CSE 123
Introduction to Computing

Lecture 8

Decision Structures (Go-To Statement, If-Then Structure, Select Case Structure)

SPRING 2012

Assist. Prof. A. Evren Tugtas

Marmara Üniversitesi

Making Decisions

- Programming languages include commands that tests conditions and make decisions
- Similar to human decisions

- **If** its cold, **then** wear a sweater

Making Decisions

- VBA provides several types of IF statements suitable for making simple or complex decisions
- VBA also has SELECT-CASE statement to work with truly involved decisions

Comparison Operations

- You may need to compare things in order to make a

TABLE 11.1: VBA's Comparison Operators

OPERATOR	MEANING	EXAMPLE
=	Equal to	`If strMyString="Hello" Then`
<>	Not equal to	`If x <> 5 Then`
<	Less than	`If y < 100 Then`
>	Greater than	`If strMyString > "handle" Then`
<=	Less than or equal to	`If intMyCash <= 10 Then`
>=	Greater than or equal to	`If Time >= 12:00 PM Then MsgBox "It's afternoon." Else MsgBox "It's morning." End If`
Is	Is the same object variable as	`If Object1 Is Object2 Then`

Ref: Mansfield R. Mastering VBA for Microsoft Office 2007. Wilwy Publishing. 2008

Comparison Operations

```
Dim objTest1 As Object
Dim objTest2 As Object
Set objTest1 = ActiveDocument.Paragraphs(1).Range
Set objTest2 = ActiveDocument.Paragraphs(1).Range
'the next statement returns False because the objects are different
MsgBox objTest1 Is objTest2
```

Ref: Mansfield R. Mastering VBA for Microsoft Office 2007. Wilwy Publishing. 2008

Testing Multiple Conditions

Operator	Meaning	Example	Comments
And	Conjunction	`If ActiveWorkbook.FullName = "c:\temp\Example.xlsm" And Year(Date) >= 2005 Then`	If both conditions are True, the result is True. If either condition is False, the result is False.
Not	Negation	`ActivePresentation.Saved = Not ActivePresentation.Saved`	Not reverses the value of x (True becomes False; False becomes True). The Saved property used in this example is Boolean.
Or	Disjunction	`If ActiveWindow.View = wdPageView Or ActiveWindow.View = wdOutlineView Then`	If either the first condition or the second is True, or if both conditions are True, the result is True.
XOr	Exclusion	`If Salary > 55000 XOr Experienced = True Then`	Tests for different results from the conditions: Returns True if one condition is False and the other is True; returns False if both conditions are True or both conditions are False.

Ref: Mansfield R. Mastering VBA for Microsoft Office 2007. Wilwy Publishing. 2008

Testing Multiple Conditions

Operator	Meaning	Example	Comments
Eqv	Equivalence	`If blnMyVar1 Eqv blnMyVar2 Then`	Tests for logical equivalence between the two conditions: If both values are `True`, or if both values are `False`, `Eqv` returns `True`. If one condition is logically different from the other (that is, if one condition is `True` and the other is `False`), `Eqv` returns `False`.
Imp	Implication	`If blnMyVar1 Imp blnMyVar2 Then`	Tests for logical implication. Returns `True` if both conditions are `True`, both conditions are `False`, or the second condition is `True`. Returns `Null` if both conditions are `Null` or if the second condition is `Null`. Otherwise, returns `False`.

Ref: Mansfield R. Mastering VBA for Microsoft Office 2007. Wilwy Publishing. 2008

IF Statements

- If statements are useful and versitile commands in VBA.
  - If .....Then
  - If.....Then.....Else
  - If.....Then.....ElseIf.....Else

IF Statements
If.....Then

- It is the simplest If statement

If the condition is met →

Execute the following statement(s)

If the condition is not met→

Skip the line immediately

IF Statements
If.....Then

- If ***block statement*** begins with **If** and ends with **End If**
- If....Then statement can be laid on multiple lines
- Multiple line *block* If statement:

```
If condition Then
    statement
    [statements]
End If
```

