Unit : 1 Problem Solving

Lesson Structure

1.0 Objective

After going through this unit, you will learn to

- implement the concept of problem solving techniques;

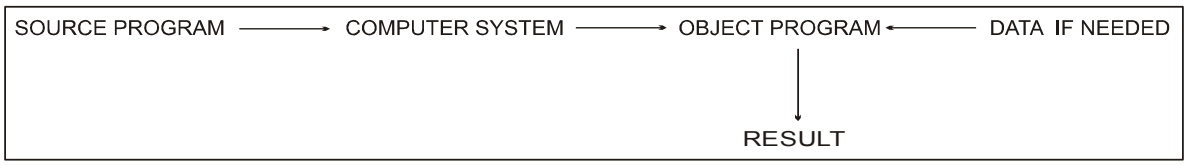
- develop the algorithm which is used in the design phase of software development process to help the programmers and users to clearly understand the solution to the problem.
- perform the analysis of algorithm efficiency;
- understand the different symbols used to design flowcharts.
- draw flowchart for the problem.

1.1 INTRODUCTION

A Computer is a very powerful machine capable of performing different tasks. It has no intelligence or thinking power. It cannot think in the way we human being can think. It is the responsibility of the user to instruct the computer in a correct manner, so that the machine is able to perform the required job in a proper way. A wrong instruction may sometimes create problem.

When you are using the computer to solve a problem, you must specify the needed initial input data, the operations (process) which need to be performed (in order of performance) and what results you want for output. If any of these instructions are missing, you will get either no results or invalid results. Therefore, several steps need to be considered before writing a program. A computer is a symbol-manipulating machine that follows a set of stored instructions called a program. A computer performs manipulations very quickly and has memory for storing input, process and output.

Set of instructions of the high level language used to code a problem to find its solution is reffered to as Source Program. A translator program called a compiler or interpreter, translates the source program into the object program. This is the compilation or interpretation phase. If there is no error, the source program is transformed into the machine language program called Object Program. The Object Program is excuted to perform calculations. This stage is the execution phase. Data, if required by the program, are supplied now and the results are obtained on the output device.



1.2 PROBLEM - SOLVING TECHNIQUES

A computer cannot solve a problem on its own. One has to provide step by step

solutions of the problem to the computer. Infact the task of problem solving is not that of the computer. It is the programmer who has to write down the solution to the problem in terms of simple operations which the computer can understand and execute.

1.2.1 Steps for Problem - Solving

Problem solving is a creative process which defines systematization and machanization. The steps of problem solving are as following.

i) Problem Understanding phase

The success in solving any problem is possible only after the problem has been fully understood. That is, we cannot hope to solve a problem, which we do not understand. So, the problem understanding is the first step towards the solution of the problem. In this phase we should be absolutely sure about the objectives of the given problem.

ii) Getting started on a problem

There are many ways of solving a problem and there may be several solution. So, it is difficult to recognize immediately which path sould be more productive. Sometimes you do not have any idea where to begin solving a problem, even if the problem has been defined. The best advice is not to get concerned with the details of the implementation.

iii) Picking of specific examples

To get started on a problem we can do it by picking a specific problem we wish to solve and try to work out the mechanism that will allow solving the particular problem. It is usually much easier to work out the details of a solution to a specific problem because the relationship between the mechanism and the problem is more clearly defined. This approach of focusing on a patricular problem can give us the foothold we need for making a start on the solution to the general problem.

iv) Similarities among problems

First of all it is importance to see if there are any similarities between the current problem and the past problems which we have solved. The more experience one has the more tools and techniques he/she can use in solving the given problem. But sometimes, it blocks us from discovering a desirable or better solution to the problem. It is important to try to develop a skill in problem - solving that is to view a problem from a variety of angles.

v) Moving backwards from the solution

In some cases we can assume that we already have the solution to the problem and

then try to work backwards to the starting point. Even a guess at the solution to the problem may be enough to give us a foothold to start on the problem.

