










Algorithms:

An algorithm is a sequence of steps done to perform some task.

-  The essential aim of an algorithm is to get a specific output,
-  An algorithm involves with several continuous steps,
-  The output comes after the algorithm finished the whole process.

So basically, all algorithms perform logically while following the steps to get an output for a given input.







Types of Algorithms:

-  Structure Diagrams
-  Flowcharts
-  Pseudo codes
-  Program Code

FLOWCHARTS:

Flow chart is a graphical representation of a program.

Flowcharts use different symbols containing information about steps or a sequence of events.

Symbol	Name	Usage
	Line	Represents the flow from one component to the next
	Process	An action
	Subroutine	Calls a subroutine
	Input/Output	An input or output
	Decision	A yes/no/true/false decision
	Terminator	The start or end of the process



PSEUDOCODE:

Pseudo code is an outline of a program, written as a series of instruction using simple English sentences. Pseudo code uses keywords commonly found in high-level *languages* and mathematical notation.

Variable:

Variable is named memory location or data holder whose value can change during running of a program. Variable must have a **meaningful name** and a **Data Type**. Variables must be **DECLARED** or created before they are used in the program.

Variables are declared as follows:

DECLARE variablename : **Data Type**

DECLARE name : **STRING**

DECLARE marks : **INTEGER**

Constants:

Just like variables, constants are "dataholders". In contrast to variable, the content of a constant can't change at runtime, it has a constant value. Constants are also declared with a value before they are used in the program.

CONSTANT <identifier> = <Value>

CONSTANT Pi ← 3.1415 or **CONSTANT** Pi = 3.14

Arithmetic operations: Use the arithmetic operators.

Operator	Comparison
+	Addition
-	Subtraction
*	Multiplication
/	Division
=	Equals to
<>	Not equal



Logical operations:

Operator	Comparison
>	Greater than
<	Less than
>=	Greater than equal to
<=	Less than equal to
=	Equals to
<>	Not equal
()	Group
AND	And
OR	Or
NOT	Not

Assignment

Assignment is the process of writing a value into a variable (a named memory location). For example, **Count** ← 1 can be read as 'Count is assigned the value 1'

Initialization:

The value of variable can be initialized before user's input. If an algorithm needs to read the value of a variable *before* it assigns input data or a calculated value to the variable, the algorithm should assign an appropriate initial value to the variable, known as Initialization.

Data types

The following table shows the data types:

A variable can store one type of data. The data types are:

Data Type	Explanation
INTEGER	Stores whole numbers e.g. 1,22, -78 etc
REAL	Stores decimal numbers e.g. 3.5, - 7.6
STRING	Stores text e.g. name, address in text
BOOLEAN	Stores True or False
CHAR	Stores a single character e.g. F for female
DATE	Stores date e.g. 12/02/2024



Input

In **Pseudocodes** we indicate input by keywords such as **INPUT**, **READ** or **ENTER**, followed by the name of a variable to which we wish to assign the input value.

Output:

In **Pseudocodes** we indicate output by words such as **OUTPUT**, **WRITE** or **PRINT**, followed by a comma-separated list of expressions.

Totaling

To keep a running total, we can use a variable such as Total or Sum to hold the running total and assignment statements such as:

Total ← **Total + Number** (ADD Number to Total)

Counting

It is sometimes necessary to count how many times something happens.

To count up or increment by 1, we can use statements such as:

Count ← **Count + 1** INCREMENT Count by 1

Structured statements

In the sequence structure the processing steps are carried out one after the other. The instructions are carried out in sequence, unless a selection or loop is encountered.

Declaration of Variables and Constant:

The process of creating a variable is called declaring a variable. Variables must be created or declared where users enter their data.

Pseudo code

```

BEGIN
DECLARE variable : Datatype

Variable ← 0 //initialization

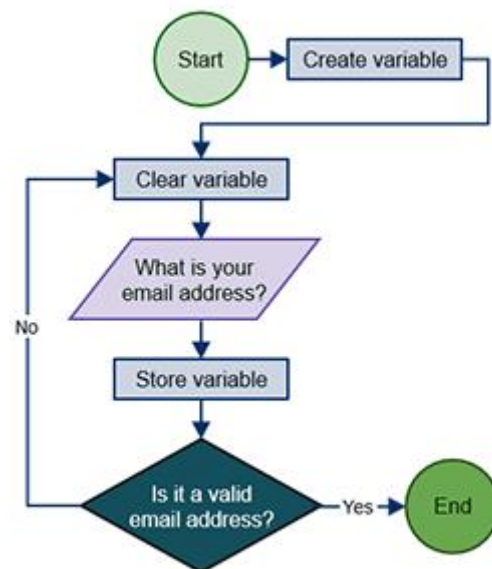
OUTPUT ("What is your Email address")
INPUT variable value

IF valid email address?

    Then ...

END
  
```

Each declaration needs 4 things:





Pseudo code

- **DECLARE** keyword
- Variable name
- : keyword
- Variable data type

DECLARE variable : Datatype

VB code example:

- **DIM** keyword
- Variable name
- **AS** keyword
- Variable data type

Dim mark As Integer

Declaring Multiple Variables:

Pseudocodes

```
DECLARE index : Integer
DECLARE grade : Integer
DECLARE counter : Integer
```

VB Code Console Mode

```
Dim index As Integer
Dim grade As Integer
Dim counter As Integer
```

The three declarations above can be rewritten as one declaration if same data type is used:

DECLARE index, grade, counter : **Integer**

```
Dim index, grade, counter As Integer
```

Constants

Creating Constants in Pseudocode is just writing constant name and value with it. In contrast to variable, the content of a constant can't change at runtime, it has a constant value.

