Flow Charts, Algorithm, Pseudo Code

Algorithm

- A set of step-by-step instructions to accomplish a task.
  - An algorithm must have start instruction
  - Each instruction must be precise.
  - Each instruction must be unambiguous.
  - Each instruction must be executed in finite time.
  - An algorithm must have stop instruction.

Algorithm Example 1

- Suppose you are given a set of mark sheets where each mark sheet bears A, B, C or F letter grades.
- Write an algorithm to read mark sheet and print the grade that it contains.

1. Start
2. Take a mark sheet and read the grade.
3. Print the grade
4. Stop

Algorithm Example 2

- Suppose you are given a set of mark sheets where each mark sheet bears A, B, C or F grades.

Write an algorithm to read a mark sheet and print the grade if the grade is A only.

1. Start
2. Take a mark sheet and read the grade.
3. **If** grade is **A** then Print the grade
4. Stop

Algorithm Example 3

- Suppose you are given a set of mark sheets where each mark sheet bears A, B, C or F grades.

Write an algorithm to read a mark sheet and print the grade if it is A or B only.

1. Start
2. Take a mark sheet and read the grade.
3. **If** grade is **A or B** then Print the grade
4. Stop

Algorithm representation

- A pseudo code
- A flowchart
- Programs statements in a programming language.

Pseudocode Example 1

```
Start
Take a mark sheet and read the grade.
Print the grade
Stop
```

Pseudocode Example 2

```
Start
Take a mark sheet and read the grade.
If grade is A then Print the grade
Stop
```

Pseudo Code

- Pseudo code is another program analysis tool that is used for planning program logic.
- "Pseudo" means imitation or false
- "Code" refers to the instructions written in a programming language.
- Pseudo code, therefore, is an imitation of actual computer instructions.
- These pseudo instructions are phrases written in ordinary natural language (e.g., English).

Pseudo code Structure

- Pseudo code is made up of the following basic logic structures that have been proved to be sufficient for writing any computer program.

- Sequence
- Selection (IF...THEN...ELSE or IF....THEN)
- Iteration (DO...WHILE or REPEAT...UNTIL)

Advantages of pseudo code

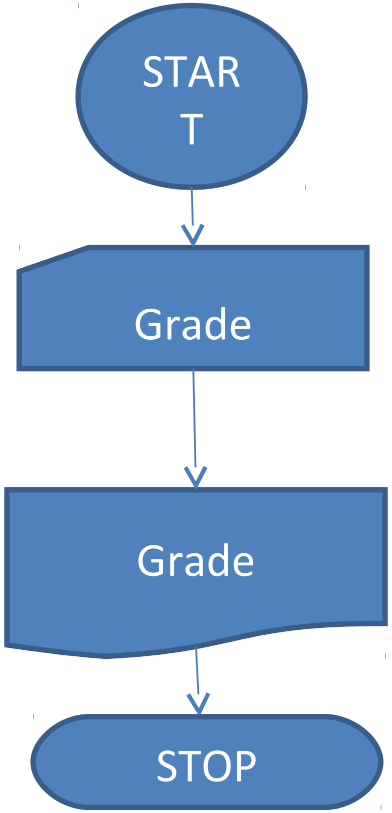
- Writing of pseudocode involves much less time and effort than drawing an equivalent flowchart.
- Converting a pseudo code to a programming language is much more easier as compared to converting a flowchart.
- It is easier to modify the pseudocode of a program logic when program modifications are necessary.

Limitations of Pseudocode

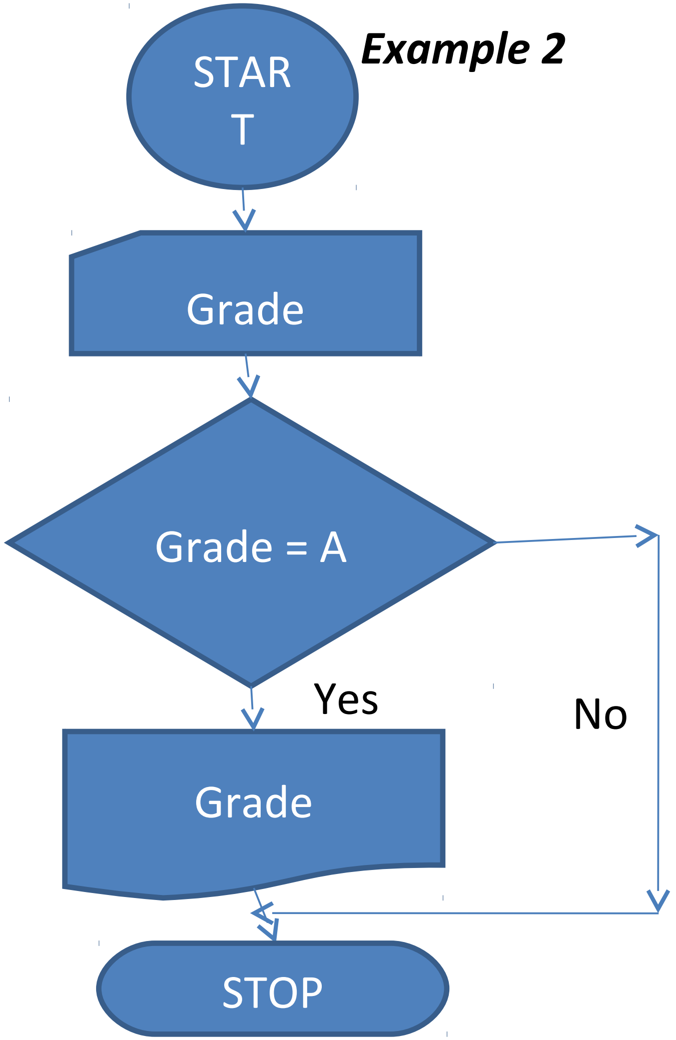
- In case of pseudo code, a graphic representation of program logic is not available.
- There are no standard rules to follow in using pseudocode.
- Different programmers use their own style of writing pseudocode
- communication problems occur due to lack of standardization.
- For a beginner, it is more difficult to follow the logic or write the pseudo code, as compared to flowcharting.

Flowchart

Example 1

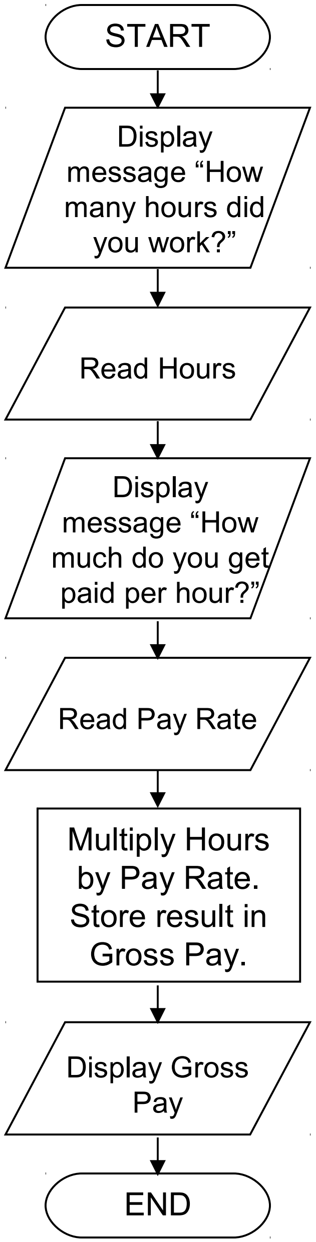


Example 2



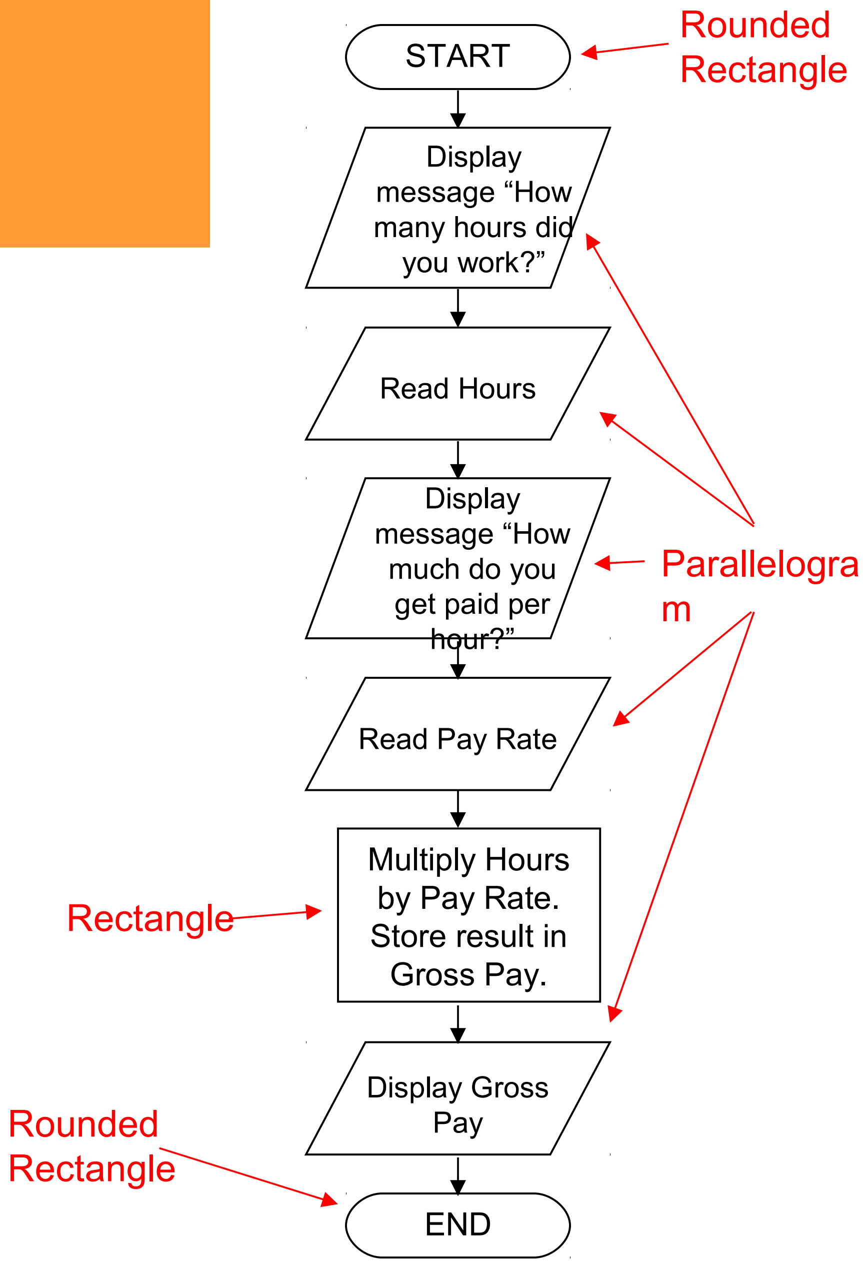
What is a Flowchart?

- A flowchart is a diagram that depicts the “flow” of a program.
- The figure shown here is a flowchart for the pay-calculating program.


START
Display message “How many hours did you work?”
Read Hours
Display message “How much do you get paid per hour?”
Read Pay Rate
Multiply Hours by Pay Rate. Store result in Gross Pay.
Display Gross Pay
END

Basic Flowchart Symbols

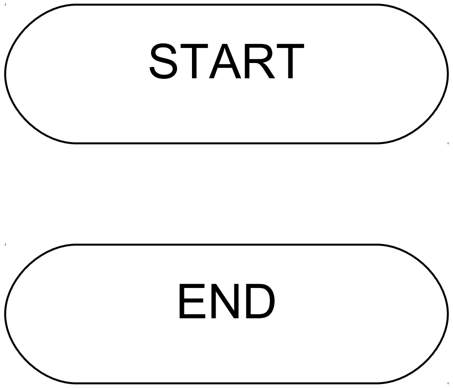
- Notice there are three types of symbols in this flowchart:
  - rounded rectangles
  - parallelograms
  - a rectangle
- Each symbol represents a different type of operation.



