Program Designing techniques

- ***Pseudocode***
- ***Algorithm***
- ***Flowchart***

Designing techniques

- 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.

Pseudocode & Algorithm

- **Example 1:** Write a pseudocode and an algorithm to convert the length in feet to inches.

Example 2

Pseudocode:

- Input the *length in feet*
- Calculate the *length in inches* by multiplying *length in feet* with 12
- Print *length in inches.*

Example 2

Algorithm

- Step 1: Input L_ft
- Step 2: L_inches ← L_ft x 12
- Step 3: Print L_inches

The Flowchart

- A schematic representation of a sequence of operations, as in a manufacturing process or computer program.
- It is a graphic representation of how a process works, showing, at a minimum, the sequence of steps.
- A flowchart consists of a sequence of instructions linked together by arrows to show the order in which the instructions must be carried out.

Cont…

Each instruction is put into a box. The boxes are different shapes depending upon what the instruction is.

Different symbols are used to draw each type of flowchart.

Cont…

A Flowchart

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

Flowchart Symbols

Basic

Example 2

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

Pseudocode:

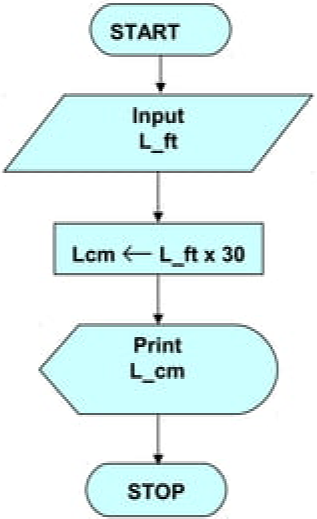
- *Input the length in feet (Lft)*
- *Calculate the length in cm (Lcm) by multiplying LFT with 30*
- *Print length in cm (LCM)*

Example 2

Algorithm

- Step 1: Input L_ft
- Step 2: Lcm $\leftarrow$ L_ft x 30
- Step 3: Print L_cm

Flowchart



Example 3

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

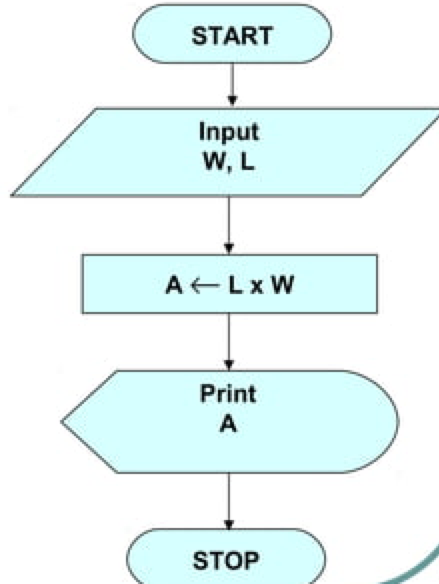
Pseudocode

- *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: **d** = sqrt ($b^2 - 4ac$), and the roots are: ***x*1** = (–*b* + *d*)/2*a* and ***x*2** = (–*b* – *d*)/2*a*

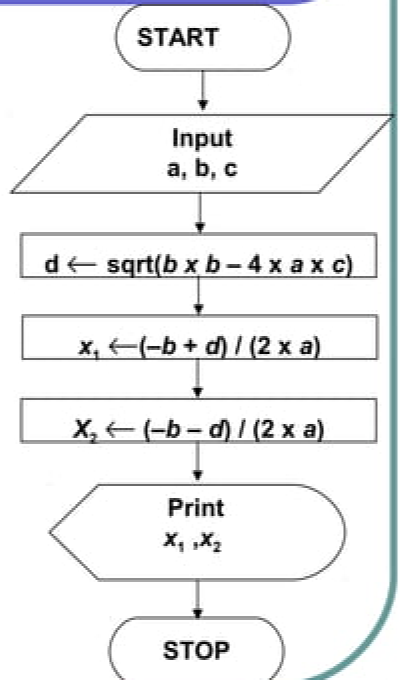
Example 4

Pseudocode:

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

Example 4

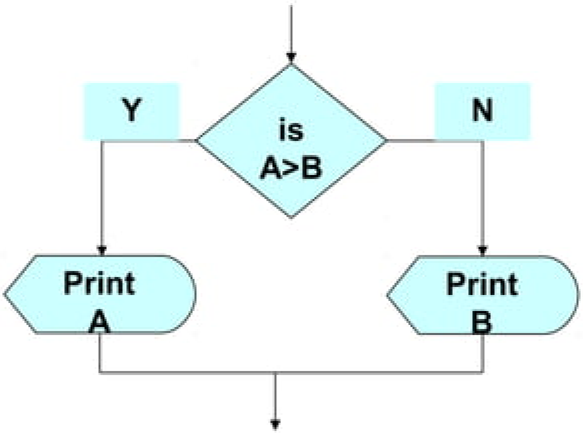
- **Algorithm:**
- Step 1: Input a, b, c
- Step 2: $d \leftarrow \text{sqrt}\,(b \times b - 4 \times a \times c)$
- Step 3: $x1 \leftarrow (-b + d) / (2 \times a)$
- Step 4: $x2 \leftarrow (-b - d) / (2 \times a)$
- Step 5: Print $x1$, $x2$



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

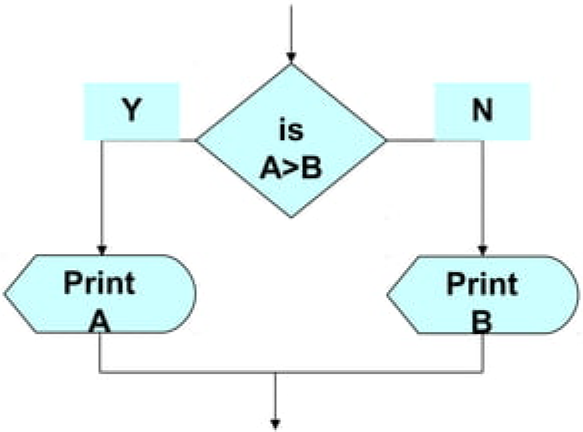
else

false alternative

endif

IF–THEN–ELSE STRUCTURE

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

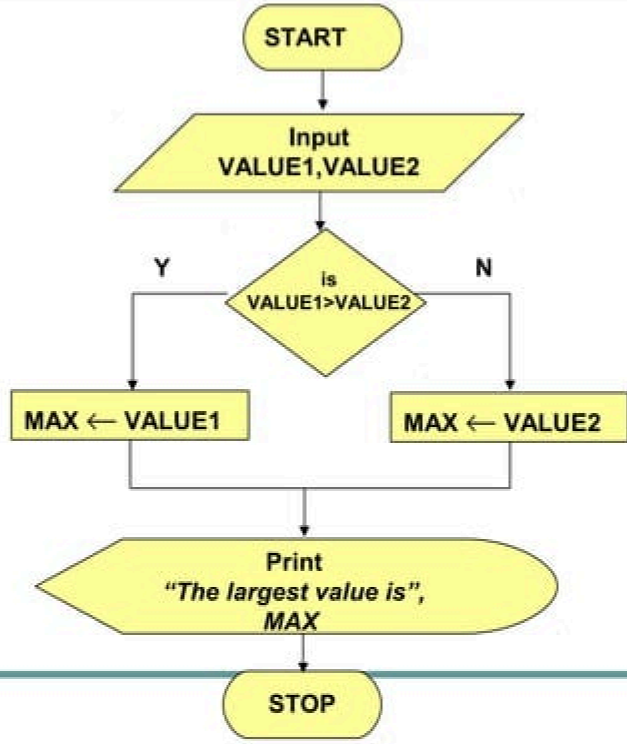
else

MAX ← VALUE2

endif

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

Example 5


START
Input
VALUE1,VALUE2
Y
is
VALUE1>VALUE2
N
MAX ← VALUE1
MAX ← VALUE2
Print
"The largest value is",
MAX
STOP

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.

