Algorithms and Flow chart

ALGORITHMS AND FLOWCHARTS

- A typical programming task can be divided into two phases:
- **Problem solving phase**
  - □ produce an ordered sequence of steps that describe solution of problem
  - □ this sequence of steps is called an **algorithm**
- **Implementation phase**
  - □ implement the program in some programming language

Steps in Problem Solving

- First produce a general algorithm (one can use *pseudocode*)
- Refine the algorithm successively to get step by step detailed **algorithm** that is very close to a computer language.
- **Pseudocode** is an artificial and informal language that helps programmers develop algorithms. Pseudocode is very similar to everyday English.

Algorithm

- Algorithm can be defined as: "A sequence of activities to be processed for getting desired output from a given input."

- Then we can say that:

    1. Getting specified output is essential after algorithm is executed.

    2. One will get output only if algorithm stops after finite time.

    3. Activities in an algorithm to be clearly defined in other words for it to be unambiguous.

Types of Algorithm

- The algorithm and flowchart, classification to the three types of *control structures*. They are:

    1. Sequence
    2. Branching (Selection)
    3. Loop (Repetition)

Sequence

- The sequence is exemplified by sequence of statements place one after the other – the one above or before another gets executed first. In flowcharts, sequence of statements is usually contained in the rectangular process box.

Branching

- The *branch* refers to a **binary decision** based on some condition. If the condition is true, one of the two branches is explored; if the condition is false, the other alternative is taken. This is usually represented by the **'if-then'** construct in pseudo-codes and programs.

- In **flowcharts**, this is represented by the **diamond-shaped decision** box. This structure is also known as the *selection* structure.

Looping

- The *loop* allows a statement or a **sequence of statements to be repeatedly executed** based on **some loop condition.**

- It is represented by the '**while' and 'for**' constructs in most programming languages, for unbounded loops and bounded loops respectively. (Unbounded loops refer to those whose number of iterations depends on the eventuality that the termination condition is satisfied; bounded loops refer to those whose number of iterations is known before-hand.)

- In the flowcharts, a **back arrow hints the presence of a loop**. **A trip around the loop is known as iteration**. You must ensure that the condition for the termination of the looping must be satisfied after some finite number of iterations, otherwise it ends up as an infinite loop, a common mistake made by inexperienced programmers. The loop is also known as the *repetition* structure.

Properties of Algorithm

Properties are:

1) **Finiteness:** An algorithm must always **terminate after a finite number of steps**. It means after every step one reach closer to solution of the problem and after a finite number of steps algorithm reaches to an end point.

2) **Definiteness:** Each **step of an algorithm must be precisely defined**. It is done by well thought actions to be performed at each step of the algorithm. Also the actions are defined unambiguously for each activity in the algorithm.

3) **Input:** Any operation you perform need some **beginning value/quantities** associated with different activities in the operation. So the value/quantities are given to the algorithm before it begins.

Properties of Algorithm

4) **Output:** One always **expects output/result** (expected value/quantities) in terms of output from an algorithm. The result may be obtained at different stages of the algorithm. If some result is from the intermediate stage of the operation then it is known as intermediate result and result obtained at the end of algorithm is known as end result. The output is expected value/quantities always have a specified relation to the inputs.

5) **Effectiveness:** Algorithms to be **developed/written using basic operations**. Actually operations should be basic, so that even they can in principle be done exactly and in a finite amount of time by a person, by using paper and pencil only.

Pseudocode & Algorithm

- **Example 1:** Write an algorithm to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

Pseudocode & Algorithm

Pseudocode:

- *Input a set of 4 marks*
- *Calculate their average by summing and dividing by 4*
- *if average is below 50*

    Print "FAIL"

else

    Print "PASS"

Pseudocode & Algorithm

- Detailed Algorithm
-      Step 1:   Input M1,M2,M3,M4
       Step 2:   GRADE $\leftarrow$ (M1+M2+M3+M4)/4
       Step 3:   if (GRADE $<$ 50) then
                         Print "FAIL"
         else
                         Print "PASS"
         endif

The Flowchart

- (Dictionary) A schematic representation of a sequence of operations, as in a manufacturing process or computer program.
- (Technical) A graphical representation of the sequence of operations in an information system or program. Information system flowcharts show how data flows from source documents through the computer to final distribution to users. Program flowcharts show the sequence of instructions in a single program or subroutine. Different symbols are used to draw each type of flowchart.

