Aim: To write a Program to demonstrate Data Types in VB .NET

**PROCEDURE:**

1. Click Start→Programs→MicrosoftVisualStudio 2005 → MicrosoftVisualStudio 2005
2. Select File →New → Project → Windows Application.
3. Place the button in the window.
4. Write the code in click event of btnresult.
5. Run the application by pressing F5 key or by clicking debug button.

**SOURCE CODE:**

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object,
                              ByVal e As System.EventArgs) Handles Button1.Click
        Dim check As Boolean
        check = True
        MsgBox("The value assigned for check is  : " & check)

        Dim count As Integer
        count = 100
        MsgBox("The value holding count is  : " & count)

        Dim str As String
        str = "String test "
        MsgBox("The value holding str is  : " & str)
    End Sub
End Class
```

Aim: To write a Program to demonstrate Type Conversion in VB .NET

**PROCEDURE:**

1. Click Start→Programs→MicrosoftVisualStudio 2005 → MicrosoftVisualStudio 2005
2. Select File →New → Project → Windows Application.
3. Place the button in the window.
4. Write the code in click event of btnresult.
5. Run the application by pressing F5 key or by clicking debug button.

**SOURCE CODE:**

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
        Dim iDbl As Double
        Dim iInt As Integer
        iDbl = 9.123
        MsgBox("The value of iDbl is " & iDbl)
        iInt = iDbl
        'after conversion
        MsgBox("The value of iInt is " & iInt)
    End Sub
End Class
```

Aim: To write a Program to demonstrate Exception Handling in VB .NET

**PROCEDURE:**

1. Click Start→Programs→MicrosoftVisualStudio 2005 → MicrosoftVisualStudio 2005
2. Select File →New → Project → Windows Application.
3. Place the button in the window.
4. Write the code in click event of btnresult.
5. Run the application by pressing F5 key or by clicking debug button.

**SOURCE CODE:**

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
        Try
            Dim i As Integer
            Dim resultValue As Integer
            i = 100
            resultValue = i / 0
            MsgBox("The result is " & resultValue)
        Catch ex As Exception
            MsgBox("Exception catch here ..")
        Finally
            MsgBox("Finally block executed ")
        End Try
    End Sub
End Class
```

Aim: To write a Program to demonstrate Option Explicit statement in VB .NET

**Option Explicit : -**

This statement ensures whether the compiler requires all variables to be explicitly declared or not before it use in the program

The **Option Explicit** has two modes. **On** and **Off** mode. If Option Explicit mode in ON , you have to declare all the variable before you use it in the program . If not , it will generate a compile-time error whenever a variable that has not been declared is encountered .If the Option Explicit mode is OFF , Vb.Net automatically create a variable whenever it sees a variable without proper declaration.

By default the ***Option Explicit is On***

With the Option Explicit On , you can reduce the possible errors that result from misspelled variable names. Because in Option Explicit On mode you have to declare each variable in the program for storing data.



**PROCEDURE:**

1. Click Start→Programs→MicrosoftVisualStudio 2005 → MicrosoftVisualStudio 2005
2. Select File →New → Project → Windows Application.
3. Place the button in the window.
4. Write the code in click event of btnresult.
5. Run the application by pressing F5 key or by clicking debug button.

***Option Explicit - On***

```
Public Class Form1
  Private Sub Button1_Click(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles Button1.Click
    Dim someVariable As String
    someVariable = "Option Explicit ON"
    MsgBox(someVariable)
```

```
    End Sub
End Class
```

***Option Explicit is Off***

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
        someVariable = "Option Explicit ON"
        MsgBox(someVariable)
    End Sub
End Class
```

***Note*** :

Here "someVariable" is not declared , because the ***Option Explicit Of*** , without any compiler error you can continue the program.

Aim: To write a Program to demonstrate Explicit Error Handling in VB .NET

**PROCEDURE:**

1. Click Start→Programs→MicrosoftVisualStudio 2005 → MicrosoftVisualStudio 2005
2. Select File →New → Project → Windows Application.
3. Place the button in the window.
4. Write the code in click event of btnresult.
5. Run the application by pressing F5 key or by clicking debug button.

