

Chapter-5

PROBLEM SOLVING METHODOLOGY

➤ Introduction

- The term problem solving is used in many disciplines, sometimes with different perspectives and often with different terminologies.
- The problem-solving process starts with the problem specification and end with a correct program.
- The steps to follow in the problem-solving process are:
 - ◆ Problem definition
 - ◆ Problem Analysis
 - ◆ Algorithm development
 - ◆ Coding
 - ◆ Testing & Debugging
 - ◆ Documentation & Maintenance
- The stages of analysis, design, programming, implementation and maintenance form the life cycle of the system.

➤ Problem definition:

- This step defines the problem thoroughly. Here requirements are specified. This step includes understanding the problem very well. The problem solver must understand problem very well to solve problem efficiently.

➤ Problem Analysis:

- Analyzing the problem or analysis involves identifying the following:
 - ◆ Inputs, i.e. the data you have to work with.
 - ◆ Outputs i.e. the desired results.
 - ◆ Any additional requirements on the solutions.

➤ ALGORITHM

- **An Algorithm is a step-by-step procedure to solve a given problem.**
- The word algorithm originates from the word 'algorism' which means process of doing arithmetic with Arabic numerals.
 - In 9th-century Arab Mathematician, **Mohammed Al-Khowarizmi**, who developed methods for solving problems which is, used specific step-by-step instructions.
- ✓ **Characteristics of algorithm:**

- A well defined algorithm has the five basic characteristics; as follows
 1. **Input:** Algorithm starts with procedural steps to accept input data. The algorithm must accept one or more data to be processed.
 2. **Definite:** Each operational step or operation must be definite i.e. each and every instruction must clearly specify that what should be done.
 3. **Effective:** Each operational step can at least in principle be carried out by a person using a paper and pencil in a minimum number of times.
 4. **Terminate:** After some minimum number of operations, the algorithm must come to an end.
 5. **Output:** An algorithm is written to solve the problem, therefore it must produce one or more computed results or answers called output.

Example: An algorithm to find the area of a rectangle can be expressed as follows:

- Given the length l and the breadth b , this algorithm finds the area of rectangle rec .

```

Step 1:      START
Step 2:      [Read the values of l, b]
              INPUT l, b
Step 3:      [Calculate area of rectangle]
              rec = l * b
Step 4:      [Print the area of rectangle]
              OUTPUT rec
Step 5:      [End of Algorithm]
              STOP
  
```

In the above example, we used $=$ that represents assignment.

1. Design an algorithm to find the average of four numbers

```

Step 1:      START
Step 2:      INPUT A, B, C, D
Step 3:      [Calculate]   AVG = (A+B+C+D)/4
Step 4:      OUTPUT AVG
Step 5:      STOP
  
```

2. Design an algorithm to calculate the Simple Interest, given the Principal (P), and Rate (R) and Time (T)

```

Step 1:      START
Step 2:      INPUT P, T, R
Step 3:      [Calculate]   SI = (P*T*R)/100
Step 4:      OUTPUT SI
Step 5:      STOP
  
```

3. Design an algorithm to find the greatest of three number (A, B, C)

Step 1: START
Step 2: INPUT A, B, C
Step 3: [Assign A to large]
 Large = A
Step 4: [Compare large and B]
 If(B > large)
 Large = B
 Endif
Step 5: [Compare large and C]
 If(C > large)
 Large = C
 Endif
Step 6: [Print the largest number]
 OUTPUT large
Step 7: STOP

4. Design an algorithm to find factorial of a number (N)

Step 1: START
Step 2: INPUT N
Step 3: [Initialize factorial to 1]
 Fact = 1
Step 4: [compute the factorial by successive multiplication]
 Repeat for I = 1 to N
 Fact = Fact * I
 [End of Step 4 for loop]
Step 5: [Print factorial of given number]
 OUTPUT Fact
Step 6: STOP

5. Design an algorithm to find Fibonacci series (N)

Step 1: START
Step 2: INPUT N
Step 3: [Initialize the variables]
 First = 0
 Second = 1
 Term = 2
Step 4: [Print the values of first and second]
 PRINT First, Second
Step 5: Third = First + Second

Step 6: Repeat while (term <= N)
 PRINT Third
 First = Second
 Second = Third
 Third = First + Second
 Term = Term + 1
 [End of While loop]

Step 7: STOP

6. Design an algorithm to find the GCD of two numbers (A, B)

Step 1: START

Step 2: INPUT A, B

Step 3: Repeat while (B != 0)
 Rem = A % B
 A = B
 B = Rem
 [End of While loop]

Step 4: [Print the last divisor]
 PRINT A

Ste 5: STOP

✓ **Advantage of Algorithm**

1. It is a step-by-step representation of a solution to a given problem, which is very easy to understand.
2. It has got a definite procedure, which can be executed within a set period of time.
3. It is independent of programming language.
4. It is easy to debug as every step has got its own logical sequence.

✓ **Disadvantage of Algorithm**

- It is time-consuming
- An algorithm is developed first which is converted into a flowchart and then into a computer program.

✓ **Analysis of Algorithm**

- There may be more than one approach to solve a problem. The choice of a particular algorithm depends on the following performance analysis and measurements.
 - **Space complexity:** The amount of memory needed by the algorithm to complete its run.
 - **Time Complexity:** The amount of time, the algorithm needed to complete its run.
- When we analyze an algorithm depends on input data, there are three cases
 - Best Case
 - Average Case

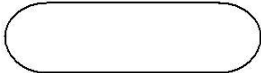
- Worst Case


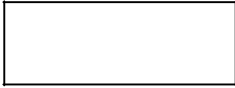

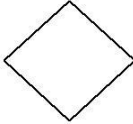
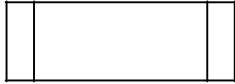
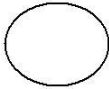
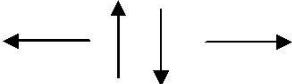
➤ FLOWCHART

- **A Flowchart is a pictorial or graphical representation of an algorithm.**
 - Flowchart plays an important role in the programming of a problem and helpful in understanding the logic of program.
 - Once the flow chart is drawn, it becomes easy to write program in any high level language.
 - Flowcharts are classified into two categories:
 1. Program Flowcharts
 2. System Flowcharts
 - **Program flowcharts** present a diagrammatic representation of a sequence of instructions for solving a program.
 - **System flowcharts** indicate the flow of data throughout a data processing system, as well as the flow into and out of the system. Such flowcharts are widely used by designers, to explain a data processing system.
- ✓ **Importance of Flowchart**
1. **Communication:** Flowcharts are better way of communication of the logic of a program.
 2. **Effective Analysis:** With the help of flowchart, problem can be analyzed in more effective way.
 3. **Proper documentation:** Program flowcharts serve as a good program documentation, which is needed for various programs.
 4. **Efficient coding:** The flowchart acts as guide or blueprint during the system analysis and program development phase.
 5. **Proper Debugging:** The flow chart helps in debugging process.
 6. **Efficient program maintenance:** The maintenance of a program become easy with the help of flowcharts.