The Flowchart

A Flowchart

- shows logic of an algorithm
- emphasizes individual steps and their interconnections
- e.g. control flow from one action to the next

Flow Chart Symbols

Symbol	Name	Function
▭	Process	Indicates any type of internal operation inside the Processor or Memory
▱	input/output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
◇	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
◯	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse flow.
▯|▯	Predefined Process	Used to invoke a subroutine or an Interrupt program.
⬭	Terminal	Indicates the starting or ending of the program, process, or interrupt program
↑↓ ←→	Flow Lines	Shows direction of flow.

General Rules for flowcharting

1. All boxes of the flowchart are connected with Arrows. (Not lines)

2. Flowchart symbols have an entry point on the top of the symbol with no other entry points. The exit point for all flowchart symbols is on the bottom except for the Decision symbol.

3. The Decision symbol has two exit points; these can be on the sides or the bottom and one side.

4. Generally a flowchart will flow from top to bottom. However, an upward flow can be shown as long as it does not exceed 3 symbols.

5. Connectors are used to connect breaks in the flowchart. Examples are:

   • From one page to another page.

   • From the bottom of the page to the top of the same page.

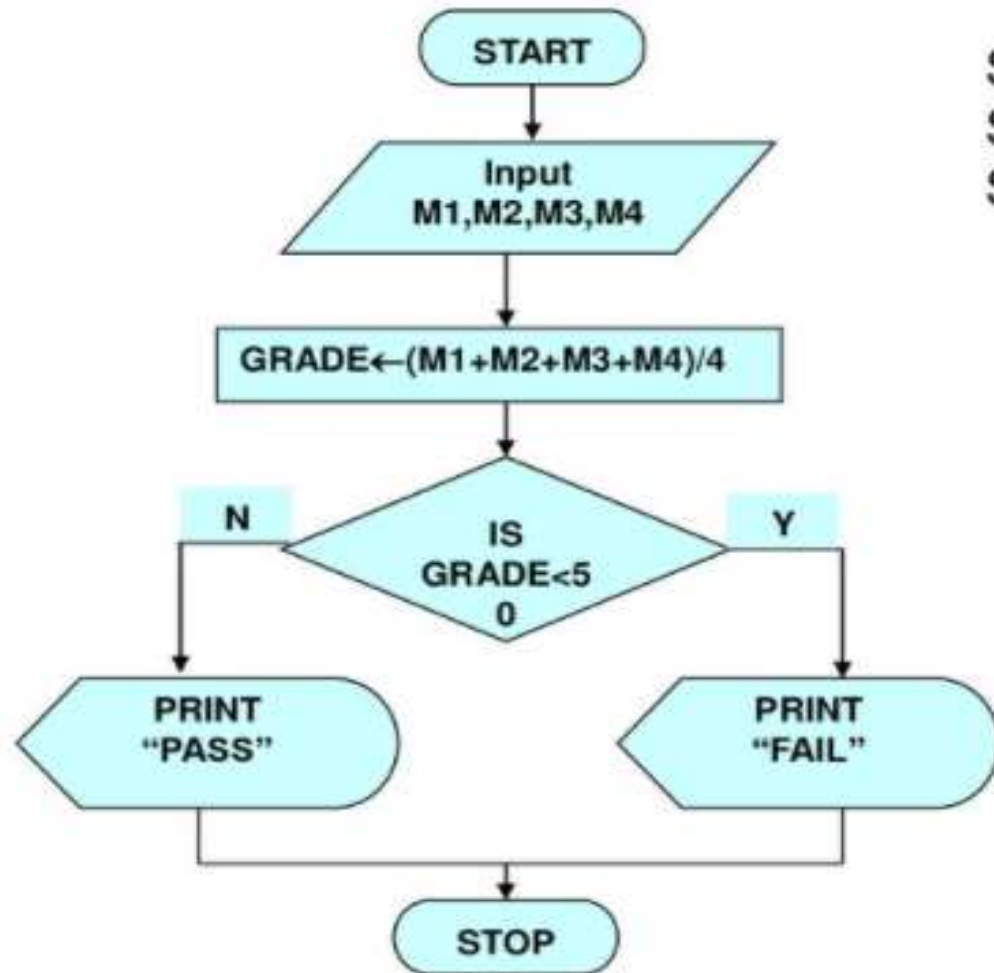   • An upward flow of more then 3 symbols

