DECISIONS & DEBUGGING

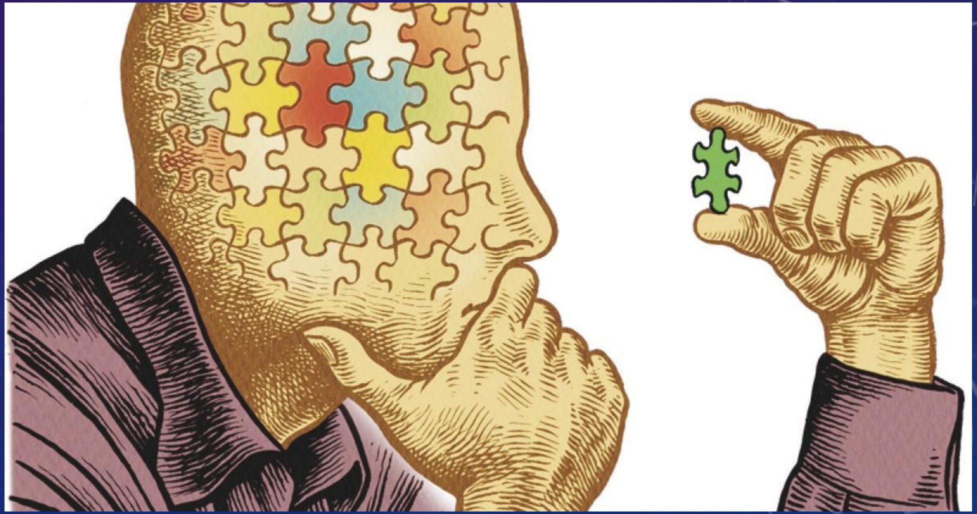
VISUAL BASIC

TABLE OF CONTENTS

1. Lecture context
2. IF then construct
3. Case construct
4. Debugging techniques
5. Single steps
6. Viewing variable contents

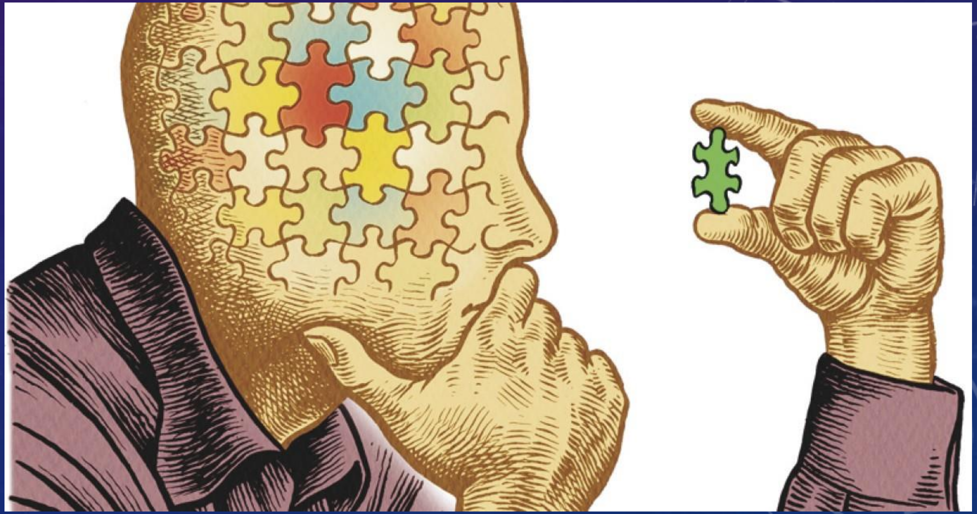
LECTURE CONTEXT

- Be familiar with techniques used to make conditional decisions within a Program
- Be able to decide when to use each technique presented here
- Understand how to nest decisions and how to flowchart conditional decisions



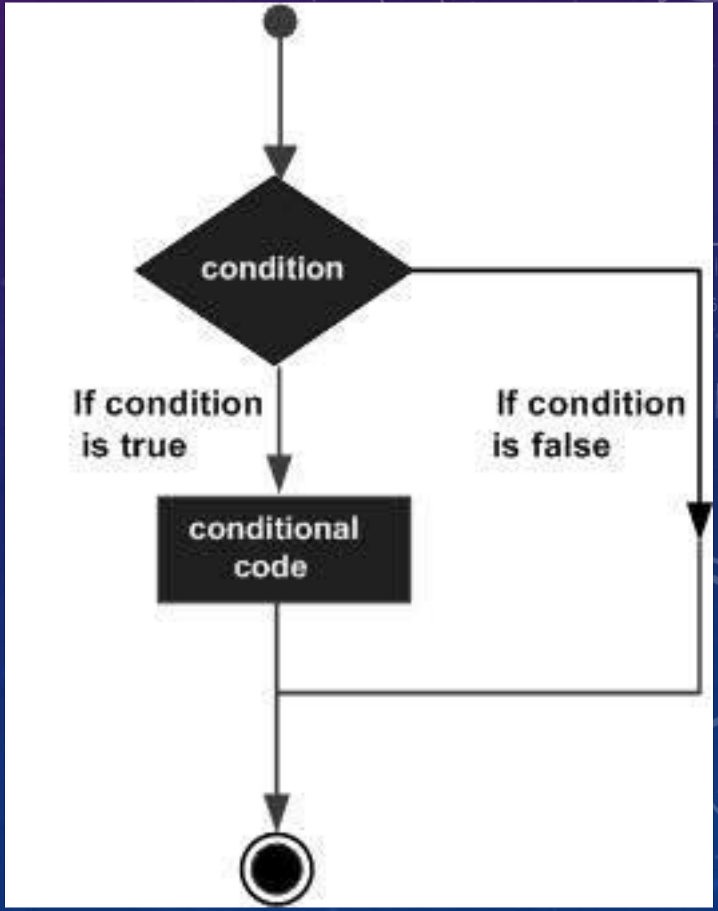
LECTURE CONTEXT

- There are three programming constructs that one can use, inline, iteration, and branching.
- This lecture will look at the tools available in VB to make decisions and divert the flow of code based upon some input or the result of some operation. In instrumentation and control applications, the ability to react and process varying inputs is an essential feature.
- Also, when writing programs, one needs to be able to debug and test them
- The VB2010 IDE incorporates several useful tools for this and we will look at some of those.