✓ **Symbols Used In Flowcharts**

✓ **Symbols Used In Flowcharts**

SYMBOLS	PURPOSE
	TERMINAL (START or STOP) The beginning, end, or point of interruption in a program

	INPUT OR OUTPUT Input or Output data or information
	PROCESSING An instruction or group of instructions which changes the program
	PREPARATION [Looping] An instruction or group of instructions which changes the program
	DECISION or BRANCHING Represents a comparison, a question or a decision that determinates alternative paths to be followed
	PREDEFINED PROCESS A group of operation not detailed in the particular set of flowcharts
	CONNECTOR An entry form, or an exit to the another part of the program flowchart
	FLOW DIRECTION The direction of processing or data flow.

Example: Design a flow chart and an algorithm to find the area of a square.

Step 1: START

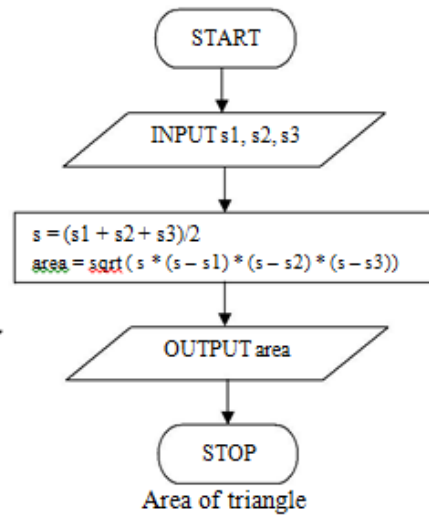
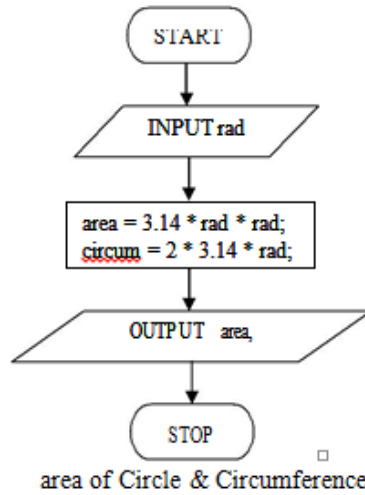
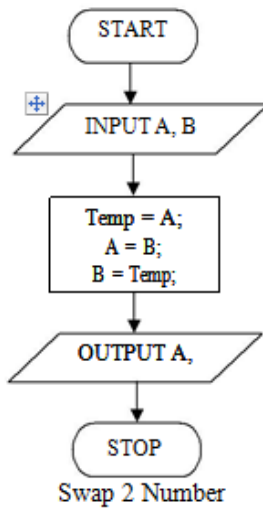
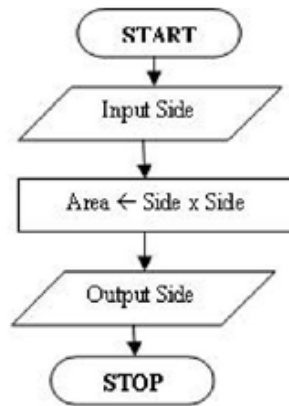
Step 2: INPUT Side

Step 3: [Calculate Area]

$$\text{Area} = \text{Side} * \text{Side}$$

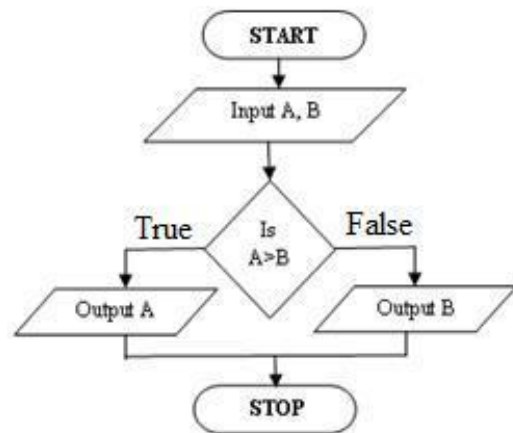
Step 4: OUTPUT Area

Step 5: STOP



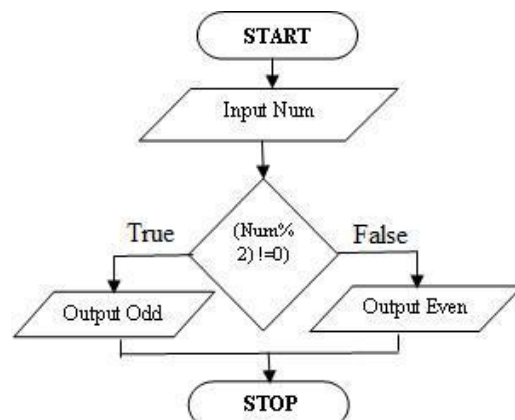
1. Write a program, design a flow chart and an algorithm to find the larger of two numbers.

- Step 1: Start
 Step 2: Input A and B
 Step 3: If(A>B) then
 Output A
 Else
 Output B
 [End if]
 Step 4: Stop



2. Write a program, design a flow chart and an algorithm to find given number is odd or even.

- Step 1: Start
 Step 2: Input Num
 Step 3: If((Num%2)!=0) then
 Output Odd
 Else
 Output Even
 [End if]
 Step 4: Stop



◆ **Advantage of Flowcharts**

1. Flowcharts provide an excellent means of communication, which is very easy to understand.
2. It has got a definite procedure, which shows all the major parts of a program, It is easy to convert it into a program.
3. It is independent of programming language.
4. It is easy to debug as every step has got its own logical sequence.

◆ **Disadvantages of Flowcharts**

1. It is time-consuming and it requires the uses of a number of symbols which are to be properly represented.
2. The represented of complex logic is difficult in a flowchart.
3. Alterations and modifications can be only made by redrawing the flowcharts.

➤ **Pseudo code:**

- This is an abstract representation of program in English statement.
- In pseudo code English words & phrases are used to represent operations.
- **Advantages:** Easy to read, understand & modify.

➤ **Coding or Programming**

- **The process of writing program instructions for an analyzed problem in a programming language.**
- It is the process of translating the algorithm or flowchart into the syntax of given purpose language.
- You must convert each step of the algorithm into one or more statements in a programming language such as C, C++, and Java etc.

