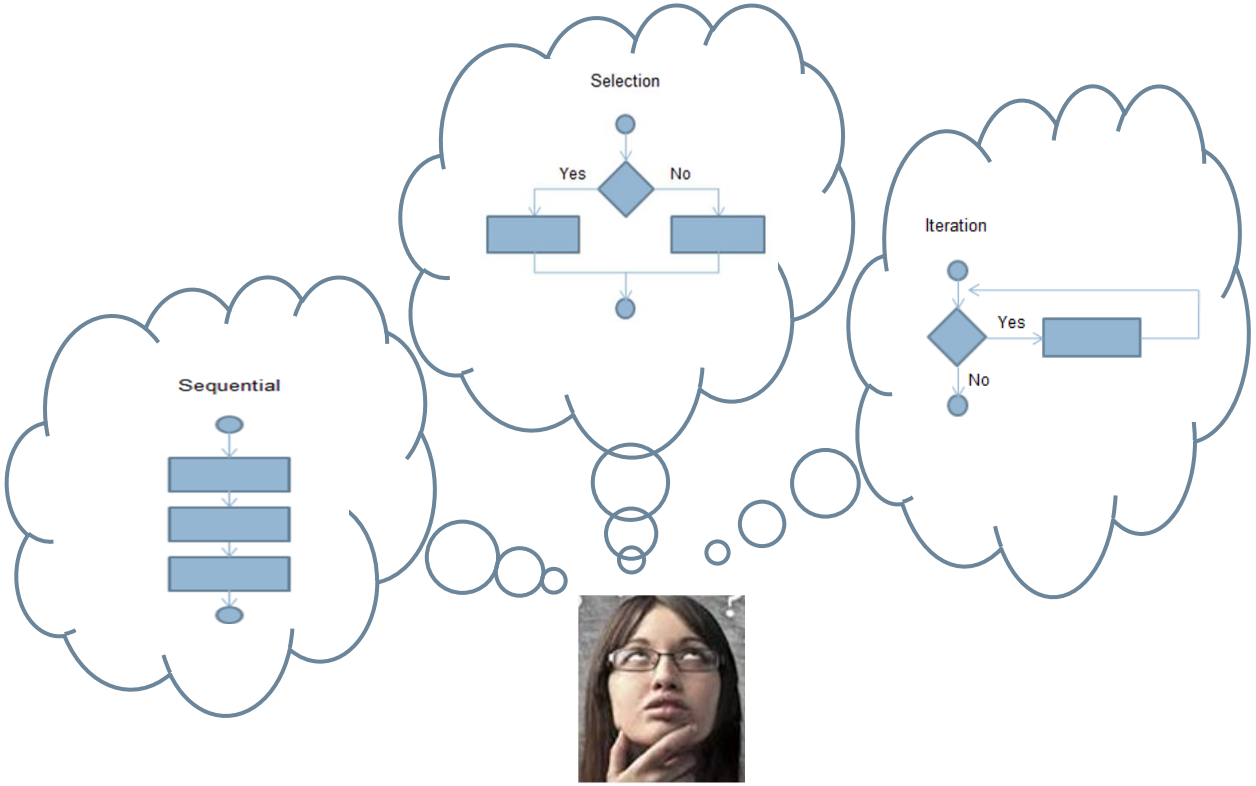
TOPIC 5
ALGORITHM DESIGN FOR REPETITION CONTROL STRUCTURE

Content

- Analysis of problems requiring repetition control structure
- Setting three requirements of a repetition structure: initialization, condition and updating
- Algorithm development for repetition control structure (pseudocode and flowchart)

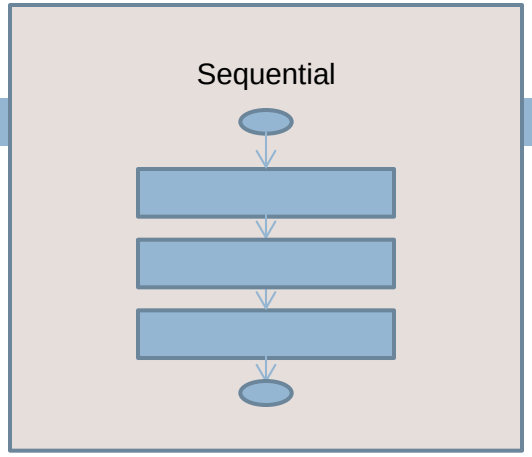
Introduction

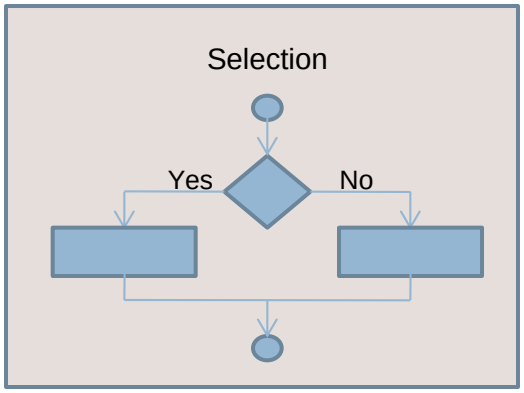
- People need to solve problems every day.
- Different problems require different control structures as problem solving techniques.
- A correct control structure is needed, in order to get intended result.

