PROBLEM SOLVING

- Problem solving is the systematic approach to define the problem and creating number of solutions.
- The problem solving process starts with the problem specifications and ends with a Correct program.

- PROBLEM SOLVING TECHNIQUES
- Problem solving technique is a set of techniques that helps in providing logic for solving a problem.

Problem Solving Techniques:

- Problem solving can be expressed in the form of

1. Algorithms.
2. Flowcharts.
3. Pseudo codes.
4. programs

ALGORITHM

- It is defined as a sequence of instructions that describe a method for solving a problem. In other words it is a step by step procedure for solving a problem.

- Properties of Algorithms
  - Should be written in simple English
  - Each and every instruction should be precise and unambiguous.
  - Instructions in an algorithm should not be repeated infinitely.
  - Algorithm should conclude after a finite number of steps.
  - Should have an end point
  - Derived results should be obtained only after the algorithm terminates.

Qualities of a good algorithm

- Time – To execute a program, the computer system takes some amount of time. The lesser is the time required, the better is the algorithm.
- Memory – To execute a program, computer system takes some amount of memory space. The lesser is the memory required, the better is the algorithm.
- Accuracy – Multiple algorithms may provide suitable or correct solutions to a given problem, some of these may provide more accurate results than others, and such algorithms may be suitable.
- Example
    - Write an algorithm to print „Good Morning"
        - Step 1: Start
        - Step 2: Print "Good Morning"
        - Step 3: Stop

BUILDING BLOCKS OF ALGORITHMS (statements, state, control flow, functions)

- Algorithms can be constructed from basic building blocks namely, sequence, selection and iteration.

- Statements: Statement is a single action in a computer.
- In a computer statements might include some of the following actions
  - input data-information given to the program
  - process data-perform operation on a given input
  - output data-processed result
- State: Transition from one process to another process under specified condition with in a time is called state.
- Control flow:The process of executing the individual statements in a given order is called control flow.

- The control can be executed in three ways
- 1. sequence
- 2. selection
- 3. iteration

- Sequence: All the instructions are executed one after another is called sequence execution.
- Example:
- Add two numbers:
    - Step 1: Start
    - Step 2: get a,b
    - Step 3: calculate c=a+b
    - Step 4: Display c
    - Step 5: Stop

- Selection:A selection statement causes the program control to be transferred to a specific part of the program based upon the condition.
- If the conditional test is true, one part of the program will be executed, otherwise it will execute the other part of the program.
- Example
- Write an algorithm to check whether he is eligible to vote?
    - Step 1: Start
    - Step 2: Get age
    - Step 3: if age >= 18 print “Eligible to vote”
    - Step 4: else print “Not eligible to vote”
    - Step 5: Stop

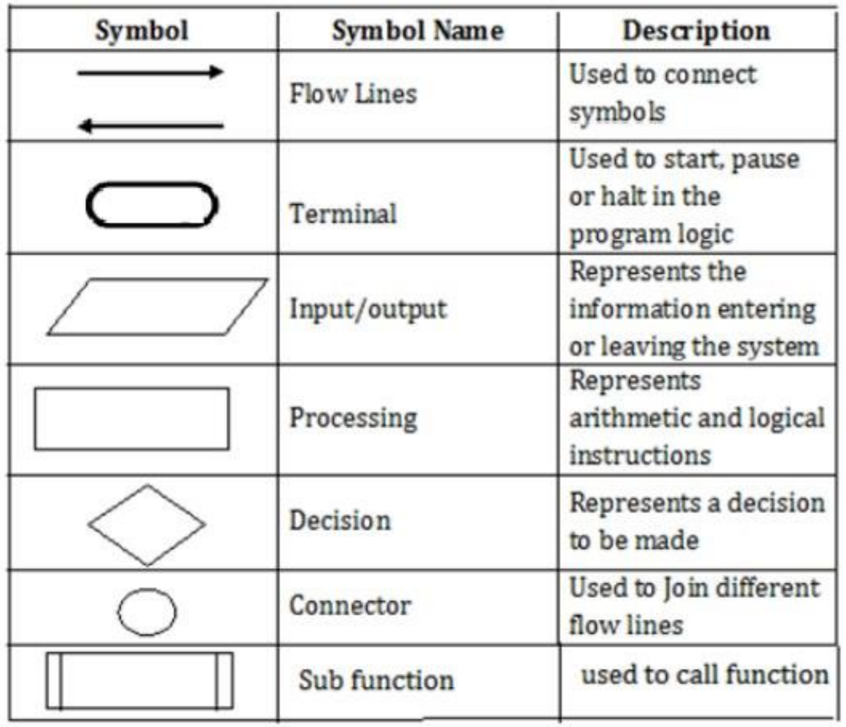
- Iteration: In some programs, certain set of statements are executed again and again based upon conditional test. i.e. executed more than one time. This type of execution is called looping or iteration.
- Example
- Write an algorithm to print all natural numbers up to n
    - Step 1: Start
    - Step 2: get n value.
    - Step 3: initialize i=1
    - Step 4: if (i<=n) go to step 5 else go to step 7
    - Step 5: Print i value and increment i value by 1
    - Step 6: go to step 4
    - Step 7: Stop