CONSTANT <identifier> = <Value>

CONSTANT Pi ← 3.1415 or **CONSTANT** Pi = 3.14

```
Const pi As Double = 3.1415
'create a constant called pi with a value 3.1415
Dim radius As Double = 10
'creates a constant called radius with a value 10
Dim circumference As Double = radius * 2 * pi
'Creates a constant with a calculation
Dim area As Double = radius ^ 2 * pi
'creating a constant with a calculation
Console.WriteLine("Circle Circumference : " & circumference)
Console.WriteLine("Circle Area : " & area)

Console.ReadLine()
```

Type of Programs:

- Sequence
- Selection
- Repetitions/Loops



Sequence

Statements are followed in sequence so the order of the statements in a program is important.

Assignment statements rely on the variables used in the expression on the right-hand side of the statement all having been given values. Input statements often provide values for assignment statements. Output statements often use the results from assignment statements.

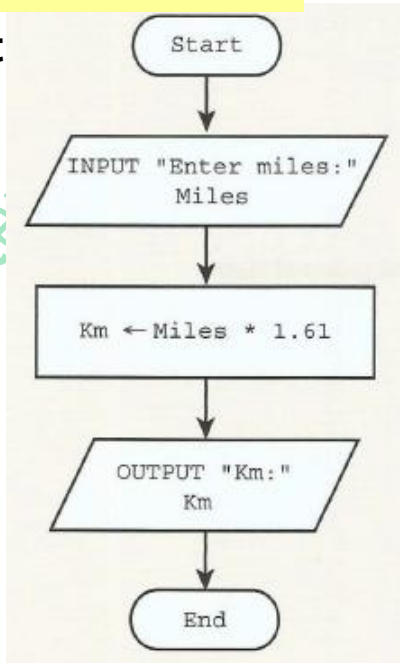
VB code example

```
Sub Main()  
Dim number1 As Integer  
Dim number2 As Integer  
Dim sum As Integer  
Dim product As Integer  
  
Console.WriteLine("Enter number 1")  
number1 = Console.ReadLine()  
  
Console.WriteLine("Enter number 2")  
number2 = Console.ReadLine()  
  
sum = number1 + number2  
product = number1 * number2  
  
Console.Write("the sum is ")  
Console.WriteLine(sum)  
  
Console.Write("the product is ")  
Console.WriteLine(product)  
  
Console.ReadLine()  
  
End Sub
```

PSEUDOCODE

```
BEGIN  
DECLARE number1 : Integer  
DECLARE number2 : Integer  
DECLARE sum : Integer  
DECLARE product : Integer  
  
PRINT ("Enter number 1")  
INPUT number1  
PRINT ("Enter number 2")  
INPUT number2  
Sum ← number1 + number2  
product ← number1 * number2  
PRINT ("the sum is", sum)  
PRINT ("the product is", product)  
END
```

Flowchart



Pseudocode



```
INPUT "Enter miles:" Miles  
Km ← Miles * 1.61  
OUTPUT "km:" Km
```

```
BEGIN  
DECLARE miles, km : REAL  
  
OUTPUT ("Enter miles")  
INPUT miles  
km ← miles * 1.61  
OUTPUT ("Km are : " & km)  
END
```




Structured statements for selection (conditional statements)

These statements are used to select alternative routes through an algorithm; selection's logical expressions often involve comparisons, which can operate on text strings as well as numbers

-  IF...THEN...ELSE...ENDIF
-  CASE...OF...OTHERWISE...ENDCASE

IF...THEN...ELSE...ENDIF

For an IF condition the THEN path is followed if the condition is true and the ELSE path is followed if the condition is false.

There may or may not be an ELSE path. The end of the statement is shown by ENDIF.

A condition can be set up in different ways:

```
IF ((Height > 1) OR (Weight > 20) OR (Age > 5)) AND (Age < 70)
  THEN
  PRINT ("You can ride")

  ELSE
  PRINT ("Too small, too young or too old")
ENDIF
```

CASE ... OF ... OTHERWISE ... ENDCASE

For a CASE condition the value of the variable decides the path to be taken. Several values are usually specified. OTHERWISE is the path taken for all other values. The end of the statement is shown by ENDCASE.

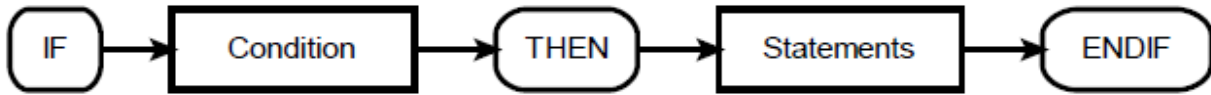
The algorithm below specifies what happens if the value of Choice is 1, 2, 3 or 4.

CASE Choice OF

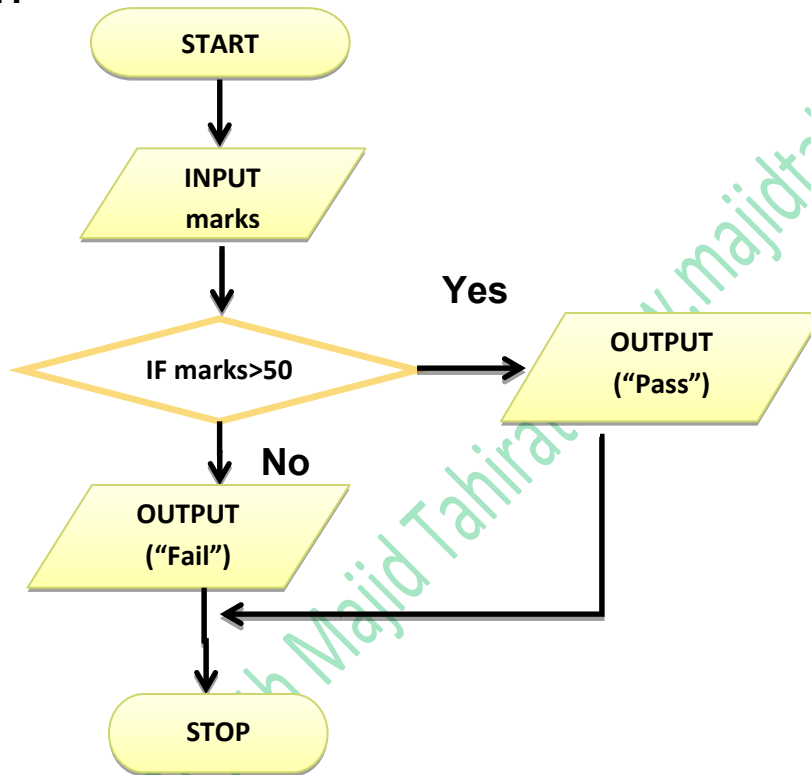
```
1: Answer ← Num1 + Num2
2: Answer ← Num1 - Num2
3: Answer ← Num1 * Num2
4: Answer ← Num1 / Num2

  OTHERWISE PRINT ("Please enter a valid choice")
ENDCASE
```