➤ **Testing and Debugging**

- **Testing is the process of checking whether the program works according to the requirement of the user.**
- **Debugging is the process of identifying and correcting or removing the Bugs (errors).**
- There are four types of errors. They are
 - ◆ Syntax errors
 - ◆ Run-time errors
 - ◆ Semantic errors
 - ◆ Logic errors (bugs)

✓ **Syntax Error**

- **Syntax is the set of rules which should followed while creating the statements of the program.**

- The grammatical mistakes in the statements of the program are called syntax errors.
- Example:

```
void main( )
{
    int a, b;
    cout<< 'Enter the numbers" ;
    cin>> a >> b;
    cout<< a + b
}
```

- In the example program, the fourth statement produces an syntax error as the missing semicolon.

✓ **Run-time Error**

- **During execution of the program, some errors may occur. Such errors are called run-time errors.**
- Example: Divide by zero.

✓ **Semantic Error**

- **An error, which occurs due to improper use of statements in programming language.**
- Consider an expression $C = A + B$, indicating the values of the variable A and B are added and assigned to variable C.
- If we written $A + B = C$, through the values of A and B are added, it cannot be assigned to variable C written to the right of = Sign.
- This is semantic error.

✓ **Logical Error**

- **Logical errors occur when there are mistakes in the logic of the program.**
- Unlike other errors logical errors are not displayed while compiling because the compiler does not understand the logic of the program.
- Example: To find the area of the circle, the formula to be used is $area = 3.14 * r * r$. But if we written $area = 3.14 * 2 * r$, then the required output is not obtained even though the program is successfully executed.

➤ **Documentation and Maintenance**

- **Documentation is a reference material which explains the use and maintenance of the program application for which it has been written.**
- There are two types of documentation.
 - Internal Documentation
 - External Documentation.

✓ **Internal Documentation:**

- This is also known as technical documentation.

- It is meant for the programmer who may update the program code at later stages.
- It is done by:
 - Defining meaningful variable names.
 - Including comments in program code.
 - Presenting the program code clearly.

✓ **External Documentation:**

- The program or application is supported with additional textual information about the application.
- It is useful for the user, administrator or developer.

➤ **Maintenance:**

- **Program maintenance means periodic review of the programs and modifications based on user's requirements.**
- Maintenance is a continuous task
- Documentation plays an important role in program maintenance. It helps speedy and efficient maintenance.

➤ **Programming Constructs**

- **A programming constructs is a statement in a program.**
- There are 3 basic programming constructs.
 - Sequential Constructs
 - Selection Constructs
 - Iteration Constructs

✓ **Sequential Constructs:**

- The program statements are executed one after another, in a sequence.
- Sequential constructs are:
 - Input Statement
 - Assignment Statement
 - Output Statement

❖ **Input Statement**

- This statement is used to input values into the variables from the input device.
- Example: INPUT A, B, C

❖ **Assignment Statement**

- This statement is used to store a value in a variable.
- In many languages '=' is used as the assignment operator.
- Example: A = 10;
 B = 5;
 C = A + B;

❖ Output Statement

- This statement is used to display the values of variables on the standard output device.
- Example: OUTPUT C;

✓ Selection construct

- It is also known as conditional construct.
- This structure helps the programmer to take appropriate decision.
- There are five kinds of selection constructs, viz.
 - Simple – if
 - if – else
 - if – else – if
 - Nested – if
 - Multiple Selection

❖ Simple - if :

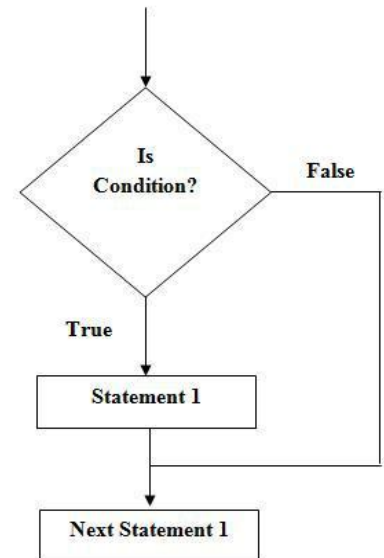
- This structure helps to decide the execution of a particular statement based on a condition.
- This statement is also called as **one-way branch**.
- The general form of simple – if statement is:

```
if (Test Condition)    // This Condition is true
    Statement 1;
    Statement 2;
```

- Here, the test condition is tested which results in either a TRUE or FALSE value. If the result of the test condition is TRUE then the Statement 1 is executed. Otherwise, Statement 2 is

executed. **Ex:** if(amount >= 5000)

```
    discount = amount * (10/100);
    net-amount = amount – discount;
```

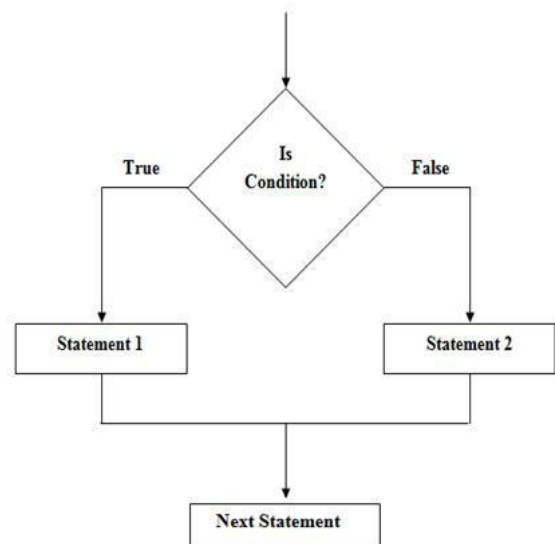


❖ if – else statement :

- This structure helps to decide whether a set of statements should be executed or another set of statements should be executed.
- This statement is also called as **two-way branch**.
- The general form of if – else statement is:

```
if (Test Condition)
    Statement 1;
else
    Statement 2;
```

- Here, the test condition is tested. If the test-condition is TRUE, statement-1 is executed. Otherwise Statement 2 is executed.



```

Ex:  if( amount >= 5000 )
        discount = amount * (10/100);
    else
        discount = amount * (5/100);

```

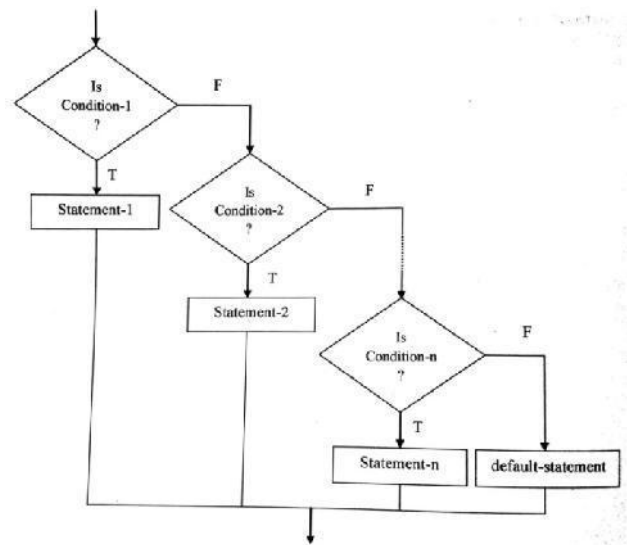
❖ **if – else - if statement :**

- This structure helps the programmer to decide the execution of a statement from multiple statements based on a condition.
- There will be more than one condition to test.
- This statement is also called as **multiple-way branch**.
- The general form of if – else – if statement is:

```

if (Test Condition 1)
    Statement 1;
else
    if (Test Condition 2)
        Statement 2;
    else
        .....
    else
        if( test Condition N)
            Statement N;
        else
            Default Statement