IF Statements
If.....Then

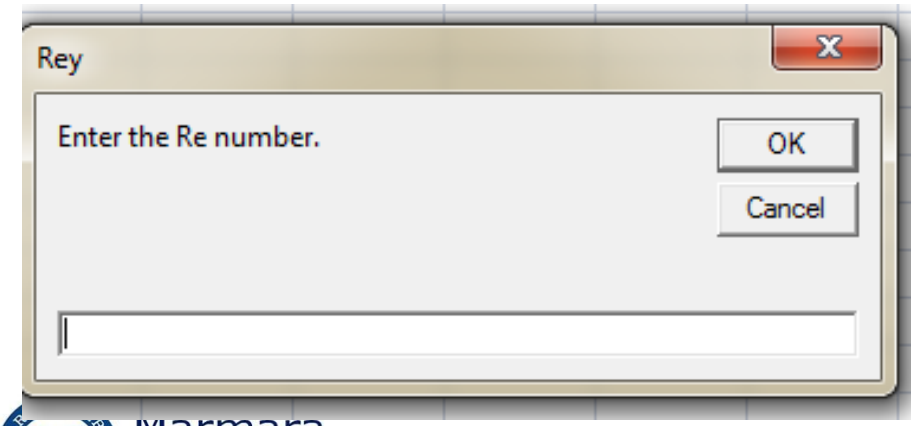
- If....Then statement can also be laid on a single line
- **End If** statement is **omitted** if Short If is used
- Single line If statement (Short If):

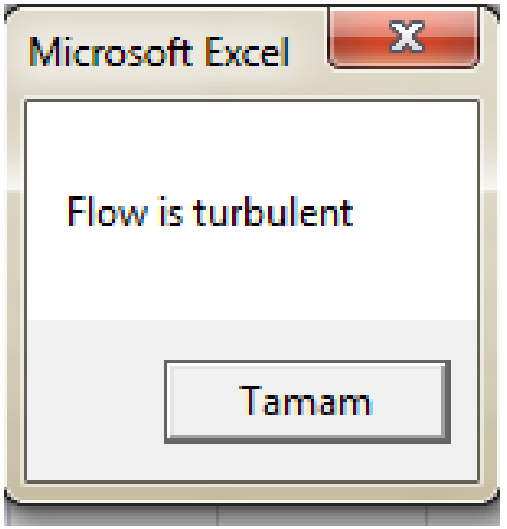
If *condition* Then statement[s]

- If condition is met, VBA executes the statements that follow, if the condition is not met, VBA does not execute the statements

Example

```
Option Explicit
  Sub Flow()
    Dim Re As Single
    Re = InputBox("Enter the Re number.", "Rey")
    If Re > 10000 Then MsgBox ("Flow is turbulent")
End Sub
```





Example

```
Option Explicit
Sub Flow()
Dim Re As Single
Re = InputBox("Enter the Re number.", "Rey")
If Re > 10000 Then MsgBox ("Flow is turbulent"): End
End Sub
```

If you want the end the procedure you could include the end statement after a colon on the same line

..... **:END**

IF Statements
If.....Then

If Re > 10000 Then MsgBox ("Flow is turbulent"): End

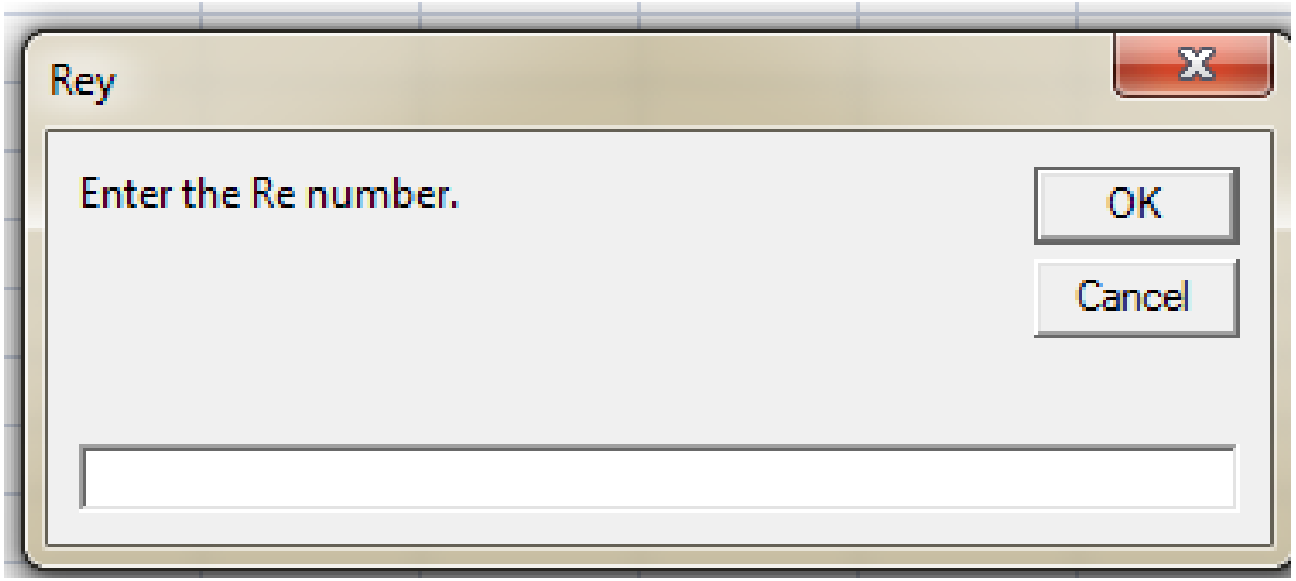
VBA Executes this code as:

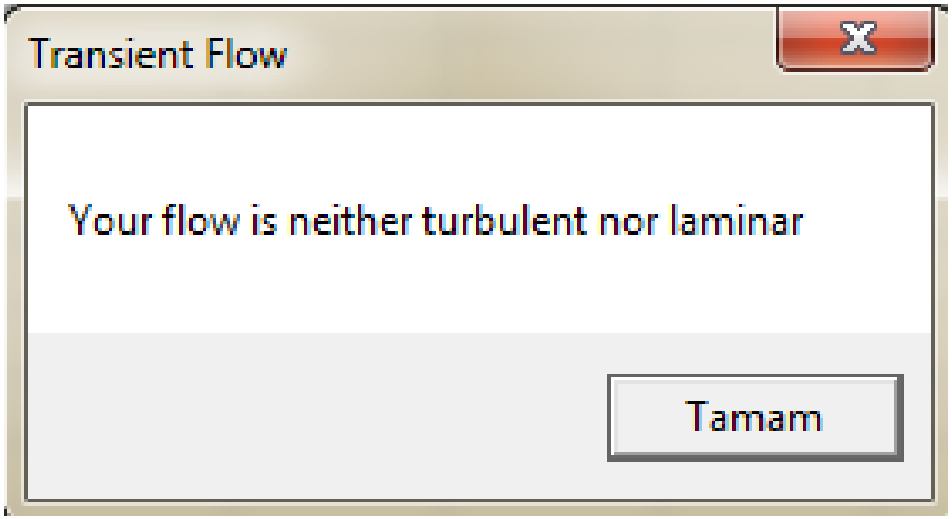
- It evaluates the condition
- If the condition is met it evaluates the first statement
- VBA executes the statement after the colon

IF Statements
If.....Then

```
Option Explicit
  Sub Flow()
  Dim Re As Single
  Re = InputBox("Enter the Re number.", "Rey")
  If Re < 10000 Then If Re > 20 Then MsgBox _
  "Your flow is neither turbulent nor laminar", , "Transient Flow": End
End Sub
```

OUTPUT


Rey
Enter the Re number.
OK
Cancel


Transient Flow
Your flow is neither turbulent nor laminar
Tamam

IF Statements
If.....Then

```
Option Explicit
  Sub Flow()
  Dim Re As Single
  Re = InputBox("Enter the Re number.", "Rey")
  If Re < 10000 Then If Re > 20 Then MsgBox _
  "Your flow is neither turbulent nor laminar", , "Transient Flow": End
End Sub
```

- Single line If statements are hard to follow and you need to break the code.