- Functions: Function is a sub program which consists of block of code(set of instructions) that performs a particular task.
- For complex problems, the problem is been divided into smaller and simpler tasks during algorithm design.
- Benefits of Using Functions
- Reduction in line of code
  - code reuse
  - Better readability
  - Information hiding
  - Easy to debug and test
  - Improved maintainability

- Example:
- Algorithm for addition of two numbers using function
- Main function()
- Step 1: Start
- Step 2: Call the function add()
- Step 3: Stop
-
- sub function add()
- Step 1: Function start
- Step 2: Get a, b Values
- Step 3: add c=a+b
- Step 4: Print c
- Step 5: Return

FLOW CHART

- Flow chart is defined as graphical representation of the logic for problem solving.
- The purpose of flowchart is making the logic of the program clear in a visual representation.



Symbol	Symbol Name	Description
→ ←	Flow Lines	Used to connect symbols
⬭	Terminal	Used to 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
▯	Sub function	used to call function

Rules for drawing a flowchart

- 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.

Advantages and Disadvtages of flowchart:

- 1. Communication: - Flowcharts are better way of communicating the logic of a system to all concerned.
- 2. Effective analysis: - With the help of flowchart, problem can be analyzed in more effective way.
- 3. Proper documentation: - Program flowcharts serve as a good program documentation, which is needed for various purposes.
- 4. Efficient Coding: - The flowcharts act as a guide or blueprint during the systems analysis and program development phase.
- 5. Proper Debugging: - The flowchart helps in debugging process.
- 6. Efficient Program Maintenance: - The maintenance of operating program becomes easy with the help of flowchart. It helps the programmer to put efforts more efficiently on that part.

- 1. Complex logic: - Sometimes, the program logic is quite complicated. In that case, flowchart becomes complex and clumsy.
- 2. Alterations and Modifications: - If alterations are required the flowchart may require re-drawing completely.
- 3. Reproduction: - As the flowchart symbols cannot be typed, reproduction of flowchart becomes a problem.
- 4. Cost: For large application the time and cost of flowchart drawing becomes costly.

Algorithm	**Flowchart**	**Pseudo code**
An algorithm is a sequence of instructions used to solve a problem	It is a graphical representation of algorithm	It is a language representation of algorithm.
User needs knowledge to write algorithm.	not need knowledge of program to draw or understand flowchart	Not need knowledge of program language to understand or write a pseudo code.

