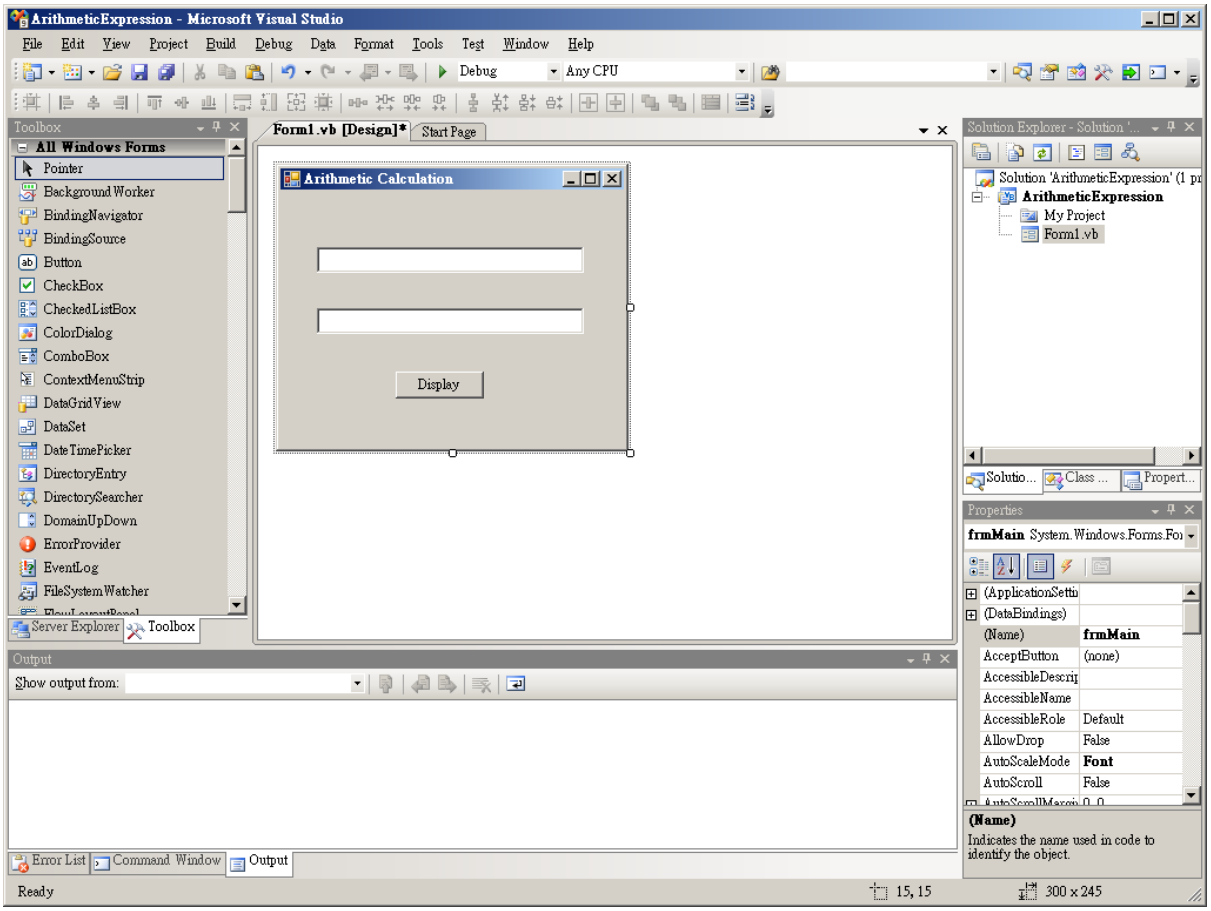
1. Arithmetic Calculation

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

Object	Name	Property	Property Value
Form	frmMain	Text	Arithmetic Calculation
Text Box	txtInputA	Text	*(Blank)*
	txtInputB	Text	*(Blank)*
Button	btnDisplay	Text	Display

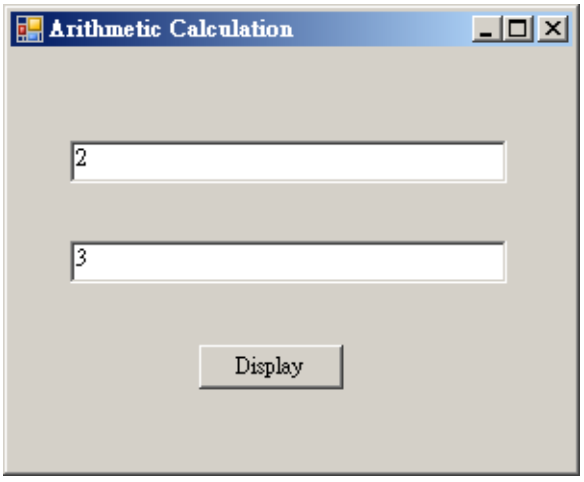


2. In the **Click** event procedure of the **btnDisplay** control, add the following code. The first two lines declare the variables A and B, which will hold the numeric values used in this program, and assign the values of the two TextBox controls to the variables A and B. The final four lines create expressions with the two variables and each of the basic arithmetic operators, and display the results of those expressions in a message box

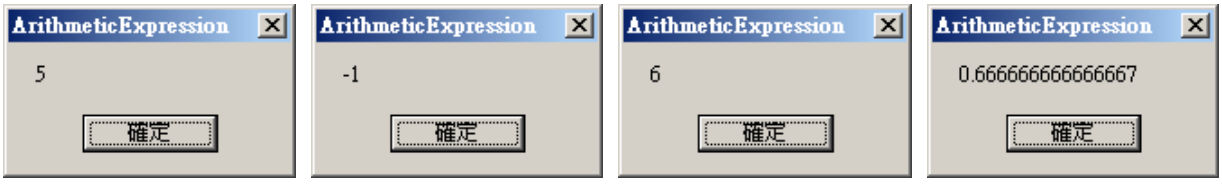
```
' Declare the variables
Dim A As Double = txtInputA.Text
Dim B As Double = txtInputB.Text

' Calculate and display the results of in a message box.
MsgBox(A + B)
MsgBox(A - B)
MsgBox(A * B)
MsgBox(A / B)
```

3. Build and run your program. Type a number in each of the text boxes and click display button.



4. Expressions are created using the two numbers that you enter and each of the four basic arithmetic operators (addition, subtraction, multiplication, and division). The result of each expression is displayed in a message box. (*Please note that an error will occur if you type any other character into the text boxes*)