BLOCK IF Statements
If.....Then

```
Option Explicit
  Sub Flow()
  Dim Re As Single
  Re = InputBox("Enter the Re number.", "Rey")
  If Re > 10000 Then
      MsgBox "Your flow is turbulent", , "Transient Flow"
  End If
End Sub
```

- Block If statements are easy to follow

IF Statements
If....Then....Else

- If....Then....Else statements are used if you need to decide between two courses of action
- One course of action → If the condition TRUE
- Another course of action→ if the condition is FALSE

- You can use If...Then...Else statements for two button message boxes

IF Statements
If....Then....Else

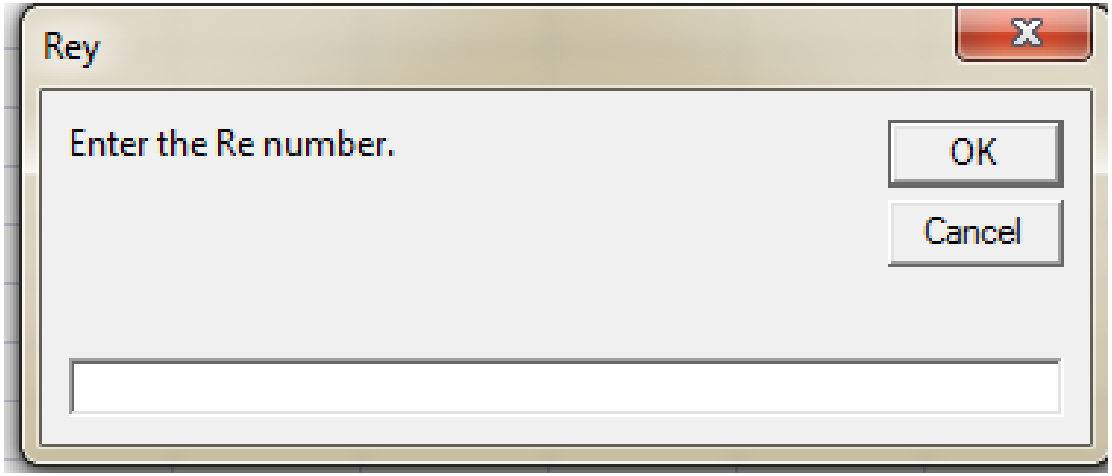
- One line or multiple line If....Then....Else statements can be used.
- However, it is easier to read multiple line statements

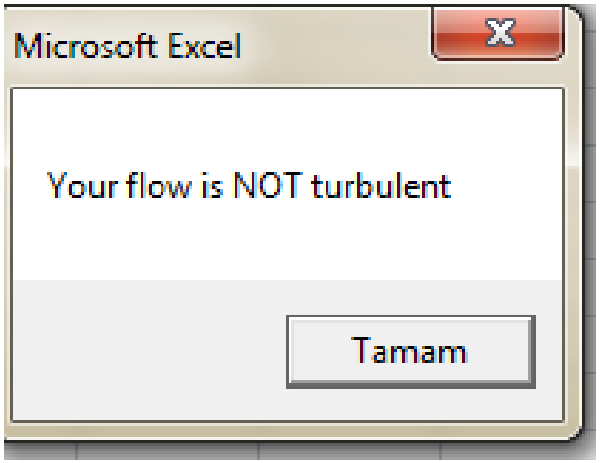
```
If condition Then
    statements1
Else
    statements2
End If
```

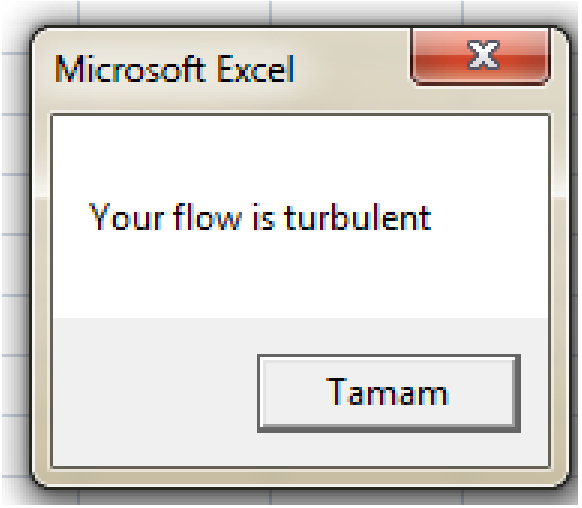
IF Statements
If....Then....Else

```
Option Explicit
Sub Flow()
Dim Re As Single
Re = InputBox("Enter the Re number.", "Rey")
If Re > 10000 Then
    MsgBox "Your flow is turbulent"
Else
    MsgBox "Your flow is NOT turbulent"
End If
End Sub
```