```



- Here, Condition 1 is tested. If it is TRUE, Statement 1 is executed control transferred out of the structure. Otherwise, Condition 2 is tested. If it is TRUE, Statement 2 is executed control is transferred out of the structure and so on.
- If none of the condition is satisfied, a statement called default statement is executed.
- **Example:**

```

if( marks >= 85 )
    PRINT "Distinction"
else
    if( marks >= 60 )
        PRINT "First Class"
    else
        if( marks >= 50 )
            PRINT "Second Class"
        else
            if( marks >= 35 )
                PRINT "Pass"
            else
                PRINT "Fail"

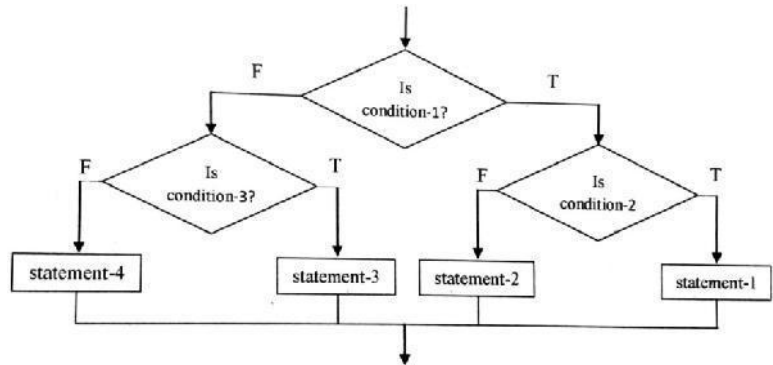
```

❖ **Nested if statement :**

- The statement within the if statement is another if statement is called Nested – if statement.
- The general form of Nested – if statement is:

```

if (Test Condition 1)
    if (Test Condition 2)
        Statement 1;
    else
        Statement 2;
else
    if (Test Condition 3)
        Statement 3;
    else
        Statement 4;
    
```



Ex: To find the greatest of three numbers a, b and c.

```

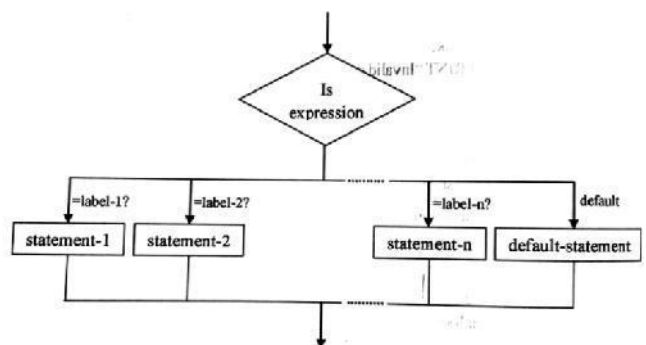
if ( a > b )
    if ( a > c )
        OUTPUT a
    else
        OUTPUT c
else
    if ( b > c )
        OUTPUT b
    else
        OUTPUT c
    
```

❖ **Multiple Selection constructs or Switch statement :**

- If there are more than two alternatives to be selected, multiple selection construct is used.
- The general form of Switch statement is:

```

Switch ( Expression )
{
    Case Label-1:    Statement 1;
                   Break;
    Case Label-2:    Statement 1;
                   Break;
    .....
    Case Label-N:    Statement N;
                   Break;
    Default      :   Default- Statement;
}
    
```



- **Ex:** To find the name of the day given the day number

```

Switch ( dayno )
{
    Case 1:        PRINT "Sunday";
                  Break;
    
```

```

Case 2:    PRINT "Monday";
           Break;
Case 3:    PRINT "Tuesday";
           Break;
Case 4:    PRINT "Wednesday";
           Break;
Case 5:    PRINT "Thursday";
           Break;
Case 6:    PRINT "Friday";
           Break;
Case 7:    PRINT "Saturday";
           Break;
default:  PRINT "Invalid Day Number";
}

```

✓ **Iterative Constructs or Looping**

- **The process of repeated execution of a sequence of statements until some condition is satisfied is called as iteration or repetition or loop.**
- Iterative statements are also called as repetitive statement or looping statements.
- There are two iterative constructs, viz.
 - Conditional Looping
 - Unconditional Looping

❖ **Conditional Looping :**

- This statement executes a group of instructions repeatedly until some logical condition is satisfied.
- The number of repetitions will not be known in advance.
- The two conditional looping constructs are:
 - **While**
 - **do while**

❖ **Unconditional Looping :**

- This statement executes a group of instructions is repeated for specified number of times.
- The unconditional looping constructs is **for** statement.

✓ **While Constructs:**

- This is a **pre-tested loop** structure.
- This structure checks the condition at the beginning of the structure.
- The set of statements are executed again and again until the condition is true.
- When the condition becomes false, control is transferred out of the structure.
- The general form of while structure is

```

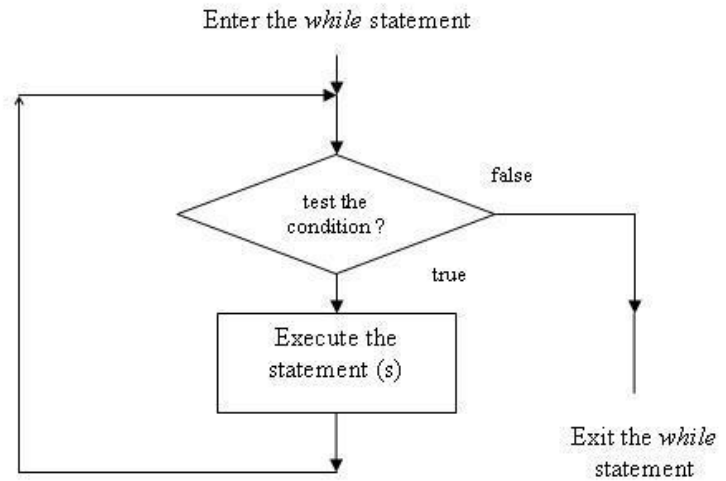
While ( Test Condition)
    Statement 1
    Statement 2

```

.....
 Statement N
 End of While

• Example:

```
i = 1;
While ( i <= 5)
    PRINT i;
    i = i + 1;
end of while
Output: 1 2 3 4 5
```



