Chapter Four

Decisions and Conditions

IF statement

In VB.NET there are many forms for the IF statement. They all work by evaluating some expression and if the expression is correct (evaluated to true) then the code within the IF block is executed. Now check out the first simple form of IF statement:

```
If expression Then
Statement
Statement
…
End If
```

The expression here is logical one. For example A>10, A<99, B>=A and so on. If the expression is correct, the statements inside the IF block get executed. The statement could be any valid VB.NET statement (even another IF statement). Now here is an example of the IF statement that always get executed:

```
If 10 < 100 Then
' display a friendly message
MsgBox("You must see this message")
End If
```

Since 10 is always smaller than 100 the condition always evaluates to true and you will always see the message. Now change it to be:

```
If 10 > 100 Then
' display a friendly message
MsgBox("You must see this message")
End If
```

The code within the block will never get executed. Now start a new project, put a button on the form and go to its event handler and add the following code:

```
Dim A As Integer
Dim B As Integer
A = InputBox("enter the value of A")
B = InputBox("enter the value of B")
If A > B Then
MsgBox("A is greater than B")
End If
If A < B Then
MsgBox("A is smaller than B")
End If
```

```
If A = B Then
MsgBox("A is equal to B")
End If
```

The InputBox is a function that reads a value from the keyboard. So the program reads two numbers and check their status. Run the program and try different values for A and B to see how it works. Also debug the program (by pressing F10 to execute one statement at a time) and see how the code get executed internally.

Another variation of the IF statement is the IF … ELSE. It has he following format:

```
If expression Then
Statement
Statement
…
Else
Statement
Statement
…
End If
```

The statements in black get executed when the expression is true, while the statements in red are ignored. However if the expression is evaluated to false then the statements in black are ignored while the statements in red are executed. To see how it works consider the following example:

```
If 10 > 100 Then
' this message never being displayed
MsgBox("10 is greater than 100")
Else
' this message is always being displayed
MsgBox("10 is smaller than 100, what a surprise!!!")
End If
```

The last form of the IF statement is the IF…ELSEIF… statement. Think of it as a multiple if statements combined into one. The form is as follows:

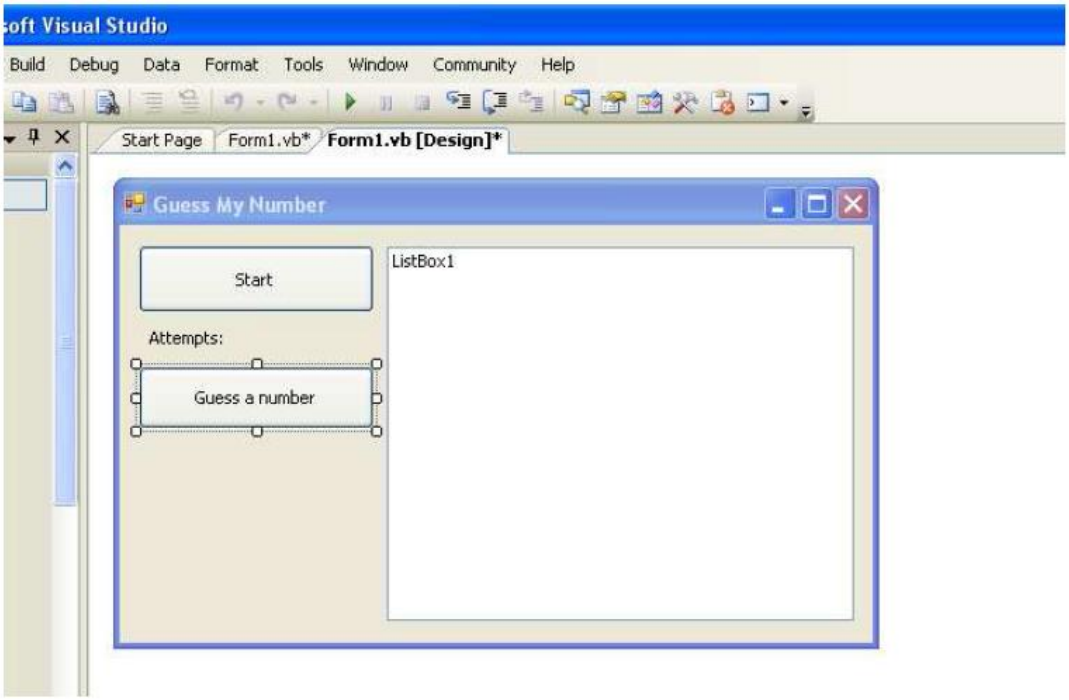
```
If expression1 Then
Statement
Statement
…
ElseIf expression2 Then
Statement
Statement
…
ElseIf expression3 Then
Statement
Statement
…
```

```
Else
Statement
Statement
…
End If
```

