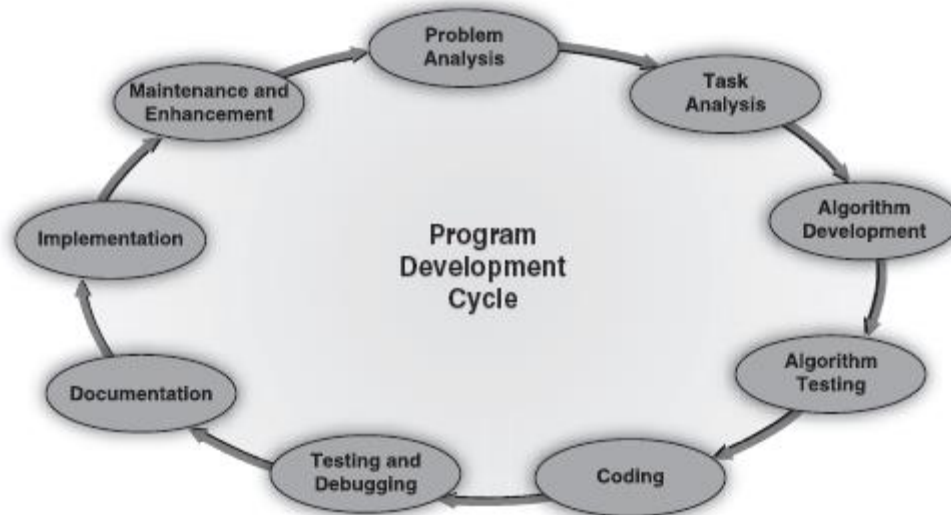


UNIT 1

CHAPTER 1: PROBLEM SOLVING TECHNIQUES

Program Development Cycle

the process of writing a program (coding), the programmer has to determine the problem that needs to be solved. There are different approaches to problem solving. One common approach is to use the *program development cycle*.



- **Problem Analysis:** The problem is analysed precisely and completely. Based on understanding, the developer knows about the scope within which the problem needs to be developed.
- **Task Analysis:** After analysing the problem, the developer needs to develop various solutions to solve the given problem. From these solutions, the optimum solution is chosen, which can solve the problem comfortably and economically.
- **Algorithm Development:** After selecting the appropriate solution, an algorithm is developed to depict the basic logic of the selected solution. An algorithm depicts the solution in logical steps (sequence of instructions). Further, an algorithm is represented by flowcharts, decision tables and pseudocodes (informal way of programming description). These tools make the program logic clear and they eventually help in coding.
- **Algorithm Testing:** Before converting the algorithms into actual code, it should be checked for accuracy. The main purpose of checking the algorithm is to identify major logical errors at an early stage because logical errors are often difficult to detect and correct at later stages. The testing also ensures that the algorithm is a “true” one and it should work for both normal as well as unusual data.
- **Coding:** After meeting all the design considerations, the actual coding of the program takes place in the chosen programming language. Depending upon the application domain and available resources, a program can be written by using

computer languages of different levels such as machine, assembly or high-level languages (HLL).

- **Testing and Debugging:** It is common for the initial program code to contain errors. A program compiler and programmer-designed test data machine tests the code for syntax errors. The results obtained are compared with results calculated manually from these test data. Depending upon the complexity of the program, several rounds of testing may be required.
- **Documentation:** Once the program is free from all the errors, it is the duty of the program developers to ensure that the program is supported by suitable documentation. These documents should be supplied to the program users. Documenting a program enables the user to operate the program correctly. It also enables other persons to understand the program clearly so that it may, if necessary, be modified, or corrected by someone other than the original programmer.
- **Implementation:** After documentation, the program is installed on the end user's machine and the user is also provided with all the essential documents in order to understand how the program works. The implementation can be viewed as the final testing because only after using the program the user can point out the drawbacks (if any) and report them to the developers. Based on the feedback from users, the programmers can modify or enhance the program.
- **Maintenance and Enhancement:** After the program is implemented, it should be properly maintained by taking care of the changing requirements of its users and the system. The program should be regularly enhanced by adding additional capabilities. This phase is also concerned with detecting and fixing the errors, which were missed in the testing phase. Since this step generates user feedback, the programming cycle continues as the program is modified or reconstructed to meet the changing needs.

ALGORITHM

Algorithms are one of the most basic tools that are used to develop the problem-solving logic. An *algorithm* is defined as a finite sequence of explicit instructions that when provided with a set of input values produces an output and then terminates

Algorithm Properties

Algorithms are not computer programs, as they cannot be executed by a computer. Some properties of algorithm are as follows.

- There must be no ambiguity in any instruction.
- There should not be any uncertainty about which instruction is to be executed next.
- The description of the algorithm must be finite. An algorithm cannot be open ended (there should be some limit).
- The algorithm should terminate after a finite number of steps.

- The algorithm must be general enough to deal with any contingency (possible but cannot be predicted).

Advantages of algorithm

- it is a step-by-step representation of a solution to a given problem ,which is very easy to understand
- it has got a definite (clearly stated or decided) procedure.
- it easy to first develop an algorithm, then convert it into a flowchart &then into a computer program.
- it is independent of programming language.
- it is easy to debug as every step is got its own logical sequence

Disadvantages of algorithm

It is time consuming & cumbersome (complicated) as an algorithm is developed first which is converted into flowchart &then into computer program

Type of Algorithms :

The algorithm and flowchart, classification to the three types of *control structures*. They are:

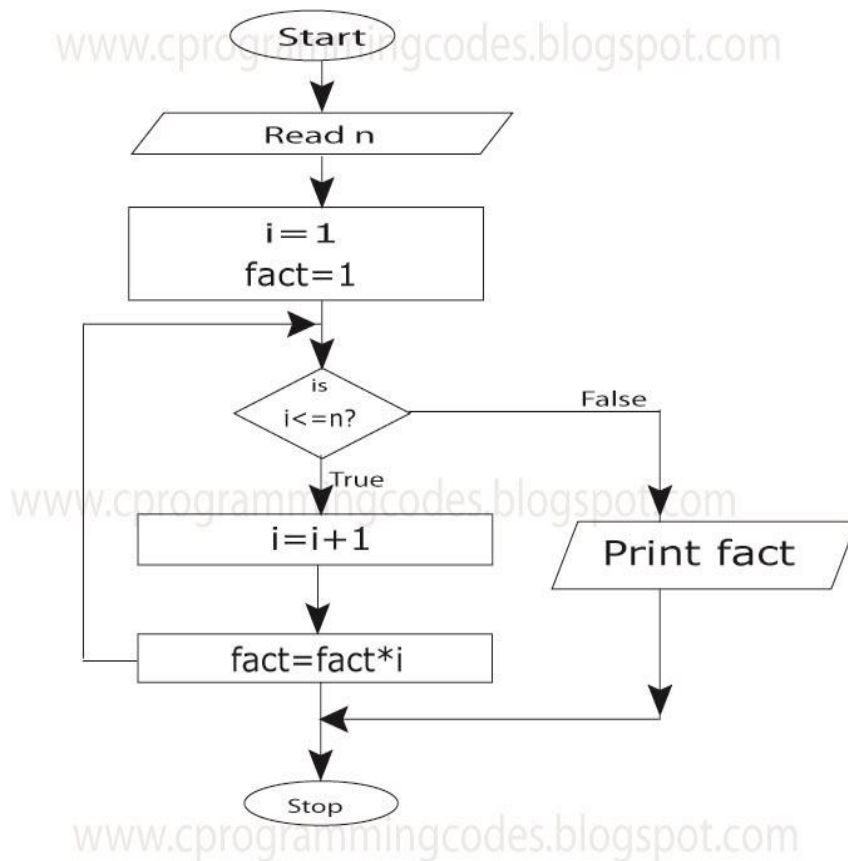
1. Sequence (particular order)
2. Branching (Selection)
3. Loop (Repetition)

Q)EXAMPLE:Write algorithm to calculate the sum and average of two numbers.

Step1 : Start
 Step2 : Read two numbers n,m
 Step3 : Calculate $\text{sum} = n + m$
 Step4 : Calculate $\text{avg} = \text{sum} / 2$
 Step5 : Print sum,avg
 Step 6 : Stop

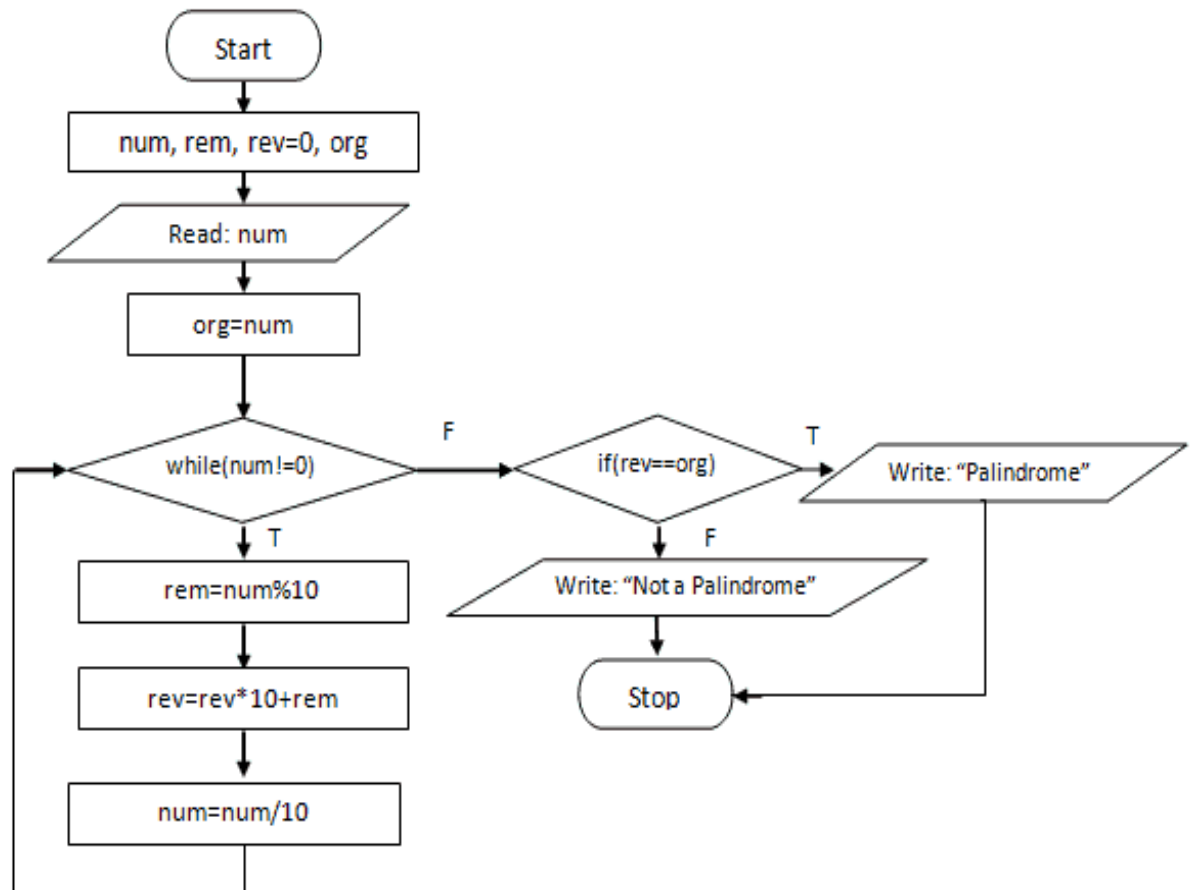
Q)WRITE AN ALGORITHM TO FIND FACTORIAL OF N NUMBER

Step 1:start
 Step 2: set $i=1, \text{fact}=1, n=5$
 Step 3:while($i \leq n$)
 (a) $\text{fact} = \text{fact} * i$
 (b) $i = i + 1$
 [end loop]
 Step 4:stop



Q)WRITE AN ALGORITHM CHECK FOR PALINDROME AND DRAW FLOWCHART

Step 1:start
 Step 2:read a number num
 Step 3 :set rev=0,rem=0
 Step 4:while num>0 true continue else goto step 8
 Step 5 :set rem=num%10
 Step 6:set rev=rev*10+rem
 step 7:set num=num/10 go to step 4
 Step 8:print rev
 Step 9:stop



