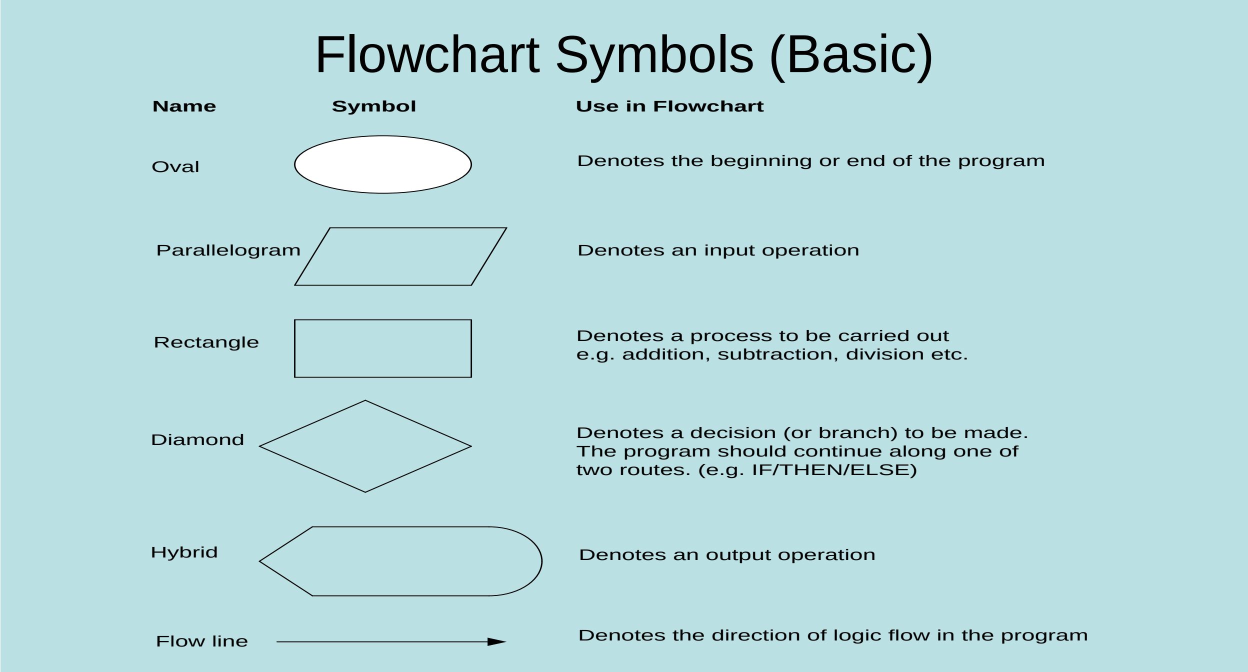
Algorithm, Pseudo code & Flowcharts

Lecturer: K. P. Ghosh

Programming Tools

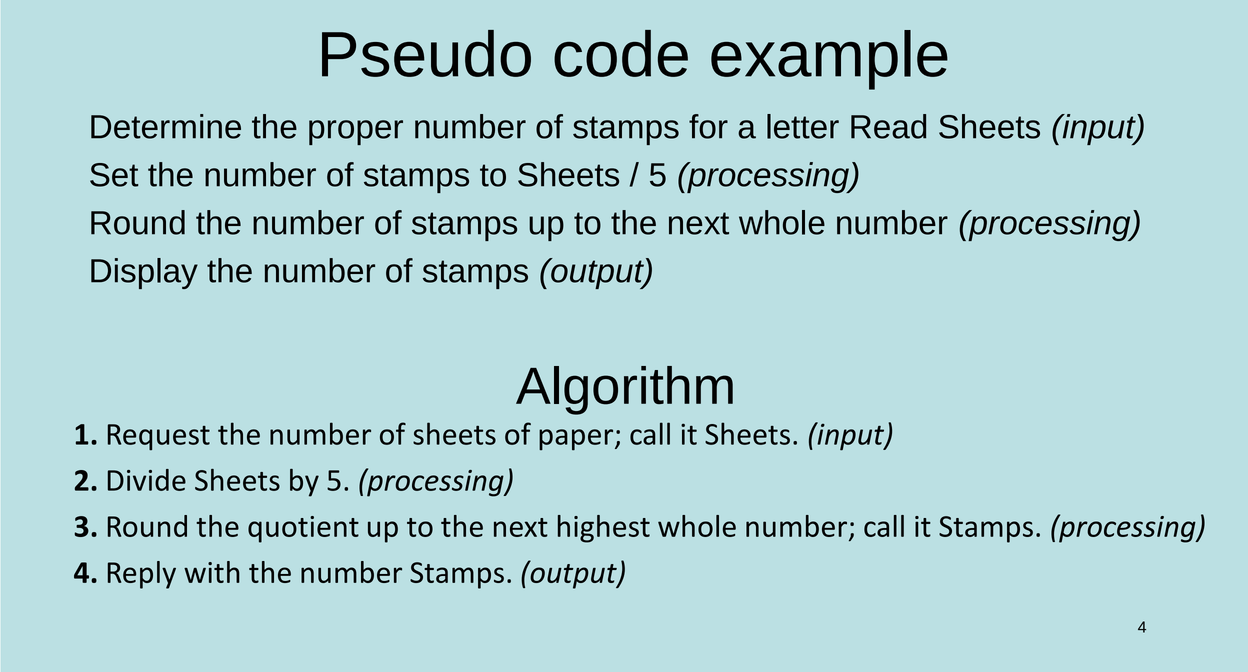
Tools are used to convert algorithms into computer programs:

- **Flowchart** - Graphically depicts the logical steps to carry out a task and shows how the steps relate to each other.
- **Pseudo code** - Uses English-like phrases with some Visual Basic terms to outline the program.



Flowchart Symbols (Basic)

Name	Symbol	Use in Flowchart
Oval	⬭	Denotes the beginning or end of the program
Parallelogram	▱	Denotes an input operation
Rectangle	▭	Denotes a process to be carried out e.g. addition, subtraction, division etc.
Diamond	◇	Denotes a decision (or branch) to be made. The program should continue along one of two routes. (e.g. IF/THEN/ELSE)
Hybrid		Denotes an output operation
Flow line	→	Denotes the direction of logic flow in the program



Pseudo code example

Determine the proper number of stamps for a letter Read Sheets *(input)*

Set the number of stamps to Sheets / 5 *(processing)*

Round the number of stamps up to the next whole number *(processing)*

Display the number of stamps *(output)*

Algorithm

1. Request the number of sheets of paper; call it Sheets. *(input)*

2. Divide Sheets by 5. *(processing)*

3. Round the quotient up to the next highest whole number; call it Stamps. *(processing)*

4. Reply with the number Stamps. *(output)*

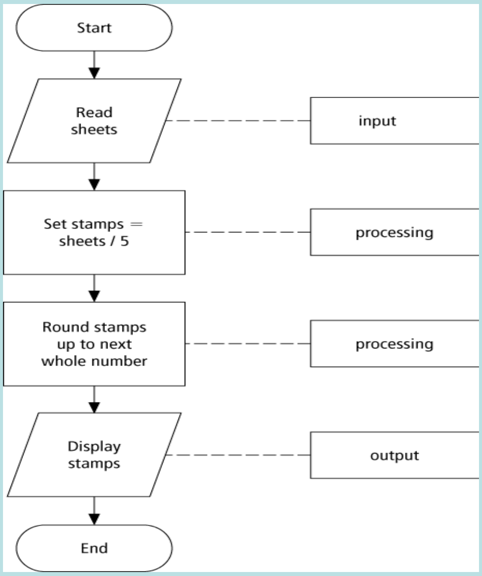
Divide-and-conquer method

- Used in problem solving – take a large problem and break it into smaller problems solving the small ones first
- Breaks a problem down into modules

Statement structures

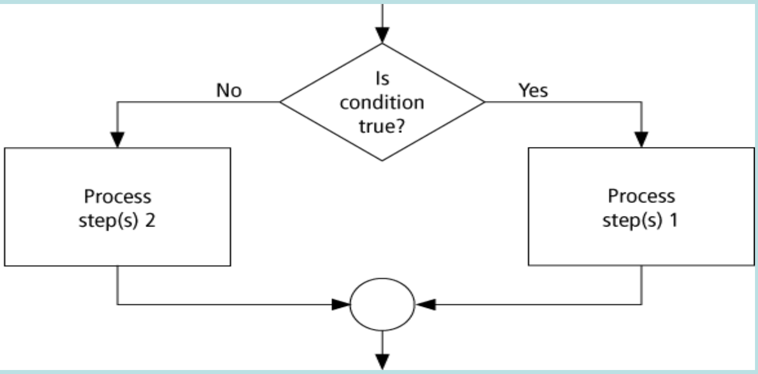
- Sequence – follow instructions from one line to the next without skipping over any lines
- Decision - if the answer to a question is “Yes” then one group of instructions is executed. If the answer is “No,” then another is executed
- Looping – a series of instructions are executed over and over