**SOURCE CODE:**

*Program 1:*

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
        Dim result As Integer
        Dim num As Integer
        num = 100
        result = num / 0
        MsgBox("here")
    End Sub
  End Class
```

*Program 2:*

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
        On Error GoTo nextstep
        Dim result As Integer
```

```
        Dim num As Integer
        num = 100
        result = num / 0
nextstep:
        MsgBox("Control Here")
    End Sub
End Class
```

Aim: To write a Program to find the date difference between two dates in VB .NET

**PROCEDURE:**

1. Click Start→Programs→MicrosoftVisualStudio 2005 → MicrosoftVisualStudio 2005
2. Select File →New → Project → Windows Application.
3. Place the button in the window.
4. Write the code in click event of btnresult.
5. Run the application by pressing F5 key or by clicking debug button.

**SOURCE CODE:**

```
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim firstDate As New System.DateTime(2000, 1, 1)
        Dim secondDate As New System.DateTime(2000, 5, 31)

        Dim diff As System.TimeSpan = secondDate.Subtract(firstDate)
        Dim diff1 As System.TimeSpan = secondDate - firstDate

        Dim diff2 As String = (secondDate - firstDate).TotalDays.ToString()

        MsgBox(diff2.ToString())
    End Sub
End Class
```

Aim: To write a Program to retrieve Operating System information in VB .NET

**PROCEDURE:**

1. Click Start→Programs→MicrosoftVisualStudio 2005 → MicrosoftVisualStudio 2005
2. Select File →New → Project → Windows Application.
3. Place the button in the window.
4. Write the code in click event of btnresult.
5. Run the application by pressing F5 key or by clicking debug button.

The following VB.NET program retrieves the current operating system information like version and platform identifier with the help of OperatingSystem Class and Environment Class.

The OperatingSystem Class provides a method to copy an instance of OperatingSystem, and a method to return a string representation of operating system information.

The OperatingSystem class is not a general purpose class and you cannot derive a more inclusive type from the OperatingSystem class. If you need a type to contain other information about an operating system, create your own type, then include a field of type OperatingSystem and any additional fields, properties, or methods you require.

**SOURCE CODE:**

```
Public Class Form1
        Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
                Dim _os As OperatingSystem
                _os = Environment.OSVersion
                MsgBox(_os.VersionString.ToString)
        End Sub
End Class
```

Aim: To write a Program to Start and Kill Processes explicitly in VB .NET

**PROCEDURE:**

1. Click Start→Programs→MicrosoftVisualStudio 2005 → MicrosoftVisualStudio 2005
2. Select File →New → Project → Windows Application.
3. Place the two buttons in the window.
4. Write the code in click event of btnresult.
5. Run the application by pressing F5 key or by clicking debug button.

The following VB.NET program allows to start multiple calculator applications and later it kill the same instances of all calculator applications.

The Process component is a useful tool for starting, stopping, controlling, and monitoring applications. A process is a running application and a thread is the basic unit to which the operating system allocates processor time. Using the Process component, you can obtain a list of the processes that are running, or you can start a new process.

GetProcessesByName(String) - Creates an array of new Process components and associates them with all the process resources on the local computer that share the specified process name.

```
Dim _proceses As Process()

  _proceses = Process.GetProcessesByName("calc")
```

The above syntax retrieve all the process associated with "calc" application in the _proceses array. System.Diagnostics provides access to local and remote processes and enables you to start and stop local system processes.

***PROGRAM SOURCE CODE***

```
Public Class Form1
```

```
        Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
                System.Diagnostics.Process.Start("calc")
        End Sub

        Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click

                Dim _proceses As Process()
                _proceses = Process.GetProcessesByName("calc")
                For Each proces As Process In _proceses
                        proces.Kill()
                Next

        End Sub

End Class
```

Aim: To write a Program to Generate a Random Number in VB .NET

The Random Class in VB.NET represents a pseudo-random number generator, a device that produces a sequence of numbers that meet certain statistical requirements for randomness.