Q) Write algorithm to find largest among three number AND FLOWCHART

Step 1;start

Step 2:read a,b,c

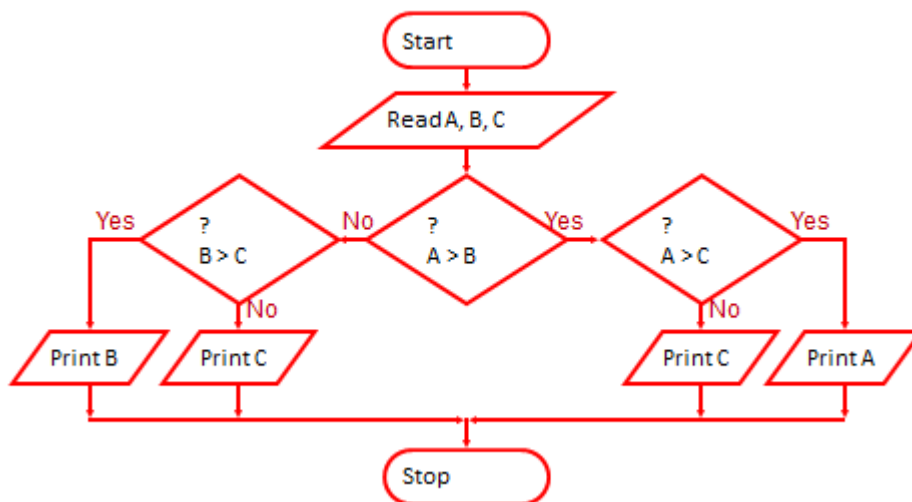
Step 3:if(a>b)and(a>c)then print a goto step 6 else goto step4

Step 4:if(b>c) then print b and goto step 6

else goto step 5

Step 5:print c

Step 6:stop



Q)WRITE AN ALGORITHM TO CHECK GIVEN NUMBER IS PRIME OR NOT

Step 1:start

Step2:[read a number] N

STEP 3:IF(N<=3)THEN

PRINT(NUMBER IS PRIME NO)

GOTO STEP 8

STEP 4:SET I=2,FLAG=0,

STEP 5:REPEAT STEP 6 WHILE(I<N)

STEP 6:IF(N%I==0),THEN

FLAG=1

BREAK ///OUT OF LOOP

ELSE

I=I+1;

[END IF]

[END LOOP]

STEP 7:IF(FLAG==1)THEN

PRINT(N NOT IS PRIME NUMBER)

ELSE

PRINT(N IS PRIME NUMBER)

STEP 8:STOP

OR

Check given number is prime or not

Step-1: start

Step-2: Read a "n" value to check prime or not

Step 3: set $i=1$, $count=0$.

Step 4: if $i \leq n$ if true go to step 5, else go to step 8

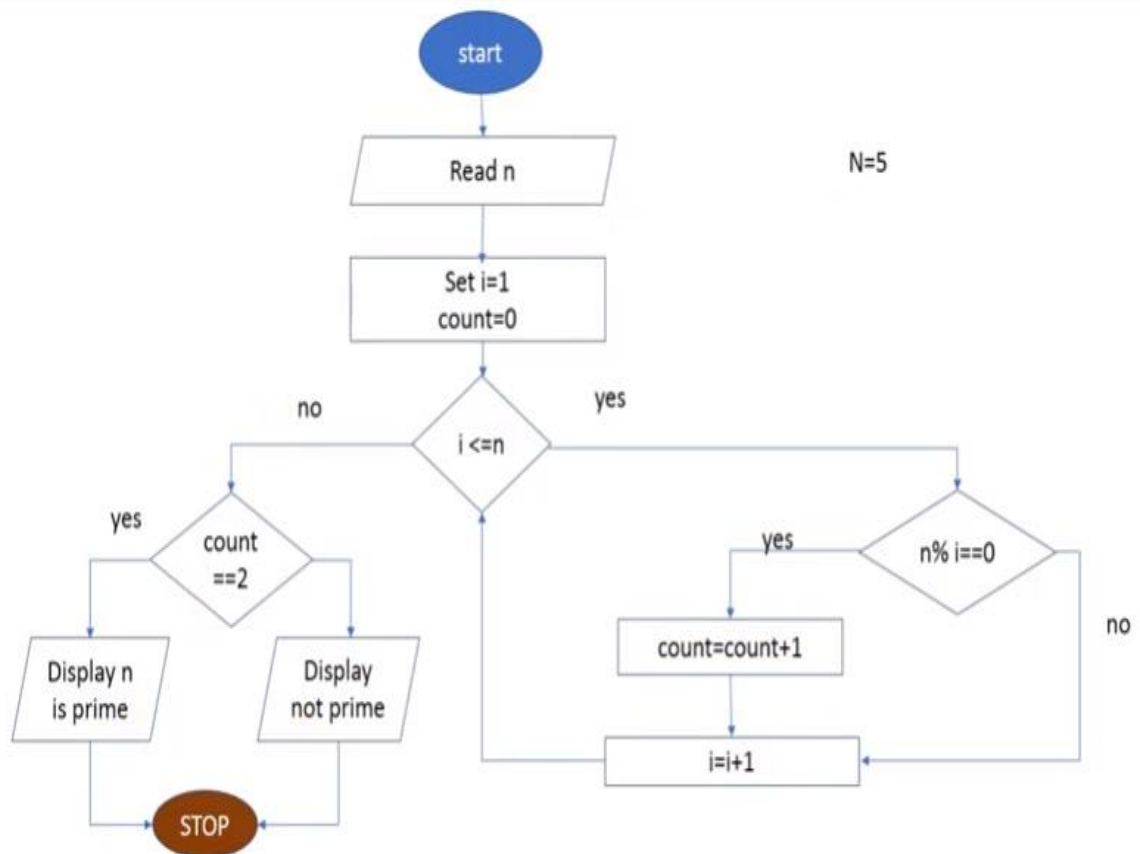
Step 5: check the condition $n \% i == 0$ if true then evaluate step 6, false go to step 7.

Step 6: set $count = count + 1$

Step 7: $i = i + 1$ go to step 4

Step 8: check count, if $count = 2$ display prime , if not display it is not prime