CONDITIONAL DECISIONS

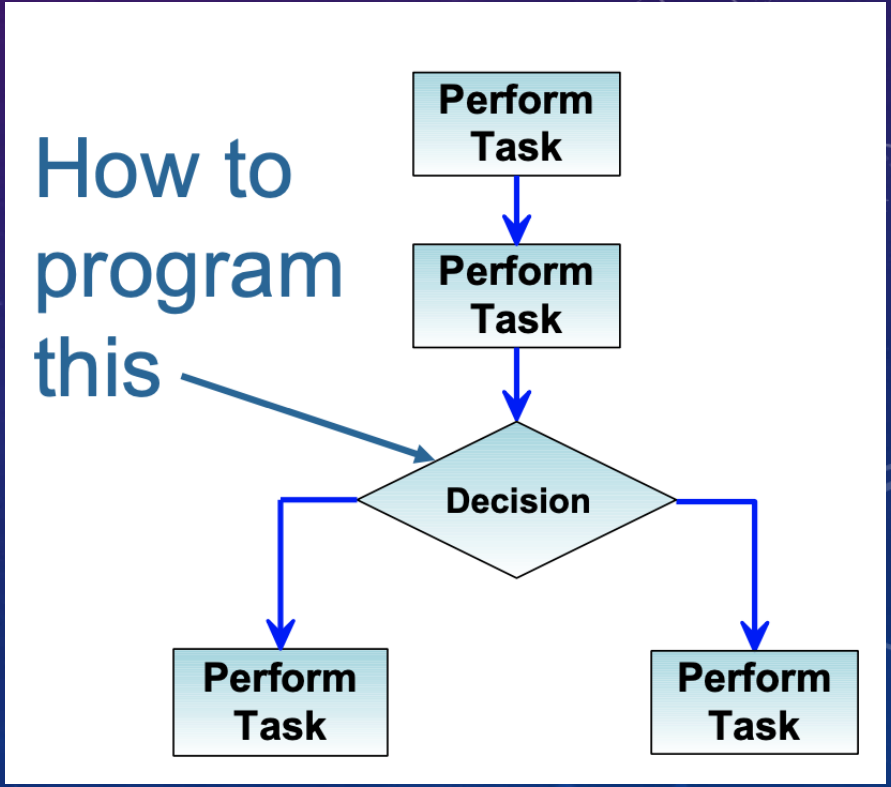
- Programs are not always sequential and may need conditional or unconditional branching constructs
- Decisions made as a result of comms, calculations, user intervention etc.
- Programmers need ways to make their programs flexible and adaptable
- Two constructs are common to most programming languages



IF THEN CONSTRUCT

- Conditional decisions and control of program flow
- Comparison can be x>y, x<y, x = y or x not = y
- Often best to avoid '='
- The **If-Then** construct has
- the basic form:

```
If ( test condition) Then
Insert user code here
End If
```



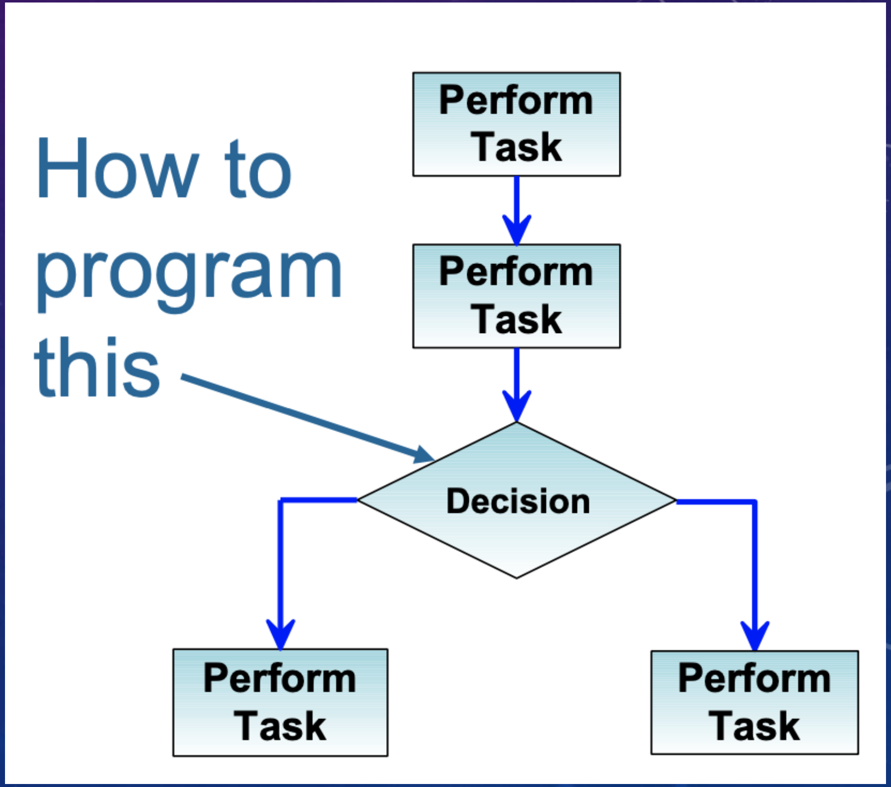
IF THEN CONSTRUCT

- Conditional decisions and control of program flow
- Comparison can be x>y, x<y, x = y or x not = y
- Often best to avoid '='
- The **If-Then** construct has
- the basic form:

If (test condition) Then

Insert user code here

End If



IF THEN CONSTRUCT

Able to test for multiple conditions using **logical** operators

Example 1:

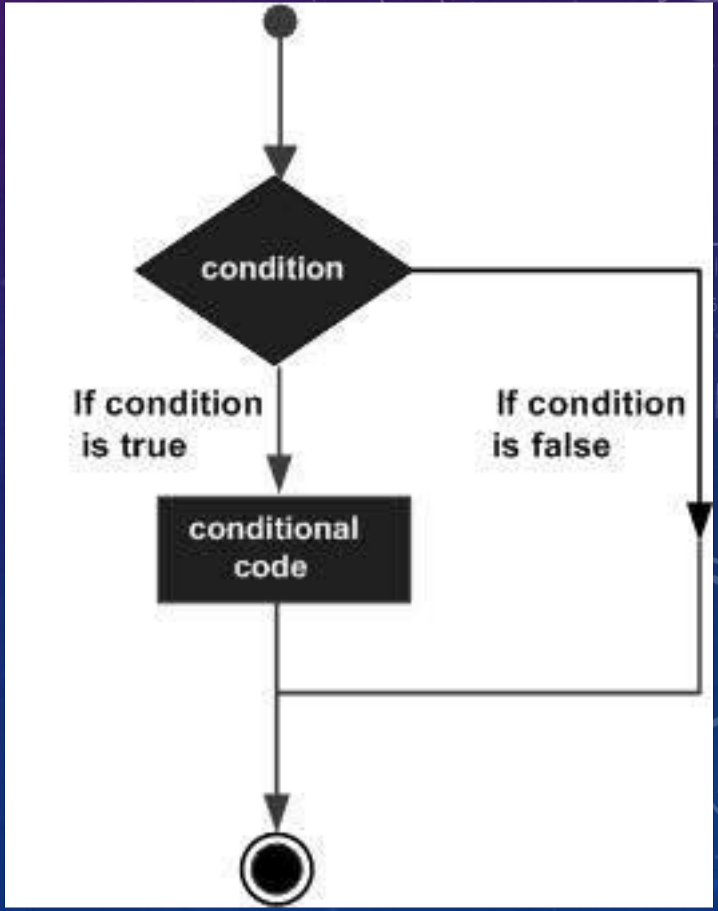
```
If(x=17ANDy=22) Then
        Insert user code here
End If
```

