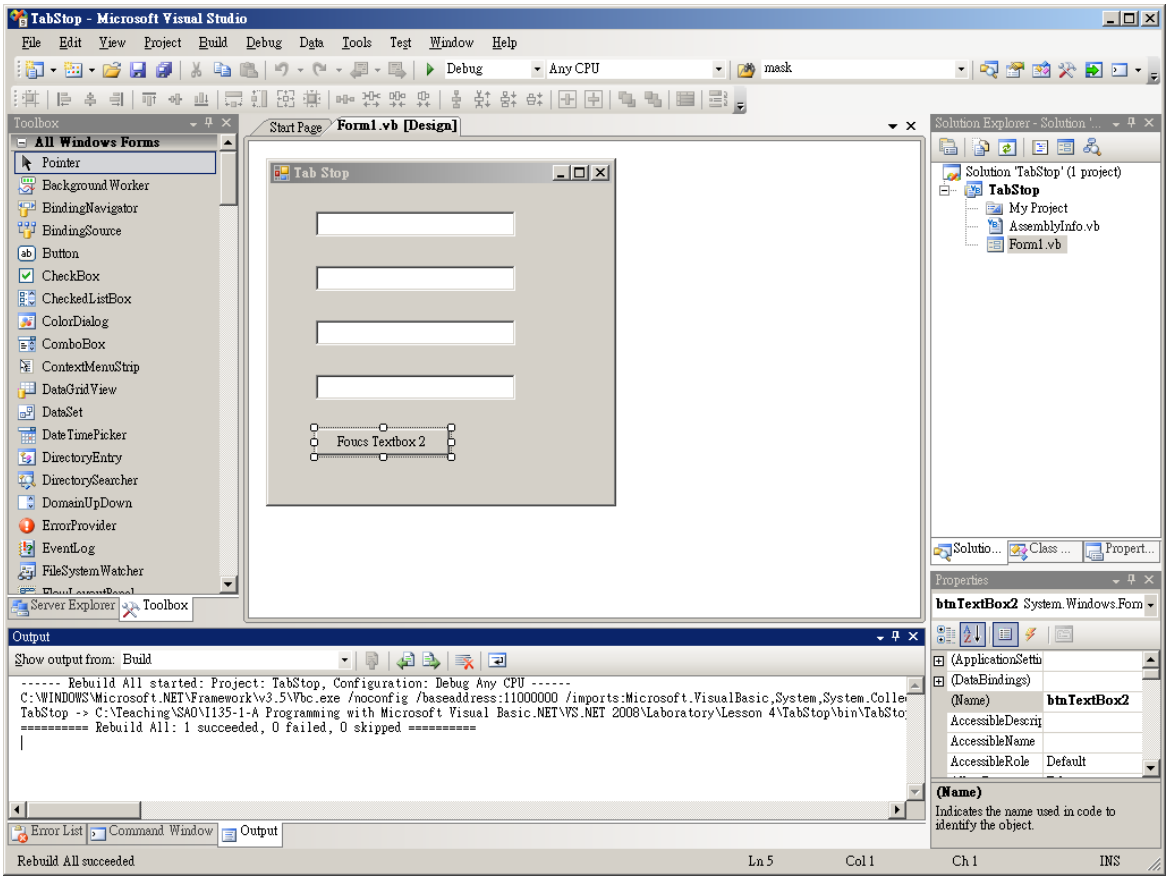
1. Tab Stop and Index

1. Open the Microsoft Visual Studio and create a new Visual Basic Project named as **TabStop**. From the Toolbox, drag 1 **Button** controls and 4 **Textbox** controls onto the form and customize the properties.

Object	Name	Property	Property Value
Form	frmMain	Text	Tab Stop Demo
Button	btnTextBox2	Text	Focus TextBox 2
		TabIndex	4
		TabStop	True
TextBox	TextBox1	TabIndex	0
		TabStop	True
	TextBox2	TabIndex	3
		TabStop	True
	TextBox3	TabIndex	2
		TabStop	False
	TextBox4	TabIndex	1
		TabStop	True



2. In the **Click** event procedure of the Focus TextBox 2 (**btnTextBox2)**, add the following code

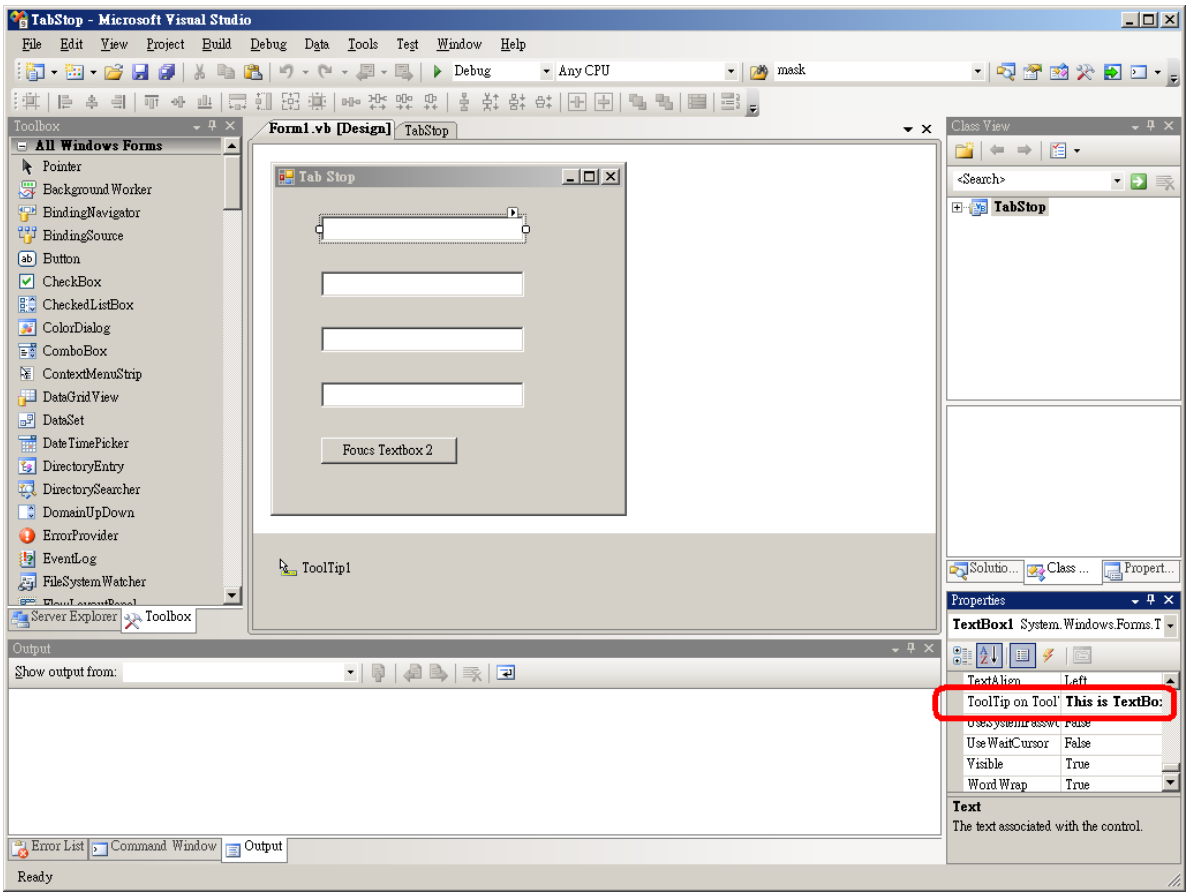
```
' Set focus in Textbox 2
TextBox2.Focus()
```

3. Save the project and build the solution, and then execute it.
   A. When you press the **[Tab]** key, which text box does not stop?
   B. What will happen if you press the **[Focus TextBox2]** button?
   C. Try to put more control to the form, which control will not stop after press the **[Tab]** key?

2. Tool Tip

1. Start the Microsoft Visual Studio and open the previous Visual Basic Project **TabStop**. From the Toolbox, drag a ToolTip control onto the form and customize the properties.

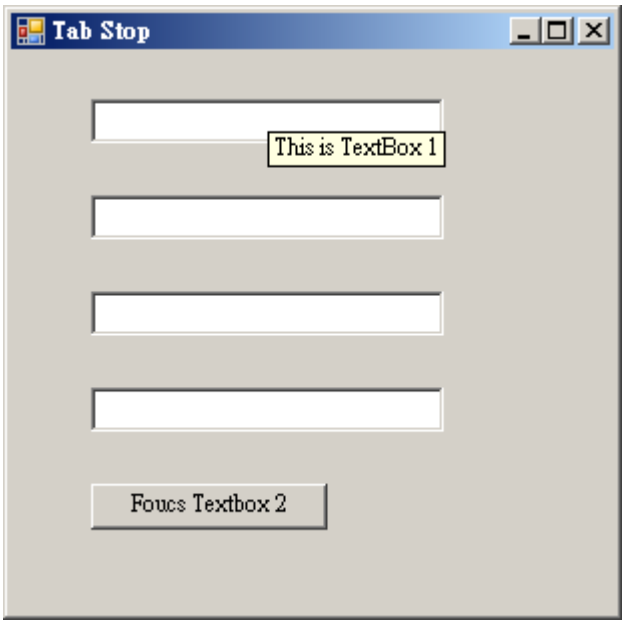
Object	Name	Property	Property Value
ToolTip	ToolTip1		



2. Add the tool tip to the control by add the tool tip in the **ToolTip on ToolTip1** properties of each control.

Object	Name	Property	Property Value
Form	frmMain	ToolTip on ToolTip1	This is Form
Button	btnTextBox2	ToolTip on ToolTip1	This is Button
TextBox	TextBox1	ToolTip on ToolTip1	This is Textbox 1
	TextBox2	ToolTip on ToolTip1	This is Textbox 2
	TextBox3	ToolTip on ToolTip1	This is Textbox 3
	TextBox4	ToolTip on ToolTip1	This is Textbox 4

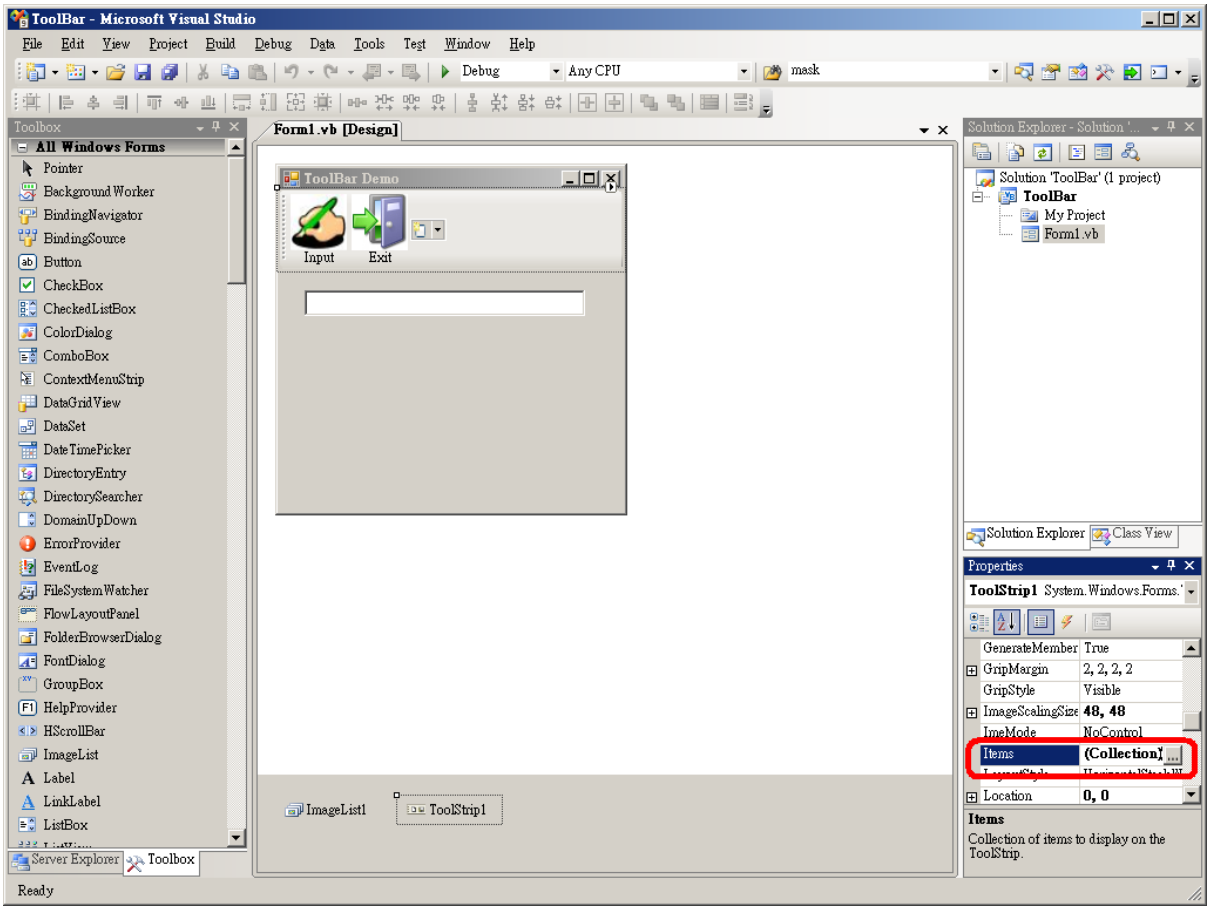
3. Save the project and build the solution, and then execute it. You can see the tool tip by putting your mouse on the control over one second.