OUTPUT


Rey
Enter the Re number.
OK
Cancel


Microsoft Excel
Your flow is NOT turbulent
Tamam


Microsoft Excel
Your flow is turbulent
Tamam

IF Statements
If....Then....ElseIf....Else

- You can ask VBA to decide between multiple courses of actions
- You can use any number of ElseIf statements, depending on the complexity of your problem
- One-line or multiple line If....Then....ElseIf....Else statements can be used.
- However, in almost all cases, block If....Then....ElseIf....Else statements are easier to construct, to read, and to debug
- One line statements do not need an End If statement

IF Statements
If....Then....ElseIf....Else

- Syntax

```
If condition1 Then
  statements1
ElseIf condition2 Then
  statements2
ElseIf condition3 Then
  statements3
Else
  statement4
End If
```

IF Statements
If....Then....ElseIf....Else

- The Else clause is optional
- You can have as much as ElseIf clauses you need
- However, if you have more than five ElseIf clauses it would be easier to use Select Case statement

IF Statements
If....Then....ElseIf....Else

```
Option Explicit
Sub Flow()
Dim Re As Single
Re = InputBox("Enter the Re number.", "Rey")
If Re > 10000 Then
   MsgBox "Your flow is turbulent"
ElseIf Re < 20 Then
   MsgBox "Your flow is Laminar"
Else
   MsgBox "Your flow is transient"
End If
End Sub
```

IF Statements
If....Then....ElseIf

```
Option Explicit
Sub Flow()
Dim Re As Single
Re = InputBox("Enter the Re number.", "Rey")
If Re > 10000 Then
   MsgBox "Your flow is turbulent"
ElseIf Re < 20 Then
   MsgBox "Your flow is Laminar"
Else
   MsgBox "Your flow is transient"
End If
End Sub
```

Marmara Üniversitesi

Creating Loops with If and GoTo

- If .....GoTo loops can create a spaghetti code
- There are neater ways to create loops
- However, they are easy to use and work perfectly well

Syntax:

GoTo line

Creating Loops with If and GoTo

```
Sub A( )
1:
If MsgBox("Go  to  line  1?", vbYesNo) = vbYes Then
GoTo 1
End If
End Sub
```

Nesting If Statements

```
If condition1 Then                      'start of first If
    If condition2 Then                   'start of second If
        If condition3 Then               'start of third If
            statements1
        ElseIf condition4 Then           'ElseIf for third If
            statements2
        Else                             'Else for third If
            statements3
        End If                           'End If for third If
    Else                                 'Else for second If
        If condition5 Then               'start of fourth If
            statements4
        End If                           'End If for fourth If
    End If                               'End If for second If
Else                                     'Else for first If
    statements5
End If                                   'End If for first If
```

```
Sub Pipe()
Dim Q, V, Pi As Single
Q = 10
Pi = 3.1415
Msg = "Do you want to design the pipe between junction and collection box?"
jun = MsgBox(Msg, vbYesNoCancel)
If jun = vbYes Then
Again:
V = InputBox("Please enter velocity smaller than 0.5 m/sec")
dia = ((8 * Q) / (Pi * V)) ^ (1 / 2)
Cells(2, 7) = "Calculated diameter"
Cells(3, 7) = dia
If V > 0.5 Then
Msg = "Your velocity should be smaller than 0.5 m/s"
GoTo Again
End If
End If
End Sub
```

Select Case Statements

Syntax

```
Select  Case  TestExpression
   Case  Expression1
        Statements1
   [Case  Expression2
        Statements2]
   [Case  Else
        StatementsElse]
End  Select
```

Select Case Statements

- Effective alternative to use of multiple ElseIf Statements
- Select Case statement is used if the decision in the code depends on one variable or expression that has more than 4 different values that needs to be evaluated
- Select Case statement is easier to read