General Rules for flowcharting

6. Subroutines and Interrupt programs have their own and independent flowcharts.

7. All flow charts start with a Terminal or Predefined Process (for interrupt programs or subroutines) symbol.

8. All flowcharts end with a terminal or a contentious loop.

Example

Step 1:   Input M1,M2,M3,M4

Step 2:   GRADE ← (M1+M2+M3+M4)/4

Step 3:   if (GRADE <50) then

                            Print "FAIL"

    else

                            Print "PASS"

    endif

Example 2

- Write an algorithm and draw a flowchart to convert the length in feet to centimeter.
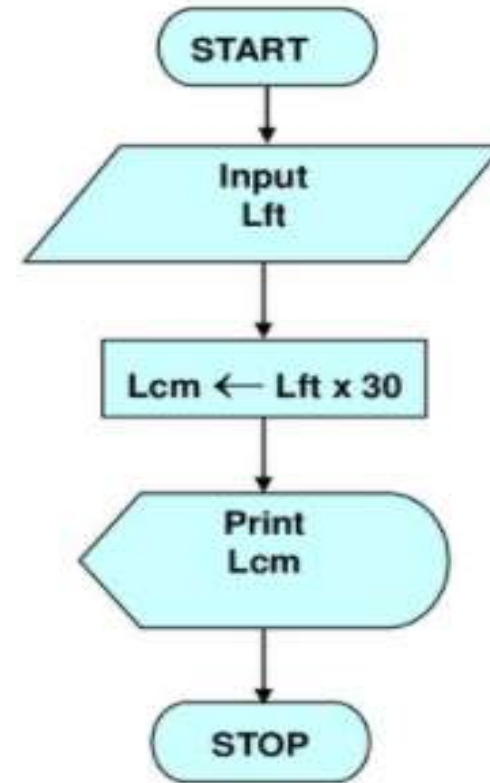
Pseudocode:

- *Input the length in feet (Lft)*
- *Calculate the length in cm (Lcm) by multiplying LFT with 30*
- *Print length in cm (LCM)*

Example 2

Algorithm

- Step 1: Input Lft
- Step 2: Lcm ← Lft x 30
- Step 3: Print Lcm

Flowchart

Example 3

Write an algorithm and draw a flowchart that will read the two sides of a rectangle and calculate its area.
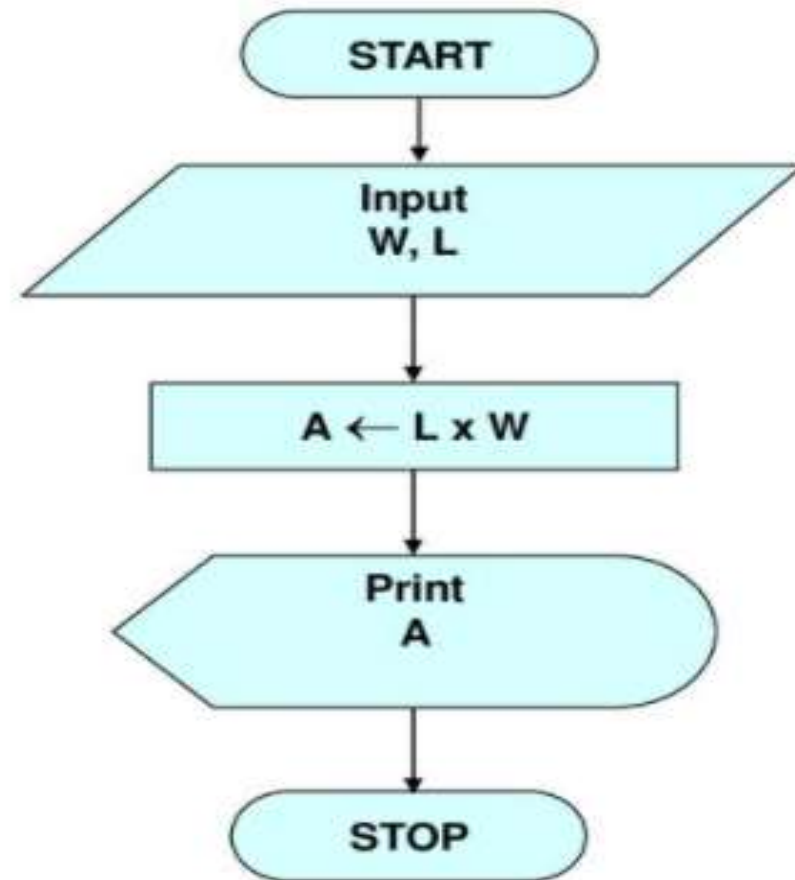
Pseudocode