3. Tool Bar

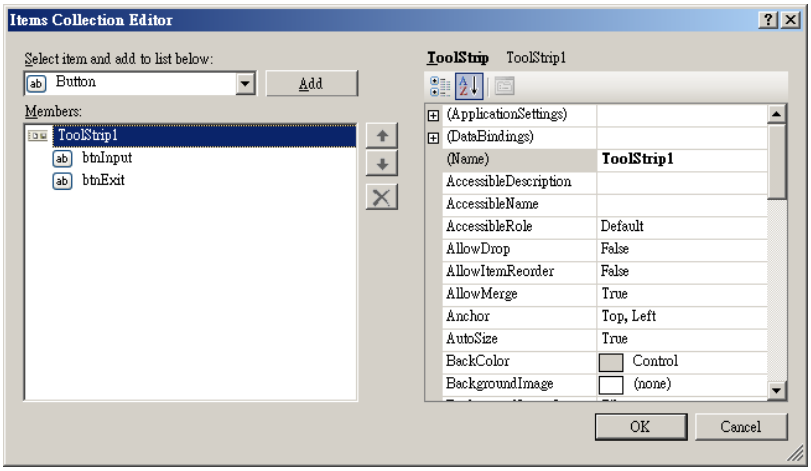
1. Start the Microsoft Visual Studio and create a new Visual Basic Project **ToolBar**. From the Toolbox, drag a **ToolStrip** control and a **TextBox** control onto the form and customize the properties.

Object	Name	Property	Property Value
Form	frmMain	Text	Tool Bar Demo
ToolStrip	ToolStrip1	ImageScaleSize Width	48
		ImageScaleSize Height	48
TextBox	txtOutput	Text	*(Blank)*



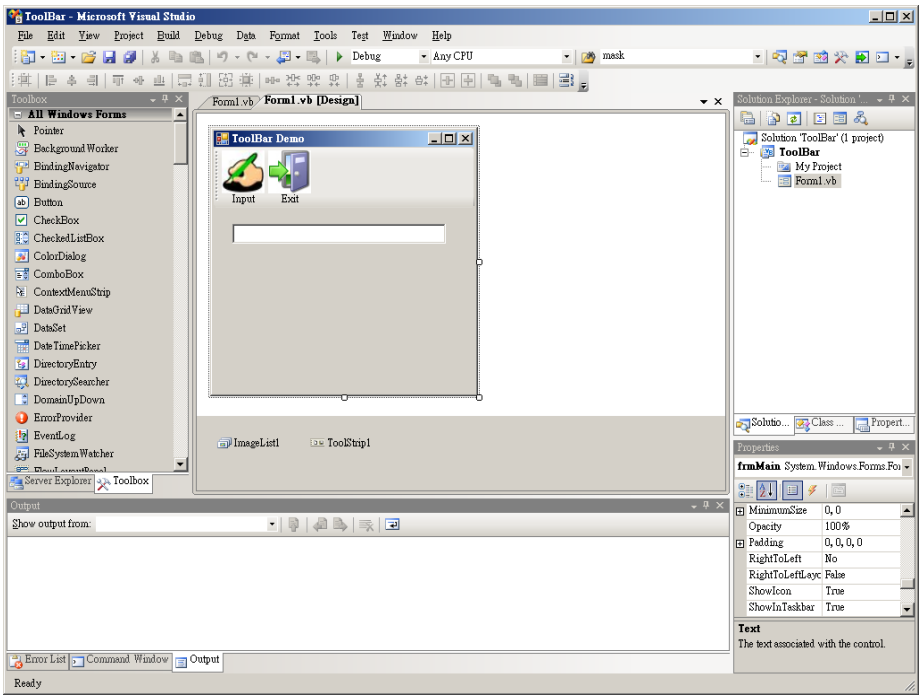
2. Download the image file from http://www.peter-lo.com/Teaching/I154-1-A/Source06.zip and unzip it to your local drive.

3. Select the **ToolStrip1** control and click on the **[(Collection)… ”]** button on Item properties to open the **ToolBoxButton Collection Editor**.



4. Use **[Add]** button to insert the two buttons to the member list. Then press **[OK]** button after you customize the properties.

Object	Name	Property	Property Value
ToolStripButton	btnInput	Text	Input
		Image	*(New.jpg)*
		Display Style	Image and Text
		Text Image Relation	ImageAboveText
	btnExit	Text	Exit
		Image	*(Exit.jpg)*
		Display Style	Image and Text
		TextImageRelation	ImageAboveText



5. In the **Click** event procedure of the Input ToolStripButton control (**btnInput)**, add the following code.

```
' Obtain user input from dialog box, and display in textbox
txtOutput.Text = Microsoft.VisualBasic.InputBox("What is your name?")
```

6. In the **Click** event procedure of the Exit ToolStripButton control (**btnExit)**, add the following code.

```
' Exit the program
End
```

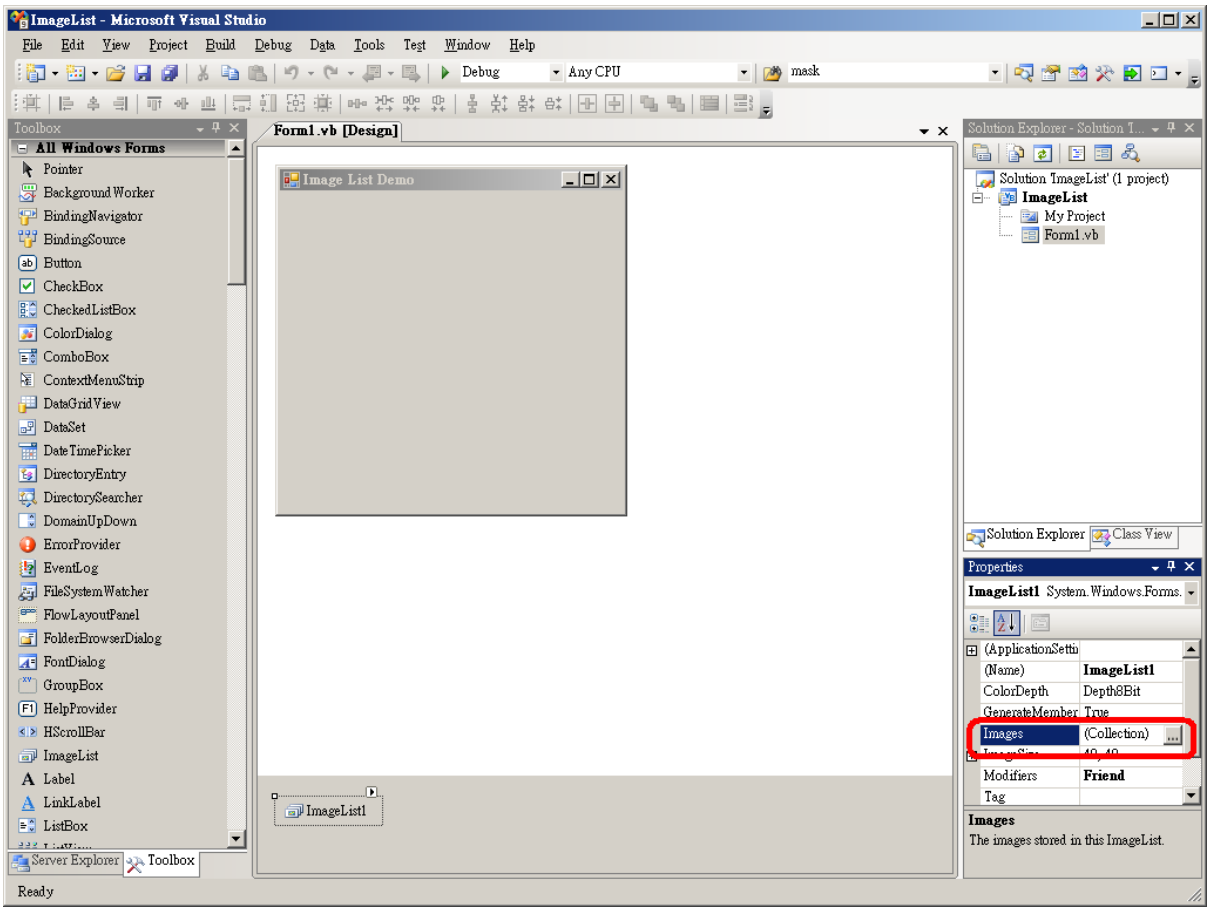
7. Save the project and build the solution, and then execute it.
   A. Press the **[Input]** button in the toolbar, and then input your name and press **[Enter]**. What can you observe?

   B. Press the **[Exit]** button to quit the application.

4. Image List

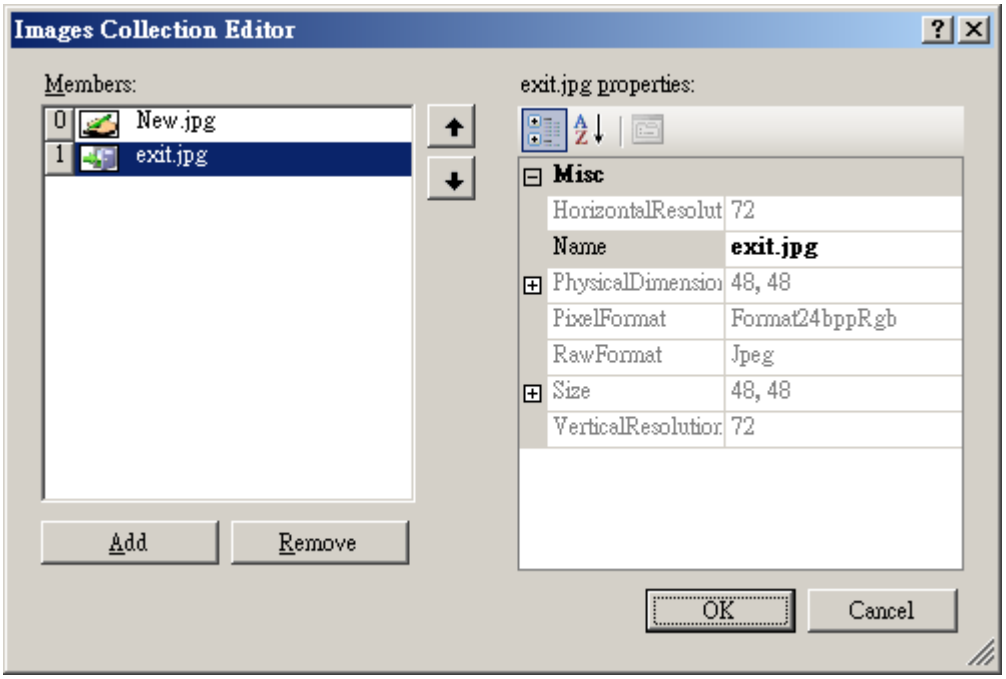
1. Start the Microsoft Visual Studio and create a new Visual Basic Project **ImageList**. From the Toolbox, drag an **Image List** component onto the form and customize the properties.

Object	Name	Property	Property Value
Form	frmMain	Text	Image List Demo
ImageList	ImageList1	Image Size Width	32
		Image Size Height	32

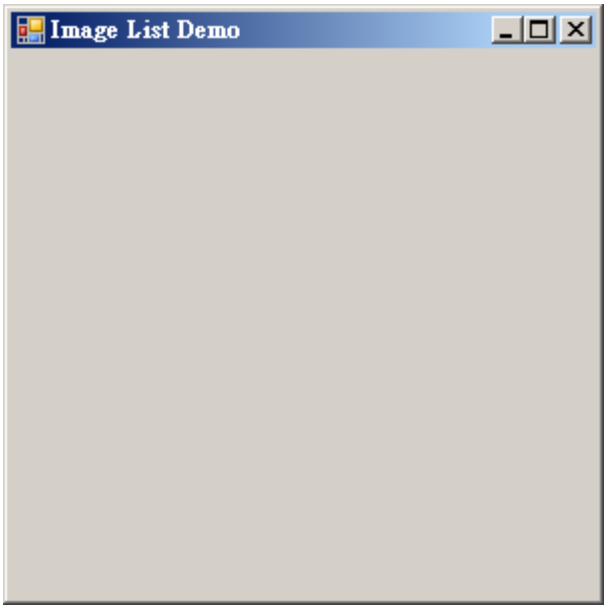


2. Download the image file from http://www.peter-lo.com/Teaching/I154-1-A/Source06.zip and unzip it to your local drive.

3. Click on the **[(Collection)…]** button of the **Images** properties of the **Image List** component. Use **[Add]** to insert the "Exit" image to the member list, and then press **[OK]** when finish.



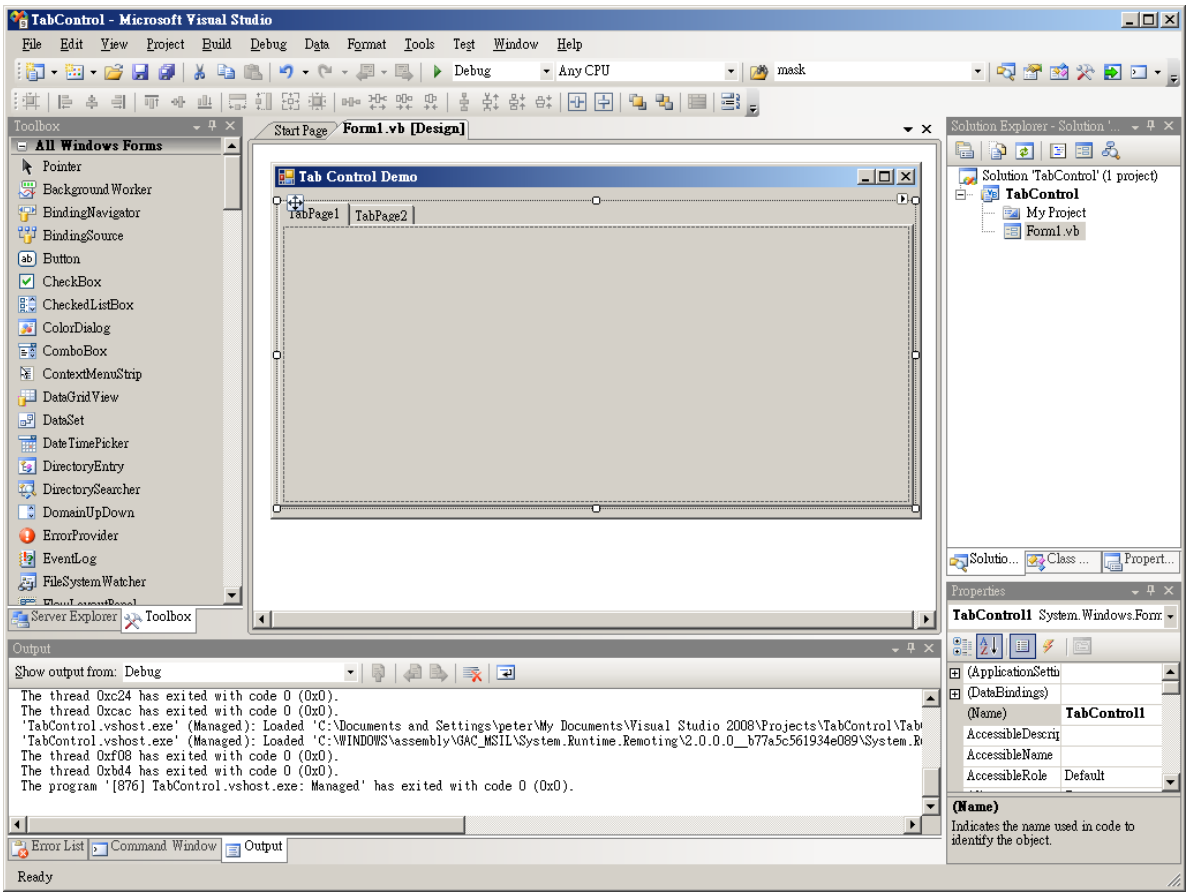
4. Save the project and build the solution, and then execute it. What can you observe?