The IF THEN ELSE statement



FLOWCHART:



PSEUDOCODE

```

BEGIN
DECLARE marks : Integer

PRINT ("Enter your grade")
INPUT marks
IF marks > 50
    THEN PRINT ("You've passed")
    ELSE PRINT ("You've failed")
END IF

END
  
```

VB Code

```

Sub Main()
    Dim grade As Integer

    Console.WriteLine("Enter your grade")
    grade = Console.ReadLine()

    If grade > 50 Then
        Console.WriteLine("You have passed")
    Else
        Console.WriteLine("You have failed")
    End If

    Console.ReadLine()
End Sub
  
```


IF THEN, ELSE-IF statements

VB code example

```

BEGIN
DECLARE marks : INTEGER
PRINT ("Enter marks")
INPUT marks
IF marks >= 80
THEN PRINT ("Grade A")
ELSE IF marks >= 60
THEN PRINT ("Grade B")
ELSE IF marks >= 60
THEN PRINT ("Grade C")
ELSE PRINT ("Grade U")
END IF
END IF
END IF
END
    
```

```

Sub Main()
Dim grade As Integer

Console.WriteLine("Enter a grade")
grade = Console.ReadLine

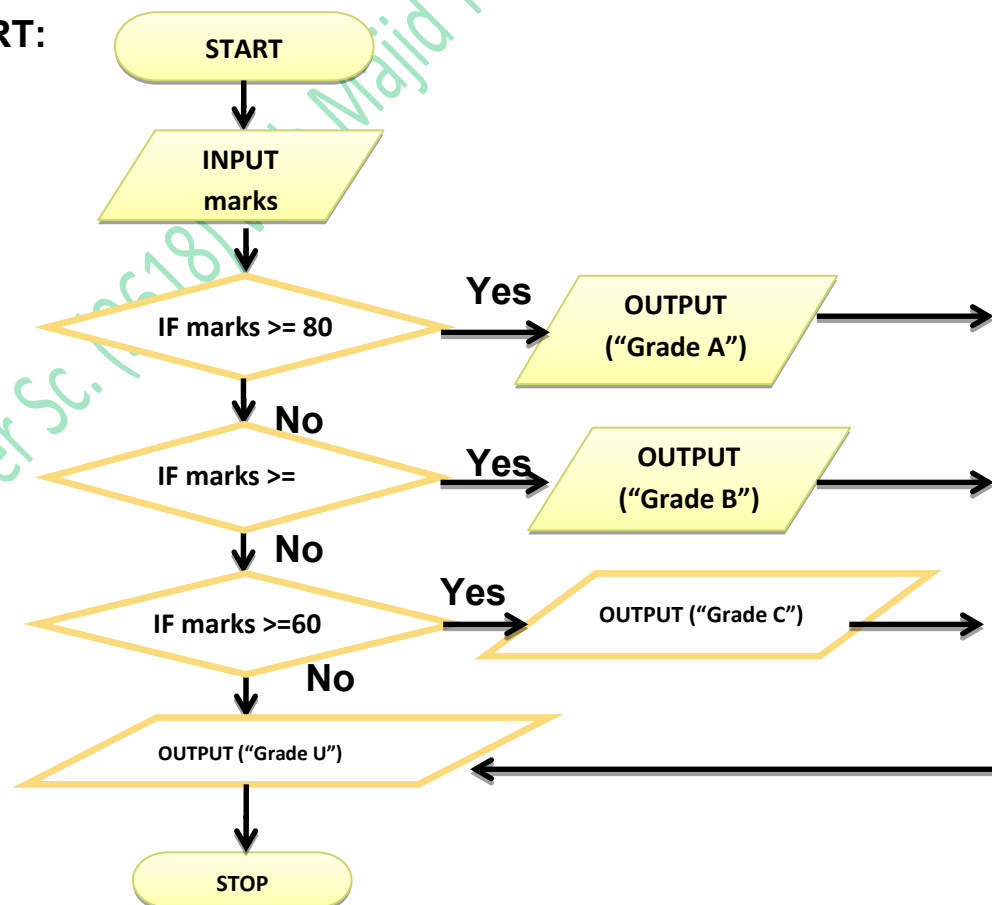
If grade > 80 Then
    Console.WriteLine("Grade A")
ElseIf grade > 60 Then
    Console.WriteLine("Grade B")
ElseIf grade > 50 Then
    Console.WriteLine("Grade C")
Else
    Console.WriteLine("Grade U")
End If

Console.ReadLine()

End Sub
    
```

The IF statement is useful, but can get clumsy if you want to consider “multi-way selections

FLOWCHART:



CASE OF OTHERWISE...