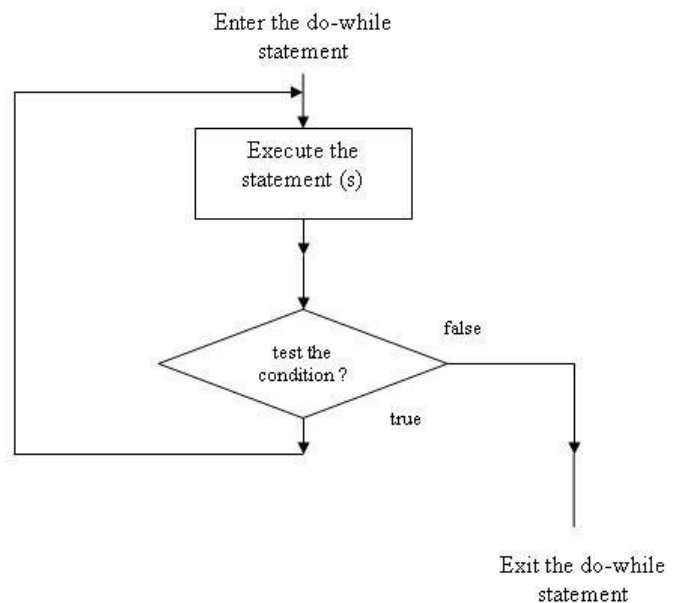
✓ **do while Constructs:**

- This is a **post-tested loop** structure.
- This structure checks the condition at the end of the structure.
- The set of statements are executed again and again until the condition is true.
- When the condition becomes false, control is transferred out of the structure.
- The general form of while structure is

```
do
    Statement 1
    Statement 2
    .....
    Statement N
while ( Test Condition)
End of While
```

• Example:

```
sum = 1;
i = 1;
do
    sum = sum + i;
    i = i + 1;
while ( i <= 100);
```



✓ **for Constructs:**

- This structure is the **fixed execution structure**.
- This structure is usually used when we know in advance exactly how many times a set of statements is to be repeatedly executed again and again.
- This structure can be used as increment looping or decrement looping structure.
- The general form of for structure is as follows:

```
for ( Expression 1; Expression 2; Expression 3)
{
    Statement 1;
```

```

Statement 2;
Statement N;
}

```

Where, Expression 1 represents Initialization

Expression 2 represents Condition Expression

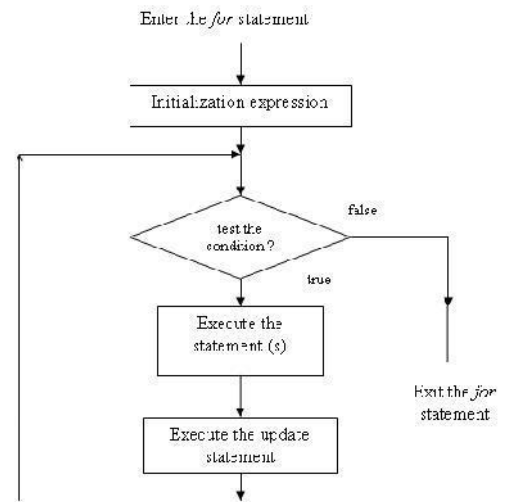
3 represents Increment/Decrement

• Example:

```

sum = 0;
for ( i=1; i<=10; i++)
    sum = sum + i;

```



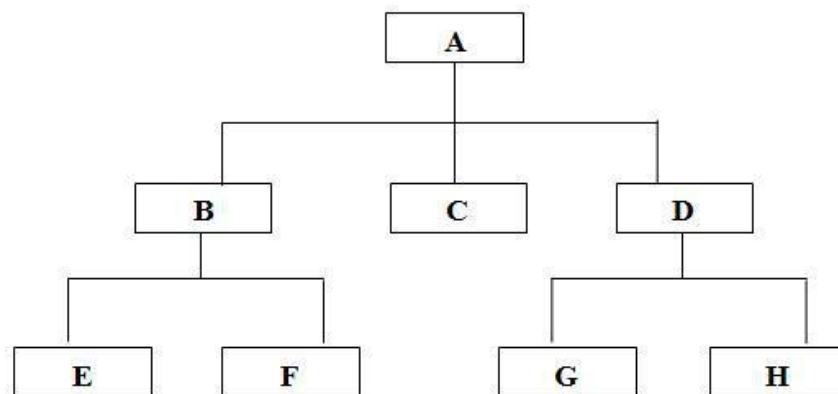
➤ **Characteristics of a good program:**

- The best program to solve a given problem is one that requires less space in memory, takes less execution time, easy to modify and portable.
- **Modification:** A good program is the one which allows any modifications easily whenever needed.
- **Portability:** A good program is the one which can be run on different type of machine with a minimum or no change.

➤ **Approaches to problem solving:**

1. **Top-down design:**

- Top-down design involves dividing a problem into sub-problems and further dividing the sub-problems into smaller sub-problems until it leads to sub-problems that can be implemented as program statements.



- Where A is the main problem and remaining are the sub-problems.
- The top-down approach is taken for program design; the programs can be developed easily, quickly, committing a minimum of errors.

2. **Stepwise refinement:**

- The process of breaking down the problem at each stage to obtain a computer solution is called *stepwise refinement*.

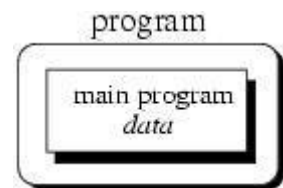
3. Bottom-up design:

- A design method, in which system details are developed first, followed by major process.
- This approach is the reverse of top-down design.
- The process starts with identification of set of modules which are either available or to be constructed.
- An attempt is made to combine the lower level modules to form modules of high level.
- Examples include object oriented programming using C++.

4. Programming techniques:

i. Unstructured programming:

- During learning stage by writing small and simple programs without planning leads to unstructured programming.



ii. Procedural programming :

- This method allows us to combine the returning sequences of statements into one single place.
- A procedure call is used to invoke the procedure. After the sequence is processed, flow of control proceeds right after the position where the call was made.
- Procedures (sub procedures) programs can now be written as more structured and error free.

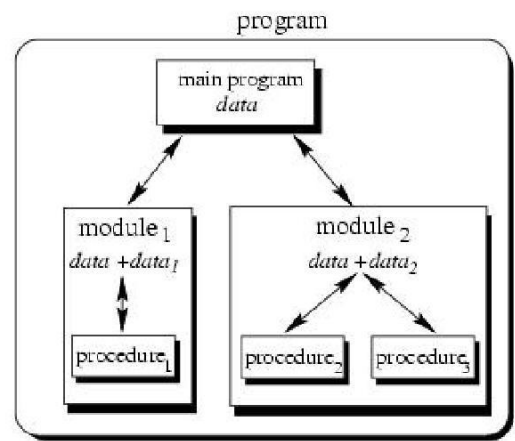
iii. Structured programming :

- Structured programming is method of programming by using the following type of code structures to write program:
 - Sequence (input, output, assignment)
 - Selection (if, if-else etc.)
 - Iteration (while, do-while, for)
 - Subroutines (functions)

iv. Modular programming:

- The process of splitting the lengthier and complex programs into number of smaller units (modules) is called modularization and programming with such an approach is called *modular programming* .

