CSE 123
Introduction to Computing

Lecture 9

Loops and Arrays

SPRING 2012

Assist. Prof. A. Evren Tugtas

Marmara Üniversitesi

Using Loops to repeat an action

- You may want to repeat an action to achieve certain effects.
- You may want to repeat an action for certain amount of time
- You may want to repeat an action until a certain condition is met
- In VBA you can use loops to repeat an action

Loops

- You can repeat an action by writting repetitive codes
- Loops have several advantages over simple repetition of codes
  - Procedures will be shorter
  - Procedures will be more flexible
  - Procedures will be easier to read and debug

Loops

- Loop is a structure that repeats a number of statements.
- Each cycle of execution of a statement is called an <u>iteration.</u>
- There are two categories of loops:
  - <u>Fixed-iteration</u> loops repeat a set number of times
  - <u>Indefinite loops</u> repeat a flexible number of times
- Running of any loop is controlled by a <u>loop variant</u> or <u>loop determinant</u>

Marmara Üniversitesi

Loops

- Loop variants can be
  - Numeric expression
  - Logical expression
- Fixed iteration loops generally use numeric expressions (e.g. 10 iterations of loop)
- Indefinite loops use logical expressions

Loop types in VBA

1) <u>For.....Next Loop</u>
   - Fixed iteration loop
   - Repeats an action for a given number of times
2) <u>For Each.....</u>

   <u>Next Loop</u>
   - Fixed iteration loop
   - Repeats an action once for each object in a VBA collection

Loop types in VBA

3) <u>Do While.... Loop</u>

- Indefinite loop,
- Performs an action if a condition is TRUE and continues to perform it until the condition becomes FALSE

4) <u>While....Wend Loop</u>

- Indefinite
- Behaves like a Do While statement. Performs an action if a condition is TRUE and continues to perform it until the condition becomes FALSE

Marmara Üniversitesi

Loop types in VBA

5) <u>Do Until.... Loop</u>

- Indefinite loop,
- Performs an action <u>while</u> a condition is FALSE and continues to perform it until the condition becomes TRUE

6) <u>Do......Loop</u>

<u>While</u>

- Indefinite
- Performs an action <u>once and then repeats it while</u> a condition is TRUE until the condition becomes TRUE

Loop types in VBA

7) <u>Do....... Loop</u>

<u>Until</u>

- Indefinite loop,
- Performs an action <u>once and repeats it while</u> a condition is FALSE until the condition becomes TRUE

For......Next Loops

- For...Next loops repeats an action or sequence for a given number of times, specified by a counter variable
- The counter variable
  - can be hard coded into the procedure
  - can be passed from an input or dialogue box
  - value can be generated by a different part of the procedure

For......Next Loops

<u>SYNTAX</u>

```
For counter=start To end [stepsize]
        [statements]
[Exit For]
        [statements]
Next [counter]
```

Marmara Üniversitesi

For……Next Loops
Counter

- Counter: a numeric variable or an expression that produces a number
- VBA increases the counter value by an increment of 1 each iteration of the loop
- Increment can be controlled by an optional Step keyword and stepsize argument.
- Counter is required for the For statement but its optional for the Next statement (however makes it easier to read)

```
For counter=start To end [stepsize]
        [statements]
[Exit For]
        [statements]
Next [counter]
```

Marmara Üniversitesi

For......Next Loops

start: A numeric variable or numeric expression giving the starting value for counter

end: A numeric variable or numeric expression giving the ending value for counter

stepsize: A numeric variable or numeric expression specifiying how much to increase or decrease the value of counter.

exit for: A statement for ending the For loop

next: The keyword indicating the end of the loop

```
For counter=start To end [stepsize]
        [statements]
[Exit For]
        [statements]
Next [counter]
```

Marmara Üniversitesi

For.....Next Loops

- You first specify a counter variable and starting and ending variables of it

```
Dim i as integer
For i=1 to 1000
```

```
For counter=start To end [stepsize]
            [statements]
[Exit For]
            [statements]
Next [counter]
```

i→ counter variable

1→ starting value

1000→ ending value

By default counter variable will increase by 1

For.....Next Loops

- Counter variables used in For....Next loops are
- i, j, k, l, m, n
- You can use other variables. However, long names may cause confusion