START
Display message “How many hours did you work?”
Read Hours
Display message “How much do you get paid per hour?”
Read Pay Rate
Multiply Hours by Pay Rate. Store result in Gross Pay.
Display Gross Pay
END
Rounded Rectangle
Parallelogram
Rectangle
Rounded Rectangle

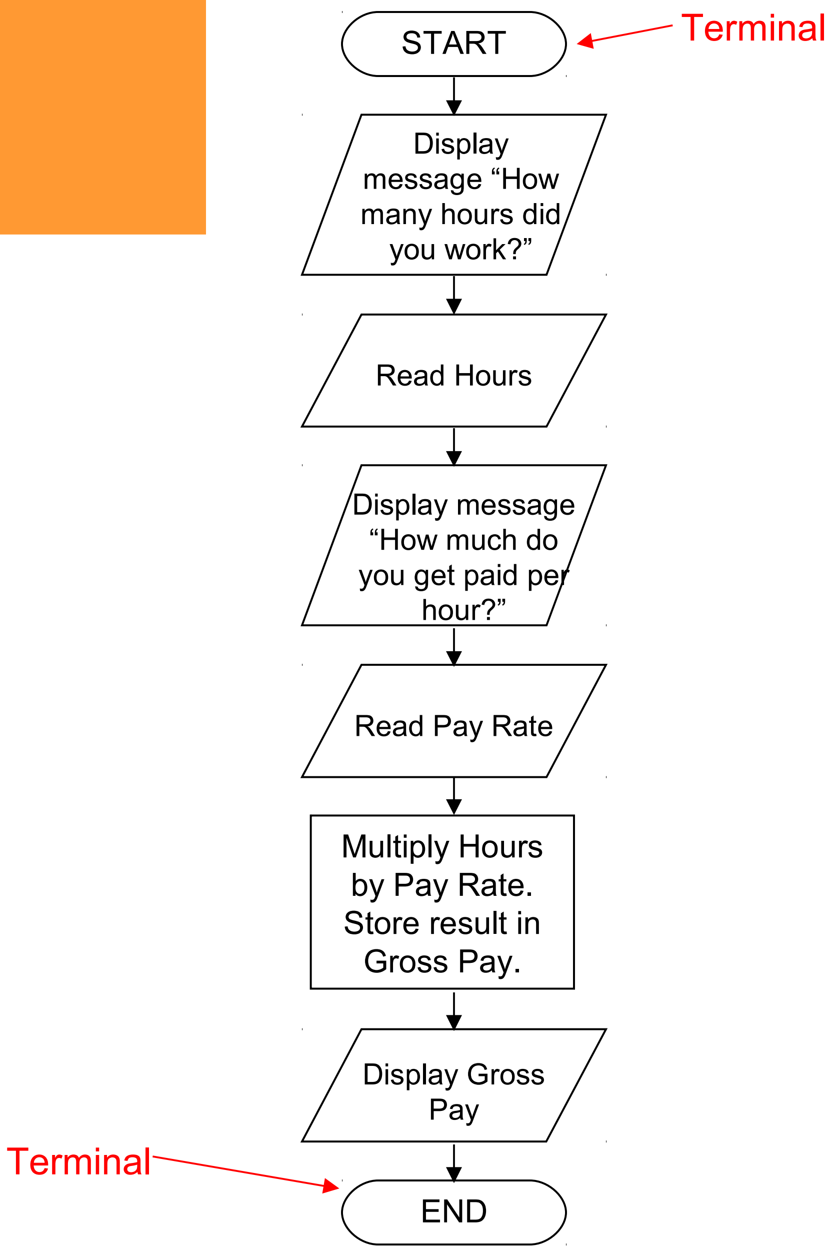
Basic Flowchart Symbols

- Terminals
  - represented by rounded rectangles
  - indicate a starting or ending point



START

END



Terminal

START

Display message “How many hours did you work?”

Read Hours

Display message “How much do you get paid per hour?”

Read Pay Rate

Multiply Hours by Pay Rate. Store result in Gross Pay.

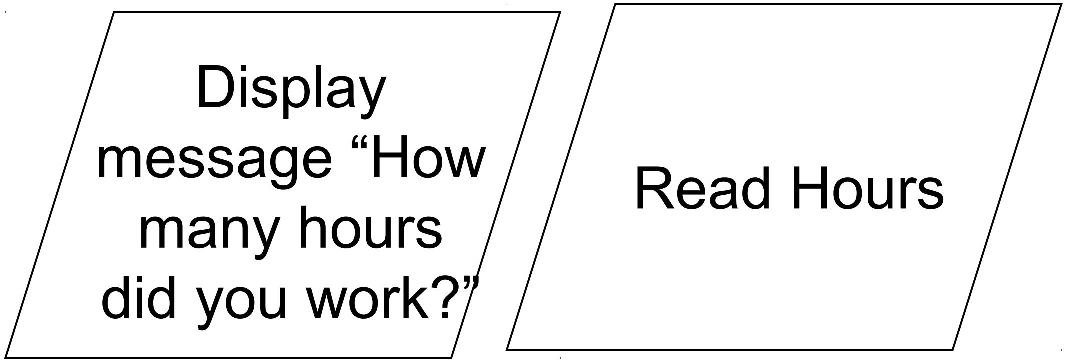
Display Gross Pay

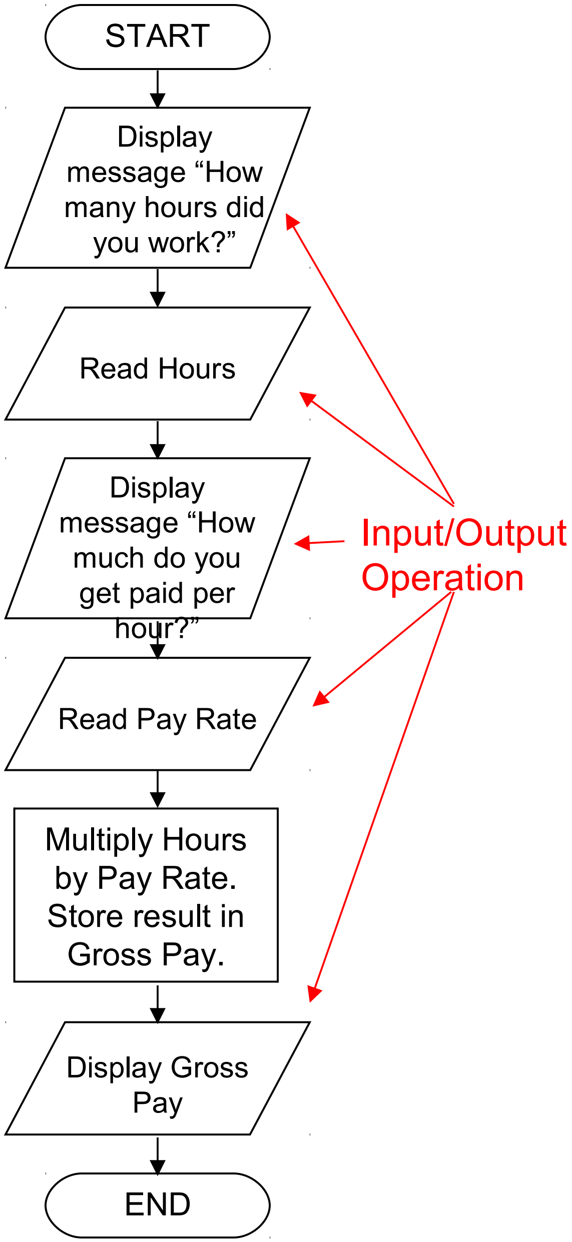
Terminal

END

Basic Flowchart Symbols

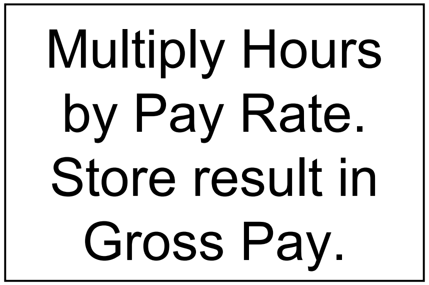
- Input/Output Operations
  - represented by parallelograms
  - indicate an input or output operation

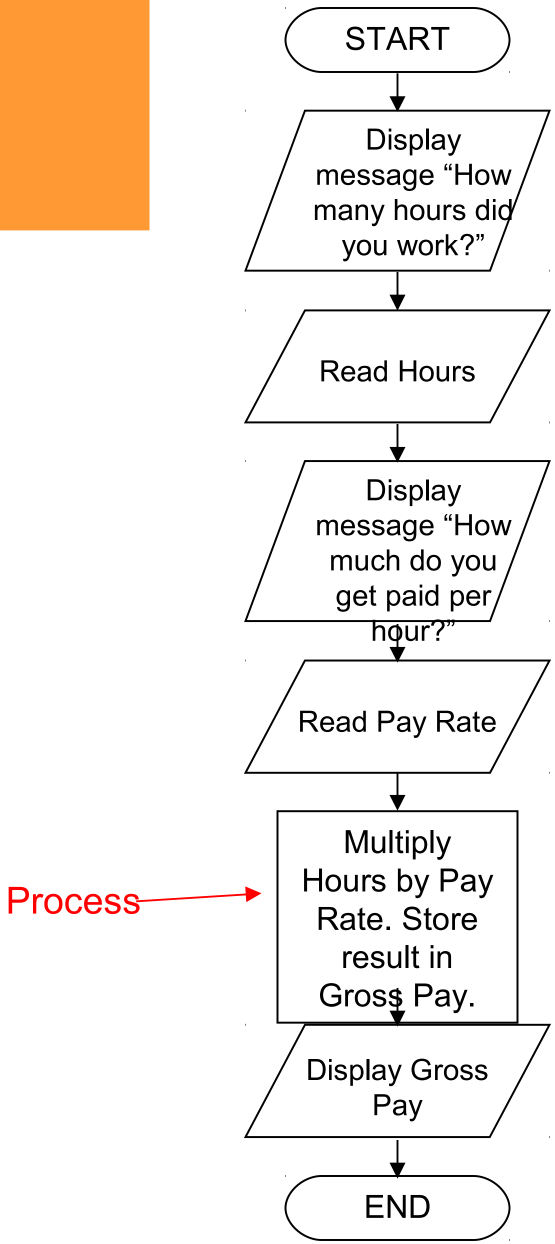



START
Display message “How many hours did you work?”
Read Hours
Display message “How much do you get paid per hour?”
Read Pay Rate
Multiply Hours by Pay Rate. Store result in Gross Pay.
Display Gross Pay
END
Input/Output Operation