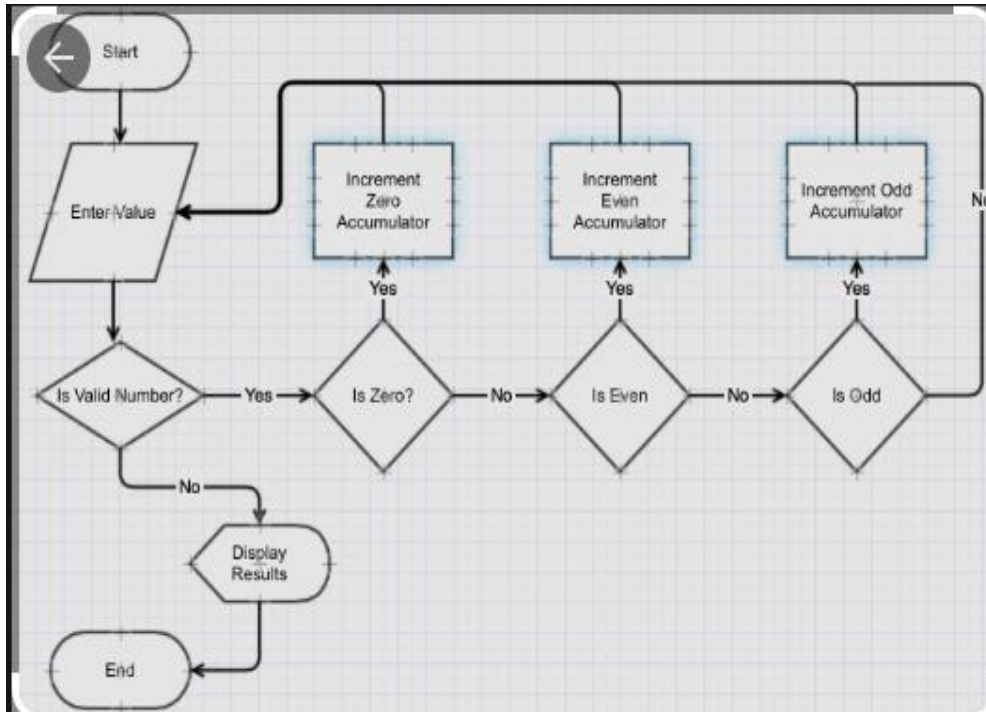
Step 9: Stop



WRITE AN ALGORITHM TO COUNT NUMBER OF ODD, EVEN AND ZEROS IN A LIST OF INTEGERS.








- **Step1:**Input the number of elements of the array.
- **Step2:**Input the array elements.
- **Step3:**Initialize $count_zero = count_odd = count_even = 0$.









- **Step4:** Traverse the array and increment count_zero if the array element is equal to zero, increment count_odd if the array element is odd, else increment count_even.
- **Step5:** Print count_zero, count_odd and count_even.



Flowchart:

- 1. Graphical representation of any program is called flowchart.
- 2. There are some standard graphics that are used in flowchart as following

Symbol	Symbol Name	Description
	Flow lines	Flow lines are used to connect symbols used in flowchart and indicate direction of flow.
	Terminal (START / STOP)	This is used to represent start and end of the flowchart.
	Input / Output	It represents information which the system reads as input or sends as output.
	Processing	Any process is represented by this symbol. For example, arithmetic operation, data movement.
	Decision	This symbol is used to check any condition or take decision for which there are two answers. Yes (True) or No (False).
	Connector	It is used to connect or join flow lines.
	Off-page Connector	This symbol indicates the continuation of flowchart on the next page.

	Document	It represents a paper document produced during the flowchart process.
	Annotation	It is used to provide additional information about another flowchart symbol which may be in the form of descriptive comments, remarks or explanatory notes.
	Manual Input	It represents input to be given by a developer or programmer.
	Manual Operation	This symbol indicates that the process has to be done by a developer or programmer.
	Online Storage	It represents online data storage such as hard disks, magnetic drums or other storage devices.
	Offline Storage	It represents offline data storage such as sales on OCR, data on punched cards.
	Communication Link	It represents the data received or to be transmitted from an external system.
	Magnetic Disk	It represents data input or output from and to a magnetic disk.

ADVANTAGES OF FLOWCHART

- **Makes Logic Clear:** The main advantage of using a flowchart to plan a task is that it provides a pictorial representation of the task, which makes the logic easier to follow. The symbols are connected in such a way that they show the movement (flow) of information through the system visibly. The steps and how each step is connected to the next can be clearly seen. Even less experienced personnel can trace the actions represented by a flowchart, that is, flowcharts are ideal for visualizing fundamental control structures employed in computer programming.
- **Communication:** Being a graphical representation of a problem-solving logic, flowcharts are a better way of communicating the logic of a system to all concerned. The diagrammatical representation of logic is easier to communicate to all the interested parties as compared to actual program code as the users may not be aware of all the programming techniques and jargons.
- **Effective Analysis:** With the help of a flowchart, the problem can be analysed in an effective way. This is because the analysing duties of the programmers can be delegated to other persons, who may or may not know the programming techniques, as they have a broad idea about the logic. Being outsiders, they often tend to test and analyse the logic in an unbiased manner.
- **Useful in Coding:** The flowcharts act as a guide or blueprint during the analysis and program development phase. Once the flowcharts are ready, the programmers can plan the coding process effectively as they know where to begin and where to end, making sure that no steps are omitted. As a result, error-free programs are developed in HLL and that too at a faster rate.
- **Proper Testing and Debugging:** By nature, a flowchart helps in detecting the errors in a program, as the developers know exactly what the logic should do. Developers can test various data for a process so that the program can handle every contingency.
- **Appropriate Documentation:** Flowcharts serve as a good program documentation tool. Since normally programs are developed for novice users, they can take the help of the program documentation to know what the program actually does and how to use the program.

Limitations of Flowcharts

- **Complex:** The major disadvantage in using flowcharts is that when a program is very large, the flowcharts may continue for many pages, making them hard to follow. Flowcharts tend to get large very quickly and it is difficult to follow the represented process. It is also very laborious to draw a flowchart for a large program. You can very well imagine the nightmare when a flowchart is to be developed for a program, consisting of thousands of statements.
- **Costly:** Drawing flowcharts are viable only if the problem-solving logic is straightforward and not very lengthy. However, if flowcharts are to be drawn for a huge application, the time and cost factor of program development may get out of proportion, making it a costly affair.

- **Difficult to Modify:** Due to its symbolic nature, any changes or modification to a flowchart usually requires redrawing the entire logic again, and redrawing a complex flowchart is not a simple task. It is not easy to draw thousands of flow lines and symbols along with proper spacing, especially for a large complex program.
- **No Update:** Usually programs are updated regularly. However, the corresponding update of flowcharts may not take place, especially in the case of large programs. As a result, the logic used in the flowchart may not match with the actual program's logic. This inconsistency in flowchart update defeats the main purpose of the flowcharts, that is, to give the users the basic idea about the program's logic.

PSEUDOCODE

Pseudocode

Pseudocode is made up of two words: *pseudo* and *code*. **Pseudo** means imitation and **code** refers to instructions, written in a programming language. As the name suggests, **pseudocode is not a real programming code, but it models and may even look like programming code.** It is a generic way of describing an algorithm without using any specific programming language-related notations. Simply put, pseudocode is an outline of a program.

Pseudocode uses some keywords to denote programming processes. Some of them are as follows.