**Shared random As New Random()**

The Next() method in Random class returns a nonnegative random number.

**random.Next()**

We can limit the generation of Random number by giving a specified range to Next() method.

**random.Next(10,20)**

The above code limit to generate the Random number within the range from 10 to 20.



**PROCEDURE:**

1. Click Start→Programs→MicrosoftVisualStudio 2005 → MicrosoftVisualStudio 2005
2. Select File →New → Project → Windows Application.
3. Place the button in the window.
4. Write the code in click event of btnresult.
5. Run the application by pressing F5 key or by clicking debug button.

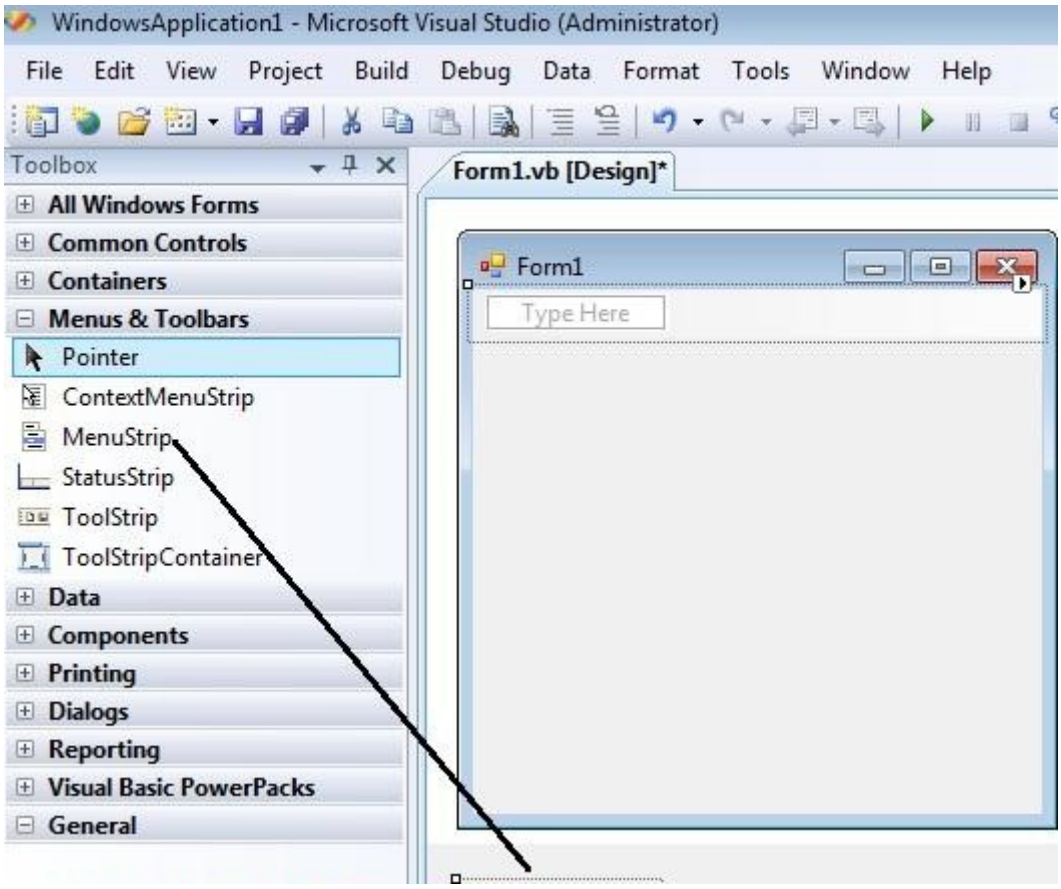
```
Public Class Form1
    Shared random As New Random()
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
```

```
        Dim i As Integer
        For i = 0 To 5
            MsgBox(Convert.ToString(random.Next(10, 20)))
        Next
    End Sub
End Class
```