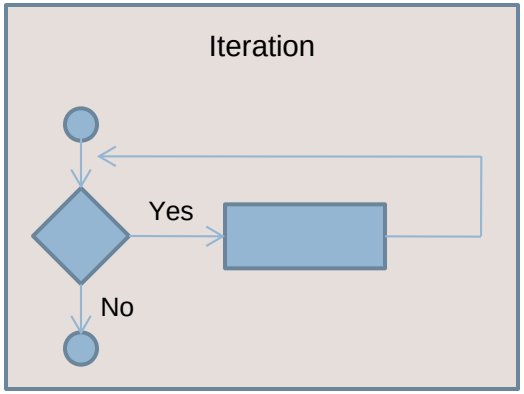


Control Structures

- Three types of control structures:
  - Sequential
    - A sequential structure is a set of sequentially executing instructions.
    - The simplest type of control structures.
  - Selection
    - A selection structure is a branch within a program based upon some condition.
  - Iteration/Loop/Repetition
    - Is a sequence of instructions that will be repeatedly executed based on some condition.
    - Sometimes known as loop or repetition.







Example of Sequential Control Structure

Example:

To calculate and display the addition, subtraction and product of two integers.

Expected Output, for the following input value:

- ✓ Two integers: 6, 5

```
Enter 2 values: 6 5
Addition: 11
Subtraction: 1
Product: 30
```

Pseudocode

```
Start
    Show "Enter 2 values: "
    Read num1, num2

    Add = num1 + num2
    Sub = num1 - num2
    Mul = num1 * num2

    Show "Addition: ", Add, newline
    Show "Subtraction: ", Sub, newline
    Show "Product: ", Mul, newline
End
```

Example of Sequential Control Structure

Example:

To calculate and display the addition, subtraction and product of two integers.

Expected Output, for the following input value:

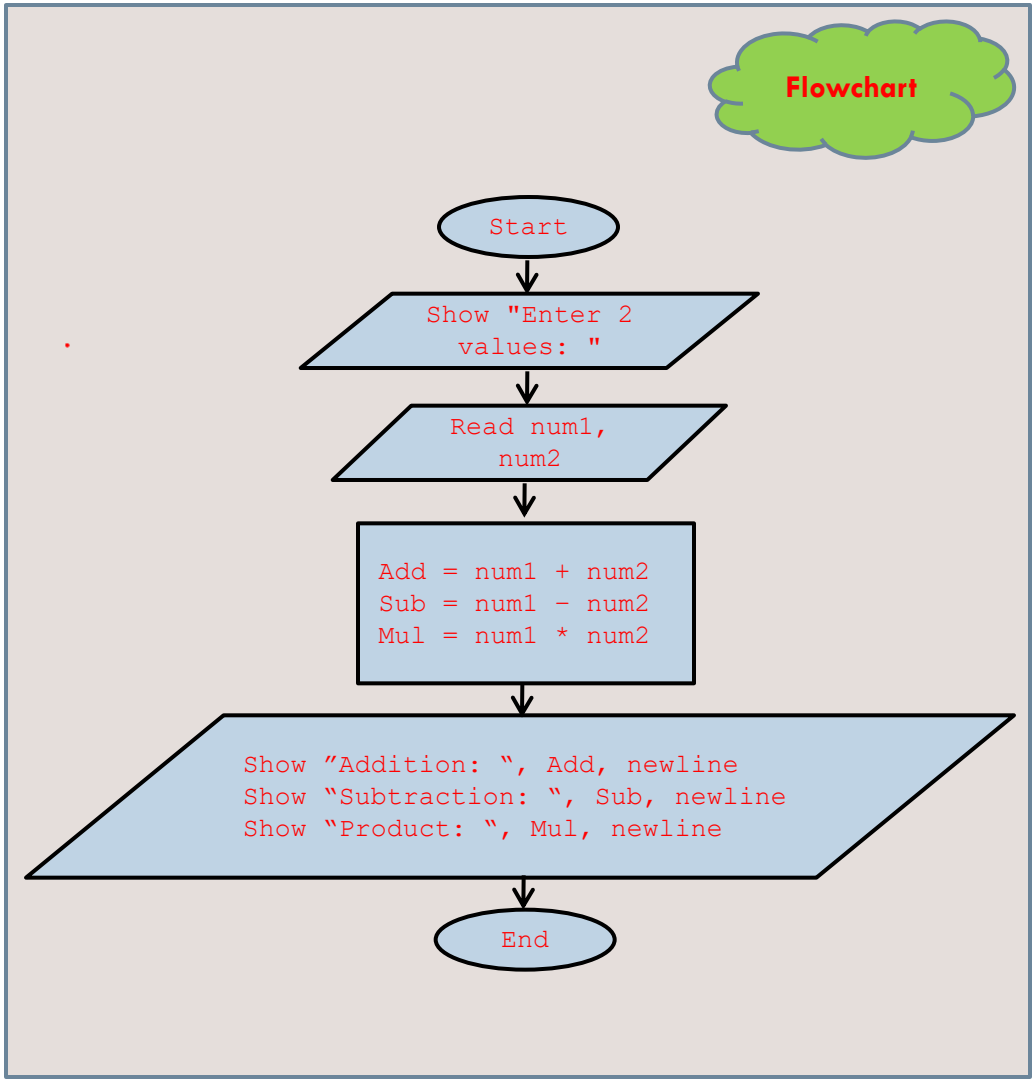
- ✓ Two integers: 6, 5

```
Enter 2 values: 6 5
Addition: 11
Subtraction: 1
Product: 30
```

Pseudocode

```
Start
    Show "Enter 2 values: "
    Read num1, num2

    Add = num1 + num2
    Sub = num1 - num2
    Mul = num1 * num2

    Show "Addition: ", Add, newline
    Show "Subtraction: ", Sub, newline
    Show "Product: ", Mul, newline
End
```

Flowchart



Example of Selection Control Structure

Example:

To calculate and display the correct answer for 2 integers based on a selected operator.