PROGRAMMING LANGUAGE

- A programming language is a set of symbols and rules for instructing a computer to perform specific tasks. The programmers have to follow all the specified rules before writing program using programming language. The user has to communicate with the computer using language which it can understand.
- Types of programming language
- 1. Machine language
- 2. Assembly language
- 3. High level language

Machine language:

- The computer can understand only machine language which uses 0’s and 1’s. In machine language the different instructions are formed by taking different combinations of 0’s and 1’s.
- Advantages:
    - Translation free:Machine language is the only language which the computer understands. For executing any program written in any programming language, the conversion to machine language is necessary. The program written in machine language can be executed directly on computer. In this case any conversion process is not required.
    - High speed:The machine language program is translation free. Since the conversion time is saved, the execution of machine language program is extremely fast.
- Disadvantage:
    - It is hard to find errors in a program written in the machine language.
    - Writing program in machine language is a time consuming process.
    - Machine dependent: According to architecture used, the computer differs from each other. So machine language differs from computer to computer. So a program developed for a particular type of computer may not run on other type of computer.
-

Assembly language:

- To overcome the issues in programming language and make the programming process easier, an assembly language is developed which is logically equivalent to machine language but it is easier for people to read, write and understand.
- Assembly language is symbolic representation of machine language. Assembly languages are symbolic programming language that uses symbolic notation to represent machine language instructions. They are called low level language because they are so closely related to the machines.
- Assembler: Assembler is the program which translates assembly language instruction in to a machine language.
    - Easy to understand and use.
    - It is easy to locate and correct errors.
-

- Disadvantage
  - Machine dependent:The assembly language program which can be executed on the machine depends on the architecture of that computer.
  - Hard to learn:It is machine dependent, so the programmer should have the hardware knowledge to create applications using assembly language.
  - Less efficient:Execution time of assembly language program is more than machine language program.
  - Because assembler is needed to convert from assembly language to machine language.

High level language

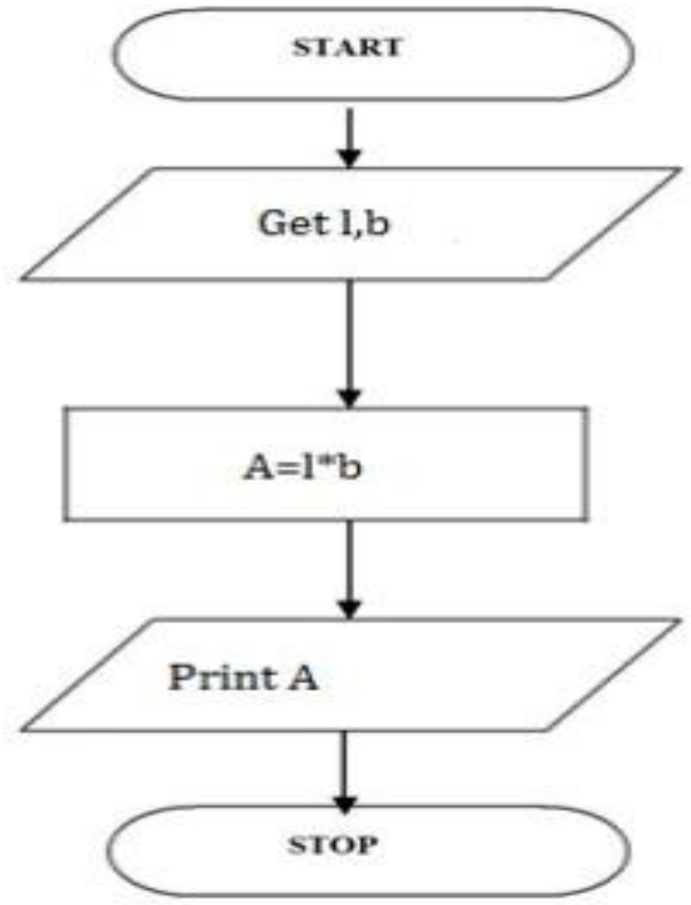
- High level language contains English words and symbols. The specified rules are to be followed while writing program in high level language. The interpreter or compilers are used for converting these programs in to machine readable form.
- Translating high level language to machine language
- The programs that translate high level language in to machine language are called interpreter or compiler.
- Compiler:A compiler is a program which translates the source code written in a high level language in to object code which is in machine language program. Compiler reads the whole program written in high level language and translates it to machine language. If any error is found it display error message on the screen.
- Interpreter: Interpreter translates the high level language program in line by line manner. The interpreter translates a high level language statement in a source program to a machine code and executes it immediately before translating the next statement. When an error is found the execution of the program is halted and error message is displayed on the screen.