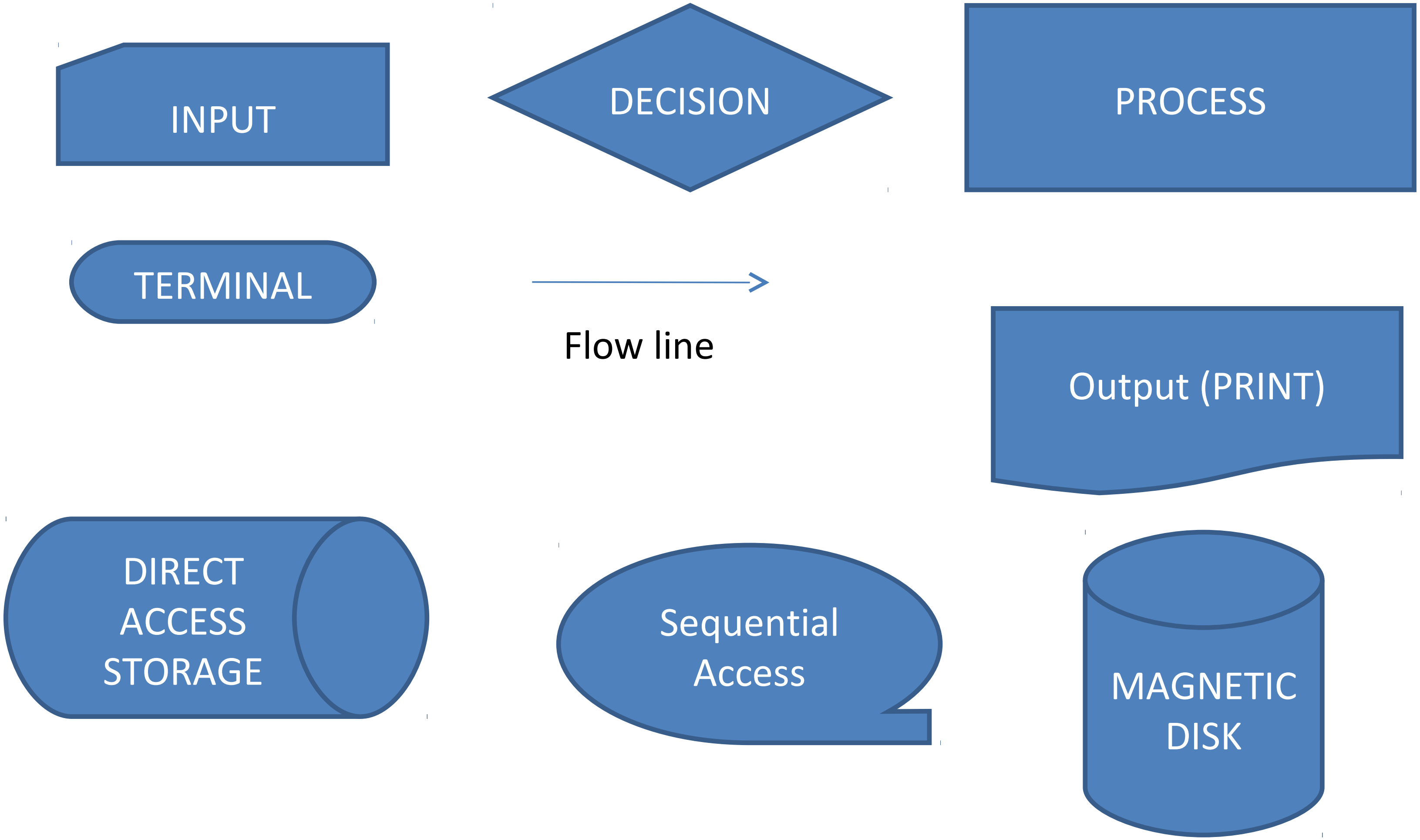
Basic Flowchart Symbols

- Processes
  - represented by rectangles
  - indicates a process such as a mathematical computation or variable assignment




START
Display message “How many hours did you work?”
Read Hours
Display message “How much do you get paid per hour?”
Read Pay Rate
Process
Multiply Hours by Pay Rate. Store result in Gross Pay.
Display Gross Pay
END

Basic Flowchart Symbols



INPUT

DECISION

PROCESS

TERMINAL

Flow line

Output (PRINT)

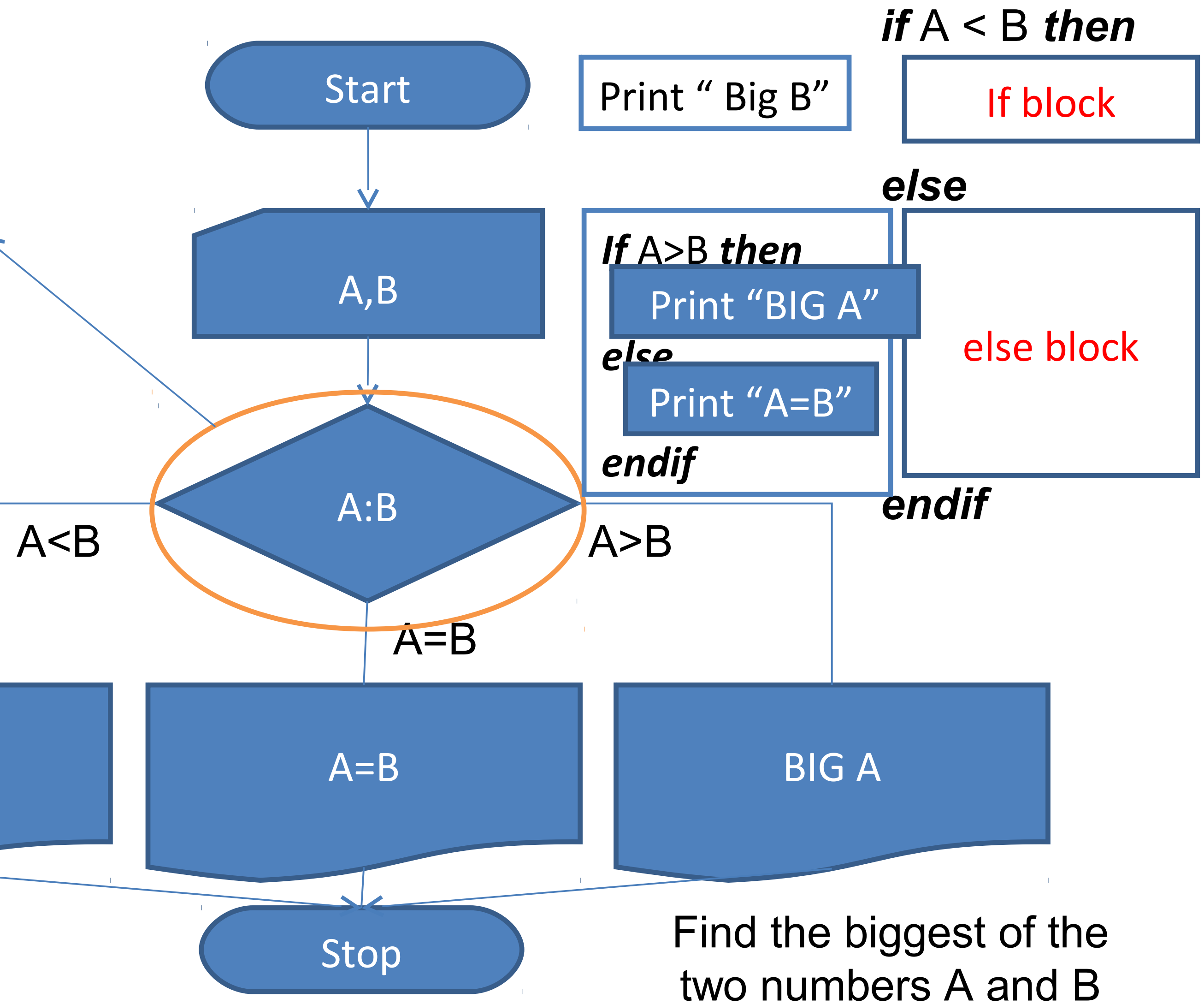
DIRECT ACCESS STORAGE

Sequential Access

MAGNETIC DISK

Flowchart/Pseudocode examples

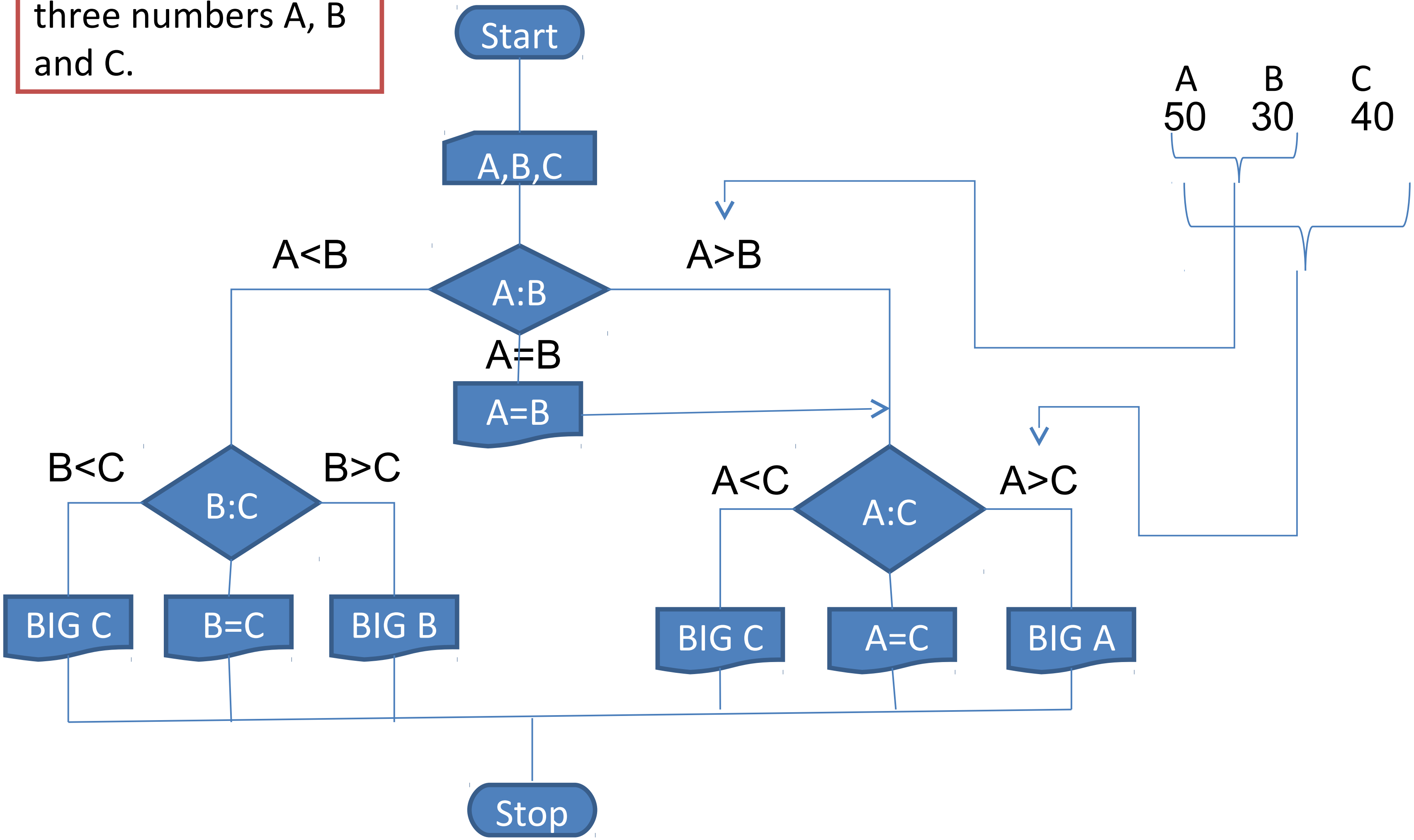
```
if A < B then
  Print "BIG B"
else
   If A>B then
     Print "BIG A"
   else
     Print "A=B"
   endif
endif
```