- ✓ If the operator is '+', the total of two integers is produced.
- ✓ If the operator is '-', the subtraction of two integers is produced.
- ✓ If the operator is '*', the product of two integers is produced.

Expected Output, for the following input value:

- ✓ Two integers: 6, 5; and Operator: '+'

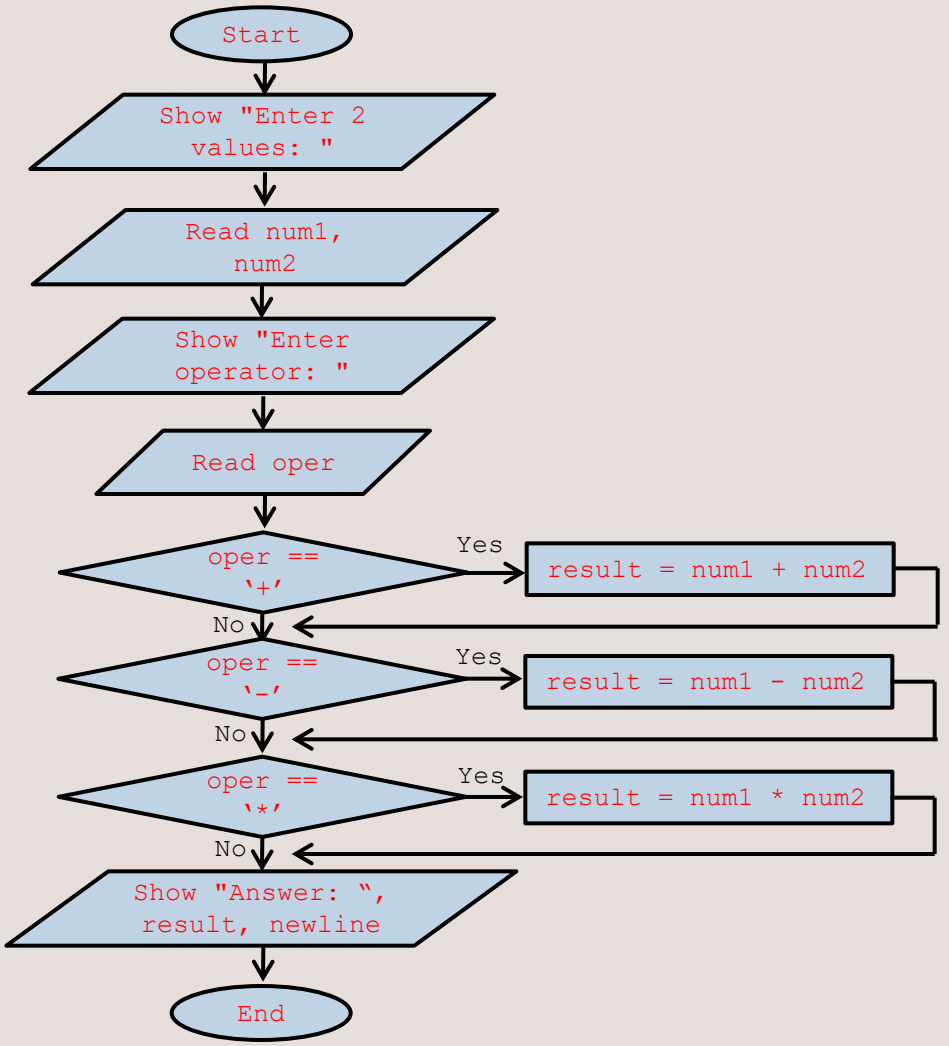
```
Enter two integers: 6 5
Enter operator: +
Answer: 11
```

Example of Selection Control Structure

Pseudocode

```
Start
    Show "Enter 2 values: "
    Read num1, num2
    Show "Enter operator: "
    Read oper

    if (oper == '+')
        result = num1 + num2
    endIf
    if (oper == '-')
        result = num1 - num2
    endIf
    if (oper == '*')
        result = num1 * num2
    endIf
    Show "Answer: ", result, newline
End
```

Flowchart



Example of Iteration Control Structure

Example:

To calculate and display the correct operation based on a selected operator.

- ✓ If the operator is '+', the total of two integers is produced.
- ✓ If the operator is '-', the subtraction of two integers is produced.
- ✓ If the operator is '*', the product of two integers is produced.

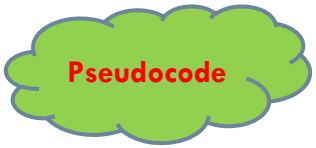
Repeat the process until the user enters n to terminate.