Example 2:

```
If(x = 2 OR x = 3 OR y = 2)
    Then Insert user code here
End If
```

CONDITIONAL DECISIONS

- Programs are not always sequential and may need conditional or unconditional branching constructs
- Decisions made as a result of comms, calculations, user intervention etc.
- Programmers need ways to make their programs flexible and adaptable
- Two constructs are common to most programming languages

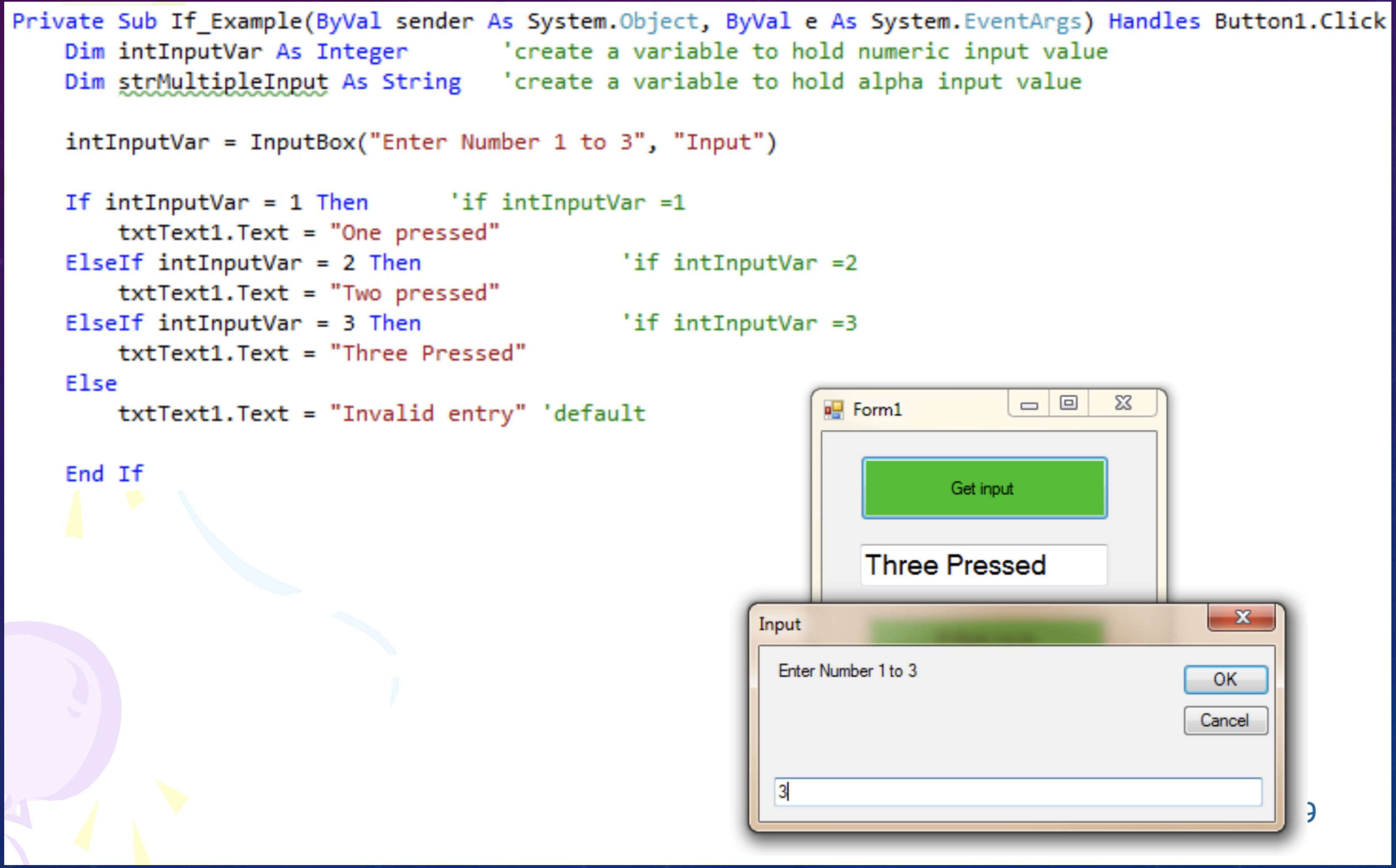


NESTED IF THEN CONSTRUCT

- May be nested
- NOTE: 2 *'If-Then'* requires 2 *'End If' – use comments to track them*
- intOutput only incremented IF BOTH intInput1 and intInput2 = 1

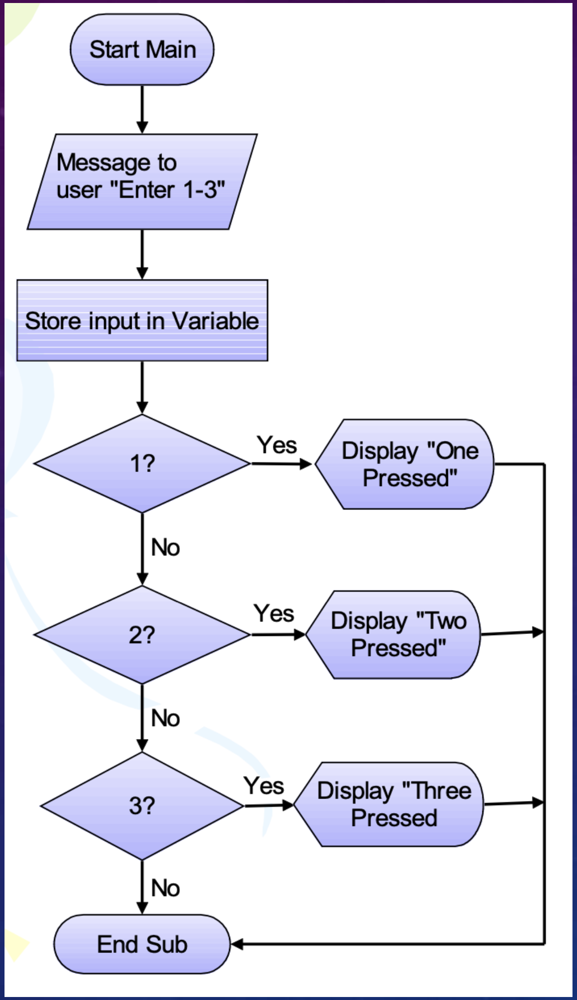
```
If intInput1 = 1 then
    If intInput2 = 1 Then
    intOutput = intOutput + 1
    End If 'intInput2
End If 'intInput1
```