- *Input the width (W) and Length (L) of a rectangle*
- *Calculate the area (A) by multiplying L with W*
- *Print A*

Example 3

Algorithm
- Step 1: Input W,L
- Step 2: $A \leftarrow L \times W$
- Step 3: Print A

Example 4

- Write an algorithm and draw a flowchart that will calculate the roots of a quadratic equation
$$ax^2 + bx + c = 0$$

- Hint: $\mathbf{d} = \text{sqrt} (b^2 - 4ac)$, and the roots are:
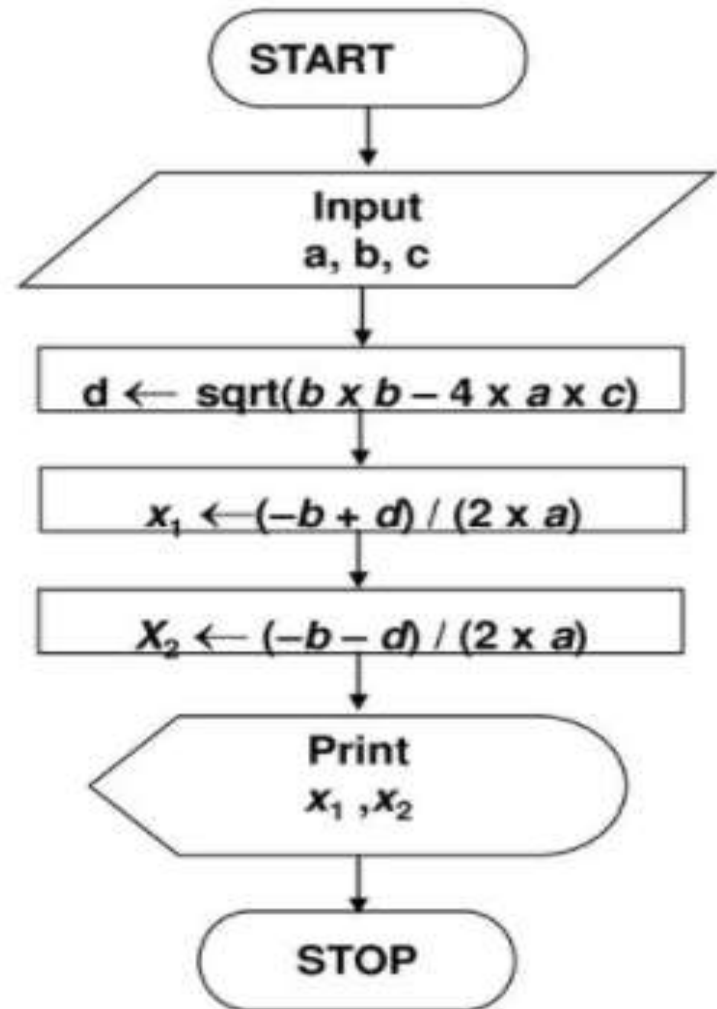$x1 = (-b + d)/2a$ and $x2 = (-b - d)/2a$

Example 4

Pseudocode:
- *Input the coefficients (a, b, c) of the quadratic equation*
- *Calculate* **d**
- *Calculate* **x1**
- *Calculate* **x2**
- *Print x1 and x2*

Example 4

Algorithm:

- Step 1: Input a, b, c
- Step 2: $d \leftarrow$ sqrt ($b \times b - 4 \times a \times c$)
- Step 3: $x1 \leftarrow (-b + d) / (2 \times a)$
- Step 4: $x2 \leftarrow (-b - d) / (2 \times a)$
- Step 5: Print $x1$, $x2$

START

Input
a, b, c

$d \leftarrow$ sqrt($b \times b - 4 \times a \times c$)

$x_1 \leftarrow (-b + d) / (2 \times a)$

$X_2 \leftarrow (-b - d) / (2 \times a)$

Print
X_1, X_2

STOP

DECISION STRUCTURES

- The expression A>B is a logical expression
- *it describes a **condition** we want to test*
- ***if A>B is true (if A is greater than B)** we take the action on left*
- print the value of A
- ***if A>B is false (if A is not greater than B)** we take the action on right*
- print the value of B