Expected Output, for the following input value:

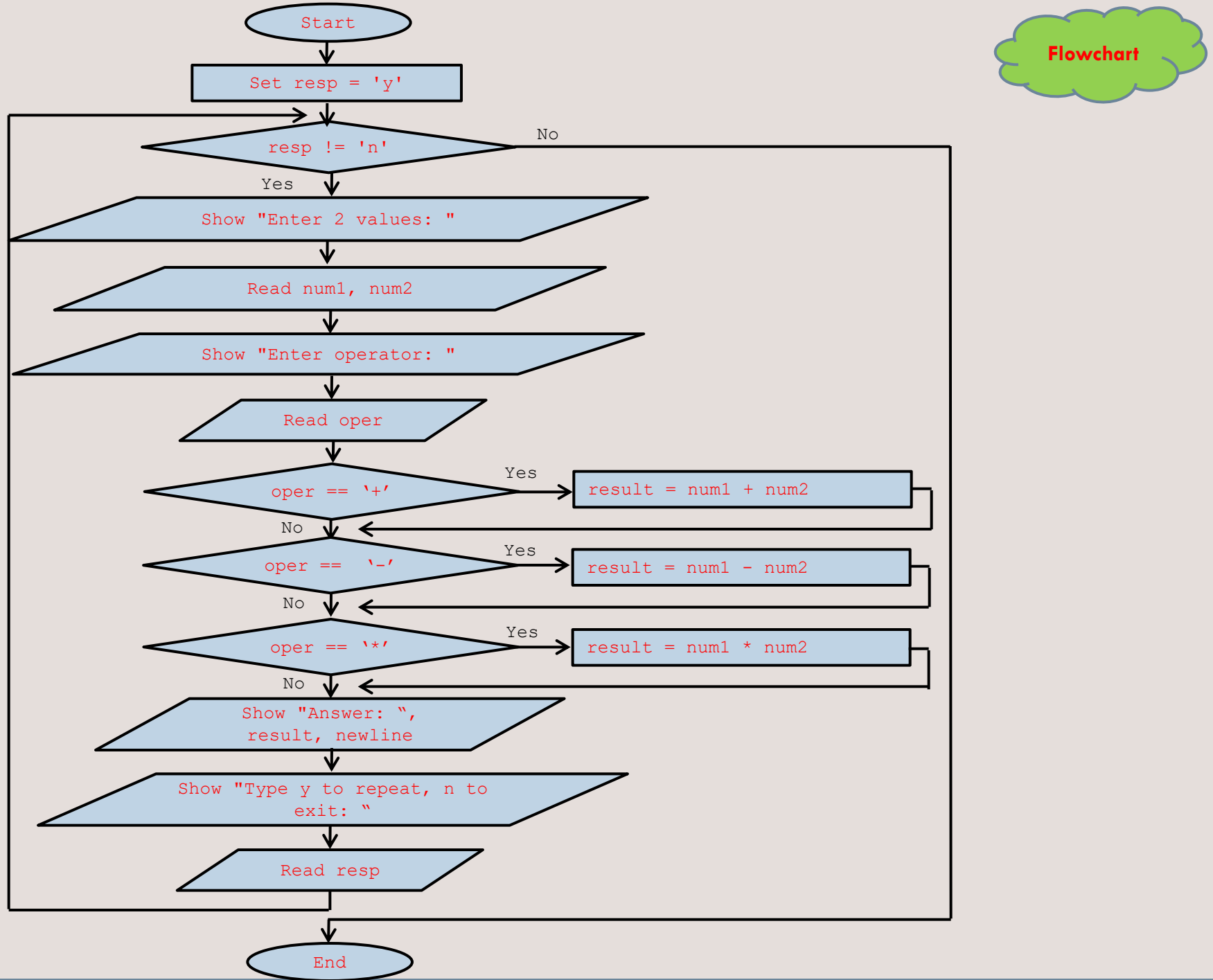
- ✓ Two integers: 4, 5; and Operator: '*'
- ✓ Two integers: 3, 2; and Operator: '-'
- ✓ Two integers: 6, 7; and Operator: '+'

```
Enter two integers: 4 5
Enter operator: *
Answer: 20

Type y to repeat, n to exit: y
Enter two integers: 3 2
Enter operator: -
Answer: 1

Type y to repeat, n to exit: y
Enter two integers: 6 7
Enter operator: +
Answer: 13

Type y to repeat, n to exit: n
```

Example of Iteration Control Structure




```
Start
  Set resp = 'y'
  while (resp != 'n')
    Show "Enter two values: "
    Read num1, num2
    Show "Enter operator: "
    Read oper
    if (oper == '+')
      result = num1 + num2
    endIf
    if (oper == '-')
      result = num2 - num2
    endIf
    if (oper == '*')
      result = num1 * num2
    endIf

    Show "Answer: ", result, newline, newline
    Show "Type y to repeat, n to exit: "
    Read resp
  EndWhile
End
```


Flowchart
Start
Set resp = 'y'
resp != 'n'
No
Yes
Show "Enter 2 values: "
Read num1, num2
Show "Enter operator: "
Read oper
oper == '+'
Yes
result = num1 + num2
No
oper == '-'
Yes
result = num1 - num2
No
oper == '*'
Yes
result = num1 * num2
No
Show "Answer: ",
result, newline
Show "Type y to repeat, n to
exit: "
Read resp
End

