Visual Basic 2010 Tutorial

Table of Contents

Lesson 1- Introduction

Visual Basic 2010 is the latest version of Visual Basic launched by Microsoft in 2010. Visual Basic 2010 is a full fledged Object-Oriented Programming (OOP) Language, so it has caught up with other OOP languages such as C++, Java,C# and others.

However, you don't have to know OOP to learn VB2010. Visual Basic 2010 Express Edition is available free for download from the Microsoft site. Go to the link

http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-basic-express

The Integrated Development Environment when you launch VB2010 Express is shown in the diagram below. The IDE Start Page consists of a few sections, namely:

- The New Project/Open Project section.
- The Recent Projects section that shows a list of projects that have been created by you recently.
- The Getting Started Pane- It provides some helpful tips to quickly develop your applications.
- The Latest News section- It provides latest online news about Visual Basic 2010 Express. It will announce new releases and updates

The Properties section-let you defines the properties of each control
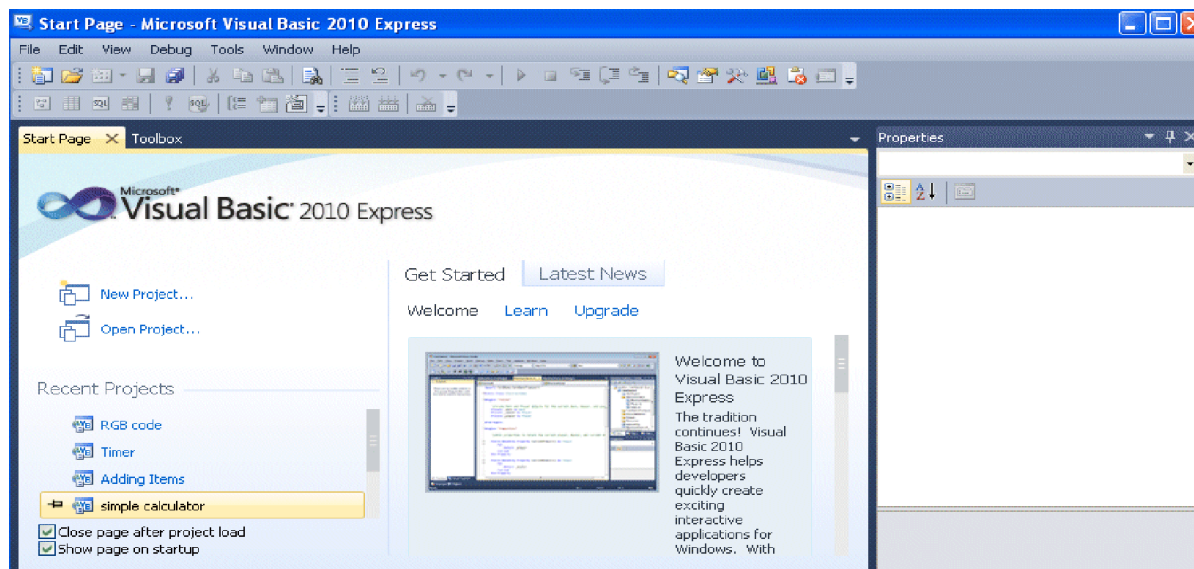
To start creating your first application, you need to click on New Project. The following VB2010 New Project dialog box will appear.
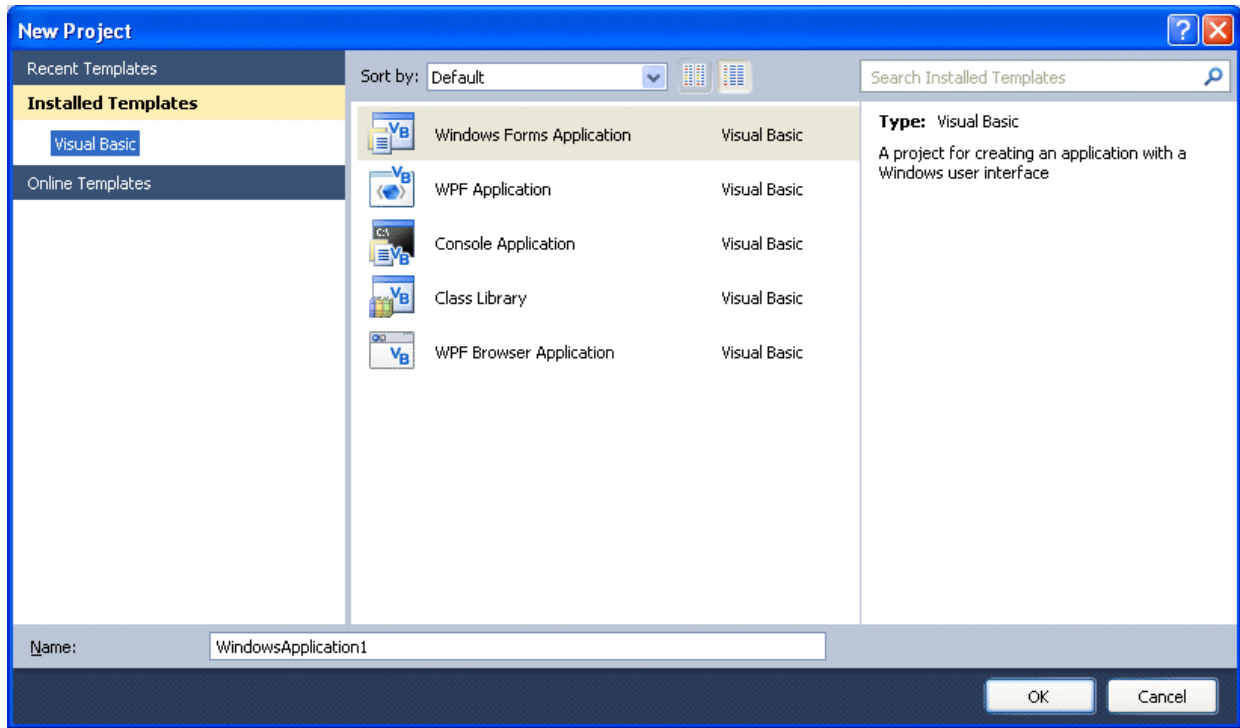
The dialog box offers you five types of projects that you can create. As we are going to learn to create windows Applications, we will select Windows Forms Application.

At the bottom of this dialog box, you can change the default project name WindowsApplication1 to some other name you like, for exampe, myFirstProgram. After you have renamed the project, click OK to continue. The following IDE Windows will appear, it is almost similar to Visual Basic 6. It consists of an empty form, the toolbox tab and the properties. The layout is slightly different from vb2008 as the Toolbox is not shown until you click on the Toolbox tab. When you click on the Toolbox tab, the common controls Toolbox will appear.
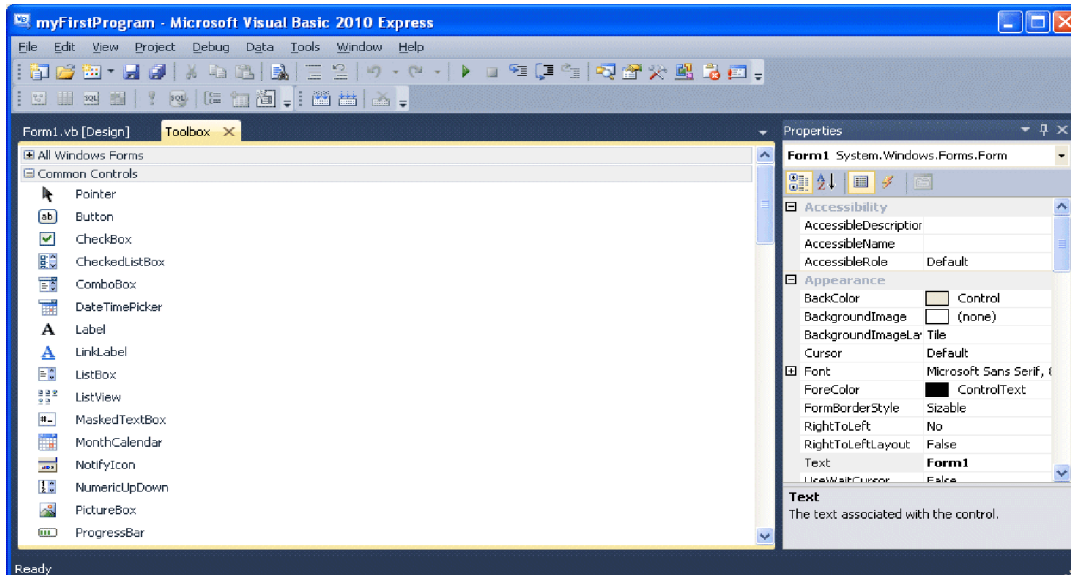
Now drag the button control into the form, and change its default Text Button1 to OK in the properties window, the word OK will appear on the button in the form, as shown below:
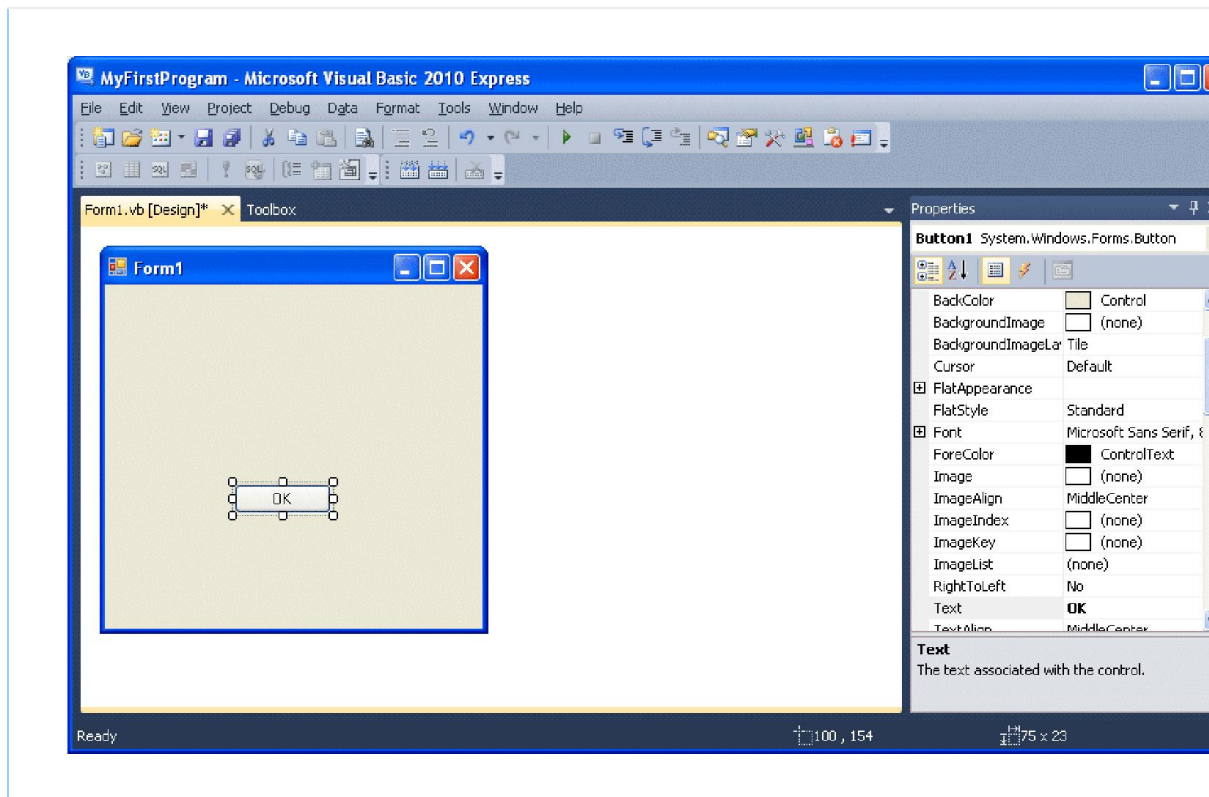
Now click on the OK button and the code window appears. Enter the code as follows:
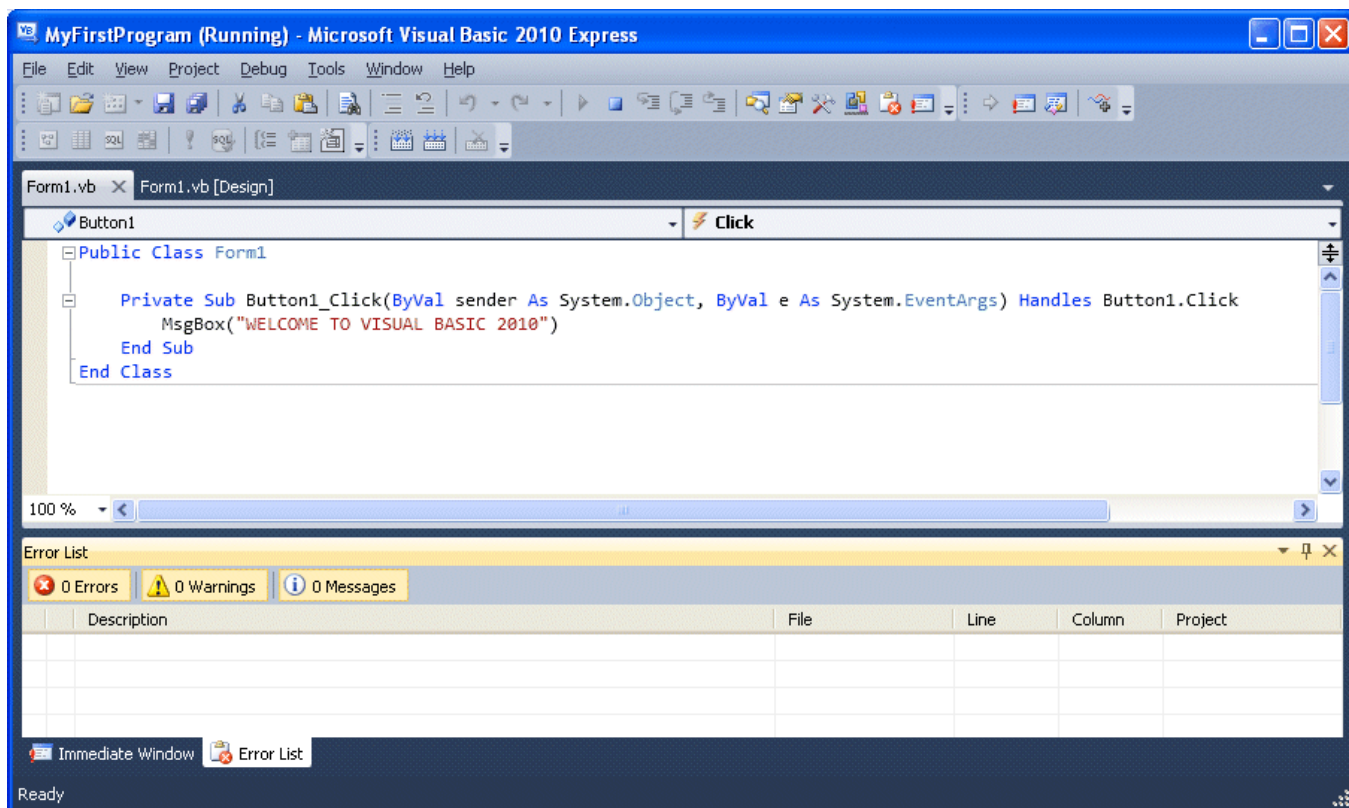
When you run the the program and click on the OK button, a dialog box will appear and display the "WELCOME TO VISUAL BASIC 2010" message,as shown below:
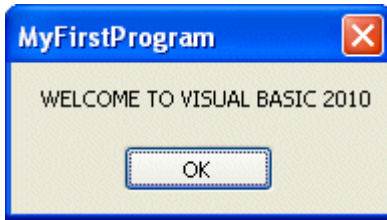
There you are, you have created your first VB2010 program.

Lesson 2-Working with Controls

Controls in Visual Basic 2010 are tools that can be placed in the form to perform various tasks. We can use them to create all kinds of Windows applications. The diagram below shows the Toolbox that contains the controls of Visual Basic 2010. They are categorized into Common Controls, Containers, Menus, Toolbars, Data, Components, Printings and Dialogs. At the moment, we will focus on the common controls. Some of the most used common controls are Button, Label, ComboBox, ListBox, PictureBox, TextBox and more.

To insert a control into your form, you just need to drag the control from the tool box and drop it into the form. You can reposition and resize it as you like. Let's examine a few examples that made use of Button, Label, TextBox , ListBox and PictureBox . You don't have to worry so much about the code yet because I will explain the program syntax as you progress to later lessons.

When you click on the Toolbox tab, the common controls Toolbox will appear.