- Advantages
  - Readability:High level language is closer to natural language so they are easier to learn and understand
  - Machine independent:High level language program have the advantage of being portable between machines.
  - Easy debugging:Easy to find and correct error in high level language
- Disadvantages
  - Less efficient:The translation process increases the execution time of the program. Programs in high level language require more memory and take more execution time to execute.

- They are divided into following categories:
-
- 1. Interpreted programming languages
- 2. Functional programming languages
- 3. Compiled programming languages
- 4. Procedural programming languages
- 5. Scripting programming language
- 6. Markup programming language
- 7. Concurrent programming language
- 8. Object oriented programming language

- Write an algorithm to find area of a rectangle
- Step 1: Start
- Step 2: get l,b values
- Step 3: Calculate A=l*b
- Step 4: Display A
- Step 5: Stop

- BEGIN
- READ l,b
- CALCULATE A=l*b
- DISPLAY A
- END



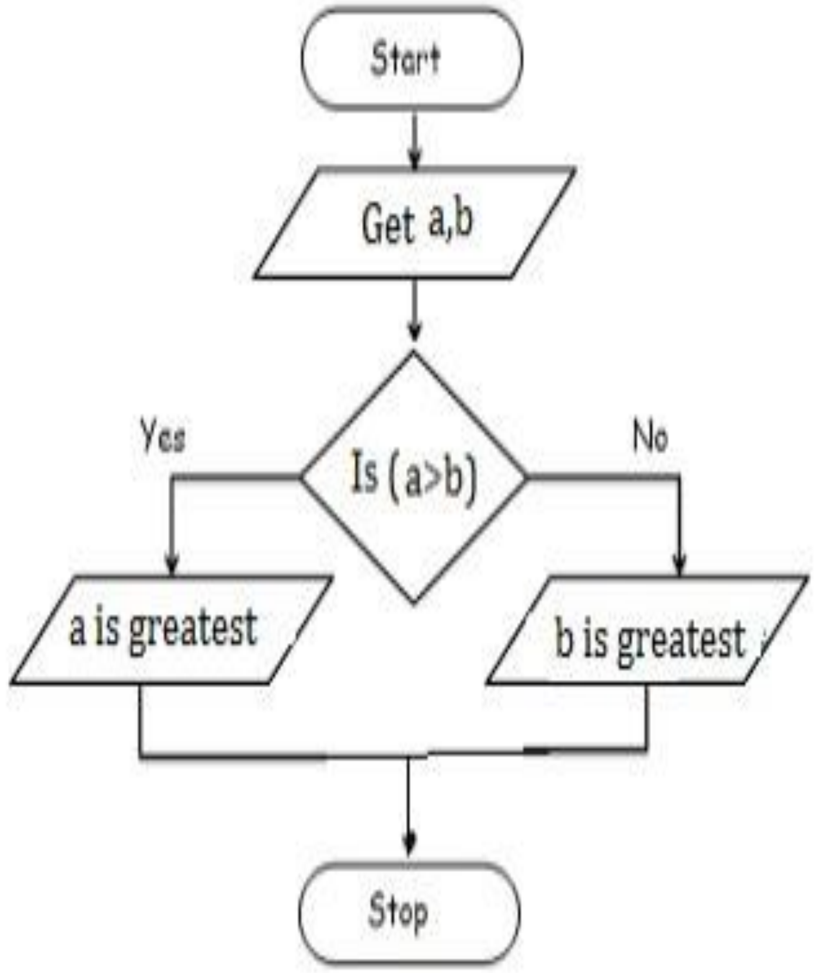
- Write an algorithm for Calculating area and circumference of circle