Sequence flow chart



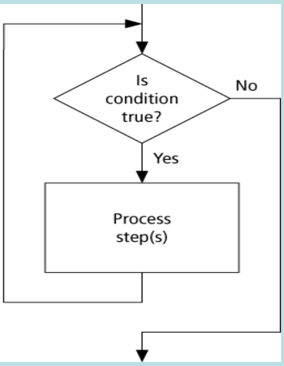
Decision flow chart

```
If condition is true Then
        Process step(s) 1
   Else
       Process step(s) 2
End If
```



Looping flow chart

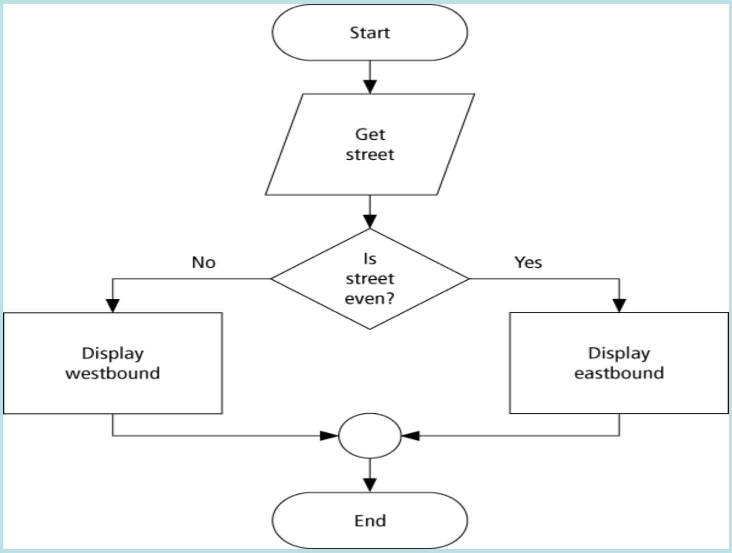
```
Do While condition is true
    Process step(s)
Loop
```



Direction of Numbered Moscow Streets Algorithm

- ***Problem:*** Given a street number of a one-way street in Moscow, decide the direction of the street, either eastbound or westbound
- ***Discussion:*** in Moscow even numbered streets are Eastbound, odd numbered streets are Westbound

Flowchart



Pseudo code

Program: Determine the direction of a numbered Moscow street

```
Get street
If street is even Then
  Display Eastbound
Else
  Display Westbound
End If
```

Class Average Algorithm

Problem: Calculate and report the grade-point average for a class

Discussion: The average grade equals the sum of all grades divided by the number of students

Output: Average grade

Input: Student grades

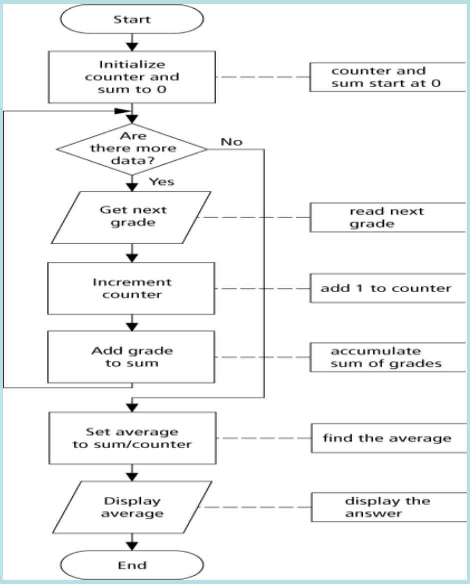
Processing: Find the sum of the grades; count the number of students; calculate average

Pseudo code

Program: Determine the average grade of a class

```
Initialize Counter and Sum to 0
Do While there are more data
    Get the next Grade
    Add the Grade to the Sum
    Increment the Counter
Loop
Computer Average = Sum /
   Counter
Display Average
```