## 2.1 Creating your first program

To create your first program, drag the button control into the form, and change its default Text Button1 to OK in the properties window, the word OK will appear on the button in the form, as shown below:
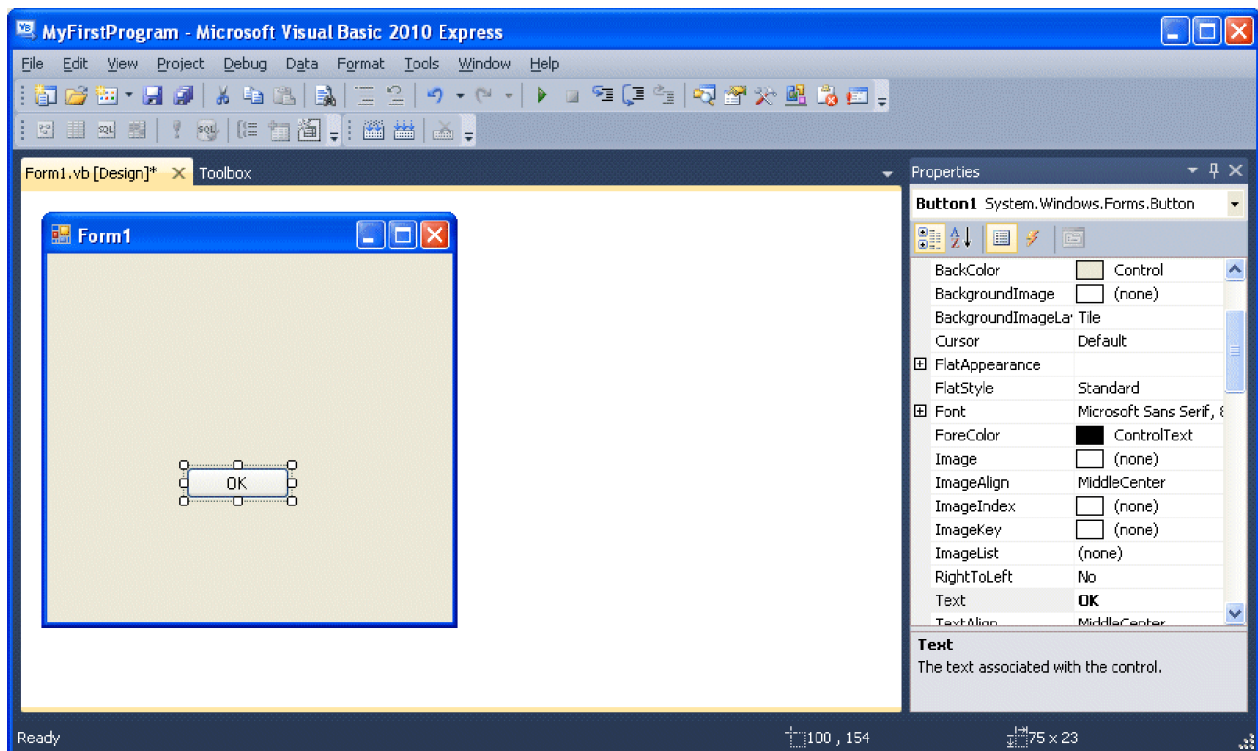
Now click on the OK button and the code window appears. Enter the code as follows:
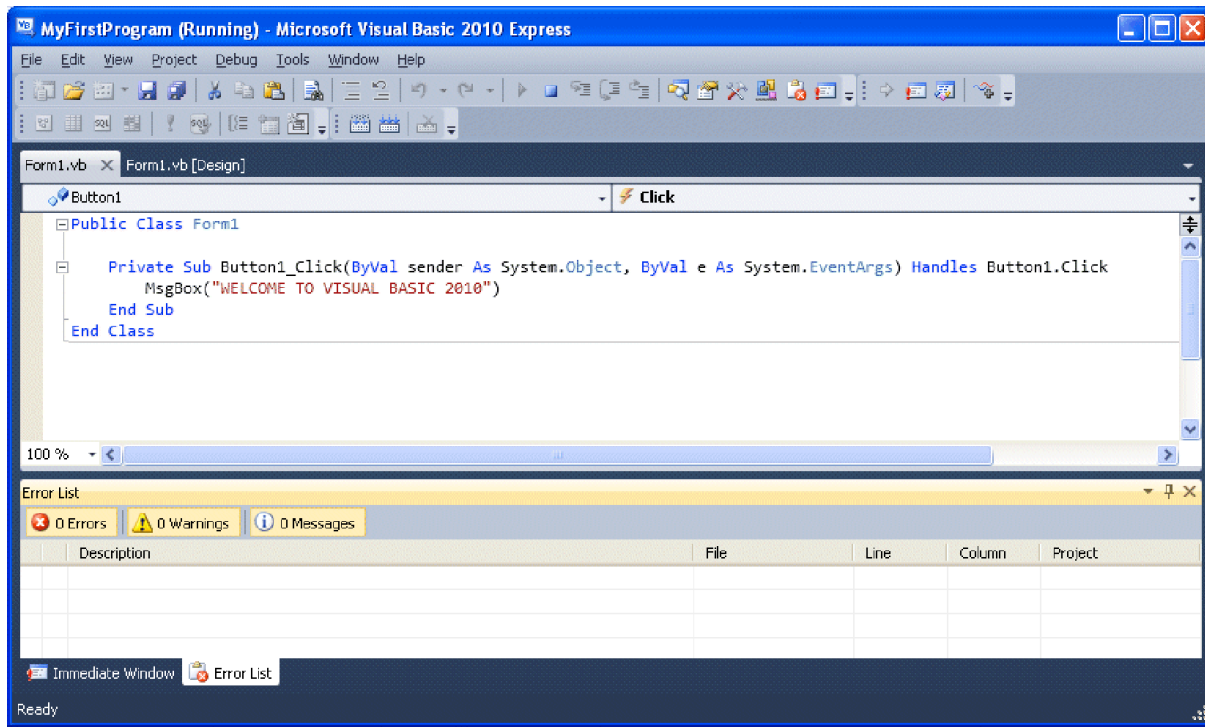
When you run the the program and click on the OK button, a dialog box will appear and display the "WELCOME TO VISUAL BASIC 2010" message,as shown below:
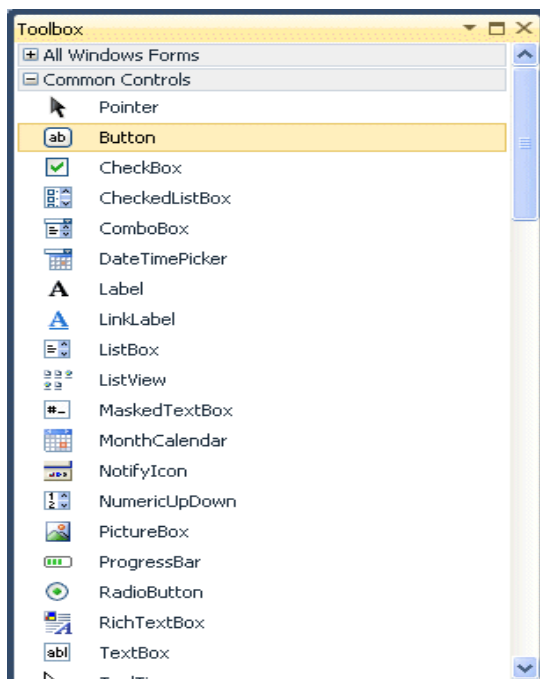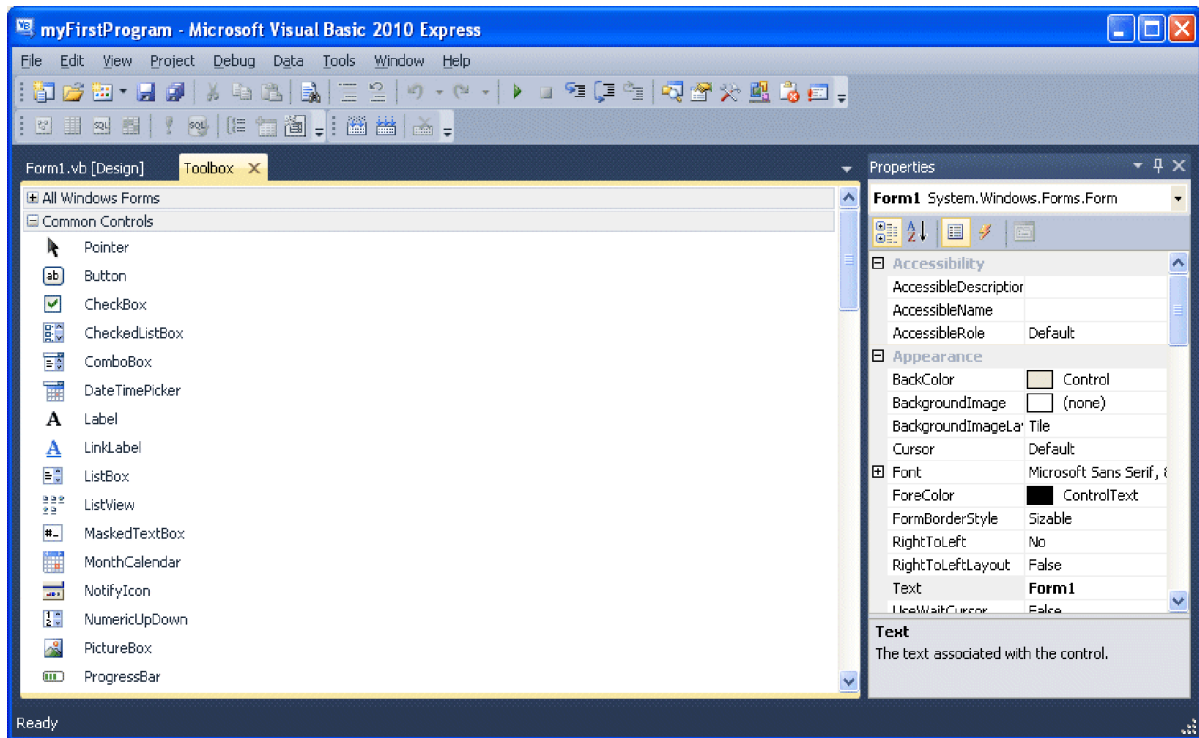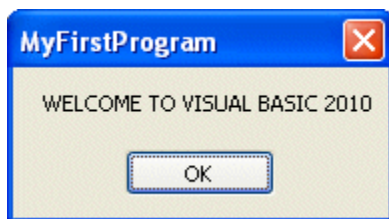

There you are, you have created your first Visual Basic 2010 program.

2.2 Using the Text Box

Next I will show you how to create a simple calculator that adds two numbers using the TextBox control. In this program, you insert two text boxes , three labels and one button. The two text boxes are for the users to enter two numbers, one label is to display the addition operator and the other label is to display the equal sign. The last label is to display the answer. Now change the label on the button to Calculate,then click on this button and enter the following code:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click

Dim num1, num2, product As Single

num1 = TextBox1.Text

num2 = TextBox2.Text

product = num1 + num2
```

Label1.Text = product

End Sub

When you run the program and enter two numbers, pressing the calculate button will allows the program to add the two numbers.
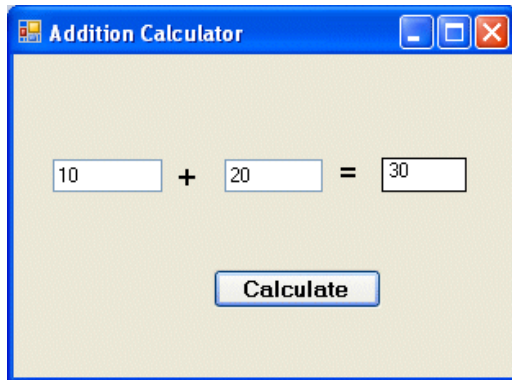
Lesson 3-Working with Control Properties

3.1 The Control Properties in VB2010

Before writing an event procedure for a control in Visual Basic 2010 to response to a user's input, you have to set certain properties for the control to determine its appearance and how it will work with the event procedure. You can set the properties of the controls in the properties window at design time or at runtime. Figure 3.1 is a typical properties window for a form in Visual Basic 2010 IDE:
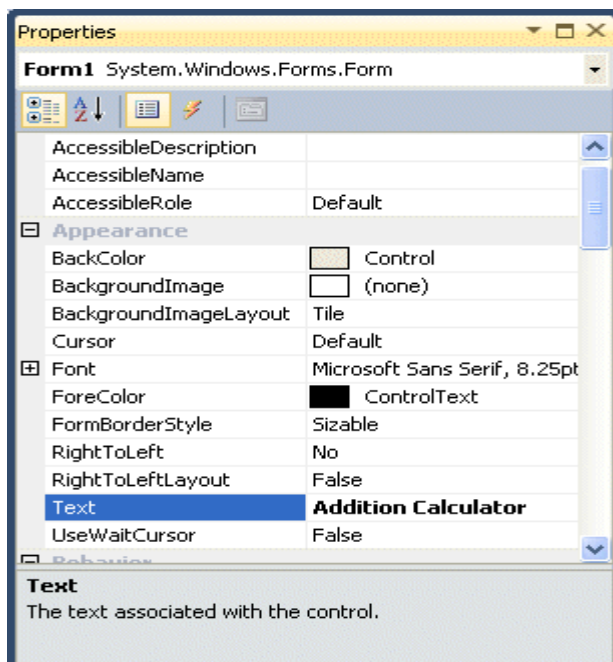
Figure 3.1

The title of the form is defined by the Text property and its default name is Form 1. To change the form's title to any name that you like, simple click in the box on the right of the Text property and type in the new name, in this example, the title is Addition Calculator. Notice that this title will appear on top of the window. In the properties window, the item appears at the top part is the object currently selected (in Figure 3.2, the object selected is Form1). At the bottom part, the items listed in the left column represent the names of various properties associated with the selected object while the items listed in the right column represent the states of the properties.
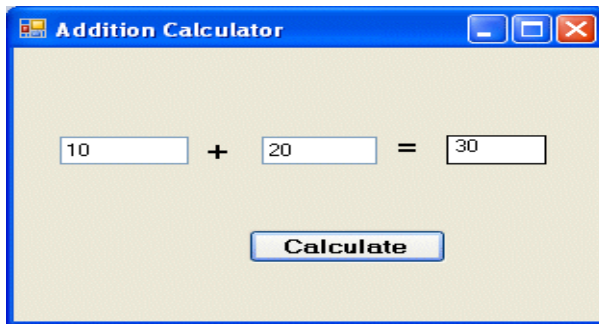
Figure 3.2

Properties can be set by highlighting the items in the right column then change them by typing or selecting the options available. You may also alter other properties of the form such as font, location, size, foreground color, background color ,MaximizeBox, MinimizeBox and etc. You can also change the properties of the object at runtime to give special effects such as change of color, shape, animation effect and so on.

For example the following code will change the form color to yellow every time the form is loaded. Visual Basic 2010 uses RGB(Red, Green, Blue) to determine the colors. The RGB code for yellow is 255,255,0. Me in the code refer to the current form and Backcolor is the property of the form's background color. The formula to assign the RGB color to the form is **Color.FormArbg(RGB code)**. The event procedure is as follows:

Public Class Form1 Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Me.BackColor = Color.FromArgb(255, 0, 255)

End Sub

End Class

You may also use the follow procedure to assign the color at run time.

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
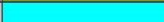
Me.BackColor = Color.Magenta

End Sub

Both procedures above will load the form with a magenta background as follows:

Here are some of the common colors and the corresponding RGB codes. You can always experiment with other combinations, but remember the maximum number for each color is 255 and the minimum number is 0.

Color	RGB code	Color	RGB code	Color	RGB Code
	255,0,0		255, 255, 0		255, 165, 0
	0,255,0		0, 255, 255		0, 0, 0
	0, 0, 255		255, 0, 255		255, 255, 255

The following is another program that allows the user to enter the RGB codes into three different text boxes and when he or she clicks the display color button, the background color of the form will change according to the RGB codes. So, this program allows users to change the color properties of the form at run time.
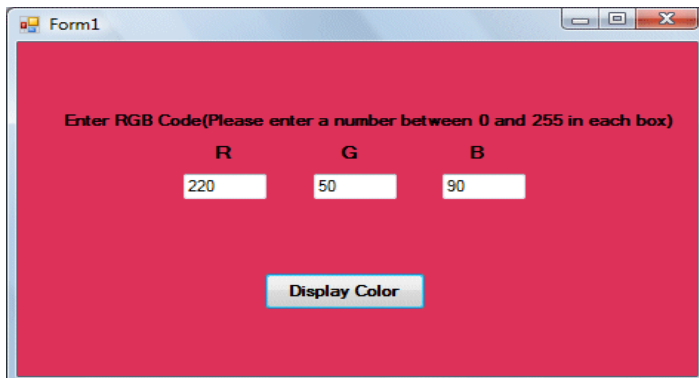
The code

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

Dim rgb1, rgb2, rgb3 As Integer
rgb1 = TextBox1.Text
rgb2 = TextBox2.Text
rgb3 = TextBox3.Text
Me.BackColor = Color.FromArgb(rgb1, rgb2, rgb3)

End Sub

Lesson 4 -Object Oriented Programming

In first three lessons, you have learned how to enter the program code and run the sample VB2010 programs but without much understanding about the logics of VB2010 programming. Now, let's get down to learning a few basic rules about writing the VB2010 program code.

First of all, let me say that though VB2010 is very much similar to VB6 in terms of Interface and program structure, their underlying concepts are quite different. The main different is that VB2010 is a full Object Oriented Programming Language while VB6 may have OOP capabilities, it is not fully object oriented. In order to qualify as a fully object oriented programming language, it must have three core technologies namely encapsulation, inheritance and polymorphism. These three terms are explained below:

Encapsulation

Encapsulation refers to the creation of self-contained modules that bind processing functions to the data. These user-defined data types are called classes. Each class contains data as well as a set of methods which manipulate the data. The data components of a class are called instance variables and one instance of a class is an object. For example, in a library system, a class could be member, and John and Sharon could be two instances (two objects) of the library class.

Inheritance

In object oriented programming, classes are created according to their hierarchies, and inheritance allows the structure and methods in one class to be passed down the hierarchy to another class. That means less programming is required when adding functions to complex systems, therefore save time and effort. If a step is added at the bottom of a hierarchy, then only the processing and data associated with that unique step needs to be added. Everything else about that step is inherited. The ability to reuse existing objects is considered a major advantage of object oriented programming.

Polymorphism
Object-oriented programming allows procedures about objects to be created whose exact type is not known until runtime. For example, a screen cursor may change its shape from an arrow to a line depending on the program mode. The routine to move the cursor on screen in response to mouse movement would be written for "cursor," and polymorphism allows that cursor to take on whatever shape is required at runtime. It also allows new shapes to be easily integrated.

VB6 is not a full OOP in the sense that it does not have inheritance capabilities although it can make use of some benefits of inheritance. However, VB2010 is a fully functional Object Oriented Programming Language, just like other OOP such as C++ and Java. It is different from the earlier versions of VB because it focuses more on the data itself while the previous versions focus more on the actions. Previous versions of VB are known as procedural or functional programming language. Some other procedural programming languages are C, Pascal and Fortran.

VB2010 allows users to write programs that break down into modules. These modules will represent the real-world objects and are knows as classes or types. An object can be created out of a class and it is known as an instance of the class. A class can also comprise subclass. For example, apple tree is a subclass of the plant class and the apple in your backyard is an instance of the apple tree class. Another example is student class is a subclass of the human class while your son John is an instance of the student class.

A class consists of data members as well as methods. In VB2010, the program structure to define a Human class can be written as follows:

```
Public Class Human
'Data Members
Private Name As String
Private Birthdate As String
Private Gender As String
Private Age As Integer
'Methods
Overridable Sub ShowInfo( )
MessageBox.Show(Name)
MessageBox.Show(Birthdate)
MessageBox.Show(Gender)
MessageBox.Show(Age)
End Sub
End Class
```

Let's look at one example on how to create a class. The following example shows you how to create a class that can calculate your BMI (Body Mass Index).

To create class, start VB2010 as usual and choose Windows Applications. In the VB2010 IDE, click on Project on the menu bar and select Add Class, the Add New Item dialog appears, as shown in the Figure below:
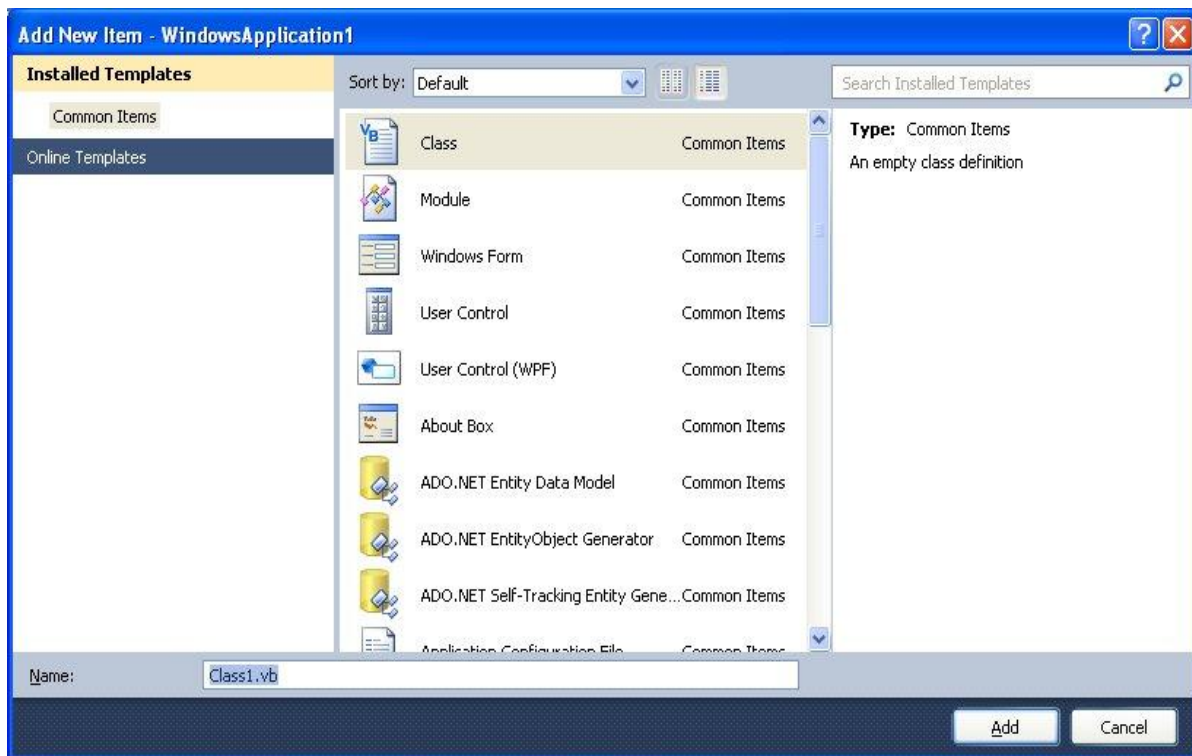
The default class Class1.vb will appear as a new tab with a code window. Rename the class as MyClass.vb. Rename the form as MyFirstClass.vb.

Now, in the MyClass.vb window, enter the follow code

Public Function BMI(ByVal height As Single, ByVal weight As Single)

BMI = Format((weight) / (height ^ 2), "0.00")

End Function

Now you have created a class (an object) called MyClass with a method known as BMI.

In order to use the BMI class, insert a button into the form and click on the button to enter the following code:

Private Sub BtnCalBmi_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnCalBmi.Click

Dim MyObject As Object

Dim h, w As Single

MyObject = New MyClass1()

h = InputBox("What is your height in meter")

w = InputBox("What is your weight in kg")

