.

Range of real constants expressed in exponential form is -3.4e38 to 3.4e38.

Ex. +3.2e-5

RULES OF CONSTRUCTING CHARACTER CONSTANTS

a. a character constant is a single alphabet, a single digit or a single special symbol enclosed within single inverted commas.

b. The maximum length of character constant can be one character.

Ex `A`

21. What is an operator and List different categories of C operators based on their functionality? Give examples?

Operators:

An operator is a symbol performs certain mathematical or logical manipulations. Operators are used in programs to manipulate data variables.

C operators can be classified into a number of categories, they are

1. Arithmetic Operators

2. Relational Operators

3. Logical Operators

4. Assignment Operators

5. Increment and decrement operators

6. Conditional operators

7. Bitwise Operators

8. Special operators

1. Arithmetic Operators:

The arithmetic operators are

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo division

Here a and b are operands, assign values for a=14 and b=4 we have the following results

a-b = 10

a+b = 18

a*b = 56

a/b = 3(coefficient)

a%b = 2(remainder)

2.Relational Operators:

Relational operators are used for comparing two quantities, and take certain decision. For example we may compare the age of two persons or the price of two items….these comparisons can be done with the help of relational operators.

An expression containing a relational operator is termed as a relational expression. The value of a relational expression is either one or zero. It is one if the specified relation is true and zero if the relation is false.

Ex:- 13<34 (true) 23>35(false)

C supports 6 relational operators

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

Ex:- 4.5<=10(true)

6.5<-10(false)

10<4+12(true)

When arithmetic expression are used on either side of a relational operator, the arithmetic expression will be evaluated first and then the results compared, that means arithmetic operators have a higher priority over relational operators.

3. Logical Operator:

C has 3 logical operators. The logical operators are used when we want to test more than one condition and make decisions.

Operator	Meaning
&&	Logical AND
||	Logical OR

!	Logical NOT

The logical operators && and || are used when we test more than one condition and make decisions.

Ex:- a>b && x==45

This expression combines two or more relational expressions, is termed as a logical expression or a compound relational expression. The logical expression given above is true only if a>b is true and x==10 is true. if both of them are false the expression is false.

OP1	**OP2**	**&&**	**||**
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Some examples of logical expression

1. if (age >55 && salary<1000)
2. if(number<0 || number>100)

4. Assignment operator:

These operators are used to assign the result of an expression to a variable. The usual assignment operator is **'='**

V op = exp;

Where v is a variable, exp is an expression and op is a C binary arithmetic operator. The operator op=is known as the shorthand assignment operator.

The assignment statement is V op=exp;

Ex:- X= X+(Y+1);

a*=a;-----→ a=a*a;

5. Increment and Decrement operator:

++ and - - are increment and decrement operators in C. The operator ++ adds 1 to the operand, while - - subtracts 1.both are unary operators.

++m; or m++; is equal to m=m+1(m+=1;)

--m; or m- - is equal to m=m-1(m- = 1;)

We use the increment and decrement statements in for and while loops extensively.

Ex:- m=5;

Y=++m; the value of y=6 and m =6.

Suppose if we write the above statement as

m=5; y= m++; the value of y=5 and m=6.

A prefix operator first adds 1 to the operand and then the result is assigned to the variable on left. On the other hand, a postfix operator first assigns the value to the variable on left and then increments the operand.

6. Conditional operator:

A ternary operator pair"? :" is available in C to construct conditional expressions of the form

exp ? exp : exp3

Where exp1,exp2 and exp3 are expressions,

The operator?: works as follows: exp1 is evaluated first. If it is non-zero (true), then the expression exp 2 is evaluated and becomes the value of the expression. If exp1 is false, exp3 is evaluated and its value becomes the value of the expression.

Ex:- a=10; b=45;

X = (a>b) ? a:b;

o/p:- X value of b (45).

7. Bitwise Operator:

C supports a special operator knows as bitwise operators for manipulation of data at bit level. These operators are used for testing the bits, or shifting them right to left. Bitwise operators may not be applied to float or double.

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	Shift left
>>	Shift right

8. Special operators:

C supports some special operators such as comma operator, size of operator, pointer operator (& and *) and member selection operators (. and ->).

The comma operator: The comma operator is used to link the related expression together. A comma linked list of expressions is evaluated left to right and the value of right- most expression is the value of the combined expression. For example, the statement

Value = (x=10, y=5, x+y);

In for loops: for (n=1 , m=10, n<=m; n++, m++);

The sizeof operator: The sizeof is a compile time operator and when used with an operand, it returns the number of bytes the operand occupies. The operand may be variable, a constant or a data type qualifier.

m = sizeof (sum);

n = sizeof (long int);

The sizeof operator is normally used to determine the lengths of arrays and structures when their sizes are not known to the programmer. It is also used to allocate memory space dynamically to variables during execution of a program.

22. Explain the types of type conversions in C with example?

Type conversions: converting a variable value or a constant value temporarily from one data type to other data type for the purpose of calculation is known as type conversion.