DECISION STRUCTURES
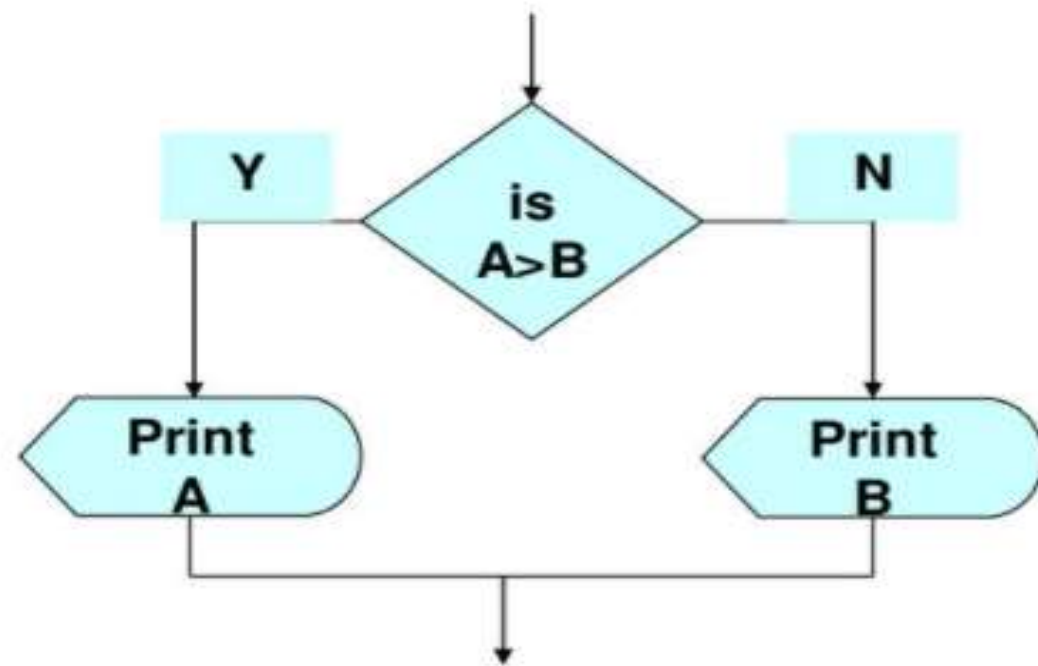
IF–THEN–ELSE STRUCTURE

■ The structure is as follows
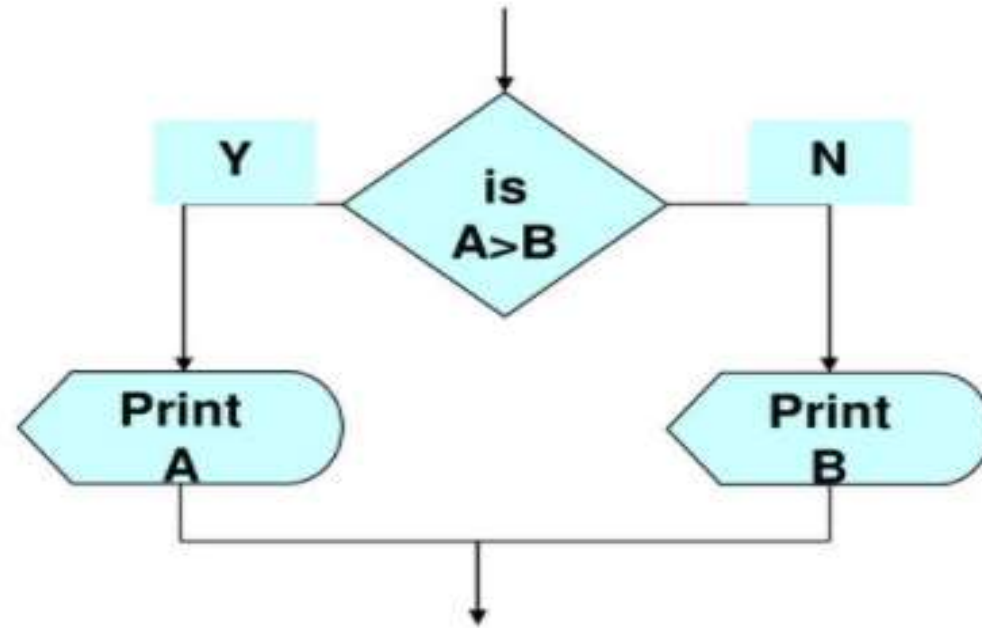
If condition then

  true alternative

 else

  false alternative

endif

IF–THEN–ELSE STRUCTURE

■ The algorithm for the flowchart is as follows:

If A>B then
*   print A*
else
*   print B*
endif

Relational Operators

Relational Operators	
Operator	**Description**
>	Greater than
<	Less than
=	Equal to
≥	Greater than or equal to
≤	Less than or equal to
≠	Not equal to

Example 5

- Write an algorithm that reads two values, determines the largest value and prints the largest value with an identifying message.