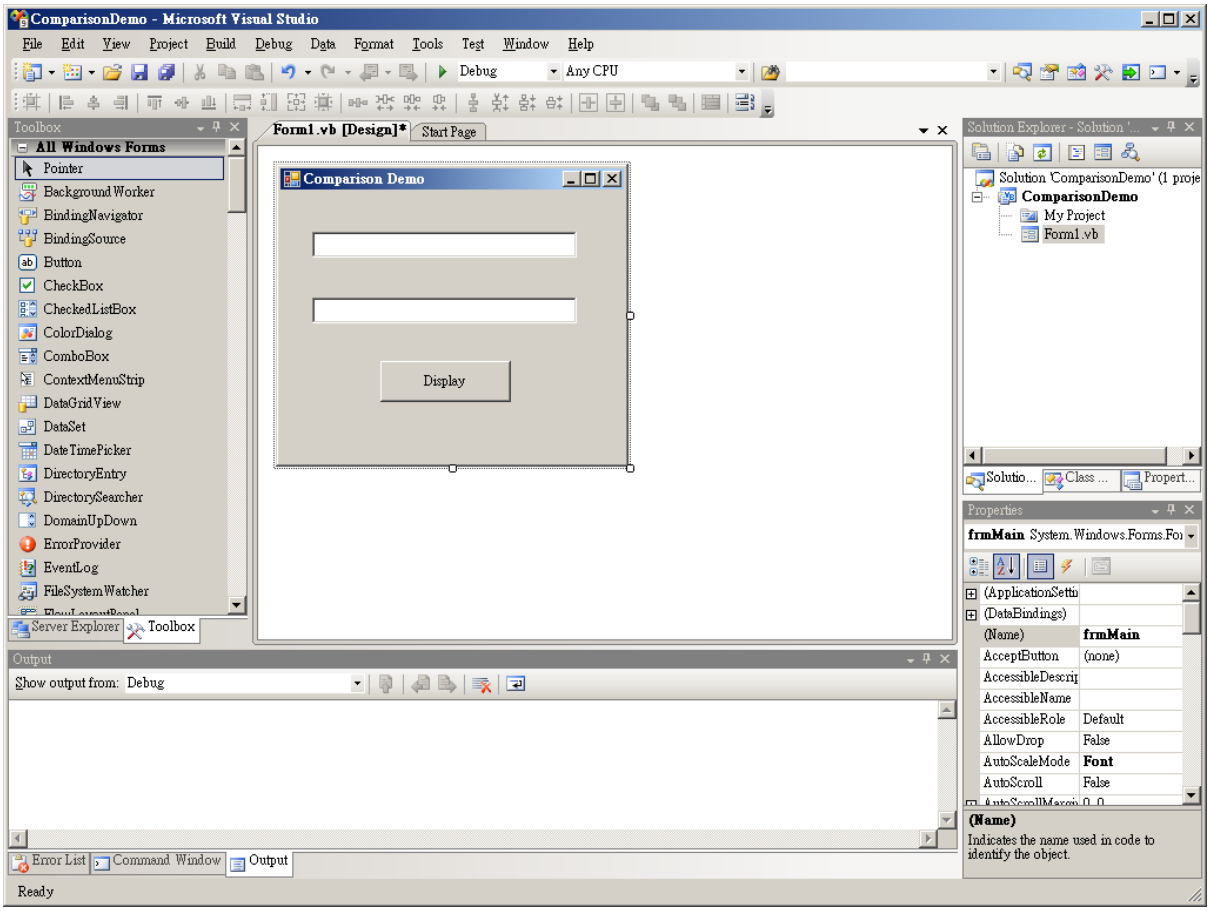


Copyright © 2010 Lo Chi Wing

2. Using Expressions to Compare Values

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

Object	Name	Property	Property Value
Form	frmMain	Text	Comparison Demo
Text Box	txtInputA	Text	*(Blank)*
	txtInputB	Text	*(Blank)*
Button	btnDisplay	Text	Display

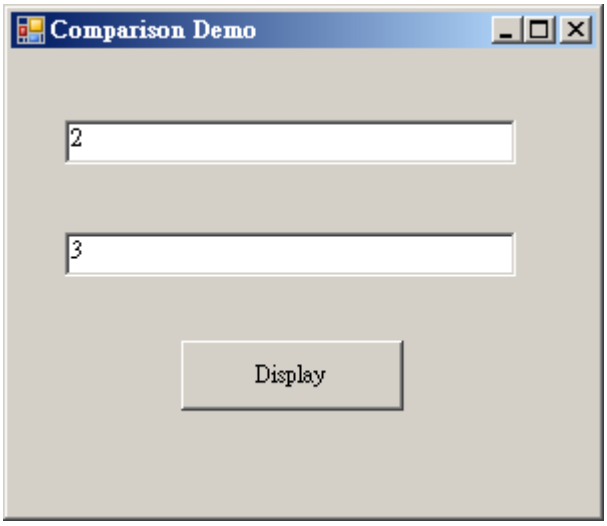


2. In the **Click** event procedure of the **btnDisplay** control, add the following code. The first two lines declare the variables A and B, which will hold the numeric values used in this program. The last three lines create expressions to compare the two variables using three basic comparison operators, and display the results of those expressions in three message boxes.

```
' Declare the variables
Dim A As Double = txtInputA.Text
Dim B As Double = txtInputB.Text

' Show the result
MsgBox(A > B)
MsgBox(A < B)
MsgBox(A = B)
```

3. Build and run your program. Type a number in each of the text boxes and click display button.



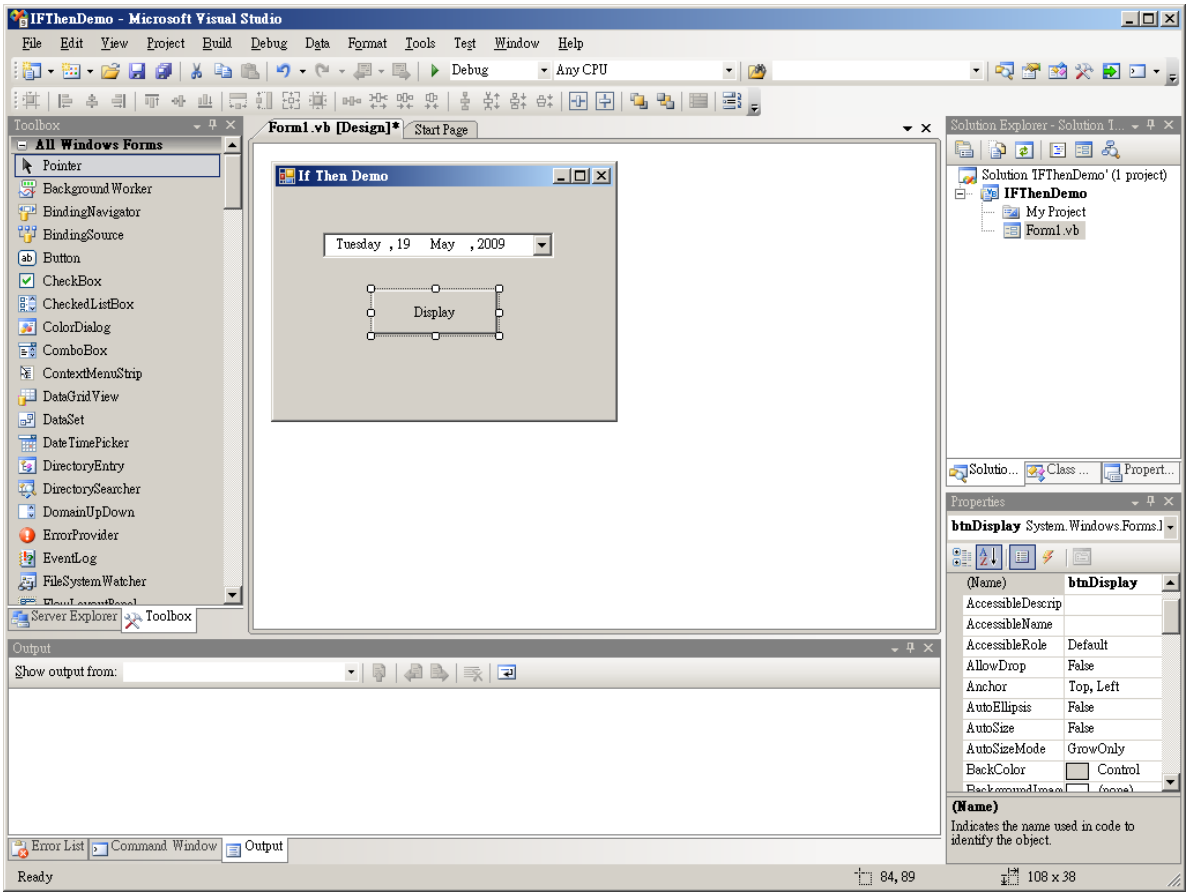
4. The first message box will display True if A (the number that you entered in the first text box) is greater than B (the number that you entered in the second text box); otherwise it will display False. The second message box will display True if A is less than B, and the third message box will display True if both numbers are the same. Try typing different numbers into the text boxes to see how the results change.

