Problem Solving Methodology

Sisira Kumar Kapat

Department of Computer Science & Engineering,
UCP Engineering School,
Berhampur, Orissa

Contents

- Algorithm, Pseudo code and Flowchart
- Generation of Programming Languages
- Structured Programming Language
- Examples of Problem solving through Flowchart

Introduction

- Problem
- Requirement to solve a problem

Algorithm

- Algorithm is a finite sequence of well-defined, computer-implementable instructions, typically used to solve a class of problems or to perform a computation
- **Properties**
  - **Clear and Unambiguous**: Algorithm should be clear and unambiguous. Each of its steps should be clear in all aspects and must lead to only one meaning.
  - **Well-Defined Inputs**: If an algorithm says to take inputs, it should be well-defined inputs.
  - **Well-Defined Outputs:** The algorithm must clearly define what output will be yielded and it should be well-defined as well.
  - **Finiteness:** The algorithm must be finite, i.e. it should not end up in an infinite loops or similar.
  - **Feasible:** The algorithm must be simple, generic and practical, such that it can be executed upon will the available resources. It must not contain some future technology, or anything.
  - **Language Independent:** The Algorithm designed must be language-independent, i.e. it must be just plain instructions that can be implemented in any language, and yet the output will be same, as expected

Pseudo code

- Pseudocode is an informal way of programming description that does not require any strict programming language syntax or underlying technology considerations.
- It is used for creating an outline or a rough draft of a program.
- Pseudocode summarizes a program’s flow, but excludes underlying details.
- System designers write pseudocode to ensure that programmers understand a software project's requirements and align code accordingly.

Pseudo code

- Pseudocode is not an actual programming language. So it cannot be compiled into an executable program.
- It uses short terms or simple English language syntaxes to write code for programs before it is actually converted into a specific programming language.
- This is done to identify top level flow errors, and understand the programming data flows that the final program is going to use.

Analyse an Algorithm

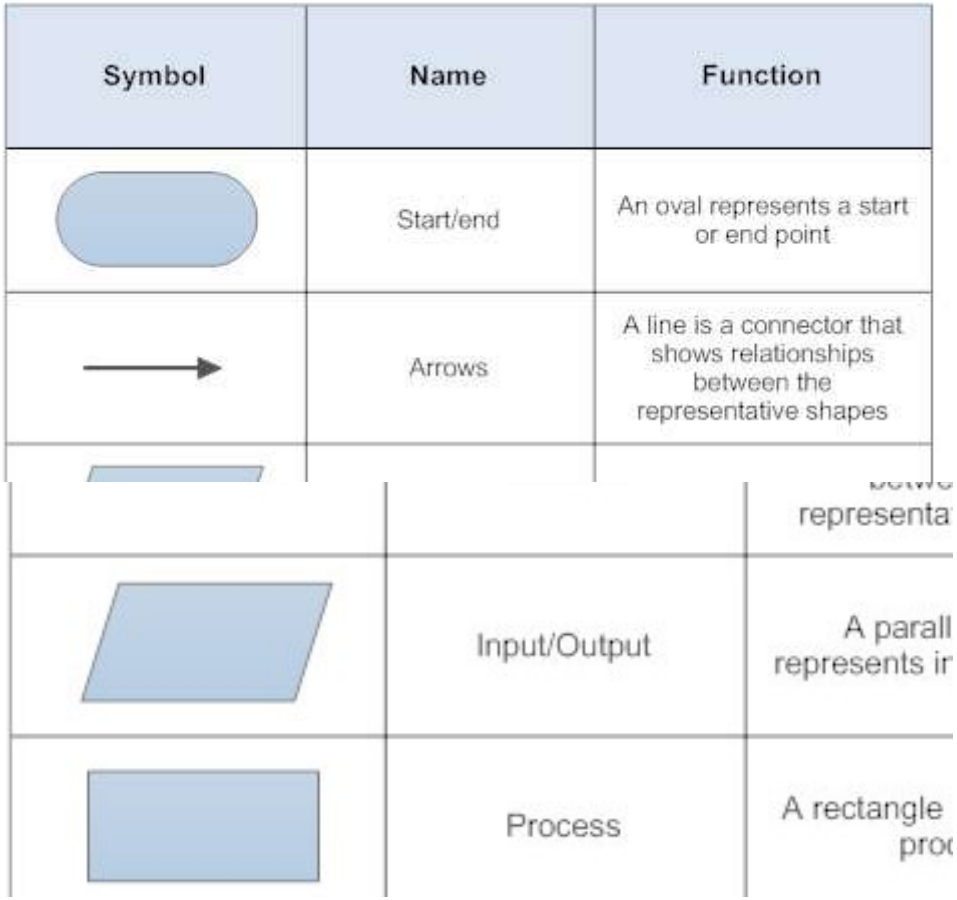
- For a standard algorithm to be good, it must be efficient. Hence the efficiency of an algorithm must be checked and maintained. It can be in two stages:
  - **Priori Analysis:** “Priori” means “before”. Hence Priori analysis means checking the algorithm before its implementation. In this, the algorithm is checked when it is written in the form of theoretical steps. This Efficiency of an algorithm is measured by assuming that all other factors, for example, processor speed, are constant and have no effect on the implementation. This is done usually by the algorithm designer. It is in this method, that the Algorithm Complexity is determined.
  - **Posterior Analysis:** “Posterior” means “after”. Hence Posterior analysis means checking the algorithm after its implementation. In this, the algorithm is checked by implementing it in any programming language and executing it. This analysis helps to get the actual and real analysis report about correctness, space required, time consumed etc.