1.2.2 Using Computer as a Problem - Solving Tool

A computer is a very powerful general - purpose tool. Computer can solve or help to solve many types of problems. There are also many ways in which a computer can enhance the effectiveness of the time and effort that you are willing to devote to solving a problem.

Program development is a multi-step process that requires you to understand the problem, develop a solutiln, write the program, and then test it. This critical process determines the overall quality and success of your program. If you carefully design each program using good structured development techniques, your programs will be efficient, error - free, and easy to maintain.

The following are the steps in detail:

1. Design of an Algorithm and a Flowchart.
2. Write the program in a computer language (for example say C programming language).
3. Enter the program using some editor.
4. Test and debug the program.
5. Run the program, input data, and get the results.

1.3 ALGORITHMS

The first step in the program development is to devise and describe a precise plan of what you want the computer to do. This plan, expressed as a sequence of operations, is called an algorithm. An algorithm is just an outline or idea behind a program.

1.3.1 Definition

The typical meaning of an algorithm is a formally defined procedure for performing some calculation. An algorithm is a finite set of steps defining the solution of a particular problem. It is expressed in pseudo code-something resembling C language or pascal, but with some statements in English rather than within the programming language.

Developing an efficient algorithm requires lot of practice and skill. It must be noted that an efficient algorithm is one which is capable of giving the solution to the problem by using minimum resources of the system such as memory and processor's time.

Algorithm is a language independent, well - structured and detailed. It will enable the programmer to translate into a computer program using any high-level language as it provides a blueprint for writing a porogram to solve a particular problem.

1.3.2 Properties of Algorithm

An algorithm must posses the following properties:

1. **Finiteness :** An algorithm must terminate in a finite number of steps.
2. **Definiteness :** Each step must be clear and unambiguous.
3. **Effectiveness :** Each step must be definite and it must also be feasible.
4. **Generality :** The algorithm must be complete in itself so that it can be used to solve problems of a specific type for any input.
5. **Input :** There should be zero or more values which are to be provided.
6. **Output :** At least one result is to be produced.

1.3.3 Key Features of Algorithms

An algorithm has a number of steps and some steps may involve decision-making and repetitions. An algorithm exhibits three key features which are as following:

i) Sequence

ii) Decision

iii) Repitition.

Sequence

It means that each step of the algorithm is executed in the specified order. For example an algorithm to add two numbers performs the steps in a purely sequential order.

Example 1.1

Write an algorithm to compute and display the sum of two numbers.

1. Start
2. Input two numbers a and b
3. Calculate sum = a + b
4. Print sum
5. Stop

Decision

Decision statements are used when the outcome of the process depends on some condition.

A decision statement can be stated in the following manner.

If condition then

Process 1

Else

Process 2

This is popularly known as the if-else construct. Here, if the condition is true then process 1 is excuted else process 2 is excuted. Example 1.2 shows an algorithum that checks if two numbers are equal.

Example 1.2

Write an algorithm to check if two entered numbers are equal.

1. Start
2. Input the first number, x
3. Input the second number, y
4. If x = y then

   Print “Both are Equal”

   else

   Print “ Both are not equal”
5. Stop

Repetition

Repitition involves executing one or more steps for a number of times.

It can be inplemented by using the constructs like while, do while and for loops. Example 1.3 shows an algorithum that prints the sum of first 10 natural numbers.

Example 1.3

1. Start.
2. Intialize and set i = 0, n = 10, sum = 0
3. Repeat steps 3 through 5 while i<=n

4. Set sum = sum + i
5. Set i = i + 1
6. Print sum
7. Stop

Some more Algorithums

Example 1.4

Write an algorithm to interchange/swap two values.

Solution :

1. Start
2. Input first number, a
3. Input second number, b
4. Set temp=a
5. Set a=b
6. Set b=temp
7. Print a, b
8. Stop.