- This technique provides grouping of procedures which are common functionality into separate modules.
- Advantages of modular programming:
 - Reusability
 - Debugging is easier
 - Building library
 - Portability



CHAPTER 5 PROBLEM SOLVING METHODOLOGY

Review questions

Short answer questions:

1. What are the steps involved in Problem solving?

The steps involved in problem solving are

- Problem definition
- Problem analysis
- Design
- Testing and debugging
- Documentation.

2. What is structured programming?

Structured programming is a method of programming by using the basic programming constructs like sequence, select and iteration.

3. Explain the sequential construct.

Ans: The ability of executing the program statement one after another in Sequence is called sequential construct.

4. What are the tools used in the design of problems?

Ans: Algorithm and flowchart.

5. What is a flowchart?

Ans: A flowchart is a pictorial representation of solution to any problem.

6. What is algorithm?

Ans: An algorithm is a step by step procedure to solve a given problem.

7. Mention the rules for drawing a flowchart.

Ans:

- Understand the problem statement clearly before developing a flowchart.
- Study the outputs to be generated and required input to solve the problem.
- Design the process in such a way that it produces the desired result.
- Test the Preparation by giving test data.
- Verify the result for correctness. Make suitable changes, if any change is required and repeat the process.

8. Give the advantage of flowchart.

- Ans: **Communication:** Flowcharts are better way of communicating the logic of a system to all concerned or involved.
- **Effective analysis:** With the help of flowchart, problem can be analyzed in more effective way therefore reducing cost and wastage of time.

- **Proper documentation:** Program flowcharts acts as a good program documentation, which is needed for various purposes.
- **Effective coding:** The flowchart acts as a guide or blueprint during the system analysis and program development phase.
- **Proper debugging:** The flowchart helps in debugging process.
- **Efficient program maintenance:** The maintenance of a program becomes easy with the help of flowchart. It helps the program to put efforts more efficiently on a specific part.

9. What is testing?

Testing means running the program and executing instructions, checking logic by entering sample data and check output.

10. Define debugging.(E.Q.33)

Ans: Debugging is the process of removing errors.

11. What is syntax error?

Ans: The grammatical mistakes in the statements of the program are called syntax error.

12. What is run time error?

Ans: Errors detected during execution of the program is called run time error.

13. What is logical error?

Ans: If the correct translation of algorithm causes the program to produce wrong results, the error is called logical error.

14. What is Top down analysis?

Top down analysis involves dividing a problem into sub problems and further dividing into smaller problems until it can be implemented independently.

15. What is Bottom up approach?

It is the opposite of top down approach, where small sub problems are combined together to form a single large solution.

16. Define step wise refinement.

The process of breaking down the problem at each stage to obtain a computer solution is called Stepwise refinement.

17. Define coding.

Coding is the process of translating the algorithm or flowchart into the syntax of programming language.

18. Give the advantages of structured programming?

It helps in clear understanding of the problem and its solution

19. What is Multiple selection?

Ans: Execution of statements from multiple statements is called multiple selection.

20. What are the types of iteration construct?

Ans: Conditional looping and unconditional looping.

21. What is unconditional looping?

Ans: This statement executes a group of instructions and is repeated for specified number of items.

22. What are the types of selection construct?

Ans:

- ❖ Simple-if statement
- ❖ if-else statement
- ❖ Nested-if statement
- ❖ if-else-if statement or else-if-ladder
- ❖ Switch statement

23. Explain single entry and single-exit concept structured programming.

Ans: single entry and single exit means the program follows a sequence construct in executing a problem. The program starts at the entry point and exits at the end .

Long answer questions:

24. Explain the concept of structured programming.(S.Q.2)

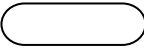
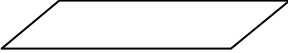
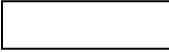
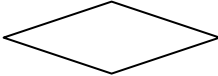
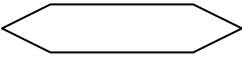


Ans: Structured programming is a method of programming by using sequence of sequentially executed statements, conditional execution of statement (i.e., if statement), looping or iteration (i.e., while, do-while, for).

25. What are the advantages of structured programming?(S.Q.18)

Ans:

- Structured programs are easy to write as the programming logic is well organized.
- Structured programs are easy to test and debug as at any instant we are looking at a smaller unit or module.
- Structured programs are easy to maintain due to their start-to-finish book like readability.
- Structured programs can be functionally decomposed into logical working units.

26. Write the various symbols used in a flowchart.

Names	Symbols	Meaning or when it is used.
Oval		Beginning or end (Start / stop)
Parallelogram		Input or output
Rectangle		Executable statements or process or calculation or assignment.
Rhombus		Decision making or branching.
Preparation		Only for “for loop”
Connector		Connection
Arrows	↑ ← ↓ →	Direction
Pre-defined process		Sub program

27. Explain the various types of errors detected during testing?

Ans:

- ❖ **Syntax error:** The grammatical mistakes in the statements of the program are called syntax errors.
- ❖ **Run-time error:** Errors that are detected during the execution of the program is called run-time error.
- ❖ **Logical error:** If the correct translation of the algorithm causes the program to produce wrong results, the errors are called as logical error.

28. What are the features of algorithm?

Ans: The features of algorithm are:

- **Input:** Algorithm must accept 1 or more data.
- **Definite:** Each step must be definite and each step must be clearly specified what should be done.
- **Effective:** The steps written must provide a proper solution or effective solution.
- **Terminate:** After some number of operation the algorithm must come to an end.
- **Output:** The algorithm must produce 1 or more output or result.

29. Explain conditional looping.

Ans: This statement executes a group of instructions repeatedly until some

logical condition is satisfied. The number of repetitions will not be known in advance is known as conditional looping.

Essay type questions:

30. What are the types of for loops?

Ans: Traditional for loop
Iterator based for loop
Vectorised for loop
Compound for loop

31. Briefly explain the various stages of problem solving. (S.Q.1, 9, 17, E.Q.33, 34)

Ans:

- **Analysis:** Analysis involves identifying the following i.e., input, output, and any additional requirements or constraints on the solution.
- **Design:** To represent the solution of the problem we use tools such as algorithm and flowcharts.
- **Coding:** Coding is the process of translating the algorithm or flowchart into the syntax of a given programming language.
- **Testing:** Testing means running the program, executing all the instructions/ functions and checking the logic by entering sample data to check the output.
- **Maintenance:** Program maintenance means periodic review of the programs and modifications based on user's requirements.