There are two types of conversions

1. automatic type conversion (or) implicit
2. casing a value (or) explicit

1. Implicit: In this higher data type can be converted into lower data type.

* Float value can be converted into integral value by removing the fractional part.

* Double value can be converted into float value by rounding of the digits.

* Long int can be converted into int value by removing higher order bits.

2. Explicit: In this type of conversion, the programmer can convert one data type to other data type explicitly.

Syntax: (datatype) (expression)

Expression can be a constant or a variable

Ex: y = (int) (a+b)

y= cos(double(x))

double a = 6.5

double b = 6.5

int result = (int) (a) + (int) (b)

result = 12 instead of 13.

➔ int a=10

float(a)->10.00000

23. **Define an expression? How can you evaluate an expression?**

Expressions:

An expression in C is some combination of constants, variables, operators

and function calls.

Sample expressions are:

a + b

tan(angle)

• Most expressions have a value based on their contents.

• A statement in C is just an expression terminated with a semicolon.

For example:

sum = x + y + z;

printf("Go Buckeyes!");

The rules given below are used to evaluate an expression,

1)If an expression has parenthesis, sub expression within the parenthesis is evaluated first and arithmetic expression without parenthesis is evaluated first.

2)the operators high level precedence are evaluated first.

3)the operators at same precedence are evaluated from left to right or right to

left depending on the associativity of operators.

Expressions are evaluated using an assignment statement of the form:

variable = expression;

Variable is any valid C variable name. when the statement is encountered, the expression is evaluated first and the result then replaces the previous value of the variable on the left-hand side. All variables used in the expression must be assigned values before evaluation is attempted.

Ex:- x = a*b-c;

Y = b/c*a;

Z = a-b / c+d;

Ex:- x= a-b/3+c*2-1 when a=9, b=12, and c=3 the expression becomes.

x = 9-12/3 +3*2-1

Step1: x = 9-4+3*2-1

Step2: x = 9-4+6-1

Step3: x = 5+6-1

Step4: x = 11-1

Step5: x = 10

24. **Define precedence and associativity? Give an example?**

or

Explain the hierarchy (priority) and associativity(clubbing)of operators in 'C' with example?

Operator Precedence:

Various relational operators have different priorities or precedence. If an arithmetic expression contains more operators then the execution will be performed according to their properties. The precedence is set for different operators in C.

Type of operator	Operators	Associativity
Unary operators	+,-,!,++,--,type,~,size of	Right to left
Arithmetic operators	*, /,%,+,-	Left to right
Bit – manipulation operators	<<,>>	Left to right
Relational operators	>,<,>=,<=,==,!=	Left to right
Logical operators	&&,||	Left to right
Conditional operators	?,:	Left to right
Assignment operators	=,+=,-=,*=,/=,%=	Right to left

Important note:

→ Precedence rules decide the order in which different operators are applied

➔ Associativity rule decides the order in which multiple occurrences of the same level operator are applied.

Hierarchy Of Operations In C

There are some operators which are given bellow with their mean. The higher the position of an operator is, higher is its priority.

Operator	Type
!	Logical NOT
*/ %	Arithmetic and modulus
+ -	Arithmetic
<> <=>=	Relational
==!=	Relational
&&	Logical AND
||	Logical OR
=	Assignment

ASSOCIATIVITY OF OPERATOR

When an expression contains two operators of equal priority the tie between them is settled using the associatively of the operators.

Associatively can be of two types—Left to Right or Right to Left.
Left to Right associatively means that the left operand must be unambiguous. Unambiguous in what sense? It must not be involved in evaluation of any other sub-expression. Similarly, in case of Right to Left associatively the right operand must be unambiguous. Let us understand this with an example.
Consider the expression
a = 3 / 2 * 5 ;

Here there is a tie between operators of same priority, that is between / and *. This tie is settled using the associatively of / and *. But both enjoy Left to Right associatively.

While executing an arithmetic statement, which has two or more operators, we may have some problem as to how exactly does it get executed.

Priority	Operators	Description
1st	*, / , %	multiplication, division, modular division
2nd	+, -	addition, subtraction
3rd	=	assignment

For Example :
i= 2*3/4+4/4+8-2+5/8
i=6/4+4+8-2+5/8
i=1+4/4+8-2+5/8
i=1+1+8-2+5/8
i=1+1+8-2+0
i=2+8-2+0
i=10-2+0
i=8+0
i=8

25. (a) Explain the formatted I/O functions with example? (8)

(b) Explain unformatted (character oriented) I/O functions with example?(7)

Managing input and output operations:

Reading, processing and writing of data are the three essential functions of a computer program. Most programs take some data as input and display the processed data. We have two methods of providing data to the program variables. One method is to assign values to variables through the assignment statements like x=5, a=0 and so on. Another method is to use the input function scanf, which can read data from a keyboard. We have used both the methods in programs. For

outputting results, we have used extensively the function printf, which sends results out to a terminal.

Input – Output functions:-

- The program takes some I/P- data, process it and gives the O/P.
- We have two methods for providing data to the program

  i) Assigning the data to the variables in a program.
  ii) By using I/P-O/P statements.