Example 1.5

Write an algorithm to print the percentage and grade of a student using the following rules:

Percentage	Grade
Above 75	A
60-75	B
40-60	C
Less than 40	Fail

Solution :

1. Start
2. Input paper 1 marks, p1.
3. Input paper 2 marks, p2.

4. Input paper 3 marks, p3
5. Calcualate per= (p1+ p2 + p3)/3
6. if per>=75 then
   Print “A”
7. if per > = 60 and per <75 then
   Print “B”
8. if per>=40 and per < 60 then
   Print “C”
9. if per < 40 then
   Print “ Fail”
10. Stop

Example 1.6

Write an algorithm to calculate the factorial of a given number.

Solution

1. Start
2. Read the number n
3. Initialize and set i =1, fact = 1
4. Repeat steps 4 through 6 until i = n
5. fact=fact * i
6. i = i +1
7. Print fact
8. Stop

Example 1.7

Write an algorithm to check that whether the entered number is prime or not.

Solution

1. Start
2. Read the number as num
3. Initilialize and set i =2, flag = 1

4. Repeat steps 4 through 6 until i < num or flag = 0
5. rem= num mod i
6. if rem = 0 then

   flag = 0

   else

   i = i +1
7. if flag = 0 then

   Print " Number is not prime"

   else

   Print "Number is prime"
8. Stop

1.3.5 Analysis of Algorithm

Every algorithm uses some of the computer's resoueces like central processing time and internal memory to complete its task. Because of high cost of computing resources, it is desirable to design algorithm that are economical in the use of CPU time and memory. Efficiency of algorithms is tied in with the design, inplementation and analysis of algorithm.

1.3.5.1 Analysis of Algorithm Complexity

Algorithms usually possess the following qualities and capabilities:

- Easily modifiable if necessary.
- They are easy, general and powerful.
- They are correct for clearly defined solution.
- Require less computer time, storage and peripherals i.e they are more economical.
- They are documented well enough to be used by others who do not have a detailed knowledge of the inner working.
- They are not dependable on being run on a particular computer.
- The solution is pleasing and satisfying to its designer and user.
- They are able to be used as a sub-procedure for other problems.

Two or more algorithms can solve the same problem in different ways. So, quantitative measures are valuable in that they provide a way of comparing the performance of two or

more algorithms that are intended to solve the same problem. This is an important step because the use of an algorithm that is more efficient in terms of time, resources required, can save time and money.

1.3.5.2 Computational Complexity

We can characterize an algorithm's performance in terms of the size (usually n) of the problem being solved. More computing resources are needed to solve larger problems in the same class.

1.3.5.3 The Order of Notation

The O-notation can often describe the running time of an algorithm merely by inspecting the algorithm's overall structure. The O-notation gives an upper bound to a function within a constant factor. For a given function g (n), we denote by O (g (n)) the set of functions.

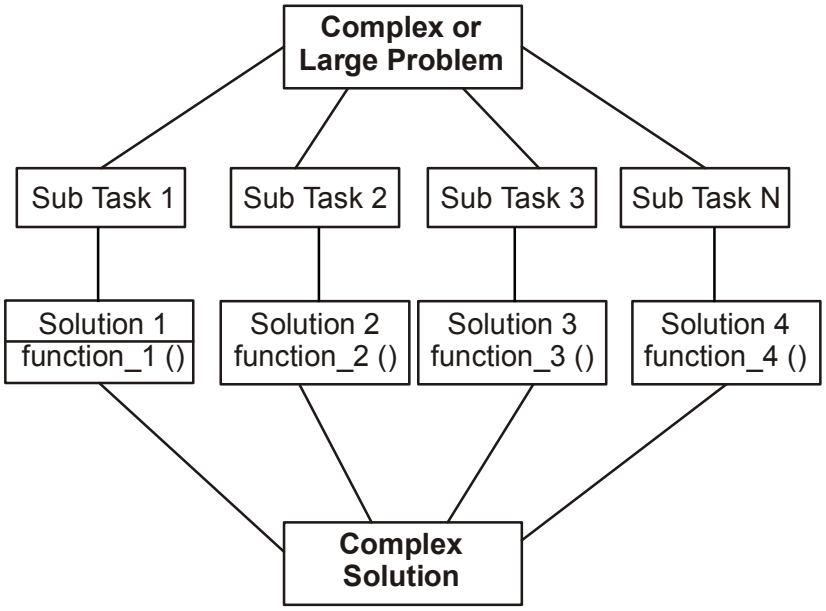
O(g(n)) = {f(n) : there exist positive constants c and no, such that o <= f(n) <= cg(n) for all n >= no }.