Iteration

- Also known as repetition or looping
- Statements are executed repeatedly while certain condition remains true.
- Requirement:
  - Loop control variable (LCV)
    - A variable whose value determines whether the loop body will be executed or not
  - Loop condition
    - If the condition is TRUE, the loop body is executed; OTHERWISE the loop exits
  - Loop body
    - A block of statements to be repeated
- The execution of the loop body is controlled by 3 operations:
  1. Initialization of LCV
  2. Evaluation of LCV in the loop condition
  3. Updating of LCV
     - Incrementing
     - Decrementing

Iteration

- The variations of iteration structure are:
  - While
  - Do..While
  - For
- The **while** iteration statement checks the condition first.
  - If the condition is true, the loop body is executed;
  - otherwise the loop terminates.
- The **do while** iteration statement resembles the **while** loop BUT,
  - the loop body is executed first and then;
  - checks the condition to decide whether to continue or terminate.
- The **while** and **for** iteration statements are called as pretest loops because the condition is checked before the loop body is executed.
- The **do..while** iteration statement is called posttest loop because the condition is checked after the loop body is executed.
- All the iteration statements are expressively equivalent EXCEPT in certain special cases.
- All the iteration statements (**while, do..while, for**) often are used when the number of repetitions is specified.
- The **while** and the **do while** iteration statements often are used when the number of repetitions is unspecified.

Iteration Statements (Pseudocode)

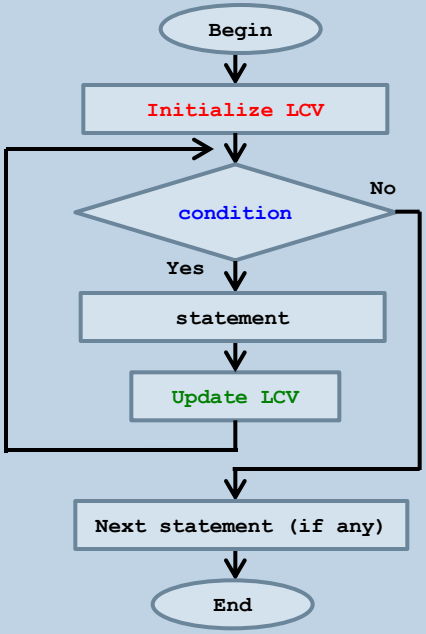
While	Do .. While	For
Pseudocode:		

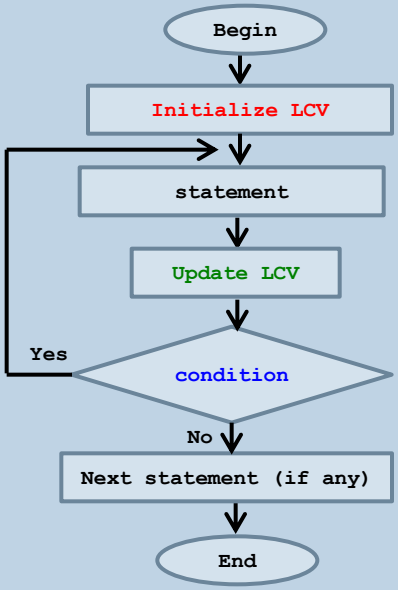
Begin
  initialize LCV
  while (loop condition)
  Statement(s)
  update LCV
  Endwhile
  Next statement //if any
End
	Begin
	  initialize LCV
	  do
	  Statement(s)
	  update LCV
	  while (loop condition)
	  Next statement //if any
	End
		Begin
		  for initial value to final value, update LCV
		  Statement(s)
		  Endfor
		  Next statement //if any
		End

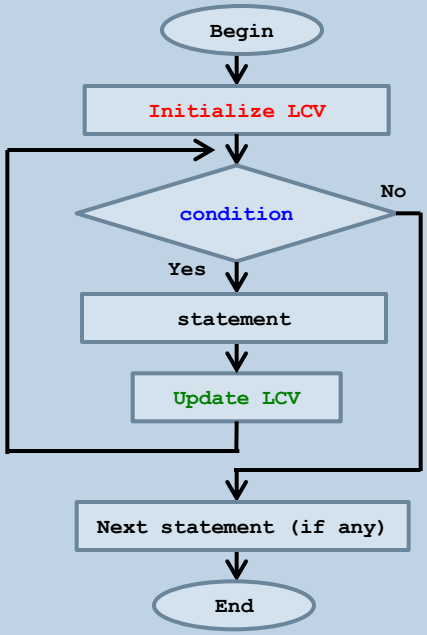
- initialize LCV (Loop Control Variable) → Refers to the initial value (if any) given to a loop variable or counter
- loop condition → Determine whether the loop execution should be continued
  - If condition false – the loop terminated
  - If condition true – the loop executed
- update LCV → increments (or decrements) the loop variable

Iteration Statements (Flowchart)

While	Do .. While	For
Flowchart:		