5. Tab Control

1. Start the Microsoft Visual Studio and open the previous Visual Basic Project **ImageList**. From the Toolbox, drag a **Tab** control onto the form and customize the properties.

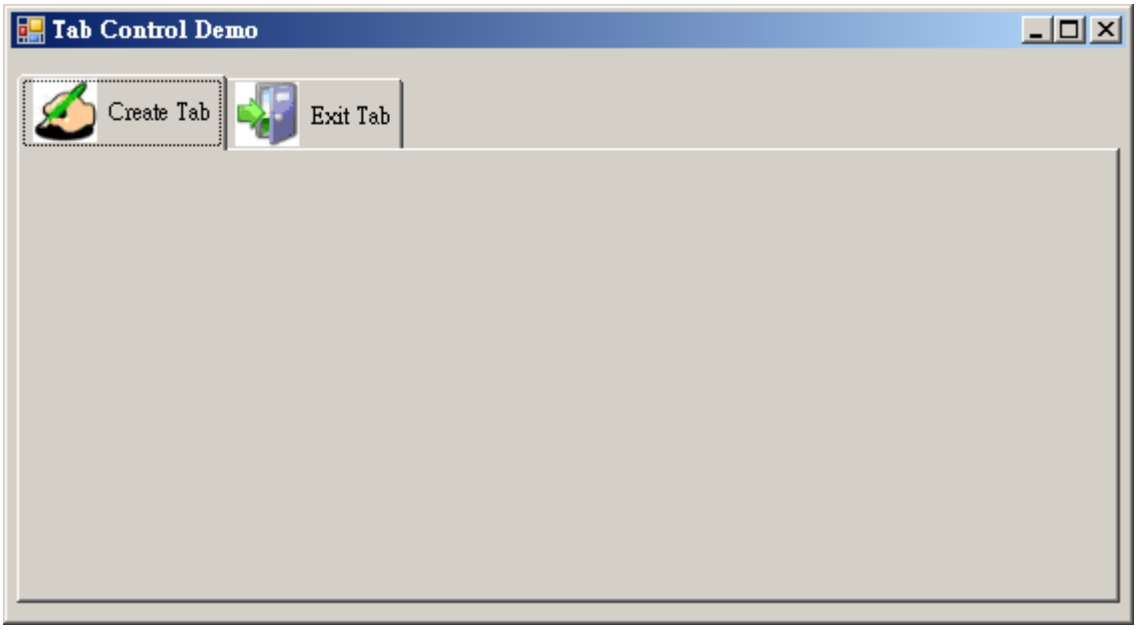
Object	Name	Property	Property Value
TabControl	TabControl1	ImageList	ImageList1



2. Select the **Tab Control** and click the **[(Collection)… ”]** button on the TabPage properties to open the **ToolPage Collection Editor** and customize the properties.

Object	Name	Property	Property Value
TabPage	TabPage1	Text	Create Tab
		ImageIndex	0
	TabPage2	Text	Exit Tab
		ImageIndex	1

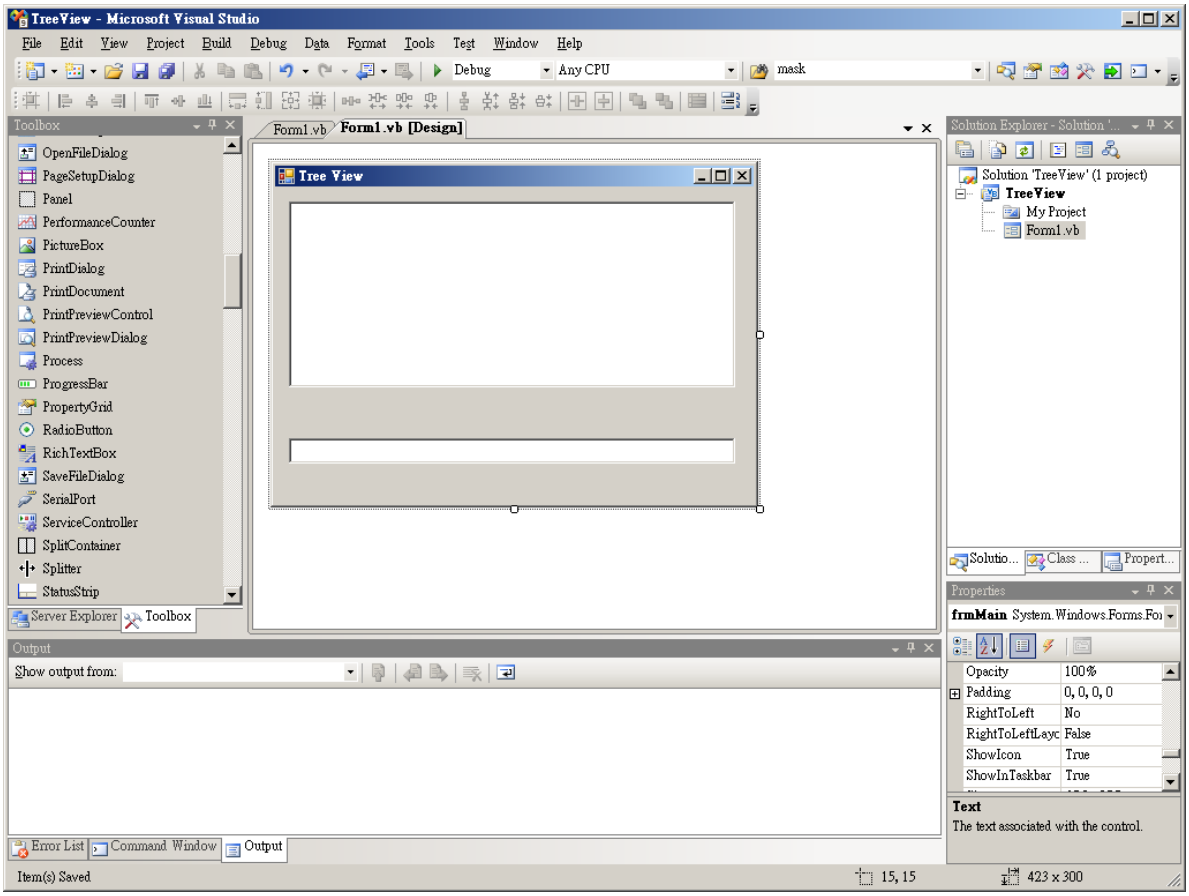
3. Save the project and build the solution, and then execute it.



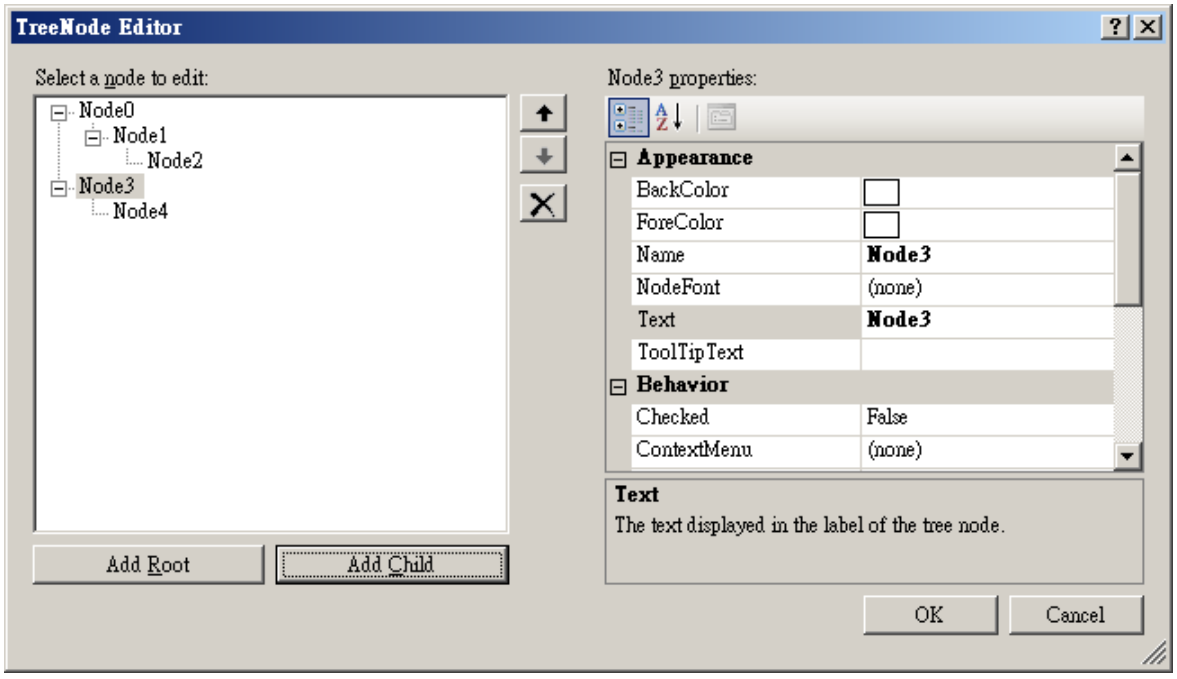
6. Tree View

1. Start the Microsoft Visual Studio and create a new Visual Basic Project **TreeView**. From the Toolbox, drag a **Textbox** control and a **Tree View** control onto the form and customize the properties.

Object	Name	Property	Property Value
Form	frmMain	Text	Tree View Demo
Textbox	txtOutput	Text	*(Blank)*
Tree View	TreeView1		



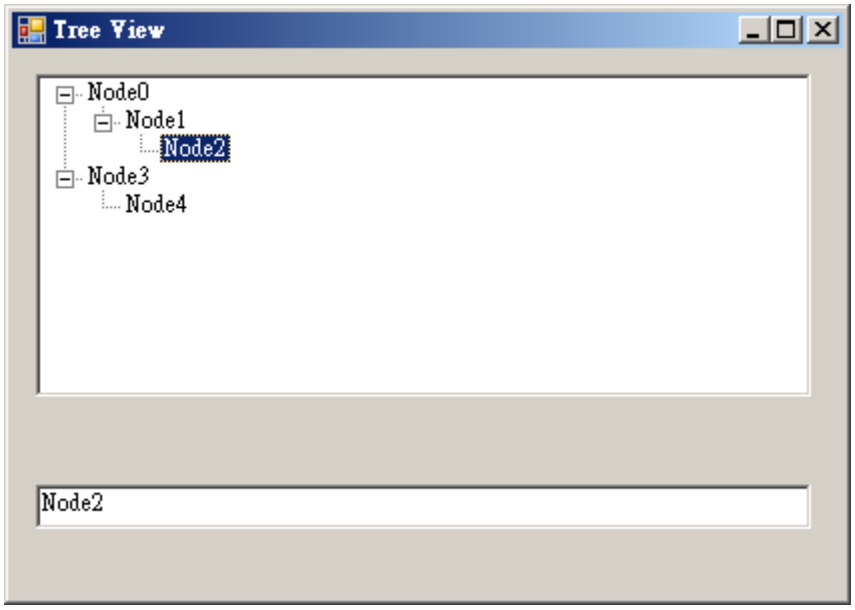
2. Click on the **[(Collection)…]** button of the **Node** properties of the **Tree View** control. Use **[Add Root]** button to add root and parent nodes, and then use **[Add Child]** button to append child nodes. You can also use **[Delete]** button to erase the unwanted nodes. Press **[OK]** when finish, then you can see the design in IDE.