- **Input:** READ, OBTAIN, GET and PROMPT
- **Output:** PRINT, DISPLAY and SHOW
- **Compute:** COMPUTE, CALCULATE and DETERMINE
- **Initialize:** SET and INITIALIZE
- **Add One:** INCREMENT

Pseudocode: To Calculate the Area of A Rectangle

- PROMPT the user to enter the height of the rectangle
- PROMPT the user to enter the width of the rectangle
- COMPUTE the area by multiplying the height with width
- DISPLAY the area
- STOP

Benefits of Pseudocode

- Since it is language independent, it can be used by most programmers. It allows the developer to express the problem logic in plain natural language.
- It is easier to develop a program from a pseudocode rather than from a flowchart or decision table. Programmers do not have to think about syntaxes; they simply have to concentrate on the underlying logic. The focus is on the steps to solve a problem rather than on how to use the computer language.

- The words and phrases used in pseudocode are in line with basic computer operations. This simplifies the translation from the pseudocode to a specific programming language.
- Unlike flowcharts, pseudocode is compact and does not tend to run over many pages. Its simple structure and readability makes it easier to Modify.

Limitations of Pseudocode

- It does not provide a visual representation of the program's logic.
- There are no accepted standards for writing pseudocodes. Programmers use their own style of writing pseudocode.
- It is quite difficult for the beginners to write pseudocode as compared to drawing a flowchart.

PROGRAM CONTROL STRUCTURES

. Program statements that affect the order in which statements are executed, or that affect whether statements are executed, are called *control structures*. They affect the flow of simulation code since a control structure evaluates statements and then executes code according to the result.

there are three control structures.

- **Sequence:** where information flows in a straight line.
- **Selection:** (decision or branched), where the decisions are made according to some predefined condition.
- **Repetition:** (looping), where the logic (sequence of steps) is repeated in a loop until the desired output is obtained.

Unit 1:
Chapter2:
Overview of c

HISTORY OF C

- The root of all modern programming language is ALGO, it is introduced in 1960. ALGO gave the new concept of the structure programming. Subsequently various languages are announced.
- In 1967 Martin Richards developed a language called BCPL(Basic Combined Programming Language) .
- In 1970, Ken Thompson created a language using many features of BCPL and it simply called B language.
- Both BCPL and B was “type less” system programming language.
- C language was developed by Dennis Ritchie at the Bell Laboratories in 1972.
- C language uses many features of the ALGO, BCPL and B Languages. It also added new concepts of the “data type” and many powerful features.
- Unix operating system is totally coded in C language
- In 1983 American National Standard Institute (ANSI) appoint a technical committee to define a standard of C.
- The committee approved a version of c in December 1989 which is now known as ANSI C
- It was approved by the International Standards Organization (ISO) in 1990. This version of the c is also referred as C89

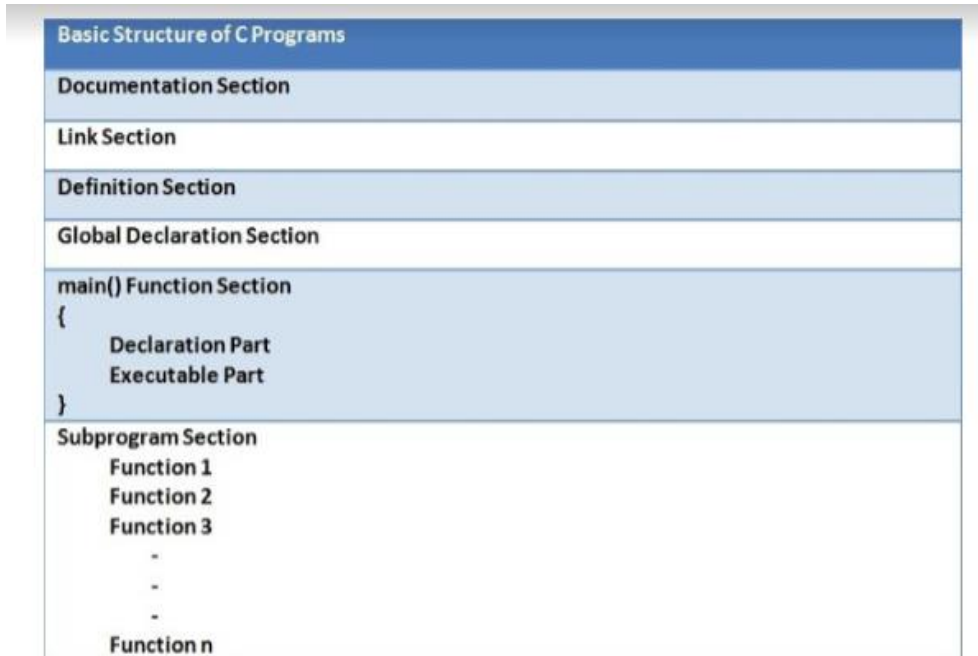
SALIENT FEATURES OF C (IMPORTANCE OF C)

C has many advantages over other high level languages.

- It is robust.
- C has the advantage of assembly level programming such as bit manipulation and all the significant features of high level language such as easy debugging, compactness etc. Therefore most of the C compilers are written in C.
- C is also called as a middle level language since it combines the features of high level as well as low level programming.
- It is highly suited for writing system software and application packages.
- C consists of a rich variety of data types and powerful operators. This makes C programs much more efficient and fast.
- It is platform independent and highly portable i.e., C can be run on almost any operating system.
- C is a structured language, wherein the program is subdivided into a number of modules. Each of these modules performs a specific task. Further structured programming helps in making program debugging, testing and maintenance, easier.
- One of the salient feature of C is its ability to add on to its library. User-defined functions can be added to the C library making the programs simpler.
- C provides manipulation of internal processor registers.

- It allows pointer arithmetic and pointer manipulation.
- Expressions can be represented in compact form using C.

Basic structure of a C program



- The **documentation section** consists of a set of comment lines giving the name of the program, the author and other details, which the programmer would like to use later.
- The **link section** provides instructions to the compiler to link functions from the system library
- The **definition section** defines all symbolic constants
- There are some variables that are used in more than one function, such variables are called global variables and are declared in the **global declaration section**, that is outside of all the function. This section also declares all the user defined functions.
- Every C program must have only **one main()** function. It indicates that the name of the function is main and that the program execution should begin here. The empty parenthesis() that follow main indicate that the function has no arguments. The actual execution of the program begins here.
- This section consists of 2 parts **declaration part and execution part**. All the variables that are used in the program are declared in the declaration part. There must be at least one statement in the execution part.
- These 2 parts are enclosed between curly braces{ }. Program execution begins at the opening brace { and the closing brace } indicates the logical end of the program.
- Every statement in this Section are terminated by a semicolon(;).
- Every C program must have a main() section.
- The sub program section contains all the user defined functions that are called in the **main function**.