Iteration Statements (Pseudocode): Example

While	Do..While	For
Example 1: To display ALL numbers from 15 to 25		

Begin
  Set value = 15
  while (value <= 25)
  Display value, newline
  value++
  Endwhile
  Display "**End of Program**"
End
	Begin
	  Set value = 15
	  do
	  Display value, newline
	  value++
	  while (value <= 25)
	  Display "**End of Program**"
	End
		Begin
		  for value = 15 to 25, Step 1
		  Display value, newline
		  value++
		  Endfor
		  Display "**End of Program**"
		End

Output: 15 16 17 18 19 20 21 22 23 24 25 **End of Program**		

Iteration Statements (Pseudocode): Example

For Loop

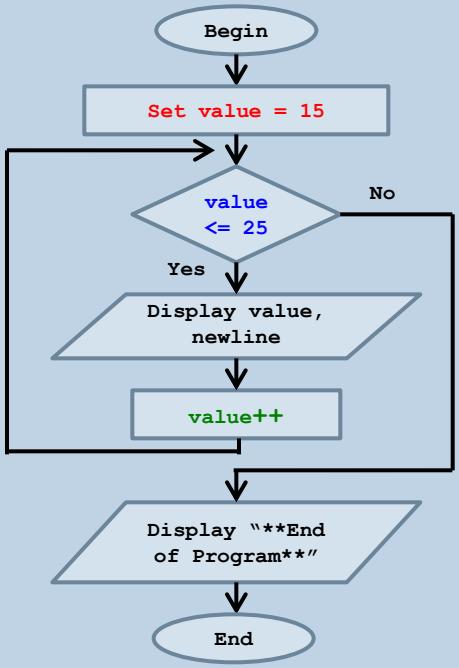
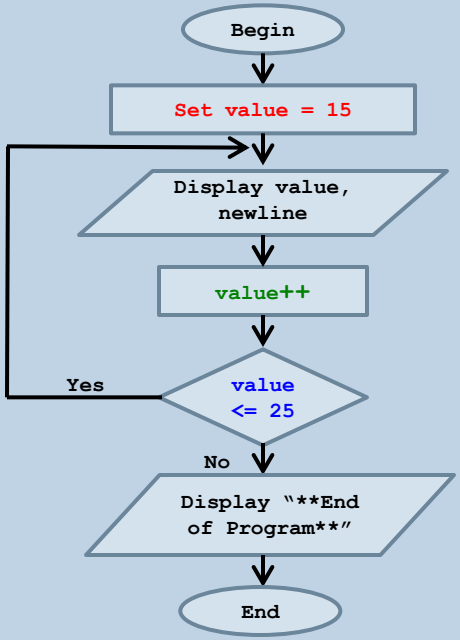
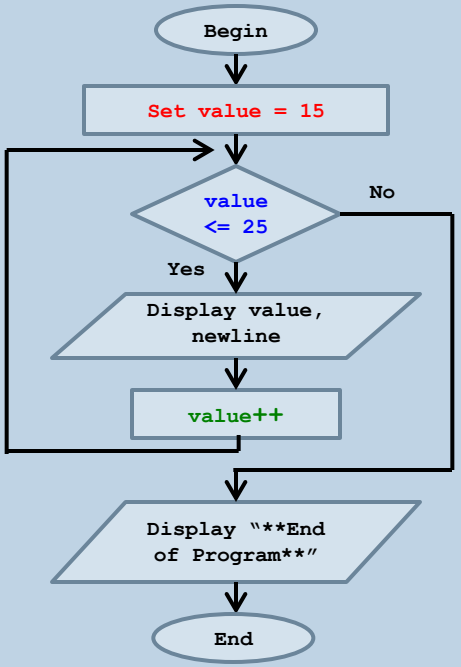
```
Begin
  for value = 15 to 25, Step 1
        Display value, newline
        value++
  Endfor
  Display "**End of Program**"
End
```

Step value define the how many the control variable will increase or decrease

Step 2 will increase the value of CV by 2

```
Begin
  for value = 15 to 25, Step 1
      Display value, newline
  Endfor
  Display "**End of Program**"
End
```

Iteration Statements (Flowchart): Example

While	Do..While	For
Flowchart:		
		

Types of Iteration Structure

- Counter-controlled loop
  - Used when the number of repetition is known in advance
  - Example: Repeat a process for 100 times

- Sentinel-controlled loop (Event-controlled loop)
  - Used when the number of repetition is not known in advance
  - The number of repetition depends on a certain condition, called as sentinel value that serves as a signal for loop termination.
  - Example: Enter a series of numbers, ends with 0 (Note: 0 is a sentinel value). If the input value is 0, the program will be terminated.