Copyright © 2010 Lo Chi Wing

3. Making Your Program Choose Between Two Possibilities

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

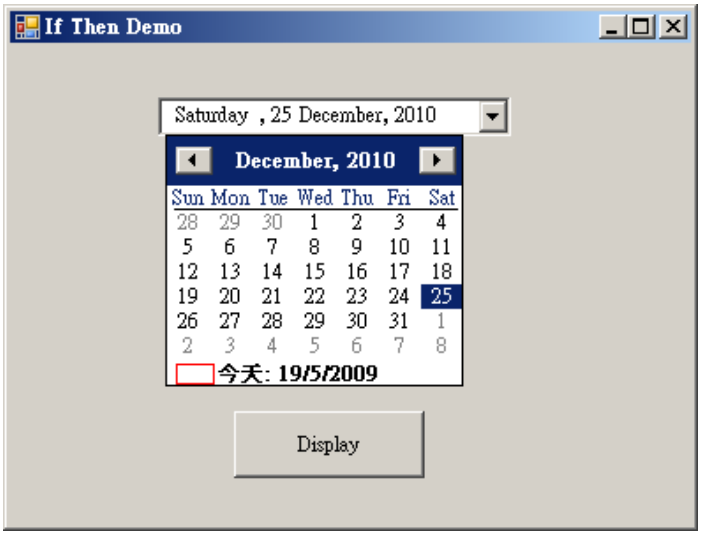
Object	Name	Property	Property Value
Form	frmMain	Text	If Then Demo
Button	btnDisplay	Text	Display
DateTimePicker	DateTimePicker1		

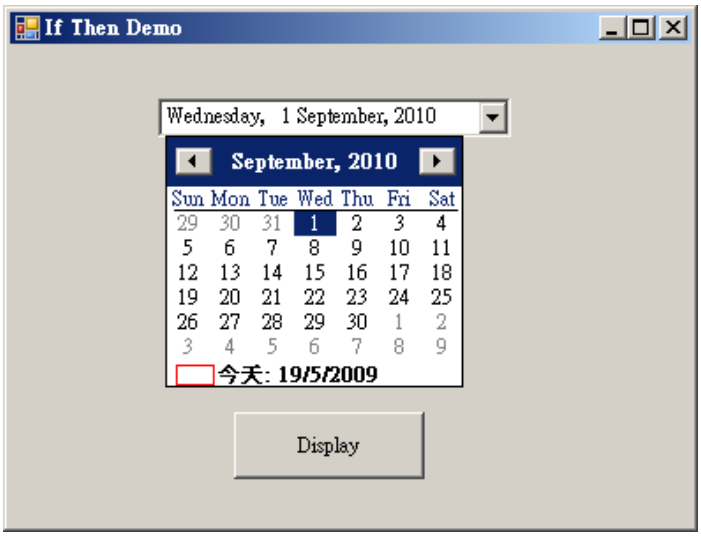


2. In the **Click** event procedure of the **btnDisplay** control, add the following code. In this example, the expression is evaluated; if it is true, then the next line of code is run, and the first message box is displayed. If it is false, then the code skips to the Else clause, and the line following Else is run, displaying the second message box.

```
If DateTimePicker1.Value.DayOfWeek = DayOfWeek.Saturday Or _
   DateTimePicker1.Value.DayOfWeek = DayOfWeek.Sunday Then
    MsgBox("Enjoy your Weekend!")
Else
    MsgBox("Happy Weekday! Work hard!")
End If
```

3. Build and execute your program. You might change the date in order to obtain different result.



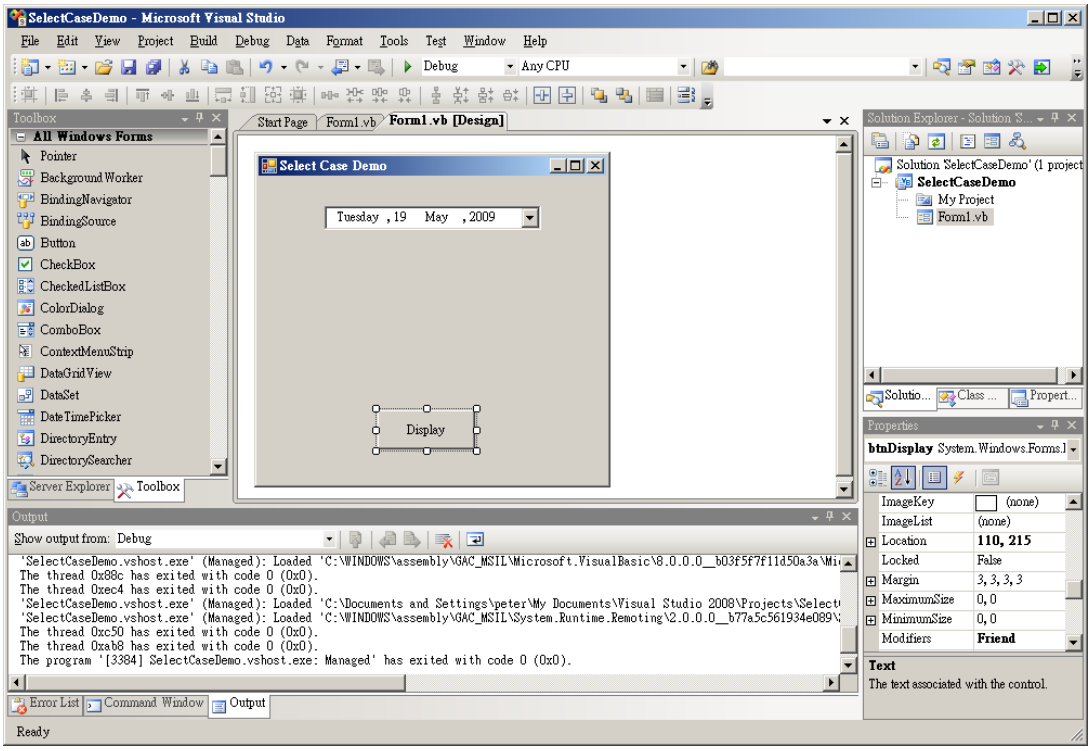


Copyright © 2010 Lo Chi Wing

4. Making Your Program Choose Between Several Possibilities

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

Object	Name	Property	Property Value
Form	frmMain	Text	Select Case Demo
Button	btnDisplay	Text	Display
DateTimePicker	DateTimePicker1		