```
MessageBox.Show(MyObject.BMI(h, w))
```

End Sub

When you run this program and click the button, the user will be presented with two input boxes to enter his or her height and weight subsequently and the value of BMI will be shown in a pop-up message box.

Lesson 5-Writing the Code

In previous lesson, you have learned that Visual Basic 2010 is an object oriented programming language. You have understood the meanings of class, object, encapsulation inheritance as well as polymorphism. You have also learned to write some simple programs without much understanding some underlying foundations and theories. In this lesson, you will learn some basic theories about Visual Basic 2010 programming but we will focus more on learning by doing, i.e. learning by writing programs .I will keep the theories short so that it would not be too difficult for beginners.

5.1 The event Procedure

Visual Basic 2010 is an object oriented and event driven programming language. In fact, all windows applications are event driven. Event driven means the user will decide what to do with the program, whether he/she wants to click the command button, or he/she wants to enter text in a text box, or he/she might wants to close the application and etc. An event is related to an object, it is an incident that happens to the object due to the action of the user, such as a click or pressing a key on the keyboard. A class has events as it creates instant of a class or an object.

When we start a windows application in Visual Basic 2010 in previous chapters, we will see a default form with the Form1 appears in the IDE, it is actually the Form1 Class that inherits from the Form class System. Windows. Forms. Form, as shown in the Form1 properties windows.
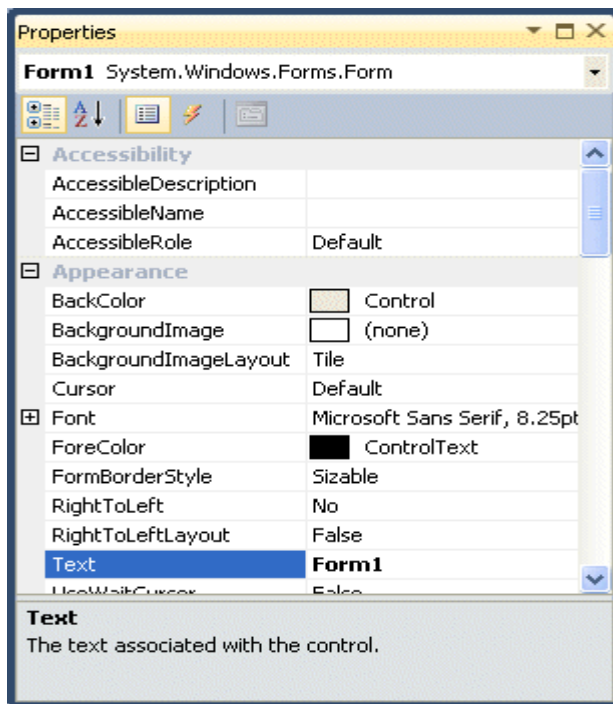
When we click on any part of the form, we will see the code window as shown below. The is the structure of an event procedure. In this case, the event procedure is to load Form1 and it starts with Private Sub and end with End Sub. This procedure includes the Form1 class and the event Load, and they are bind together with an underscore, i.e. Form_Load. It does nothing other than loading an empty form. You don't have to worry the rest of the stuff at the moment, they will be explained in later lessons.

Public Class Form1

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

End Sub

End Class

The are other events associated with the Form1 class, such as click, cursorChanged, DoubleClick, DragDrop, Enter as so on, as shown in the diagram below (It appears when you click on the upper right pane of the code window)
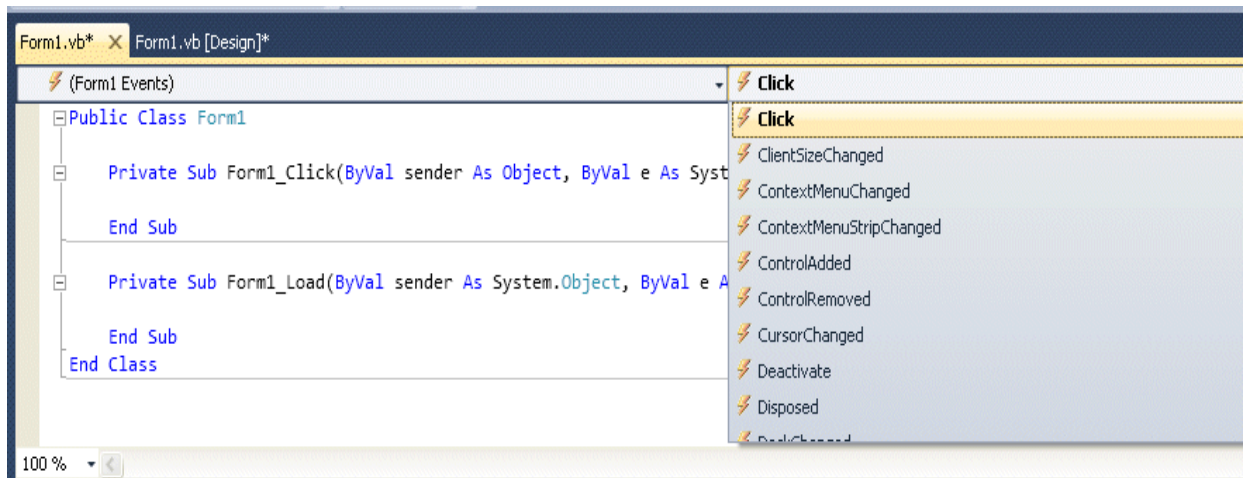
5.2 Writing the code

Now you are ready to write the code for the event procedure so that it will do something more than loading a blank form. The code must be entered between Private Sub…….End Sub. Let's enter the following code :

Public Class Form1

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Me.Text = "My First VB2010 Program"

Me.ForeColor = Color.ForestGreen

Me.BackColor = Color.Cyan

End Sub

End Classs

The first line of the code will change the title of the form to My First Visual Basic 2010 Program, the second line will change the foreground object to Forest Green(in this case, it is a label that you insert into the form and change its name to Foreground) and the last line changes the background to Cyan color.

The equal (=)in the code actually is used to assign something to the object, like assigning yellow color to the foreground of the Form1 object (or an instance of Form1). Me is the name given to the Form1 class. We can also call those lines as Statements. So, the actions of the program will depend on the statements entered by the programmer.

The output is shown in the windows below:

here is another example.

Private Sub Button1_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

Dim name1, name2, name3 As String
name1 = "John"
name2 = "Georges"
name3 = "Ali"
MsgBox(" The names are " & name1 & " , " & name2 & " and " & name3)

End Sub

In this example, you insert one command button into the form and rename its caption as Show Hidden Names. The keyword Dim is to declare variables name1, name2 and name3 as string, which means they can only handle text. The function MsgBox is to display the names in a message box that are joined together by the "&" signs. The output is shown below:
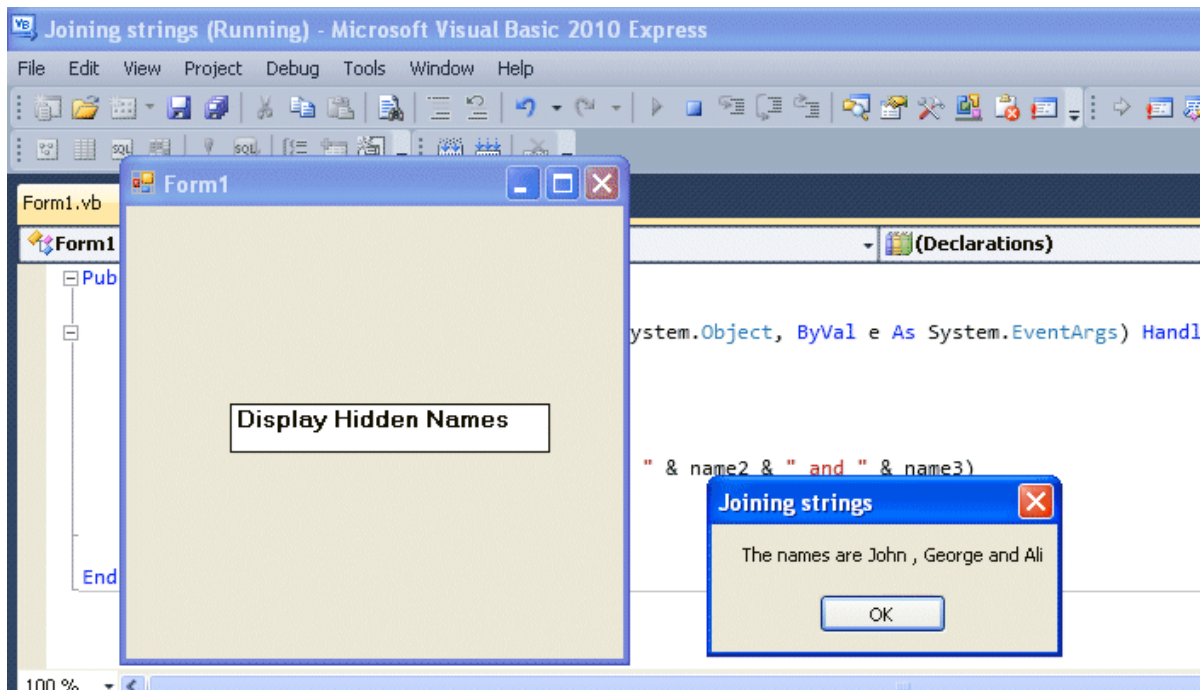
Lesson 6- Managing Data

We come across many types of information or data in our daily life. For example, we need to handle data such as names, addresses, money, date, stock quotes, statistics and more every day. Similarly in Visual Basic 2010, we have to deal with all sorts of data, some can be mathematically calculated while some are in the form of text or other forms. VB2010 divides data into different types so that it is easier to manage when we need to write the code involving those data.

6.1 Visual Basic 2010 Data Types

Visual Basic 2010 classifies the information mentioned above into two major data types, they are the numeric data types and the non-numeric data types.

6.1.1 Numeric Data Types

Numeric data types are types of data that consist of numbers, which can be computed mathematically with various standard operators such as add, minus, multiply, divide and so on. In Visual Basic 2010, numeric data are divided into 7 types, depending on the range of values they can store. Calculations that only involve round figures or data that don't need precision can use Integer or Long integer in the computation. Programs that require high precision calculation need to use Single and Double decision data types, they are also called floating point numbers. For currency calculation, you can use the currency data types. Lastly, if even more precision is requires to perform calculations that involve a many decimal points, we can use the decimal data types. These data types summarized in Table 6.1

Table 6.1: Numeric Data Types

Type	Storage	Range of Values
Byte	1 byte	0 to 255
Integer	2 bytes	-32,768 to 32,767
Long	4 bytes	-2,147,483,648 to 2,147,483,648
Single	4 bytes	-3.402823E+38 to -1.401298E-45 for negative values 1.401298E-45 to 3.402823E+38 for positive values.
Double	8 bytes	-1.79769313486232e+308 to -4.94065645841247E-324 for negative values 4.94065645841247E-324 to 1.79769313486232e+308 for positive values.
Currency	8 bytes	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
Decimal	12 bytes	+/- 79,228,162,514,264,337,593,543,950,335 if no decimal is use +/- 7.9228162514264337593543950335 (28 decimal places).

6.1.2 Non-numeric Data Types

Nonnumeric data types are data that cannot be manipulated mathematically using standard arithmetic operators. The non-numeric data comprises text or string data types, the Date data types, the Boolean data types that store only two values (true or false), Object data type and Variant data type .They are summarized in Table 6.2

Table 6.2: Nonnumeric Data Types

Data Type	Storage	Range
String(fixed length)	Length of string	1 to 65,400 characters
String(variable length)	Length + 10 bytes	0 to 2 billion characters
Date	8 bytes	January 1, 100 to December 31, 9999
Boolean	2 bytes	True or False
Object	4 bytes	Any embedded object
Variant(numeric)	16 bytes	Any value as large as Double
Variant(text)	Length+22 bytes	Same as variable-length string

6.1.3 Suffixes for Literals

Literals are values that you assign to a data. In some cases, we need to add a suffix behind a literal so that VB2010 can handle the calculation more accurately. For example, we can use num=1.3089# for a Double type data. Some of the suffixes are displayed in Table 6.3.

Table 6.3

Suffix	Data Type
&	Long
!	Single
#	Double
@	Currency

In addition, we need to enclose string literals within two quotations and date and time literals within two # sign. Strings can contain any characters, including numbers. The following are few examples:

memberName="Turban, John."
TelNumber="1800-900-888-777"
LastDay=#31-Dec-00#
ExpTime=#12:00 am#

6.2 Managing Variables

Variables are like mail boxes in the post office. The contents of the variables changes every now and then, just like the mail boxes. In term of VB2010, variables are areas allocated by the computer memory to hold data. Like the mail boxes, each variable must be given a name. To name a variable in Visual Basic 2010, you have to follow a set of rules.

6.2.1 Variable Names

The following are the rules when naming the variables in Visual Basic 2010

It must be less than 255 characters
No spacing is allowed
It must not begin with a number
Period is not permitted
Examples of valid and invalid variable names are displayed in Table 6.4

Table 6.4

Valid Name	Invalid Name	
My_Car	My.Car	
ThisYear	1NewBoy	
Long_Name_Can_beUSE	He&HisFather	*& is not acceptable

6.2.2 Declaring Variables

In Visual Basic 2010, one needs to declare the variables before using them by assigning names and data types. If you fail to do so, the program will show an error. They are normally declared in the general section of the codes' windows using the Dim statement. The format is as follows:

Dim Variable Name As Data Type

Example 6.1

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Dim password As String
Dim yourName As String
Dim firstnum As Integer
Dim secondnum As Integer
Dim total As Integer
Dim doDate As Date

End Sub

You may also combine them in one line , separating each variable with a comma, as follows:

Dim password As String, yourName As String, firstnum As Integer,………….

For string declaration, there are two possible formats, one for the variable-length string and another for the fixed-length string. For the variable-length string, just use the same format as example 6.1 above. However, for the fixed-length string, you have to use the format as shown below:

Dim VariableName as String * n, where n defines the number of characters the string can hold.

Example 6.2:

Dim yourName as String * 10
yourName can holds no more than 10 Characters.

6.2.3 Assigning Values to Variables

After declaring various variables using the Dim statements, we can assign values to those variables. The general format of an assignment is

Variable=Expression

The variable can be a declared variable or a control property value. The expression could be a mathematical expression, a number, a string, a Boolean value (true or false) and etc. The following are some examples:

firstNumber=100
secondNumber=firstNumber-99
userName="John Lyan"
userpass.Text = password
Label1.Visible = True
Command1.Visible = false
Label4.Caption = textbox1.Text
ThirdNumber = Val(usernum1.Text)
total = firstNumber + secondNumber+ThirdNumber

6.3 Constants

Constants are different from variables in the sense that their values do not change during the running of the program.

6.3.1 Declaring a Constant

The format to declare a constant is
Const Constant Name As Data Type = Value

Example 6.3

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Const Pi As Single=3.142
Const Temp As Single=37
Const Score As Single=100
End Sub

Lesson 7- Mathematical Operations

Computer can perform mathematical calculations much faster than human beings. However, computer itself will not be able to perform any mathematical calculations without receiving instructions from the user. In Visual Basic 2010, we can write code to instruct the computer to perform mathematical calculations such as addition, subtraction, multiplication, division and other kinds of arithmetic operations. In order for Visual Basic 2010 to carry out arithmetic calculations, we need to write code that involve the use of various arithmetic operators. The Visual Basic 2010 arithmetic operators are very similar to the normal arithmetic operators, only with slight variations. The plus and minus operators are the same while the multiplication operator use the * symbol and the division operator use the / symbol. The list of Visual Basic 2010 arithmetic operators are shown in table 7.1 below:

Table 7.1: Arithmetic Operators

Operator	Mathematical function	Example
+	Addition	1+2=3
--	Subtraction	4-1=3
^	Exponential	2^4=16
*	Multiplication	4*3=12, (5*6))2=60
/	Division	12/4=3
Mod	Modulus (return the remainder from an integer division)	15 Mod 4=3 255 mod 10=5
\	Integer Division (discards the decimal places)	19\4=4

Example 7.1

In this program, you need to insert two Text boxes, four labels and one button. Click the button and key in the code as shown below. Note how the various arithmetic operators are being used. When you run the program, it will perform the four basic arithmetic operations and display the results on the four labels.

```
Dim num1, num2, difference, product, quotient As Single
num1 = TextBox1.Text
num2 = TextBox2.Text
sum=num1+num2
difference=num1-num2
product = num1 * num2
quotient=num1/num2
Label1.Text=sum
Label2.Text=difference
Label3.Text = product
Label4.Text = quotient
```

Example 7.2:Pythagoras Theorem

The program can use Pythagoras Theorem to calculate the length of hypotenuse c given the length of the adjacent side a and the opposite side b. In case you have forgotten the formula for the Pythagoras Theorem, it is written as $c^2=a^2+b^2$

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click

Dim a, b, c As Single
a = TextBox1.Text
b = TextBox2.Text
c=(a^2+b^2)^(1/2)
Label3.Text=c

End Sub
```

Example 7.3: BMI Calculator

A lot of people are obese now and it could affect their health seriously . Obesity has proven by the medical experts to be a one of the main factors that brings many adverse medical problems, including the the heart disease. If your BMI is more than 30, you are considered obese. You can refer to the following range of BMI values for your weight status.

Underweight = <18.5
Normal weight = 18.5-24.9
Overweight = 25-29.9
Obesity = BMI of 30 or greater
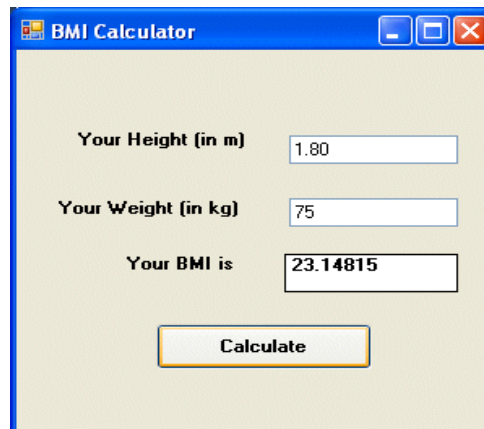
In order to calculate your BMI, you do not have to consult your doctor, you could just use a calculator or a home made computer program, this is exactly what I am showing you here. The BMI calculator is a Visual Basic program that can calculate the body mass index, or BMI of a person based on the body weight in kilogram and the body height in meter. BMI can be calculated using the formula weight/(height)2, where weight is measured in kg and height in meter. If you only know your weight and height in lb and feet, then you need to convert them to the metric system (you could indeed write a VB program for the conversion).

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArsgs)
Handles Button1.Click

Dim height, weight, bmi As Single
height = TextBox1.Text
weight = TextBox2.Text
bmi = (weight) / (height ^ 2)
Label4.Text = bmi

End Sub
```

The output is shown in the figure below. In this example, your height is 1.80m(about 5 foot 11),your weight is 75 kg(about 168Ib), and your BMI is about 23.14815. The reading suggests that you are healthy. (Note; 1 foot=0.3048, 1 lb=.45359237 kilogram)

From the above examples, you can see that writing code that involve arithmetic operations is relatively easy. Here are more arithmetic projects you can try to program:

From the above examples, you can see that perform arithmetic operations is relatively easy. Here are more arithmetic projects you can try to programs:

Area of a triangle
Area of a rectangle
Area of a circle
Volume of a cylinder
Volume of a cone
Volume of a sphere
Compound interest
Future value
Mean
Variance
Sum of angles in polygons
Conversion of lb to kg
Conversion of Fahrenheit to Celsius

Lesson 8- String Manipulation

String manipulation is an important part of programming because it help to process data that come in the form of non-numeric types such as name, address, gender, city, book title and more.

8.1 String Manipulation Using + and & signs.

In Visual Basic 2010 ,Strings can be manipulated using the & sign and the + sign, both perform the string concatenation which means combining two or more smaller strings into larger strings. For example, we can join "Visual" and "Basic" into "Visual Basic" using "Visual"&"Basic" or "Visual "+"Basic", as shown in the example below:

Example 8.1(a)

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
Dim text1, text2, text3 As String
text1 = "Visual"
text2 = "Basic"
text3 = text1 + text2
Label1.Text = text3

End Sub
```

The line text3=text1+ text2 can be replaced by text3=text1 & text2 and produced the same output. However, if one of the variables is declared as numeric data type, you cannot use the + sign, you can only use the & sign.

Example 8.1(b)

```
Dim text1, text3 as string
Dim Text2 As Integer

text1 = "Visual"

text2=22

text3=text1+text2

Label1.Text = text3
```

This code will produce an error because of data mismatch.However, using & instead of + will be all right.
```
Dim text1, text3 as string
Dim Text2 As Integer
text1 = "Visual"
text2=22
text3=text1 & text2
Label1.Text = text3
```
You can combine more than two strings to form a larger strings, like the following example:
```
Public Class Form1
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim text1, text2, text3, text4, text5, text6 As String
text1 = "Welcome"
text2 = " to"
text3 = " Visual"
text4 = " Basic"
text5 = " 2010"
```

```
text6 = text1 + text2 + text3+text4+text5
Label1.Text = text6
End Sub
End Class
```
Running the above program will produce the following screen shot.
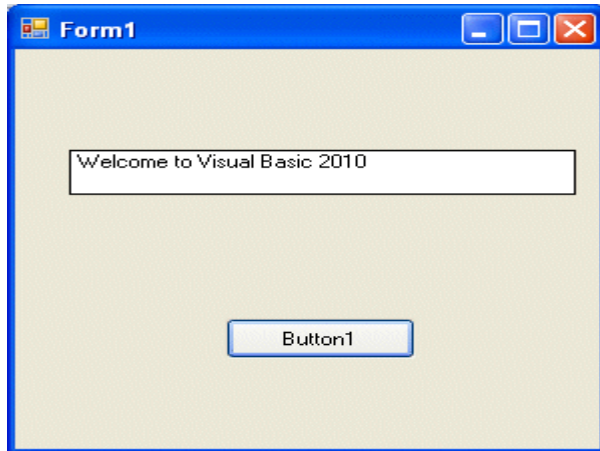
8.2 String Manipulation Using Visual Basic 2010 Built-in Functions

A function is similar to a normal procedure but the main purpose of the function is to accept a certain input and return a value which is passed on to the main program to finish the execution.There are numerous string manipulation functions built into Visual Basic 2010 but I will only discuss a few here and will explain the rest of them in later lessons.

