B.Sc. (I.T.) / M.Sc. (I.T.) 1s Semester

Course: 104: Fundamentals of Programming Using C-1

Unit 1: Phases of Problem-Solving Methodology

- **1.1 Problem Analysis**
  - Gathering available data, identifying relevant facts, Defining the problem, generating alternative methods of solution, Selecting the optimum approach
- **1.2 Problem solving techniques**
  - Simplification, Divide and conquer: break down a large, complex problem into smaller solvable problems, Constraint examination
- **1.3 Algorithm**
- **1.4 Flowchart**

1.1 Problem Analysis

Definition:
Problem Analysis is the process of studying a problem in detail to understand what is required, what is given, and what constraints exist before developing a solution. It is the first and most important step in programming.

Steps in Problem Analysis

1. **Gathering Available Data**
   - Collect all relevant information about the problem.
   - *Example:* To calculate the average marks of students, first gather all students' marks.
2. **Identifying Relevant Facts**
   - Separate useful data from unnecessary details.
   - *Example:* For average marks, student roll numbers may be useful, but their home address is not.
3. **Defining the Problem**
   - Clearly state the problem in simple terms.
   - *Example:* "Calculate the average marks of 50 students."
4. **Generating Alternative Methods of Solution**
   - Think of multiple approaches to solve the problem.
   - *Example:* Using a loop to sum marks OR using built-in functions.
5. **Selecting the Optimum Approach**
   - Choose the most efficient and practical method.
   - *Example:* Beginners can use loops for clarity; advanced users may use library functions.

1.2 Problem Solving Techniques

1. **Simplification**
    - Break a complex problem into simpler parts.
    - *Example:* To calculate compound interest, first calculate simple interest and then extend the logic.
2. **Divide and Conquer**
    - Divide a large problem into smaller sub-problems, solve each, and combine.
    - *Example:* Sorting 1000 numbers can be done by breaking them into groups, sorting each, then merging.
3. **Constraint Examination**
    - Identify restrictions such as time, memory, input size, etc.
    - *Example:* If only 1 MB memory is available, we cannot load a very large dataset at once.

1.3 Algorithm

Definition:
An Algorithm is a step-by-step procedure to solve a given problem in a finite number of steps.

Characteristics of an Algorithm

- Must be finite (end after limited steps).
- Must be definite (each step clear and unambiguous).
- Must be correct (produce desired output).
- Must be feasible (practically implementable).
- Must have input and output.

Example: Algorithm to Find Maximum of Two Numbers

Step 1: Start

Step 2: Input two numbers A and B

Step 3: If A > B then

Print "A is greater"

Else

Print "B is greater"

Step 4: Stop

1.4 Flowchart

Definition:
A **Flowchart** is a **diagrammatic representation** of a process or algorithm using standard symbols.

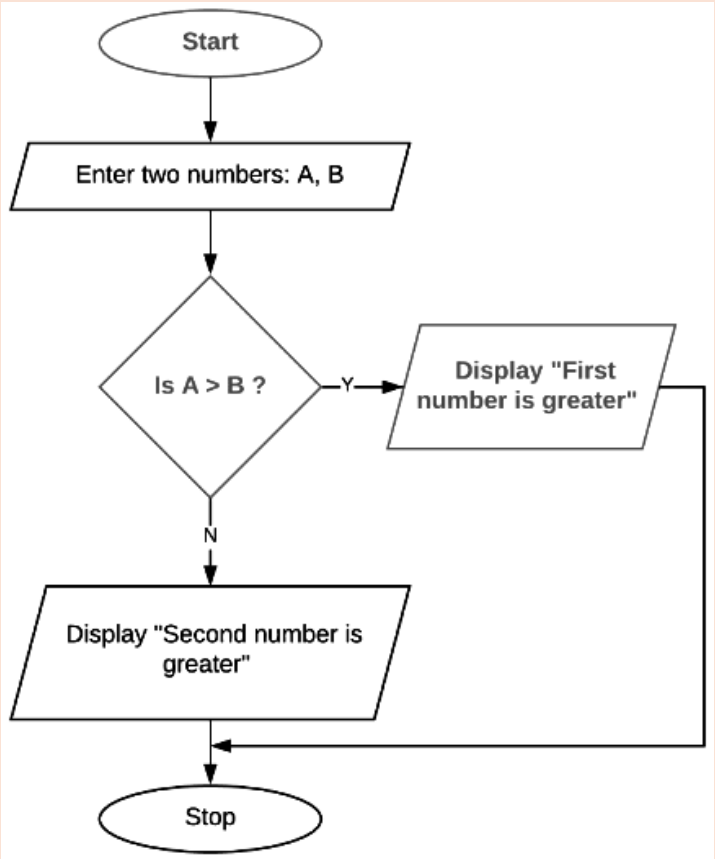
Advantages of Flowcharts

- Easy to understand logic.
- Simplifies program development.
- Helps in debugging and documentation.

Common Flowchart Symbols

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

Example: Flowchart to Find Maximum of Two Numbers



Quick Revision Notes

- Problem Analysis → Understanding, Defining, Selecting best approach.
- Techniques → Simplification, Divide & Conquer, Constraints.
- Algorithm → Step-by-step, finite, clear, correct.
- Flowchart → Graphical representation using symbols.
- Classic Exam Q: Max of two numbers (both Algorithm + Flowchart).