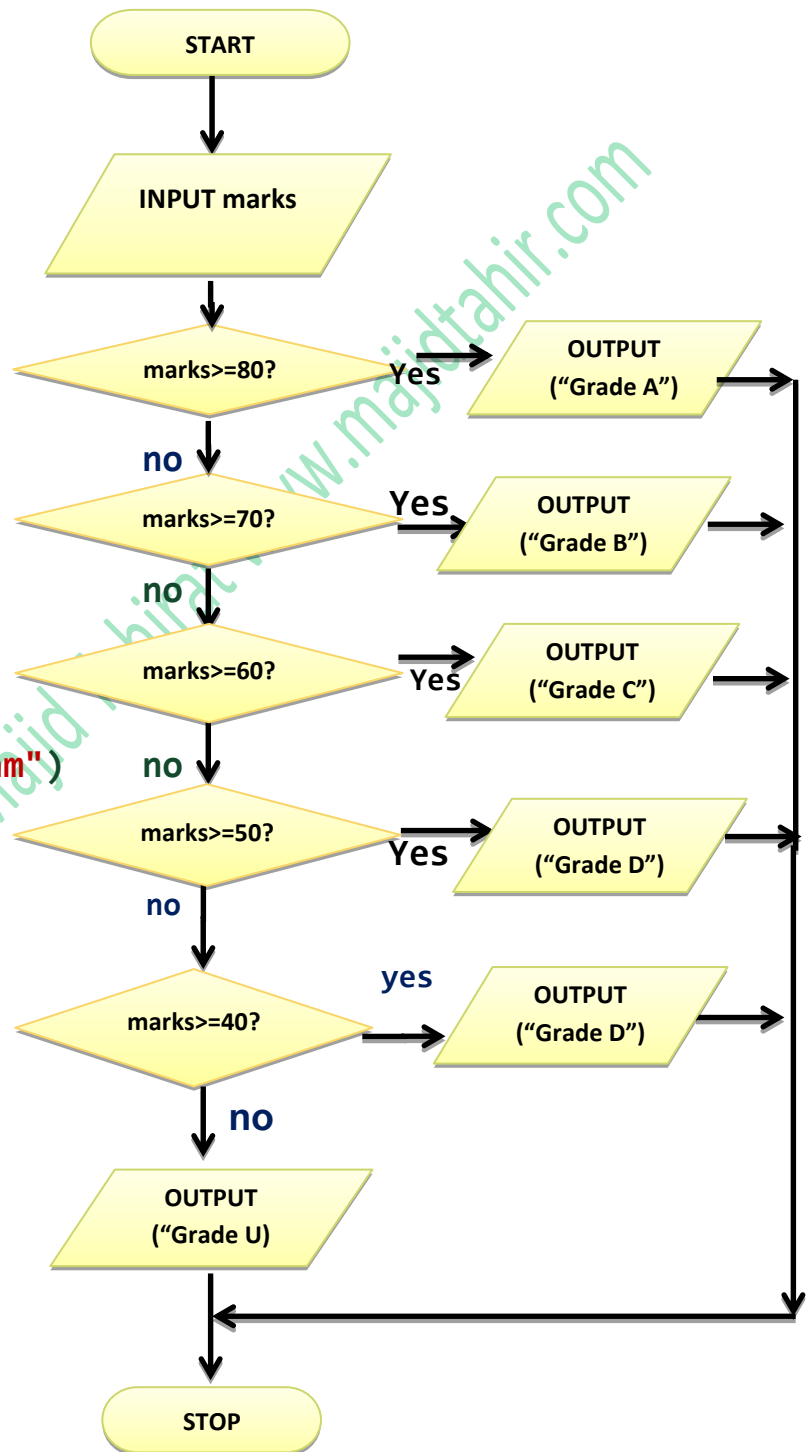
FLOWCHART

Pseudo code

```

BEGIN
DECLARE marks : Integer

PRINT ("Enter your marks")
INPUT marks
CASE OF marks
80 >= :PRINT("Grade A")
70 >= :PRINT("Grade B")
60 >= :PRINT("Grade C")
60 >= :PRINT("Grade D")
40 >= :PRINT ("Grade E")
OTHERWISE
PRINT("Grade U, Repeat Exam")
END CASE
END
    
```





Program Code in Visual Basic Console Mode:

```
Sub Main()
    Dim marks As Integer
    Console.WriteLine("Please Input your marks")
    marks = Console.ReadLine()

    While marks > 100 Or marks < 0
        Console.WriteLine("Wrong Entry, Please Enter Between 0 and 100")
        Console.WriteLine("Please Input your marks")
        marks = Console.ReadLine()
    End While

    Select Case marks 'NOTES by Sir Majid Tahir
        Case Is >= 90 'Download notes at www.majidtahir.com
            Console.WriteLine("Your Grade is A* ")
        Case Is >= 80
            Console.WriteLine("Your Grade is A ")
        Case Is >= 70
            Console.WriteLine("Your Grade is B ")
        Case Is >= 60
            Console.WriteLine("Your Grade is C ")
        Case Is >= 50
            Console.WriteLine("Your Grade is D ")
        Case Else
            Console.WriteLine("Your Grade is U, Please Repeat the Exam")
    End Select
    Console.Read()
End Sub
End Module
```

```
file:///C:/Users/Majid/AppData/Local/Temporary Projec...
Please Input your marks
-1
Wrong Entry, Please Enter Between 0 and 100
Please Input your marks
120
Wrong Entry, Please Enter Between 0 and 100
Please Input your marks
92
Your Grade is A*
```

LOOPS (Structured statements for iteration (repetition))

Many problems involve repeating one or more statements, so it is useful to have structured statements for controlling these iterations or repetitions. Exit conditions consist of logical expressions whose truth can be tested, such as Count = 10 or Score < 0. At a particular time, a logical expression is either **True** or **False**.

- FOR...TO...NEXT
- WHILE...DO...ENDWHILE
- REPEAT...UNTIL



FOR ... NEXT LOOP

This is to be used when loop is to be repeated a known fixed number of times. The counter is automatically increased each time the loop is performed.

```
FOR count = 1 to 10
  INPUT number
  total = total + number
NEXT count
```

WHILE ... Do LOOP

This loop is used when we don't know how many times the loop is to be performed. The Loop is ended when a certain condition is true.

This condition is checked before starting the loop.

```
While COUNT < 10 DO
  Input NUMBER
  TOTAL = TOTAL + NUMBER
  COUNT = COUNT + 1
Endwhile
Output TOTAL
```

REPEAT ... UNTIL LOOP

REPEAT UNTIL Loop is used when we do not know how many times loop will be performed.