8.2 (a) The Len Function

The length function returns an integer value which is the length of a phrase or a sentence, including the empty spaces. The format is
Len ("Phrase")
For example,
Len (Visual Basic) = 12 and Len (welcome to VB tutorial) = 22
Example 8.3
Public Class Form1

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Label1.Text = Len(TextBox1.Text)

End Sub
End Class
```

The output:
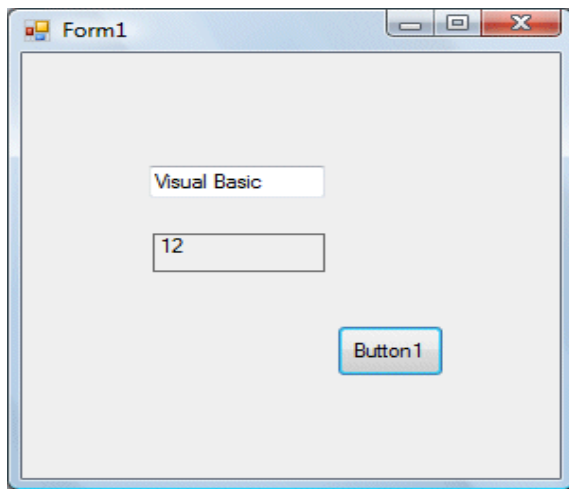
8.2(b) The Right Function

The Right function extracts the right portion of a phrase. The format for Visual Basic 6 is

Right ("Phrase", n)

Where n is the starting position from the right of the phase where the portion of the phrase is going to be extracted. For example,

Right("Visual Basic", 4) = asic

However, this format is not applicable in VB2010. In VB2010, we need use the following format

Microsoft.VisualBasic.Right("Phrase",n)

Example 8.2(a)

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim text1 As String

text1 = TextBox1.Text

Label1.Text = Microsoft.VisualBasic.Right(text1, 4)

End Sub
```

The above program will return four right most characters of the phrase entered into the textbox.

The Output:
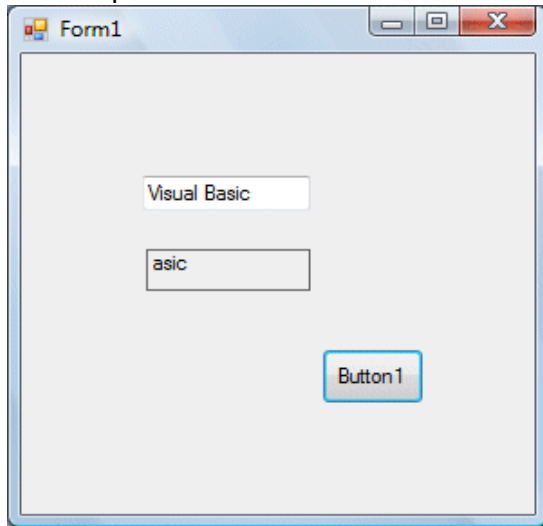
*The reason of using the full reference is because many objects have the Right properties so using Right on its own will make it ambiguous to VB2010.

8.2(c)The Left Function

The Left function extract the left portion of a phrase. The format is

Microsoft.VisualBasic.Left("Phrase",n)

Where n is the starting position from the left of the phase where the portion of the phrase is going to be extracted. For example,

Microsoft.VisualBasic.Left ("Visual Basic", 4) = Visu .

Lesson 9- Using If....Then....Else

In the previous lessons, we have learned how to write code that can accept input from the users and then display the output without controlling the program flow. In this lesson, you will learn how to write Visual Basic 2010 code that can make decision when it process input from the users, and control the program flow in the process.

Decision making process is an important part of programming because it can solve practical problems intelligently and provide useful output or feedback to the user. For example, we can write a Visual Basic 2010 program that can ask the computer to perform certain task until a certain condition is met, or a program that will reject non-numeric data. In order to control the program flow and to make decisions, we need to use the conditional operators and the logical operators together with the If control structure.

9.1 Conditional Operators

The conditional operators are powerful tools that resemble mathematical operators . These operators allow a VB2010 program to compare data values and then decide what actions to take, whether to execute a program or terminate the program and more. They are also known as numerical comparison operators. Normally they are used to compare two values to see whether they are equal or one value is greater or less than the other value. The comparison will return a true or false result. These operators are shown in Table 9.1.

Table 9.1: Conditional Operators

Operator	Meaning
=	Equal to
>	More than
<	Less Than
>=	More than and equal
<=	Less than and equal
<>	Not Equal to

9.2 Logical Operators

Sometimes we might need to make more than one comparisons before a decision can be made and an action taken. In this case, using numerical comparison operators alone is not sufficient, we need to use additional operators, and they are the logical operators. These logical operators are shown in Table 9.2.

Table 9.2

Operator	Meaning
And	Both sides must be true
or	One side or other must be true
Xor	One side or other must be true but not both
Not	Negates truth

* Normally the above operators are use to compare numerical data. However, you can also compare strings with the above operators. In making strings comparison, there are certain rules to follows: Upper case letters are less than lowercase letters, "A"<"B"<"C"<"D"…….<"Z" and number are less than letters.

9.3 Using the If control structure with the Comparison Operators

To effectively control the Visual Basic 2010 program flow, we shall use the If control structure together with the conditional operators and logical operators. There are basically three types of If control structures, namely If….Then statement, If….Then… Else statement and If….Then….ElseIf statement.

9.3(a)   If….Then   Statement
This is the simplest control structure which ask the computer to perform a certain action

specified by the Visual Basic 2010 expression if the condition is true. However, when the condition is false, no action will be performed. The general format for the if...then.. statement is
If condition Then
Visual Basic 2010 expression
End If

Example 9.1

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim myNumber As Integer
myNumber = TextBox1.Text
If myNumber > 100 Then
Label2.Text = " You win a lucky prize"
End If
End Sub
```

* When you run the program and enter a number that is greater than 100, you will see the "You win a lucky prize" statement. On the other hand, if the number entered is less than or equal to 100, you don't see any display.

9.3(b) If....Then...Else Statement

Using only If....Then statement is not very useful in programming and it does not provide choices for the users. In order to provide a choice, we can use the If....Then...Else Statement. This control structure will ask the computer to perform a certain action specified by the Visual Basic 2010 expression if the condition is true. And when the condition is false , an alternative action will be executed. The general format for the if...then.. Else statement is

```
If condition Then

Visual Basic 2010 expression
Else
Visual Basic 2010 expression
End If
```

Example 9.2

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim myNumber As Integer
myNumber = TextBox1.Text
If myNumber > 100 Then
Label2.Text = " Congratulation! You win a lucky prize"
Else
Label2.Text = " Sorry, You dif not win any prize"
End If
End Sub
```

* When you run the program and enter a number that is greater than 100, the statement "Congratulation! You win a lucky prize" will be shown. On the other hand, if the number entered is less than or equal to 100, you will see the "Sorry, You dif not win any prize" statement

Example 9.3

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim myNumber, MyAge As Integer
myNumber = TextBox1.Text
MyAge = TextBox2.Text

If myNumber > 100 And myAge > 60 Then
Label2.Text = " Congratulation! You win a lucky prize"
Else
Label2.Text = " Sorry, You did not win any prize"
End If
End Sub
```

* This program use the logical And operator beside the conditional operators. This means that both the conditions must be fulfilled in order for the conditions to be true, otherwise the second block of code will be executed. In this example, the number entered must be more than 100 and the age must be more than 60 in order to win a lucky prize, any one of the above conditions not fulfilled will disqualify the user from winning a prize.

9.3(c) If….Then…ElseIf Statement

If there are more than two alternative choices, using jus If….Then….Else statement will not be enough. In order to provide more choices, we can use the If….Then…ElseIf Statement. executed. The general format for the if…then.. Else statement is

```
If condition Then
Visual Basic 2010 expression
ElseIf condition Then
Visual Basic 2010 expression
ElseIf condition Then
Visual Basic 2010 expression
.
.
Else
Visual Basic 2010 expression
End If
```

Example 9.4
```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim Mark As Integer
Dim Grade as String
```

```
Mark = TextBox1.Text
If myNumber >=80 Then
Grade="A"
ElseIf Mark>=60 and Mark<80 then
Grade="B"
ElseIf Mark>=40 and Mark<60 then
Grade="C"
Else
Grade="D"
End If
End Sub
```

Lesson 10- Using Select Case

In the previous lesson, we have learned how to control the program flow using the If...ElseIf control structure. In this lesson, you will learn another way to control the program flow in Visual Basic 2010, that is, the Select Case control structure.

Select Case control structure is slightly different from the If....ElseIf control structure . The difference is that the Select Case control structure basically only make decision on one expression or dimension (for example the examination grade) while the If ...ElseIf statement control structure may evaluate only one expression, each If....ElseIf statement may also compute entirely different dimensions. Select Case is preferred when there exist multiple conditions because using If...Then..ElseIf statements will become too messy.

10.1 The Select Case…End Select Structure

The format of the Select Case control structure is as follows:

Select Case test expression

Case expression list 1
Block of one or more Visual Basic 2010 statements
Case expression list 2
Block of one or more Visual Basic 2010 Statements
Case expression list 3
Block of one or more Visual Basic 2010 statements
Case expression list 4
.
.
.
Case Else
Block of one or more Visual Basic 2010 Statements

End Select

10.2 The usage of Select Case is shown in the following examples

Example 10.1	Example 10.2
| ```
' Examination Grades
Dim grade As String
Private Sub Compute_Click( )
grade=txtgrade.Text
Select Case grade
Case "A"
Label1.Text="High Distinction"
Case "A-"
Label1.Text="Distinction"
Case "B"
Label1.Text="Credit"
Case "C"
Label1.Text="Pass"
Case Else
Label1.Text="Fail"
End Select

End Sub
``` | In this example, you can use the keyword Is together with the comparison operators.<br><br>```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

'Examination Marks

Dim mark As Single
mark = mrk.Text
Select Case mark
Case Is >= 85
Label1.Text= "Excellence"
Case Is >= 70
Label2.Text= "Good"
Case Is >= 60
Label3.Text = "Above Average"
Case Is >= 50
Label4.Text= "Average"
Case Else
Label5.Text = "Need to work harder"
End Select

End Sub
``` |

Example 10.3	Example 10.4
| Example 10.2 can be rewritten as follows:<br><br>```
Private Sub Button1_Click(ByVal sender
As System.Object, ByVal e As
System.EventArgs) Handles
Button1.Click

'Examination Marks

Dim mark As Single
mark = Textbox1.Text
``` | Grades in high school are usually presented with a single capital letter such as A, B, C, D or E. The grades can be computed as follow:<br><br>```
Private Sub Button1_Click(ByVal sender
As System.Object, ByVal e As
System.EventArgs) Handles
Button1.Click
``` |

Select Case mark Case 0 to 49 Label1.Text = "Need to work harder" Case 50 to 59 Label1.Text = "Average" s Case 60 to 69 Label1.Text= "Above Average" Case 70 to 84 Label1.Text = "Good" Case 85 to 100 Label1.Text= "Excellence" Case Else Label1.Text= "Wrong entry, please reenter the mark" End Select End Sub	'Examination Marks Dim mark As Single mark = TextBox1.Text Select Case mark Case 0 To 49 Label1.Text = "E" Case 50 To 59 Label1.Text = "D" Case 60 To 69 Label1.Text = "C" Case 70 To 79 Label1.Text = "B" Case 80 To 100 Label1.Text = "A" Case Else Label1.Text = "Error, please reenter the mark" End Select End Sub

The output of Example 10.4

Lesson 11- Looping

Visual Basic 2010 allows a procedure to be repeated as many times as long as the processor and memory could support. This is generally called looping . Looping is required when we need to process something repetitively until a certain condition is met. For example, we can design a program that adds a series of numbers until the sum exceeds a certain value, or a program that asks the user to enter data repeatedly until he/she keys in the word 'Finish'. In Visual Basic 2010, we have three types of Loops, they are the For…..Next loop, the Do loop. and the While…..End while loop

11.1 For….Next Loop

The format is:

For counter=startNumber to endNumber (Step increment)
One or more Visual Basic 2010 statements
Next

To exit a For…..Next Loop, you can place the Exit For statement within the loop; and it is normally used together with the If….Then…..sstatement. For its application, you can refer to example 11.1 d.

Example 11.1 a	Example 11.1b
Dim counter as Integer For counter=1 to 10 ListBox1.Items.Add (counter) Next * The program will enter number 1 to 10 into the list box.	Dim counter , sum As Integer For counter=1 to 100 step 10 sum+=counter ListBox1.Items.Add (sum) Next * The program will calculate the sum of the numbers as follows: sum=0+10+20+30+40+……

Example 11.1c	Example 11.1d

Dim counter, sum As Integer
sum = 1000
For counter = 100 To 5 Step -5
sum – = counter
ListBox1.Items.Add(sum)
	Dim n as Integer
	For n=1 to 10
	If n>6 then
	Exit For
	End If

Next
*Notice that increment can be negative.
 	
The program will compute the
subtraction as follow:
1000-100-95-90-……….
	Else
	ListBox1.Items.Add (n)
	Next
	End If
	Next
	
	The process will stop when n is greater than 6

11.2 Do Loop

The Do Loop formats are

a) Do While condition
Block of one or more Visual Basic 2010 statements
Loop

b) Do
Block of one or more Visual Basic 2010 statements
Loop While condition

c) Do Until condition
Block of one or more Visual Basic 2010 statements
Loop

d) Do
Block of one or more Visual Basic 2010 statements
Loop Until condition

* Exiting the Loop

Sometime we need exit to exit a loop prematurely because of a certain condition is fulfilled. The syntax to use is known as Exit Do. Let's examine the following examples

Example 11.2(a)	Example 11.2(b)
Do while counter <=1000 TextBox1.Text=counter counter +=1 Loop * The above example will keep on adding	Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click Dim sum, n As Integer ListBox1.Items.Add("n" & vbTab & "Sum")

until counter >1000.
 	
   The above example can be rewritten as
 	
Do
TextBox1.Text=counter
counter+=1
Loop until counter>1000
	ListBox1.Items.Add("——————————-")
	Do
	n += 1
	sum += n
	ListBox1.Items.Add(n & vbTab & sum)
	If n = 100 Then
	Exit Do
	End If
	Loop
	
	End Sub

Lesson 12- Functions Part 1

A function is similar to a normal procedure but the main purpose of the function is to accept a certain input and return a value which is passed on to the main program to finish the execution. There are two types of functions in Visual Basic 2010, the built-in functions (or internal functions) and the functions created by the programmers.

The general syntax of a function is
FunctionName (arguments)
The arguments are values that are passed on to the function.In this lesson, we are going to learn two very basic but useful internal functions of Visual Basic 2010 , i.e. the MsgBox( ) and InputBox ( ) functions.

12.1 MsgBox ( ) Function

The objective of MsgBox is to produce a pop-up message box and prompts the user to click on a command button before he or she can continues. This syntax is as follows:

yourMsg=MsgBox(Prompt, Style Value, Title)

The first argument, Prompt, will display the message in the message box. The Style Value will determine what type of command buttons appear on the message box, please refer to Table 12.1 for types of command button displayed. The Title argument will display the title of the message board.

Table 12.1: Style Values

Style Value	Named Constant	Buttons Displayed
0	vbOkOnly	Ok button
1	vbOkCancel	Ok and Cancel buttons
2	vbAbortRetryIgnore	Abort, Retry and Ignore buttons.
3	vbYesNoCancel	Yes, No and Cancel buttons
4	vbYesNo	Yes and No buttons
5	vbRetryCancel	Retry and Cancel buttons

We can use named constants in place of integers for the second argument to make the programs more readable. In fact, Visual Basic 2010 will automatically shows up a list of named constants where you can select one of them.

example: yourMsg=MsgBox("Click OK to Proceed", 1, "Startup Menu")

and yourMsg=Msg("Click OK to Proceed". vbOkCancel,"Startup Menu")

are the same.

yourMsg is a variable that holds values that are returned by the MsgBox ( ) function. The values are determined by the type of buttons being clicked by the users. It has to be declared as Integer data type in the procedure or in the general declaration section. Table 12.2 shows the values, the corresponding named constant and buttons.

Table 12.2 : Return Values and Command Buttons

Value	Named Constant	Button Clicked
1	vbOk	Ok button
2	vbCancel	Cancel button
3	vbAbort	Abort button
4	vbRetry	Retry button
5	vbIgnore	Ignore button
6	vbYes	Yes button
7	vbNo	No button

Example 12.1

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim testmsg As Integer
testmsg = MsgBox("Click to test", 1, "Test message")
If testmsg = 1 Then
MessageBox.Show("You have clicked the OK button")
Else
```

MessageBox.Show("You have clicked the Cancel button")
End If
End Sub

To make the message box looks more sophisticated, you can add an icon besides the message. There are four types of icons available in Visual Basic 2010 as shown in Table 12.3

Table 12.3: Message box with icon

Value	Named Constant	Icon
16	vbCritical	
32	vbQuestion	
48	vbExclamation	
64	vbInformation	

Example 12.2

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

Dim testMsg As Integer
testMsg = MsgBox("Click to Test", vbYesNoCancel + vbExclamation, "Test Message")
If testMsg = 6 Then
MessageBox.Show("You have clicked the yes button")
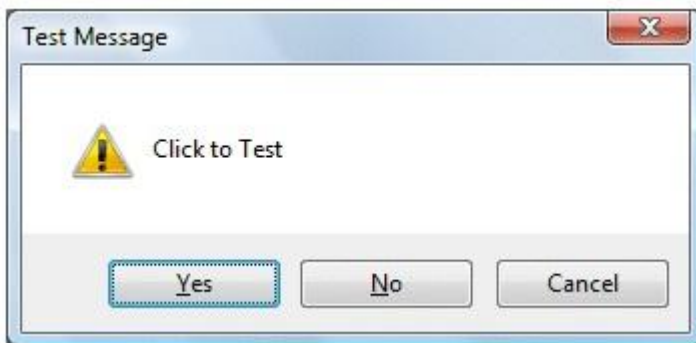ElseIf testMsg = 7 Then
MessageBox.Show("You have clicked the NO button")
Else
MessageBox.Show("You have clicked the Cancel button")
End If
End Sub

The first argument, Prompt, will display the message

12.2 The InputBox( ) Function

An InputBox( ) function will display a message box where the user can enter a value or a message in the form of text. In VB2005, you can use the following format:

myMessage=InputBox(Prompt, Title, default_text, x-position, y-position)

myMessage is a variant data type but typically it is declared as string, which accept the message input by the users. The arguments are explained as follows:

Prompt – The message displayed normally as a question asked.
Title – The title of the Input Box.
default-text – The default text that appears in the input field where users can use it as his intended input or he may change to the message he wish to enter.
x-position and y-position – the position or tthe coordinates of the input box.

However, the format won't work in Visual Basic 2010 because InputBox is considered a namespace. So, you need to key in the full reference to the Inputbox namespace, which is

Microsoft.VisualBasic.InputBox(Prompt, Title, default_text, x-position, y-position)

The parameters remain the same.

Example 12.3

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
Dim userMsg As String

userMsg = Microsoft.VisualBasic.InputBox("What is your message?", "Message Entry Form", "Enter your messge here", 500, 700)

If userMsg <> "" Then
MessageBox.Show(userMsg)
Else
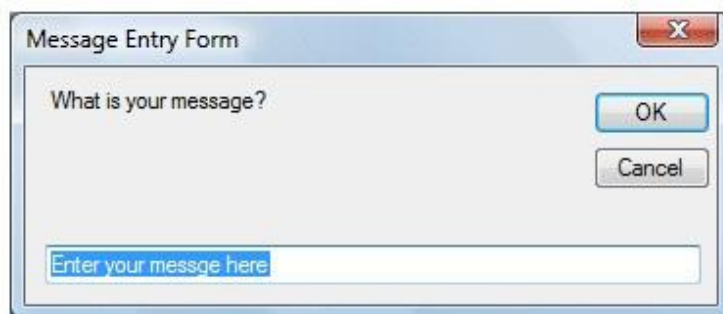MessageBox.Show("No Message")
End If
End Sub

The inputbox will appear as shown in the figure below when you press the command button

Lesson 13- Function Part II

We have learned the basic concept of functions in Visual Basic 2010 , in particular the usage of the MsgBox and InputBox functions in Lesson 12. I. In fact, I have already shown you a few string manipulation functions in Lesson 8, they are the Len function, the Left function and the Right Function. In this lesson, we will learn how to write codes involving other string manipulation functions in Visual Basic 2010.