Marmara
Üniversitesi

For.....Next Loops

```
Sub example1()
Dim i As Integer
For i = 1 To 1000
    Statements
Next i
End Sub
```

For.....Next Loops

```
Sub example1()
Dim i, a As Integer
For i = 0 To 10
    a = i + 1
    Cells(1, 2) = "My integer numbers"
    Cells(i + 2, 2) = a
    Next i
End Sub
```

Marmara Üniversitesi

For.....Next Loops with Step Values

```
Sub example2()
Dim i, a As Integer
For i = 0 To 10 Step 2
   a = i + 1
   Cells(1, 3) = "My integer numbers"
   Cells(i + 2, 3) = a
Next i
End Sub
```

```
For counter=start To end [stepsize]
        [statements]
[Exit For]
        [statements]
Next [counter]
```

Marmara Üniversitesi

For.....Next Loops with Step Values

```
Sub example2()
Dim i, a As Integer
For i = 10 To 0 Step -2
    a = i + 1
    Cells(1, 3) = "My integer numbers"
    Cells(i + 2, 3) = a
Next i
End Sub
```

Getting the Counter Variable Elsewhere

```
Sub example3()
Dim i, a, stepno, counterno As Integer
Sheet1.Cells.Clear
counterno = InputBox("Please enter the counter number")
stepno = InputBox("Please enter the stepnumber")
For i = 0 To counterno Step stepno
a = i + 1
Cells(1, 4) = "My integer numbers"
Cells(i + 2, 4) = a
Next i
End Sub
```

For Each....Next Loops

- For Each...Next Loops are unique to Visual Basic
- Works with known number of repetitions
- Counter is the number of objects in a collection
- e.g. Documents collection in Word
- For Each means→ each object in a collection
- The programmer do not have to know the number of iterations in advance

For Each....Next Loops

SYNTAX

```
For Each object In collection
    [statements]
    [Exit for]
    [statements]
Next [object]
```

Marmara
Üniversitesi

For Each....Next Loops

SYNTAX

```
For Each object In collection
    [statements]
    [Exit for]
    [statements]
Next [object]
```

Marmara Üniversitesi

For Each....Next Loops

```
Sub example4()
counter = 0
For Each cellobject In Worksheets("Sheet1").Range("A2:D20").Cells
If cellobject.Value > 100 Then counter = counter + 1
Cells(1, 1) = counter
Next
End Sub
```

Do....Loops

- Do loops give more flexibility compared to For loops
- Do While....Loop
- Do......Loop While
- Do Until.....Loop
- Do.....Loop Until

Do....Loops

- Loops can be classified in two categories
- Loops that test a condition before performing an action.
  - Do While.....Loop
  - DoUntil....Loop
- Loop perform an action before testing a condition
  - Do......Loop While
  - Do......Loop Until

Do....Loops

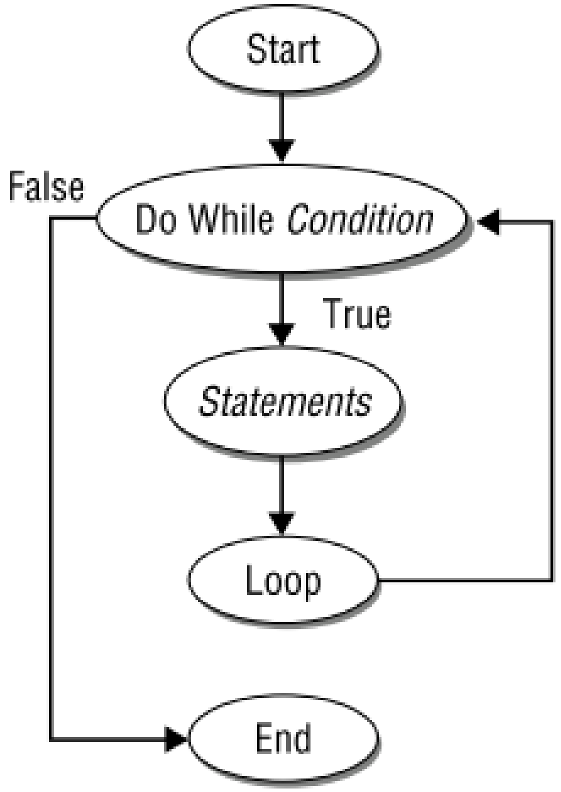
- While Loop repeats itself <u>while the condition is</u> TRUE
- Until Loop repeats itself <u>until the condition becomes</u> TRUE