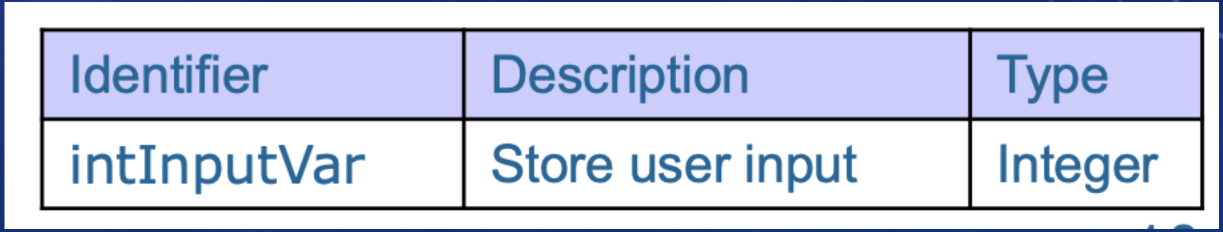
CODE TOGETHER EXAMPLE



```
Private Sub If_Example(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim intInputVar As Integer       'create a variable to hold numeric input value
    Dim strMultipleInput As String   'create a variable to hold alpha input value

    intInputVar = InputBox("Enter Number 1 to 3", "Input")

    If intInputVar = 1 Then      'if intInputVar =1
        txtText1.Text = "One pressed"
    ElseIf intInputVar = 2 Then             'if intInputVar =2
        txtText1.Text = "Two pressed"
    ElseIf intInputVar = 3 Then             'if intInputVar =3
        txtText1.Text = "Three Pressed"
    Else
        txtText1.Text = "Invalid entry" 'default

    End If
```

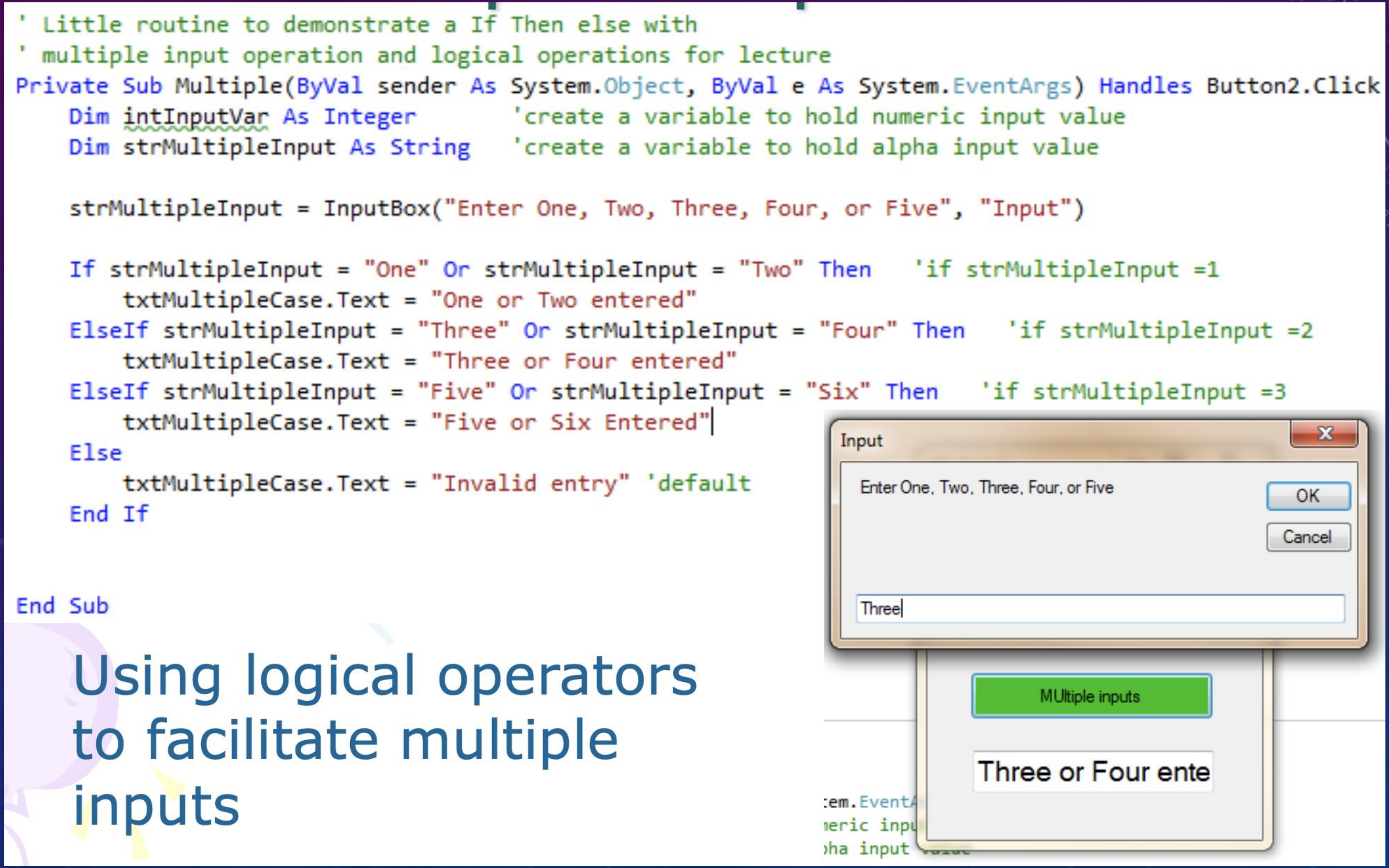
LOGIC BEHIND CODE TOGETHER EXAMPLE





Identifier	Description	Type
intInputVar	Store user input	Integer

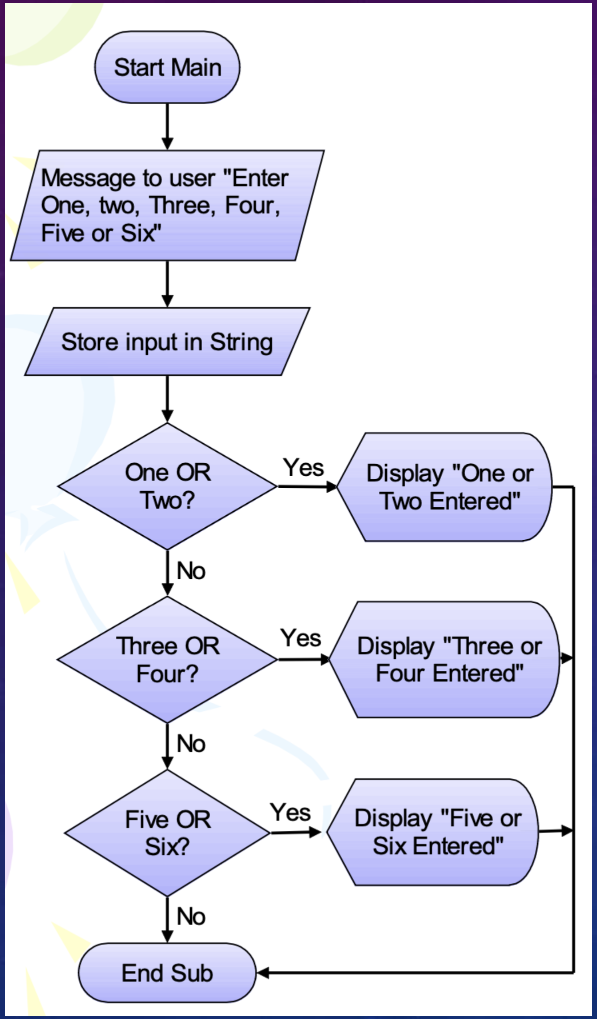
EXAMPLE IF-THEN-ELSE WITH MULTIPLE COMPARISONS