Example:

```

/* Program to calculate area of a circle */ Document section
#include<stdio.h>                // Link section
#define PI 3.14159              // Definition section
main()                          // main()function section
{
    int radius;
    float area;                 // DECLARE VARIABLE
    printf("Enter the radius of a circle ");
    scanf("%d",&radius);
    area=PI*radius*radius;      // Executable
    printf("Area of a circle=%.2f",area);
}

```

Rules for writing a C program:

- C program statements must be written in lowercase letters. Only symbolic constants are written in uppercase letters.
- Every statement should end with a semicolon.
- Braces are used to group and mark together the beginning and end of functions.
- Proper indentation of braces and statements makes the program easier to read and debug.
- C is a case sensitive language. C is a free-form language. i.e. More than one statement can be example: written on one line separated by a semicolon.

```

a=b;
x=y+1;
z=a+x;

```

Can be written on one line as

```
a=b;x=y+1;z=a+x;
```

Executing a C program

Executing a program written in C involves series of steps

1. Creating the program
2. Compiling the program
3. Linking the program with functions that are needed from the C library
4. Executing the program

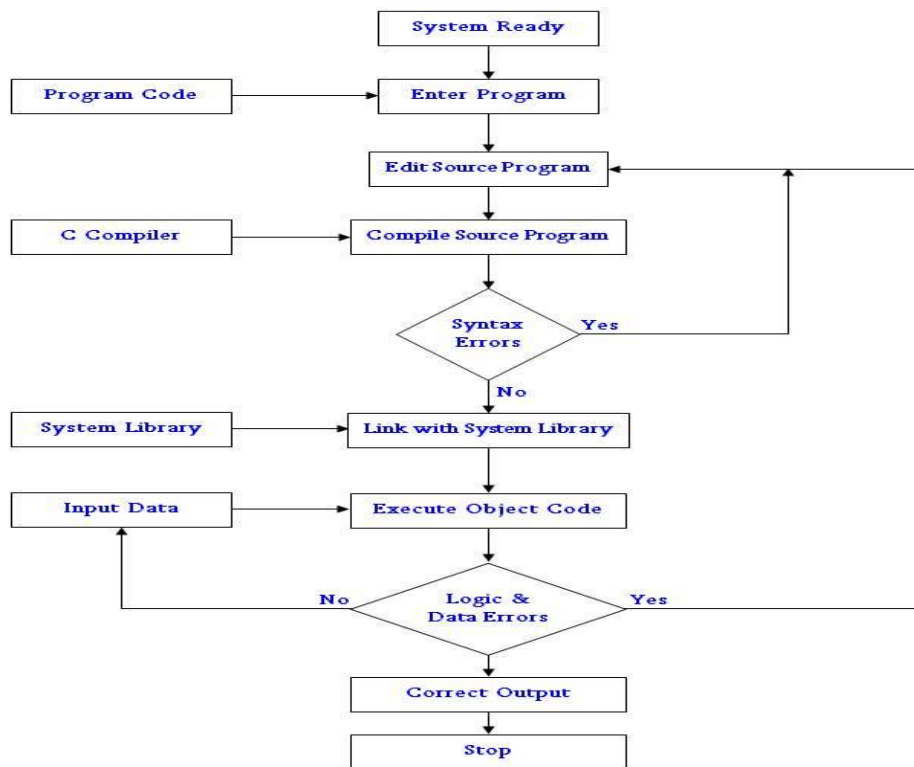


Figure : Process of compiling and running a C program.

The source program translated into suitable execution format. The translation is done after examining each instruction correctness. If everything is all right, the program is translated into **object code**.

Linking is the process of putting together other program files and functions that are required by the program.

Example : library function of system linked to main program. the linking is done automatically.

UNIT 1:
CHAPTER 3
CONSTANTS, VARIABLES AND DATA TYPES

Introduction :-

- The Sequence of the Instruction written to perform specific task is called the **Program**.
- These Instructions are formed using special symbols and words according to some rigid rule is known as **syntax rules**.
- Every Instruction must be written according to the syntax rule of the languages.
- Like any other languages C languages has its vocabulary and grammar. They are follow up for making the program.

In this Chapter we will discuss about various constants, variables and data types are used in C Programming

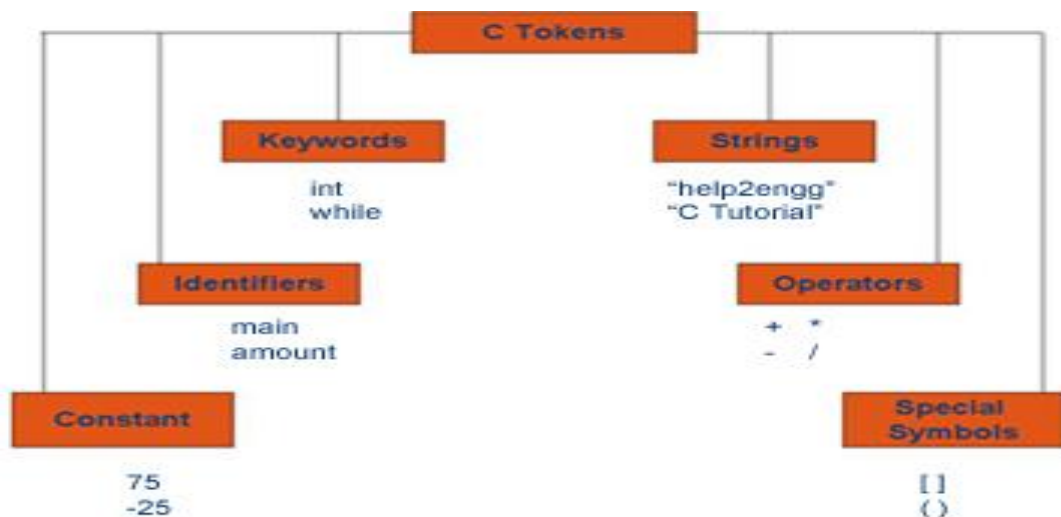
Character set:

The characters that are used to form statements and expressions in a programming language is known as character set. The characters set in C are grouped as follows.