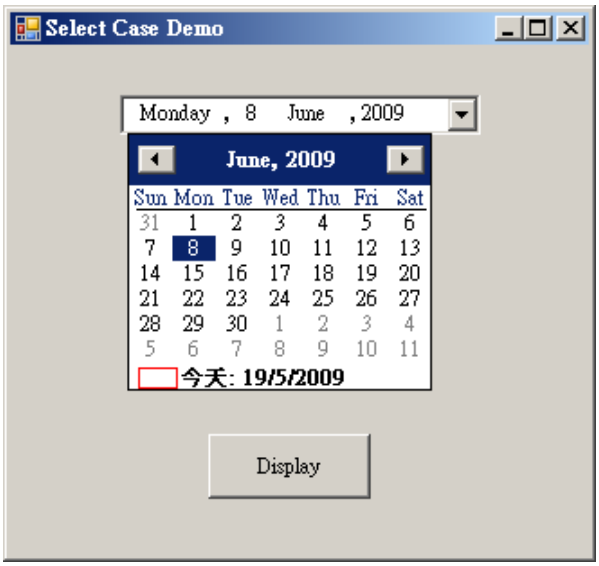


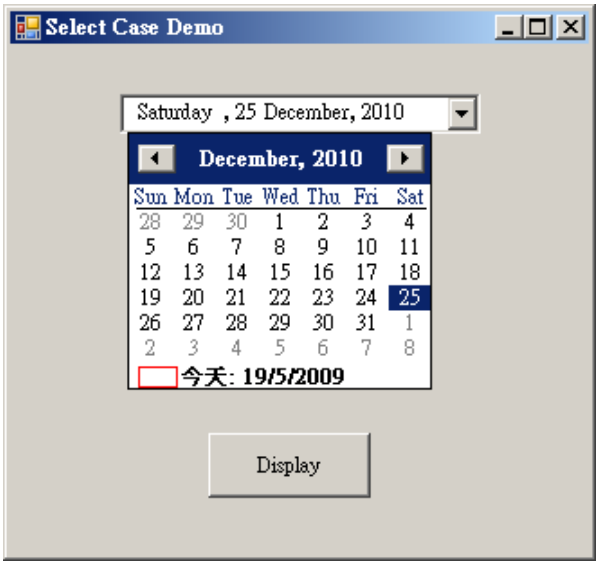
2. In the **Click** event procedure of the **btnDisplay** control, add the following code.

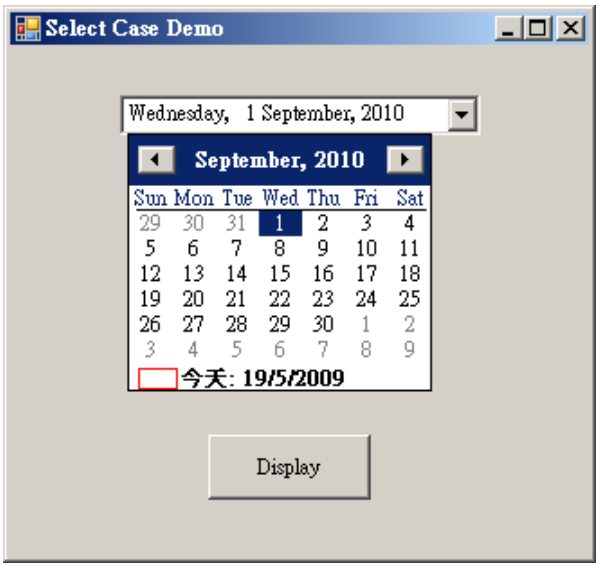
```
Select Case DateTimePicker1.Value.DayOfWeek
    Case DayOfWeek.Monday
        MsgBox("Today is Monday. Enjoy the VB.NET Lesson")
    Case DayOfWeek.Tuesday To DayOfWeek.Friday
        MsgBox("Work hard! Revision!")
    Case Else
        MsgBox("Happy Weekend!")
End Select
```