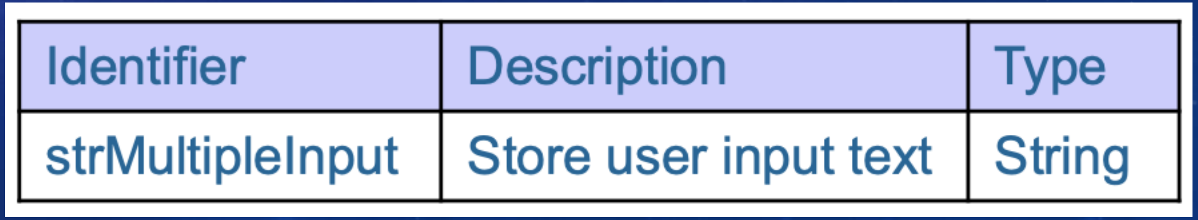


```
' Little routine to demonstrate a If Then else with
' multiple input operation and logical operations for lecture
Private Sub Multiple(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    Dim intInputVar As Integer       'create a variable to hold numeric input value
    Dim strMultipleInput As String   'create a variable to hold alpha input value

    strMultipleInput = InputBox("Enter One, Two, Three, Four, or Five", "Input")

    If strMultipleInput = "One" Or strMultipleInput = "Two" Then   'if strMultipleInput =1
        txtMultipleCase.Text = "One or Two entered"
    ElseIf strMultipleInput = "Three" Or strMultipleInput = "Four" Then   'if strMultipleInput =2
        txtMultipleCase.Text = "Three or Four entered"
    ElseIf strMultipleInput = "Five" Or strMultipleInput = "Six" Then   'if strMultipleInput =3
        txtMultipleCase.Text = "Five or Six Entered"
    Else
        txtMultipleCase.Text = "Invalid entry" 'default
    End If


End Sub
```

Using logical operators to facilitate multiple inputs

LOGIC BEHIND EXAMPLE IF-THEN-ELSE WITH MULTIPLE COMPARISONS


Start Main
Message to user "Enter One, two, Three, Four, Five or Six"
Store input in String
One OR Two?
Yes
Display "One or Two Entered"
No
Three OR Four?
Yes
Display "Three or Four Entered"
No
Five OR Six?
Yes
Display "Five or Six Entered"
No
End Sub



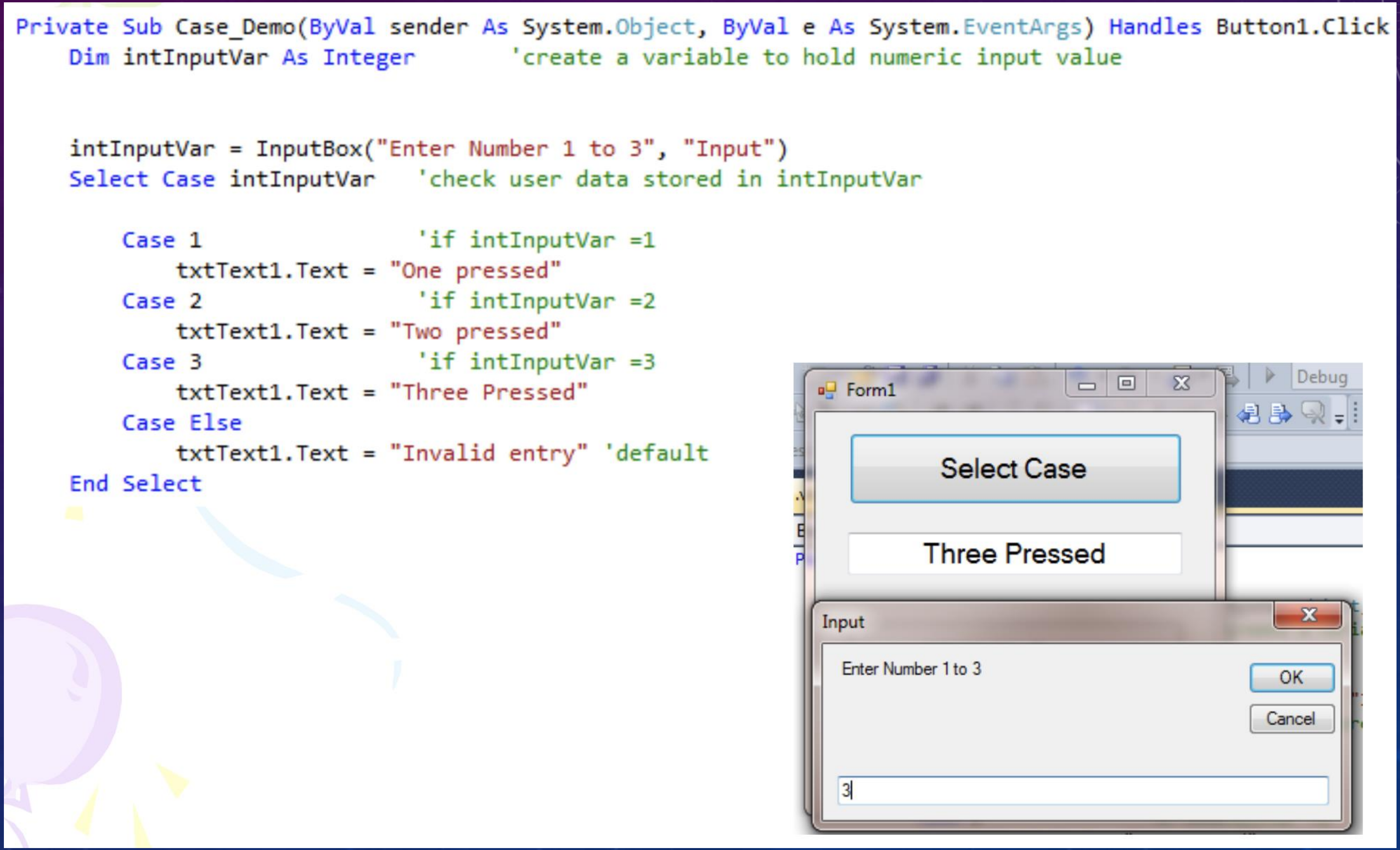
Identifier	Description	Type
strMultipleInput	Store user input text	String