**AIM : TO perform various operations in Menu Control of VB.NET**

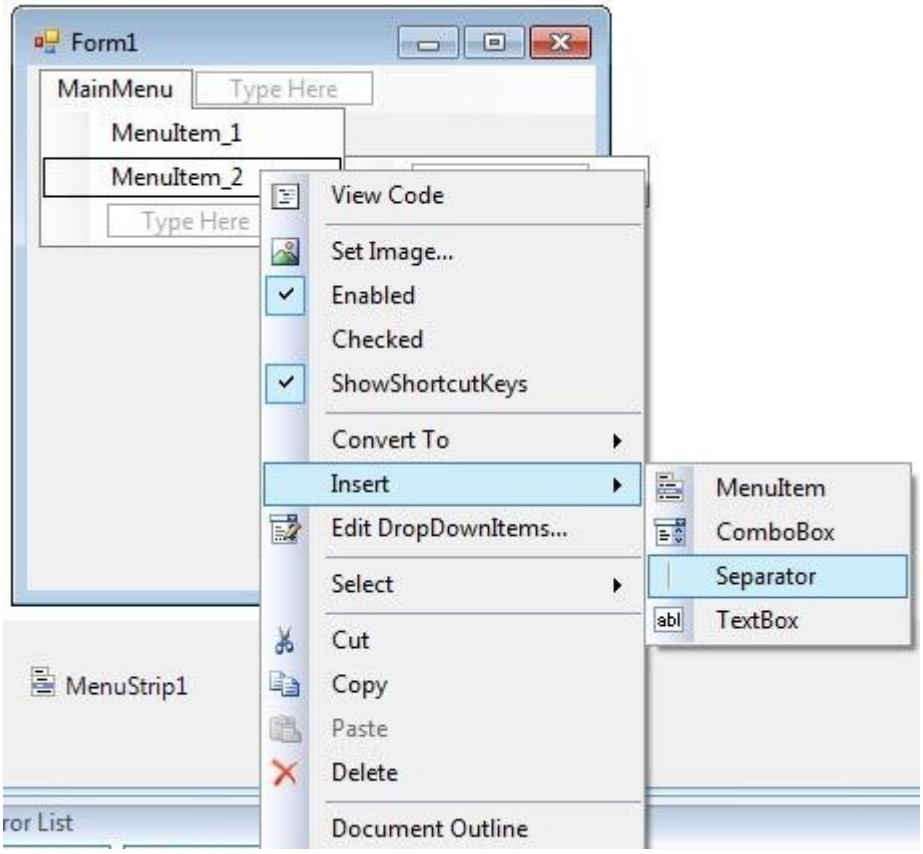
A menu is located on the menu bar and contains a list of related commands. MainMenu is the container for the Menu structure of the form and menus are made of MenuItem objects that represent individual parts of a menu.

You can create a main menu object on your form using the MainMenu control. The following picture shows how to drag the Menustrip Object to the Form.




After drag the Menustrip on your form you can directly create the menu items by type a value into the "Type Here" box on the menubar part of your form. From the following picture you can understand how to create each menu items on mainmenu Object.


If you need a separator bar , right click on your menu then go to insert->Separator.