In this case if expression1 is evaluated to true, then its statements are executed and then the rest of the IF statement is ignored. If not, the expression2 is evaluated and its corresponding statements are executed and the rest of the checks are ignored… so check out the example below to have an idea about how it works:

```
If MyAge < 13 Then
' you must be a child
MsgBox("Child")
ElseIf MyAge < 20 Then
' you are a teenager
MsgBox("Hello Teenager")
ElseIf MyAge < 35 Then
' Your age is acceptable
MsgBox("Hi there young man")
Else
' the person is old
MsgBox("Hello there old man")
End If
```

So basically this is how the if statement works. We will create a simple Number Guessing Game and see how the IF statement helps us to do it. So basically start a new project, and Create a form with two buttons, a list box and a label as shown below:



Now for the Start button's event write the following code:

```
Dim SecretNumber As Integer
Dim Attempts As Integer
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
Randomize()
SecretNumber = Int(Rnd() * 100)
Attempts = 0
ListBox1.Items.Clear()
Label1.Text = "Attempts:" & Attempts.ToString
End Sub
```

The variables SecretNumber and Attempts are declared outside the subroutine so that their value will persist during program execution. The statements

```
Randomize()
SecretNumber = Int(Rnd() * 100)
```

Are used to generate a random number. The numbers are usually generated using some pattern. Each execution the same pattern of numbers appears. The first statement Randomize()makes sure that does not happen. The Rnd()function is used to generate a random number between 0 and 1. Multiply that by 100 you get a value between 0 and 100.

```
Attempts = 0
ListBox1.Items.Clear()
Label1.Text = "Attempts:" & Attempts.ToString
```

These statements resets the number of guessing attempts the play has made, and clears the listbox from previous attempts. The code for the second button is:

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
Dim MyNumber As Integer
Dim Tmp As String
Tmp = InputBox("Enter a number between 1 and 100", "Guessing
game")
If IsNumeric(Tmp) Then
MyNumber = Tmp
Else
MsgBox("you should enter a number")
Exit Sub
End If
If MyNumber = SecretNumber Then
MsgBox("You Guessed the correct number",
MsgBoxStyle.OkOnly)
ElseIf MyNumber > SecretNumber Then
ListBox1.Items.Add("you should enter a lower number")
```

```
MsgBox("your guess is wrong")
Else
ListBox1.Items.Add("you should enter a higher number")
MsgBox("your guess is wrong")
End If
Attempts = Attempts + 1
Label1.Text = "Attempts:" & Attempts.ToString
End Sub
```

The code is explained as follows:

```
Dim MyNumber As Integer
Dim Tmp As String
Tmp = InputBox("Enter a number between 1 and 100", "Guessing
game")
```

Here we define a number variable to store our guess in. We also need a string variable. This one will hold the value enter by the user so that we can check if it is a number of not (because the user can enter text value instead of a number).