ALTERNATIVE TO IF ELSE

- Used as an alternative to a series of If / else
- Each ‘*expressionlist*’ is a list of one or more conditions to be checked

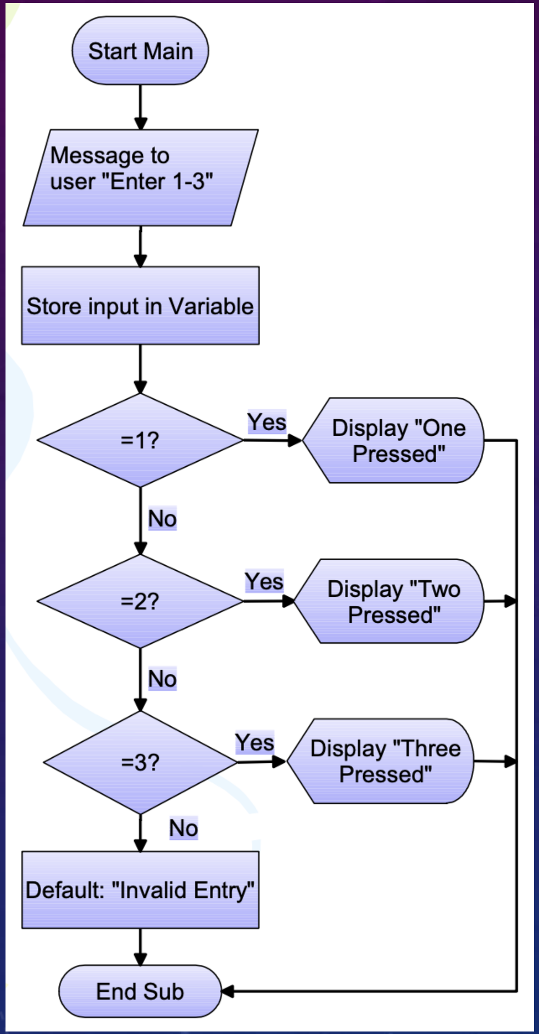
Select Case *testexpression* ***Case*** *expressionlist1*
[statementblock-1] ***Case*** *expressionlist2*
[statementblock-2].
.....
Case Else
[statementblock-n]
End Select

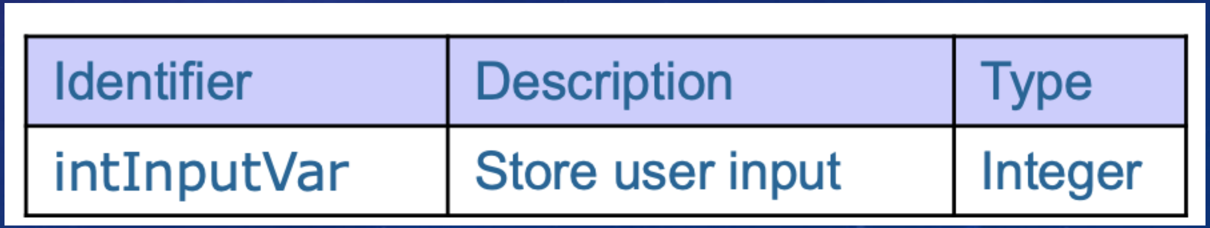
CODE TOGETHER EXAMPLE



```
Private Sub Case_Demo(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim intInputVar As Integer       'create a variable to hold numeric input value


    intInputVar = InputBox("Enter Number 1 to 3", "Input")
    Select Case intInputVar   'check user data stored in intInputVar

        Case 1                  'if intInputVar =1
            txtText1.Text = "One pressed"
        Case 2                  'if intInputVar =2
            txtText1.Text = "Two pressed"
        Case 3                  'if intInputVar =3
            txtText1.Text = "Three Pressed"
        Case Else
            txtText1.Text = "Invalid entry" 'default
    End Select
```