32. Explain the problem definition phase.

Ans: In the problem definition, the problem is defined by solving the following questions like what program does?, what are its tasks?, what kind of data is required ?, what will be the output? how to interact with the user and the system. This stage focuses on the root of the problem and try to understand the problem clearly.

33. Write a note on testing and debugging?

Testing means running the program, executing all its instructions and checking the logic by entering sample data and verify the output.

Debugging is the process of finding and correcting program code mistakes that's called errors.

During this stage the program errors like syntax , run time and logic errors are found , which has to be removed and the program has to be run again , this process continues / until the program has all the errors fixed and is ready with the output.

34. Briefly explain documentation and maintenance.

Documentation is a reference material which explains the use and maintenance of the program , it can be in the separate form of printed document or as a help file in the program itself. There are two types of documentation:

- a. Internal Documentation: this is known as technical documentation which is meant for the programmer to update the code.
- b. External Documentation: this is the additional information about the application which is useful for the user.

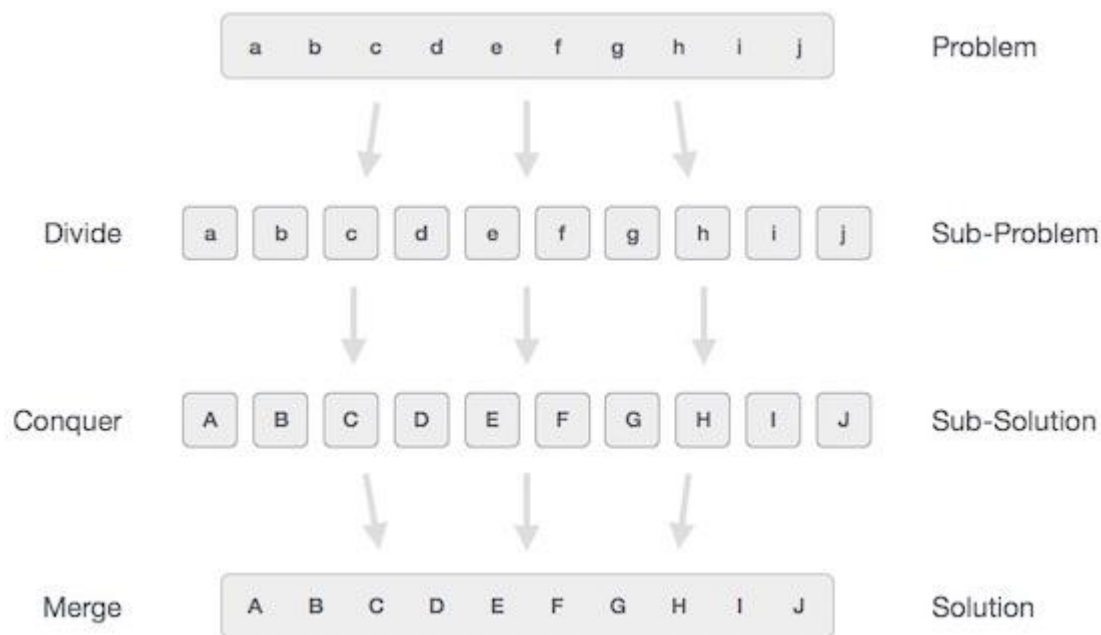
Maintenance means periodic review of the programs and modification based on users requirements. It is a continuous task , where documentation plays an important role by helping in speedy and efficient maintenance.

35. Explain modular design.

Ans: The larger programs are divided into a number of smaller logical components, each of which serves a specific task. The process of splitting the lengthier and complex programs into no of smaller units called modules is called modularization. And programming with such an approach is called modular programming.

36. Explain Divide and conquer method.

Ans: In divide and conquer approach, the problem in hand, is divided into smaller sub-problems and then each problem is solved independently. When we keep on dividing the subproblems into even smaller sub-problems, we may eventually reach a stage where no more division is possible. Those "atomic" smallest possible sub-problem (fractions) are solved. The solution of all sub-problems is finally merged in order to obtain the solution of an original problem.



Broadly, we can understand **divide-and-conquer** approach in a three-step process.

Divide/Break

This step involves breaking the problem into smaller sub-problems. Sub-problems should represent a part of the original problem. This step generally takes a recursive approach to divide the problem until no sub-problem is further divisible. At this stage, sub-problems become atomic in nature but still represent some part of the actual problem.

Conquer/Solve

This step receives a lot of smaller sub-problems to be solved. Generally, at this level, the problems are considered 'solved' on their own.

Merge/Combine

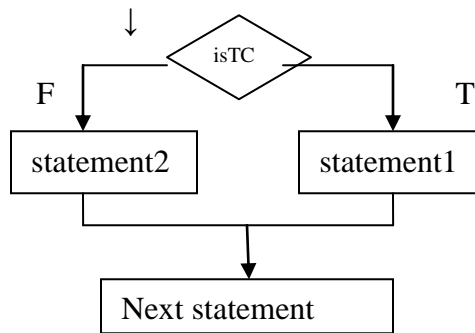
When the smaller sub-problems are solved, this stage recursively combines them until they formulate a solution of the original problem. This algorithmic approach works recursively and conquer& merge steps works so close that they appear as one.

38. Give syntax and flowchart of if-else and else-if construct with an example. (Q. 37)

Ans: **if-else construct:** It is also known as two-way branching. It is used when there are alternative statements needed to be executed based on the condition.

Syntax:Flowchart:

```
if(test_condition)
{
    True statement;
}
else
{
    False statement;
}
```



EXAMPLE:

To check whether a given year is a leap year or not

```
#include<iostream.h>
#include<iomanip.h>
#include<conio.h>
void main()
{
    int year;
    clrscr();
    cout<<"Enter the year:";
    cin>>year;
    if( year%4==0 && year%100!=0 || year%400 ==0)
    cout<<"It is a leap year"<<endl;
    else
    cout<<"It is not leap year"<<endl;
    getch();
}
```

else-if construct: It will verify the range of values and a choice is made between different possible alternatives.