13.1 The Mid Function

The Mid function is used to retrieve a part of text form a given phrase. The syntax of the Mid Function is

Mid(phrase, position,n)
where
phrase is the string from which a part of text is to be retrieved.
position is the starting position of the phrase from which the retrieving process begins.
n is the number of characters to retrieve.

Example 13.1:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim myPhrase As String
myPhrase = Microsoft.VisualBasic.InputBox("Enter your phrase")
Label1.Text = Mid(myPhrase, 2, 6)
End Sub
```

* In this example, when the user clicks the command button, an input box will pop up asking the user to input a phrase. After a phrase is entered and the OK button is pressed, the label will show the extracted text starting from position 2 of the phrase and the number of characters extracted is 6.

The diagrams are shown below:

13.2 The Right Function

The Right function extracts the right portion of a phrase. The syntax is

Microsoft.Visualbasic.Right ("Phrase", n)

Where n is the starting position from the right of the phase where the portion of the phrase is going to be extracted. For example:

Microsoft.Visualbasic.Right ("Visual Basic", 4) = asic

Example 13.2: The following code extracts the right portion any phrase entered by the user.

```
Private Sub Button1_Click (ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim myword As String
myword = TextBox1.Text
Label1.Text = Microsoft.VisualBasic.Right (myword, 4)
End Sub
```

13.3 The Left Function

The Left function extracts the left portion of a phrase. The syntax is

Microsoft.Visualbasic.Right ("Phrase", n)

Where n is the starting position from the left of the phase where the portion of the phrase is going to be extracted. For example:

Microsoft.Visualbasic.Left("Visual Basic", 4) = asic

Example 13.3: The following code extracts the left portion any phrase entered by the user.

```
Private Sub Button1_Click (ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim myword As String
myword = TextBox1.Text
Label1.Text = Microsoft.VisualBasic.Left (myword, 4)
End Sub
```

13.4 The Trim Function

The Trim function trims the empty spaces on both side of the phrase. The format is

Trim("Phrase")

.For example, Trim (" Visual Basic 2010 ") = Visual basic 2010

Example 13.4

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim myPhrase As String
myPhrase = Microsoft.VisualBasic.InputBox("Enter your phrase")
Label1.Text = Trim(myPhrase)
End Sub
```

13.5 The Ltrim Function

The Ltrim function trims the empty spaces of the left portion of the phrase. The syntax is
Ltrim("Phrase")
.For example,
Ltrim (" Visual Basic 2010 ")= Visual basic 2010

13.6 The Rtrim Function

The Rtrim function trims the empty spaces of the right portion of the phrase. The syntax is
Rtrim("Phrase")
.For example,
Rtrim ("Visual Basic ") = Visual Basic

13.7 The InStr function

The InStr function looks for a phrase that is embedded within the original phrase and returns the starting position of the embedded phrase. The syntax is

Instr (n, original phase, embedded phrase)

Where n is the position where the Instr function will begin to look for the embedded phrase.

For example

Instr(1, "Visual Basic 2010 ","Basic")=8

*The function returns a numeric value.

You can write a program code as shown below:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Label1.Text = InStr(1, "Visual Basic 2010", "Basic")
End Sub
```

13.8 The Ucase and the Lcase Functions

The Ucase function converts all the characters of a string to capital letters. On the other hand, the Lcase function converts all the characters of a string to small letters.

The format is

Microsoft.VisualBasic.UCase(Phrase)

Microsoft.VisualBasic.LCase(Phrase)

For example,

Microsoft.VisualBasic.Ucase("Visual Basic 2010") =VISUAL BASIC 2010

Microsoft.VisualBasic.Lcase("Visual Basic 2010") =visual basic 2010

13.9 The Chr and the Asc functions

The Chr function returns the string that corresponds to an ASCII code while the Asc function converts an ASCII character or symbol to the corresponding ASCII code. ASCII stands for "American Standard Code for Information Interchange". Altogether there are 255 ASCII codes and as many ASCII characters. Some of the characters may not be displayed as they may represent some actions such as the pressing of a key or produce a beep sound. The syntax of the Chr function is

Chr(charcode)

and the syntax of the Asc function is

Asc(Character)

The following are some examples:

Chr(65)=A, Chr(122)=z, Chr(37)=% ,

Asc("B")=66, Asc("&")=38

Lesson 14- Functions Part III- Math Functions

We have learned how to write programs in Visual Basic 2010 to perform arithmetic operations using standard mathematical operators. However, for more complex mathematical calculations, we need to use the built-in math functions in Visual Basic 2010. There are numerous built-in mathematical functions in Visual Basic 2010 which we will introduce them one by one.

14.1 The Abs function

The Abs function returns the absolute value of a given number.

The syntax is

Math. Abs (number)

* The Math keyword here indicates that the Abs function belong to the Math class. However, not all mathematical functions belong to the Math class.

14.2 The Exp function

The Exp of a number x is the exponential value of x, i.e. e^x . For example, Exp(1)=e=2.71828182

The syntax is Math.Exp (number)

Example:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

Dim num1, num2 As Single
num1 = TextBox1.Text
num2 = Math.Exp(num1)
Label1.Text = num2

End Sub
```

14.3 The Fix Function

The Fix function truncates the decimal part of a positive number and returns the largest integer smaller than the number. However, when the number is negative, it will return smallest integer larger than the number. For example, Fix(9.2)=9 but Fix(-9.4)=-9

Example:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

Dim num1, num2 As Single
num1 = TextBox1.Text
```

```
num2 = Fix(num1)
Label1.Text = num2

End Sub
```

14.4 The Int Function

The Int is a function that converts a number into an integer by truncating its decimal part and the resulting integer is the largest integer that is smaller than he number. For example

Int(2.4)=2, Int(6.9)=6 , Int(-5.7)=-6, Int(-99.8)=-100

14.5 The Log Function

The Log function is the function that returns the natural logarithm of a number. For example, Log(10)=2.302585

Example:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click

Dim num1, num2 As Single
num1 = TextBox1.Text
num2 = Math.Log(num1)
Label1.Text = num2

End Sub
```

* The logarithm of num1 will be displayed on label1

14.6 The Rnd( ) Function

The Rnd is very useful when we deal with the concept of chance and probability. The Rnd function returns a random value between 0 and 1. Random numbers in their original form are not very useful in programming until we convert them to integers. For example, if we need to obtain a random output of 6 integers ranging from 1 to 6, which makes the program behave like a virtual dice, we need to convert the random numbers to integers using the formula Int(Rnd*6)+1.

Example:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim num as integer

Randomize( )

Num=Int(Rnd()*6)+1
```

Label1.Text=Num

End Sub

In this example, Int(Rnd*6) will generate a random integer between 0 and 5 because the function Int truncates the decimal part of the random number and returns an integer. After adding 1, you will get a random number between 1 and 6 every time you click the command button. For example, let say the random number generated is 0.98, after multiplying it by 6, it becomes 5.88, and using the integer function Int(5.88) will convert the number to 5; and after adding 1 you will get 6.

14.7 The Round Function

The Round function is the function that rounds up a number to a certain number of decimal places. The Format is Round (n, m) which means to round a number n to m decimal places. For example, Math.Round (7.2567, 2) =7.26

Example

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim num1, num2 As Single
num1 = TextBox1.Text
num2 = Math.Round(num1, 2)
Label1.Text = num2

End Sub
```

* The Math keyword here indicates that the Round function belong to the Math class.

Lesson 15 – Functions Part IV- Formatting Functions

The Format function in Visual Basic 2010 is a very powerful formatting function which can display the numeric values in various forms. There are two types of Format functions in Visual Basic 2010, one of them is the built-in or predefined format while another one can be defined by the users.

(i) The syntax of the predefined Format function is
Format (n, "style argument")
where n is a number and the list of style arguments is given in Table 15.1.

Table 15.1 List of style arguments

Style argument	Explanation	Example
General Number	To display the number without having separators between thousands.	Format(8972.234, "General Number")=8972.234
Fixed	To display the number without having separators between thousands and rounds it up to two decimal places.	Format(8972.2, "Fixed")=8972.23
Standard	To display the number with separators or separators between thousands and rounds it up to two decimal places.	Format(6648972.265, "Standard")= 6,648,972.27
Currency	To display the number with the dollar sign in front, has separators between thousands as well as rounding it up to two decimal places.	Format(6648972.265, "Currency")= $6,648,972.27
Percent	Converts the number to the percentage form and displays a % sign and rounds it up to two decimal places.	Format(0.56324, "Percent")=56.32%

Example 15.1

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click, Button5.Click, Button4.Click, Button3.Click
Label1.Text = Format(8972.234, "General Number")
Label2.Text = Format(8972.2, "Fixed")
Label3.Text = Format(6648972.265, "Standard")
Label4.Text = Format(6648972.265, "Currency")
Label5.Text = Format(0.56324, "Percent")
End Sub

The Output window is shown below:

(ii) The syntax of the user-defined Format function is

Format (n, "user's format")

Although it is known as user-defined format, we still need to follows certain formatting styles. Examples of user-defined formatting style are listed in Table 15.2

Table15.2: User-Defined format

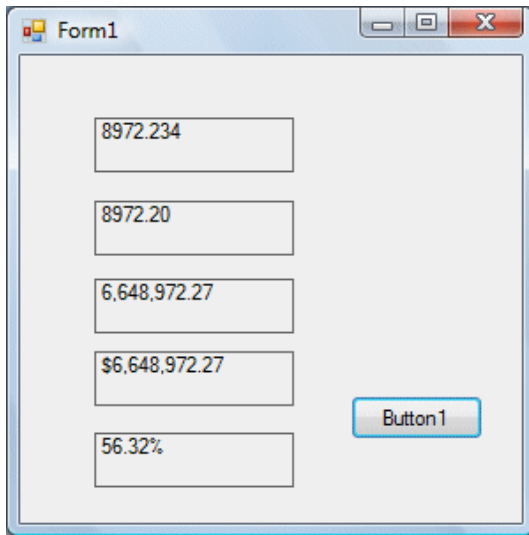| Example	Explanation	Output
Format(781234.57,"0")	Rounds to whole number without separators between thousands.	781235
Format(781234.57,"0.0")	Rounds to 1 decimal place without separators between thousands.	781234.6
Format(781234.576,"0.00")	Rounds to 2 decimal places without separators between thousands.	781234.58
Format(781234.576,"#,##0.00")	Rounds to 2 decimal places with separators between thousands.	781,234.58
Format(781234.576,"$#,##0.00")	Shows dollar sign and rounds to 2 decimal places with separators between thousands.	$781,234.58
Format(0.576,"0%")	Converts to percentage form without decimal places.	58%
Format(0.5768,"0.00%")	Converts to percentage form with 2 decimal places.	57.68%

Example 15.2

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click, Button5.Click, Button4.Click, Button3.Click
Label1.Text = Format(8972.234, "0.0")
Label2.Text = Format(8972.2345, "0.00")
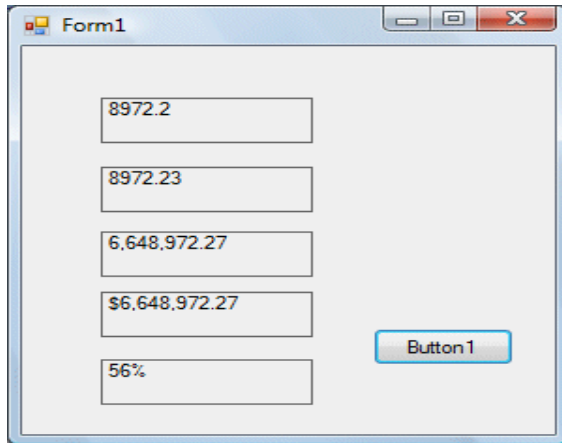Label3.Text = Format(6648972.265, "#,##0.00")
Label4.Text = Format(6648972.265, "$#,##0.00")

Label5.Text = Format(0.56324, "0%")

End Sub

The Output window is shown below:

Lesson 16 – Functions Part V- Formatting Date and Time

16.1 Formatting Date and time using predefined formats

In Visual Basic 2010, date and time can be formatted using predefined formats and also user-defined formats. The predefined formats of date and time are shown in Table 16.1.

Table 16.1 Predefined formats of date and time

Format	Explanation
Format (Now, "General date")	Formats the current date and time.
Format (Now, "Long Date")	Displays the current date in long format.
Format (Now, "Short date")	Displays current date in short format
Format (Now, "Long Time")	Display the current time in long format.
Format (Now, "Short Time")	Display the current time in short format.

* Instead of "General date", you can also use the abbreviated format "G" , i.e. Format (Now, "G"). And for "Long Time", you can use the abbreviated format "T". As for "Short Time", you may use the abbreviated format "t"
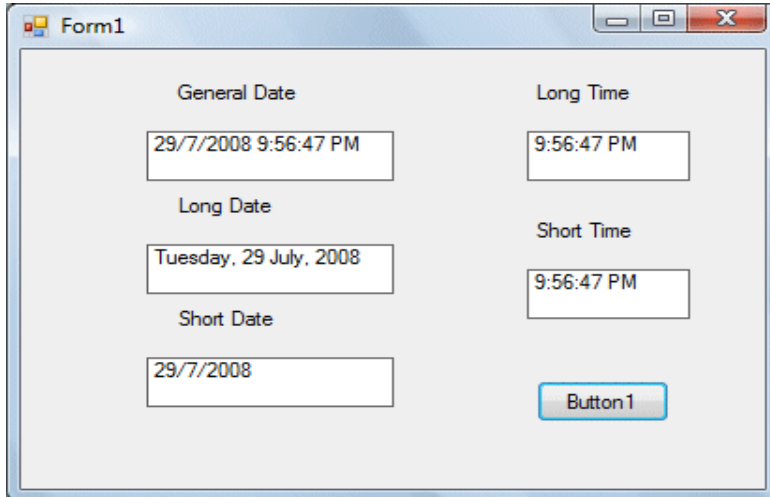
Example 16.1

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Label1.Text = Format(Now, "General Date")

```
Label2.Text = Format(Now, "Long Date")
Label3.Text = Format(Now, "short Date")
Label4.Text = Format(Now, "Long Time")
Label5.Text = Format(Now, "Short Time")
End Sub
```

The output is shown in the diagram below:

16.2 Formatting Date and time using user-defined formats

Beside using the predefined formats, you can also use the user-defined formatting functions. The general syntax of a user-defined for date/time is

Format (expression,style)

Table 16.2 Some of the user-defined format functions for date and time

Format	Explanation
Format (Now, "M")	Displays current month and date
Format (Now, "MM")	Displays current month in double digits.
Format (Now, "MMM")	Displays abbreviated name of the current month
Format (Now, "MMMM")	Displays full name of the current month.
Format (Now, "dd/MM/yyyy")	Displays current date in the day/month/year format.
Format (Now, "MMM,d,yyyy")	Displays current date in the Month, Day, Year Format
Format (Now, "h:mm:ss tt")	Dispalys current time in hour:minute:second format and show am/pm
Format (Now, "MM/dd/yyyy h:mm:ss")	Dispalys current date and time in hour:minute:second format

Example 16.2

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click, Button2.Click, Button3.Click

Label1.Text = Format(Now, "M")
Label2.Text = Format(Now, "MM")
```
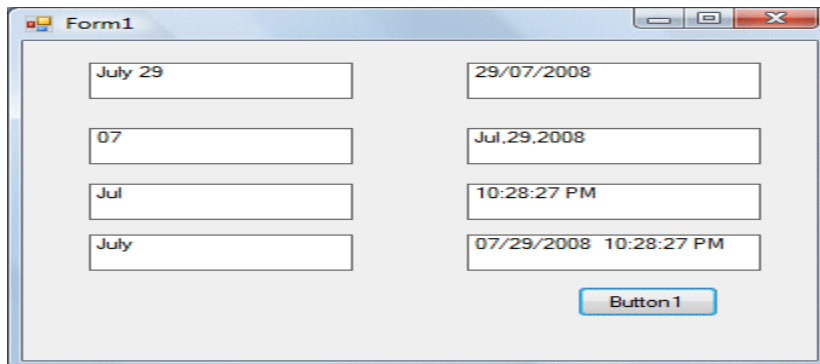
Label3.Text = Format(Now, "MMM")
Label4.Text = Format(Now, "MMMM")
Label5.Text = Format(Now, "dd/MM/yyyy")
Label6.Text = Format(Now, "MMM,d,yyyy")
Label7.Text = Format(Now, "h:mm:ss tt")
Label8.Text = Format(Now, "MM/dd/yyyy h:mm:ss tt")
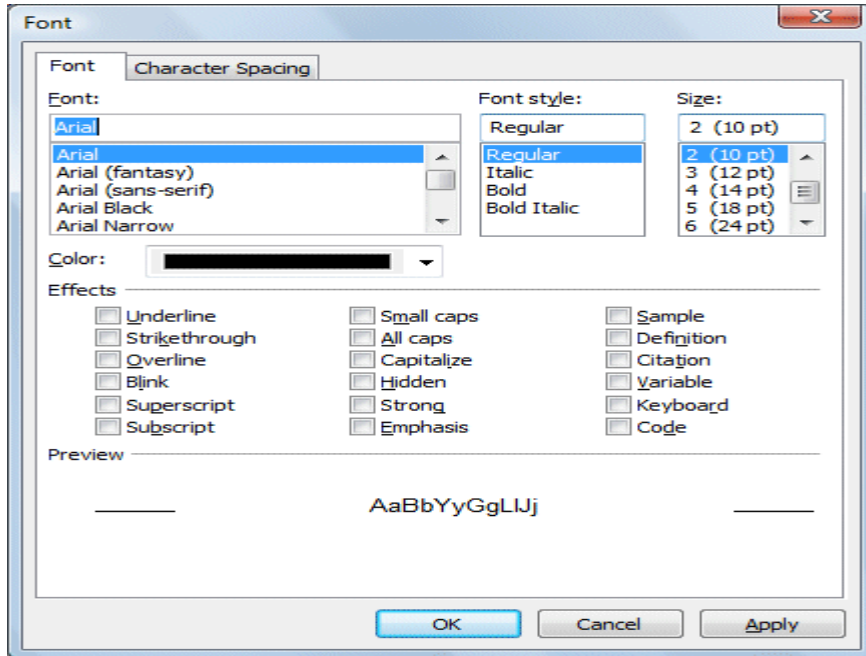
End Sub

The output is shown in the diagram below:

Lesson 17 – Using Check Box

The Check box is a very useful control in Visual Basic 2010. It allows the user to select one or more items by checking the check box/check boxes concerned. For example, in the Font dialog box of any Microsoft Text editor like FrontPage, there are many check boxes under the Effects section such as that shown in the figure below. The user can choose underline, subscript, small caps, superscript, blink and etc. In Visual Basic 2010, you may create a shopping cart where the user can click on check boxes that correspond to the items they intend to buy, and the total payment can be computed at the same time as shown in Example 17.1.

Example 17.1:Shopping Cart

The program code for shopping cart:

Private Sub BtnCalculate_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnCalculate.Click

```vbnet
Const LX As Integer = 100
Const BN As Integer = 500
Const SD As Integer = 200
Const HD As Integer = 80
Const HM As Integer = 300
Const AM As Integer = 150
Dim sum As Integer

If CheckBox1.Checked = True Then
sum += LX
End If

If CheckBox2.Checked = True Then
sum += BN
End If

If CheckBox3.Checked = True Then
sum += SD
End If
If CheckBox4.Checked = True Then
sum += HD
End If

If CheckBox5.Checked = True Then
sum += HM
End If

If CheckBox6.Checked = True Then
sum += AM
End If
Label5.Text = sum.ToString(“c”)
```
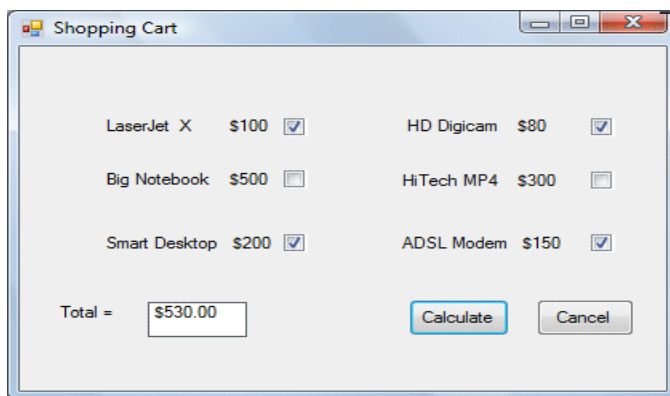
Here is another example

Example 17.2