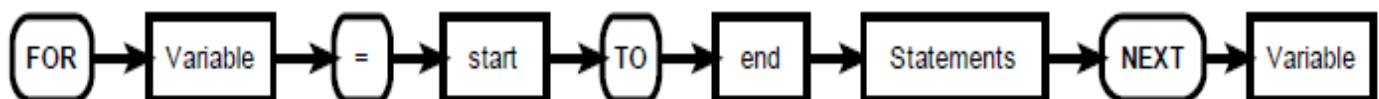
The Loop is ended when a certain condition is true.

The Condition is checked at the end of the Loop and so a REPEAT Loop always has to be performed at least once.

```
REPEAT
  Input NUMBER
  TOTAL = TOTAL + NUMBER
  COUNT = COUNT + 1
Until COUNT = 10
Output Total
```

FOR Loop PSEUDOCODE

The fore loop repeats statements a set number of time. It uses a variable to count how many time it goes round the loop and stops when it reaches its limit.





BEGIN

DECLARE count, number : Integer

OUTPUT ("Input a number for its times table")

INPUT number

FOR count = 1 To 20

PRINT (number , "times" , count , " = " number * Count")

NEXT

VB code example:

```
Sub Main(args As String())
```

```
    Console.WriteLine("Times Table Program")
```

```
    Dim count, num As Integer
```

```
    Console.WriteLine("please Input a number for its TimesTable")
```

```
    num = Console.ReadLine()
```

```
    For count = 1 To 20
```

```
        Console.WriteLine(num & " Times " & count & " = " & num * count)
```

```
    Next
```

```
End Sub
```

OUTPUT of Code

```
C:\Users\Lenovo\source\repos\Loops\Loops\bin\Debug\netcoreapp3.1\Loops.exe
Times Table Program
please Input a number for its TimesTable
7
7 Times 1 = 7
7 Times 2 = 14
7 Times 3 = 21
7 Times 4 = 28
7 Times 5 = 35
7 Times 6 = 42
7 Times 7 = 49
7 Times 8 = 56
7 Times 9 = 63
7 Times 10 = 70
7 Times 11 = 77
7 Times 12 = 84
7 Times 13 = 91
7 Times 14 = 98
7 Times 15 = 105
7 Times 16 = 112
7 Times 17 = 119
7 Times 18 = 126
7 Times 19 = 133
7 Times 20 = 140
```



Other examples of FOR loop

```

BEGIN
DECLARE BiggestSoFar, NextNumber, Counter : Integer

INPUT BiggestSoFar

    FOR Counter ← 1 TO 5

        INPUT NextNumber
        IF NextNumber > BiggestSoFar
            THEN
                BiggestSoFar ← NextNumber
            ENDIF

        END FOR

OUTPUT ("The biggest number so far is" & BiggestSoFar)
END
  
```

Sample VB Code of above Pseudocode:

```

Module Module1

    Sub Main()
        Dim biggestSoFar, NextNum, counter As Integer

        Console.WriteLine("Enter Biggest number")
        biggestSoFar = Console.ReadLine()

        For counter = 1 To 5

            Console.WriteLine("Enter Next biggest number")
            NextNum = Console.ReadLine()

            If NextNum > biggestSoFar Then
                biggestSoFar = NextNum
            End If

        Next

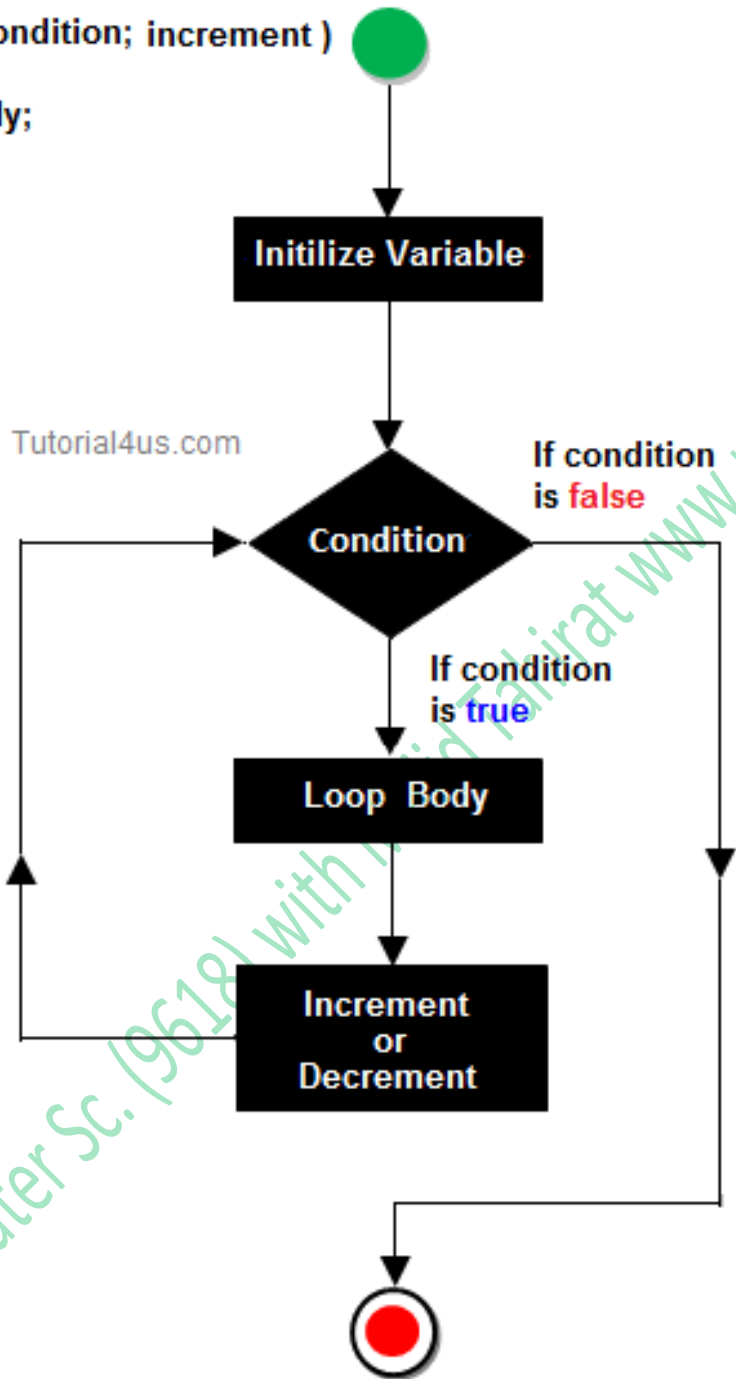
        Console.WriteLine("The biggest number entered is" & biggestSoFar)
        Console.ReadLine()

    End Sub

End Module
  