- Step 1: Start
- Step 2: get r value
- Step 3: Calculate A=3.14*r*r
- Step 4: Calculate C=2.3.14*r
- Step 5: Display A,C
- Step 6: Stop

- Write an algorithm for Calculating simple interest

- Step 1: Start
- Step 2: get P, n, r value
- Step3:Calculate
- SI=(p*n*r)/100
- Step 4: Display S
- Step 5: Stop

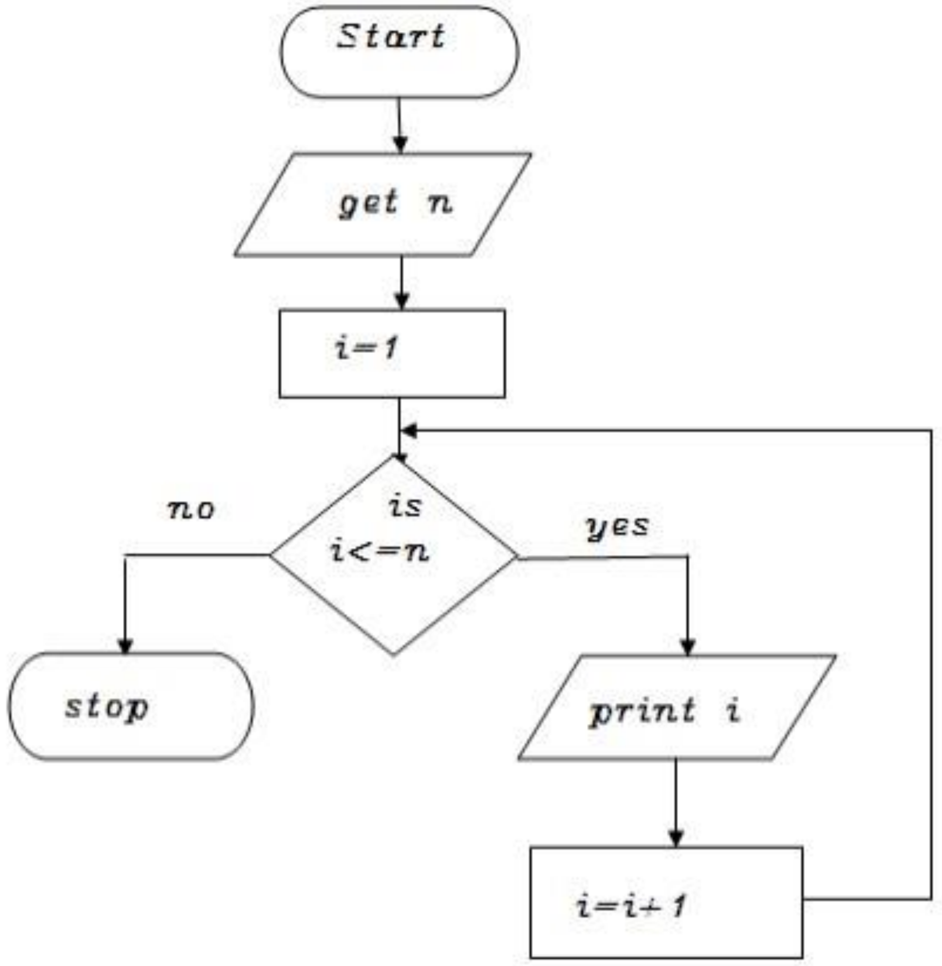
- To check greatest of two numbers
-
- Step 1: Start
- Step 2: get a,b value
- Step 3: check if(a>b) print a is greater
- Step 4: else b is greater
- Step 5: Stop