```
If IsNumeric(Tmp) Then
MyNumber = Tmp
Else
MsgBox("you should enter a number")
Exit Sub
End If
```

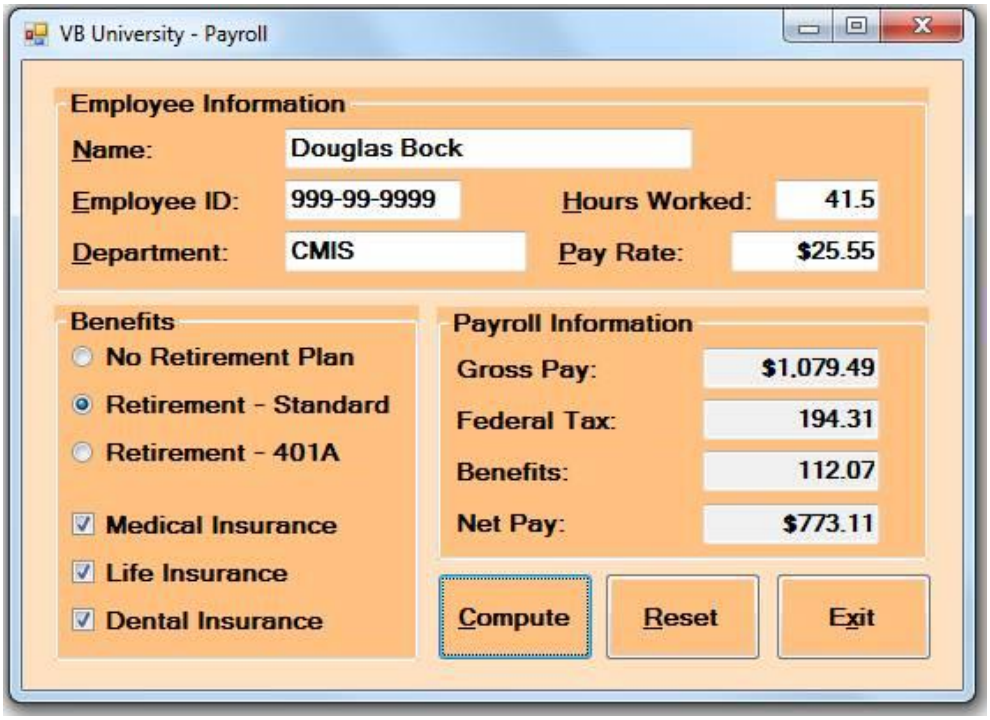
The IsNumeric is a function that is used to check if a string represent a number or not. So this part will assign the number inside Tmp into MyNumber if it is a proper number representation. Otherwise you get a message telling you about the error and the execution to the subroutine terminates because of the Exit Sub statement. Next:

```
If MyNumber = SecretNumber Then
MsgBox("You Guessed the correct number",
MsgBoxStyle.OkOnly)
ElseIf MyNumber > SecretNumber Then
ListBox1.Items.Add("you should enter a lower number")
MsgBox("your guess is wrong")
Else
ListBox1.Items.Add("you should enter a higher number")
MsgBox("your guess is wrong")
End If
```

This is the important part were we check the number against what the computer generated. If the numbers are a match then we display a message telling the user about his guess. If not the user get a wrong guess message and the computer tells if you should guess a higher or lower number. Finally:

```
Attempts = Attempts + 1
Label1.Text = "Attempts:" & Attempts.ToString
```

Will only update the number of attempts.



Solution to In-Class Exercise

```
'Project:  Ch04VBUniversity (Solution)
'D. Bock
'Today's Date

Public Class Payroll
    'Module level variable/constant declarations

    'Declare retirement benefit constants
    Const RETIREMENT_STANDARD_DECIMAL As Decimal = 0.05D
    Const RETIREMENT_401A_DECIMAL As Decimal = 0.08D
    Private RetirementRateDecimal As Decimal

    Private Sub ComputeButton_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles ComputeButton.Click
        Try
```

```
            'Declare variables and constants

            Dim HoursDecimal, PayRateDecimal, GrossPayDecimal,
FederalTaxDecimal, BenefitsCostDecimal, NetPayDecimal As Decimal