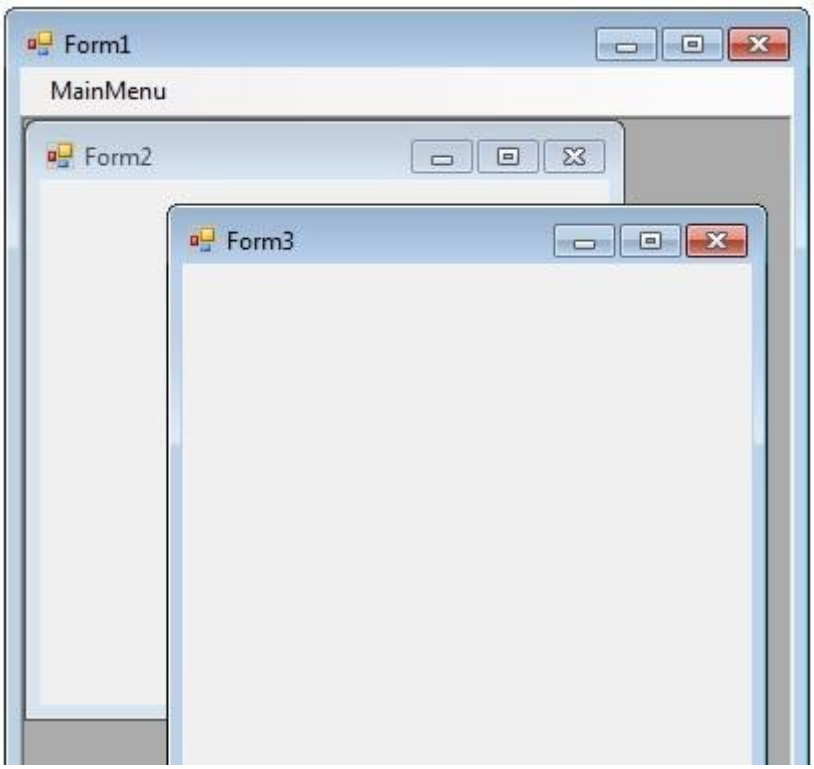
After creating the Menu on the form , you have to double click on each menu item and write the programs there depends on your requirements. The following Vb.Net program shows how to show a message box when clicking a menu item.

**Source code:**

```
Public Class Form1

    Private Sub MenuItem1ToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem1ToolStripMenuItem.Click

        MsgBox("You are selected MenuItem_1")

    End Sub

End Class
```

# MDI Form

A Multiple Document Interface (MDI) programs can display multiple child windows inside them.




This is in contrast to single document interface (SDI) applications, which can manipulate only one document at a time. Visual Studio Environment is an example of Multiple Document Interface (MDI) and notepad is an example of an SDI application, opening a document closes any previously opened document. Any windows can become an MDI parent, if you set the IsMdiContainer property to True.

```
IsMdiContainer = True
```

The following vb.net program shows a MDI form with two child forms. Create a new VB.Net project, then you will get a default form Form1 . Then add two mnore forms in the project (Form2 , Form 3) . Create a Menu on your form and call these two forms on menu click event.

NOTE: If you want the MDI parent to auto-size the child form you can code like this.

```
form.MdiParent = Me
```

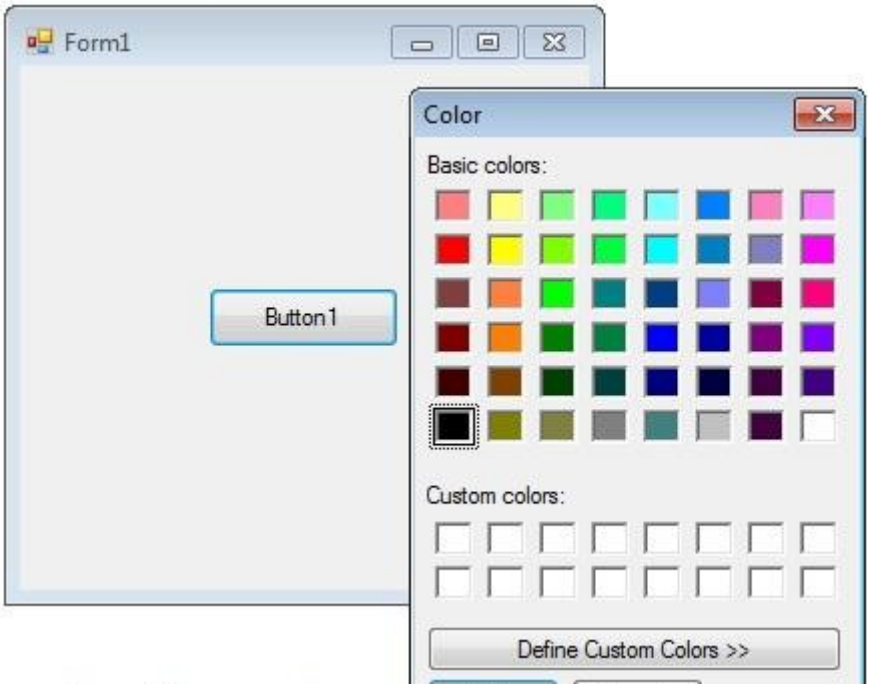
```
form.Dock = DockStyle.Fill

form.Show()
```