Counter Control Loop vs Sentinel Control Loop: Example

Counter Control Loop	Sentinel Control Loop
Example 1: A pseudocode for a program that prompt the user to enter 5 integers. The program has to do the following calculation and print the results: - Calculate the total of all numbers - Calculate the average of all numbers - Count a number of odd and even numbers	Example 1: A pseudocode for a program that prompt the user to enter a series of numbers, ends with 999. The program has to do the following calculation and print the results: - Calculate the total of all numbers - Calculate the average of all numbers - Count a number of odd and even numbers
Expected Output for input values: 2, 3, 1, 5, 6 Enter value: 2 Enter value: 3 Enter value: 1 Enter value: 5 Enter value: 6 Sum: 17 Average: 3.4 Even: 2 Odd: 3	Expected Output for input values: 2, 3, 1, 5, 6 Enter value: 2 Enter value: 3 Enter value: 1 Enter value: 5 Enter value: 6 Enter value: 999 Sum: 17 Average: 3.4 Even: 2 Odd: 3

Counter Control Loop vs Sentinel Control Loop: Pseudocode

Counter Control Loop

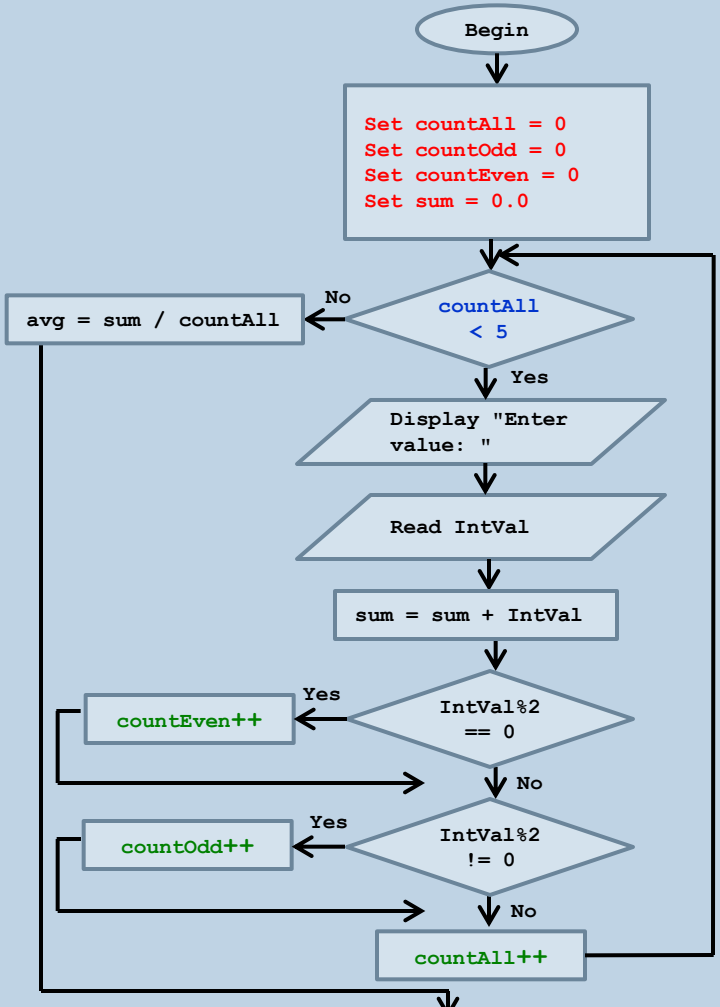
```
Begin
   Set countAll = 0
   Set countOdd = 0
   Set countEven = 0
   Set sum = 0.0

   while (countAll < 5)
      Display "Enter value: "
      Read IntVal

      sum = sum + IntVal
      if (IntVal%2 == 0)
            countEven++
      endIf
      if (IntVal%2 != 0)
            countOdd++
      endIf

      countAll++
   endWhile
   avg = sum / countAll
   Display "Sum: ", sum, newline
   Display "Average: ", avg, newline
   Display "Even: ", countEven, newline
   Display "Odd: ", countOdd, newline
End
```

Sentinel Control Loop

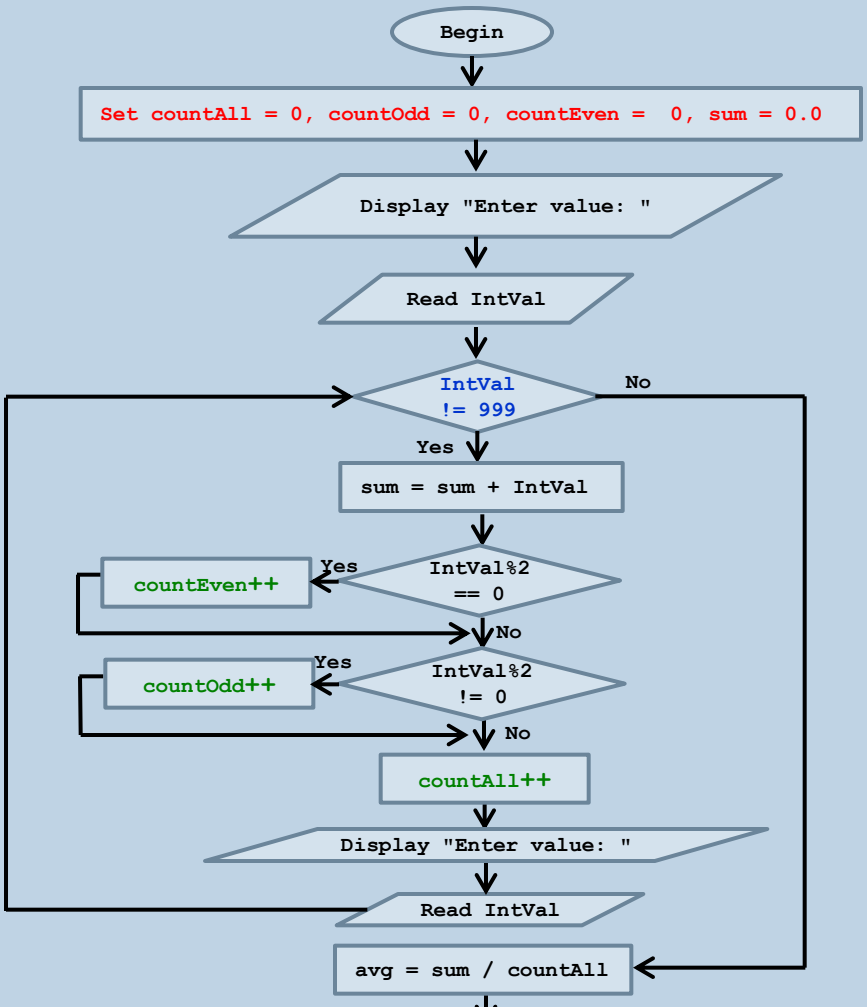
```
Begin
   Set countAll = 0
   Set countOdd = 0
   Set countEven = 0
   Set sum = 0.0

   Display "Enter value: ";
   Read IntVal

   while (IntVal != 999)
         sum = sum + IntVal
         if (IntVal%2 == 0)
               countEven++
         endIf
         if (IntVal%2 != 0)
               countOdd++
         endIF

         countAll++
         Display "Enter value: "
         ReadIntVal
   endWhile
   avg = sum / countAll
   Display "Sum: ", sum, newline
   Display "Average: ", avg, newline
   Display "Even: ", countEven, newline
   Display "Odd: ", countOdd, newline
End
```