What we mean by saying "the running time is O (n^2)" is that the worst case running time (which is a function of n) is O (n^2). Or equivalently, no matter what particular input of size n is chosen for each value of n, the running time on that set of input is O (n^2).

1.4 TOP DOWN DESIGN

Once we have defind the problem and have an idea of how to solve it, we can then use the powerful techniques for designing algorithms. Most of the problems are complex or large problems and to solve them we have to focus on to comperhend at one time, a very limited span of logic or instructions. A technique for algorithm design that tries to accommodate this human limitation is known as top-down design or stepwise refinement.

Top down design suggests taking the general statements about the solution one at a time, and then breaking them down into a more precise subtask / sub-problem. These sub-problems should more accurately describe how the final goal can be reached. The process of repeatedly breaking a task down into a subtask and then each subtask into smaller subtask must continue until the sub-problem can be implemented as the program statement. With each spitting, it is essential to define how sub-problems interact with each other. In this way, the overall structure of the solution to the problem can be maintained. Preservation of the overall structure is important for making the algorithm comprehensible and also for making it possible to prove the correctness of the solution.



Schematic breakdown of a problem into subtasks as employed in top down design.

1. 5 FLOW CHARTS

The next step after the algorithm development is the flowcharting. Flowcharts are used in programming to diagram the path in which information is processed through a computer to obtain the desired results. Flowcharts is a graphical repesentation of an algorithm. It makes use of symbols which are connected among them to indicate the flow of information and procesing. It will show the general outline of how to solve a problem or perform a task. It is prepared for better understanding of the algorithm.

Advantages of flowcharts

1. The flowcharts shows the logic of a problem displayed in pictorial fashion which felecitates easier checking of an algorithm.
2. The flowchart is good means of communication to other users. It is also a compact means of recording an algorithm solution to a problem.
3. The flowchart allows the problem solver to break the problem into parts. These parts can be connected to make master chart.
4. The flowchart is a permanent record of the soluition which can be consulted at a later time.

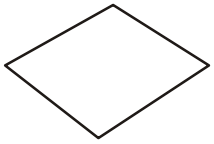
Difference between Algorithm and Flowchart.

Algorithm	Flowchart
1. A method of representing the step-by-step logical procedure for solving a a problem.	1. Flowchart is a diagrammatic representation of an algorithum. It is constructed using different types of boxes and symbols.
2. It contains step - by - step English description, each step representating a particular operation leading to solution of problem.	2. The flowchart employs a series of blocks and arrows, each of which represents a particular step in an algorithm.
3. These are particularly useful for small problems.	3. These are useful for detailed representations of complicated programs.
4. For complex programs, algorithms prove to be inadequate.	4. For complex programs, flowcharts prove to be adequate.

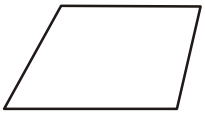
1.5.1 Basic Symbols used in flowchart design

1. Start / stop

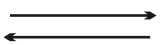
2. Questions, Decision (Use in Branching)