1. Letter- uppercase A-Z and lowercase a-z
2. Digits-0-9
3. Special characters (, > < ? ' " ; :)
4. Whitespaces (blank space, new line, tab)

C TOKENS

- In a passage of text, individual words and punctuation marks are called tokens,
 - In C program the smallest individual units are known as tokens
 - C programs are written using these tokens and the syntax of the language
- . C supports 6 types of tokens. They are
- Keywords
 - Identifiers
 - Constants
 - Strings
 - Special symbols
 - Operators



Keywords and identifiers:

- Every C word is classified either as a keyword or an identifier
- All keywords have a fixed predefined meaning that cannot be changed. Hence keywords are also known as reserve words.
- All keywords must be written in lowercase.
- C supports a set of 32 keywords

Ex: int, float, double, extern, static, auto, continue, if, goto short, long etc. are the keywords

Keywords in C Language

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

Identifiers

Identifiers refer to the names of variables, functions and arrays

These are user defined names and consists of a sequence of letters and digits, with a letter as a first character, both uppercase and lowercase letters are permitted.

Rules for identifiers

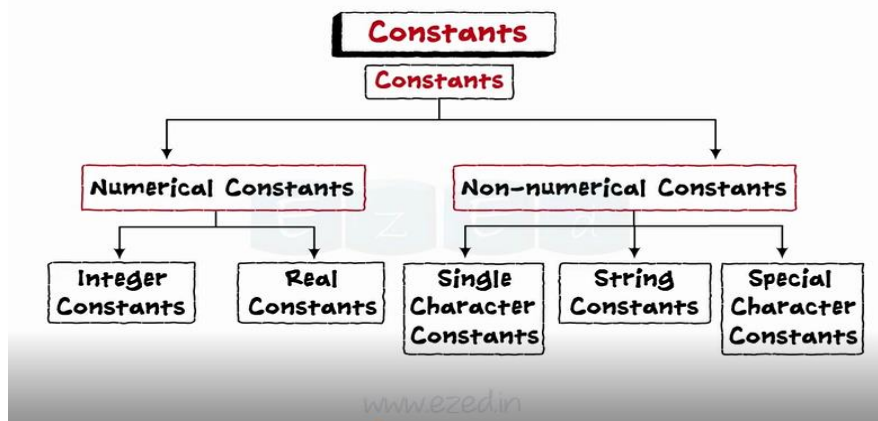
- 1 First character must be an alphabet (or underscore)
- 2 Must consist of only letters, digits or underscore.

- 3 Only first 31 characters are significant
- 4 Cannot use a keyword
- 5 Must not contain white space.

Constants

Constants:

- Constants referred as a fixed value that don't change during the execution of the program.



Integer Constants:

Integer constants refer to a sequence of digits preceded by an optional +/- sign.

E.g. +123, -567,

- Decimal integers consist of a set of digits 0-9.
- An octal integer consists of combination of digits from 0 to 7 with a leading **0**. E.g. 032, 056, 0435.
- A hexadecimal integer consists of 0-9 digits and alphabets A through F to represent numbers from 10-15. A hexadecimal integer is preceded by **0x** or **0X**. E.g. 0xF2, 0X35A, 0x56

Real constants:

- Numbers containing fractional part are called real or floating point constants.
- These can be represented either in decimal notation or exponential notation.
- In decimal notation a whole number is followed by a decimal point and a fractional part. E.g. 0.045, -0.25
- A real number can also be expressed in scientific or exponential notation as mantissa exponent.
- The mantissa could be either a real number expressed in decimal notation or an integer.
- The exponent is an integer with an optional ^/-sign.
- E.g. 0.65e4, 1.25e-5, 56.23E-1

Character constants:

- Character constants are either single character constants or string constants
- Single character constants consist of a single character in single quotes.

E.g. 'a', 'Q', '5'

- String constants are a group of characters enclosed in double quotes. The characters may include letters, numbers or special characters. E.g. "abc", "1956", "Hello!"

Backslash character constants:

- C supports backslash character constants that are used in output function.
- A backslash character constant consists of a backslash followed by a character. These character combinations are also known as escape sequences

Backslash character	Meaning
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\"	Double quote
\'	Single quote
\\	Backslash
\v	Vertical tab
\a	Alert or bell

Variables:

- A variable is a data name that may be used to store a data value.
- A variable may take different values at different times during execution.
- The value of a variable changes during the execution of program.

Rules for naming a variable:

- Variable names should begin with a letter.
- Variable names should not be keywords.
- Blank spaces are not allowed.
- Uppercase and lowercase letters are significant because C is a case sensitive language.
- Length of a variable name should not exceed 8 characters

DECLARATION OF VARIABLE

After designing suitable variable names, we must declare them to the compiler.

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

DATA TYPE VAR1,VAR 2,VAR3;

- Variables are separated by commas
- The declaration statement must end with a semicolon

Assigning values to Variables:

- Values are assigned to variables using the assignment operator '='.

Syntax: var-name=value;

E.g.: x=10;

INITIALIZATION

It is also possible to assign values to a variable during declaration. The process of assigning initial values to variables is known as initialization.

Syntax: data type var-name=value;

E.g.: int x=10;
char x='y';

Declaring variable as a constant:

Sometimes in programming, we would like variable values unchanged during program execution. This can be achieved by declaring the variable with a qualifier *constant* at the time of initialization.

Defining constant

- Using *#define* preprocessor directive
- Using a *const* keyword

Syntax: const data type var-name=value;

EX: const int x=40;

Difference between constant variable and variable

- The variable whose value does not change during the execution of program is known as *constant variable*.
- a variable may take different value at different time.

DEFINING SYMBOLIC CONSTANTS

A symbolic constant is a name, which substitutes either a numeric constant or a character or a string constant. When the program is compiled, each occurrence of a symbolic constant is replaced by its corresponding character sequence. Symbolic constants are usually defined at the beginning of a program.

Symbolic constant can be defined using **#define statement**.

#define symbolic-name value-of constant

Ex:-

- #define PI 3.142
- #define MAX 100
- #define CLASS "IBCA"

Following are the rules used to define the symbolic constant:

- Symbolic name is similar to variable but it is not variable.
- No blank space is allow between '#' and 'define' word.
- '#' must be first in the line.
- A blank space is require between '#define' and 'symbolic name' & between 'symbolic name' and 'constant value'
- It is not statement so it does not end with semicolon
- You cannot put assignment operator '=' between 'symbolic name' and constant value.
- You can put anywhere in program but before it is used in the program.