            'Declare constant used in this sub procedure

            'Tax rate constants

            Const TAX_RATE_08_DECIMAL As Decimal = 0.08D

            Const TAX_RATE_18_DECIMAL As Decimal = 0.18D

            Const TAX_RATE_28_DECIMAL As Decimal = 0.28D

            Const TAX_LEVEL_08_DECIMAL As Decimal = 985D

            Const TAX_LEVEL_18_DECIMAL As Decimal = 2450D


            'Benefit constants

            Const MEDICAL_RATE_DECIMAL As Decimal = 35.75D

            Const LIFE_RATE_DECIMAL As Decimal = 18.35D

            Const DENTAL_RATE_DECIMAL As Decimal = 4D


            'Enforce data validation rules

            If NameTextBox.Text.Trim = String.Empty Then

                'Required employee name is missing

                MessageBox.Show("Name is required", "Name Missing Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)

                NameTextBox.Focus()

                NameTextBox.SelectAll()

            ElseIf EmployeeIDMaskedTextBox.MaskCompleted = False Then

                'Required employee ID is not complete

                MessageBox.Show("Employee ID is not complete", "Employee
ID Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

                EmployeeIDMaskedTextBox.Focus()

                EmployeeIDMaskedTextBox.SelectAll()
```

```
            ElseIf DepartmentTextBox.Text.Trim = String.Empty Then

                'Required department is missing

                MessageBox.Show("Department is required", "Department
Missing Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

                DepartmentTextBox.Focus()

                DepartmentTextBox.SelectAll()

            ElseIf IsNumeric(HoursTextBox.Text)
= False OrElse (Decimal.Parse(HoursTextBox.Text,
Globalization.NumberStyles.Number) <=
0D OrDecimal.Parse(HoursTextBox.Text, Globalization.NumberStyles.Number)
> 60D) Then

                'Hours must be numeric and within allowable range

                MessageBox.Show("Hours worked must be a number between 0
and 60", "Hours Value Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

                HoursTextBox.Focus()

                HoursTextBox.SelectAll()

            ElseIf IsNumeric(PayRateTextBox.Text)
= False OrElse Decimal.Parse(PayRateTextBox.Text,
Globalization.NumberStyles.Currency) <= 0D Then

                'Pay rate must be numeric and greater than zero

                MessageBox.Show("Pay rate worked must be a number and
greater than zero.", "Pay Rate Value Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)

                PayRateTextBox.Focus()

                PayRateTextBox.SelectAll()

            Else

                'Data rules are all valid -- Use IPO model to process
data

                'Parse textbox values to memory variables

                HoursDecimal = Decimal.Parse(HoursTextBox.Text,
Globalization.NumberStyles.Number)

                PayRateDecimal = Decimal.Parse(PayRateTextBox.Text,
Globalization.NumberStyles.Currency)
```

```
                'Compute gross pay
                If HoursDecimal <= 40D Then  'pay only regular time
                    GrossPayDecimal = Decimal.Round(HoursDecimal *
PayRateDecimal, 2)
                Else  'pay regular + overtime
                    GrossPayDecimal = Decimal.Round((40D *
PayRateDecimal) _
                        + ((HoursDecimal - 40D) * PayRateDecimal * 1.5D),
2)
                End If

                'Compute federal tax
                Select Case GrossPayDecimal
                    Case Is <= TAX_LEVEL_08_DECIMAL  '8% tax bracket
                        FederalTaxDecimal
= Decimal.Round(TAX_RATE_08_DECIMAL * GrossPayDecimal, 2)
                    Case Is <= TAX_LEVEL_18_DECIMAL  '18% tax bracket
                        FederalTaxDecimal
= Decimal.Round(TAX_RATE_18_DECIMAL * GrossPayDecimal, 2)
                    Case Else  '28% tax bracket
                        FederalTaxDecimal
= Decimal.Round(TAX_RATE_28_DECIMAL * GrossPayDecimal, 2)
                End Select