Algorithm Complexity

- **Space Complexity:** Space complexity of an algorithm refers to the amount of memory that this algorithm requires to execute and get the result. This can be for inputs, temporary operations, or outputs.
- **Time Complexity:** Time complexity of an algorithm refers to the amount of time that this algorithm requires to execute and get the result. This can be for normal operations, conditional if-else statements, loop statements, etc.

Flowchart

- Symbols used
  - Start/end
  - Input/Output
  - Process
  - Decision
  - Control flow



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
		representa
	Input/Output	A parall represents ir
	Process	A rectangle proc

Solving a problem

- **Example:** Consider the example to add three numbers and print the sum.
- **Step 1: Fulfilling the pre-requisites**As discussed above, in order to write an algorithm, its pre-requisites must be fulfilled.
  - **The problem that is to be solved by this algorithm**: Add 3 numbers and print their sum.
  - **The constraints of the problem that must be considered while solving the problem**: The numbers must contain only digits and no other characters.
  - **The input to be taken to solve the problem:** The three numbers to be added.
  - **The output to be expected when the problem the is solved:** The sum of the three numbers taken as the input.
  - **The solution to this problem, in the given constraints:** The solution consists of adding the 3 numbers. It can be done with the help of '+' operator, or bit-wise, or any other method.

Solving a problem (cont…)

- **Step 2: Designing the algorithm**Now let's design the algorithm with the help of above pre-requisites:
- **Algorithm to add 3 numbers and print their sum:**
  - START
  - Declare 3 integer variables num1, num2 and num3.
  - Take the three numbers, to be added, as inputs in variables num1, num2, and num3 respectively.
  - Declare an integer variable sum to store the resultant sum of the 3 numbers.
  - Add the 3 numbers and store the result in the variable sum.
  - Print the value of variable sum
  - END
- **Step 3: Testing the algorithm by implementing it.**Inorder to test the algorithm, let's implement it in C language.