```vbnet
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Const large As Integer = 10.0
Const medium As Integer = 8
Const small As Integer = 5
Dim sum As Integer
If CheckBox1.Checked = True Then
sum += large
End If
If CheckBox2.Checked = True Then
sum += medium
End If
```

```
If CheckBox3.Checked = True Then
sum += small
End If
Label5.Text = sum.ToString("c")
```

Example 17.3

In this example, the user can enter text into a text box and format the font using the three check boxes that represent bold, italic and underline.

The code is as follow:

```
Private Sub CheckBox1_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CheckBox1.CheckedChanged
If CheckBox1.Checked Then
TextBox1.Font = New Font(TextBox1.Font, TextBox1.Font.Style Or FontStyle.Bold)
Else
TextBox1.Font = New Font(TextBox1.Font, TextBox1.Font.Style And Not FontStyle.Bold)

End If
End Sub

Private Sub CheckBox2_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CheckBox2.CheckedChanged
If CheckBox2.Checked Then
TextBox1.Font = New Font(TextBox1.Font, TextBox1.Font.Style Or FontStyle.Italic)
Else
TextBox1.Font = New Font(TextBox1.Font, TextBox1.Font.Style And Not FontStyle.Italic)

End If
End Sub

Private Sub CheckBox3_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CheckBox3.CheckedChanged
If CheckBox2.Checked Then
TextBox1.Font = New Font(TextBox1.Font, TextBox1.Font.Style Or FontStyle.Underline)
```

Else
TextBox1.Font = New Font(TextBox1.Font, TextBox1.Font.Style And Not FontStyle.Underline)
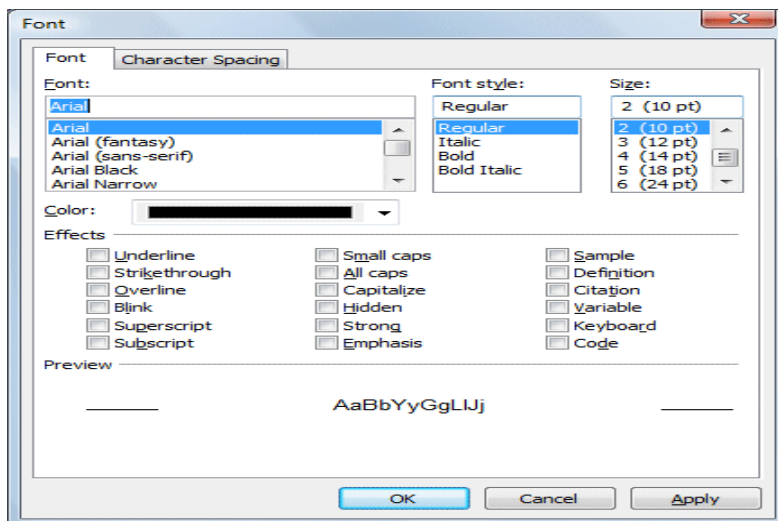
End If
End Sub

* The above program uses the CheckedChanged event to respond to the user selection by checking a particular checkbox, it is similar to the click event. The statement

TextBox1.Font = New Font(TextBox1.Font, TextBox1.Font.Style Or FontStyle.Italic)

will retain the original font type but change it to italic font style.

TextBox1.Font = New Font(TextBox1.Font, TextBox1.Font.Style And Not FontStyle.Italic)

will also retain the original font type but change it to regular font style. (The other statements employ the same logic)

Format	Explanation
Format (Now, "General date")	Formats the current date and time.
Format (Now, "Long Date")	Displays the current date in long format.
Format (Now, "Short date")	Displays current date in short format
Format (Now, "Long Time")	Display the current time in long format.
Format (Now, "Short Time")	Display the current time in short format.

* Instead of "General date", you can also use the abbreviated format "G" , i.e. Format (Now, "G"). And for "Long Time", you can use the abbreviated format "T". As for "Short Time", you may use the abbreviated format "t"

Lesson 18 – Using Radio Button

The radio button is also a very useful control in Visual Basic 2010. However, it operates differently from the check boxes. While the checkboxes work independently and allow the user to select one or more items , radio buttons are mutually exclusive, which means the user can only choose one item only out of a number of choices. Here is an example which allows the user to select one color only.
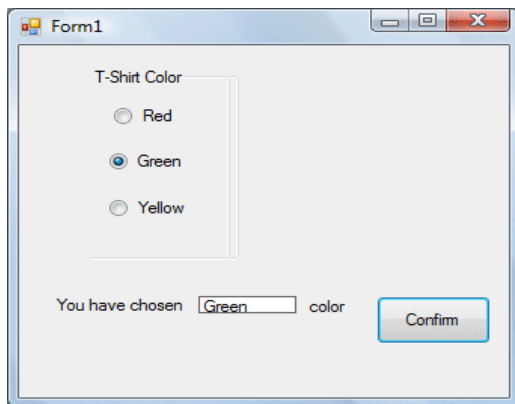
The Code:

```
Dim strColor As String

Private Sub RadioButton8_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioButton8.CheckedChanged
strColor = "Red"
End Sub

Private Sub RadioButton7_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioButton7.CheckedChanged
strColor = "Green"
End Sub

Private Sub RadioYellow_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioYellow.CheckedChanged
strColor = "Yellow"
End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Label2.Text = strColor
End Sub
```

Although the user may only select one item at a time, he may make more than one selection if those items belong to different categories. For example, the user wishes to choose T-shirt size and color, he needs to select one color and one size, which means one selection in each category. This is easily achieved in Visual Basic 2010 by using the Groupbox control under the

containers categories. After inserting the Groupbox into the form, you can proceed to insert the radio buttons into the Groupbox. Only the radio buttons inside the Groupbox are mutually exclusive, they are not mutually exclusive with the radio buttons outside the Groupbox. In Example 18.2, the user can select one color and one size of the T-shirt.

Example 18.2

The code:

```
Dim strColor As String
Dim strSize As String

Private Sub RadioButton8_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioButton8.CheckedChanged
strColor = "Red"
End Sub

Private Sub RadioButton7_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioButton7.CheckedChanged
strColor = "Green"
End Sub

Private Sub RadioYellow_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioYellow.CheckedChanged
strColor = "Yellow"
End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Label2.Text = strColor
Label4.Text = strSize
End Sub

Private Sub RadioXL_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioXL.CheckedChanged
```
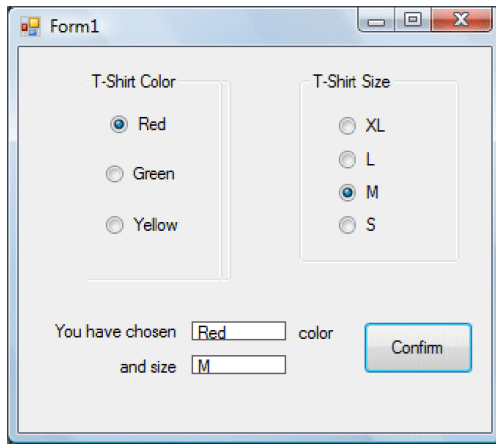
```
strSize = "XL"
End Sub

Private Sub RadioL_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioL.CheckedChanged
strSize = "L"
End Sub

Private Sub RadioM_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioM.CheckedChanged
strSize = "M"
End Sub

Private Sub RadioS_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioS.CheckedChanged
strSize = "S"
End Sub
```

Lesson 19 – Creating A Simple Web Browser

Basically everyone likes to navigate the Internet using commercially produced web browsers such the Internet Explorer produced by Microsoft or those open source browsers designed by the experts such FireFox , Opera and the latest Chrome created by Google. However, isn't it cool that if you can create your very own web browser that you can customize to your own taste ? Yes, you can do that in Visual Basic 2010, and pretty easy too. In this chapter, I will show you how to create a simple web browser and get it running in a few minutes.

First of all, start a new project in Visual Basic 2010 and name it with any name you like. Here I am just using the name Web Browser. Change the name of Form1 to MyWebBrowser and the text property to Web Browser and set its size property to 640,480. Next, you need to add an engine so that your web browser can connect to the Internet, and this very engine is the WebBrowser control, located on the Toolbox on the left side, set the size property to 600,400. Next, drag a text box and place it at the top of the WebBrowser control, this will be the address bar where the user can enter the URL. Lastly, place a command button beside the text box and label it as Go and change its name to Go as well. The design interface is shown below:
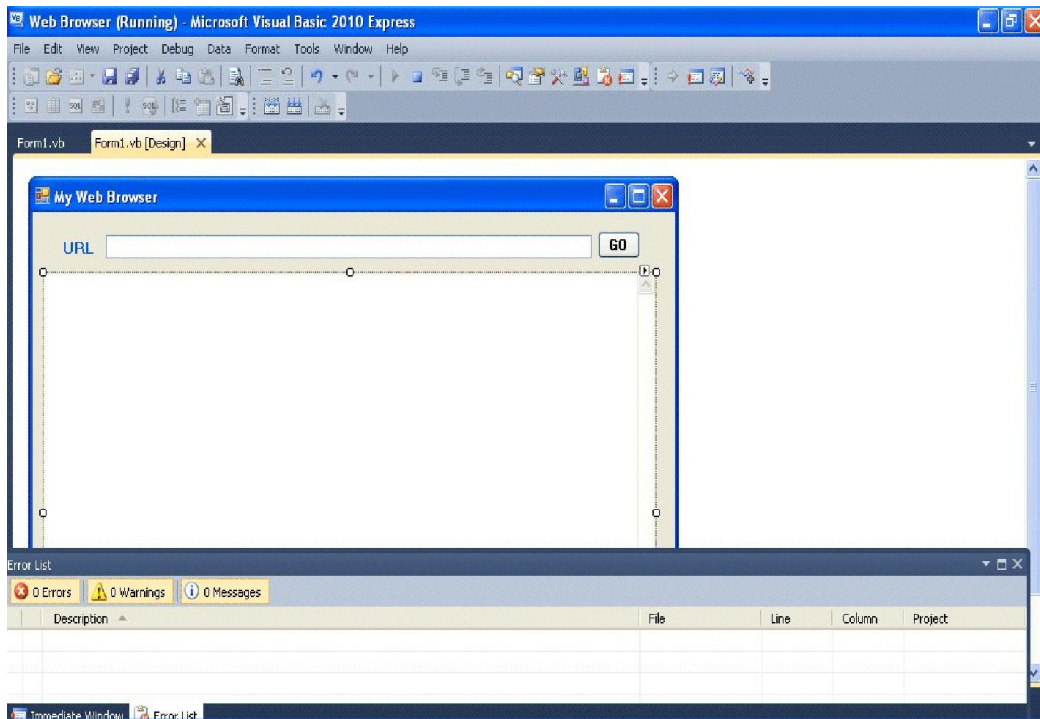
The code for the web browser is surprisingly simple, it is only a single line code! Double click on the Go button and key in the following code:

Public Class Form1

Private Sub Go_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

MyWebBrowser.Navigate(TextBox1.Text)

End Sub

End Class

Now run the the program , type in any URL and click the Go button. You will be able to browse any web page you want.

Figure 19.2: The Runtime Interface

Lesson 20 – Errors Handling

20.1 Introduction

Error handling is an essential procedure in Visual Basic 2010 programming because it helps make a program error-free. An error-free program can run smoothly and efficiently, and the user does not have to face all sorts of problems such as program crashes or system hangs.

Errors often occur due to incorrect input from the user. For example, the user might make the mistake of attempting to enter text (string) to a box that is designed to handle only numeric values such as the weight of a person, the computer will not be able to perform arithmetic calculation for text therefore will create an error. These errors are known as synchronous errors.

Therefore a good programmer should be more alert to the parts of program that could trigger errors and should write errors handling code to help the user in managing the errors. Writing errors handling code is a good practice for Visual Basic programmers, so do not try to finish a program fast by omitting the errors handling code. However, there should not be too many errors handling code in the program as it create problems for the programmer to maintain and troubleshoot the program later. VB2010 has improved a lot in its built-in errors handling capabilities compared to Visual Basic 6. For example, when the user attempts to divide a number by zero, Vb2010 will not return an error message but gives the 'infinity' as the answer (although this is mathematically incorrect, because it should be undefined)

20.2 Using On Error GoTo Syantax

Visual Basic 2010 still supports the VB6 errors handling syntax, that is the On Error GoTo program_label structure. Although it has a more advanced error handling method, we shall deal with that later.We shall now learn how to write errors handling code in VB2010. The syntax for errors handling is

On Error GoTo program_label

where program_label is the section of code that is designed by the programmer to handle the error committed by the user. Once an error is detected, the program will jump to the program_label section for error handling.

Example 20.1: Division by Zero

In this example, we will deal with the error of entering non-numeric data into the textboxes that suppose to hold numeric values. The program_label here is error_hanldler. when the user enter a non-numeric values into the textboxes, the error message will display the text"One of the entries is not a number! Try again!". If no error occurs, it will display the correct answer. Try it out yourself.

The Code

```
Public Class Form1
Private Sub CmdCalculate_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles CmdCalculate.Click
Lbl_ErrorMsg.Visible = False
Dim firstNum, secondNum As Double
On Error GoTo error_handler
firstNum = Txt_FirstNumber.Text
secondNum = Txt_SecondNumber.Text
Lbl_Answer.Text = firstNum / secondNum
Exit Sub 'To prevent error handling even the inputs are valid
error_handler:
Lbl_Answer.Text = "Error"
Lbl_ErrorMsg.Visible = True
Lbl_ErrorMsg.Text = " One of the entries is not a number! Try again!"
End Sub
End Class
```

The Output
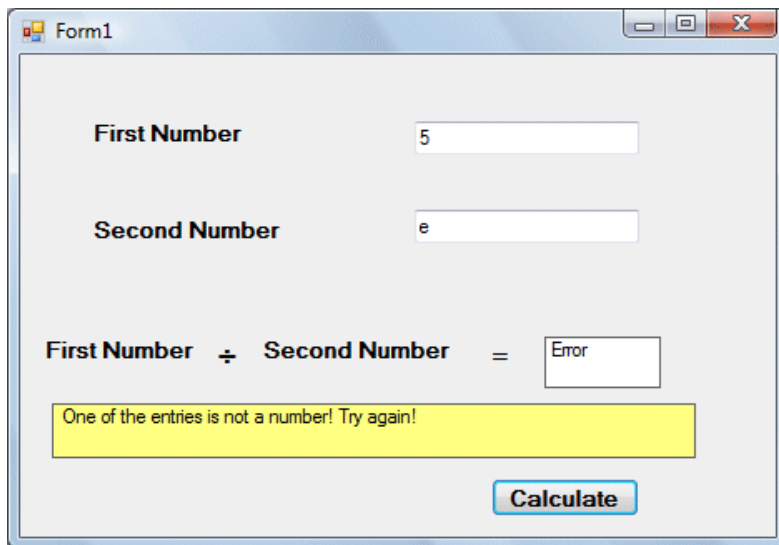
20.3 Errors Handling using Try…..Catch….End Try Structure

VB2010 has adopted a new approach in handling errors, or rather exceptions handling. It is supposed to be more efficient than the old On Error Goto method, where it can handles various types of errors within the Try…Catch…End Try structure.

The structure looks like this

Try

statements

Catch exception_variable as Exception

statements to deal with exceptions

End Try

Example 20.2

This is a modification of Example 20.1. Instead of using On Error GoTo method, we use the Try…Catch…End Try method. In this example, the Catch statement will catch the exception when the user enters a non-numeric data and return the error message. If there is no exception, there will not any action from the Catch statement and the program returns the correct answer.

The code

Public Class Form1

```vbnet
Private Sub CmdCalculate_Click(ByVal sender As System.Object, ByVal eAs System.EventArgs)
Handles CmdCalculate.Click

Lbl_ErrorMsg.Visible = False

Dim firstNum, secondNum, answer As Double

Try

firstNum = Txt_FirstNumber.Text

secondNum = Txt_SecondNumber.Text

answer = firstNum / secondNum

Lbl_Answer.Text = answer

Catch ex As Exception

Lbl_Answer.Text = "Error"

Lbl_ErrorMsg.Visible = True

Lbl_ErrorMsg.Text = " One of the entries is not a number! Try again!"

End Try

End Sub

End Class
```

The output

Lesson 21- Managing Graphics 1-Basic Concepts

21.1 Introduction to Graphics in Visual Basic 2010

Though Managing graphics in earlier versions of Visual Basic seem easier as they have built-in drawing tools, Visual Basic 2010 is much more versatile in handling graphics. For example, In Visual Basic 6, the drawing tools are included in the toolbox where the programmer just need to drag the shape controls into the form to create rectangle, square, ellipse,circle and more. However, its simplicity has the shortcomings, you don't have much choices in creating customized drawings.

Since Visual Basic evolved into a fully OOP language under the VB.net framework, shape controls are

no longer available. Now the programmer needs to write code to create various shapes and drawings.

Even though the learning curve is steeper, the programmer can write powerful code to create all kinds

of graphics. You can even design your own controls

Visual Basic 2010 offers various graphics capabilities that enable programmers to write code that can draw all kinds of shapes and even fonts. In this lesson,you will learn how to write code to draw lines and shapes on the VB interface.

21.1 Creating the Graphics Object

Before you can draw anything on a form, you need to create the Graphics object in vb2010. A graphics object is created using a CreateGraphics() method. You can create a graphics object that draw to the form itself or a control. For example, if you wish to draw to the form, you can use the following statement:

Dim myGraphics As Graphics =me.CreateGraphics

*Always use Dim to define the object. Using me instead of Form1 because it is not allowed in Visual Basic 2010.

Or if you want the graphics object to draw to a picturebox, you can write the following statement:

Dim myGraphics As Graphics = PictureBox1.CreateGraphics

You can also use the text box as a drawing surface, the statement is:

Dim myGraphics As Graphics = TextBox1.CreateGraphics

The Graphics object that is created does not draw anything on the screen until you call the methods of the Graphics object. In addition,you need to create the Pen object as the drawing tool. We will examine the code that can create a pen in the following section.

21.2 Creating a Pen

A Pen can be created using the following code:

myPen = New Pen(Brushes.DarkMagenta, 10)

where myPen is a Pen variable. You can use any variable name instead of myPen. The first argument of the pen object define the color of the drawing line and the second argument define the width of the drawing line.

You can also create a Pen using the following statement:

Dim myPen As Pen

myPen = New Pen(Drawing.Color.Blue, 5)

Where the first argument define the color(here is blue, you can change that to red or whatever color you want) and the second argument is the width of the drawing line.

Having created the Graphics and the Pen objects, you are now ready to draw graphics on the screen which we will show you in the following section.s

21.3 Drawing a Line

In this section, we will show you how to draw a straight line on the Form.

First of all, launch Visual basic 2010 Express. In the startup page, drag a button into the form. Double click on the button and key in the following code.

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

Dim myGraphics As Graphics = me.CreateGraphics

Dim myPen As Pen

myPen = New Pen(Brushes.DarkMagenta, 10)

myGraphics.DrawLine(myPen, 10, 10, 100, 10)

End Sub

The second created the Graphics object and the third and fourth line create the Pen object. The fifth draw a line on the Form using the DrawLine method. The first argument use the Pen object created by you, the second argument and the third arguments define the coordinate the starting point of the line, the fourth and the last arguments define the ending coordinate of the line. The general syntax of the Drawline argument is

object.DrawLine(Pen, x1, y1, x2, y2)

The output of the program is shown below:

Lesson 22 – Managing Graphics -Drawing Rectangle

We have learned how to create the Graphics and the Pen objects to draw straight lines in Visual Basic 2010. Now we shall learn how to draw various shapes such as rectangle, square,ellipse and circle.

22.1 Creating Rectangles

To draw a rectangle on the screen in Visual Basic 2010, there are two ways:

(i)The first way is to draw a rectangle directly using the DrawRectangle method by specifying its upper-left corner's coordinate and it width and height. You also need to create a Graphics and a Pen object to handle the actual drawing. The method to draw the rectangle is DrawRectangle .

The syntax is:

myGrapphics.DrawRectangle(myPen, X, Y, width, height)

Where myGraphics is the variable name of the Graphics object and myPen is the variable name of the Pen object created by you. You can use any valid and meaningful variable names. X, Y is the coordinate of the upper left corner of the rectangle while width and height are self-explanatory, i.e, the width and height of the rectangle.

The sample code is shown below:

Dim myPen As Pen

myPen = New Pen(Drawing.Color.Blue, 5)

Dim myGraphics As Graphics = Me.CreateGraphics

myGraphics.DrawRectangle(myPen, 0, 0, 100, 50)

(ii) The second way is to create a rectangle object first and then draw this triangle using the DrawRectangle method. The syntax is as shown below:

myGraphics.DrawRectangle(myPen,myRectangle) where myRectangle is the rectangle object created by you, the user.

The code to create a rectangtle object is as shown below:

Dim myRectangle As New Rectangle

myRect.X = 10

myRect.Y = 10

myRect.Width = 100

myRect.Height = 50

You can also create a rectangle object using a one-line code as follows:

Dim myRectangle As New Rectangle(X,Y,width, height)

and the code to draw the above rectange is

myGraphics.DrawRectangle(myPen, myRectangle)

The sample code is shown below:

22.2 Customizing Line Style of the Pen Object

The shape we draw so far are drawn with solid line, we can actually customize the line style of the Pen object so that we have dotted line, line consisting of dashes and more. For example, the syntax to draw with dotted line is shown below:

myPen.DashStyle=Drawing.Drawing2D.DashStyle.Dot

Where the last argument Dot specifies a particular line DashStyle value, a line that makes up of dots here. The following code draws a rectangle with red dotted line.
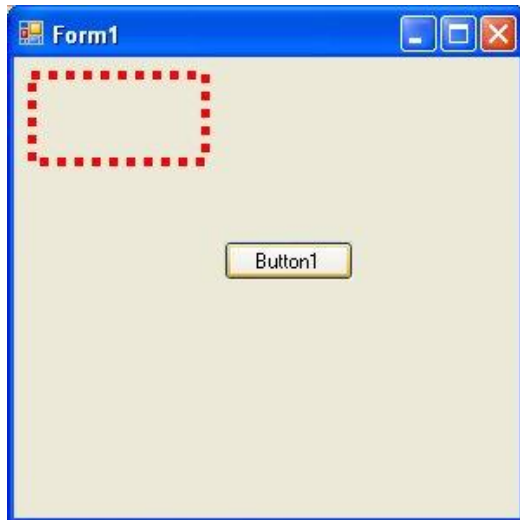
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

Dim myPen As Pen

myPen = New Pen(Drawing.Color.Red, 5)