                'Compute insurance benefits deduction
                If MedicalCheckBox.Checked Then
                    BenefitsCostDecimal +=
MEDICAL_RATE_DECIMAL  'selected medical insurance
                End If
                If LifeCheckBox.Checked Then
                    BenefitsCostDecimal += LIFE_RATE_DECIMAL 'selected
life insurance
```

```
                End If
                If DentalCheckBox.Checked Then
                    BenefitsCostDecimal += DENTAL_RATE_DECIMAL  'selected
dental insurance
                End If

                ''Remark out this part to test use of the CheckedChanged
event to
                ''set the retirement rate
                ''Compute retirement benefits deduction
                'If Retirement401ARadioButton.Checked Then
                '    BenefitsCostDecimal +=
Decimal.Round(RETIREMENT_401A_DECIMAL * GrossPayDecimal, 2)
                'ElseIf RetirementStandardRadioButton.Checked Then
                '    BenefitsCostDecimal +=
Decimal.Round(RETIREMENT_STANDARD_DECIMAL * GrossPayDecimal, 2)
                'Else
                '    'No charge for not taking retirement benefit
                'End If

                'Use the retirement rate set in the CheckedChanged event
                'for the retirement radio button controls
                BenefitsCostDecimal += Decimal.Round(GrossPayDecimal *
RetirementRateDecimal, 2)

                'Compute the net pay - no need to round because
                'all values are already rounded
                NetPayDecimal = GrossPayDecimal - FederalTaxDecimal -
BenefitsCostDecimal

                'Display output - this shows all four outputed values
```

```
                GrossPayTextBox.Text = GrossPayDecimal.ToString("C")

                FederalTaxTextBox.Text = FederalTaxDecimal.ToString("N")

                BenefitsTextBox.Text = BenefitsCostDecimal.ToString("N")

                NetPayTextBox.Text = NetPayDecimal.ToString("C")

            End If  'matches If statement for validating data

        Catch ex As Exception

            MessageBox.Show("Unexpected error: " & ControlChars.NewLine &
ex.Message, "Compute Button Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)

        End Try

    End Sub


    Private Sub ExitButton_Click(ByVal sender As System.Object, ByVal e A
s System.EventArgs) Handles ExitButton.Click

        'Close the form if the system user responds Yes

        Dim MessageString As String = "Do you want to close the form?"

        Dim ButtonDialogResult As DialogResult =
MessageBox.Show(MessageString, "Quit?", MessageBoxButtons.YesNo,
MessageBoxIcon.Question, MessageBoxDefaultButton.Button2)

        If ButtonDialogResult = Windows.Forms.DialogResult.Yes Then

            Me.Close()

        End If

    End Sub


    Private Sub ResetButton_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles ResetButton.Click

        'Clear all textbox controls

        NameTextBox.Clear()

        EmployeeIDMaskedTextBox.Clear()

        DepartmentTextBox.Clear()

        HoursTextBox.Clear()
```

```
        PayRateTextBox.Clear()
        GrossPayTextBox.Clear()
        FederalTaxTextBox.Clear()
        BenefitsTextBox.Clear()
        NetPayTextBox.Clear()

        'Reset retirement benefits status to none
        NoneRadioButton.Checked = True

        'Uncheck benefits checkboxes
        MedicalCheckBox.Checked = False
        LifeCheckBox.Checked = False
        DentalCheckBox.Checked = False

        'Set focus to name textbox
        NameTextBox.Focus()
    End Sub

    Private Sub NoneRadioButton_CheckedChanged(ByVal sender As System.Obj
ect, ByVal e As System.EventArgs) Handles NoneRadioButton.CheckedChanged,
Retirement401ARadioButton.CheckedChanged,
RetirementStandardRadioButton.CheckedChanged
        'Create a radio button in memory and store the values of sender
to it
        Dim CheckedRadioButton As RadioButton = CType(sender,
RadioButton)

        'Use Select Case to evaluate the name of the radio button
        'to decide which controls to enable/disable
        Select Case CheckedRadioButton.Name
            Case "NoneRadioButton"  'Cost is zero
```