3. In the **AfterSelect** event procedure of the Tree View control (**TreeView1)**, add the following code.

```
‘ Show the text of each node on textbox
txtOutput.Text = e.Node.Text
```

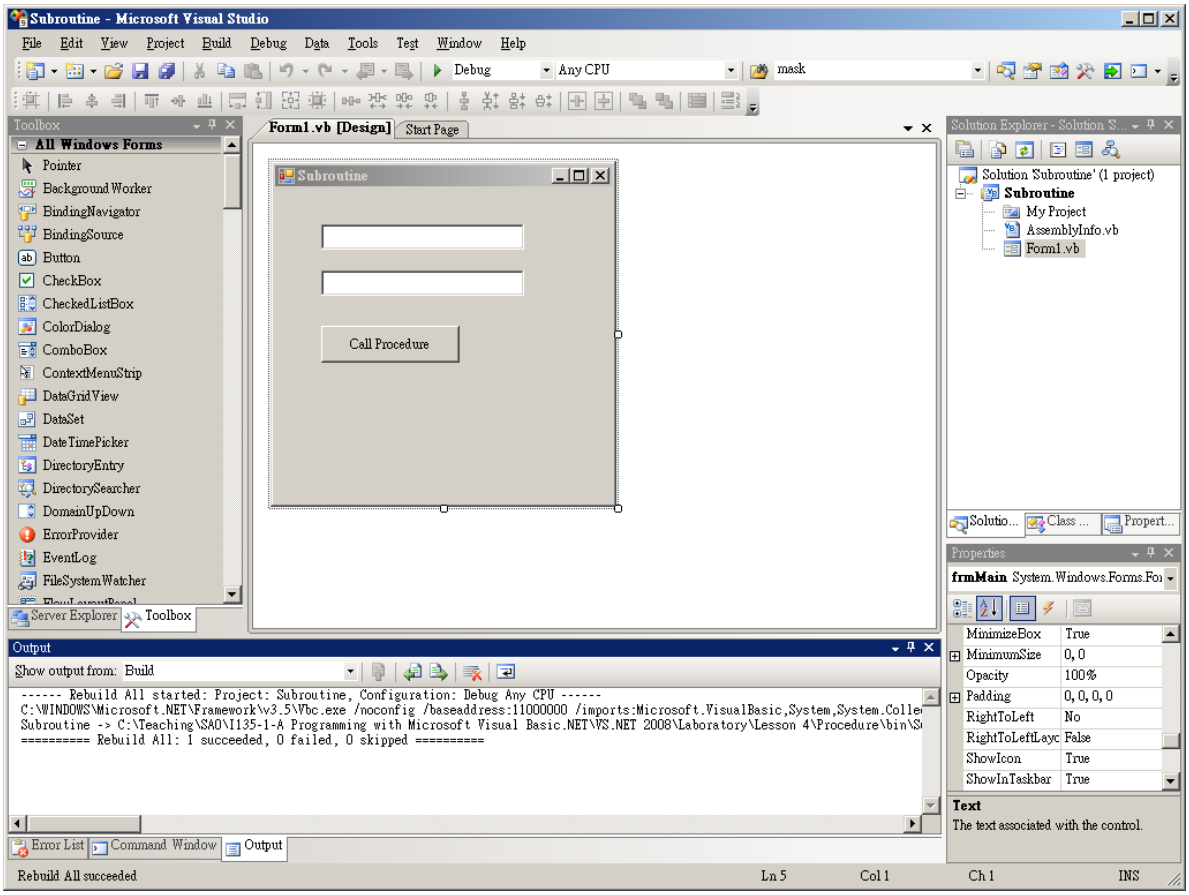
4. Save the project and build the solution, and then execute it.



7. Subroutine – Calling Procedure

1. Open the Microsoft Visual Studio and create a new Visual Basic Project **Subroutine**. From the Toolbox, drag a **Textbox** control and two **Button** controls onto the form and customize the properties.

Object	Name	Property	Property Value
Form	frmMain	Text	Subroutine
Textbox	TextBox1	Text	*(Blank)*
	TextBox2	Text	*(Blank)*
Button	btnProcedure	Text	Call Procedure



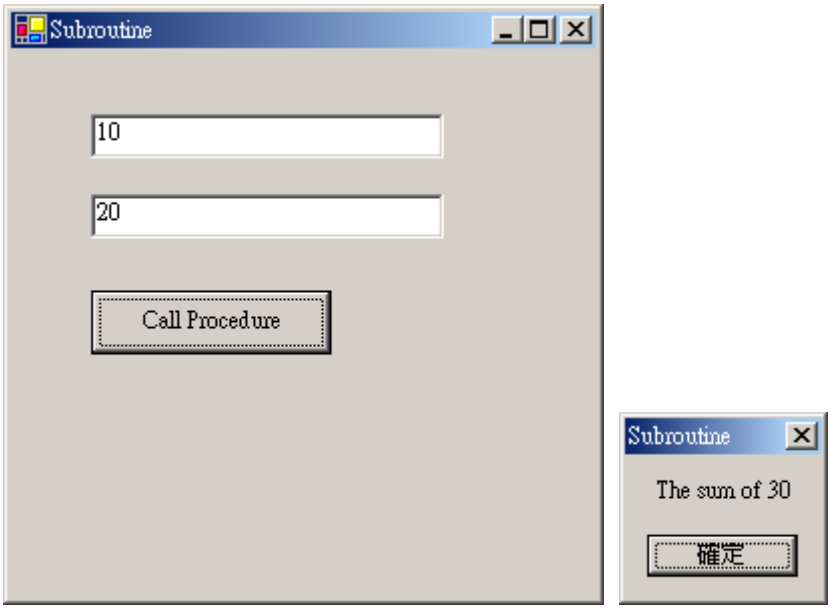
2. Create a new procedure **ProcedureDemo** within the class to calculate the sum for two variables.

```
Private Sub ProcedureDemo(ByVal var1 As Integer, ByVal var2 As Integer)
    ' Declare the variables
    Dim sum As Integer

    ' Sum the var1 and var2
    sum = var1 + var2

    ' Output the Sum
    MsgBox("The sum is " & sum)
End Sub
```

3. In the **Click** event procedure of the Call Procedure button control (**btnProcedure**), add the following code.

```
' Declare the variables
Dim a, b As Integer

' Convert Textbox value to integer
a = CInt(TextBox1.Text)

' Convert Textbox value to integer
b = CInt(TextBox2.Text)

' Call the subroutine to calculate the sum
Call ProcedureDemo(a, b)
```

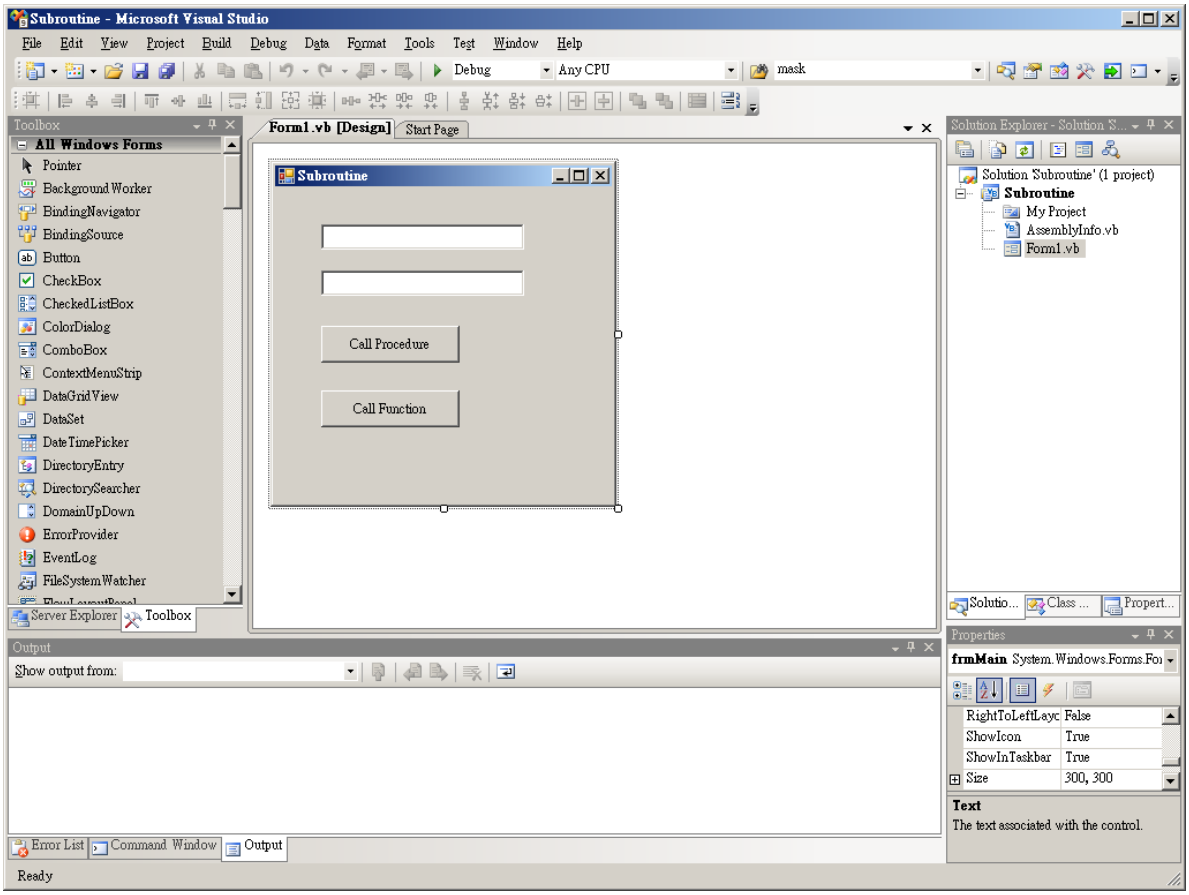
4. Save the project and build the solution, and then execute it.



8. Subroutine – Calling Function

1. Start the Microsoft Visual Studio and open the previous Visual Basic Project **Subroutine**. From the Toolbox, drag a **Button** control onto the form and customize the properties.

Object	Name	Property	Property Value
Button	btnFunction	Text	Call Function



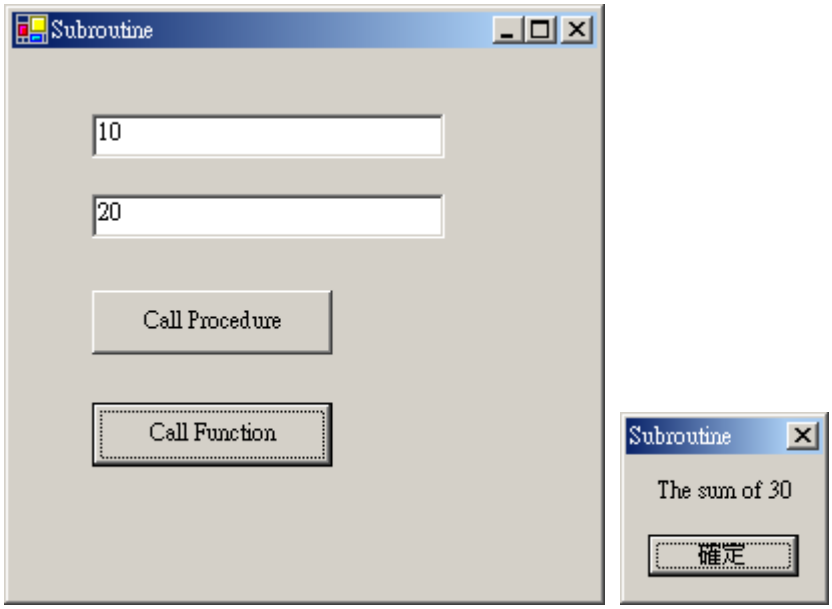
2. Create a new procedure **FunctionDemo** within the class to calculate the sum for two variables.

```
Private Function FunctionDemo(  ByVal var1 As Integer, _
                                ByVal var2 As Integer) As Integer
    ' Sum the var1 and var2
    FunctionDemo = var1 + var2
End Function
```

3. In the **Click** event procedure of the Call Function button control (**btnFunction**), add the following code.

```
' Declare the variables
Dim a, b, c As Integer

' Convert Textbox value to integer
a = CInt(TextBox1.Text)

' Convert Textbox value to integer
b = CInt(TextBox2.Text)

' Call the subroutine to calculate the sum
c = FunctionDemo(a, b)

' Output the Sum
MsgBox("The sum is " & c)
```

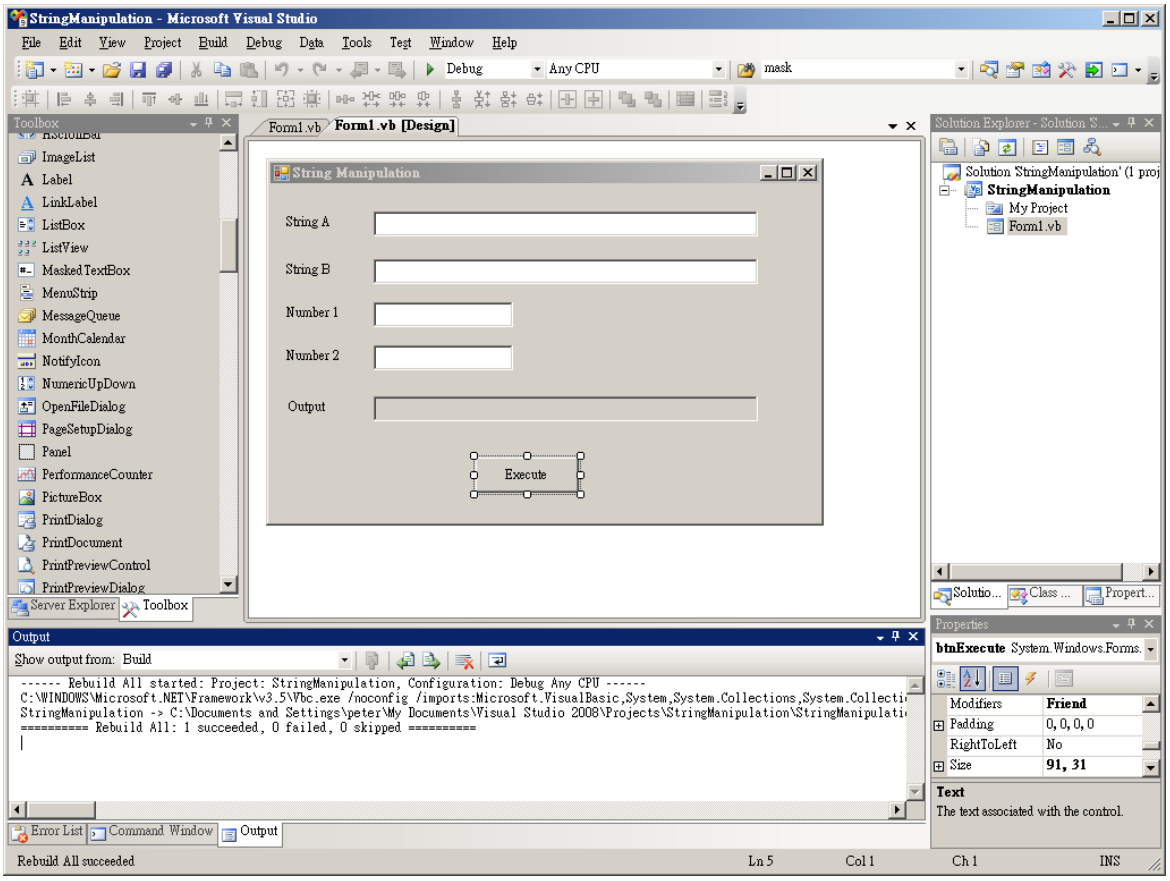
4. Save the project and build the solution, and then execute it.