Do While....Loops

SYNTAX

```
Do While condition
    [statements]
    [Exit Do]
    [statements]
Loop
```

Returns the execution to Do While line



Ref: Mansfield R, *Mastering VBA for Microsoft Office 2007*. Wiley Publishing, 2008

Do While....Loops

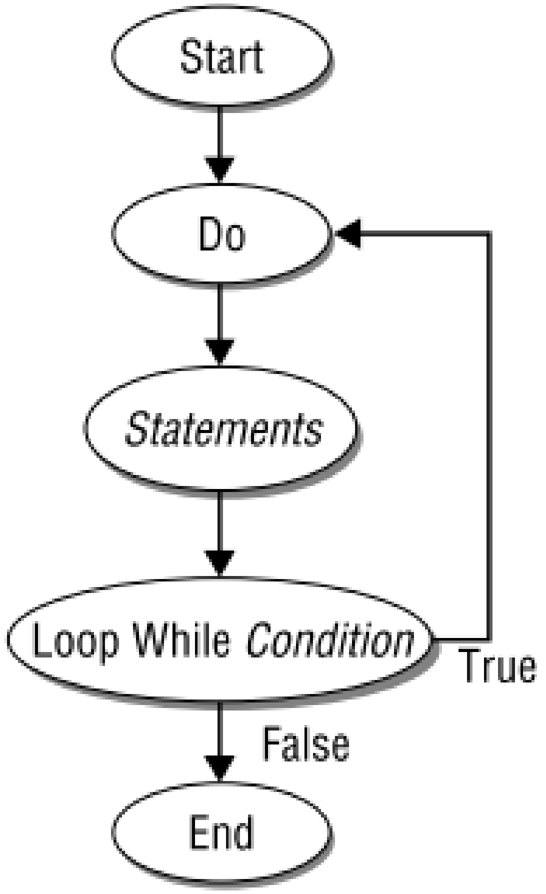
```
Sub example5()
Dim x As Integer
Sheet1.Cells.Clear
x = 2
i = 0
Do While x < 20
    i = i + 1
    x = x * 2
    Cells(i, 2) = x
Loop
End Sub
```

Marmara Üniversitesi

Do....Loop While Loops

- Actions in the loops is executed at least once whether the condition is TRUE or FALSE

```
Do
    [statements]
    [Exit Do]
    [statements]
Loop While Condition
```



Ref: Mansfield R, *Mastering VBA for Microsoft Office 2007*. Wiley Publishing, 2008

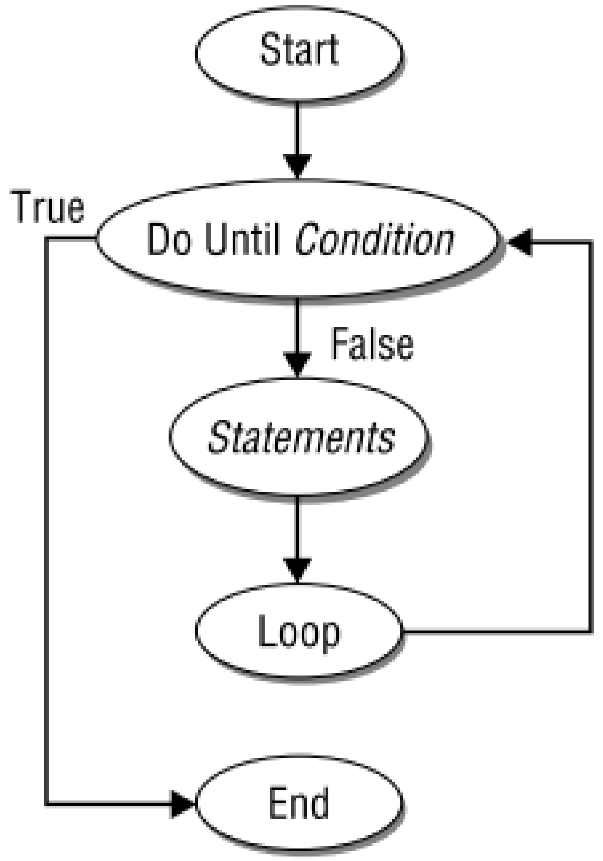
Do....Loop While Loops

```
Sub example6()
Dim x As Integer
Sheet1.Cells.Clear
x = 50
Do
x = x * 2
Loop While x < 20
End Sub
```