```
                RetirementRateDecimal = 0D
            Case "RetirementStandardRadioButton"  'Standard rate
                RetirementRateDecimal = RETIREMENT_STANDARD_DECIMAL
            Case "Retirement401ARadioButton"  '401A rate
                RetirementRateDecimal = RETIREMENT_401A_DECIMAL
        End Select
    End Sub

End Class
```

For Loop

Almost every language has some kind of looping statement (in case you don't know what that does, it allows the execution of a number of statements several times).
In VB.NET there are a number of looping statements, these are REPEAT, DO and FOR. We will talk about the easiest of them all which is the FOR loop. The FOR loop is written like this:

```
For variable = Min To Max Step JumpStep
Statement
Statement
…
Next
```

The code will execute the statements between the For and Next parts by setting the variable to Min, increasing it by one every time until it reaches Max. To make things clear consider this example

```
For A = 1 To 10
MsgBox("The value of A is:" & A)
Next
```

The result of executing the code above is ten message boxes telling you the value of A every time.
Now let us consider another example. Here you have a form with a textbox and a ComboBox. You select font size from the combo box and the text size changes accordingly.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
' this part fills the combobox with the sizes of font that we
' can pick from
```

```
Dim I As Double
For I = 12 To 70
ComboBox1.Items.Add(I)
Next
End Sub
Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
' this part changes font size
Dim F As Font
F = New Font("COURIER NEW", ComboBox1.Text)
TextBox1.Font = F
End Sub
```

You notice two things in the example, first the loop does not start from 1, it starts from 12, you can start from any value you like, for example start from 283732, -12, 0, 88888, etc. Second the data type of the variable I is double. You can use Single, Double, Integers, Long… You are not restricted here. If we want to display the numbers between 5 and 50 by adding 5 to the previous in each step then:

```
Dim Counter As Integer
For Counter = 5 To 50 Step 5
MsgBox(Counter)
Next
```

Assume we need the values 0, 0.1, 0.2, 0.3, 0.4, 0.5… 1.0. This can be done in two ways:

```
Dim Counter As Integer
Dim V As Double
For Counter = 0 To 10
V = Counter / 10.0
MsgBox(V)
Next
```

This method requires extra variable, and does not take advantage of the for loop. A better way is to use the STEP keyword with double or single data type to make it easy for us:

```
Dim Counter As Double
For Counter = 0 To 1 Step 0.1
MsgBox(Counter)
Next
```

One last important thing to notice is that the initial value of the variable should always be smaller than or equal to the value after the To keyword, otherwise the for loop does not get executed and it is skipped. For example:

```
For Counter = 10 To 1
```

MsgBox(Counter)

Next

Will never give you message box at all. To fix this and make the countdown work, just put a negative step value:

For Counter = 10 To 1 Step -1

MsgBox(Counter)

Next

These are most of the details needed to work with the For loop. The next example is a simple one showing how to use the FOR loop to identify Prime number. Prime numbers are numbers that can only be divided by themselves and 1 with remainder=0. So this means if we have number 9212, we should check the remainder of
dividing this number over all the values from 9212 to 2 and it should never give a zero if it is a prime. Without for loop this is very hard to compute. The code to calculate the prime number is:

```
Dim MyNumber As Integer
Dim RemainderIsZeroFlag As Boolean
Dim I As Integer
' read a number from the screen
MyNumber = InputBox("Enter a number")
' this is a flag to tell us when the condition
' of prime number is not satisfied
RemainderIsZeroFlag = False
' start checking all the numbers
For I = 2 To MyNumber - 1
' if the condition is not satisfied
If MyNumber Mod I = 0 Then
' mark that the remainder is not zero
RemainderIsZeroFlag = True
End If
Next
' if there was any remainder then tell the user
' that the number is not prime, else it is.
If RemainderIsZeroFlag Then
MsgBox("The number is not prime")
Else
MsgBox("The number is prime")
End If
```