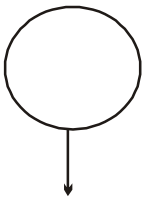
3. Input/output



4. Flow Lines or direction of the flow of control



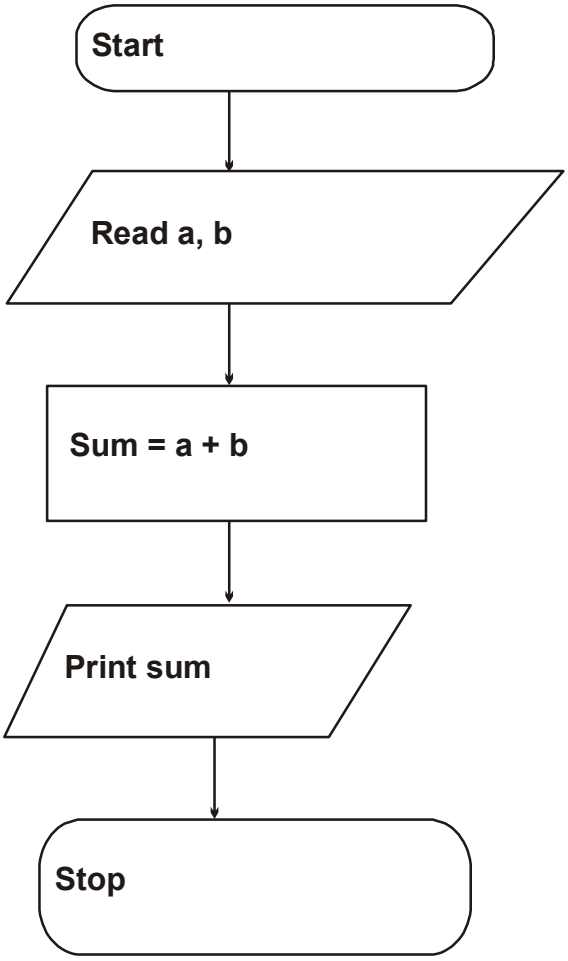
5. Connector (connect one part of Flowchart to another)