Copyright © 2010 Lo Chi Wing

3. Build and execute your program. You might change the date in order to obtain different result.





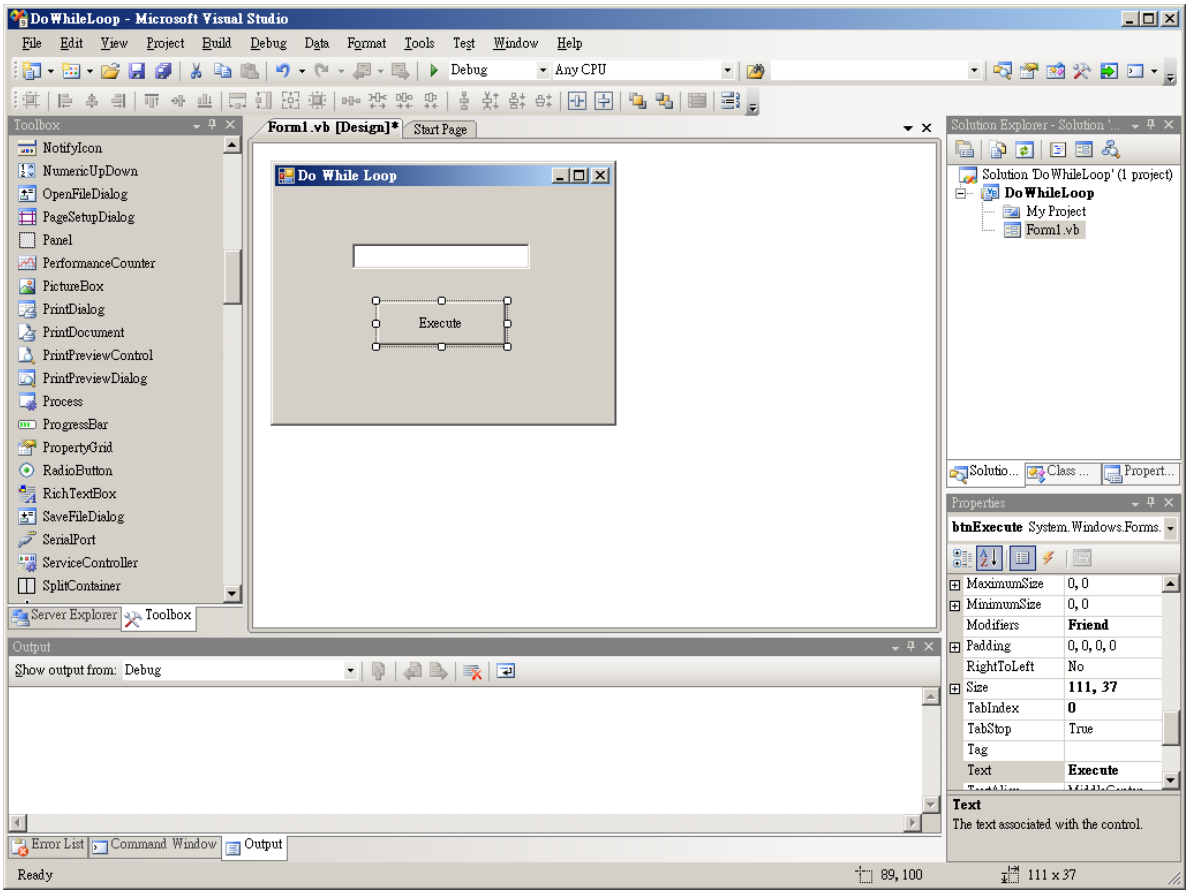


Copyright © 2010 Lo Chi Wing

5. Making Your Program Repeat Actions – Do While Loop

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

Object	Name	Property	Property Value
Form	frmMain	Text	Do While Loop
Text Box	txtInput	Text	*(Blank)*
Button	btnExecute	Text	Execute

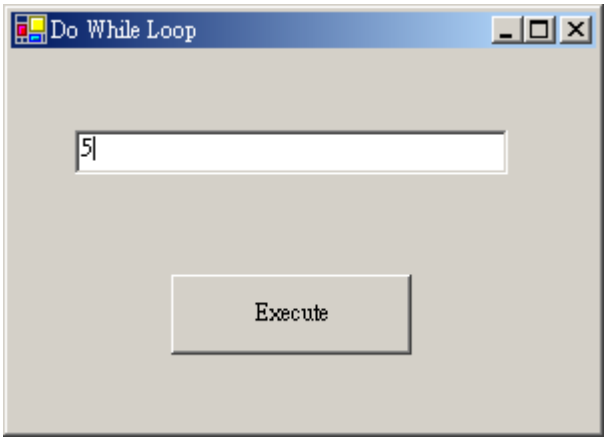


2. In the **Click** event procedure of the **btnExecute** control, add the following code.

```
Dim i As Integer = 0
Dim NumberOfRepetitions As Integer = txtInput.Text

Do While i <= NumberOfRepetitions
    i = i + 1
    MsgBox("This line has been repeated " & i & " times")
Loop
```

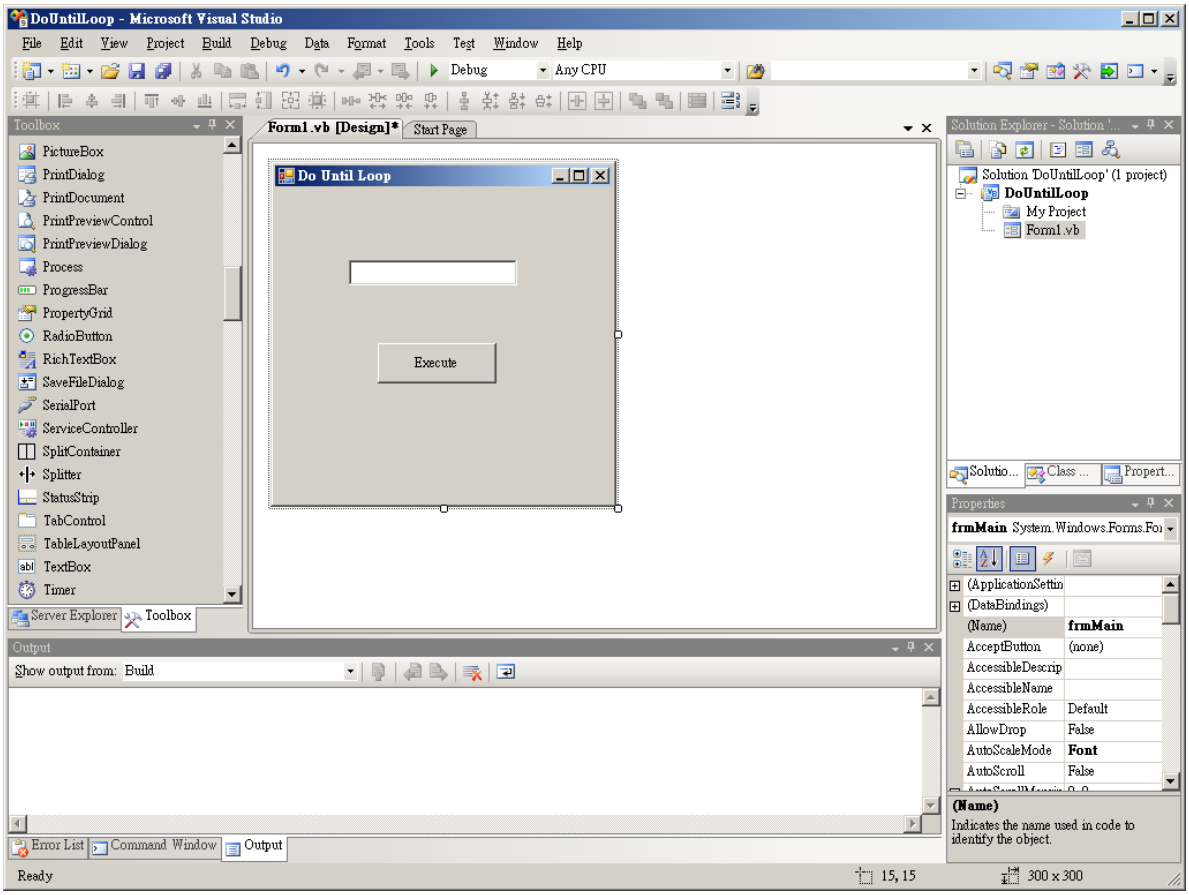
3. Build and execute your program. How many times the statement in the loop execute?