Find the biggest of the two numbers A and B

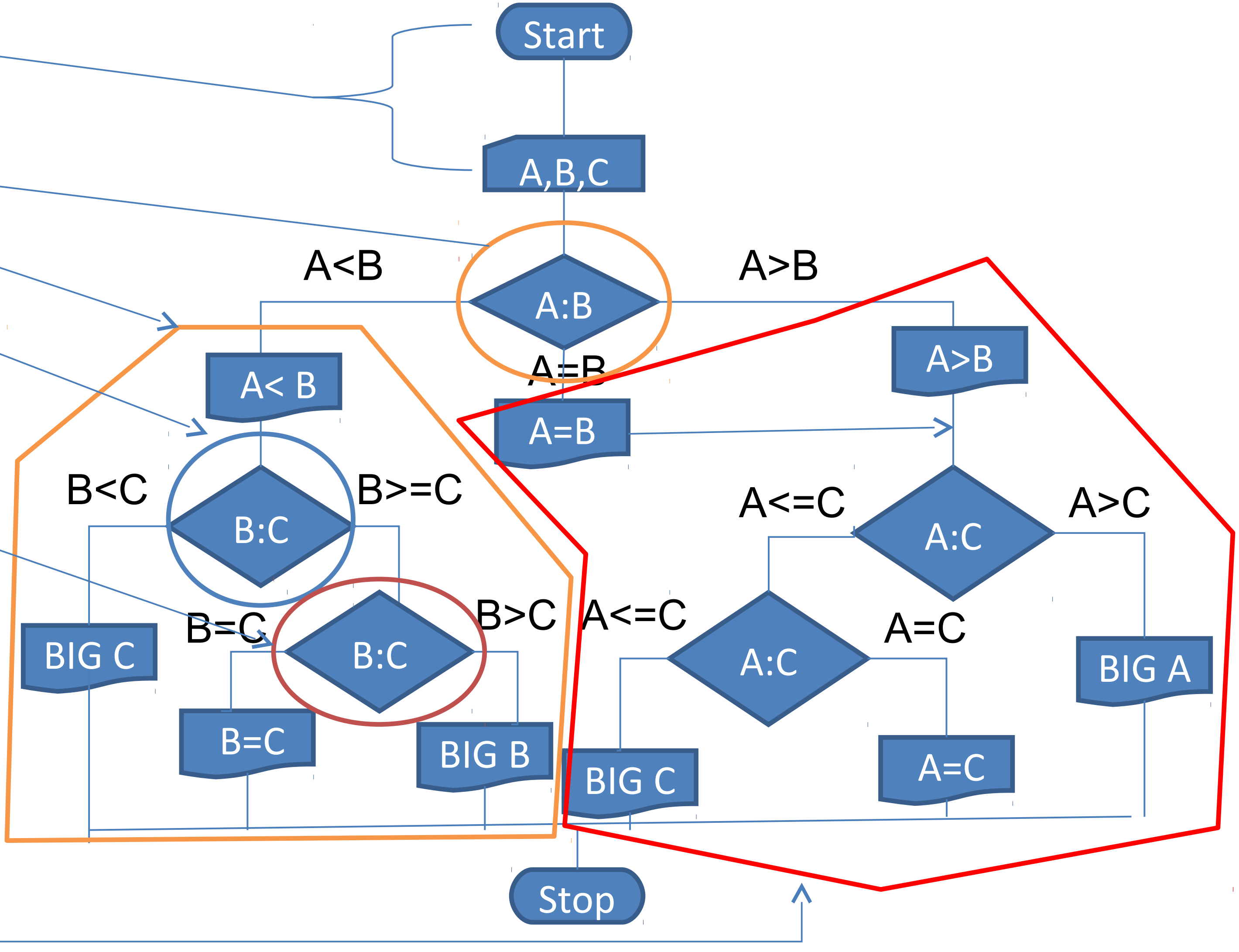
Flowchart/Pseudocode examples

Find the largest of three numbers A, B and C.



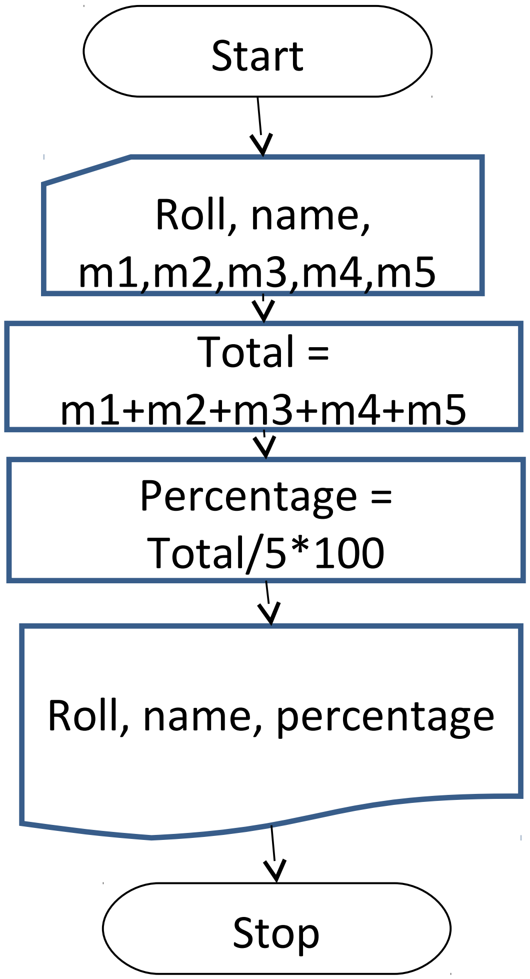
Flowchart/Pseudocode Selection

```
Start
Input A,B,C

If  A < B then
    Print "A<B"
    If B < C then
        Print "Big C"
    else
        if B = C then
            Print " B=C"
        else
            Print "Big B"
        end if
    end if
else

end if
```

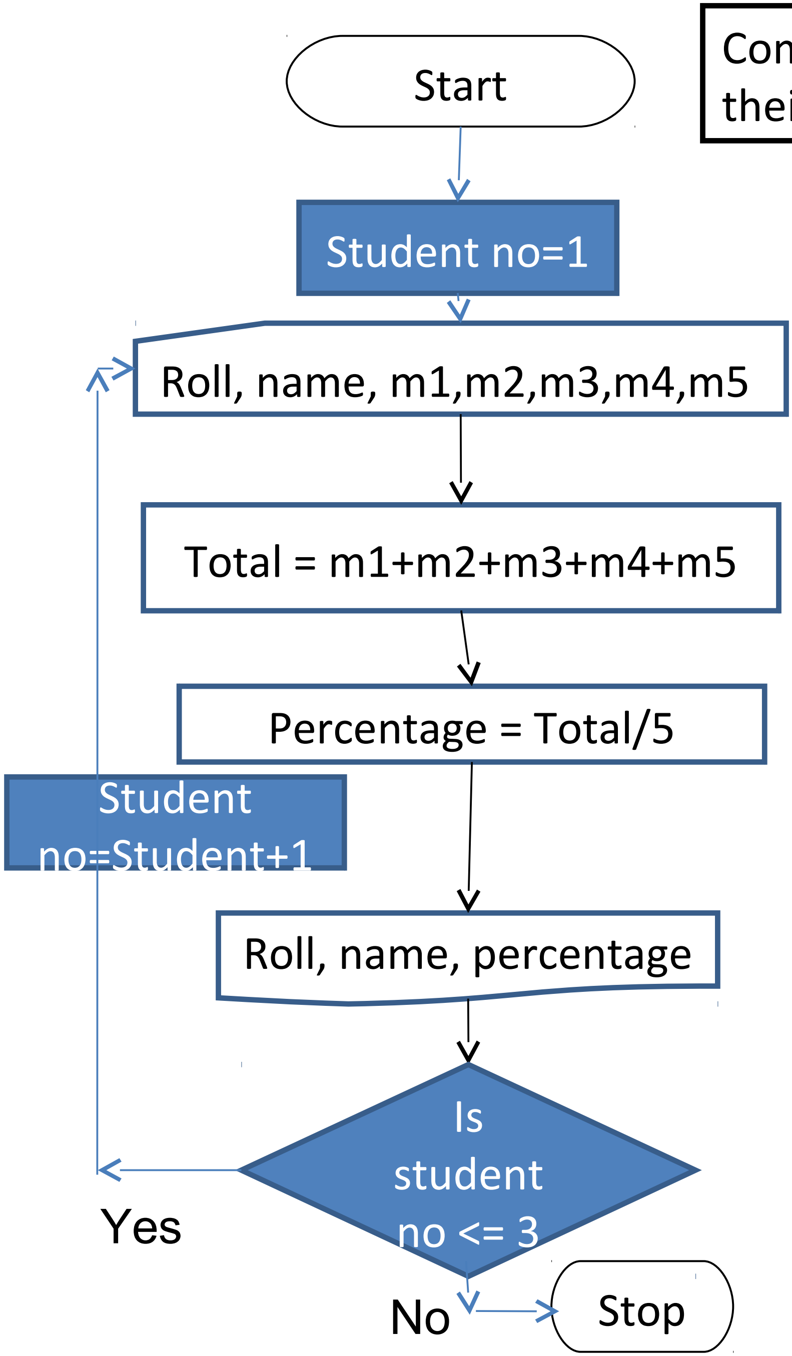


Flowcharts

- Given the input data:
  - Student Roll number, student name, the marks obtained in 5 subjects, each subject having maximum marks 100.
- Draw a flowchart for the algorithm to:
  - Calculate the percentage marks obtained , and
  - Print student's roll number and percentage of marks.



Compute percentages for **three** students and print their roll numbers, names, and percentage of marks.



Start

Student no=1

Roll, name, m1,m2,m3,m4,m5

Total = m1+m2+m3+m4+m5

Percentage = Total/5

Roll, name, percentage

Is student no <= 3

Yes

Student no=Student+1

No

Stop

Student no	1	2	3
Roll	25	35	27
name	A	B	C
m1	70	80	60
m2	80	70	45
m3	80	70	90
m4	90	70	60
m5	40	50	30
Total	360	340	285
Percentage	72	68	57

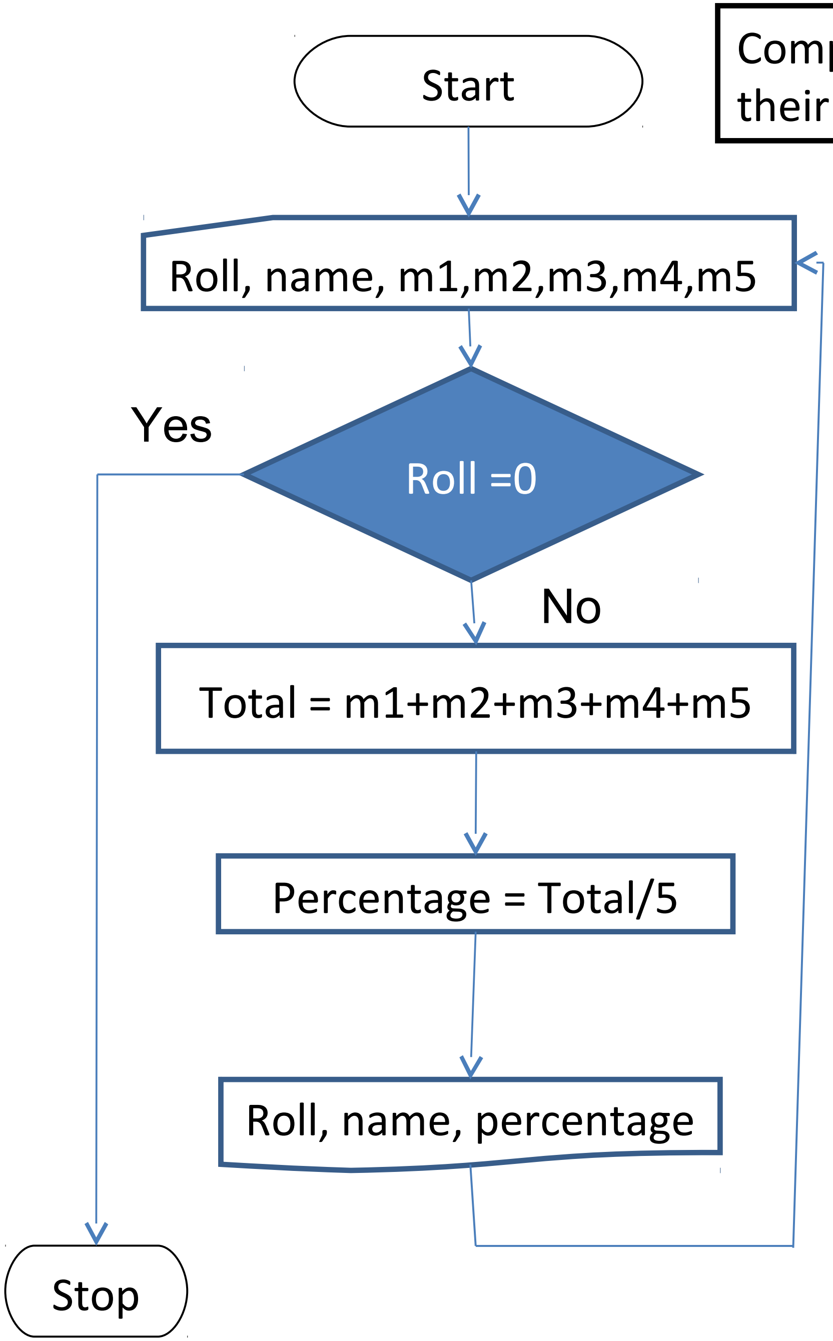
Variables or Memory address

First iteration

Second iteration

Third iteration

Compute percentages for **many** students and print their roll numbers, names, and percentage of marks.