6. Process

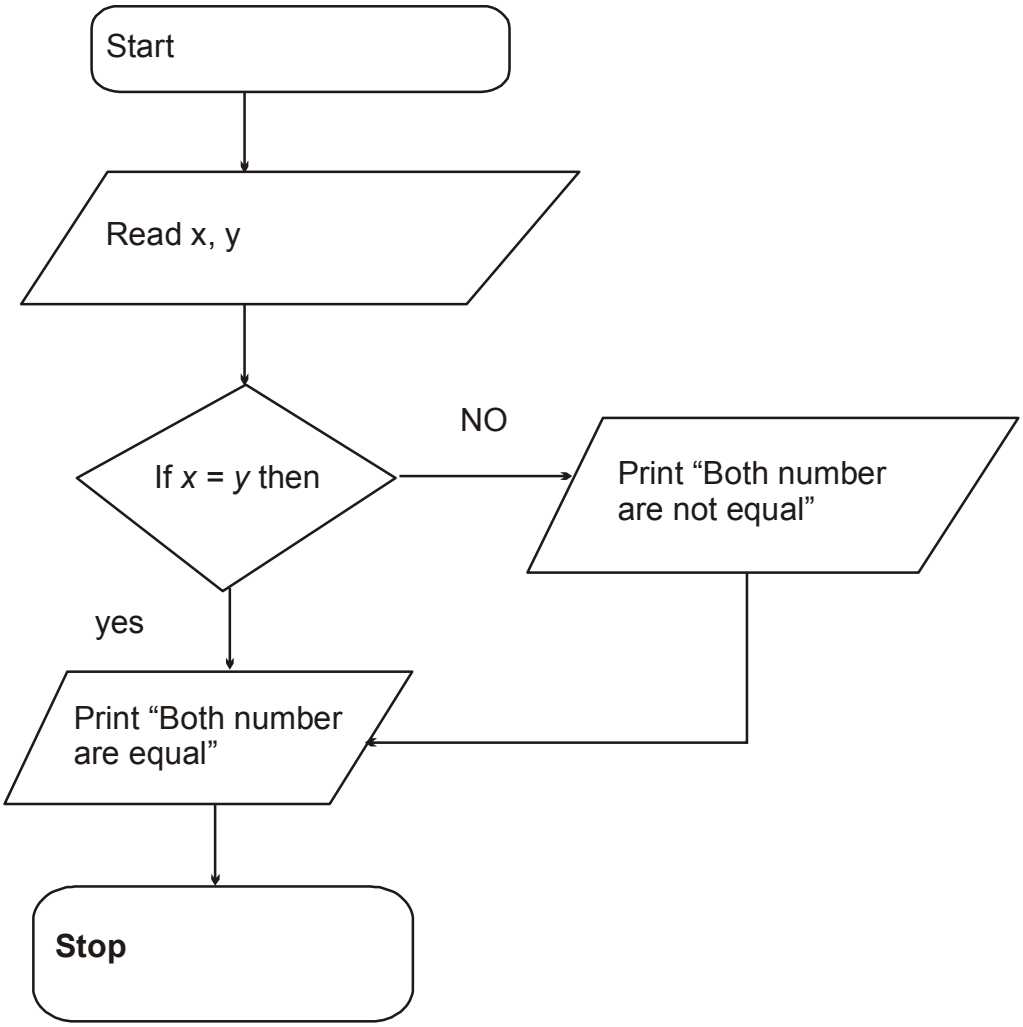
Example 1. 8

The flowchart for the Example 1.1 is shown below :



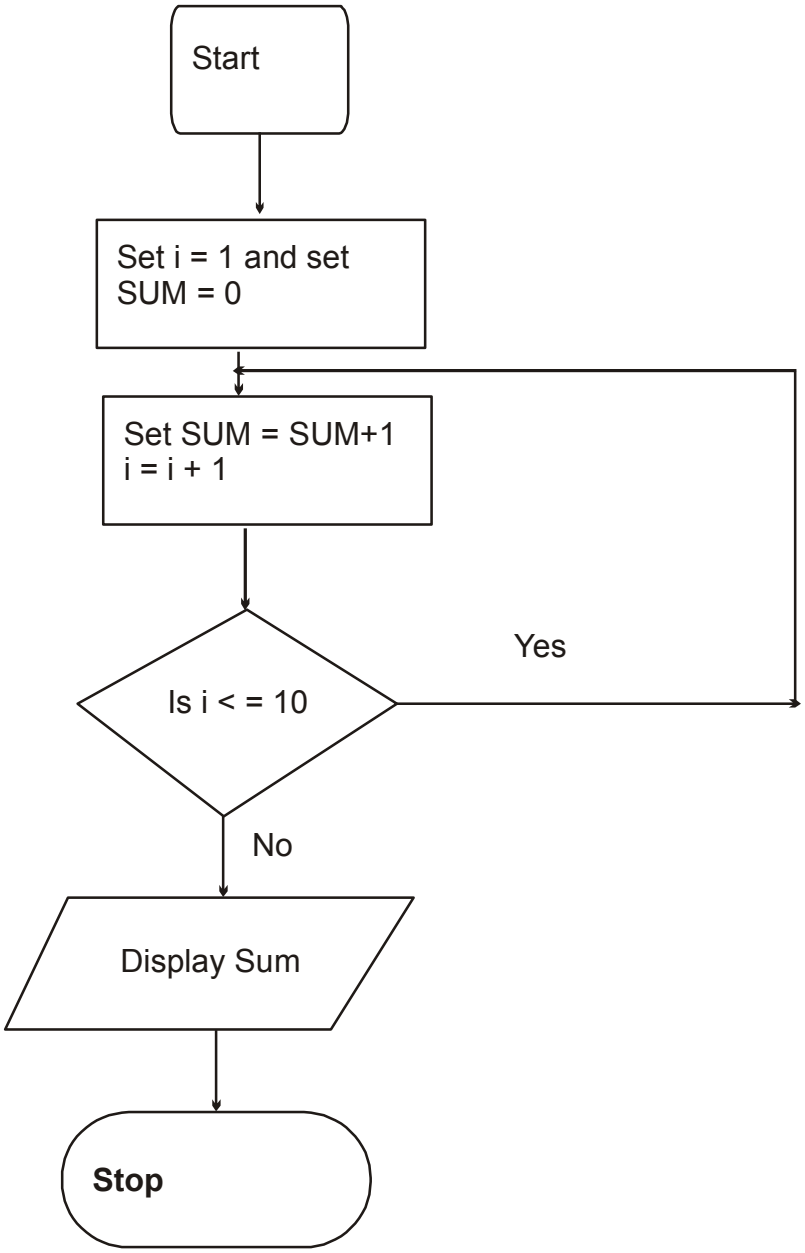
Example 1. 9

The flowchart for the Example 1.2 is shown below :