Flowchart



12

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:

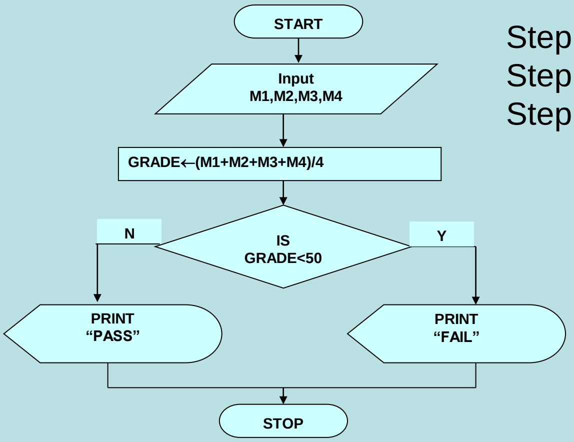
```
• Input a set of 4 marks
• Calculate their average by summing and dividing by 4
• if average is below 50
        Print "FAIL"
  else
        Print "PASS"
```

Pseudocode & Algorithm

- Detailed Algorithm
-

```
Step 1:     Input M1,M2,M3,M4
Step 2:     GRADE ← (M1+M2+M3+M4)/4
Step 3:     if (GRADE < 50) then
                 Print “FAIL”
            else
                 Print “PASS”
            endif
```

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

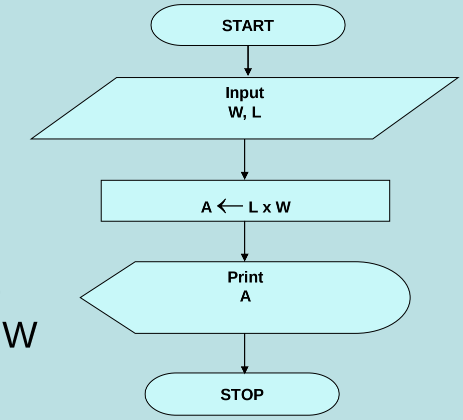
Write an algorithm and draw a flowchart that will read the two sides of a rectangle and calculate its area

Pseudo code

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

Algorithm

- Step 1: Input W,L
- Step 2: A ← L x W
- Step 3: Print A



Example 3

- 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*

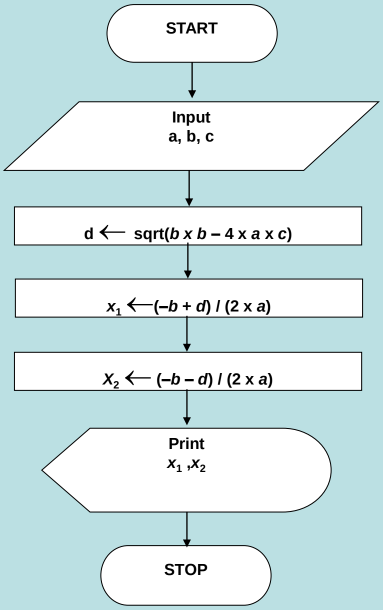
Pseudocode:

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

Algorithm:

- Step 1: Input a, b, c
- Step 2: *d* ← sqrt ($b \times b - 4 \times a \times c$)
- Step 3: *x*1 ← (–*b* + *d*) / (2 x *a*)
- Step 4: *x*2 ← (–*b* – *d*) / (2 x *a*)
- Step 5: Print *x*1, *x*2

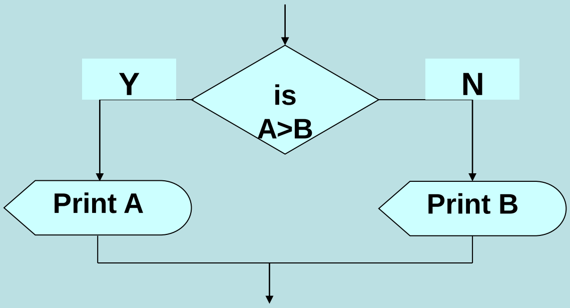
Example 3


START
Input
a, b, c
d ← sqrt(b x b – 4 x a x c)
x_1 ←(–b + d) / (2 x a)
X_2 ← (–b – d) / (2 x a)
Print
x_1 ,x_2
STOP

IF–THEN–ELSE STRUCTURE

- The algorithm is as follows:

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



Relational Operators

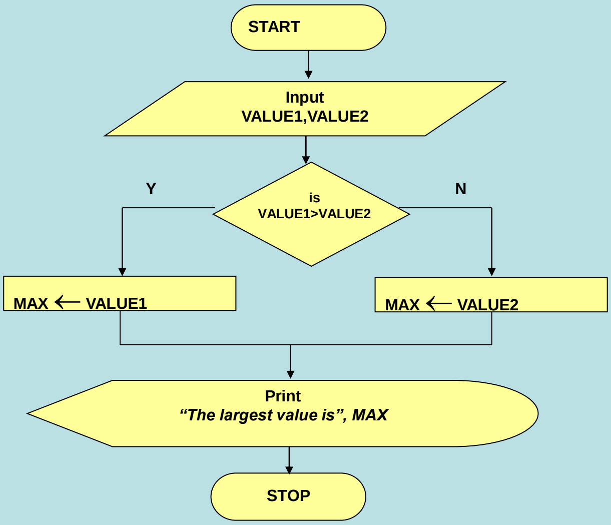
Relational Operators	
Operator	**Description**
>	Greater than
<	Less than
=	Equal to
≥	Greater than or equal to
≤	Less than or equal to
≠	Not equal to

Example 4

- 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
```



NESTED IFS

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

Example 5

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

ALGORITHM:

```
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

- 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).