```


FLOWCHART FOR LOOP

```
for( init; condition; increment )
{
  loop body;
}
```

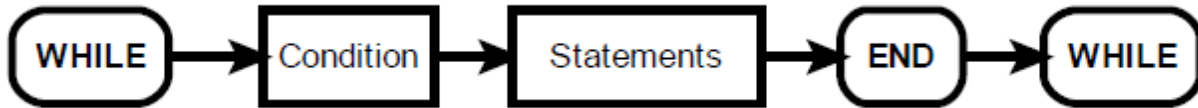


Computer Sc. (9618) with Majid Tahirat www.majidtahir.com

WHILE DO ENDWHILE loop

The while loop is known as a **test before loop**. The condition is tested before entering the loop, but tested each time it goes round the loop. The number of times the statements within the loop are executed varies. The test before loop goes round 0 or more times.

This method is useful when processing files and using "read ahead" data



```

BEGIN
DECLARE name : String

INPUT name

    WHILE name <> "x"
    PRINT ("Your name is: "name)
    INPUT name
    END WHILE
  
```

VB Code example

```

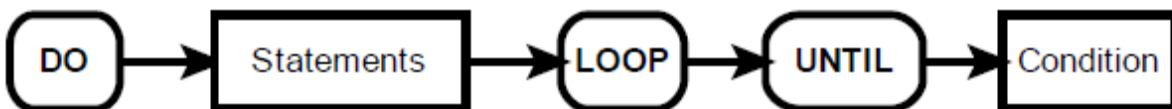
Sub Main()
    Dim name As String

    name = Console.ReadLine()
    'Test before loop -
    'only enter the loop is name not equal "X"
    While name <> "X"
        Console.WriteLine(name)
        name = Console.ReadLine()
    End While
End Sub
  
```

END

REPEAT UNTIL loop

The repeat loop is similar to the while loop, but it tests the condition after the statements have been executed once. This means that this test after loop goes round 1 or more times.



```

BEGIN
DECLARE name : String

    REPEAT
    INPUT name
    PRINT ("Your name is:" name)
    UNTIL name = "x"
  
```

VB code example

```

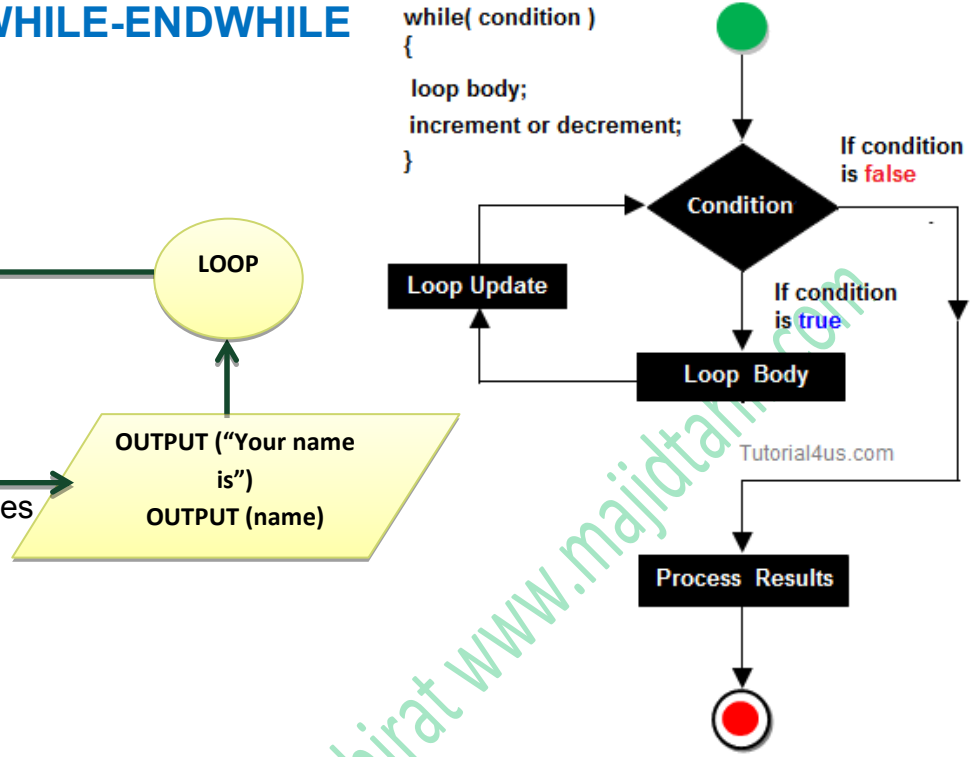
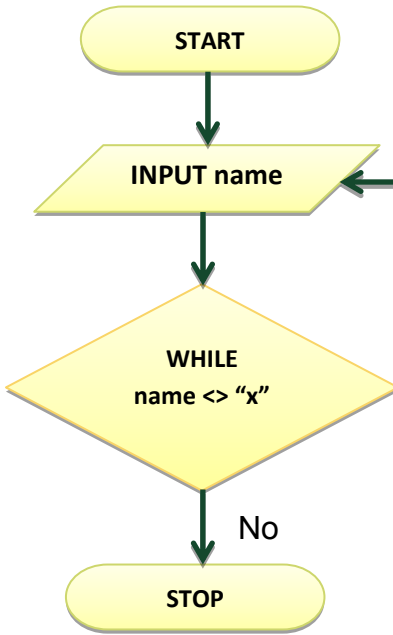
Sub Main()
    Dim name As String

    Do
        name = Console.ReadLine()
        Console.WriteLine(name)
    Loop Until name = "X"
    'Test after loop
End Sub
  