Start

Roll, name, m1,m2,m3,m4,m5

Roll =0

Yes

No

Total = m1+m2+m3+m4+m5

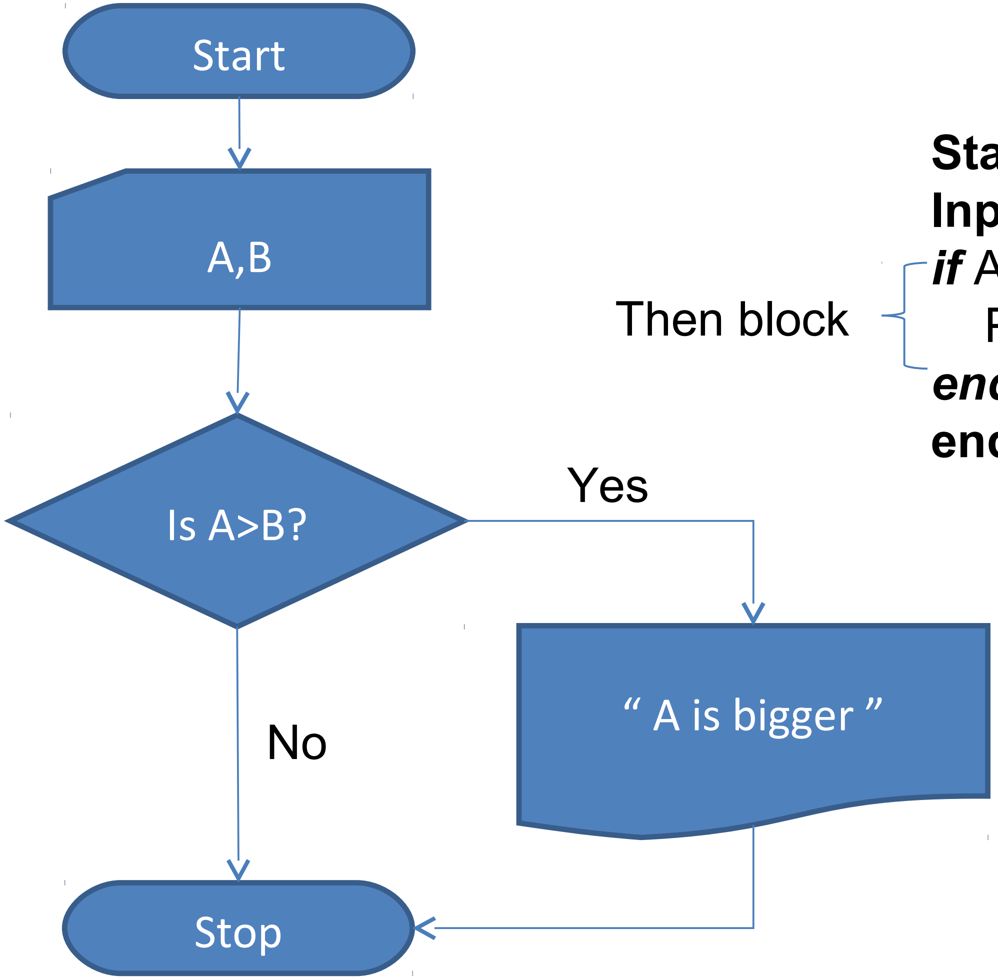
Percentage = Total/5

Roll, name, percentage

Stop

Roll	25	35	27
name	A	B	C
m1	70	80	60
m2	80	70	45
m3	80	70	90
m4	90	70	60
m5	40	50	30
Total	360	340	285
Percentage	72	68	57
Variables or Memory address	First iteration	Second iteration	Third iteration

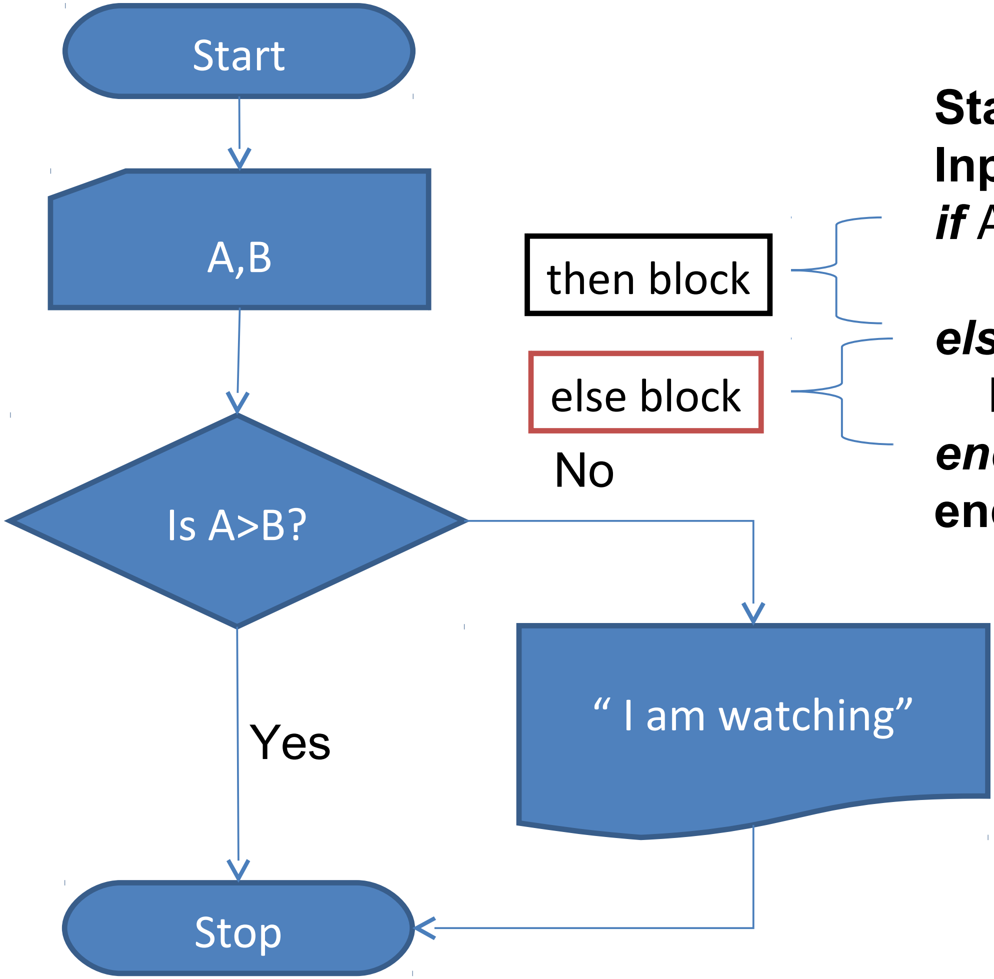
Flowchart Selection



Then block

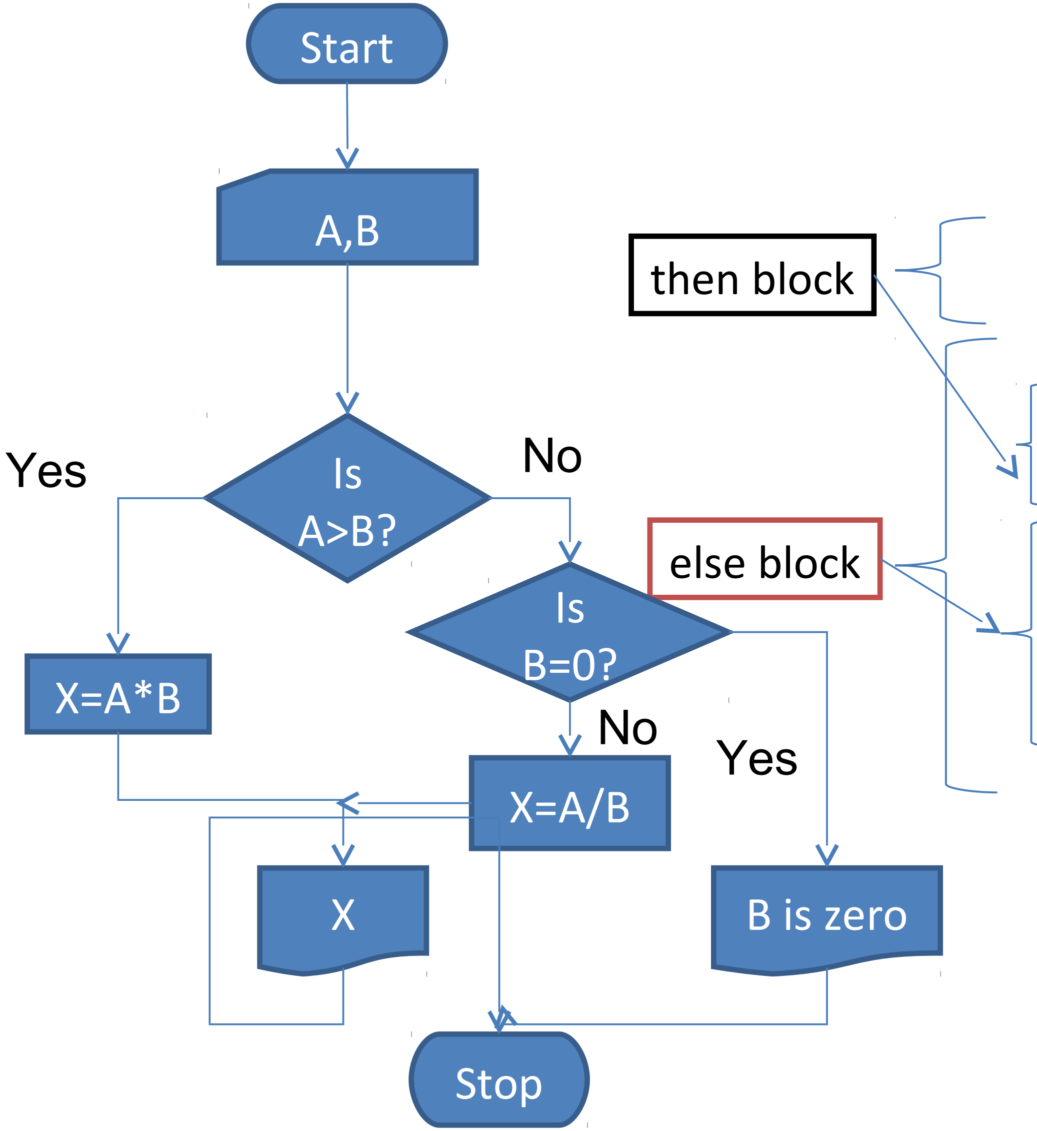
```
Start
Input A, B
if A > B then
    Print “ A is bigger”
end if
end
```

Flowchart Selection



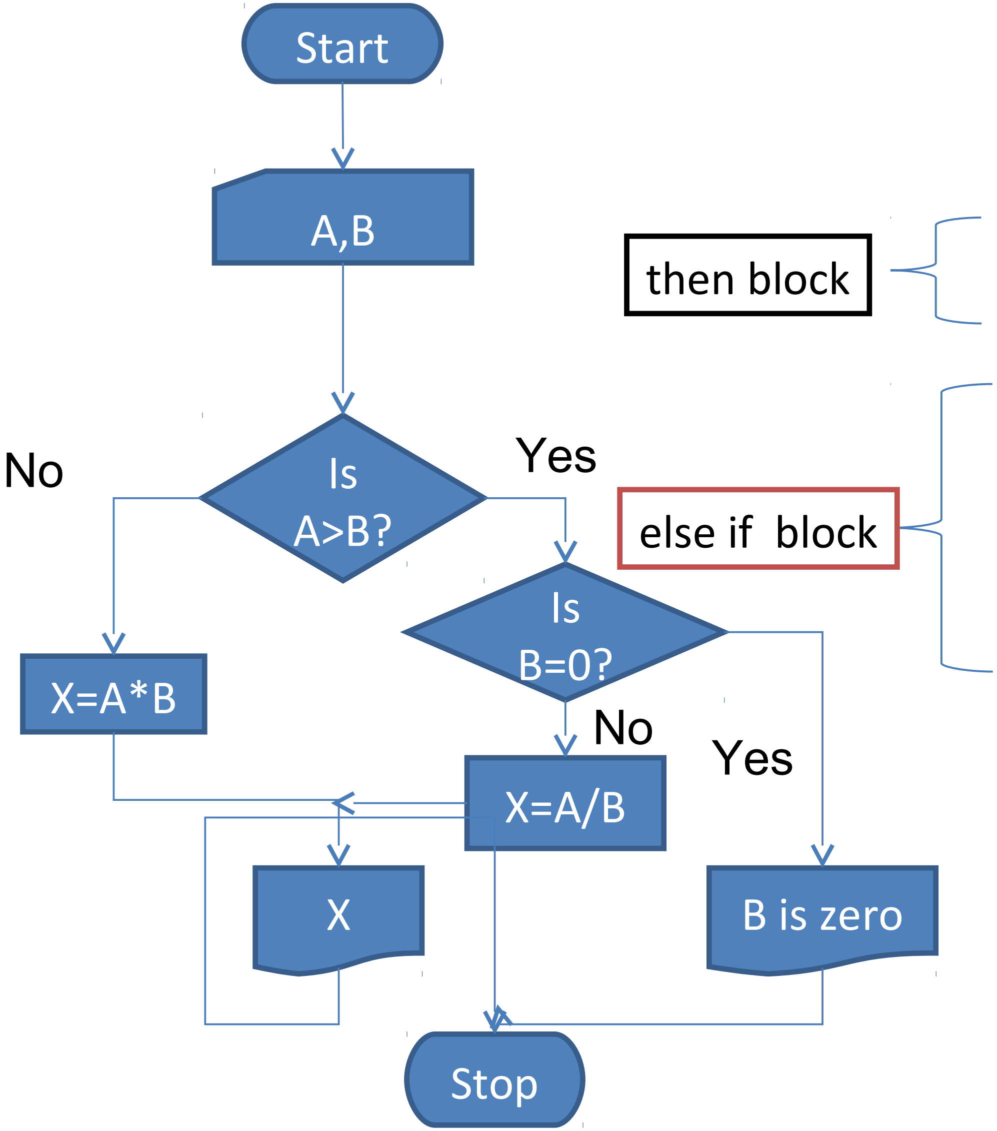
```
Start
Input A, B
if A > B then

else
    Print " I am watching"
end if
end
```

