

4

CHAPTER

PROBLEM SOLVING TECHNIQUE

4.1 NEED FOR LOGICAL ANALYSIS AND THINKING

A computer can do a variety of tasks like receiving data, processing the data and producing the results. It needs to be instructed to do a task. The computer works on a set of instructions called a program, to carry out a particular task.

To produce an effective execution of the computer program, it is very important that the programmer must take care of each and every step or instruction in a proper sequence.

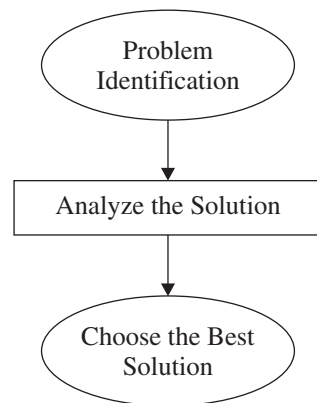
To design a program, a programmer must determine three basic steps:

1. The instructions to be performed.
2. The sequence in which those instructions are to be performed.
3. The data required to perform those instructions.

For example, to calculate the area of a rectangle, read the value of length and breadth, then multiply length and breadth and store the area value in A.

To solve a problem using computer, the following steps are to be followed:

- (a) Problem must be analysed carefully.
- (b) Find the various solution methods, and then choose the best solution.
- (c) Solution method is that the problem is broken down into a sequence of small tasks.
- (d) Based on the analysis, an algorithm or a pseudo code must be prepared to solve the problem.
- (e) The computer program is fed into the computer.
- (f) Test the instructions with the sample input.



Input

Length, Breadth

Processing

$A = \text{Length} \times \text{Breadth}$

Output

A

4.2 CHARACTERISTICS OF A GOOD PROGRAM

Every computer requires an appropriate set of instructions to perform the required task. The quality of the processing depends upon the given instructions. Hence, proper and correct instructions should be provided to the computer so that it provides the correct output. Important characteristics of a good program are:

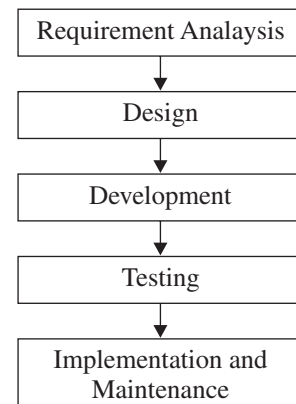
- ❖ Portability
- ❖ Readability
- ❖ Efficiency
- ❖ Structural
- ❖ Flexibility
- ❖ Generality
- ❖ Documentation

4.3 SOFTWARE DEVELOPMENT STEPS

The software development process passes through different phases before a useable application emerges, thus there are secured ways to progress through these phases, and each way is called a “software process”. The phases of the software development steps are considered as the system development life cycle.

The phases of the software development steps are:

- (a) Requirement Analysis
- (b) Design
- (c) Development
- (d) Testing
- (e) Implementation and Maintenance



4.3.1 Requirement Analysis

The objective of the requirement analysis is to identify and document the user requirements for the proposed system. It is also the process of understanding what is required, and expressing the result in writing. This is the first and the most important step in the software development.

The main objective of the requirement analysis is to produce a document that properly specifies all the requirements of the customer. That is called Software Requirement Specification (SRS) document.

4.3.2 Design Process

The design phase is the process of designing how the specifications are to be implemented. It defines specifically how the software is to be written including an object model with properties and method.

Design phase could be very expansive to solve in the later stage of the software development, so much care is taken during this phase.

4.3.3 Development or Coding

The design must be translated into a machine readable form. The coding or development step performs this task. The development phase involves the actual coding of the entire application.

Programming tools like compiler, interpreter, and debugger are used to generate the code. Different High level languages like C, C++, java, COBOL, etc., are used for writing programs. Depending upon the application, the right programming language is chosen.

4.3.4 Testing

Testing is the process of executing software with sample, or test data that is put into regular use. Different testing methodologies are used for correcting the error. Different testing tools are available for detecting and correcting the errors.

4.3.5 Implementation and Maintenance

This involves installation and initial training and may involve hardware and network upgrades. Software will definitely undergo changes once it is delivered to the customer.

Changes may happen because of some unexpected input values into the system. Changes in the system can directly affect the software operation.

The maintenance phase of the project is the last component and it continues as long as warranty, extended warranty or support contract is in place.