C language has 2 types of I/O statements; all these operations are carried out through function calls.

1. Unformatted I/O statements
2. Formatted I/O statements

Unformatted I/O statements:-

getchar( ):- It reads single character from standard input device. This function don't require any arguments.

Syntax:- char variable_name = getchar( );

Ex:- char x;

x = getchar( );

putchar ( ):- This function is used to display one character at a time on the standard output device.

Syntax:- putchar(char_variable);

Ex:- char x;

Putchar(x);

Ex:- main( )

{

Char ch;

Printf("enter a char");

Ch = getchar( );

Printf("enter char is");

Putchar(ch);

}

getc() :- This function is used to accept single character from the file.

Syntax: char variable_name = getc();

Ex:- char c;

c = getc();

putc():- This function is used to display single character.

Syntax:- putc(char variable_name);

Ex:- charc;

Putc(c);

These functions are used in file processing.

gets( ):- This function is used to read group of characters(string) from the standard I/P device.

Syntax:- gets(character array variable);

Ex:- gets(s);

Puts( ):- This function is used to display string to the standard O/P device.

Syntax:- puts(char array variables);

Ex:- puts(s);

Ex:- main()

{
char s[10];

Puts("enter name");

```
gets(s);

puts("print name");

puts(s);

}
```

getch():- This function reads single character directly from the keyboard without displaying on the screen. This function is used at the end of the program for displaying the output (without pressing (Alt-F5).

Syntax: char variable_name = getch();

```
Ex:-   char c

       c = getch();
```

getche():- This function reads single character from the keyboard and echoes(displays) it to the current text window.

Syntax:- char variable_name = getche();

```
      Ex:- char c

           c = getche();
```

```
ex:-  main()

{
char ch, c;

Printf("enter char");

ch = getch();

printf("%c", ch);

printf("enter char");

c = getche();

printf("%c",c);

}
```

O/P:- enter character a

Enter character b

b

Character test function:-

Function	Test
isainum(c)	Is c an alphanumeric character?
isalpha(c)	Is c an alphabetic character
isdigit(c)	Is c is a digit?
islower(c)	Is c lower case letter?
isprint(c)	Is c printable character?
ispunct(c)	Is c a punctuation mark?
isspace(c)	Is c a while space character?
isupper(c)	Is c an upper case letter?
tolower(c)	Convert ch to lower case
toupper(c)	Convert ch to upper case

Ex:- main()

{

Char a;

Printf("enter char");

a = getchar();

if (isupper(a))

{

x= tolower(a);

putchar(x);

}

else

```
putchar(toupper(a));

}
```

O/P:- enter char A

a

Formatted I/O Functions: Formatted I/O refers to input and output that has been arranged in a particular format.

Formatted I/P functions → scanf( ) , fscanf()

Formatted O/P functions---→ printf() , fprintf()

scanf( ) :- scanf() function is used to read information from the standard I/P device.

Syntax:- scanf("controlstring", &variable_name);

Ex:- int n;

Scanf("%d",n);

Control string represents the type of data that the user is going to accept. & gives the address of variable.(char-%c , int-%d , float - %f , double-%lf). Control string and the variables going to I/P should match with each other.

Simple format specification as follows

%w type – specified ex:- %4d , %6c……

Here 'w' represents integer value specifies total number of columns.

Ex:- scanf("%5d",&a); a = 4377

	4	3	7	7

Printf( ): This function is used to output any combination of data. The outputs are produced in such a way that they are understandable and are in an easy to use form. It is necessary for the programmer to give clarity of the output produced by his program.

Syntax:- printf("control string", var1,var2……);

Control string consists of 3 types of items

1. Chars that will be printed on the screen as they appear.
2. Format specifications that define the O/P format for display of each item.
3. Escape sequence chars such as \n , \t and \b…..

O/P of integer number: -

Format specifications %wd

Format	**O/P**
Printf("%d", 9876)	9 8 7 6
Printf("%6d",9876)	_ _ 9 8 7 6
Printf("%2d",9876)	9 8
Printf("%-6d",9876)	9 8 7 6 _ _
Printf("%06d",9876)	0 0 9 8 7 6

O/P of real number: %w.pf

w---- Indicates minimum number of positions that are to be used for display of the value.

p------Indicates number of digits to be displayed after the decimal point.

Format **y=98.7654**	**O/P**
Printf("%7.4f",y)	9 8 . 7 6 5 4
Printf("%7.2f",y)	_ _ 9 8 . 7 7
Printf("%-7.2f",y)	9 8 . 7 7 _ _
Printf("%10.2e",y)	_ _ 9 . 8 8 e + 0 1

O/P of single characters and string:

%wc %ws

Format O/P

%s

R	a	j	u		R	a	j	e	s	h		R	a	n	i

%18s

	R	a	j	u		R	a	J	e	s	h		R	a	n	i

%18s

R	A	j	u		R	a	j	E	s	h		R	a	n	i	