6. Making Your Program Repeat Actions – Do Until Loop

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

Object	Name	Property	Property Value
Form	frmMain	Text	Do Until Loop
Text Box	txtInput	Text	*(Blank)*
Button	btnExecute	Text	Execute



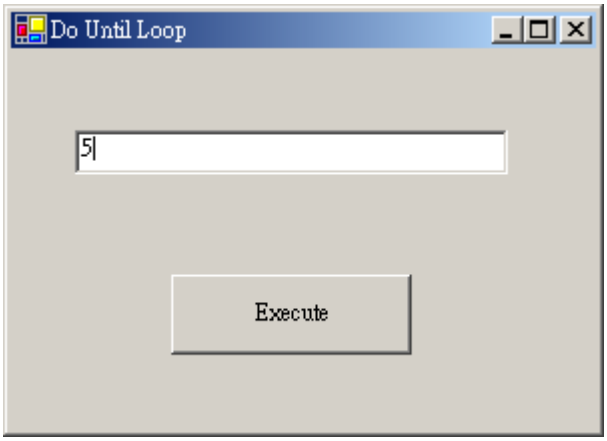
2. In the **Click** event procedure of the **btnExecute** control, add the following code.

```
Dim i As Integer = 0
Dim NumberOfRepetitions As Integer = txtInput.Text

Do Until i <= NumberOfRepetitions
    i = i + 1
    MsgBox("This line has been repeated " & i & " times")
Loop
```

Copyright © 2010 Lo Chi Wing

3. Build and run your program. How many times the statement in the loop execute?

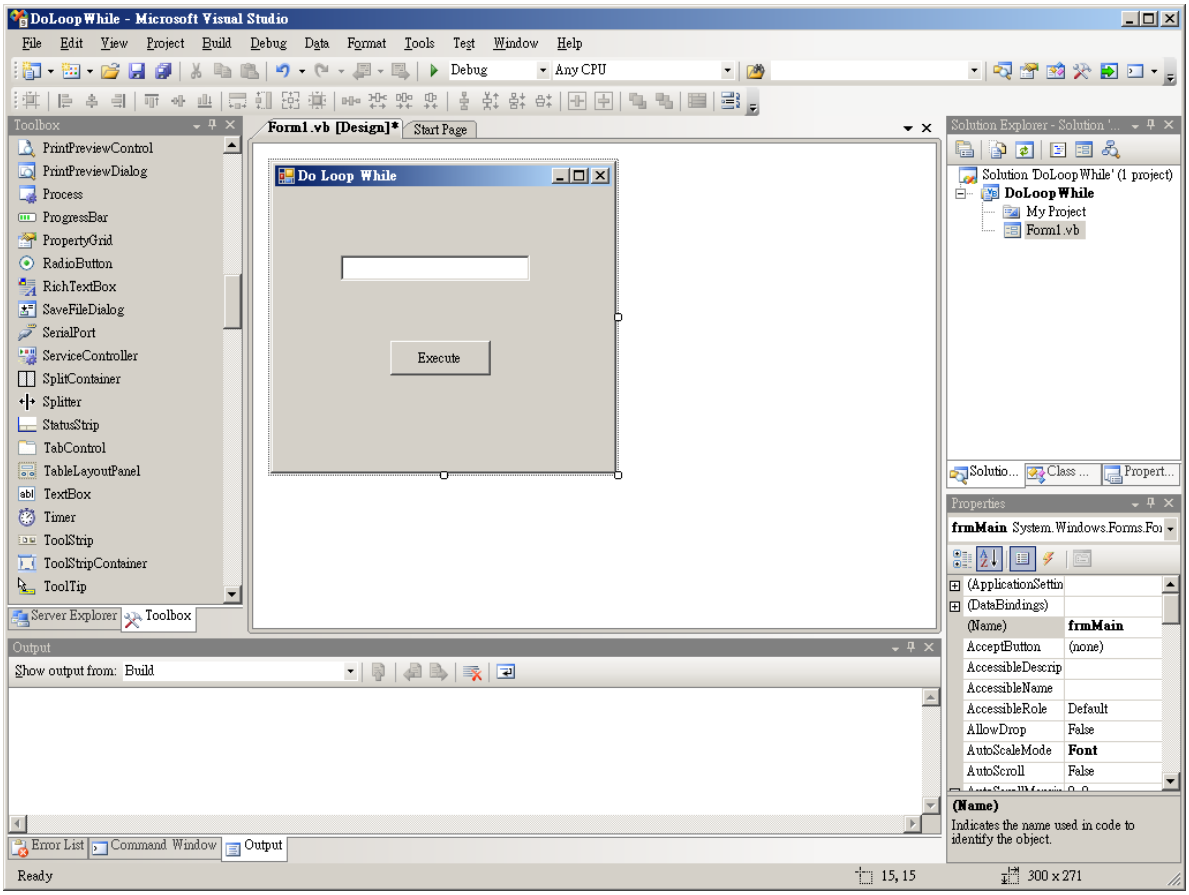


Copyright © 2010 Lo Chi Wing

7. Making Your Program Repeat Actions – Do Loop While

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

Object	Name	Property	Property Value
Form	frmMain	Text	Do Loop While
Text Box	txtInput	Text	*(Blank)*
Button	btnExecute	Text	Execute

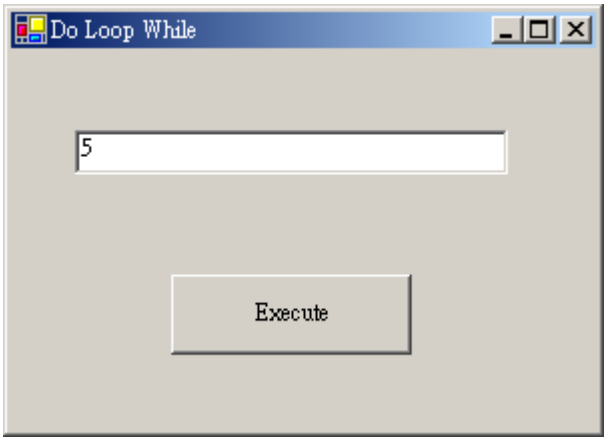


2. In the **Click** event procedure of the **btnExecute** control, add the following code.

```
Dim i As Integer = 0
Dim NumberOfRepetitions As Integer = txtInput.Text

Do
    i = i + 1
    MsgBox("This line has been repeated " & i & " times")
Loop While (i <= NumberOfRepetitions)
```