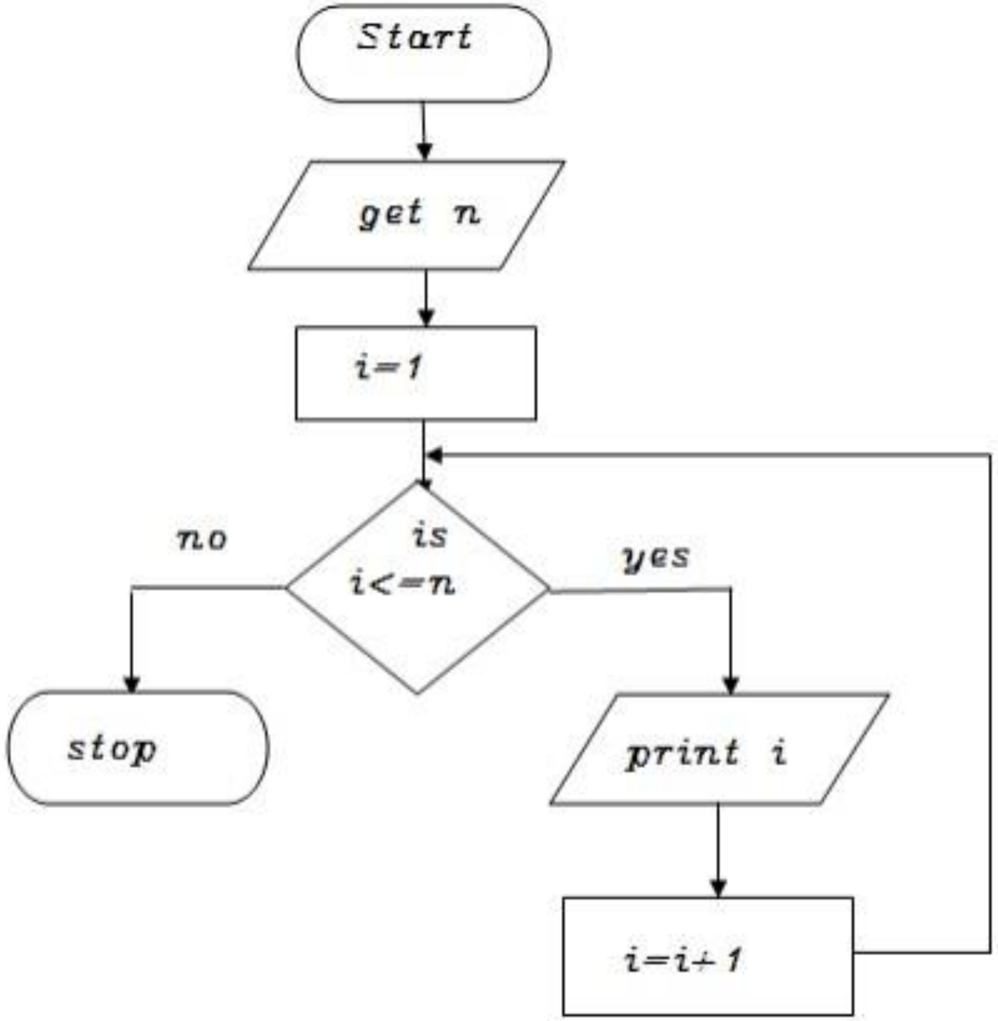
- To check positive or negative number
-
- Step 1: Start
- Step 2: get num
- Step 3: check if(num>0) print a is positive
- Step 4: else num is negative
- Step 5: Stop

- Leap year?
- even/odd?
- greatest of 3 numbers?
- all natural numbers

- Write an algorithm to print all natural numbers up to n
-
- Step 1: Start
- Step 2: get n value.
- Step 3: initialize i=1
- Step 4: if (i<=n) go to step 5 else go to step 8
- Step 5: Print i value
- step 6 : increment i value by 1
- Step 7: go to step 4
- Step 8: Stop
-



- Write an algorithm to print n odd numbers
-
- Step 1: start
- step 2: get n value
- step 3: set initial value i=1
- step 4: check if(i<=n) goto step 5 else goto step 8
- step 5: print i value
- step 6: increment i value by 2
- step 7: goto step 4
- step 8: stop



- factorial ?
- n even numbers