Statement	Validity	Remark
#define X = 2.5	Invalid	'=' sign is not allowed
# define MAX 10	Invalid	No white space between # and define
#define N 25;	Invalid	No semicolon at the end
#define N 5, M 10	Invalid	A statement can define only one name.
#Define ARRAY 11	Invalid	define should be in lowercase letters
#define PRICE\$ 100	Invalid	\$ symbol is not permitted in name

Data Types:

- C language is rich in its data types. A variety of data types allow the programmer to select the type according to the application.
- The 4 basic data types supported by C are
 1. Fundamental / Primary Data type
 2. User defined Data type
 3. Derived data type
 4. Empty dataset

Primary Data Types:

- All C compilers support this data type. Primary data types can be classified as follows
 1. Integer
 2. Floating point
 3. Character

Data type	Range of Values
Char	-128 to +127
Int	-32,768 to +32,767
Float	3.4e-38 to 3.4e+38
Double	1.7e-308 to 1.7e+308

Integer Data Type:

- Integers are whole numbers with a range of values. Integers occupy one word of storage.
- In a 16-bit machine, the range of integers is -32768 to +32767
- In order to provide some control over a range of numbers and storage space, C

has 3 classes of storage for integers namely, short int, int and long int in both signed and unsigned format

- Short int is used for small range of values and requires half the amount of storage as a regular int number uses.
- Long int requires double the storage as an int value.
- In unsigned integers, all the bits in the storage are used to store the magnitude and are always positive.

short int	int	long int
1 Byte	2 Bytes	4 Bytes

Created by RAJKISHOR

Floating point DataType:

- All floating point numbers require 32 bits with 6 digits of precision
- They are defined in C using the keyword float
- For higher precision, double data type can be used.
- A **double type number uses 64 bits (8 bytes) giving a precision of 14 digits.**
- Range of values for double type is **1.7E-308 to 1.7E+308**
- To extend the precision further, we can use **long double which uses 80 bits.**

float	double	long double
4 Bytes	8 Bytes	10 Bytes

Created by RAJKISHOR

Character DataType:

- A single character can be defined as a character data type using the keyword char
- Characters usually need 8 bits for storage
- The qualifier **signed or unsigned** can be explicitly applied to char.
- While **unsigned characters have values between 0 and 255,**
- **signed characters have values from -128 to 127.**

Unit 1

MCQ QUESTIONS:

1. Select the correct answer for the question.

1) Who invented C Language.?

- A) Charles Babbage
- B) Grahambel
- C) Dennis Ritchie**
- D) Steve Jobs

2. _____ it is a step-by-step representation of a solution to a given problem

- A) algorithm**
- B) flowchart
- C) Pseudocode
- D) None of the above

3. The _____ provides pictorial representation of given problem.

- A. Algorithm
- B. Flowchart**
- C. Pseudocode
- D. All of these

4. Limitation of flowchart _____

- A. Complex
- B. Costly
- C. Difficult to modify
- D. all of the above**

5. Find a correct C Keyword below.

- A) work
- B) case**
- C) constant
- D) permanent

6. Every C program must have only one _____ function.

- A. main()**
- B. User defined
- C. Built in
- D. none of the above

7. Every statement in this Section are terminated by a _____

- A. Dot (.)
- B. colon (:)
- C. semicolon(;)**

D. Question mark (?)

8. The Sequence of the Instruction written to perform specific task is called the_____.

A. Statement

B. Program

C. Algorithm

D. None of the above

9. Types of Integers are.?

A) short

B) int

C) long

D) All the above

10. Size of float, double and long double in Bytes are.?

A) 4, 8, 16

B) 4, 8, 10

C) 2, 4, 6

D) 4, 6, 8

11. Keywords are also known as_____.

A. Keypads

B. reserve words

C. Characters

D. words

12. C supports a set of _____keywords

A. 32

B. 18

C. 34

D.64

13. An Identifier may contain?

A) Letters a-z, A-Z in Basic character set. Unicode alphabet characters other languages

B) Underscore _ symbol

C) Numbers 0 to 9 Unicode Numbers in other languages

D) All the above

14. An Identifier can start with?

A) Alphabet

B) Underscore (_) sign

C) Any character that can be typed on a keyboard

D) Option A & Option B

15. Choose correct statements

- A) **A constant value does not change. A variable value can change according to needs.**
- B) A constant can change its values. A variable can have one constant value only.
- C) There is no restriction on number of values for constants or variables.
- D) Constants and Variables can not be used in a single main function.

16. Find an integer constant.

- A) 3.145
- B) 34**
- C) "125"
- D) None of the above

17. A Variable of a particular type can hold only a constant of the same type. Choose right answer.

- A) TRUE**
- B) FALSE
- C) It depends on the place the variable is declared.
- D) None of the above.

18. Each statement in a C program should end with?

- A) Semicolon ;**
- B) Colon :
- C) Period . (dot symbol)
- D) None of the above.

19. _____ is a procedure or step by step process for solving a problem.

- A. Algorithm**
- B. Flowchart
- C. Pseudocode
- D. All of these

20. The _____ symbol is used to represent decision in flowchart.

- A. Circle
- B. Rectangle
- C. Diamond**
- D. None of these

21. Which of the following is not a valid C variable name?

- a) int number;
- b) float rate;
- c) int variable_count;
- d) int \$main;**

22. A _____ is a data name that may be used to store a data value

- A. Keyword
- B. Constant
- C. Variable**
- D. Data type

23. Symbolic constant can be defined using _____

- A. #define statement.**
- B. #include
- C. #definition
- D. #Declaration

24. A single character can be defined as a character data type using the keyword _____

- A. String
- B. char**
- C. character
- D. int

25. Data types are derived from the fundamental data types are called as _____

- A. Derived data types**
- B. Detailed data types.
- C. Defined data types
- D. None of the above

Descriptive Questions:

- 1) Explain any two phases of Program Development Cycle with diagram
- 2) Define algorithm and give the advantages and disadvantages of algorithm.
- 3) Write an algorithm check for palindrome
- 4) Write algorithm to find largest among three number.
- 5) Define flowchart and explain any 6 symbols using in flowchart.
- 6) List and explain features of C
- 7) Give Structure of C program and Explain each section.
- 8) Write a note on Tokens in 'C'
- 9) Write a note on Constants in 'C'
- 10) Explain in brief classification of a) integer types and b) floating point types
- 11) List and explain primary data types available in 'C'
- 12) Explain with examples declaring, initializing and assigning value to variable
- 13) Explain with example #define statement in 'C'. List rules apply to it.
- 14) Explain the any 4 advantages of flowchart.
15. Explain the Limitations of Flowcharts
16. Draw the flowchart to find factorial of n number