```

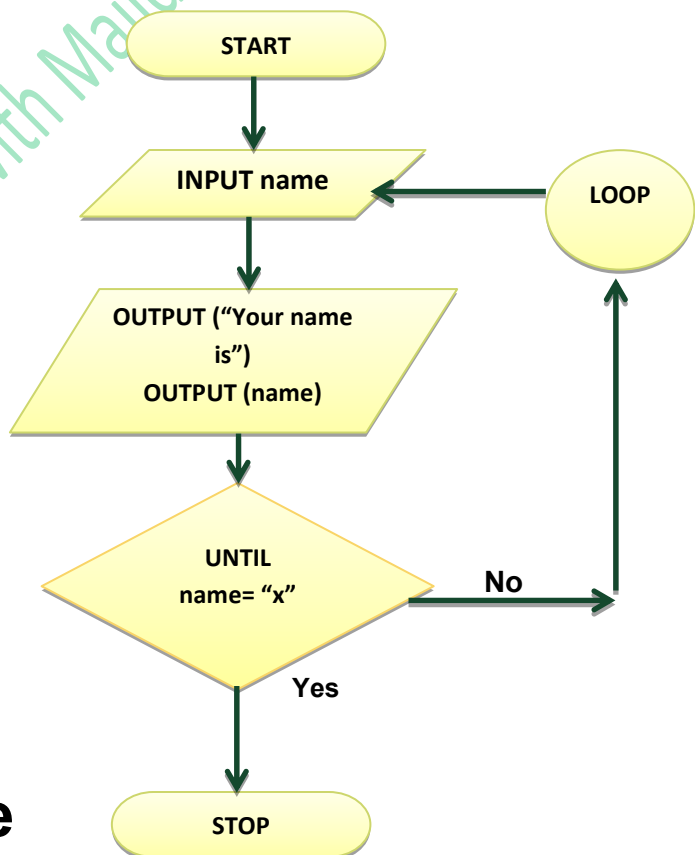
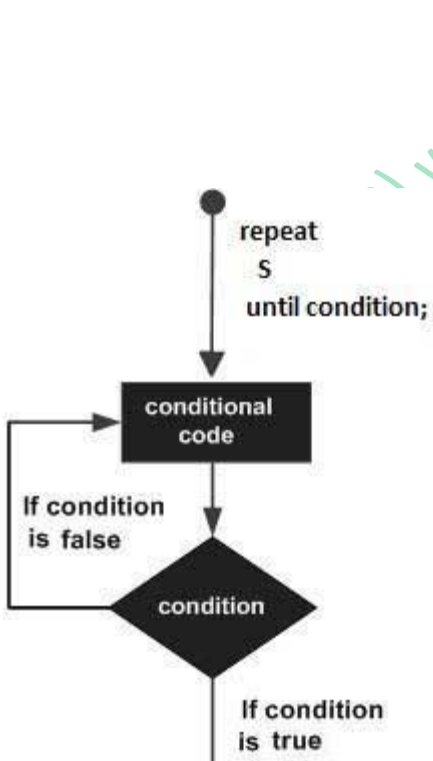
END

Keeps inputting name and keeps printing name until user enters "X"

FLOWCHART...WHILE-ENDWHILE



FLOWCHART...REPEAT-UNTIL



Array Data Type



An array is a special variable that has one name, but can store multiple values. Each value is stored in an element pointed to by an index.

The first element in the array has index value 0, the second has index 1, etc

One Dimensional Arrays

A one dimensional array can be thought as a list. An array with 10 elements, called names, can store 10 names and could be visualized as this:

index	Element
0	Fred
1	James
2	Tom
3	Robert
4	Jonah
5	Chris
6	Jon
7	Matthew
8	Mikey
9	Jack

Arrays (One-dimensional arrays)

In order to use a one-dimensional array in a computer program, you need to consider:

- What the array is going to be used for, so it can be given a meaningful name
- How many items are going to be stored, so the size of the array can be determined.
- What sort of data is to be stored, so that the array can be the appropriate data type.

This array would be created by:

```
DECLARE names(9): String
```

```
PRINT (names(1))
```

will display James

```
PRINT (names(7))
```

Will display Mathew

VB code example:

```
Dim names(9) As String
```

Elements indexed from 0 to 9

The statement:

```
Console.WriteLine(names(1))
```

Will display James

```
Console.WriteLine(names(7))
```

Will display Matthew



Entering Values in One-Dimension Array

```
BEGIN
DECLARE count : Integer
DECLARE name (5) : String // for declaring 5 elements in ARRAY
DECLARE marks (5) : Integer

FOR count = 1 to 5 // for inputting 5 names and grades
PRINT ("Enter Name "& count)
INPUT name (count)
PRINT ("Enter grade for "& name(count))
INPUT marks (count)
NEXT count

// for displaying 5 names and grades
FOR count 1 to 5
PRINT (name (count) & "has marks " & marks(count))
NEXT count
```

END

VB Code in Console Mode

```
ConsoleApplication1 - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Architecture Test Analyze Window Help
Module1.vb* x
Module1 (Declarations)
Module Module1
Sub Main()
Dim name(5) As String 'Declaration of Array (Notes by Sir Majid Tahir)
Dim marks(5) As Double 'Declaration of Array (www.majidtahir.com)
For count = 1 To 5 'Loop used to Enter values in an array
Console.WriteLine("please Enter your name " & count)
name(count) = Console.ReadLine()
Console.WriteLine("please enter your marks " & count)
marks(count) = Console.ReadLine()
Next
For count = 1 To 5 'Loop used to display values of Arrays
Console.WriteLine("Our Student " & name(count) & " has scored " & marks(count))
Next
Console.ReadKey()
End Sub
End Module
```