9. String Manipulation – String Length

1. Start the Microsoft Visual Studio and create a new Visual Basic Project **StringLength**. From the Toolbox, drag 5 **Label** controls, 5 **Textbox** controls and a **Button** control onto the form and customize the properties.

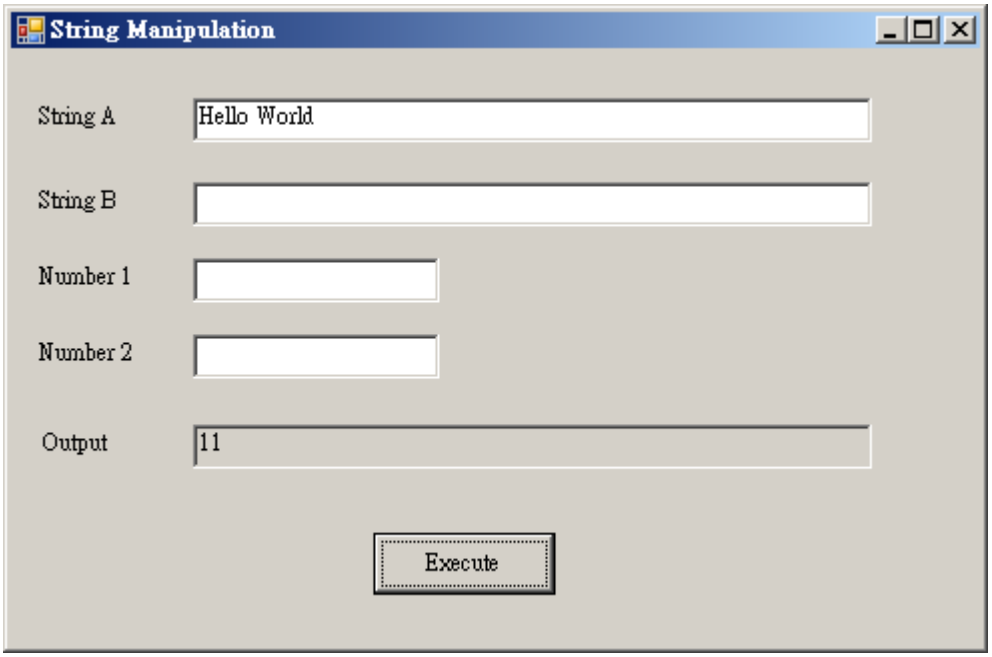
Object	Name	Property	Property Value
Form	frmMain	Text	String Manipulation
Label	Label1	Text	String A
	Label2	Text	String B
	Label3	Text	Number 1
	Label4	Text	Number 2
	Label5	Text	Output
Textbox	txtStrA	Text	*(Blank)*
	txtStrB	Text	*(Blank)*
	txtNum1	Text	*(Blank)*
	txtNum2	Text	*(Blank)*
	txtOutput	Text	*(Blank)*
Button	btnExecute	Text	Execute



2. In the **Click** event procedure of the Execute button (**btnExecute**), add the following code:

```
' Calculate the length of the String
txtOutput.Text = txtStrA.Text.Length
```

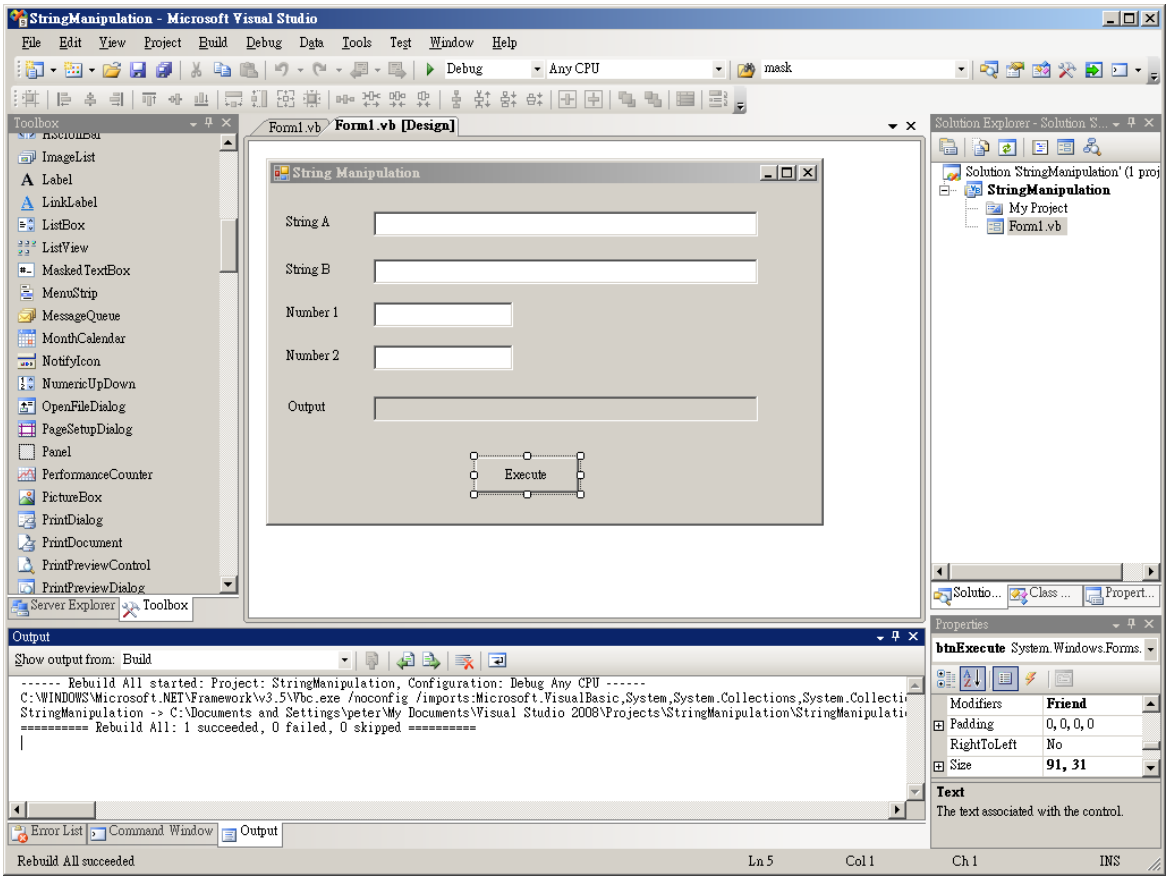
3. Save the project and build the solution, and then execute it. Input a string in “String A” textbox, and then press the **[Execute]** button, can you get the length of your input string?



10. String Manipulation – Trimming

1. Start the Microsoft Visual Studio and open a new Visual Basic Project **Trimming**. From the Toolbox, drag 5 **Label** controls, 5 **Textbox** controls and a **Button** control onto the form and customize the properties.

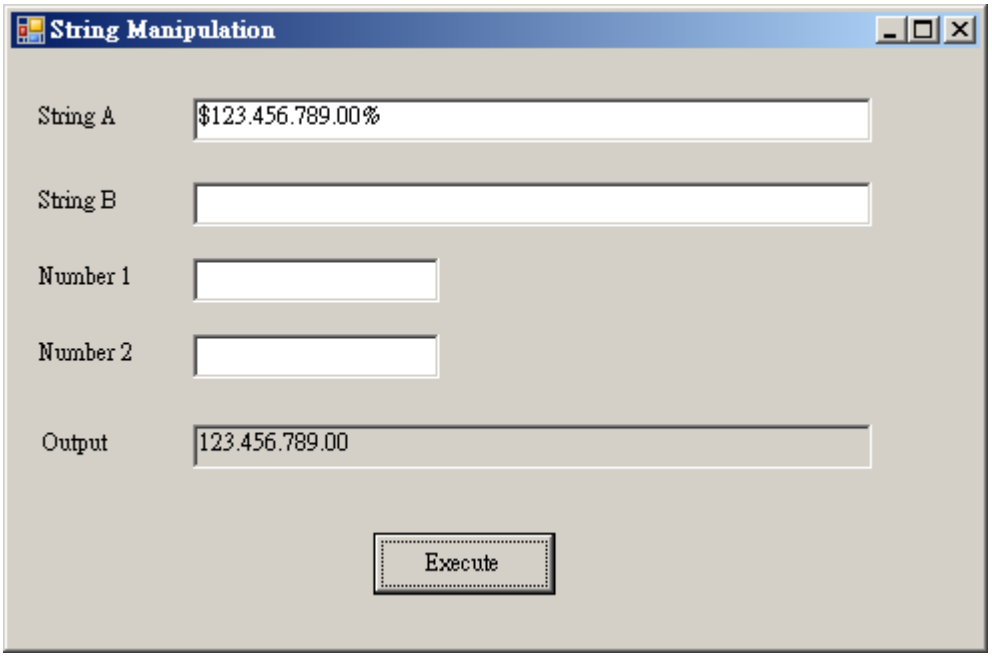
Object	Name	Property	Property Value
Form	frmMain	Text	String Manipulation
Label	Label1	Text	String 1
	Label2	Text	String 2
	Label3	Text	Number 1
	Label4	Text	Number 2
	Label5	Text	Output
Textbox	txtStrA	Text	*(Blank)*
	txtStrB	Text	*(Blank)*
	txtNum1	Text	*(Blank)*
	txtNum2	Text	*(Blank)*
	txtOutput	Text	*(Blank)*
Button	btnExecute	Text	Execute



2. In the **Click** event procedure of the Execute button (**btnExecute**), add the following code:

```
' Trim all leading and trailing dollar signs, spaces and percentage sign
txtOutput.Text = txtStrA.Text.Trim("$", " ", "%")
```

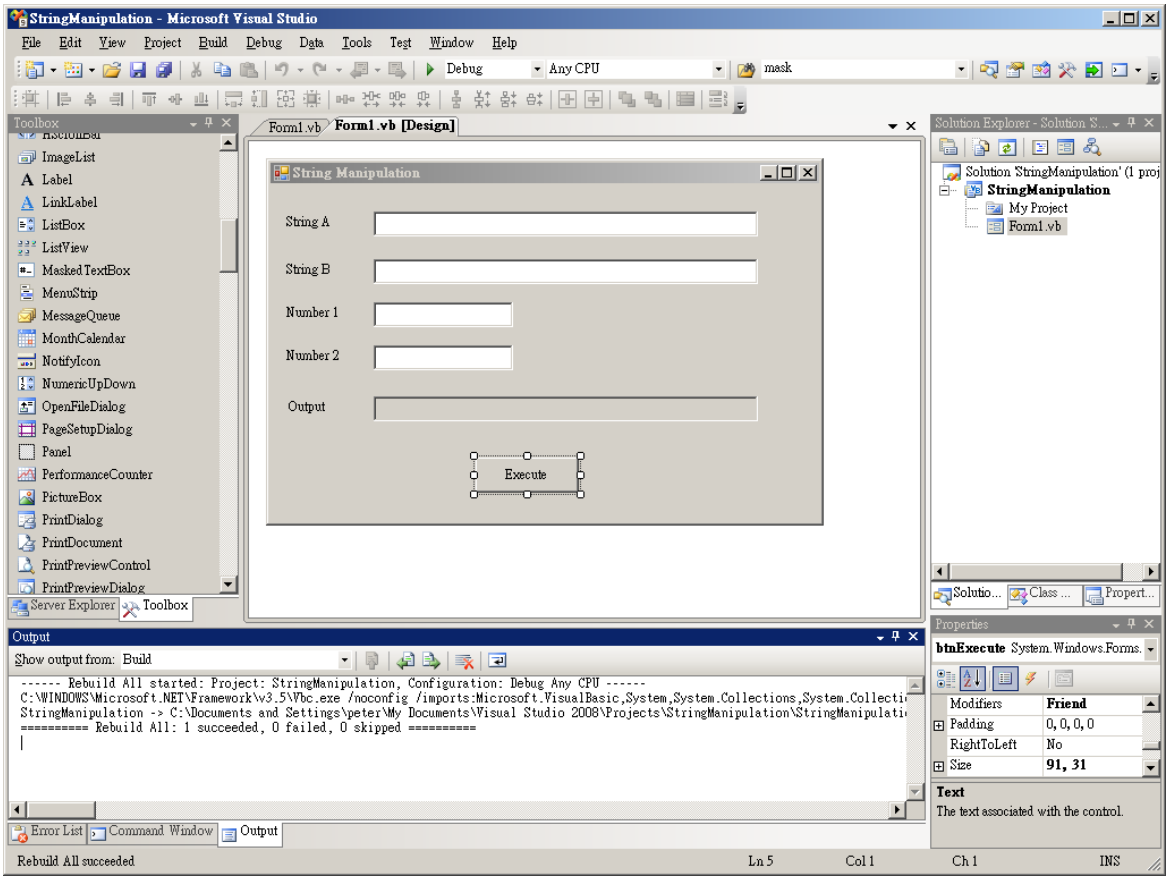
3. Save the project and build the solution, and then execute it. Input a string in “String A” textbox, and then press the **[Execute]** button, can you observe the different?



11. String Manipulation – Replacement

1. Start the Microsoft Visual Studio and open a new Visual Basic Project **Replacement**. From the Toolbox, drag 5 **Label** controls, 5 **Textbox** controls and a **Button** control onto the form and customize the properties.

