Chapter 3 - Problem Solving & Program Planning

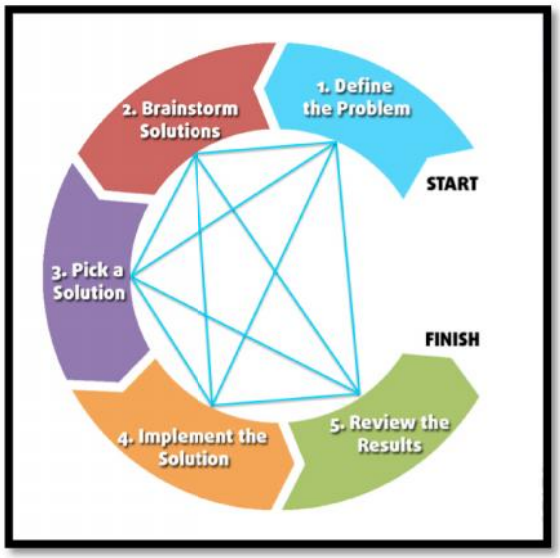
Need for problem solving and planning a program

Problem – a state of desire for reaching a definite goal from a present condition

Solution – the management of a problem in a way that successfully meets the goals set for treating it.

THE FIVE STEPS OF PROBLEM SOLVING

With that understanding of problem solving, let's talk about the steps that can get you there. The five problem solving steps are shown in the chart below:



1. Define the Problem

- Input: something is wrong or something could be improved.
- Output: a clear definition of the opportunity and a goal for fixing it.

2. Brainstorm Ideas

- Input: a goal; research of the problem and possible solutions; imagination.
- Output: pick-list of possible solutions that would achieve the stated goal.

3. Decide on a Solution

- Input: pick-list of possible solutions; decision-making criteria.
- Output: decision of what solution you will implement.

4. Implement the Solution

- Input: decision; planning; hard work.
- Output: resolution to the problem.

5. Review the Results

- Input: resolutions; results of the implementation.
- Output: insights; case-studies; bullets on your resume.

Program design tools: Algorithms, Flowcharts and Pseudocode

Algorithms

Algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output. Algorithms are generally created independent of underlying languages, i.e. an algorithm can be implemented in more than one programming language.

From the data structure point of view, following are some important categories of algorithms −

- **Search** − Algorithm to search an item in a data structure.
- **Sort** − Algorithm to sort items in a certain order.
- **Insert** − Algorithm to insert item in a data structure.
- **Update** − Algorithm to update an existing item in a data structure.
- **Delete** − Algorithm to delete an existing item from a data structure.

Characteristics of an Algorithm

Not all procedures can be called an algorithm. An algorithm should have the following characteristics −

- **Unambiguous** − Algorithm should be clear and unambiguous. Each of its steps (or phases), and their inputs/outputs should be clear and must lead to only one meaning.
- **Input** − An algorithm should have 0 or more well-defined inputs.
- **Output** − An algorithm should have 1 or more well-defined outputs, and should match the desired output.
- **Finiteness** − Algorithms must terminate after a finite number of steps.
- **Feasibility** − Should be feasible with the available resources.
- **Independent** − An algorithm should have step-by-step directions, which should be independent of any programming code.

Algorithm to find if a number is even or odd:

Step 1: Start
Step 2: Input n (where n is integer)
Step 3: if(n%2==0) (where % gives us remainder)
Print "Even Number"
else
Print "Odd Number"
Step 4: Stop

Algorithm to find if a number is prime or not:

Step 1: Start
Step 2: Declare variables n,i,flag.
Step 3: Initialize variables
flag←1
i←2
Step 4: Read n from user.
Step 5: Repeat the steps until i < (n/2)
5.1 If remainder of n divide by i equals 0
flag←0
Go to step 6
5.2 i←i+1
Step 6: If flag=0
Display n is not prime
else
Display n is prime
Step 7: Stop