Output of VB code displayed above

The screenshot shows a Visual Basic IDE with a code editor on the left and a console window on the right. The code in the editor is as follows:

```

Module1
  Sub Main()
    Dim name(5) As String 'Declaration of Array (Notes by Sir Majid Tahir)
    Dim marks(5) As Double 'Declaration of Array (www.majidtaahir.com)

    For count = 1 To 5 'Loop used to Enter values in an array
      Console.WriteLine("please Enter your name " & count)
      name(count) = Console.ReadLine()

      Console.WriteLine("please enter your marks " & count)
      marks(count) = Console.ReadLine()
    Next

    For count = 1 To 5 'Loop used to display values of Arrays
      Console.WriteLine("Our Student " & name(count) & " has scored " & marks(count))
    Next

    Console.ReadKey()
  End Sub
End Module

```

The console window shows the following output:

```

please Enter your name 1
Majid
please enter your marks 1
99
please Enter your name 2
Sajid
please enter your marks 2
88
please Enter your name 3
Tahir
please enter your marks 3
90
please Enter your name 4
Waris
please enter your marks 4
78
please Enter your name 5
Mustafa
please enter your marks 5
11
Our Student Majid has scored 99
Our Student Sajid has scored 88
Our Student Tahir has scored 90
Our Student Waris has scored 78
Our Student Mustafa has scored 11

```

Another example of One-Dimensional Array

```

Module Module1
  Sub Main()
    Dim count As Integer
    Dim name(4) As String
    Dim marks(4) As Integer
    Dim gender(4) As String
    For count = 0 To 4
      Console.WriteLine("please enter your name" & count)
      name(count) = Console.ReadLine()
      Console.WriteLine("please enter your gender" & count)
      gender(count) = Console.ReadLine()
      Console.WriteLine("please enter your marks" & count)
      marks(count) = Console.ReadLine()
    Next count
    For count = 0 To 4
      Console.WriteLine("your name is : " & name(count))
      Console.WriteLine("your gender is : " & gender(count))
      Console.WriteLine("your marks are : " & marks(count))
    Next count
    Console.ReadKey()
  End Sub
End Module

```




Two Dimensional Arrays (2-D Arrays)

Using pseudocode, the algorithm to set each element of array `ThisTable` to zero is:

```
FOR Row ← 1 TO MaxRows
  FOR Column ← 1 TO MaxColumns
    ThisTable[Row, Column] ← 0
  ENDFOR
ENDFOR
```

When we want to output the contents of a 2D array, we again need nested loops. We want to output all the values in one row of the array on the same line. At the end of the row, we want to output a new line.

```
FOR Row ← 1 TO MaxRows
  FOR Column ← 1 TO MaxColumns
    OUTPUT ThisTable[Row, Column] // stay on same line
  ENDFOR
  OUTPUT Newline // move to next line for next row
ENDFOR
```

PSEUDOCODE Example of Two-Dimension Array

BEGIN

```
DECLARE table(3, 4) : Integer
FOR row = 1 To 3
  FOR column = 1 To 4
    PRINT("Please Input Value in Row: ", row, "column : ", column)
    INPUT table(row, column)
  NEXT
NEXT
```

```
FOR row = 1 To 3
  FOR column = 1 To 4
    PRINT ("Row = " & row & "column = " & column & "has Value")
    PRINT (table(row, column))
  NEXT
NEXT
```

END

VB Code Example of Two-Dimension Array



```

Sub Main()
    Dim table(2, 3) As Integer
    For row = 0 To 2
        For column = 0 To 3
            Console.WriteLine("Please Input Value in Row: " & row & "column : " & column)
            table(row, column) = Console.ReadLine()
        Next
    Next
    Console.Clear()

    For row = 0 To 2
        For column = 0 To 3
            Console.WriteLine("Row = " & row & "column = " & column & "has Value")
            Console.WriteLine(matrix(row, column))
        Next
    Next
    Console.ReadKey()
End Sub

```

Multi-Dimensional Arrays:

A multi-dimensional array can be thought of as a table, each element has a row and column index. Following example declares a two-dimensional array called `matrix` and would be declared by

```
Dim matrix(2,3) As Integer
```

Usually we refer to the first dimension as being the rows, and the second dimension as being the columns.

index	0	1	2	3
0	A	B	C	D
1	E	F	G	H
2	I	J	K	L

The following statements would generate the following

```
Console.WriteLine(matrix(0, 0))
```

Would display A

```
Console.WriteLine(matrix(2, 1))
```

Would display J

```
Console.WriteLine("first row, first column : " & matrix(2, 3))
```

Would display first row, first column : L

VB Code for 2-D Array is:



```
Module1 (Declarations)
Module Module1
Sub Main() ' Notes by Sir Majid Tahir ( Download free at www.majidtahir.com)
Dim table(3, 4) As Integer ' DECLARING TWO-DIMENSIONAL ARRAY
For row = 1 To 3 ' Variable Row is used to use in loop for rows
    For column = 1 To 4 ' Variable column is used to use in Columns
        Console.WriteLine("please Enter data in row= " & row & " column = " & column)
        table(row, column) = Console.ReadLine()
    Next
Next
For row = 1 To 3
    For column = 1 To 4
        Console.WriteLine("Data is Row= " & row & " column = " & column & " = " & table(row, column))
    Next
Next
Console.ReadKey()
End Sub
End Module
```

References:

- Computer Science by David Watson & Helen Williams
- Visual Basic Console Cook Book
- Computer Science AS and A level by Sylvia Langfield and Dave Duddell
- <https://www.sitesbay.com/javascript/javascript-looping-statement>
- <http://wiki.jikexueyuan.com/project/lua/if-else-if-statement.html>