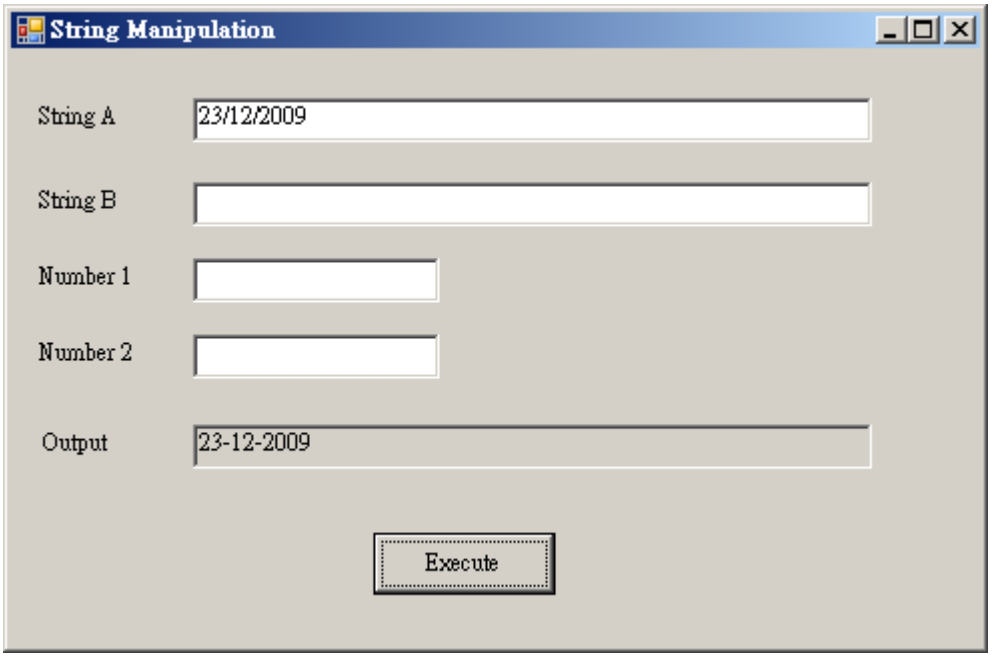
Object	Name	Property	Property Value
Form	frmMain	Text	String Manipulation
Label	Label1	Text	String 1
	Label2	Text	String 2
	Label3	Text	Number 1
	Label4	Text	Number 2
	Label5	Text	Output
Textbox	txtStrA	Text	*(Blank)*
	txtStrB	Text	*(Blank)*
	txtNum1	Text	*(Blank)*
	txtNum2	Text	*(Blank)*
	txtOutput	Text	*(Blank)*
Button	btnExecute	Text	Execute



2. In the **Click** event procedure of the Execute button (**btnExecute**), add the following code:

```
' Replace all "/" by "-"
txtOutput.Text = txtStrA.Text.Replace("/", "-")
```

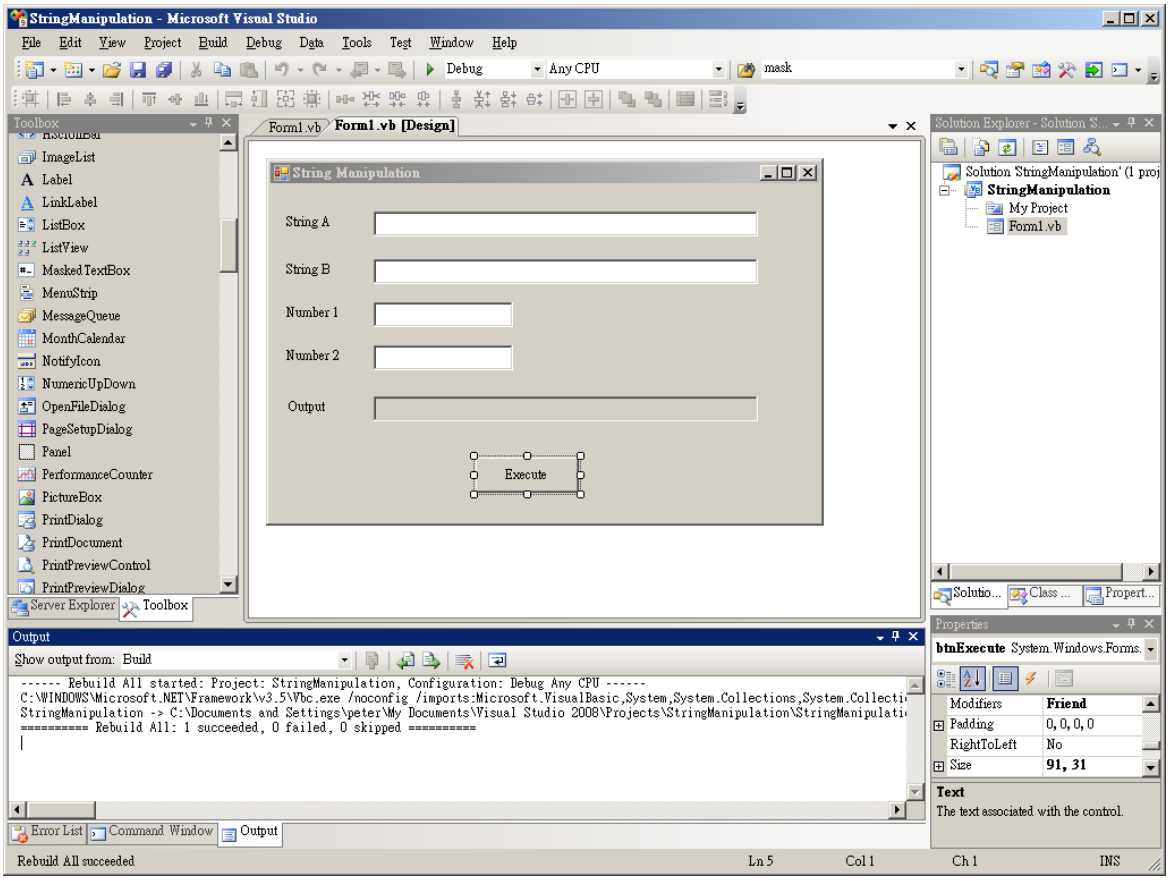
3. Save the project and build the solution, and then execute it. Input a string in “String A” textbox, and then press the **[Execute]** button, can you observe the result?



12. String Manipulation – Substring

1. Start the Microsoft Visual Studio and open a new Visual Basic Project **Substring**. From the Toolbox, drag 5 **Label** controls, 5 **Textbox** controls and a **Button** control onto the form and customize the properties.

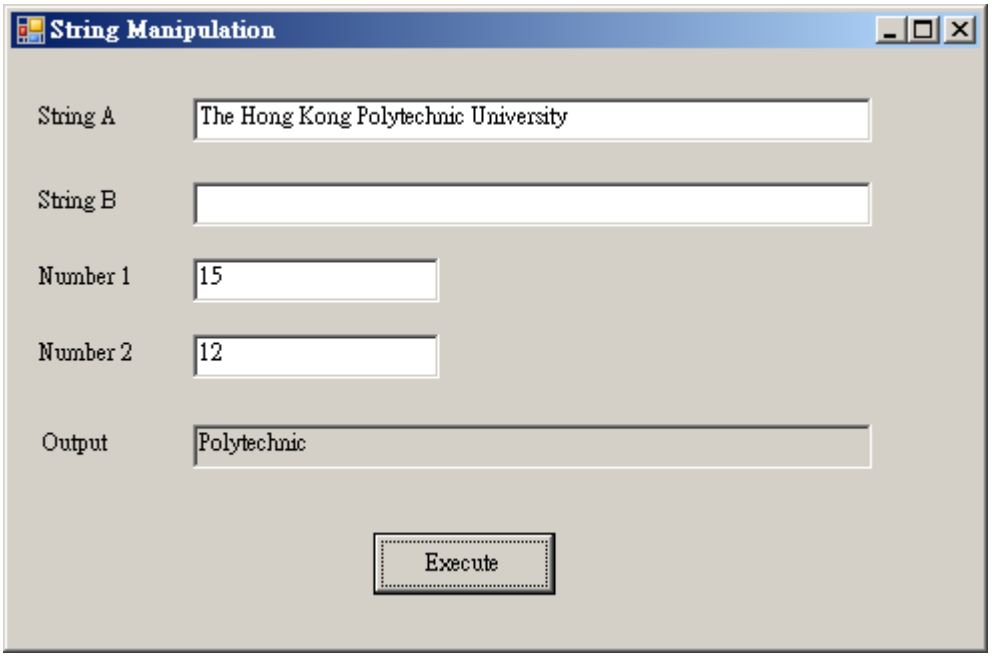
Object	Name	Property	Property Value
Form	frmMain	Text	String Manipulation
Label	Label1	Text	String 1
	Label2	Text	String 2
	Label3	Text	Number 1
	Label4	Text	Number 2
	Label5	Text	Output
Textbox	txtStrA	Text	*(Blank)*
	txtStrB	Text	*(Blank)*
	txtNum1	Text	*(Blank)*
	txtNum2	Text	*(Blank)*
	txtOutput	Text	*(Blank)*
Button	btnExecute	Text	Execute



2. In the **Click** event procedure of the Execute button (**btnExecute**), add the following code:

```
' Capture Specified Location of Character within a string
txtOutput.Text = Mid(txtStrA.Text, txtNum1.Text, txtNum2.Text)
```

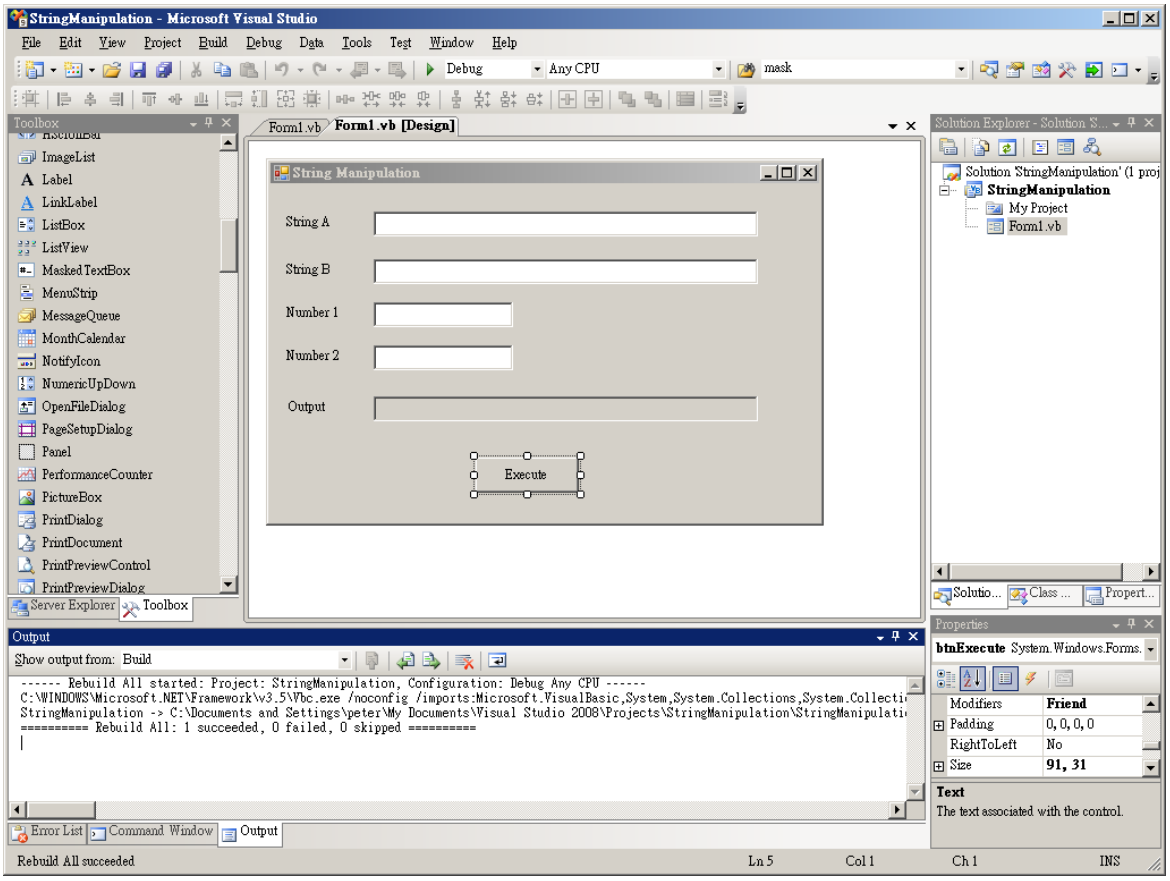
3. Save the project and build the solution, and then execute it. Input a string in “String A” textbox, the start position in “Number 1” textbox, and the number of characters in “Number 2” textbox, and then press the **[Execute]** button, can you observe the result?



13. String Manipulation – Searching

1. Start the Microsoft Visual Studio and open a new Visual Basic Project **Searching**. From the Toolbox, drag 5 **Label** controls, 5 **Textbox** controls and a **Button** control onto the form and customize the properties.