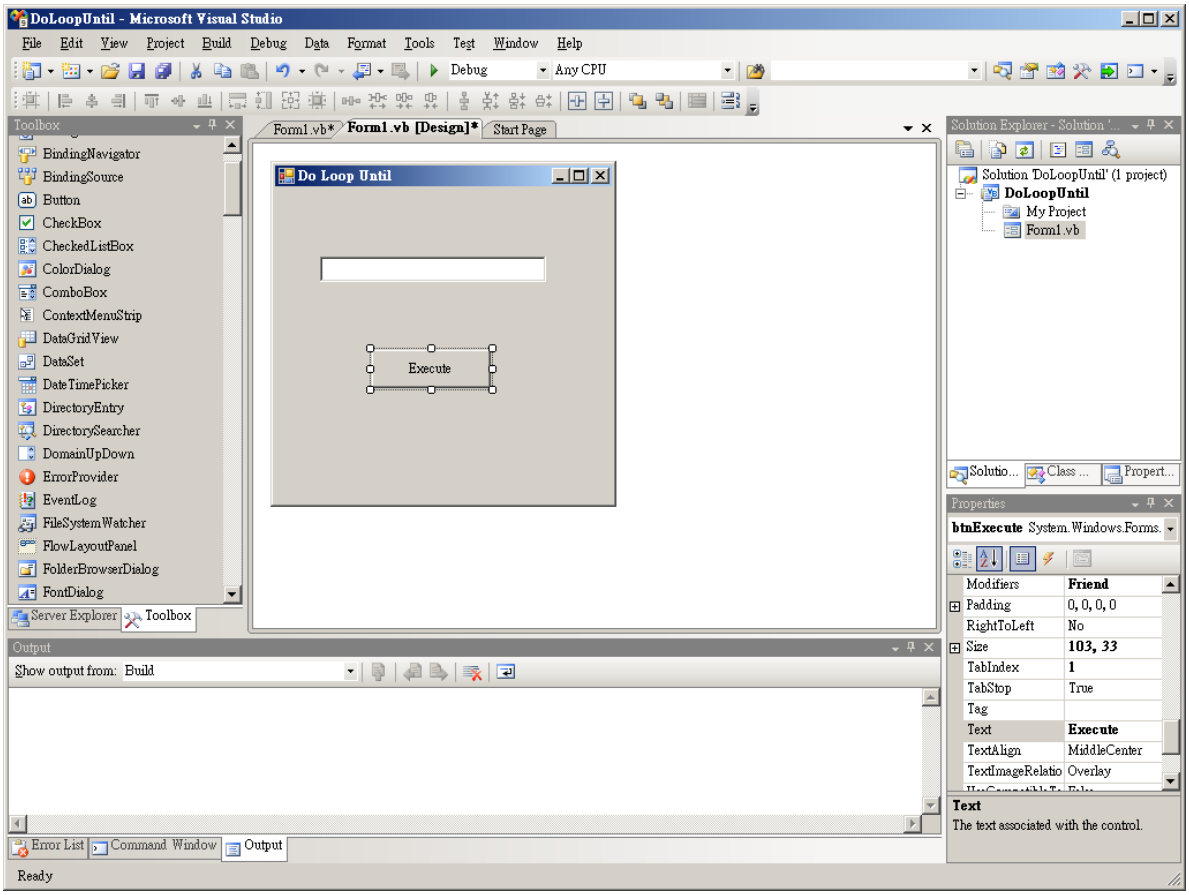
3. Build and run your program. How many times the statement in the loop execute?



8. Making Your Program Repeat Actions – Do Loop Until

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

Object	Name	Property	Property Value
Form	frmMain	Text	Do Loop Until
Text Box	txtInput	Text	*(Blank)*
Button	btnExecute	Text	Execute



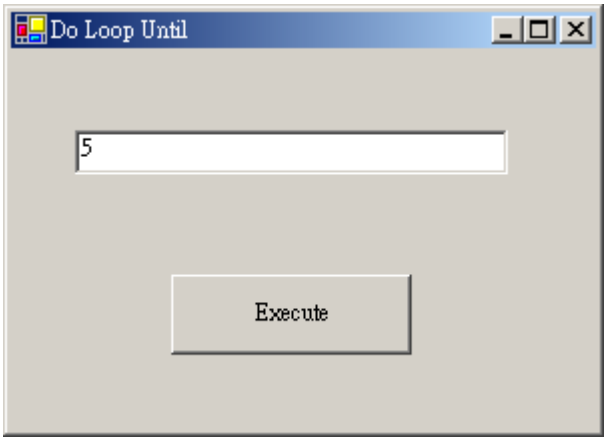
2. In the **Click** event procedure of the **btnExecute** control, add the following code.

```
Dim i As Integer = 0
Dim NumberOfRepetitions As Integer = txtInput.Text

Do
    i = i + 1
    MsgBox("This line has been repeated " & i & " times")
Loop Until (i <= NumberOfRepetitions)
```

Copyright © 2010 Lo Chi Wing

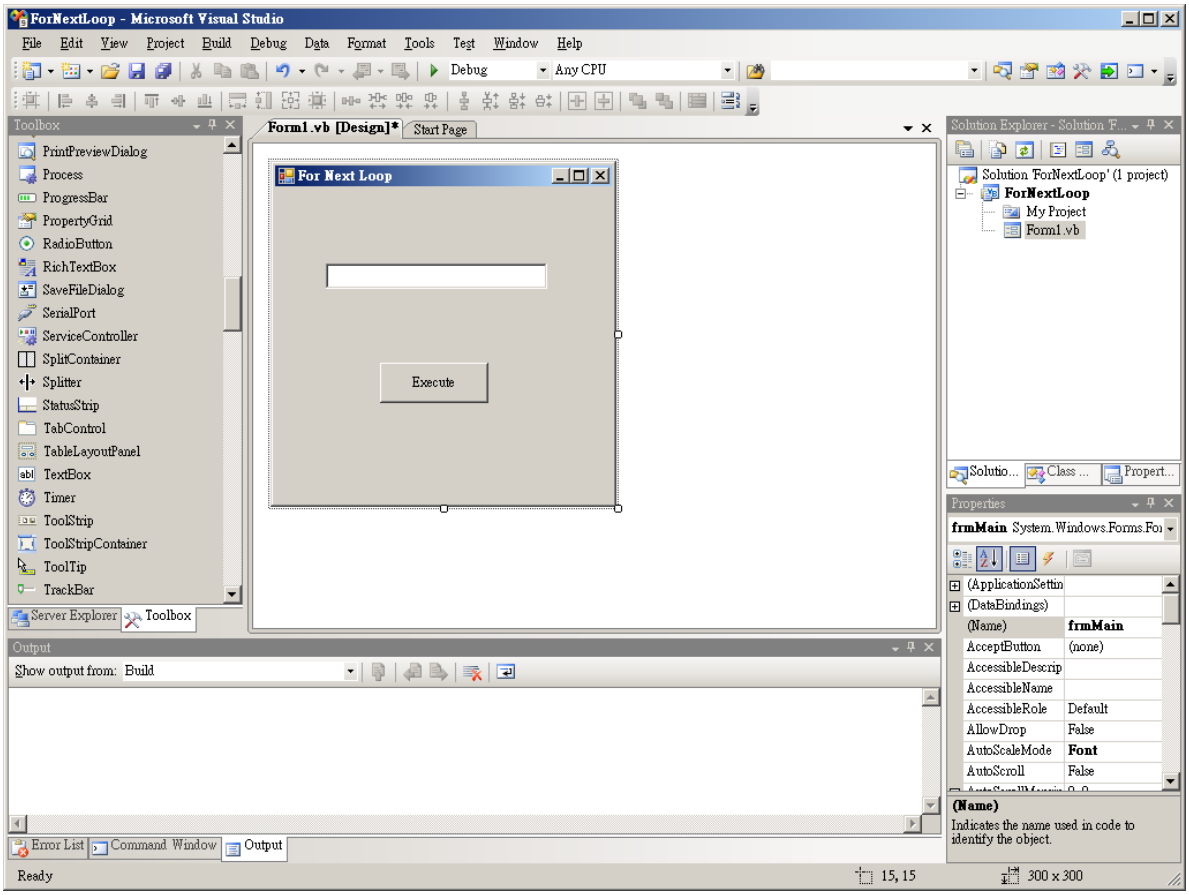
3. Build and run your program. How many times the statement in the loop execute?