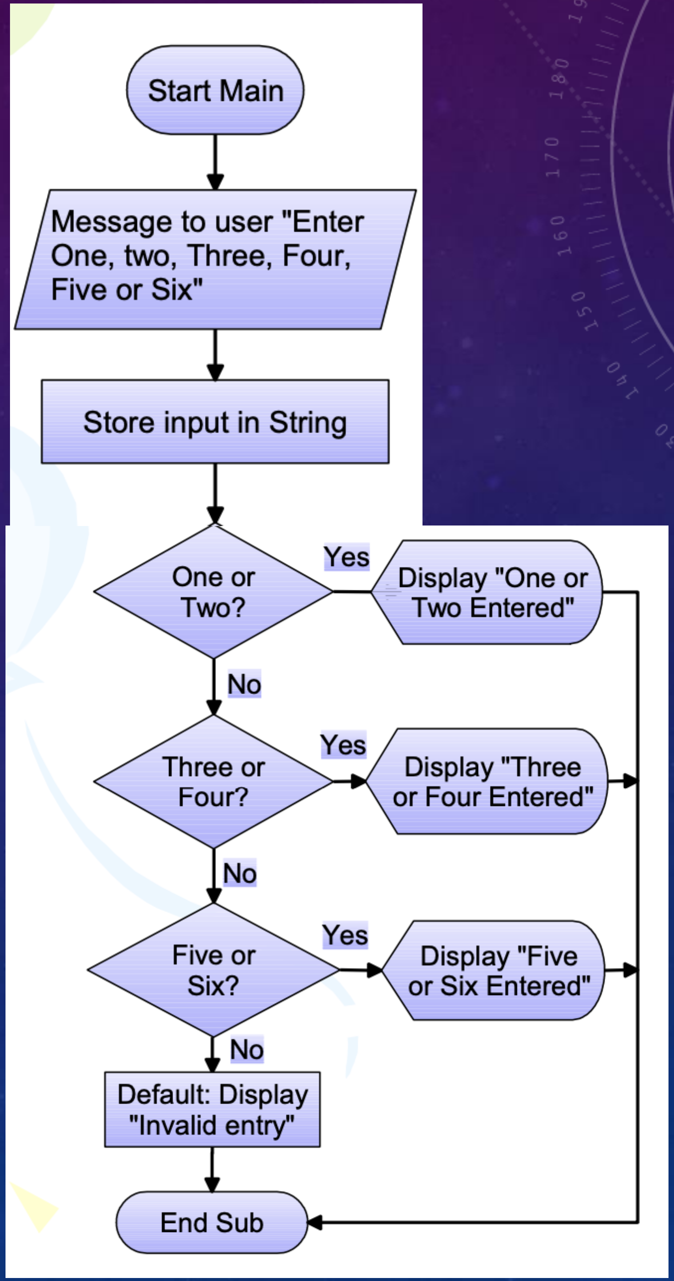
LOGIC BEHIND CODE TOGETHER EXAMPLE





Identifier	Description	Type
intInputVar	Store user input	Integer

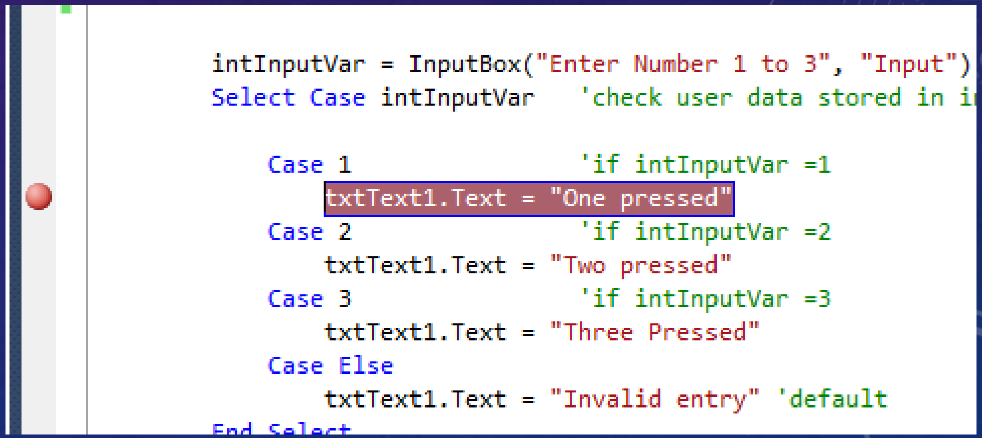
CODE TOGETHER EXAMPLE


Start Main
Message to user "Enter One, two, Three, Four, Five or Six"
Store input in String
One or Two?
Yes
Display "One or Two Entered"
No
Three or Four?
Yes
Display "Three or Four Entered"
No
Five or Six?
Yes
Display "Five or Six Entered"
No
Default: Display "Invalid entry"
End Sub

Identifier	Description	Type
strMultipleInput	Store user input text	String

DEBUGGING TECHNIQUES

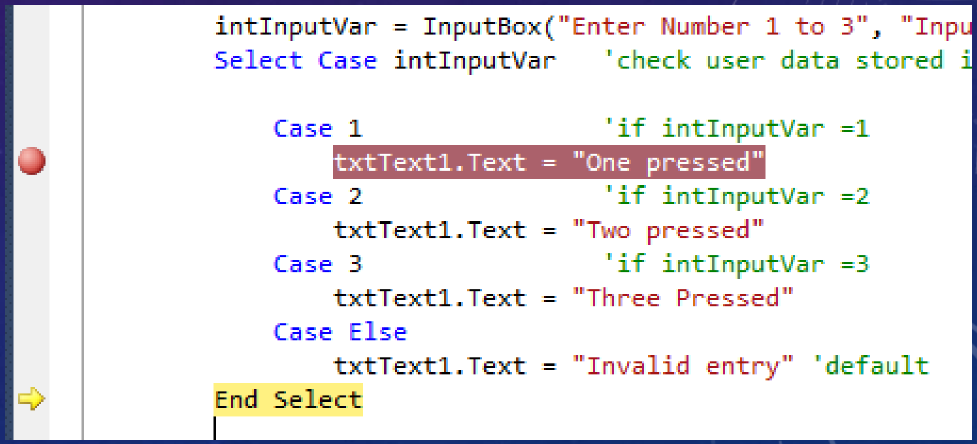
- These are essential techniques for use when writing code – **write small sections**
  and TEST each before moving on
- Execution halted with ***breakpoints***
- Click to left side of code – line highlighted in red and red dot shown
- Code runs line-by- line until *breakpoint* reached
- Similar to other IDE's like Intellij or visual studio code



```
intInputVar = InputBox("Enter Number 1 to 3", "Input")
Select Case intInputVar    'check user data stored in i

    Case 1                 'if intInputVar =1
        txtText1.Text = "One pressed"
    Case 2                 'if intInputVar =2
        txtText1.Text = "Two pressed"
    Case 3                 'if intInputVar =3
        txtText1.Text = "Three Pressed"
    Case Else
        txtText1.Text = "Invalid entry" 'default
End Select
```

SINGLE STEPS

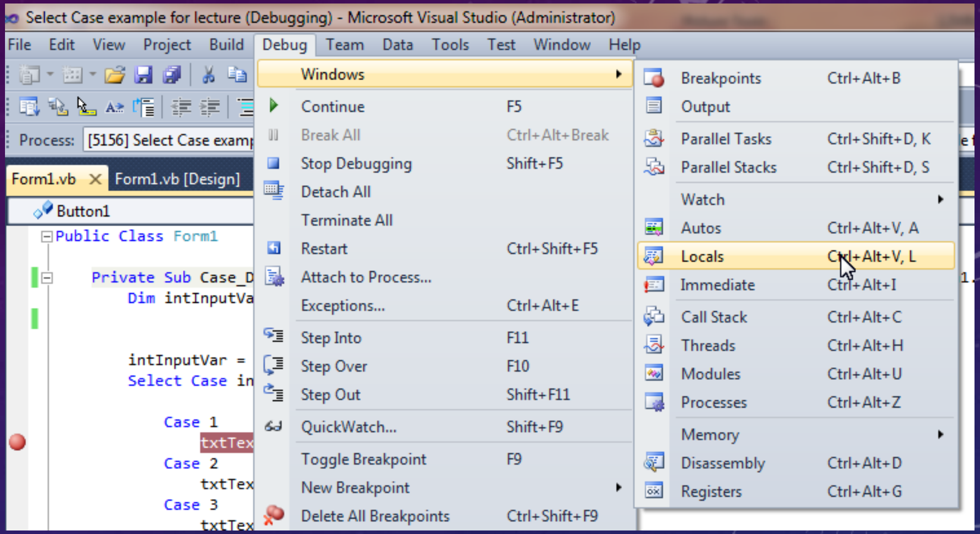
- After breakpoint has paused the code, single step by pressing F11
- Code lines run one at a time which is highlighted in yellow
- Hover mouse over variable to view its contents
- Code runs line-by- line until *breakpoint* reached Or view in one of the debugging windows
-