Flowcharts

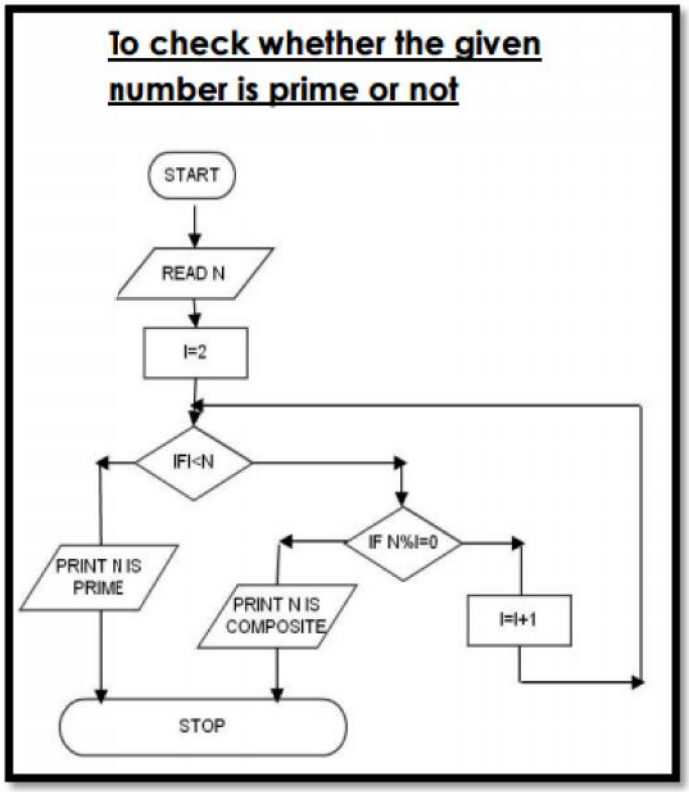
A flow chart is a graphical or symbolic representation of a process. Each step in the process is represented by a different symbol and contains a short description of the process step. The flow chart symbols are linked together with arrows showing the process flow direction.

Flowcharts use simple geometric symbols and arrows to define relationships. In programming, for instance, the beginning or end of a program is represented by an oval. A process is represented by a rectangle, a decision is represented by a diamond and an I/O process is represented by a parallelogram. The Internet is represented by a cloud.

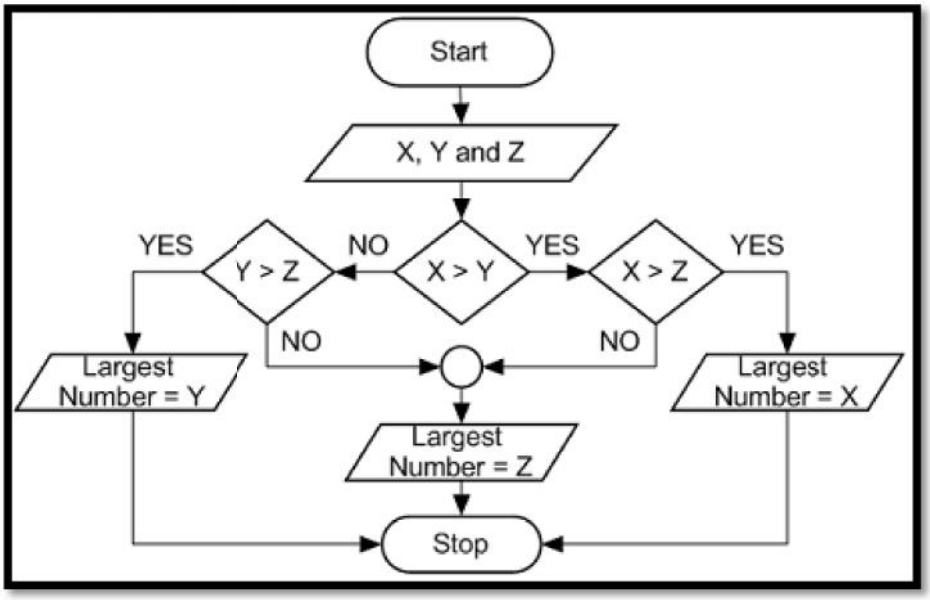
Symbols used in flowcharts are as follows:

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

Flowchart to find if a number is prime or not:



Flowchart to find the largest out of three numbers:



Pseudocode

Pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm.

It uses the structural conventions of a normal programming language, but is intended for human reading rather than machine reading. Pseudocode typically omits details that are essential for machine understanding of the algorithm, such as variable declarations, system-specific code and some subroutines. The programming language is augmented with natural language description details, where convenient, or with compact mathematical notation. The purpose of using pseudocode is that it is easier for people to understand than conventional programming language code, and that it is an efficient and environment-independent description of the key principles of an algorithm. It is commonly used in textbooks and scientific publications that are documenting various algorithms, and also in planning of computer program development, for sketching out the structure of the program before the actual coding takes place.

Example: Write a pseudocode for printing Fibonacci series:

- Declare an integer variable called n
- Declare an integer variable sum
- Declare an integer variable f1
- Declare an integer variable f2
- set sum to 0
- set f1 and f2 to 1
- set n to 50
- repeat n times
  - ✓ sum = f1 + f2
  - ✓ f2 = f1
  - ✓ f1 = sum
  - ✓ print sum
- end loop