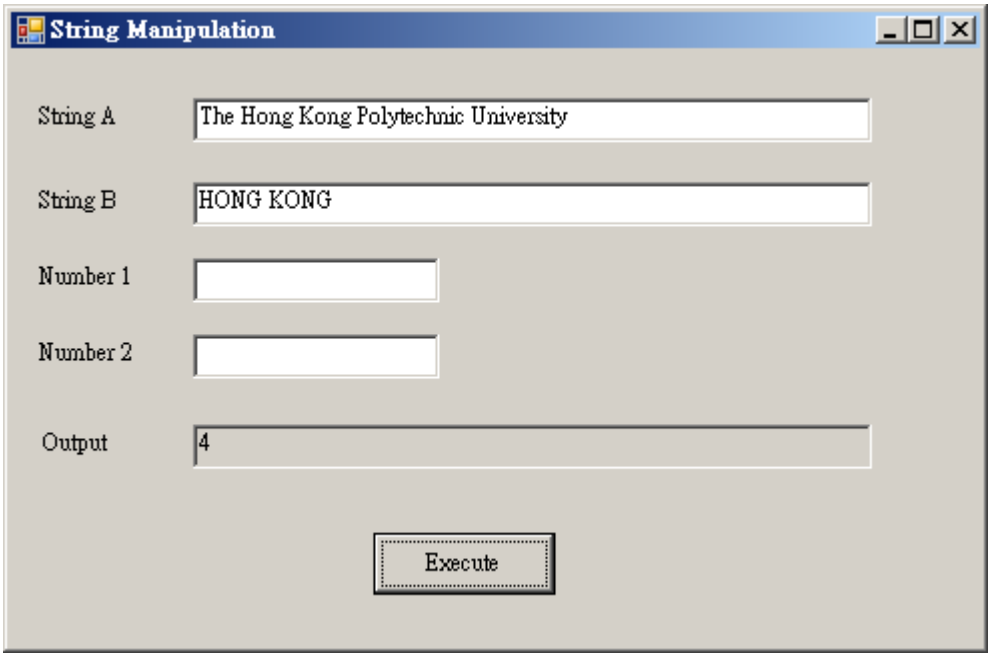
Object	Name	Property	Property Value
Form	frmMain	Text	String Manipulation
Label	Label1	Text	String 1
	Label2	Text	String 2
	Label3	Text	Number 1
	Label4	Text	Number 2
	Label5	Text	Output
Textbox	txtStrA	Text	*(Blank)*
	txtStrB	Text	*(Blank)*
	txtNum1	Text	*(Blank)*
	txtNum2	Text	*(Blank)*
	txtOutput	Text	*(Blank)*
Button	btnExecute	Text	Execute



2. In the **Click** event procedure of the Execute button (**btnExecute**), add the following code:

```
' Searching for a subsring
txtOutput.Text = txtStrA.Text.ToUpper.IndexOf(txtStrB.Text.ToUpper)
```

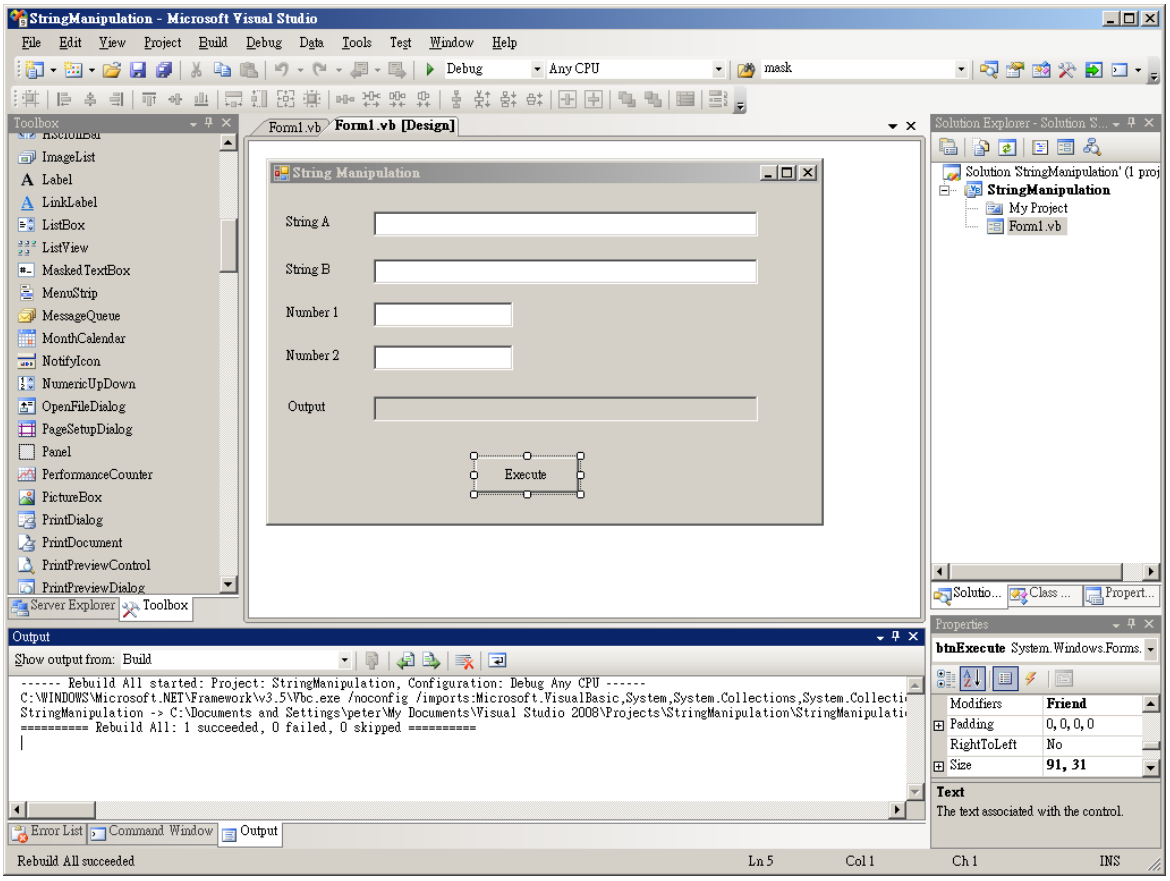
3. Save the project and build the solution, and then execute it. Input a string in "String A" textbox, and then press the **[Execute]** button, can you observe the result?



14. String Manipulation – Compare

1. Start the Microsoft Visual Studio and open a new Visual Basic Project **Searching**. From the Toolbox, drag 5 **Label** controls, 5 **Textbox** controls and a **Button** control onto the form and customize the properties.

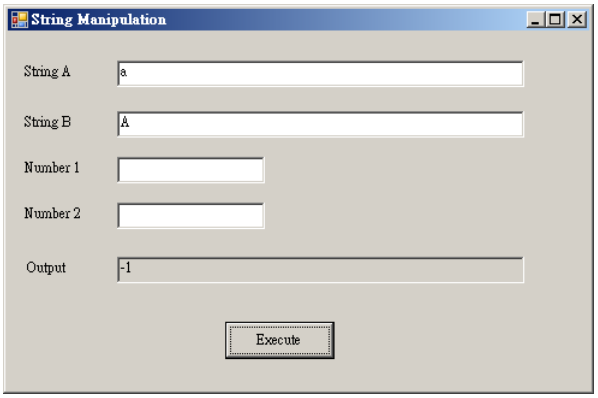
Object	Name	Property	Property Value
Form	frmMain	Text	String Manipulation
Label	Label1	Text	String 1
	Label2	Text	String 2
	Label3	Text	Number 1
	Label4	Text	Number 2
	Label5	Text	Output
Textbox	txtStrA	Text	*(Blank)*
	txtStrB	Text	*(Blank)*
	txtNum1	Text	*(Blank)*
	txtNum2	Text	*(Blank)*
	txtOutput	Text	*(Blank)*
Button	btnExecute	Text	Execute

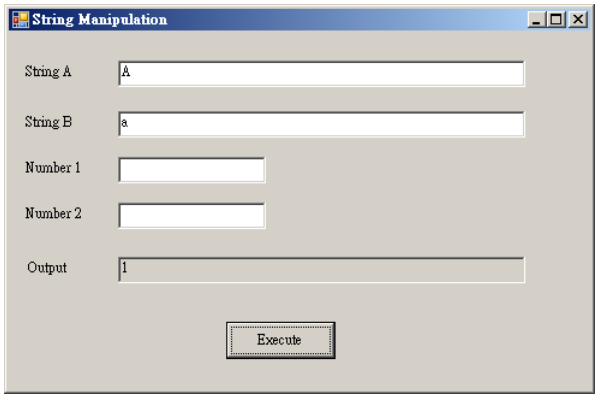


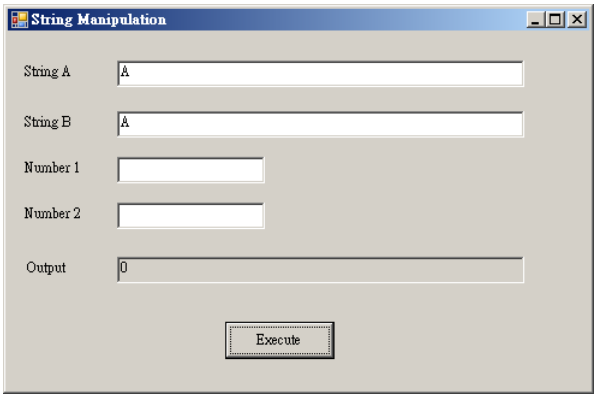
2. In the **Click** event procedure of the Execute button (**btnExecute**), add the following code:

```
' Compare String
txtOutput.Text = String.Compare(txtStrA.Text, txtStrB.Text)
```

3. Save the project and build the solution, and then execute it. Input a string in “String A” textbox, and then press the **[Execute]** button, can you observe the result?



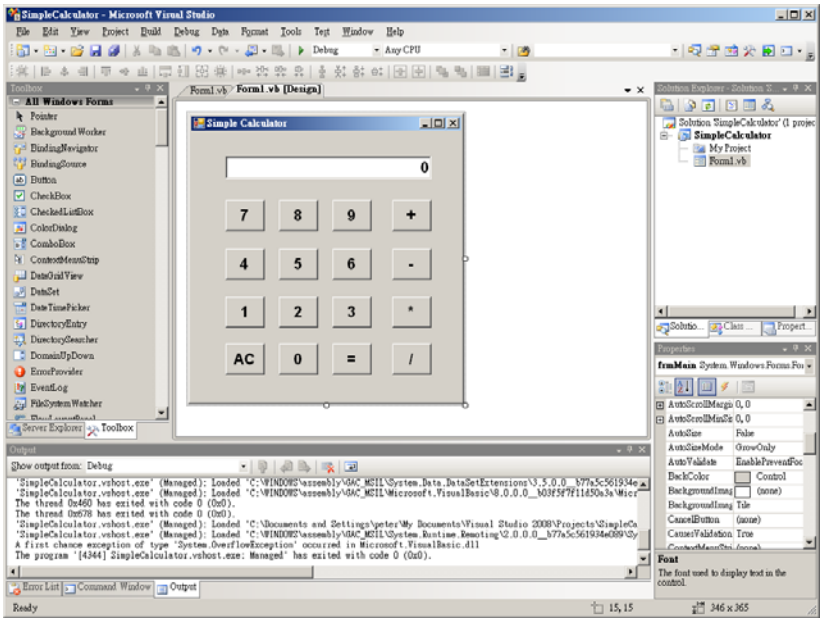




15. Simple Calculator

1. Open the Microsoft Visual Studio and start a new Visual Basic Project named as **SimpleCalculator**. From the Toolbox, drag a **Textbox** control and 16 **Button** controls onto the form and customize the properties.

Object	Name	Property	Property Value
Form	frmMain	Text	Simple Calculator
Text Box	txtOutput	Text	0
Button	btn0	Text	0
	btn1	Text	1
	btn2	Text	2
	btn3	Text	3
	btn4	Text	4
	btn5	Text	5
	btn6	Text	6
	btn7	Text	7
	btn8	Text	8
	btn9	Text	9
	btnAdd	Text	+
	btnSubtract	Text	-
	btnMultiply	Text	*
	btnDivide	Text	/
	btnEqual	Text	=
	btnAC	Text	AC



2. Declare the module level variables within the **frmMain** class.

```
Public Class frmMain
    Private Total As Integer
    Private AppendStatus As Boolean
    Private op As Char
```

3. Create a new procedure **OutputValue** within the class for displaying the output value

```
Private Sub OutputValue(ByVal InputDigit As Char)
    If AppendStatus = True Then
        txtOutput.Text += InputDigit
    Else
        txtOutput.Text = InputDigit
    End If
    AppendStatus = True
End Sub
```

4. Create a new procedure **CalculateResult** within the class for result calculation

```
Private Sub CalculateResult(ByVal ArithmeticOperator As Char)
    ' Calaulate the result
    Select Case op
        Case "+"
            txtOutput.Text = Total + txtOutput.Text
        Case "-"
            txtOutput.Text = Total - txtOutput.Text
        Case "*"
            txtOutput.Text = Total * txtOutput.Text
        Case "/"
        txtOutput.Text = Total / txtOutput.Text
    End Select

    ' Display the value on screen
    Total = txtOutput.Text
    AppendStatus = False
    op = ArithmeticOperator
End Sub
```

5. In the **Click** event procedure of the **btn0** control, add the following code. This code used to capture the input when user clicking the [0] button.

```
Call OutputValue("0")
```

6. In the **Click** event procedure of the **btn1** control, add the following code. This code used to capture the input when user clicking the [1] button.

```
Call OutputValue("1")
```

7. In the **Click** event procedure of the **btn2** control, add the following code. This code used to capture the input when user clicking the [2] button.

```
Call OutputValue("2")
```

8. In the **Click** event procedure of the **btn3** control, add the following code. This code used to capture the input when user clicking the [3] button.

```
Call OutputValue("3")
```

9. In the **Click** event procedure of the **btn4** control, add the following code. This code used to capture the input when user clicking the [4] button.

```
Call OutputValue("4")
```

10. In the **Click** event procedure of the **btn5** control, add the following code. This code used to capture the input when user clicking the [5] button.

```
Call OutputValue("5")
```

11. In the **Click** event procedure of the **btn6** control, add the following code. This code used to capture the input when user clicking the [6] button.

```
Call OutputValue("6)
```

12. In the **Click** event procedure of the **btn7** control, add the following code. This code used to capture the input when user click the [7] button.

```
Call OutputValue("7")
```

13. In the **Click** event procedure of the **btn8** control, add the following code. This code used to capture the input when user clicking the [8] button.

```
Call OutputValue("8")
```

14. In the **Click** event procedure of the **btn9** control, add the following code. This code used to capture the input when user clicking the [9] button.

```
Call OutputValue("9")
```

15. In the **Click** event procedure of the **btnAdd** control, add the following code. This code used to perform calculation when user clicking the [+] button.

```
Call CalculateResult("+")
```

16. In the **Click** event procedure of the **btnSubtract** control, add the following code. This code used to perform calculation when user clicking the [-] button.

```
Call CalculateResult("-")
```

17. In the **Click** event procedure of the **btnMultiply** control, add the following code. This code used to perform calculation when user clicking the [*] button.

```
Call CalculateResult("*")
```