Counter Control Loop vs Sentinel Control Loop: Flowchart

Counter Control Loop



Sentinel Control Loop



Nested Loop

- A loop within another a loop

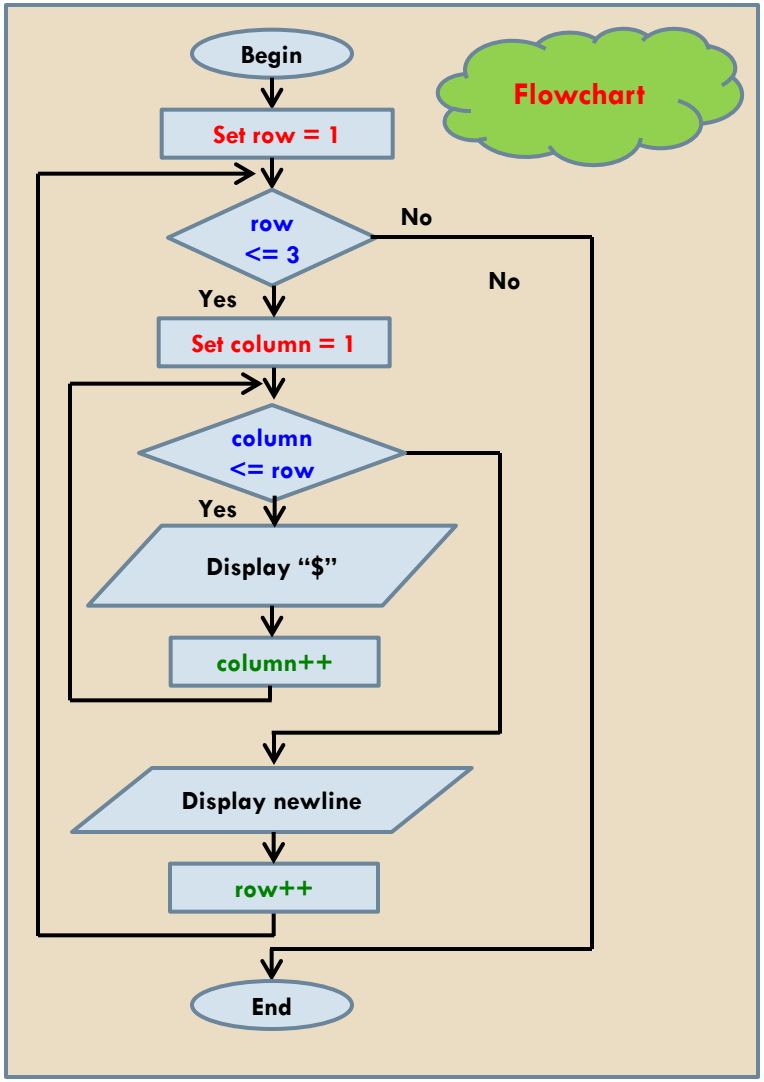
Pseudocode

Example:

Outer loop

```
Start
    for row = 1 to 3, Step 1
        for column = 1 to row, Step 1
            Display "$"
        Display newline
    endFor
End
```

Inner loop



Nested Loop: Output Tracing

row	row <= 3	col	col <= row	Display
1	1<=3 → yes	1	1<=1 → yes	$
		2	2<=1 → no	-
				newline
2	2<=3 → yes	1	1<=2 → yes	$
		2	2<=2 → yes	$
		3	3<=2→ no	-
				newline
3	3<=3 → yes	1	1<=3 → yes	$
		2	2<=3 → yes	$
		3	3<=3 → yes	$
		4	4<=3 → no	-
				newline
4	4<=3 → no	-		

Expected Output:

$
$ $
$ $ $