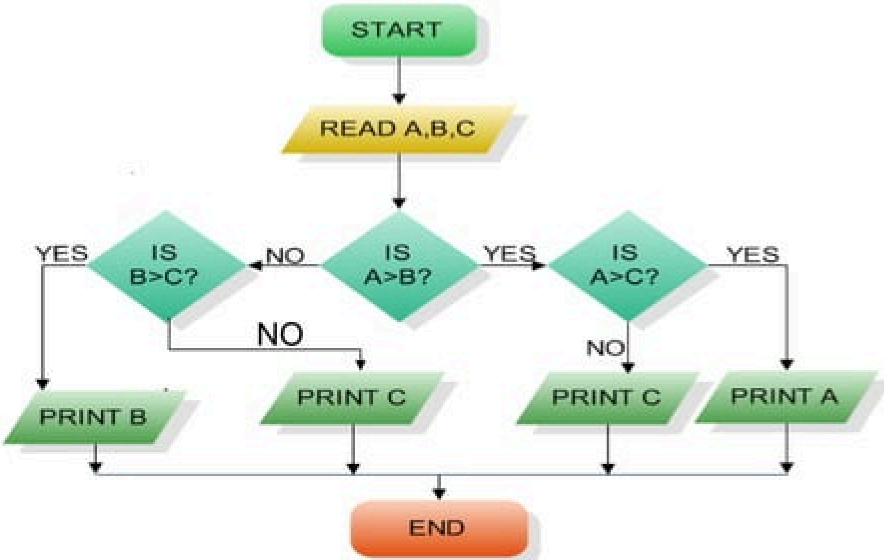
Example 6

```
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.**

Flowchart



START
READ A,B,C
IS A>B?
NO
YES
IS B>C?
YES
NO
IS A>C?
YES
NO
PRINT B
PRINT C
PRINT C
PRINT A
END

Example 7

- Write and algorithm and draw a flowchart to read an employee name (NAME), overtime hours worked (OVERTIME), hours absent (ABSENT) and determine the bonus payment (PAYMENT).

Example 7

Bonus Schedule

OVERTIME – (2/3)*ABSENT	Bonus Paid
>40 hours	\$50
>30 but ≤ 40 hours	\$40
>20 but ≤ 30 hours	\$30
>10 but ≤ 20 hours	\$20
≤ 10 hours	\$10

Algorithm

Step 1: *Input* NAME,OVERTIME,ABSENT
Step 2: *if* (OVERTIME–(2/3)*ABSENT > 40) *then*
PAYMENT $\leftarrow$ 50
*else if (*OVERTIME–(2/3)*ABSENT > 30 &&
OVERTIME–(2/3)*ABSENT<= 40) *then*
PAYMENT $\leftarrow$ 40
*else if (*OVERTIME–(2/3)*ABSENT > 20 &&
OVERTIME–(2/3)*ABSENT<= 30) *then*
PAYMENT $\leftarrow$ 30

else if (OVERTIME–(2/3)*ABSENT > 10 &&
OVERTIME–(2/3)*ABSENT<= 20) *then*

PAYMENT ←20

else

PAYMENT ← 10

endif

Step 3: *Print* “Bonus for”, NAME “is $”, PAYMENT

Example 7

- **Flowchart: Draw the flowchart of the above algorithm?**