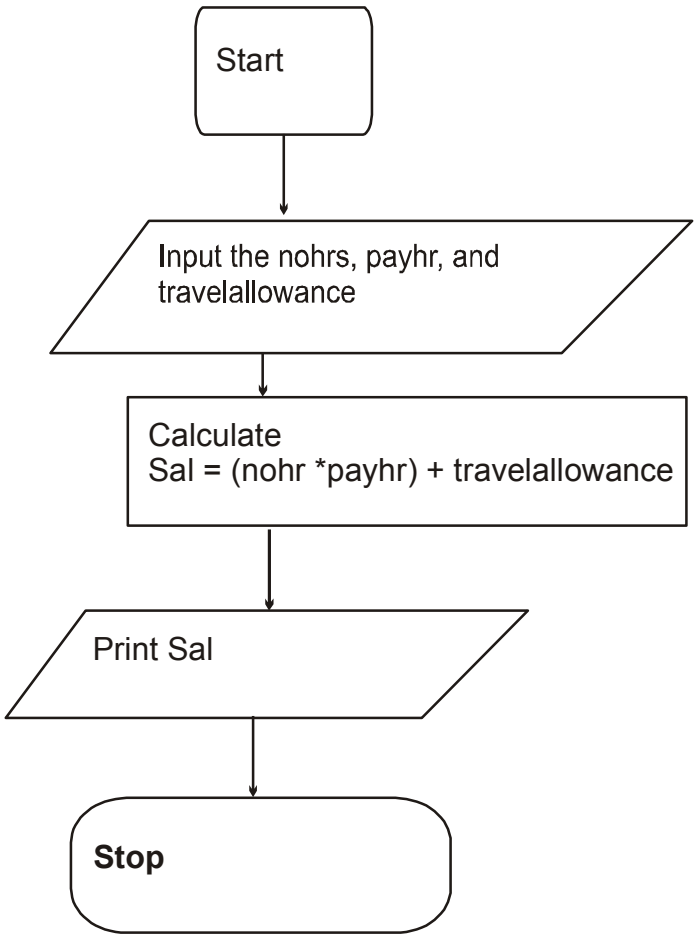
Example 1.10.

The flowchart for the Example 1.3 is shown below :



Example 1.11

Draw a flowchart to calculate the salary of a daily wager.



1. 6 Summary

Programming Languages are used to creare programs that control the behaviour of a system, to express algorithms or as a mode of human computation. To process a problem different problem - solving tools are available that help in finding the solution to the problem in an efficient way. Writing the algorithm and drawing the flowchart for the solution to the stated problem are the major steps which should be followed. Top down design provides the way of handing the logical complexity in computer algorithm. It allows building solutions to problems in a stepwise fashion. We are using C language - a standardized, industrial-strength programming language known for its power and portability as an implementation vechicle for these problem solving techniques using computer.

1. 7 Questions for Exercise

1. Define an algorithum. How is it useful in the context of software development?
2. Draw an algorithum and flowchart to calculate the roots of quadratic equation $Ax^2 + Bx + C = 0$.
3. What is the difference between flowchart and algorithm.
4. Draw an algorithm and flowchart to read the marks of 10 students. If marks is greater than 50, the student passes else the student fails.
5. Write the steps which are suggested to faciliate the problem solving process using computer.
6. Define Flowchart and the basic symbols used to design the flowchart.

1. 8 Suggested Reading

1. How to solve it by Computer, 5th Edition, R G Dromey, PHI, 1992.
2. Introduction to Computer Algorithms, Second Edition, Thomas H.Cormen, MIT press, 2001.
3. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition, Donald E Knuth, July 17,1997.
4. Fundamentals of Algorithmics, Gilles Brassword, Paul Bratley, PHI, 1996.
5. Fundamentals Algorithms, Third Edition, Donald E Knuth, Addison-Wesley, 1997.
6. Programming in C by R Subburaj, Vikas Publishing House, Delhi, Year 2000.

●●●