Do Until.....Loops

SYNTAX

```
Do Until condition
    [statements]
    [Exit Do]
    [statements]
Loop
```



Ref: Mansfield R, *Mastering VBA for Microsoft Office 2007.* Wiley Publishing, 2008

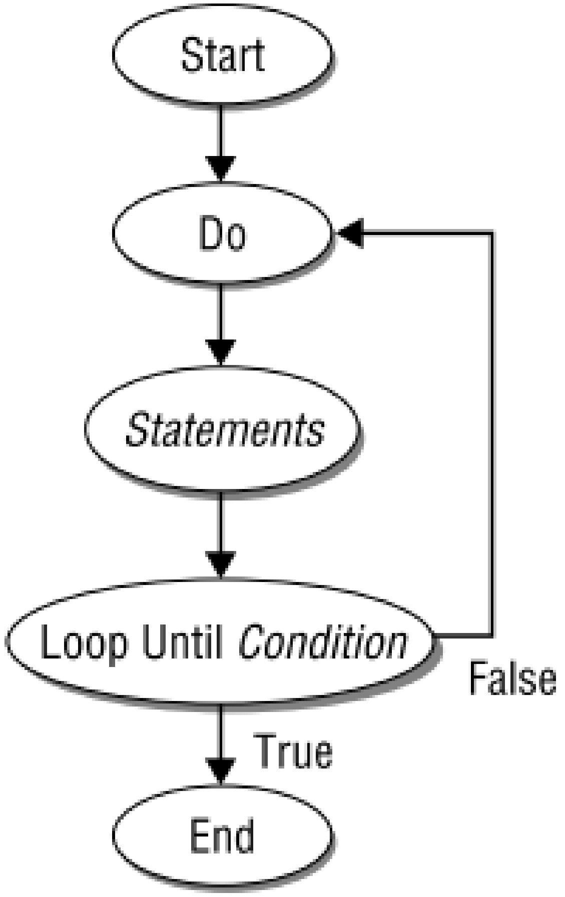
Do Until.....Loops

```
Sub example7()
Dim x As Integer
Sheet1.Cells.Clear
x = 5
Do Until x < 20
   x = x * 2
Loop
End Sub
```

Marmara Üniversitesi

Do......Loop Until Loops

```
SYNTAX
Do
    [statements]
    [Exit Do]
    [statements]
Loop Until condition
```



Ref: Mansfield R, *Mastering VBA for Microsoft Office 2007*. Wiley Publishing, 2008

Do......Loop Until Loops

```
Sub example8()
Dim x As Integer
Sheet1.Cells.Clear
x = 5
Do
   x = x * 2
Loop Until x < 50
End Sub
```

Marmara
Üniversitesi

Using Exit Do Statement

- The Exit Do statement is optional
- Exit Do statements are generally used with a condition

```
Sub example9()
    Dim i As Integer
    Dim Num As Single
    Sheet1.Cells.Clear
    'infinite loop.
    Do
        For i = 1 To 1000
                Cells(i, 2) = Num
                Num = Int(Rnd() * 50)
                Cells(i, 2) = Num
            Select Case Num
                 Case 9: Exit For
                 Case 15: Exit Do
                 Case 78: Exit Sub
            End Select
        Next i
    Loop
End Sub
```

Marmara Üniversitesi

Avoiding Infinite Loops

```
Sub InfiniteLoop()
    Dim x
    x = 1
    Do
        Application.StatusBar = _
           "Your computer is stuck in an endless loop: " & x
        x = x + 1
    Loop
End Sub
```

Ref: Mansfield R, *Mastering VBA for Microsoft Office 2007*. Wiley Publishing, 2008