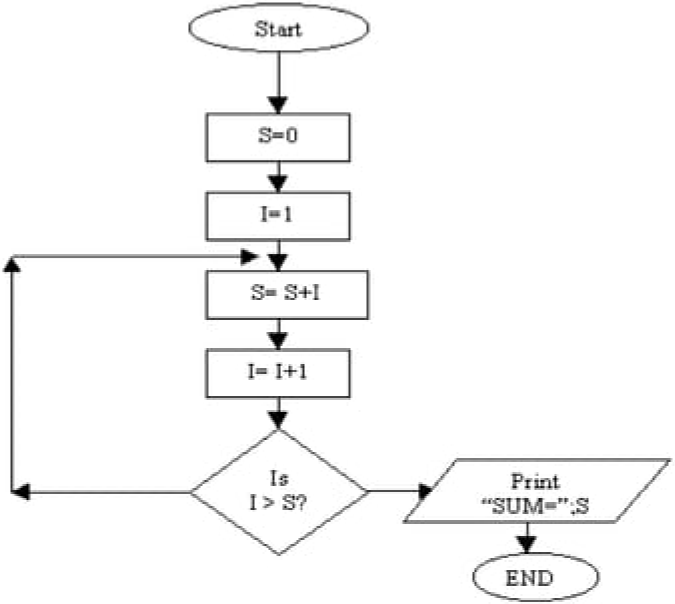
LOOPS

- Computers are particularly well suited to applications in which operations are repeated many times.
- If the same task is repeated over and over again a loop can be used to reduce program size and complexity

Example 8:

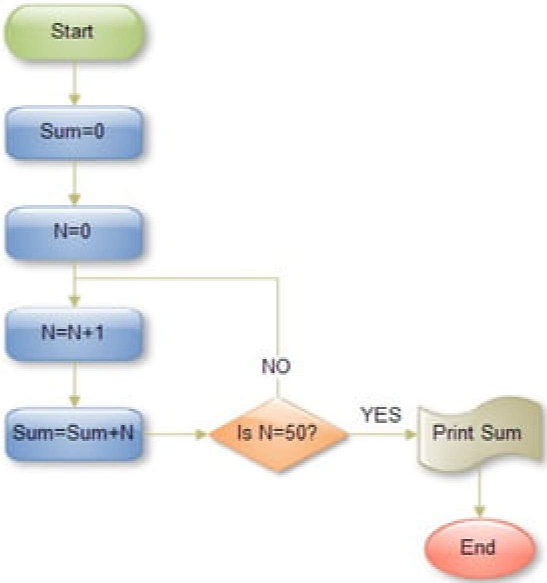
Flowchart for finding the sum of first five natural numbers

(i.e. 1,2,;



Example 9

Flowchart to find the sum of first 50 natural numbers.



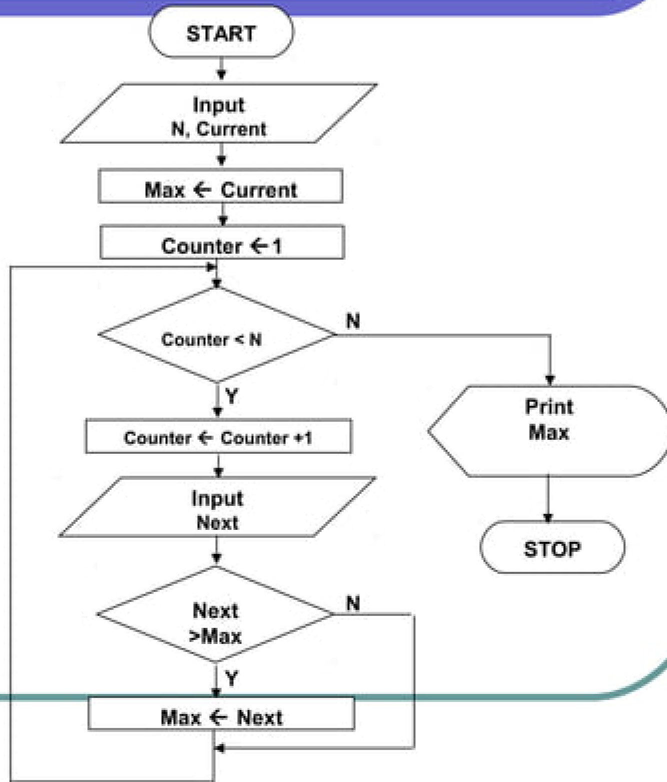
Example 10

- Write down an algorithm and draw a flowchart to find and print the largest of N (N can be any number) numbers. (Assume N to be 5 and the following set to be the numbers {1 4 2 6 8 })

Algorithm:

- Step 1: *Input* N
- Step 2: *Input* Current
- Step 3: Max ← Current
- Step 4: Counter ← 1
- Step 5: *While* (Counter < N)
  Repeat steps 5 through 8
- Step 6: Counter ← Counter + 1
- Step 7: *Input* Next
- Step 8: *If* (Next > Max) then
  Max ← Next
  endif
- Step 9: *Print* Max

Flowchart



START
Input
N, Current
Max ← Current
Counter ← 1
Counter < N
N
Y
Counter ← Counter +1
Input
Next
Next
>Max
N
Y
Max ← Next
Print
Max
STOP