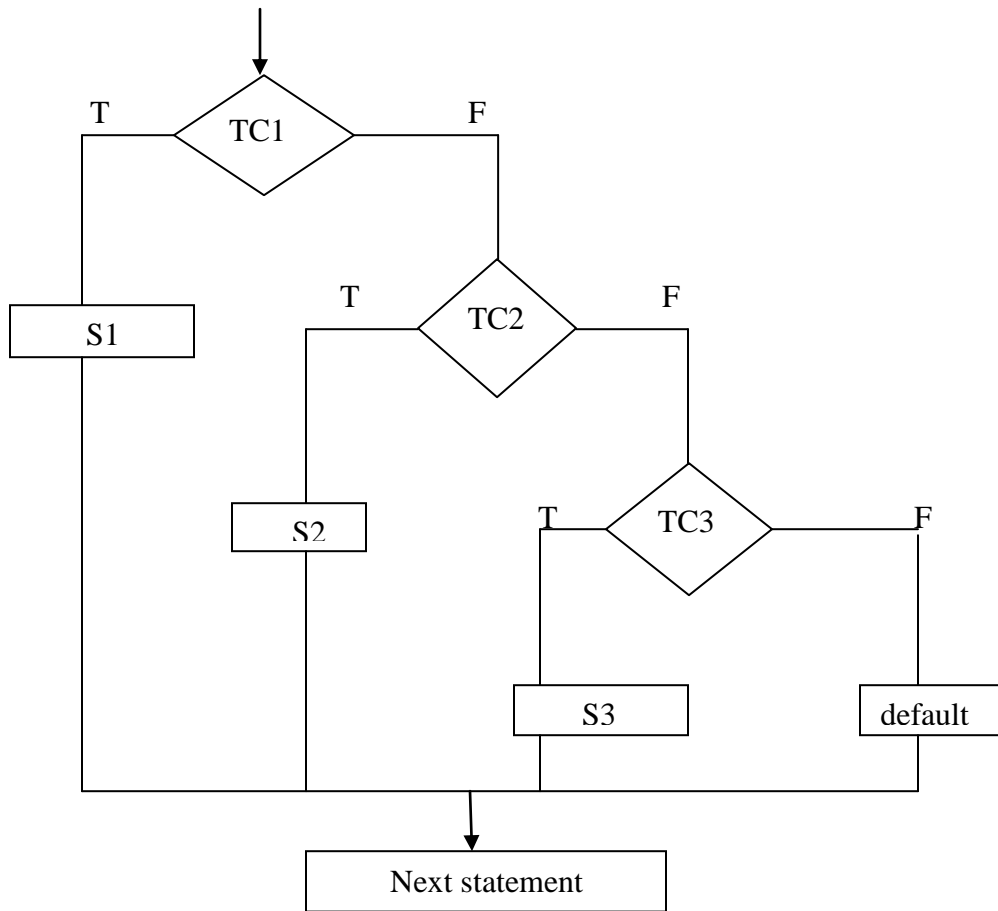
Syntax:

```
if(test_condition1)
    {
        Statement 1;
    }
else if(test_condition2)
    {
        Statement 2;
    }
else if(test_condition3)
    {
        Statement 3;
    }
else
    {
default statement;
    }
```

Example:

```
#include<iostream.h>
#include<iomanip.h>
#include<conio.h>
void main()
    {
int units;
float amount;
clrscr();
cout<<"Enter the units consumed:";
cin>>units;
if(units < 30)
amount = 3.50 * units;
else if( units >=30 && units <50)
amount = 4.25 * units;
else if( units >=50 && units <100)
amount = 5.25 * units;
else
amount = 5.85 * units;
cout<<"The repaid amount is"<<"="<<amount<<endl;
getch();
    }
```

Flowchart:

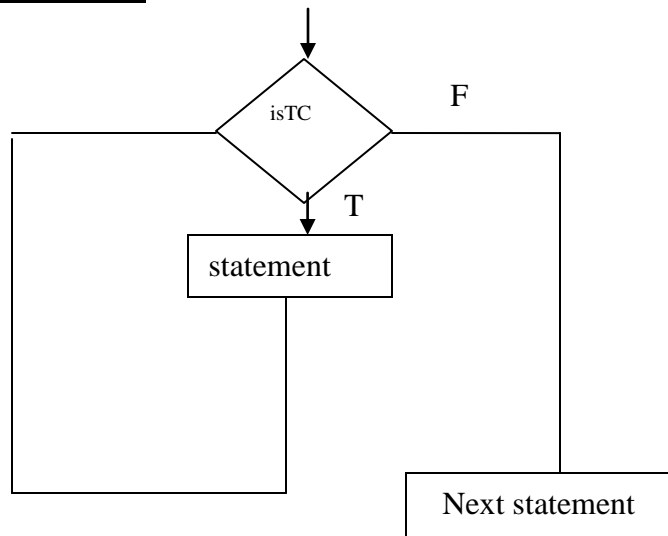


39. Explain while-do construct with an example.

Ans: **While-do construct:** It is also known as pre-tested loop. Condition is checked before looping.

Syntax: while(test_condition)
{
 statement1;
 ;
 statement n;
}

Flowchart:



Example:

```
#include<iostream.h>
#include<iomanip.h>
#include<conio.h>
void main()
{
int time, year;
floatpriamt, netamt, rate, ci;
clrscr();
cout<<"Enter the priamt, time and rate";
cin>>priamt>>time>>rate;
netamt = priamt;
year = 1;
while(year <= time)
netamt = netamt*(1+rate/100);
year++;
ci = netamt - priamt;
cout<<"Compound interst ="<<ci<<endl;
cout<<"Nett amount ="<<netamt<<endl;
getch();
}
```

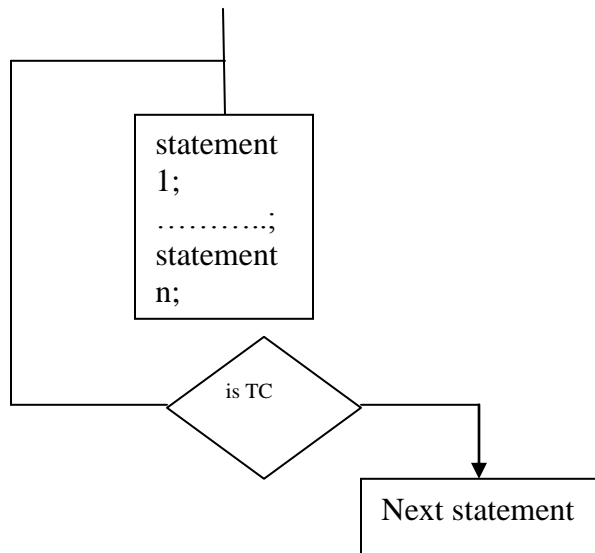
40. Explain do-while construct with an example.

Ans: **While-do construct:** It is also known as pre-tested loop. Condition is checked before looping.

Syntax: do

```
{
statement1;
.....;
statement n;
} while(test_condition);
```

Flowchart:



Example:

```
#include<iostream.h>
#include<iomanip.h>
#include<conio.h>
void main()
{
intn,temp, sum, rem;
```

```
clrscr();
cout<<"Enter a number:";
cin>>n;
temp = n;
sum = 0;

do
    {
rem = n%10;
sum = sum + rem*rem*rem;
    n =n/10;
    } while(n!=0)
if(sum==temp)
cout<<"It is an Armstrong number"<<endl;
else
cout<<"It is not an Armstrong number"<<endl;
getch();
}
```

```
*****
*****
```