## SOURCE CODE:

```
Public Class Form1

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

    Private Sub MenuItem1ToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem1ToolStripMenuItem.Click
        Dim frm2 As New Form2
        frm2.Show()
        frm2.MdiParent = Me
    End Sub

    Private Sub MenuItem2ToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem2ToolStripMenuItem.Click
        Dim frm3 As New Form3
        frm3.Show()
        frm3.MdiParent = Me
    End Sub

End Class
```

# Color Dialog Box

There are several classes that implement common dialog boxes, such as color selection and print setup etc.




The user can choose a color from either a set of basic or custom color palettes. You can invite a color dialog box by calling ShowDialog() method.

```
Dim dlg As New ColorDialog

dlg.ShowDialog()
```

The Color dialog box has a full version and a partial version of the user interface. The full version includes the basic controls and has additional controls that allow the user to create custom colors. The partial version has controls that display the basic and custom color palettes from which the user can select a color value. The system stores internal colors as 32-bit RGB values that have the following hexadecimal form: 0x00bbggrr.

The following Vb.Net program invites a color dialog box and retrieve the selected color to a string.

## Source code:

```
Public Class Form1
        Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

                Dim dlg As New ColorDialog
                dlg.ShowDialog()

                If dlg.ShowDialog = Windows.Forms.DialogResult.OK Then
                        Dim str As String
                        str = dlg.Color.Name
                        MsgBox(str)
                End If

        End Sub
End Class
```

# KeyPress event in VB.NET

## Handle Keyboard Input at the Form Level in VB.NET

Windows Forms processes keyboard input by raising keyboard events in response to Windows messages. Most Windows Forms programs process keyboard input by handling the keyboard events.

## How do I detect keys pressed in VB.NET

You can detect most physical key presses by handling the KeyDown or KeyUp events. Key events occur in the following order:

```
KeyDown
KeyPress
KeyUp
```

## VB.Net KeyPress Event



```
Public Class Form1
    Private Sub TextBox1_KeyPress(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles TextBox1.KeyPress
        If e.KeyChar = Convert.ToChar(13) Then
            MsgBox("enter key pressd ")
        End If
    End Sub
End Class
```

## VB.Net KeyDown Event

```
Public Class Form1
    Private Sub TextBox1_KeyDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyEventArgs) Handles TextBox1.KeyDown
        If e.KeyCode = Keys.Enter Then
            MsgBox("enter key pressd ")
        End If
    End Sub
End Class
```

## CODE TO DETECT ARROW KEYS:

```
Protected Overrides Function ProcessCmdKey(ByRef
msg As Message, ByVal keyData As Keys) As Boolean
    'detect up arrow key
    If keyData = Keys.Up Then
        MessageBox.Show("You pressed Up arrow key")
        Return True
    End If
    'detect down arrow key
    If keyData = Keys.Down Then
        MessageBox.Show("You  pressed  Down  arrow
key")
        Return True
    End If
    'detect left arrow key
    If keyData = Keys.Left Then
        MessageBox.Show("You  pressed  Left  arrow
key")
        Return True
    End If
    'detect right arrow key
    If keyData = Keys.Right Then
        MessageBox.Show("You  pressed  Right  arrow
key")
        Return True
    End If
    Return MyBase.ProcessCmdKey(msg, keyData)
End Function
```