```
Dim myGraphics As Graphics = Me.CreateGraphics

myPen.DashStyle = Drawing.Drawing2D.DashStyle.Dot

myGraphics.DrawRectangle(myPen, 10, 10, 100, 50)

End Sub
```

The output image is shown below:

The possible values of the line DashStyle of the Pen are listed in the table below:

DashStyle Value	Line Style
Dot	Line consists of dots
Dash	Line consists of dashes
DashDot	Line consists of alternating dashes and dots
DashDotDot	Line consists of alternating dashes and double dots
Solid	Solid line
Custom	Custom line style

Lesson 23 – Managing Graphics -Drawing Ellipse and Circle

We have learned how to draw rectangles with various line styles in Visual Basic 2010. Now we shall learn how to draw ellipse and circle.

23.1 Drawing Ellipse

First of all we need to understand the principle behind drawing an ellipse in Visual Basic 2010. The basic structure of most shapes is a rectangle, ellipse is no exception. Ellipse is an oval shape that is bounded by a rectangle, as shown below:

Therefore, we need to create a Rectangle object before we can draw an ellipse. This rectangle serves as a bounding rectangle for the ellipse. We still need to use the DrawEllipse method to complete the job. On the other hand, we can alaso draw an ellipse with the DrawEllipse method without first creating a rectangle. We shall illustrates both ways.

In the first method, let say you have created a rectangle object known as myRectangle and a pen object as myPen, then you can draw an ellipse using the following statement:

myGraphics.DrawEllipse(myPen, myRectangle)

* Assume you have also already created the Graphics object myGraphics.

The following is an example of the full code:

Example 23.1(a)

Dim myPen As Pen

myPen = New Pen(Drawing.Color.Blue, 5)

Dim myGraphics As Graphics = Me.CreateGraphics
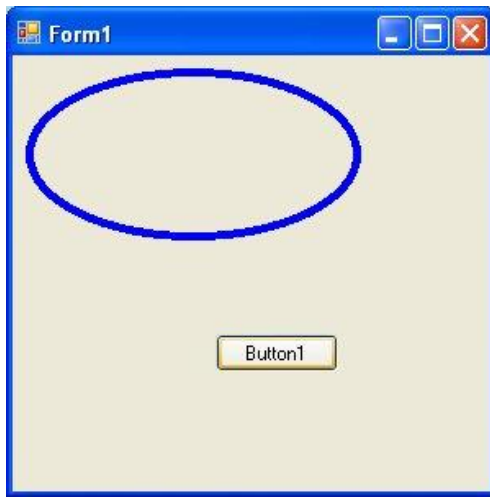
Dim myRectangle As New Rectangle

myRectangle.X = 10

myRectangle.Y = 10

myRectangle.Width = 200

myRectangle.Height = 100

myGraphics.DrawEllipse(myPen, myRectangle)

The output image is shown in the following diagram:

The second method is using the DrawEllipse method without creating a rectangle object. Offcourse you still have to create the Graphics and the Pen objects. The syntax is:

myGraphics.DrawEllipse(myPen, X,Y,Width, Height)

Where (X,Y) are the coordinates of the upper left corner of the bounding rectangle, width is the width of the ellipse and height is the height of the ellipse.

The following is an example of the full code:

Example 23.1(b)

Dim myPen As Pen

myPen = New Pen(Drawing.Color.Blue, 5)

Dim myGraphics As Graphics = Me.CreateGraphics

myGraphics.DrawEllipse(myPen, 10, 10, 200, 100)

23.2 Drawing a Circle

After you have learned how to draw an ellipse, drawing a circle becomes very simple. We use

exactly the same methods used in the preceding section but modify the width and height so

that they are of the same values.

The following examples draw the same circle.

Example 23.2(a)

Dim myPen As Pen

myPen = New Pen(Drawing.Color.Blue, 5)

Dim myGraphics As Graphics = Me.CreateGraphics

Dim myRectangle As New Rectangle

myRectangle.X = 10

myRectangle.Y = 10

myRectangle.Width = 100

myRectangle.Height = 100

myGraphics.DrawEllipse(myPen, myRectangle)

Example 23.2(b)

Dim myPen As Pen

myPen = New Pen(Drawing.Color.Blue, 5)

Dim myGraphics As Graphics = Me.CreateGraphics

myGraphics.DrawEllipse(myPen, 10, 10, 100, 100)

The output image is show below:
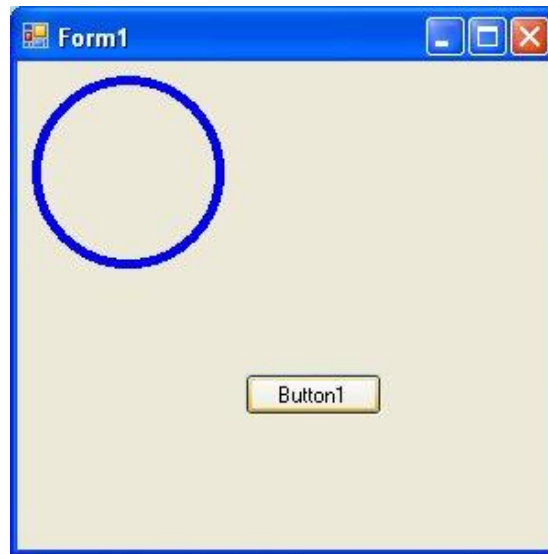
Lesson 24 – Managing Graphics -Drawing Text

We have learned how to draw rectangle, ellipse and circle in Visual Basic 2010 in the preceding chapters, now let's learn how to draw text on the screen. Yes, instead of using the Print command, you can also draw text on the screen.

24.1 Drawing Text

In order to draw text on the screen, we can use the DrawString method. The syntax is as follows:

myGraphics.DrawString(myText, myFont, mybrush, X , Y)

Where myGraphics is the Graphics object, myText is the text you wish to display on the screen, myFont is the font object created by you, myBrush is the brush style created by you and X, Y are the coordinates of upper left corner of the Text.

You can create the Font object in Visual Basic 2010 using the following statement:

myFont = New System.Drawing.Font("Verdana", 20)

Where the first argument of the font is the font typeface, and the second argument is the font size. You can add a third argument as font style, either bold, italic, underline. Here are the examples:

myFont = New System.Drawing.Font("Verdana", 20, FontStyle.Bold)

myFont = New System.Drawing.Font("Verdana", 20, FontStyle.Underline)

myFont = New System.Drawing.Font("Verdana", 20, FontStyle.Italic)

myFont = New System.Drawing.Font("Verdana", 20, FontStyle.Regular)

To create your Brush object, you can use the following statement:

Dim myBrush As Brush

myBrush = New Drawing.SolidBrush(Color.BrushColor)

Besides the seven colors, some of the common Brush Colors are AliceBlue, AquaMarine Beige, DarkMagenta, DrarkOliveGreen, SkyBlue and more. You don't have to remember the names of all the colors, the intelliSense will let you browse through the colors in a drop-down menu once you type the dot after the word Color.

Now we shall proceed to draw the font using the sample code below:

Example 24.1

Dim myGraphics As Graphics = Me.CreateGraphics

Dim myFont As Font
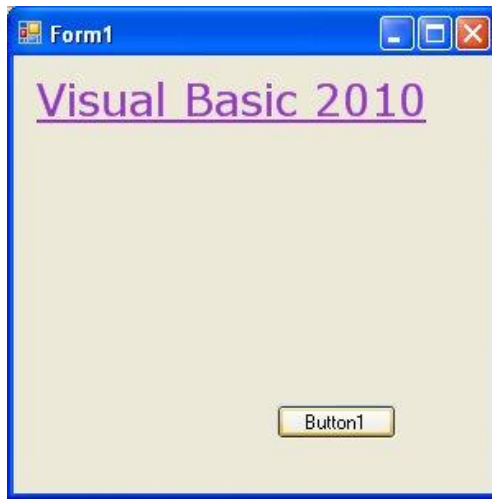
Dim myBrush As Brush

myBrush = New Drawing.SolidBrush(Color.DarkOrchid)

myFont = New System.Drawing.Font("Verdana", 20, FontStyle.Underline)

myGraphics.DrawString("Visual Basic 2010", myFont, myBrush, 10, 10)

Run the program above and you can see the following output:

The preceding can be modified if you don't want to create the Font and the Brush objects. You can use the font of an existing object such as the Form and the System Colors. Replace the last line in the preceding example with this line(you need to delete the lines that create the Brush and the Font objects as well)

myGraphics.DrawString("Visual Basic 2010", me.Font, System.Drawing.Brushes.DarkOrchid, 10, 10)

You can also add a InputBox which let the user enter his or her message then display the message on the screen.

This is shown in Example 24.2

Example 24.2

Dim myGraphics As Graphics = Me.CreateGraphics

Dim myFont As Font

Dim myBrush As Brush

Dim userMsg As String

userMsg = InputBox("What is your message?", "Message Entry Form", "Enter your message here", 100, 200)

myBrush = New Drawing.SolidBrush(Color.DarkOrchid)

myFont = New System.Drawing.Font("Verdana", 20, FontStyle.Underline)

myGraphics.DrawString(userMsg, myFont, myBrush, 10, 10)

Lesson 25 – Managing Graphics -Drawing Polygon and Pie

We have learned how to draw rectangle, ellipse ,circle and text in Visual Basic 2010 in the preceding chapters, now let's learn how to draw polygons on the screen. Besides that, we shall also learn how to draw pie.

25.1: Drawing a Polygon

Polygon is a closed plane figure bounded by three or more straight sides. In order to draw a polygon on the screen, we need to define the coordinates of all the points (also known as vertices) that joined up to form the polygon.

The syntax to defines the points of a polygon with vertices A1,A2,A3,A4…….An is as follows;

```
Dim A1 As New Point(X1,Y1)

Dim A2 As New Point(X2,Y2)

Dim A3 As New Point(X3,Y3)

Dim A4 As New Point(X4,Y4)

.

.

Dim An as New Point(Xn,Yn)
```

After declaring the points, we need to define a point structure that group all the points together using the following syntax:

Dim myPoints As Point() = {A1, A2, A3,….,An}

.Finally, create the graphics object and use the DrawPolygon method to draw the polygon using the following syntax:

Dim myGraphics As Graphics = Me.CreateGraphics

myGraphics.DrawPolygon(myPen, myPoints)

where myPen is the Pen object created using the following syntax:

myPen = New Pen(Drawing.Color.Blue, 5)

Example 25.1: Drawing a Triangle

A triangle is a polygon with three vertices. Here is the sample code:

Dim myPen As Pen

Dim A As New Point(10, 10)

Dim B As New Point(100, 50)

Dim C As New Point(60, 150)

Dim myPoints As Point() = {A, B, C}

myPen = New Pen(Drawing.Color.Blue, 5)

Dim myGraphics As Graphics = Me.CreateGraphics

myGraphics.DrawPolygon(myPen, myPoints)

Running the program produces the image below:

Example 25.2: Drawing a Quadrilateral

A quadrilateral is a polygon consists of four sides,so you need to define four vertices. The following is the code:

```
Dim myPen As Pen
Dim A As New Point(10, 10)
Dim B As New Point(100, 50)
Dim C As New Point(120, 150)
Dim D As New Point(60, 200)
Dim myPoints As Point() = {A, B, C, D}
myPen = New Pen(Drawing.Color.Blue, 5)
Dim myGraphics As Graphics = Me.CreateGraphics
myGraphics.DrawPolygon(myPen, myPoints)
```
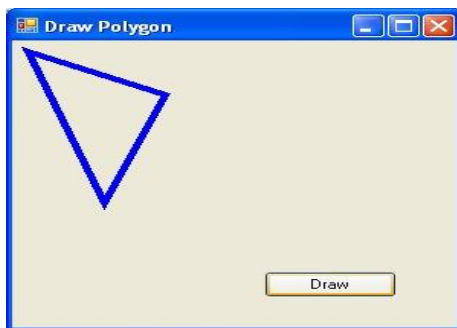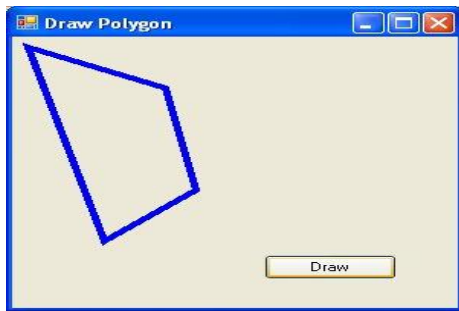
The output image is as shown below:

25.2: Drawing Pie

In order to draw a pie, you can use the DrawPie method of the graphics object. As usual, you need to create the Graphics and the Pen objects. The syntax for drawing a pie is:

myGraphics.DrawPie(myPen, X, Y, width,height, StartAngle, SweepAngle)

Where X and Y are the coordinates the bounding rectangle, other arguments are self-explanatory. Both StartAngle and SweepAngle are measured in degree. SweepAngle can take possible or negative values. If the value is positive, it sweep through clockwise direction while negative means it sweep through anticlockwise direction.

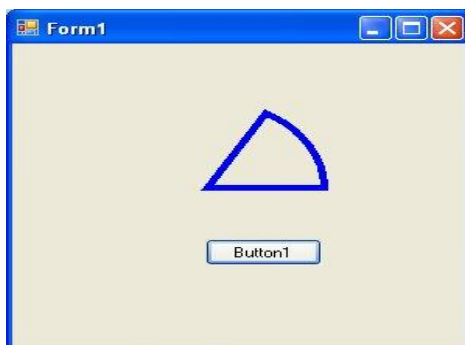Example 25.3: Drawing a pie that starts with 0 degree and sweep clockwise through 60 degree.

Dim myPen As Pen

myPen = New Pen(Drawing.Color.Blue, 5)

Dim myGraphics As Graphics = Me.CreateGraphics

myGraphics.DrawPie(myPen, 50,50, 150,150,0,60)

The output image is as shown below:

Lesson 26 – Managing Graphics-Filling Shapes with Color

In previous lessons, we have learned how to draw rectangle, ellipse ,circle ,polygon and pie with outlines only. In this lesson, we will show you how to fill the shapes with color, or simply solid shapes. In Visual Basic 2010, three methods are used to fill shapes are FillRectangle, FillEllipse , FillPolygon and FillPie.

In order to fill the above shapes with color, we need to create the Brush object using the following syntax:

myBrush = New SolidBrush(Color.myColor)

Where myColor can be any color such as red,blue, yellow and more. You don't have to worry about the names of the colors because the intellisense will display the colors and enter the period after the Color key word.

26.1 Drawing and Filling a Rectangle

In Visual Basic 2010 ,the syntax to fill a rectangle with the color defined by the brush object is:

myGraphics.FillRectangle (myBrush, 0, 0, 150, 150)

The complete code is shown in the example below:

Example 26.1

Dim myPen As Pen

Dim myBrush As Brush

Dim myGraphics As Graphics = Me.CreateGraphics

myPen = New Pen(Drawing.Color.Blue, 5)

myBrush = New SolidBrush(Color.Coral)

myGraphics.DrawRectangle(myPen, 0, 0, 150, 150)

myGraphics.FillRectangle(myBrush, 0, 0, 150, 150)

The Output is shown below:

26.2 Drawing and Filling an Ellipse

The syntax to fill a ellipse with the color defined by the brush object is:

myGraphics.FillEllipse (myBrush, 0, 0, 150, 150)

The complete code is shown in the example below:

Example 26.2

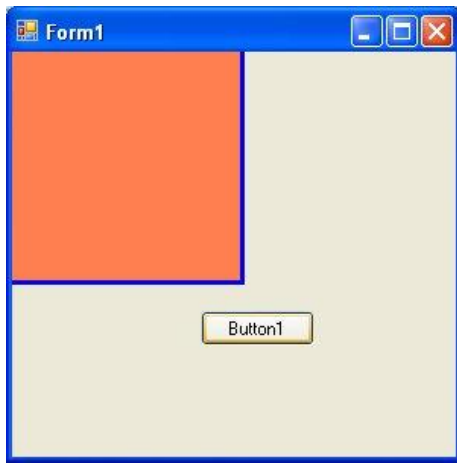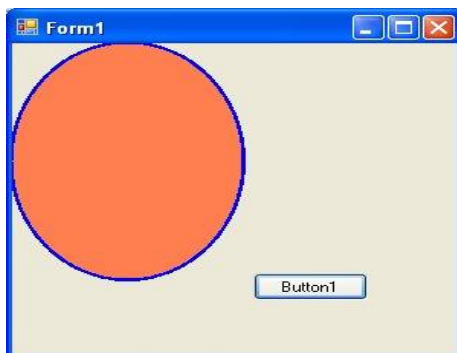Dim myPen As Pen

Dim myBrush As Brush

Dim myGraphics As Graphics = Me.CreateGraphics

myPen = New Pen(Drawing.Color.Blue, 5)

myBrush = New SolidBrush(Color.Coral)

myGraphics.DrawEllipse(myPen, 0, 0, 150, 150)

myGraphics.Ellipse(myBrush, 0, 0, 150, 150)

The output is shown below:

26.3 Drawing and Filling a Polygon

The syntax to fill a polygon with the color defined by the brush object is:

myGraphics.FillPolygon(myBrush, myPoints)

The complete code is shown in the example below:

Dim myPen As Pen

Dim myBrush As Brush

Dim A As New Point(10, 10)

Dim B As New Point(100, 50)

Dim C As New Point(120, 150)

Dim D As New Point(60, 200)

Dim myPoints As Point() = {A, B, C, D}

myPen = New Pen(Drawing.Color.Blue, 5)

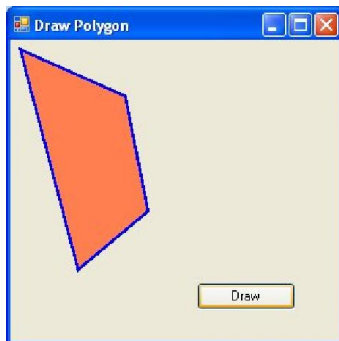myBrush = New SolidBrush(Color.Coral)

Dim myGraphics As Graphics = Me.CreateGraphics

myGraphics.DrawPolygon(myPen, myPoints)

myGraphics.FillPolygon(myBrush, myPoints)

Running the code produces the image below:

26.4 Drawing and Filling a Pie

The syntax to fill a pie with the color defined by the brush object is:

myGraphics.FillPie(myBrush, X, Y, width, height, StartAngle, SweepAngle)

The complete code is shown in the example below:

Dim myPen As Pen

Dim myBrush As Brush

myPen = New Pen(Drawing.Color.Blue, 5)

myBrush = New SolidBrush(Color.Coral)

Dim myGraphics As Graphics = Me.CreateGraphics

myGraphics.DrawPie(myPen, 30, 40, 150, 150, 0, 60)

myGraphics.FillPie(myBrush, 30, 40, 150, 150, 0, 60)

The output is shown below:
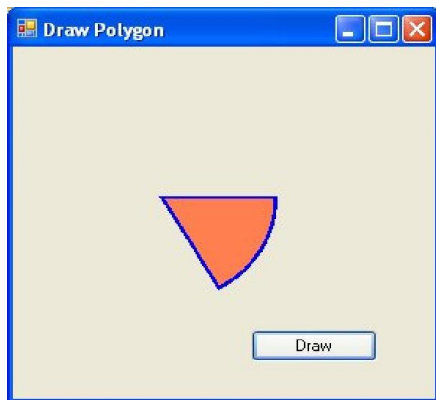
Lesson 27 – Using Timer

In this lesson, we shall show you how to use timer in Visual Basic 2010. Timer is used to control and manage events that are time related. For example, you need timer to create a clock, a stop watch, a dice, animation and more.

27.1 Creating a Clock

In order to create a clock, you need to use the Timer control that comes with Visual Basic 2010 . The Timer control is a control object that is only used by the developer, it is invisible during runtime and it does not allow the user to interact with it.

To create the clock, first of all start a new project in Visual Basic 2010 and select a new Windows Application. You can give the project any name you wish, but we will name it MyClock. Change the caption of the Form1 to MyClock in the properties window. Now add the Timer control to the form by dragging it from the ToolBox. Next, insert a label control into the

form. Change the Font size of the label to 14 or any size you wish, and set the Font alignment to be middle center. Before we forget, you shall also set the Interval property of the Timer control to 1000, which reflects a one second interval(1 unit is 1 millisecond).

Now, you are ready for the coding. Actually you would be surprise that what you need to create a clock is only a one-line code, that is:

Label1.Text = TimeOfDay

To actually create the clock, click on the Timer control and insert the code above ,as shown below:

Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick

Label1.Text = TimeOfDay

End Sub

The Clock is shown below:

27.2 Creating a Stopwatch

We can create a simple stopwatch using the Timer control. Start a new project and name it stopwatch. Change the Form1 caption to Stopwatch. Insert the Timer control into the form and set its interval to 1000 which is equal to one second. Also set the timer Enabled property to False so that it will not start ticking when the program is started. Insert three command buttons and change their names to StartBtn, StopBtn and ResetBtn respectively. Change their text to "Start", "Stop" and "Reset" accordingly. Now,key in the code as follows:

Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick

'To increase one unit per second

Label1.Text = Val(Label1.Text) + 1

End Sub

Private Sub StopBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles StopBtn.Click

'To stop the Timer

Timer1.Enabled = False

End Sub

Private Sub StartBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles StartBtn.Click

'To start the Timer

Timer1.Enabled = True

End Sub

Private Sub ResetBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ResetBtn.Click

'To reset the Timer to 0

Label1.Text = 0

End Sub

The Interface of the Stopwatch is as shown below:

27.3 Creating a Digital Dice

We can create a digital dice easily using the Timer Control. To create a dice, you need to generate random numbers using the Rnd function. Rnd generates numbers between 0 and 1. The statement

n = Int(1 + Rnd() * 6)

generates integers from 1 to 6 randomly.

In the code, we introduce the variable m to control the length of time of the rolling process. If m is more than 1000, then the rolling process will stop by setting the timer enabled property to False.

The compete is shown below:

```
Public Class Form1

Dim n, m As Integer

Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick

m = m + 10

If m < 1000 Then

n = Int(1 + Rnd() * 6)

LblDice.Text = n

Else

Timer1.Enabled = False

m = 0

End If

End Sub

Private Sub RollDiceBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles RollDiceBtn.Click

Timer1.Enabled = True

End Sub

End Class
```
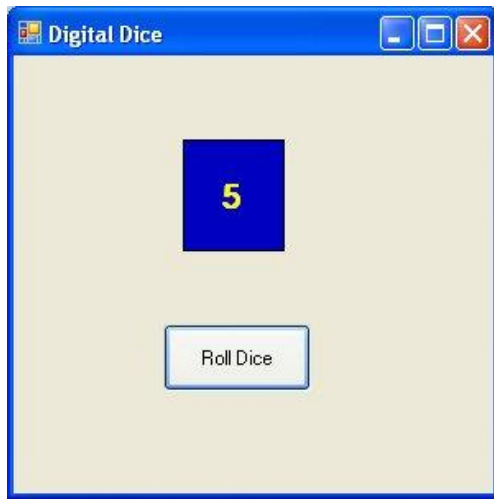
Running the program produces a dice with fast changing numbers which stop at a certain number. The interface is shown below:

Lesson 28 – Creating Animation

Although Visual Basic 2010 is generally a programming language designed for creating business and other industrial applications and not for creating animation, it can be used to create animation. In this lesson, we will show you just that.

28.1 Moving an object

In this section, we will show you how to move an object by pressing a command button. You need to make use of the Top and Left properties of an object to create animation. The Top property defines the distance of the object from the top most border of the screen while the Left property defines the distance of the object from leftmost border of the screen. By adding or subtracting the distance of the object we can create the animated effect of moving an object.

Start a new project and name it as Movable Object, or any name you wish. Now insert a PictureBox and in its Image property import a picture from your hard drive or other sources. Next, insert four command buttons, change their captions to Move Up, Move Down, Move Left and Move Right. Name them as MoveUpBtn, MoveDowbBtn, MoveLeftBtn and MoveRightBtn.

Now, click on the buttons and key in the following code:

```
Private Sub MoveDownBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MoveDownBtn.Click

PictureBox1.Top = PictureBox1.Top + 10

End Sub

Private Sub MoveLeftBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MoveLeftBtn.Click

PictureBox1.Left = PictureBox1.Left – 10
```

End Sub

Private Sub MoveRightBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MoveRightBtn.Click

PictureBox1.Left = PictureBox1.Left + 10

End Sub

Explanation:

Each time the user clicks on the Move Down button, the distance of the PictureBox increases by 10 pixels from the top border, creating a downward motion. On the other hand, each time the user clicks on the Move Up button, the distance of the PictureBox decreases by 10 pixels from the top borders, thus creating an upward motion. In addition, each time the user clicks on the Move Left button, the distance of the PictureBox decreases by 10 pixels from the left border, thus creating a leftward motion. Lastly, each time the user clicks on the Move Right button, the distance of the PictureBox increases by 10 pixels from the left border, thus creating an rightward motion.

The interface is shown below:
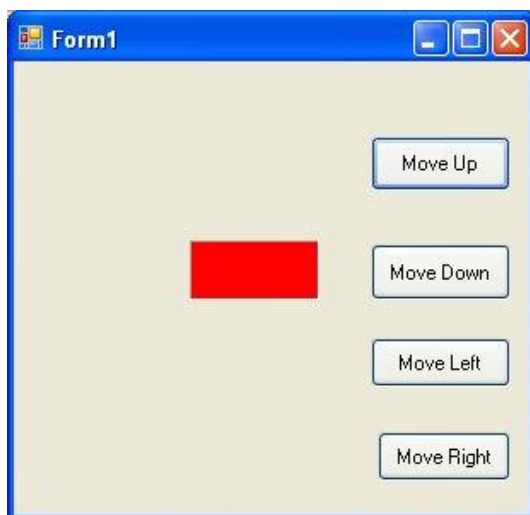
28.2 Creating Animation using Timer

We can create continuous animation using timer without the need to manually clicking a command button. We can create left to right or top to bottom motion by writing the necessary code.

First of all, insert a PictureBox into the form. In the PictureBox properties window,select the image property and click to import an image file from your external sources such as your hard drive, your Pendrive or DVD. We have inserted an image of a bunch of grapes.Next, insert a Timer control into the form set its interval property to 100, which is equivalent to 0.1 second.

Finally, add two command button to the form, name one of them as AnimateBtn and the other one as StopBtn, and change to caption to Animate and Stop respectively.

We make use of the Left property of the PictureBox to create the motion. PictureBox.Left means the distance of the PictureBox from the left border of the Form . Now click on the Timer control and type in the following code:

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick

If PictureBox1.Left < Me.Width Then

PictureBox1.Left = PictureBox1.Left + 10

Else

PictureBox1.Left = 0

End If

End Sub
```

In the code above, Me.Width represents the width of the Form. If the distance of the PictureBox from the left is less than the width of the Form, a value of 10 is added to the distance of the PictureBox from the left border each time the Timer tick, or every 0.1 second in this example. When the distance of the PictureBox from the left border is equal to the width of the form, the distance from the left border is set to 0, which move the PictureBox object to the left border and then move left again, thus creates an oscillating motion from left to right. We need to insert a button to stop motion. The code is:

```
Timer1.Enabled = False
```

To animate the PictureBox object, we insert a command button and key in the following code:

```
Timer1.Enabled = True
```

The Image of the Animation program is shown below:

Lesson 29 Working with Databases Part 1

29.1 Introduction to Database in Visual Basic 2010

In our daily life, we deal with many types of information or data such as names, addresses, money, date, stock quotes, statistics and more. If you are in business or working as a professional, you have to handle even more data. For example, a doctor need to keep track of patients' personal and medical information such as names, addresses, phone numbers as well as blood pressure readings, blood sugar readings,surgical history, medicines prescribed in the past and more. On the other hand, businesses usually have to manage large amount of data pertaining to products and customers. All these data need to be organized into a database for the ease of data management.

In the past , people usually deal with data manually like using cards and folders. However, in present day fast pace global environment and Information age, it is no longer feasible to manage data manually. Most data are now managed using computer-based database management systems. Computer-based Database management systems can handle data much faster and more efficient than human beings do. With the advent of the network and the Internet technologies, data can now be managed locally and remotely. Companies usually invest heavily in database management systems in order to run the organizations efficiently and effectively. Database management systems are usually used in running payroll system, inventory system, accounting system, payment system, order handling system, customer relationship management system(CRM) and more. Some of the commercial database management system(DBMS) are Oracle, Microsoft SQL server and Microsoft Access

### 29.2 Creating a Database Application in Visual Basic 2010

A database management system typically deals with storing, modifying, and extracting information from a database. It can also add, edit and delete records from the database. However, a DBMS can be very difficult to handle by ordinary people or business men who have no technological backgrounds. Fortunately, we can create user friendly database applications to handle the aforementioned jobs with the DBMS running in the background. One of the best programs that can create such database application is none other than Visual Basic 2010.

Visual Basic 2010 uses ADO.NET to handle databases. ADO.NET is Microsoft's latest database technology which can works with many other advanced database management system such as Microsoft SQL server. In this lesson, we will develop codes that make use of Microsoft SQL Server 2008, therefore you need to have Microsoft SQL Server 2008 installed in your PC, otherwise you can download it from http://www.microsoft.com/en-us/download/details.aspx?id=1695. Besides, you might want to download Microsoft SQL Server 2008 Management Studio Express, the SQL database management system that allows you to create and manage databases.

To begin building the database project in Visual Basic 2010, launch Visual Basic 2010. You can name your project as Database Project 1 or what ever name you wish to call it. Next, change the default form's Text property to Contacts as we will be building a database of contact list. There are a few objects in ADO.NET that are required to build the database. There are:

- SqlConnection- to connect to a data source in SQL Server
- DataTable -to store data for navigation and manipulation
- DataAdapter- to populate a DataReader

The aforementioned objects belongs to the System.Data and the System.Xml namespace. Therefore, we need to reference them in the beginning before we can work with them. To reference the ADO.NET object, choose project from the menu then select Database Project 1 properties to display the project properties. Next click the References tab to show the active references for the project, as shown in Figure 29.1
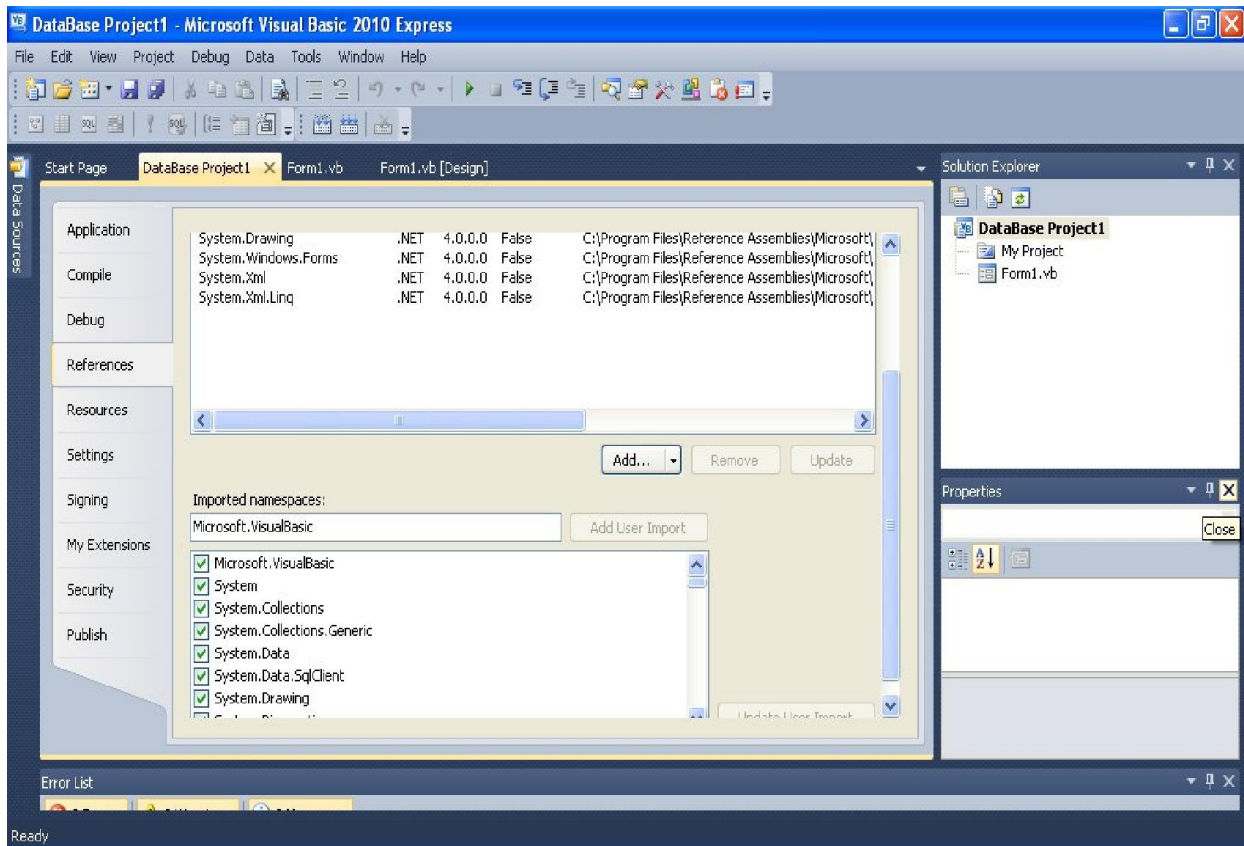
Under imported namespaces, make sure system.data, System.Data.Sqlclient are selected, otherwise check them. Having done that you need to click the Save All button on the toolbar and then return to the Visual Basic 2010 IDE.

We shall proceed to create connection to the database source file in the next lesson.

Lesson 30- Working with Databases Part 2

30.1 Creating Connection to a Database using ADO.NET

In Visual Basic 2010, we need to create connection to a database before we can access its data. Before we begin, let's create a new database. Since we are using SQL Server 2008 as the database engine, we will use **Microsoft Studio Management Express** to create a database with the mdf extension. We shall name this database file as test.mdf. After creating the database, build a table called Contacts and create two fields and name them ContactName and State respectively. Enter a few data in the table and click Save All to save the data. Now we are ready to connect to this new database.

ADO.NET offers a number of connection objects such as **OleDbConnection, SqlConnection** and

more. OleDbConnection is used to access OLEDB data such as Microsoft Access whilst

SqlCOnnection is used to access data provided by Microsoft SQL server. Since we will work with

SQL database in our example, we will use the SqlConnection object. To initialize the variable to a new

SqlConnection object, we use the following syntax:**Private MyCn As New SqlConnection**

Having created the instance of the SqlConnecton object, the next step is to establish a connection to the data source using the **SQL ConnectionString** property. The syntax is:

MyCn.ConnectionString = "Data Source=lenovo-4903350b\mssmlbiz;
AttachDbFilename=C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\Test.mdf;
" & _
"User Instance=True;Integrated Security=SSPI"

* You need to change the reference to the SQL server (*lenovo-4903350b\mssmlbiz*) as well as the path to database file *Test.mdf* .

After establishing connection to the database, you can open the database using the following syntax:

MyCn.Open()

30.2 Populating Data in ADO.NET

Establishing connection to a database in Visual Basic 2010 using SqlConnection alone will not present anything tangible things to the user to manipulate the data until we add more relevant objects and write relevant codes to the project.

The next step is to create an instance of the **SqlDataAdpater** in our code so that we can populate the **DataTable** with data from the data source. Besides, you also need to create an instance of the DataTable. Other than that, you should also create an instance of the**SqlCommandBuilder** which is used to manipulate data such as updating and deleting data in the Datatable and send the changes back to the data source. The statements are:

```
Private MyDatAdp As New SqlDataAdapter
Private MyCmdBld As New SqlCommandBuilder
 Private MyDataTbl As New DataTable
```

Having created the above of objects, you need to include the following statements in the Sub Form_Load event to start filling the DataTable with data from the data source. The statements are as follows:

```
MyDatAdp = New SqlDataAdapter("Select* from Contacts", MyCn)
MyCmdBld = New SqlCommandBuilder(MyDatAdp)
MyDatAdp.Fill(MyDataTbl)
```

After filling up the DataTable , we need to write code to access the data. To access data in the DataTable means that we need to access the rows in the table. We can achieve this by using the DataRow object. For example, we can write the following to access the first row of the table and present the data via two text boxes with the name txtName and txtState respectively:

```
Dim MyDataRow As DataRow = MyDataTbl.Rows(0)
Dim strName As String
Dim strState As String
strName = MyDataRow("ContactName")
strState = MyDataRow("State")
txtName.Text = strName.ToString
txtState.Text = strState.ToStringMe.showRecords()
```

* The two fields being referenced here are ContactName and State. Note Index 0 means first row.

showRecords() is a sub procedure created to show data in the text boxes. The code is as follows:

```
Private Sub showRecords()

If MyDataTbl.Rows.Count = 0 Then
txtName.Text = ""
txtState.Text = ""
Exit Sub
End If

txtName.Text = MyDataTbl.Rows(MyRowPosition)("ContactName").ToString
txtState.Text = MyDataTbl.Rows(MyRowPosition)("State").ToString

End Sub
```

The output interface:

We will discuss how to manipulate data in the next lesson.

Lesson 31: Working with Databases Part 3

In previous lessons, you have learned how to connect to a database as well as filling up the table with data in Visual Basic 2010, now you shall learn how to manipulate data in the database. Manipulating data means adding news records, editing records, deleting records, browsing records and more.

31.1 Browsing Records

In previous lesson, we have learned how to display the first record using the showRecords sub procedure. In this lesson, we will create command buttons and write relevant codes to allow the user to browse the records forward and backward as well as fast forward to the last record and back to the first record.

The first button we need to create is for the user to browse the first record. We can use button's text << to indicate to the user that it is the button to move to the first record and button's text >> to move to the last record. Besides we can use button's text < for moving to previous record and button's text > for moving to next record.

The code for moving to the first record is:

MyRowPosition = 0
Me.showRecords()

The code for moving to previous record is:

If MyRowPosition > 0 Then
MyRowPosition = MyRowPosition – 1
Me.showRecords()
End If

The code for moving to next record is:

If MyRowPosition < (MyDataTbl.Rows.Count – 1) Then
MyRowPosition = MyRowPosition + 1
Me.showRecords()
End If

The code for moving to last record is:

If MyDataTbl.Rows.Count > 0 Then
MyRowPosition = MyDataTbl.Rows.Count – 1
Me.showRecords()
End If

31.2 Editing, Saving, Adding and Deleting Records

You can edit any record by navigating to the record and change the data values. However, you need to save the data after editing them. You need to use the update method of the SqlDataAdapter to save the data. The code is:

```
If MyDataTbl.Rows.Count <> 0 Then
MyDataTbl.Rows(MyRowPosition)("ContactName") = txtName.Text
MyDataTbl.Rows(MyRowPosition)("state") = txtState.Text
MyDatAdp.Update(MyDataTbl)
End If
```

You can also add new record or new row to the table using the following code :

```
Dim MyNewRow As DataRow = MyDataTbl.NewRow()
MyDataTbl.Rows.Add(MyNewRow)
MyRowPosition = MyDataTbl.Rows.Count – 1
Me.showRecords()
```

The code above will present a new record with blank fields for the user to enter the new data. After entering the data, he or she can then click the save button to save the data.

Lastly, the user might want to delete the data. The code to delete the data is:

```
If MyDataTbl.Rows.Count <> 0 Then
MyDataTbl.Rows(MyRowPosition).Delete()
MyDatAdp.Update(MyDataTbl)
MyRowPosition = 0
Me.showRecords()
End If
```