9. Making Your Program Repeat Actions – For Next Loop

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

Object	Name	Property	Property Value
Form	frmMain	Text	For Next Loop
Text Box	txtInput	Text	*(Blank)*
Button	btnExecute	Text	Execute



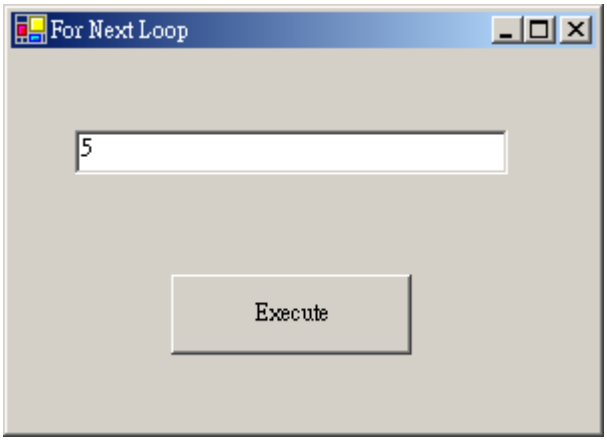
2. In the **Click** event procedure of the **btnExecute** control, add the following code.

```
Dim i As Integer = 0
Dim NumberOfRepetitions As Integer = txtInput.Text

For i = 1 To NumberOfRepetitions
    MsgBox("This line has been repeated " & i & " times")
Next
```

Copyright © 2010 Lo Chi Wing

3. Build and run your program. How many times the statement in the loop execute?

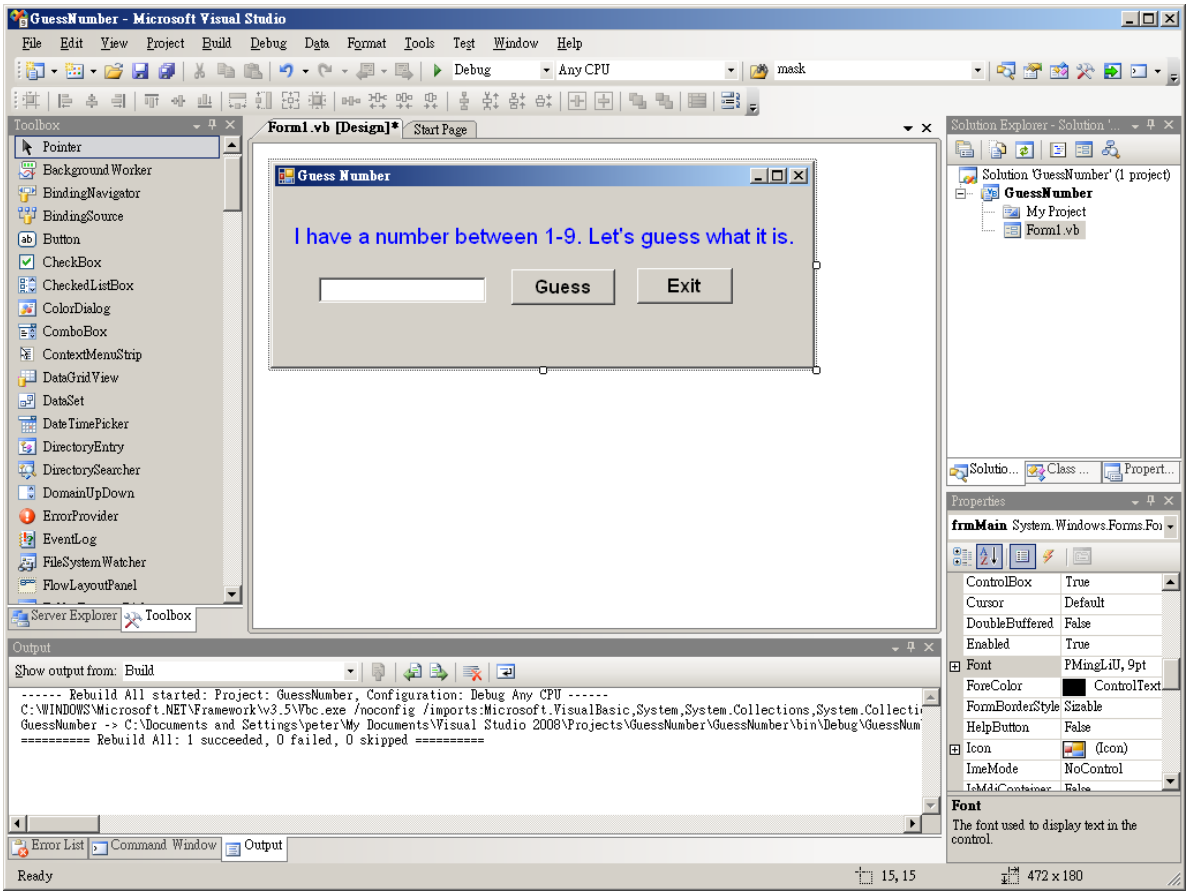


Copyright © 2010 Lo Chi Wing

10. Guess Number

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

Object	Name	Property	Property Value
Form	frmMain	Text	Guess Number
Label	Label1	Text	I have a number between 1-9. Let's guess what it is.
		Fore Color	Blue
		Font Size	12
Text Box	txtInput	Text	*(Blank)*
Button	btnGuess	Text	Guess
	btnExit	Text	Exit



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

```
Public Class frmMain
    Private RandomNumber As Integer
```

3. In the **Load** event procedure of the **frmMain** control, add the following code.

```
' Declare a random generator
Dim RandomGenerator As New Random

' Generate a random number from 1 to 9
RandomNumber = RandomGenerator.Next(1, 9)
```

4. In the **Click** event procedure of the **btnGuess** control, add the following code.

```
' Declare the variable to capture user action
Dim returnValue As DialogResult

' Validate the result
If txtInput.Text = RandomNumber Then
    MessageBox.Show("You are correct!", _
                    "Congratulation", _
                    MessageBoxButtons.OK, _
                    MessageBoxIcon.Information)
ElseIf txtInput.Text > RandomNumber Then
    returnValue = MessageBox.Show("Your number is too Large!", _
                                  "Sorry", _
                                  MessageBoxButtons.RetryCancel, _
                                  MessageBoxIcon.Exclamation)
    If returnValue = DialogResult.Cancel Then
        End
    End If
ElseIf txtInput.Text < RandomNumber Then
    returnValue = MessageBox.Show("Your number is too Small!", _
                                  "Sorry", _
                                  MessageBoxButtons.RetryCancel, _
                                  MessageBoxIcon.Exclamation)
    If returnValue = DialogResult.Cancel Then
        End
    End If
End If
```

5. In the **Click** event procedure of the **btnExit** control, add the following code.

```
' Exit the program
End
```

Copyright © 2010 Lo Chi Wing

6. Build and run your program. How many times you used to find the correct number?