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  ← 50
        else if (OVERTIME–(2/3)*ABSENT > 30) then
             PAYMENT  ← 40
        else if (OVERTIME–(2/3)*ABSENT > 20) then
             PAYMENT  ← 30
        else if (OVERTIME–(2/3)*ABSENT > 10) then
             PAYMENT  ←20
        else
             PAYMENT ← 10
        endif
Step 3: Print "Bonus for", NAME "is $", PAYMENT
```

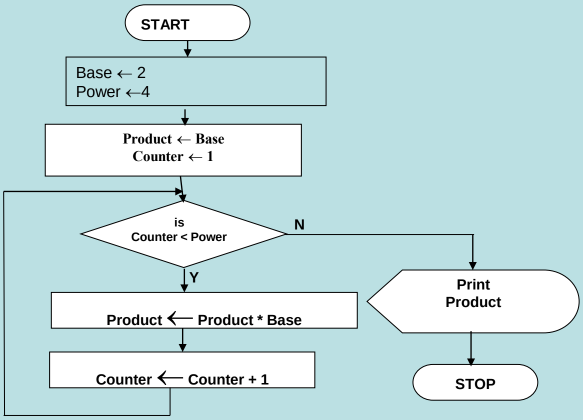
LOOPS

- Example: Write an algorithm and draw a flowchart to calculate 2^4 using a loop approach?

Algorithm:

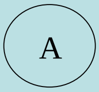
```
Step 1: Base ← 2
Step 2:   Power ← 4
Step 3: Product ← Base
Step 4:   Counter ← 1
Step 5: While Counter < Power
        Repeat Step 5 through step 7
Step 6:           Product ← Product * Base
Step 7:           Counter ← Counter +1
Step 8:           Print  Product
```

Flowchart

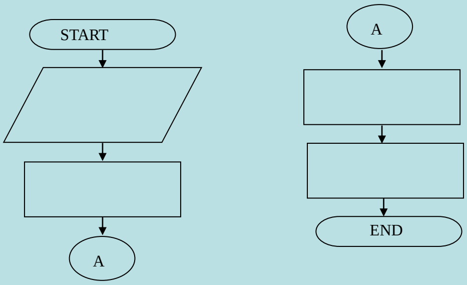


Connectors

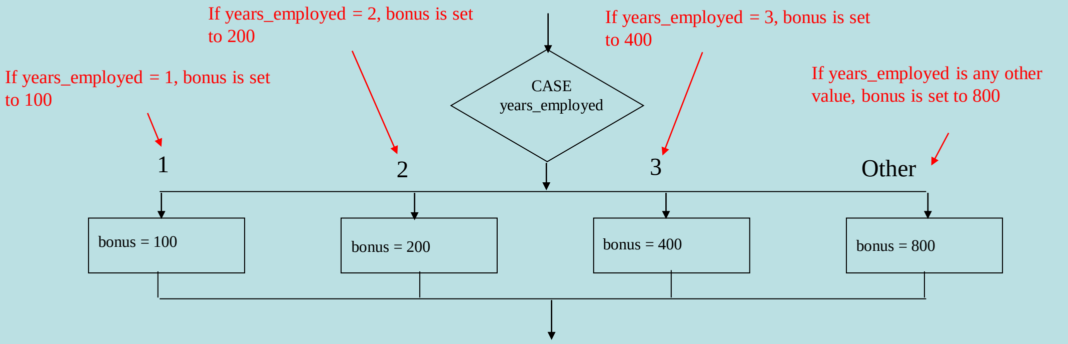
- Sometimes a flowchart will not fit on one page.
- A connector (represented by a small circle) allows you to connect two flowchart segments.



The "A" connector indicates that the second flowchart segment begins where the first segment ends.



Case Structure



If years_employed = 1, bonus is set to 100

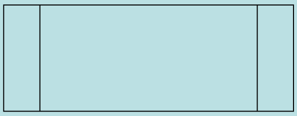
If years_employed = 2, bonus is set to 200

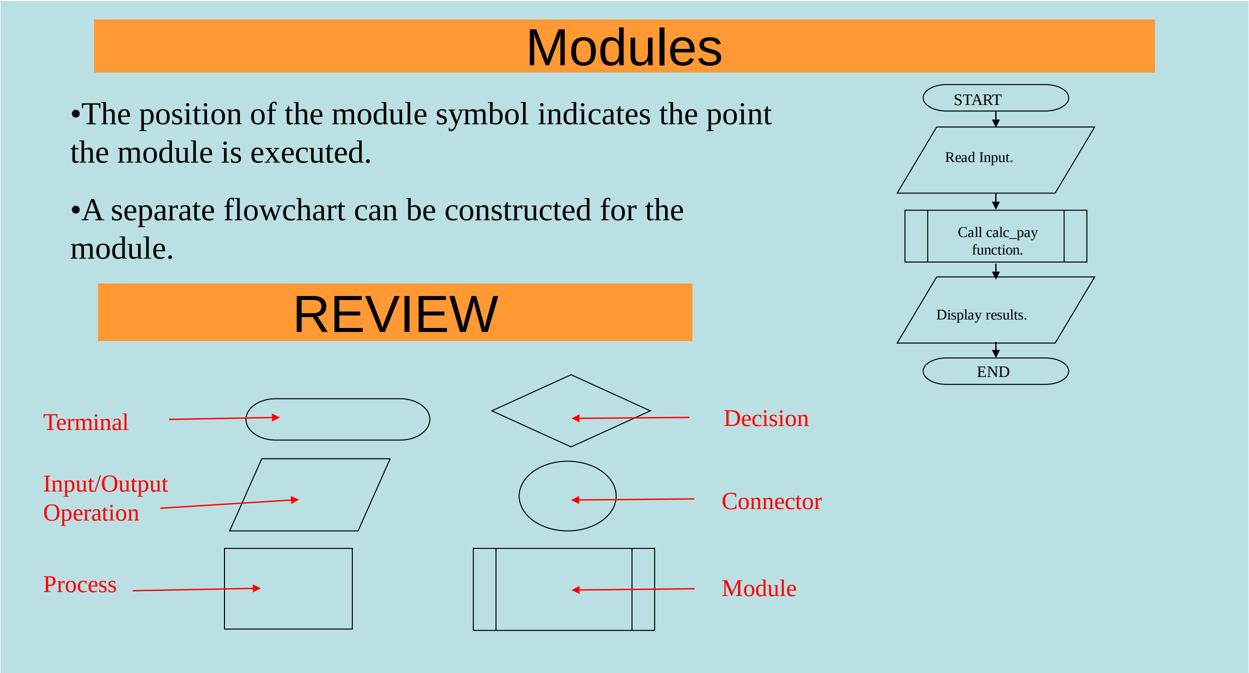
If years_employed = 3, bonus is set to 400

If years_employed is any other value, bonus is set to 800

Modules

- A program module (such as a function in C) is represented by a special symbol.





Modules

- The position of the module symbol indicates the point the module is executed.
- A separate flowchart can be constructed for the module.

START

Read Input.

Call calc_pay function.

Display results.

END

REVIEW

Terminal

Input/Output Operation

Process

Decision

Connector

Module