Flowchart Selection



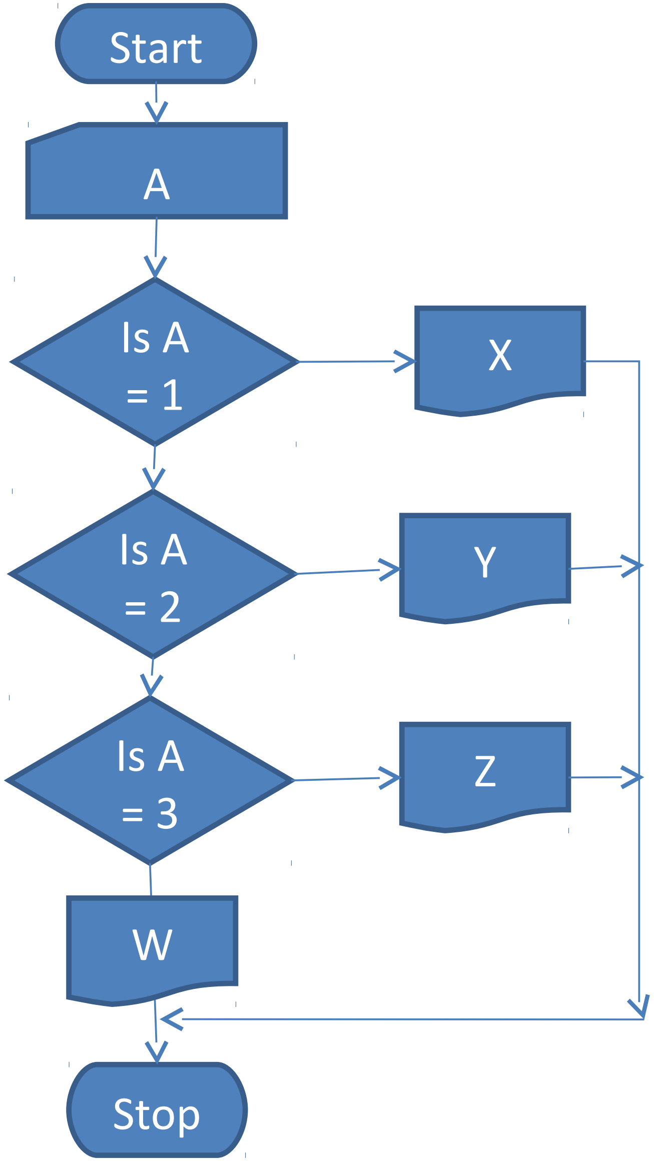
```
Start
Input A, B
if A > B then
   X=A*B
   Print X
else
  if B=0 then
    Print “ B is zero”
   else
     X=A/B
   Print X
  end if
end if
end
```

Flowchart Selection



```
Start
Input A, B
if A > B then
   X=A*B
   Print X
else if B=0 then
    Print " B is zero"
   else
     X=A/B
     Print X
end if
end
```

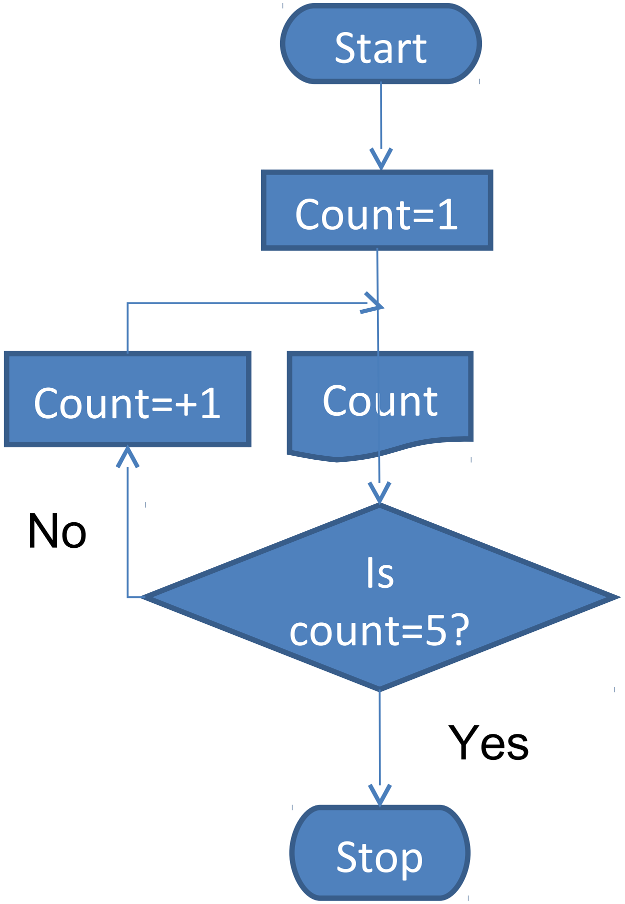
Flowchart Selection (Case)



```
start
Input A
If  A =1 then
    Print “X”
    elseif A =2 then
         Print “Y”
         elseif A=3 then
           Print “Z”
           else
               Print “W”
endif
end
```

```
start
Input A
case A of
                  1: Print “X”
                  2: Print “Y”
                  3: Print “Z”
otherwise: Print “W”
endcase
end
```

Flowchart Iteration



FOR LOOP

Range

```
Start
for Count = 1 to 5 by 1 do
    Print Count
end for
end
```

WHILE LOOP

Range

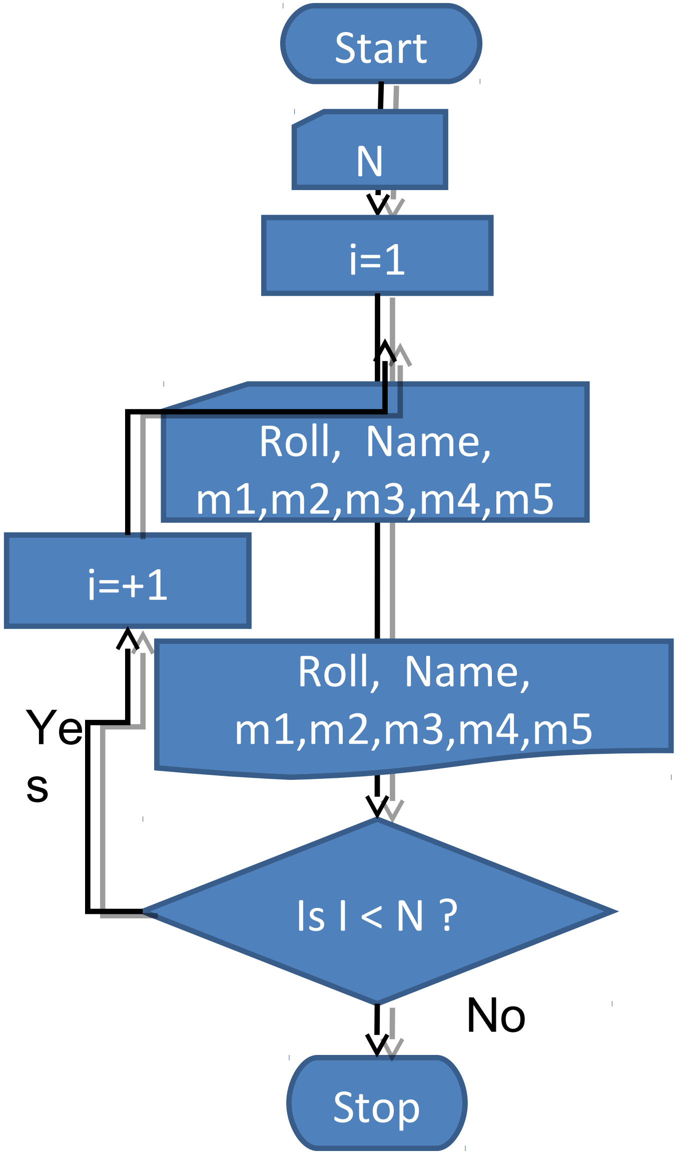
```
Start
Count = 1
while Count  <= 5 do
      Print Count
       Count=Count+1
end while
end
```

REPEAT LOOP

Range

```
Start
Count = 1
repeat
      Print Count
       Count=Count+1
until count > 5
end
```

Flowchart Iteration



```
**Start**
***Input*** N
**for** I = 1 **to** N **by** 1 ***do***
    ***Input*** Roll, Name, m1, m2, m3, m4, m5
    ***Print*** Roll, Name, m1, m2, m3, m4, m5
***end for***
**end**
```

```
**Start**
***Input*** N
i  **=** 1
**repeat**
    ***Input*** Roll, Name, m1, m2, m3, m4, m5
    ***Print*** Roll, Name, m1, m2, m3, m4, m5
i++
***until***  i > N
**end**
```

Advantages of Flowchart

1) Conveys Better Meaning

2) Analyses the Problem Effectively

3) Effective Joining of a Part of a System

A group of programmers are normally associated with the design of large software systems. Each programmer is responsible for designing only a part of the entire system. So initially, if each programmer draws a flowchart for his part of design, the flowcharts of all the programmers can be placed together to visualize the overall system design. Any problem in linking the various parts of the system can be easily detected at this stage and the design can be accordingly modified. Flowcharts can thus be used.

4) Efficient Coding

5) Systematic Testing

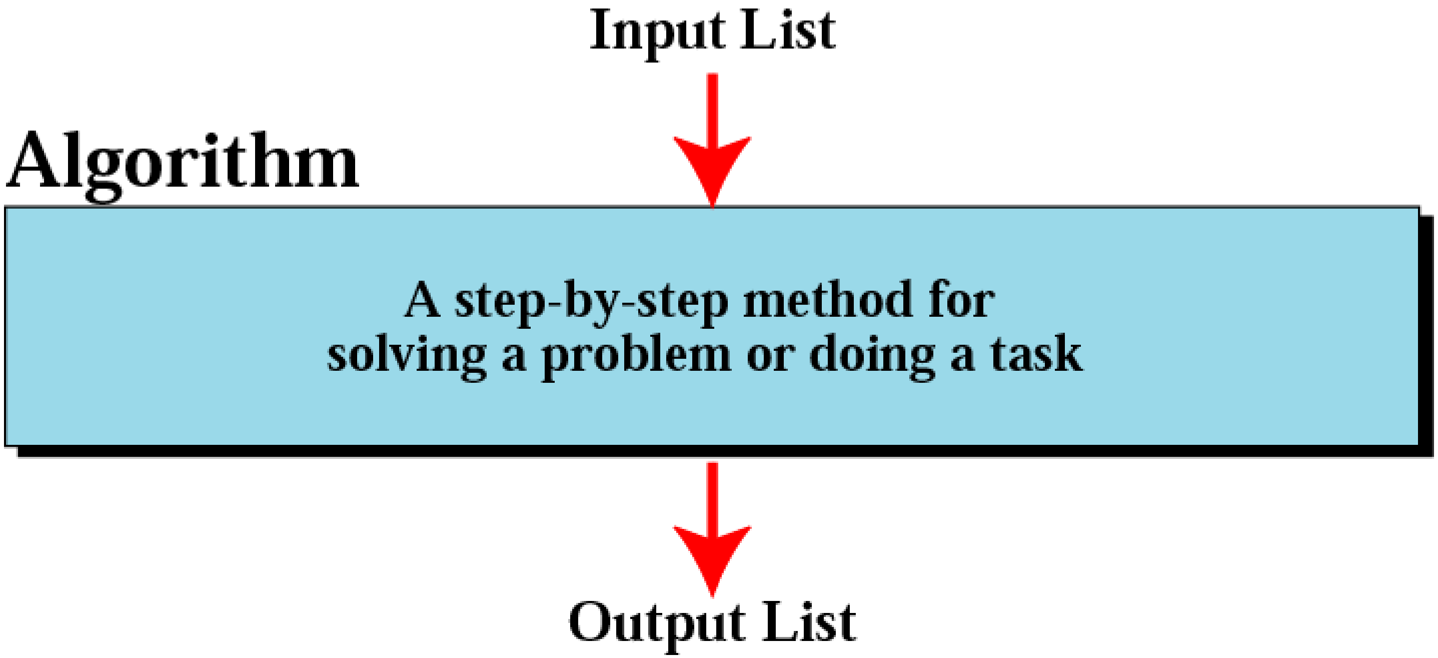
Limitations of Flowcharts

1) Takes More Time to Draw

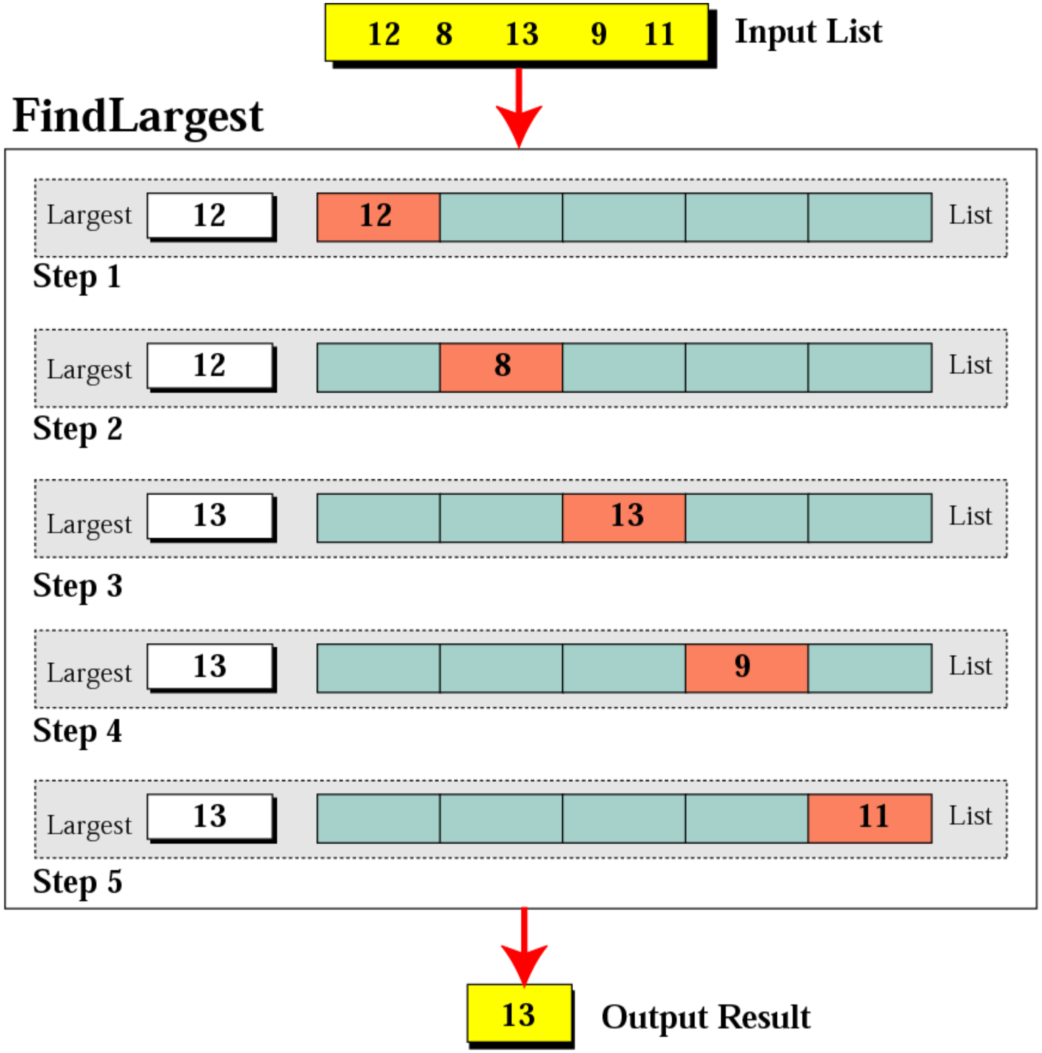
2) Difficult to Make Changes:- Owing to the symbol-string nature of flowcharting, any changes or modifications in the program logic will usually require a completely new flowchart. Redrawing a flowchart is tedious.

3) Non-standardization :- There are no standards determining the amount of detail that should be included in a flowchart.

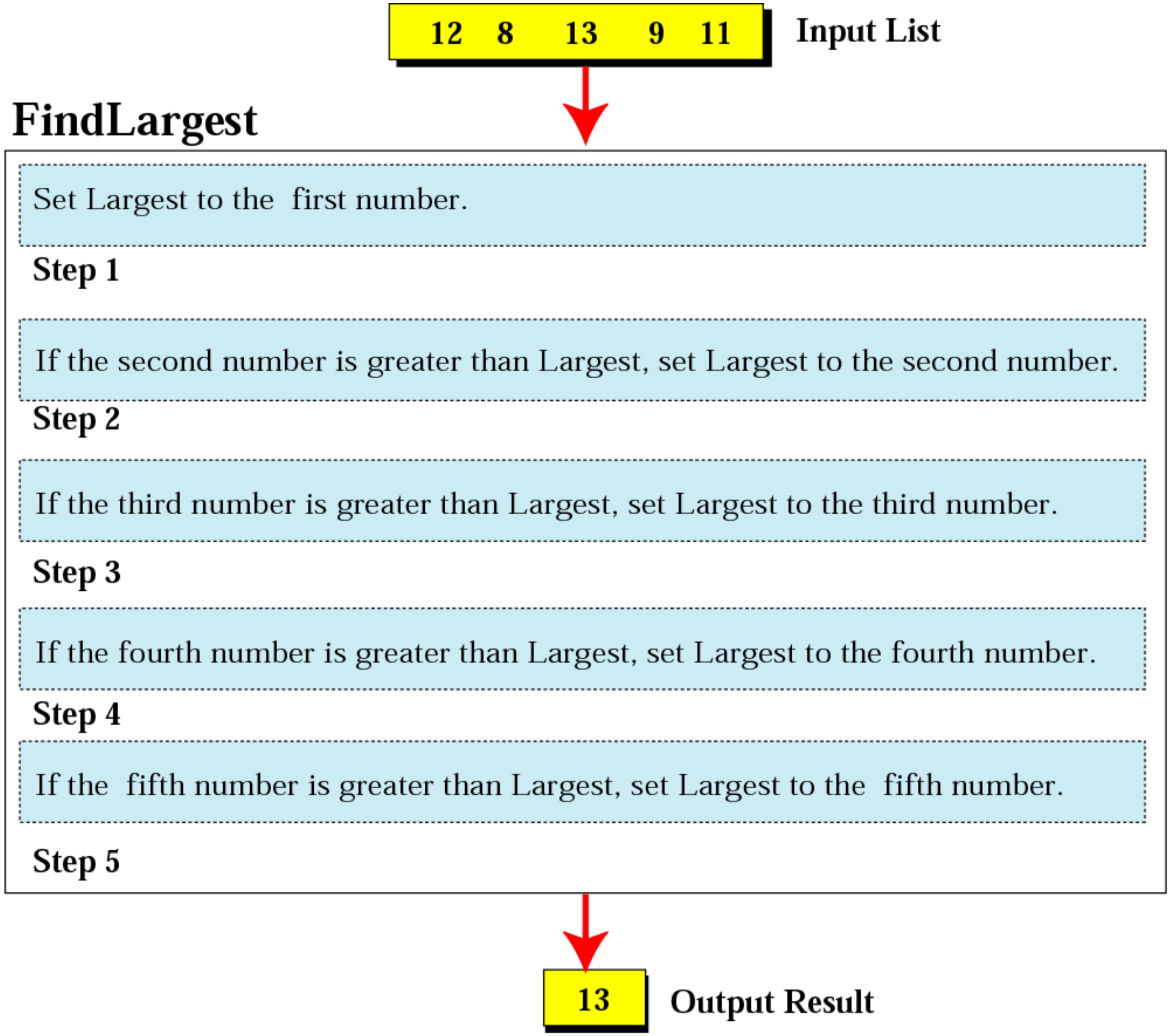
Informal definition of an algorithm used in a computer



Finding the largest integer among five integers



Defining actions in FindLargest algorithm



12 8 13 9 11

Input List

FindLargest

Set Largest to the first number.

Step 1

If the second number is greater than Largest, set Largest to the second number.

Step 2

If the third number is greater than Largest, set Largest to the third number.

Step 3

If the fourth number is greater than Largest, set Largest to the fourth number.

Step 4

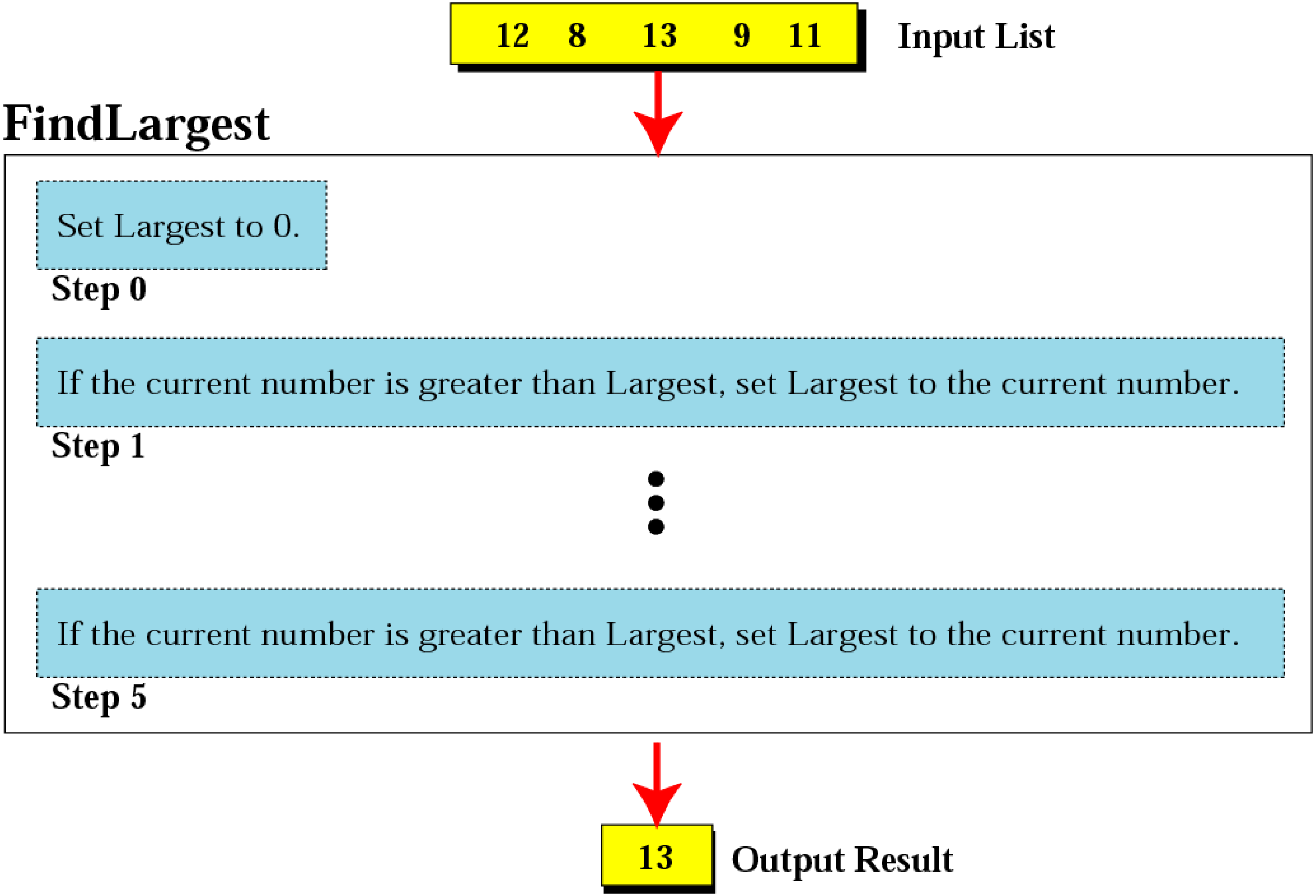
If the fifth number is greater than Largest, set Largest to the fifth number.

Step 5

13

Output Result

FindLargest refined



12 8 13 9 11 Input List

FindLargest

Set Largest to 0.

Step 0

If the current number is greater than Largest, set Largest to the current number.

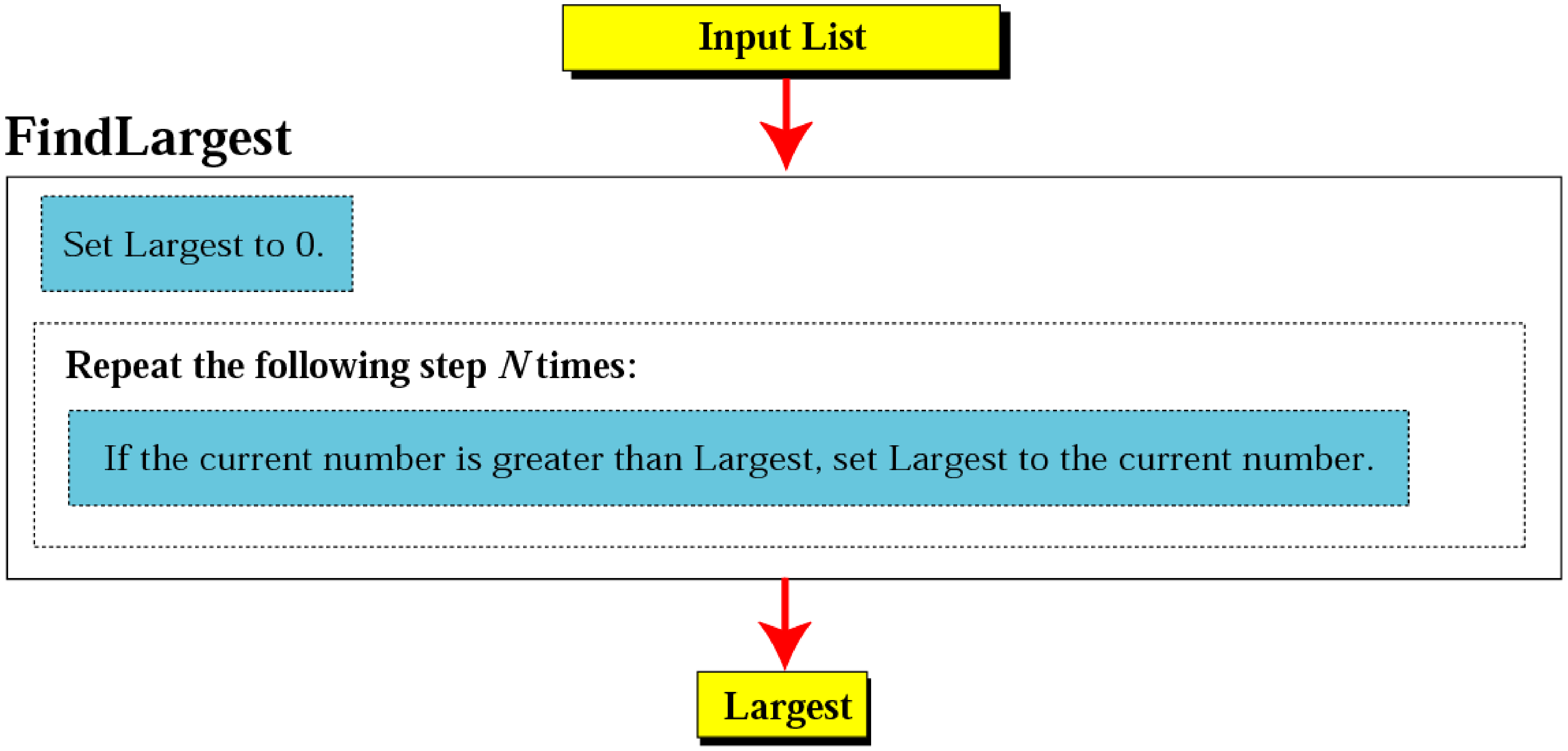
Step 1

If the current number is greater than Largest, set Largest to the current number.

Step 5

13 Output Result

Generalization of FindLargest



Input List

FindLargest

Set Largest to 0.

Repeat the following step *N* times:

If the current number is greater than Largest, set Largest to the current number.

Largest

Example 1

Write an algorithm in pseudocode that finds the average of two numbers

Solution

See Algorithm 8.1 on the next slide.

Algorithm 8.1:Average of two

```
   AverageOfTwo
   Input: Two numbers
1. Add the two numbers
2. Divide the result by 2
3. Return the result by step 2
   End
```

Example 2

Write an algorithm to change a numeric grade to a pass/no pass grade.

Solution

See Algorithm 8.2 on the next slide.

Algorithm 8.2:Pass/no pass Grade

```
    Pass/NoPassGrade
    Input: One number
1.  if (the number is greater than or equal to 33)
    then
        1.1  Set the grade to “pass”
    else
        1.2   Set the grade to “nopass”
    End if
2.  Return the grade
    End
```

Example 3

Write an algorithm to change a numeric grade to a letter grade.

Solution

See Algorithm 8.3 on the next slide.

Algorithm 8.3: Letter grade

```
    Letter Grade
    Input: One number
1.  if (the number is between 90 and 100,
    inclusive)
    then
        1.1  Set the grade to “A”
    End if
2.  if (the number is between 80 and 89,
    inclusive)
    then
        2.1  Set the grade to “B”
    End if
```

Algorithm 8.3: Letter grade

```
3.  if (the number is between 70  and 79,
    inclusive)
    then
        3.1  Set the grade to “C”
    End if
4.  if (the number is between 60  and 69,
    inclusive)
    then
        4.1  Set the grade to “D”
    End if
```

Algorithm 8.3: Letter grade

```
5.  If (the number is less than 60)
    then
        5.1  Set the grade to “F”
    End if
6.  Return the grade
    End
```

Example 4

Write an algorithm to find the largest of a set of numbers. You do not know the number of numbers.

Solution

See Algorithm 8.4 on the next slide.

Algorithm 8.4: Find largest

```
    Find Largest
    Input: A list of positive integers
1.  Set Largest to 0
2.  while (more integers)
     2.1  if (the integer is greater than Largest)
               then
                 2.1.1  Set largest to the value of the integer
               End if
    End while
3.  Return Largest
    End
```

Example 5

Write an algorithm to find the largest of 1000 numbers.

Solution

See Algorithm 8.5 on the next slide.

Algorithm 8.5: *Find largest of 1000 numbers*

```
       FindLargest
       Input: 1000 positive integers
1.     Set Largest to 0
2.     Set Counter to 0
3.     while (Counter less than 1000)
        3.1  if (the integer is greater than Largest)
                 then
                   3.1.1  Set Largest to the value of the integer
               End if
         3.2  Increment Counter
       End while
4.     Return Largest
       End
```

8.4

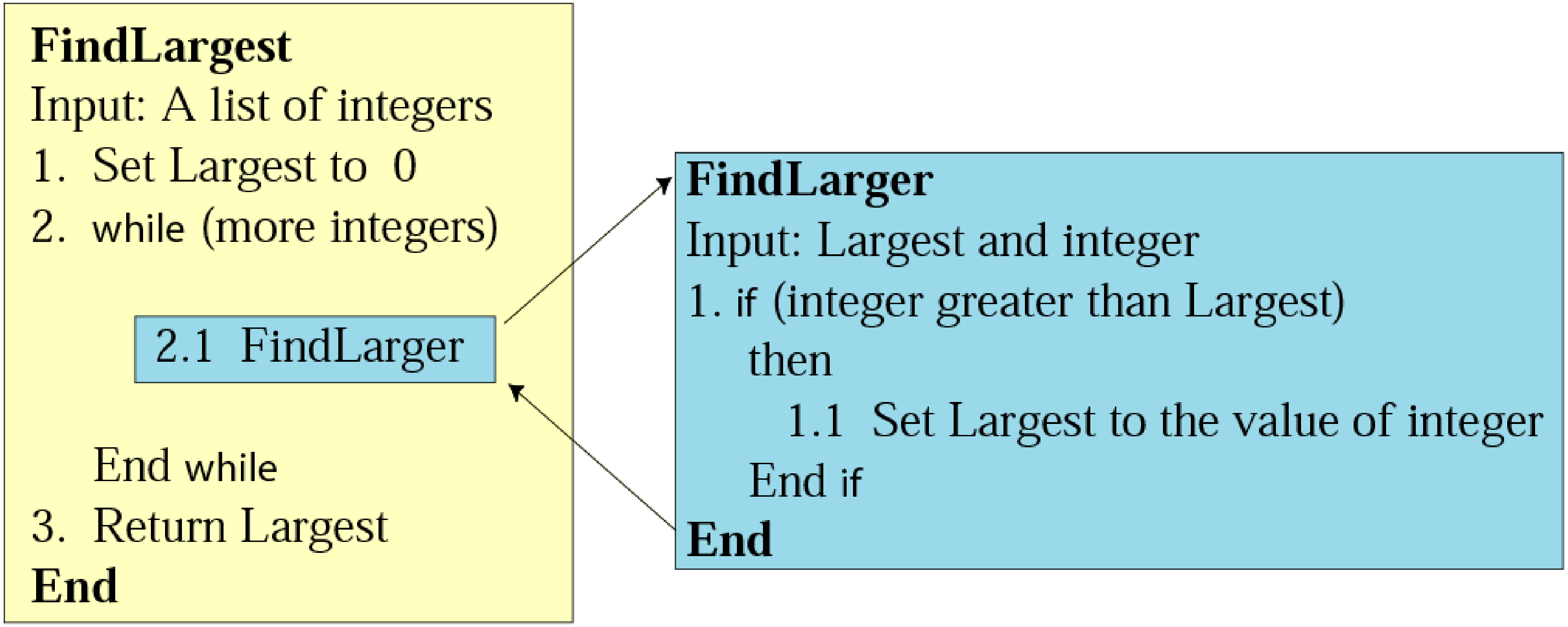
MORE FORMA DEFINITION

- Ordered set
- Unambiguous steps
- Effectiveness
- Termination

8.5

SUBALGORITHMS

Concept of a subalgorithm



```
FindLargest
Input: A list of integers
1. Set Largest to 0
2. while (more integers)

      2.1 FindLarger

   End while
3. Return Largest
End
```

```
FindLarger
Input: Largest and integer
1. if (integer greater than Largest)
     then
        1.1 Set Largest to the value of integer
     End if
End
```

Algorithm 8.6: Find largest

```
    FindLargest
    Input: A list of positive integers
1.  Set Largest to 0
2.  while (more integers)
     2.1  FindLarger
    End while
3.  Return Largest
    End
```

Subalgorithm: Find larger

```
    FindLarger
    Input: Largest and current integer
1.  if (the integer is greater than Largest)
    then
            1.1 Set Largest to the value of the integer
    End if
    End
```