Select Case Statements

- Select Case → Starts the statement
- End Case → Ends the statement

```
Option Explicit
Sub CheckStudyHours()
Dim varStudyhours   As Variant
 Dim strMsg   As String
varStudyhours = InputBox("How  many hours do you study for CSE123 _
    each week?", "Study Hour")
     Select Case varStudyhours
       Case ""
 End
         Case Is < 0, 0, 0.1 To 1
 strMsg = "Please  study more"
         Case 1 To 3
```

```
strMsg = "You can work little bit more  "
        Case 3 To 5
strMsg = "I am  satisfied  with  the time you spend for CSE123."
        Case 5 To 7
strMsg = " More than enough for this course "
        Case  7 To 9
strMsg = " You will be VBA expert if you continue to study this
    much "
        Case Is > 9
strMsg = "I am speechless"
    End Select
    MsgBox strMsg, vbOKOnly, "Study Hour"
End Sub
```

Hydraulic Radius Example

```
Private Sub CommandButton1_Click()
Dim RRfull, QQfull, vvfull, PPfull, AAfull, Areafull, Aact, Vact, Vfull,
    Pact, Pfull As Double
Dim teta, fi, invercos, D, h, Qact, Qfull As Double
h = Val(TextBox1.Text)
Qact = Val(TextBox2.Text)
D = Val(TextBox3.Text)
UserForm1.Hide
If h > D Then
MsgBox ("Height of water cannot be greater than Diameter of pipe. _
    Please enter all the values again")
UserForm1.Show
UserForm1.Hide
End If
```

```
fi = 1 - ((2 * h) / D)
inverscos = WorksheetFunction.Acos(fi)
teta = 2 * inverscos
Pi = WorksheetFunction.Pi
PPfull = (1 / Pi) * inverscos
RRfull = 1 - (((fi * (1 - fi ^ 2) ^ (1 / 2)) / inverscos))
vvfull = (RRfull) ^ (2 / 3)
AAfull = (1 / Pi) * inverscos - (1 / Pi) * fi * ((1 - fi ^ 2) ^ (1 / 2))
QQfull = vvfull * AAfull
Areafull = (Pi / 4) * (D ^ 2)
Aact = AAfull * Areafull
Pfull = Pi * D
Pact = Pfull * PPfull
Rfull = D / 4
```

```
Ract = Rfull * RRfull
Qfull = Qact / QQfull
Vfull = Qfull / Areafull
Vact = vvfull * Vfull
Cells(3, 1) = "Vact/Vfull"
Cells(3, 2) = "Qact/Qfull"
Cells(3, 3) = "Areaact/Areafull"
Cells(3, 4) = "Ract/Rfull"
Cells(3, 5) = "Pact/Pfull"
Cells(3, 6) = "heigth/Diameter"
Cells(4, 1) = vvfull
Cells(4, 2) = QQfull
Cells(4, 3) = AAfull
Cells(4, 4) = RRfull
Cells(4, 5) = PPfull
Cells(4, 6) = h / D
```

```
If Vact < 0.6 Or Vact > 3 Then
    MsgBox ("Velocity is " & Vact & " and it is not in the range of 0.6-3 _
     m/sec.")
    If MsgBox("Would you like to enter the parameters again?", vbYesNo) _
     = vbYes Then
    UserForm1.Show
    UserForm1.Hide
    Else
    GoTo Terminate
    End If
End If
 End If
```

```
Msg = "Do you want to see all calculated values eg. vfull, Qfull ?"
dec = MsgBox(Msg, vbYesNo)
    If dec = vbYes Then
        Cells(5, 1) = "Vactual"
        Cells(6, 1) = Vact
        Cells(7, 1) = "Vfull"
        Cells(8, 1) = Vfull
        Cells(5, 2) = "Actual flow rate"
        Cells(6, 2) = Qact
        Cells(7, 2) = "Full flow rate"
        Cells(8, 2) = Qfull
        Cells(5, 3) = "Actual area"
        Cells(6, 3) = Aact
```

```
Cells(7, 3) = "Full area"
    Cells(8, 3) = Areafull
    Cells(5, 4) = "Actual hyraulic radius"
    Cells(6, 4) = Ract
    Cells(7, 4) = "Full Hydraulic radius"
    Cells(8, 4) = Rfull
    Cells(5, 5) = "Actual perimeter"
    Cells(6, 5) = Pact
    Cells(7, 5) = "Full perimeter"
    Cells(8, 5) = Pfull
    Cells(5, 6) = "Height of water"
    Cells(6, 6) = h
    Cells(7, 6) = "Diameter of pipe"
    Cells(8, 6) = D
    End If
Terminate:
End Sub
```