4.4 PROBLEM SOLVING TECHNIQUES

Problem solving techniques is a set of techniques and graphical tools that helps in providing logic for solving a problem. These tools are used to express the logic of the problem by specifying the correct sequence of all the instructions to be carried out.

There are three important problem solving tools used for program development. They are:

- (i) Algorithms
- (ii) Flowcharts
- (iii) Pseudo codes

4.4.1 Algorithms

Algorithms are one of the most basic tools that are used to develop the problem solving logics. An algorithm is defined as a finite sequence of explicit instructions, that when provided with a set of input values, process an output and then terminates. Algorithms can have steps that repeat or require decisions until the task is completed.

Example: To find out the largest number among three numbers A, B & C.

Step 1: Start

Step 2: Read three numbers A, B & C

Step 3: Find the largest number between A & B and store it in Max_AB

Step 4: Find the larger number between MAX_AB and C and store it in MAX

Step 5: Display MAX

Step 6: Stop

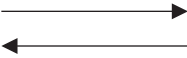

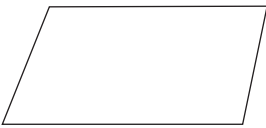

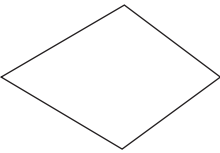
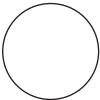
Properties of Algorithms

1. There must be no ambiguity in any instruction.
2. There should not be any uncertainty about which instruction is to be executed next.
3. The algorithm should conclude after a finite number of steps.
4. The algorithm must be general enough to deal with any contingency.

4.4.2 Flowchart

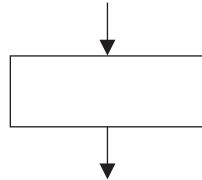
A flowchart is a pictorial representation of an algorithm in which the steps are drawn in the form of different shapes of boxes and the logical flow is indicated by interconnecting arrows. The boxes represent operations and the arrows represent the sequence in which the operations are implemented.

Flowchart Symbols

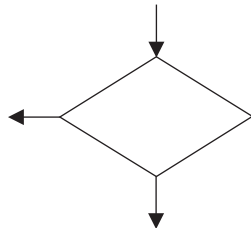
Symbol	Symbol Name	Description
	Flow Lines	Used to connect symbols
	Terminal	Used to represent start, pause or halt in the program logic.
	Input/Output	Represents the information entering or leaving the system.
	Processing	Represents arithmetic and logical instructions.
	Decision	Represents a decision to be made.
	Connector	Used to join different flow lines.

Guidelines for preparing flowcharts:

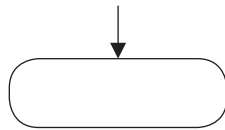
1. The flowchart should be clear, neat and easy to follow.
2. The flowchart must have a logical start and finish.
3. Only one flow line should come out from a process symbol.



4. Only one flow line should enter a decision symbol. However, two or three flow lines may leave the decision symbol.



5. Only one flow line is used with a terminal symbol.



6. Within standard symbols, write briefly and precisely.
7. Intersection of flow lines should be avoided.
8. It is useful to test the validity of the flowchart with normal/unusual test data.

Benefits of Flowcharts:

A flowchart helps to clarify how things are currently working and how they could be improved. It also assists in finding the key elements of a process by drawing clear lines between the end of one process and the start of next one.

Reasons for using flowcharts as a problem solving tool are given below:

- (i) Makes logic clear
- (ii) Communication
- (iii) Effective analysis
- (iv) Useful in coding
- (v) Proper testing and debugging
- (vi) Appropriate documentation

Limitations of Flowcharts:

The limitations of flowcharts are as follows:

- (i) Complex
- (ii) Costly
- (iii) Difficult to modify
- (iv) No update

4.4.3 Pseudo Code

Pseudo code is made up of two words, pseudo and code. It means imitation and code referring to instructions written in a programming language. Pseudo code is an outline of a program. It uses English statements rather than symbols. It is also known as PDL (Program Design Language).