Example: Addition of first 10 natural number

- Algorithm

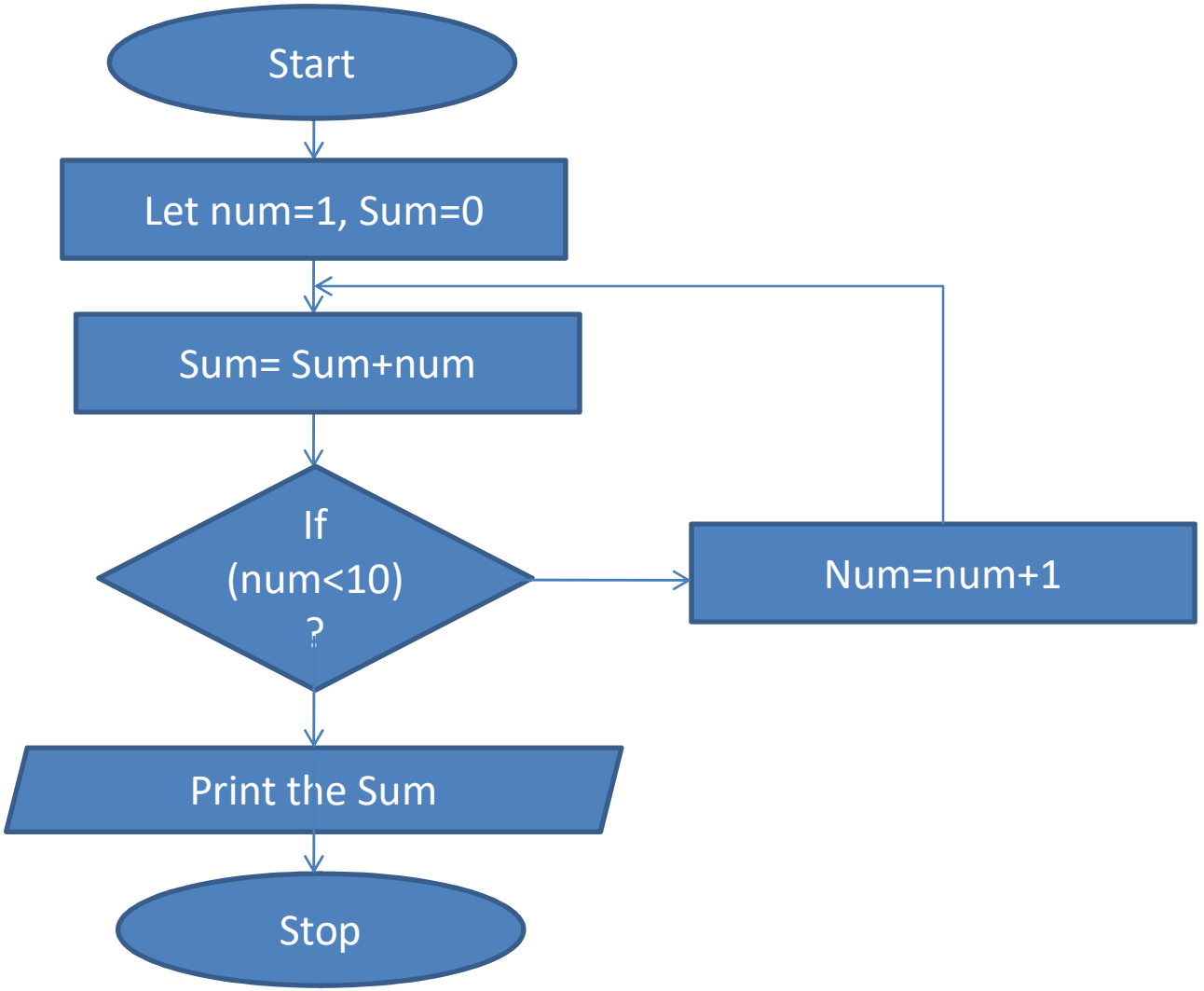
Step 1: Start
Step 2: Declare and initialize variable, num=1, sum=0.
Step 3: sum=sum+num
Step 4: if the value of num is less than 10, then goto Step-5
Step 5: num=num+1 and goto step-3
Step 5: Print the sum
Step 6: Stop

Example: Addition of first 10 natural number (cont…)

- Pseudocode

1. Int num=1, s=0
2. sum=sum+num
3. if(num<10) then goto step-4 else goto step-5
4. do num=num+1 and goto step-2
5. till the value of num<10
6. Print the sum of numbers.

- Flow chart


Start
Let num=1, Sum=0
Sum= Sum+num
If
(num<10)
?
Num=num+1
Print the Sum
Stop

Example: Addition of first 10 natural number (cont…)

- C Program

```
#include<stdio.h>
void main()
{
        int num=1, sum=0;
        ab: sum=sum+num;
        if(num<10)
        {
                num=num+1;
                goto ab;
        }
        else
                printf("\n The sum of first 10 natural number is %d\n", sum);
}
```