ALGORITHM

Step 1:   *Input* VALUE1, VALUE2

Step 2:   *if (*VALUE1 > VALUE2*) then*

                       MAX ← VALUE1

         else
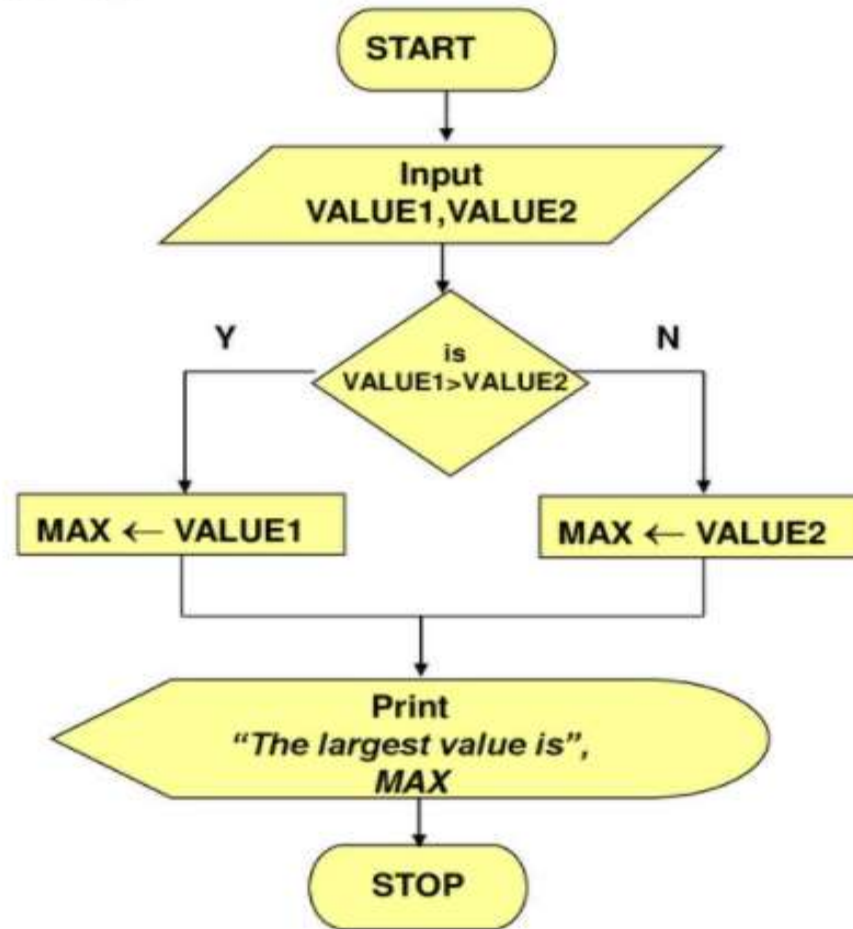
                       MAX ← VALUE2

         endif

Step 3:   *Print "The largest value is", MAX*

Example 5

NESTED IFS

- One of the alternatives within an IF–THEN–ELSE statement
  - may involve further IF–THEN–ELSE statement

Example 6

- Write an algorithm that reads **three** numbers and prints the value of the largest number.

Example 6

Step 1: *Input* **N1, N2, N3**
Step 2: *if (N1>N2) then*
       if (N1>N3) then
          MAX ← N1   **[N1>N2, N1>N3]**
      else
          MAX ← N3   **[N3>N1>N2]**
     endif
   else
     if (N2>N3) then
          MAX ← N2   **[N2>N1, N2>N3]**
     else
          MAX ← N3   **[N3>N2>N1]**
     endif
   endif
Step 3: *Print "The largest number is", MAX*

Example 6

- **Flowchart: Draw the flowchart of the above Algorithm.**

Example 7

- Write and algorithm and draw a flowchart to

a) read an employee name (NAME), overtime hours worked (OVERTIME), hours absent (ABSENT) and

b) determine the bonus payment (PAYMENT).