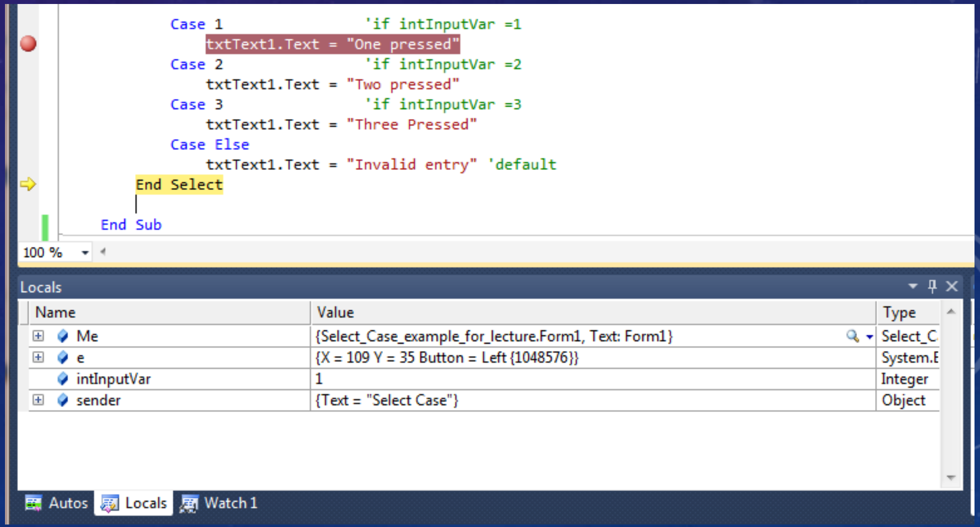
```
intInputVar = InputBox("Enter Number 1 to 3", "Inpu
Select Case intInputVar   'check user data stored i

    Case 1                 'if intInputVar =1
        txtText1.Text = "One pressed"
    Case 2                 'if intInputVar =2
        txtText1.Text = "Two pressed"
    Case 3                 'if intInputVar =3
        txtText1.Text = "Three Pressed"
    Case Else
        txtText1.Text = "Invalid entry" 'default
End Select
```

VIEWING VARIABLE CONTENTS

- Local variable contents can be displayed in locals window – updated on F11 press as code steps
- C Run code to breakpoint and select (only available when running debug) Debug>Windows>Locals



```
Public Class Form1

    Private Sub Case_D
        Dim intInputVa

        intInputVar =
        Select Case in

            Case 1
                txtTex
            Case 2
                txtTex
            Case 3
                txtTex
```



```
            Case 1              'if intInputVar =1
                txtText1.Text = "One pressed"
            Case 2              'if intInputVar =2
                txtText1.Text = "Two pressed"
            Case 3              'if intInputVar =3
                txtText1.Text = "Three Pressed"
            Case Else
                txtText1.Text = "Invalid entry" 'default
        End Select

    End Sub
```

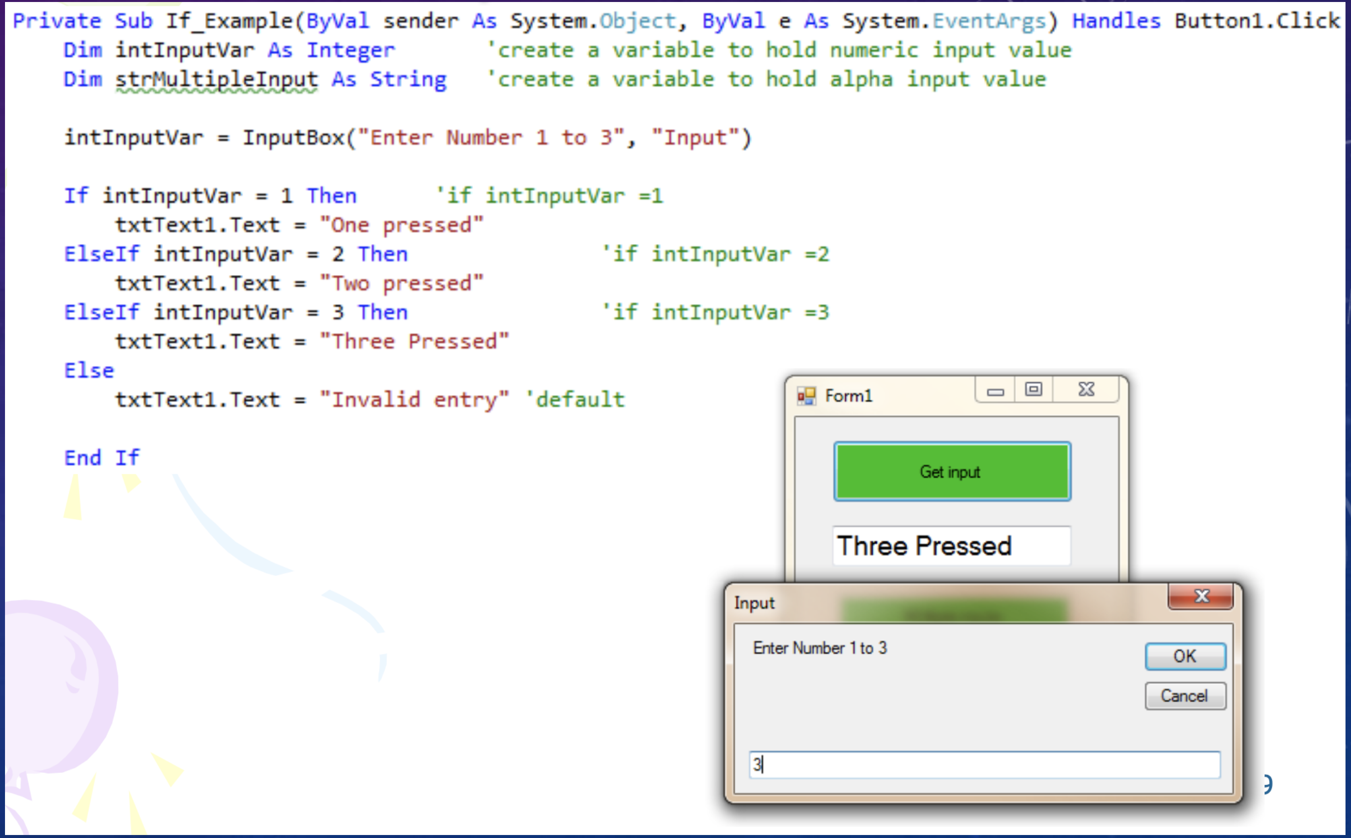
100 %

Locals

Name	Value	Type
Me	{Select_Case_example_for_lecture.Form1, Text: Form1}	Select_C
e	{X = 109 Y = 35 Button = Left {1048576}}	System.E
intInputVar	1	Integer
sender	{Text = "Select Case"}	Object

EXERCISE (10 – 15 MINS)

- Submit this piece of code for this exercise for this particular chapter
- Feel free to expand on this, but minimally your app should have the minimal functionally as shown in the screenshot here



```
Private Sub If_Example(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim intInputVar As Integer       'create a variable to hold numeric input value
    Dim strMultipleInput As String   'create a variable to hold alpha input value

    intInputVar = InputBox("Enter Number 1 to 3", "Input")

    If intInputVar = 1 Then      'if intInputVar =1
        txtText1.Text = "One pressed"
    ElseIf intInputVar = 2 Then             'if intInputVar =2
        txtText1.Text = "Two pressed"
    ElseIf intInputVar = 3 Then             'if intInputVar =3
        txtText1.Text = "Three Pressed"
    Else
        txtText1.Text = "Invalid entry" 'default

    End If
```