18. In the **Click** event procedure of the **btnDivide** control, add the following code. This code used to perform calculation when user clicking the [/] button.

```
Call CalculateResult("/")
```

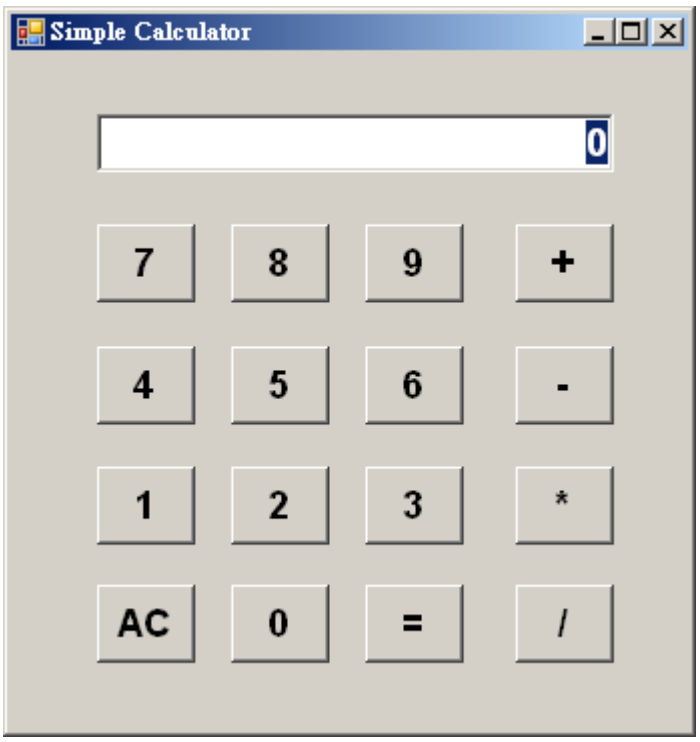
19. In the **Click** event procedure of the **btnEqual** control, add the following code. This code used to perform calculation when user clicking the [=] button.

```
Call CalculateResult("")
```

20. In the **Click** event procedure of the **btnAC** control, add the following code. This code used to clear the memory buffer when user clicking the [AC] button.

```
AppendStatus = False
txtOutput.Text = "0"
Total = 0
```

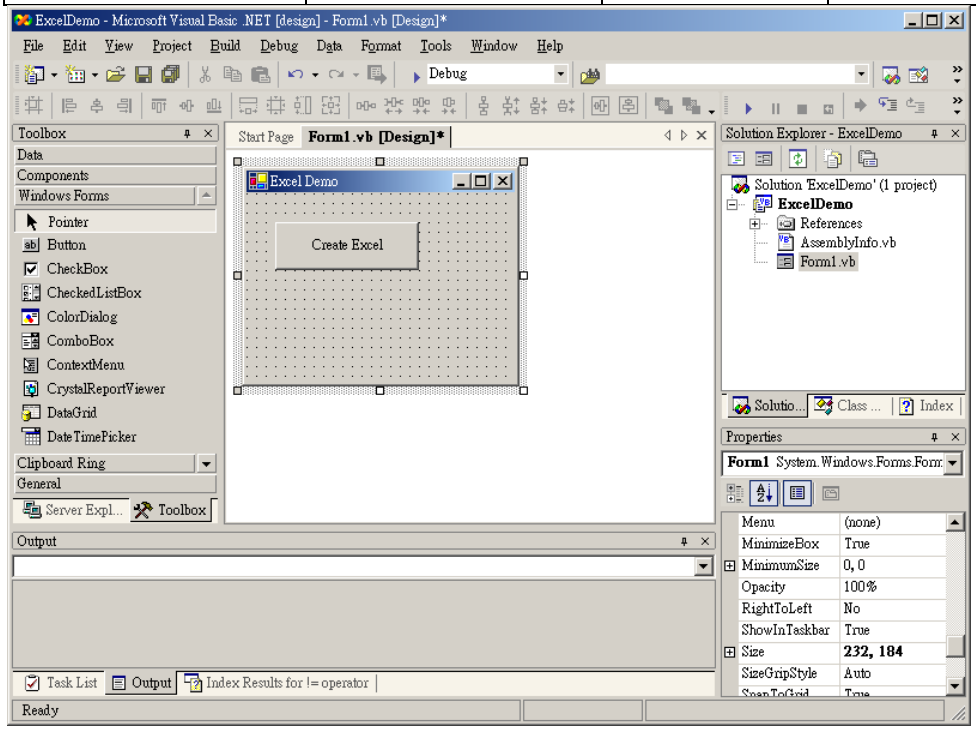
21. Build and execute your program. Now you can use your simple calculator for calculation



16. Export Data to Microsoft Excel

1. Open the Microsoft Visual Studio and start a new Visual Basic Project named as **ExportExcel**. From the Toolbox, drag a **Button** controls onto the form and customize the properties.

Object	Name	Property	Property Value
Form	frmMain	Text	Excel Export Demo
Button	btnExcel	Text	Create Excel

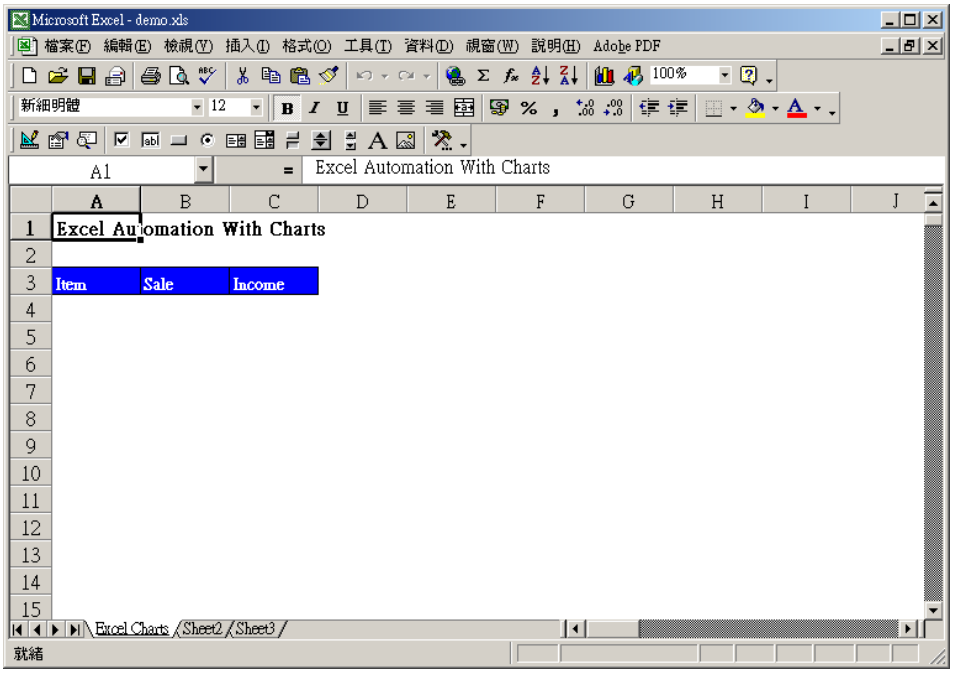


2. In the **Click** event procedure of the Create Excel button control (**btnExcel**), add the following code.

```
'File name and path
Dim Filename = AppDomain.CurrentDomain.BaseDirectory & "demo.xls"

' Create new excel application
Dim objExcel = CreateObject("Excel.Application")

' Add a new workbook
Dim objBook = objExcel.Workbooks.Add

' Set the application alerts not to be displayed for confirmation
objExcel.Application.DisplayAlerts = True

' This procedure populate the sheet
objExcel.Visible = True

' Add a new sheet
Dim objSheet = objExcel.Worksheets(1)
```

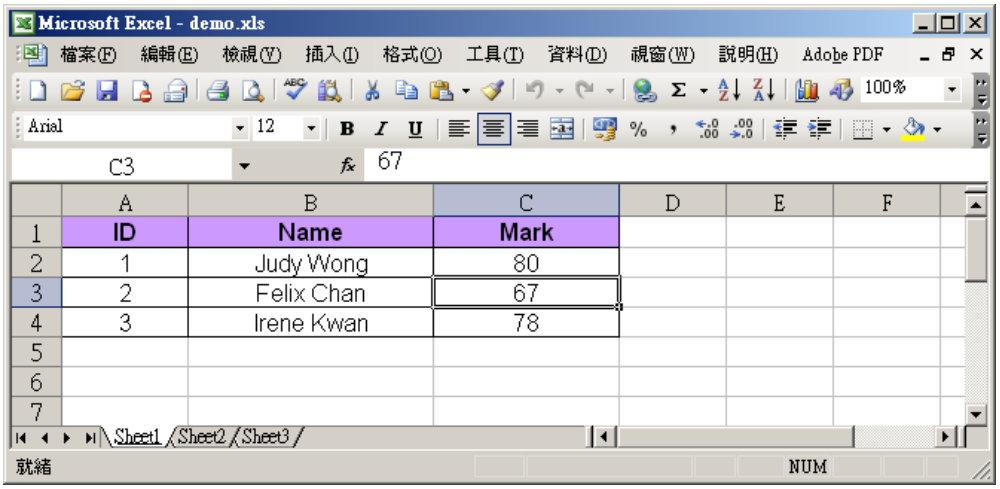
```
' Rename the sheet
objSheet.Name = "Excel Charts"

' Format Title
objSheet.Range("A1:AZ400").Interior.ColorIndex = 2
objSheet.Range("A1").Font.Size = 12
objSheet.Range("A1").Font.Bold = True
objSheet.Range("A1:I1").Merge()
objSheet.Range("A1").Value = "Excel Automation With Charts"
objSheet.Range("A1").EntireColumn.AutoFit()

' Format headings
objSheet.Range("A3:C3").Font.Color = RGB(255, 255, 255)
objSheet.Range("A3:C3").Interior.ColorIndex = 5
objSheet.Range("A3:C3").Font.Bold = True
objSheet.Range("A3:C3").Font.Size = 10

' Columns heading
objSheet.Range("A3").Value = "Item"
objSheet.Range("A3").BorderAround(8)
objSheet.Range("B3").Value = "Sale"
objSheet.Range("B3").BorderAround(8)
objSheet.Range("C3").Value = "Income"
objSheet.Range("C3").BorderAround(8)

' Save the Excel
objBook.SaveAs(Filename)
```

3. Save and build the project, you can press the **[Create Excel]** button to create and open a new Microsoft Excel file.



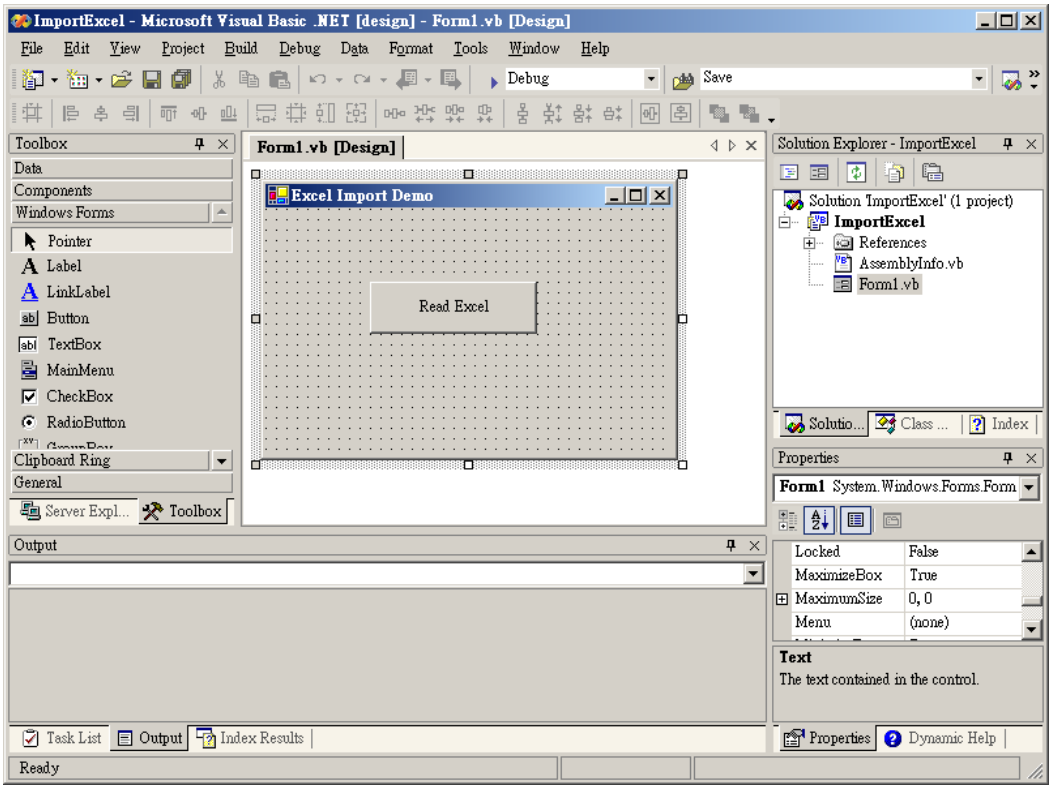
17. Read Data to Microsoft Excel

1. Create an Excel file with the following content and save to “C:\Temp\demo.xls”.



2. Open the Microsoft Visual Studio and start a new Visual Basic Project named as **ImportExcel**. From the Toolbox, drag a **Button** controls onto the form and customize the properties.

Object	Name	Property	Property Value
Form	frmMain	Text	Excel Import Demo
Button	btnExcel	Text	Read Excel



3. In the **Click** event procedure of the Read Excel button control (**btnExcel**), add the following code.

```
' Declare a string
Dim var As String

' Define the File name and path
Dim Filename = "C:\Temp\demo.xls"

' Open the Excel
Dim objExcel = GetObject(Filename)

' Get the value from Excel
var = objExcel.Worksheets(1).Cells(2, 2).Value

' Display the message
MsgBox(var)

' Close the Excel file
objExcel.close()
```

4. Save and build the project, you can press the **[Read Excel]** button to read the Excel content.