Example Pseudo code 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.
```

Some of the keywords used in pseudo code are given below:

- ❖ **Input** READ, OBTAIN, GET and PROMPT
- ❖ **Output** PRINT, DISPLAY and SHOW
- ❖ **Compute** COMPUTE, CALCULATE and DETERMINE
- ❖ **Initialize** SET and INITIALISE
- ❖ **Add** ONE INCREMENT

Benefits of Pseudo Code:

- ❖ Since it is independent of any language, it can be used by most programmers.
- ❖ It is easy to develop a program from pseudo code than with a flowchart.
- ❖ It is easy to translate pseudo code into a programming language.
- ❖ Pseudo code is compact and does not tend to run over many pages.

Limitations of Pseudo Code:

- ❖ It does not provide visual representation of the program's logic.
- ❖ There are no accepted standards for writing pseudo codes.

4.5 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.

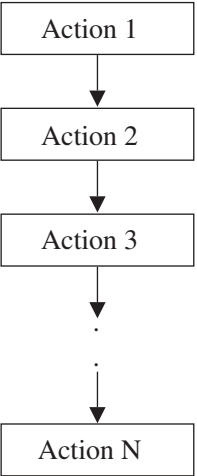
They are as follows:

- (i) **Sequence Control Structures** Information flows in a straight line.
- (ii) **Selection Control Structures** Decisions are made according to predefined conditions.
- (iii) **Repetition Control Structures** Sequence of steps is repeated in a loop until the desired output is obtained.

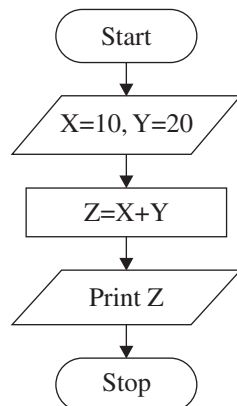
4.5.1 Sequence Control Structure

In a sequence control structure, the instructions to be computed follow one another in a logical progression. This structure is denoted by writing one action after another, each action on a line by itself. In addition, all actions are aligned with the logical indent.

Diagrammatic Representation

Flow Chart	Pseudo code
 <pre> graph TD A1[Action 1] --> A2[Action 2] A2 --> A3[Action 3] A3 --> Dots[...] Dots --> AN[Action N] </pre>	<pre> Action 1 Action 2 Action 3 ... Action N </pre>

Example:



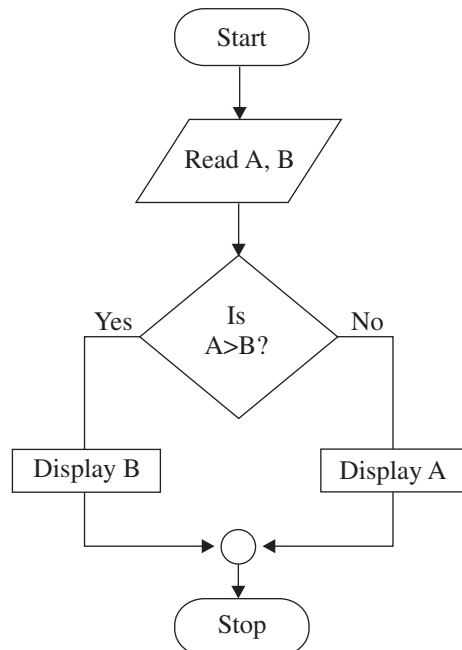
4.5.2 Selection Control Structure

A selection control structure allows the program to make a choice between two alternate paths, whether it is true or false. The first statement of a selection structure is a conditional statement. Once the sequence of steps in the selected path has been carried out, the paths are rejoined and then the next instruction is carried out. Thus, the selection structure has only a single entry and single exit.

Diagrammatic Representation

Flow Chart	Pseudo Code
<pre> graph TD Entry(()) --> Decision{Is Condition true?} Decision -- No --> Action2[Action 2] Decision -- Yes --> Action1[Action 1] Action2 --> Exit(()) Action1 --> Exit Exit --> Exit </pre>	<pre> . . IF (Condition is True) then List of Actions ELSE List of Different Actions ENDIF </pre>

Example:



4.5.3 Repetition Control Structure

Repetition or loop pattern causes an interruption in the normal sequence of processing. It directs the system to loop back to a previous statement in the program, repeating the same sequence over and again usually with new data. When a sequence of statements is repeated against a condition, it is said to be in a loop. The looping process can either be one time or multiple times until the desired output is obtained within a single program.

Diagrammatic Representation

Flow Chart	Pseudo Code
<pre> graph TD RT[Repeated Task] --> C{Condition} C -- Flase --> RT C -- True --> Exit(()) </pre>	<pre> REPEAT Sequence 1 Sequence 2 . . Sequence N UNTIL condition is false </pre>

Example:

