Hướng dẫn lập trình VB.NET

Chương 1:

Mở và chạy một chương trình Visual Basic.NET

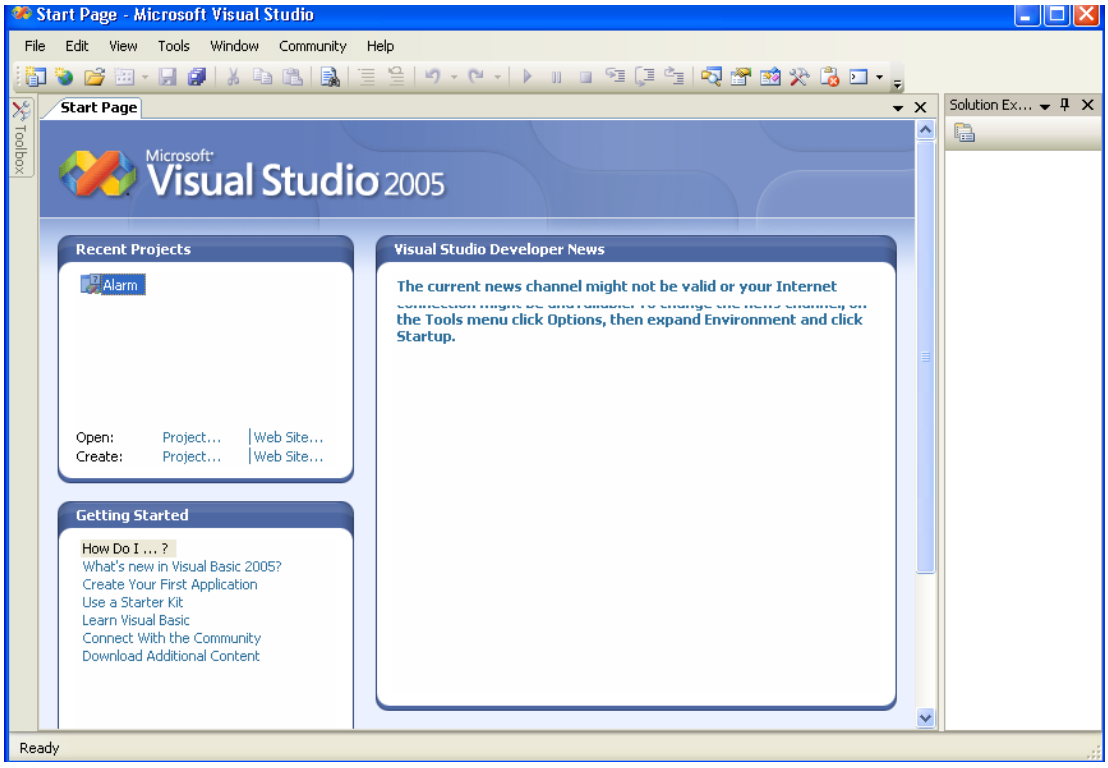
--------oOo--------

1. Môi trường phát triển visual studio.net

VS.NET là bộ công cụ giúp phát triển các ứng dụng từ ứng dụng desktop đến ứng dụng mạng. Nó là môi trường để phát triển tất cả các ngôn ngữ như VB.NET, Visual C^{++}, Visual C#.Net hết sức mạnh mẽ và thuận lợi.

1.1. Khởi động visual studio.NET

Việc khởi động vs.NET cũng tương tự như các phần mềm khác như ms.Word hay excel. Nếu lần đầu khởi động vs.NET sẽ yêu cầu xem bạn chạy nó với ưu tiên ứng dụng và ngôn ngữ nào. Bạn chọn Visual Basic và *start vs.net* là xong. Màn hình bắt đầu như sau:



Nếu trang start page không hiện, bạn có thể làm nó xuất hiện bằng cách chọn menu View | Other Windows | Start Page.

1.2. Mở một dự án của visual basic

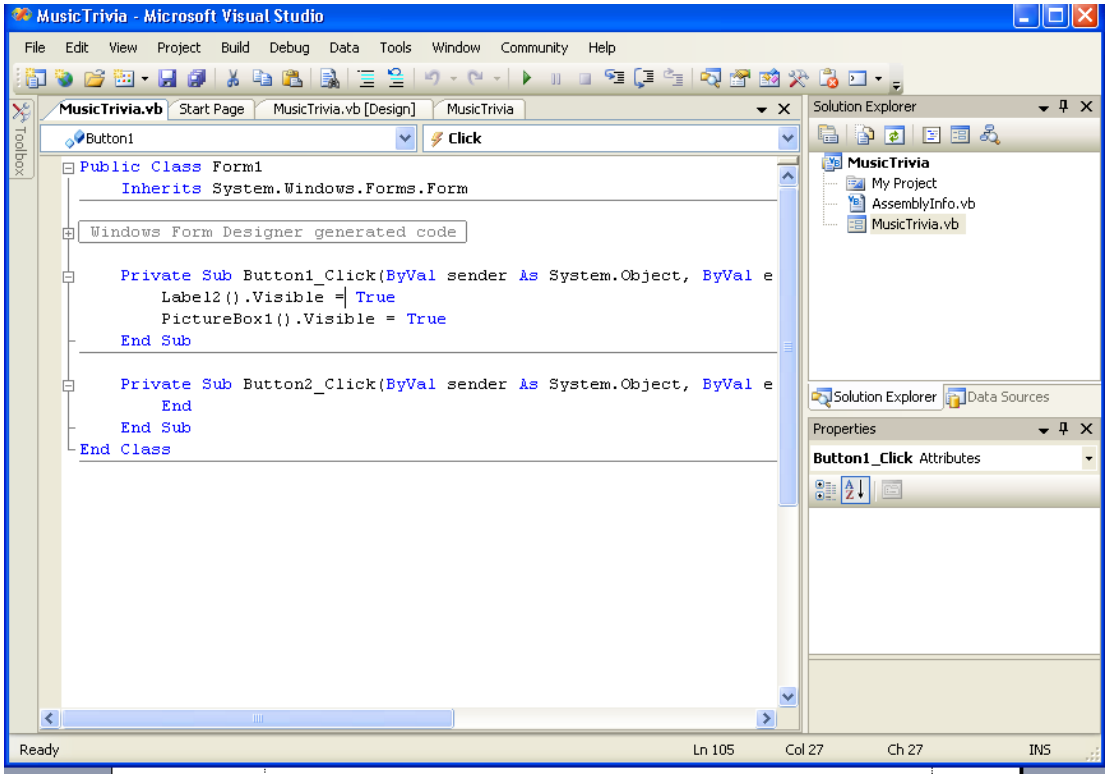
Tại trang start page, bạn có thể click vào *project* của phần *Open* và duyệt một dự án trong phần *Baitap* sẵn có của chương 1 xem sao.

1.3. Mở một dự án cụ thể

- Click chuột vào *project* của phần *Open* tại trang *Start Page*.

- Duyệt đến thư mục chứa dự án của chương 1

- Mở file *MusicTrivia.sln*. Khi này trang *Start Page* sẽ tạm ẩn đi, một cửa sổ mới xuất hiện:



1.4. Dự án (Project) và giải pháp (Solutions)

Khi mở dự án ở trên chắc hẳn bạn đã nhìn thấy hai file là MusicTrivia.sln và MusicTrivia.vbproj. file .sln là file giải pháp và file .vbproj là file dự án.

Vậy phân biệt chúng thế nào?

Trong VS, các chương trình đang triển khai và phát triển được gọi là dự án (Projects) hoặc giải pháp (Solution) bởi chúng chứa rất nhiều file và do nhiều thành phần, đối tượng riêng lẻ hợp lại. Một chương trình vs.NET bao gồm một file giải pháp và một file dự án hợp lại. File dự án chứa thông tin đặc biệt liên quan đến một tác vụ lập trình đơn lẻ. File giải pháp lại chứa thông tin về một hay nhiều dự án.

Như vậy về tương lai thì file .sln sẽ được ưa chuộng hơn.

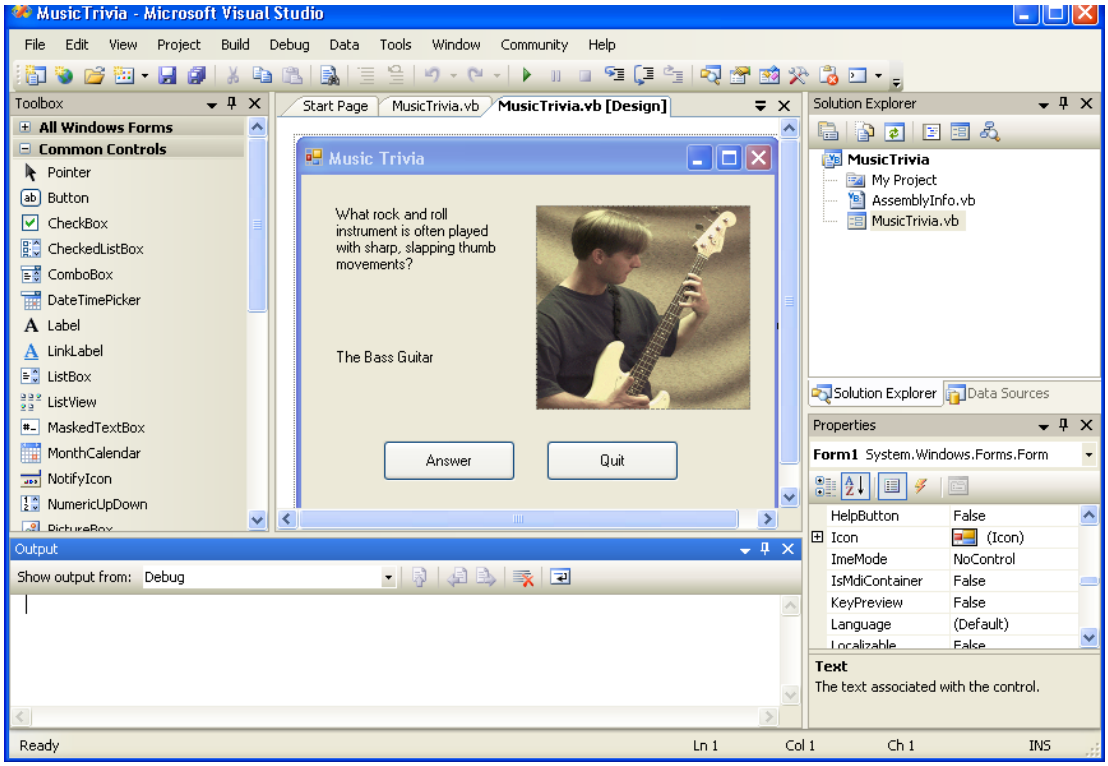
1.5. Các công cụ của VS.NET

Công cụ trong vs.NET rất phong phú. Bạn có thể khám phá từ từ. Sau đây là mô phỏng màn hình làm việc của bộ vs.NET:

Windows Form Designer

Menu Bar Standard Toolbar Solution Explorer

Tools Box



Output Windows Properties Windows

1.6. Bộ thiết kế Windows Forms Designer

VS.NET có một bộ thiết kế form trực quan, bạn sẽ làm quen dần dần bởi vì thường trong các dự án phát triển có thể có rất nhiều Form.

1.7. Hiển thị bộ thiết kế Form

Góc phải màn hình là cửa sổ Solution Explorer. Hiển thị nó View | Solution Explorer. Cửa sổ này chứa toàn bộ các phần tử có sd trong dự án.

Double Click vào MusicTrivia.vb trong cửa sổ Solution Explorer bạn sẽ thấy tất cả các file chứa form.

Nhấp chuột vào biểu tượng View Designer trong solution để hiển thị form thiết kế ở dạng trực quan.

2. Chạy một chương trình Visual Basic

- Nhấp chuột vào nút *start* màu xanh trên *standard bar* để chạy chương trình (bạn cũng có thể ấn phim F5).

- Ấn thử nút *Answer* và *Quit* thử xem.

3. Cửa sổ thuộc tính Properties

Cho phép thay đổi thông số của đối tượng thiết kế form sau này.

Bạn thử mở giao diện chương trình MusicTrivia và click vào một phần tử bất kỳ rồi thay đổi thử các thuộc tính của chúng xem sao.

4. Di chuyển và thay đổi kích thước cửa sổ công cụ lập trình

Tất cả các cửa sổ của bộ công cụ vs.NET đều có thể di chuyển cũng như thay đổi được, bạn thử xem.

5. Xem trợ giúp

Bạn có thể xem trợ giúp trực tuyến hay cài bộ MSDN để xem trợ giúp. Có nhiều cách xem trợ giúp khác nhau.

Bạn có thể dần dần tìm hiểu.

6. Thoát khỏi VISUAL STUDIO.NET

- Lưu lại những gì chúng ta làm bằng cách chọn *File* | *Save all*.
- Chọn *File* | *Exit* để thoát khỏi vs.NET.

7. Tổng kết

Bạn hãy làm một bảng tổng kết những công việc đã làm và thực hiện trong chương 1.

Chương 2:

Viết một chương trình Visual Basic.NET đầu tay

--------oOo--------

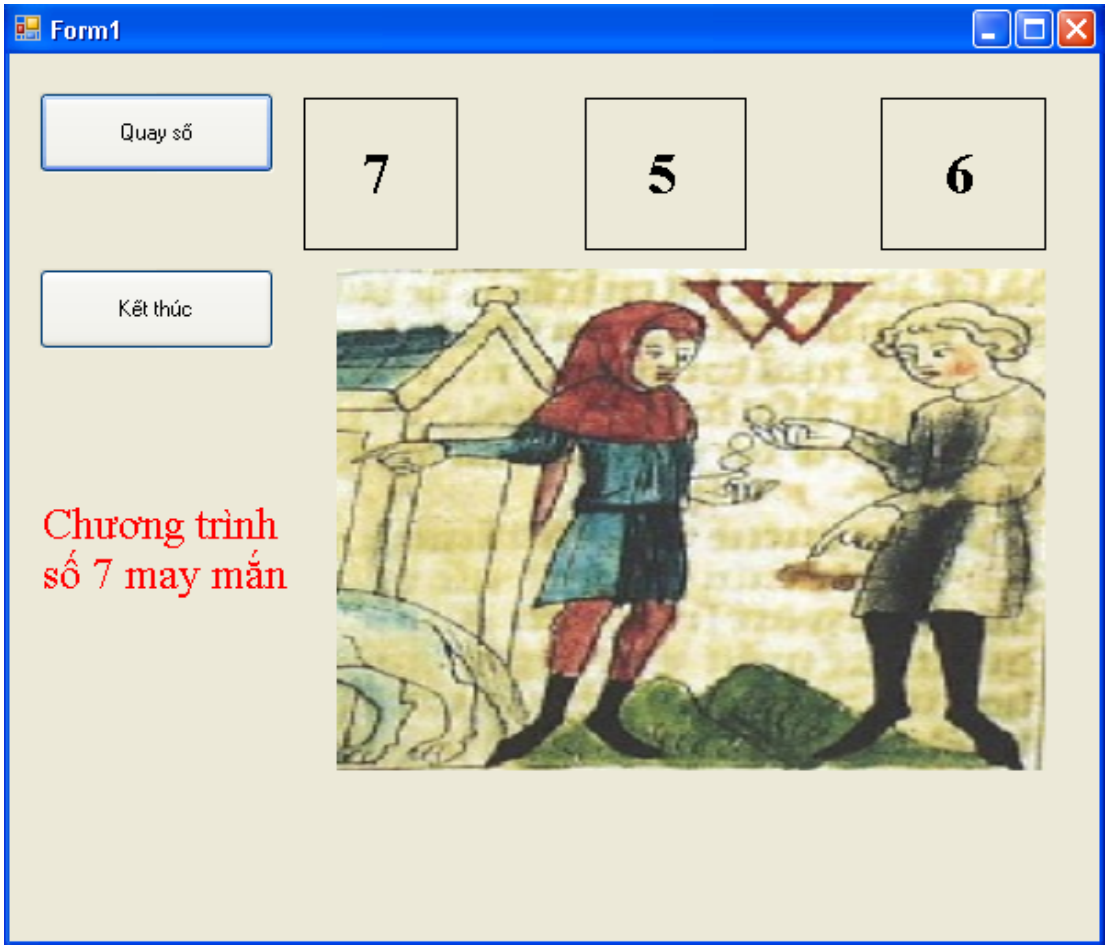
Nội dung thảo luận:

- Tạo giao diện cho chương trình
- Thiết lập thuộc tính cho các đối tượng trong giao diện
- Viết mã chương trình
- Lưu và chạy chương trình
- Biên dịch file thực thi .exe

1. Chương trình LUCKY SEVEN – chương trình đầu tay

1.1. Tìm hiểu chương trình

Luckyseven có giao diện như sau:



Form chính gồm có hai nút (quay số và kết thúc), bốn nhãn (1, 2, 3 – chứa ba số ngẫu nhiên, 4 – chứa tên chương trình và hiện dòng "Bạn chiến thắng nếu cả 3 nhãn 1, 2, 3 đều là số 7").

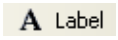
Khi bạn click vào nút *Quay số* thì chương trình phát sinh ngẫu nhiên ba số ở ba nhãn. Nếu một trong ba số là chữ số 7 thì hiện ảnh trả tiền ở đối tượng *picturebox1*.

1.2. Xây dựng giao diện và thuộc tính

- Tạo nút *button1* trên form1: bạn tạo *button1*theo nhiều cách khác nhau. Đó là kéo từ toolbox vào form; double click vào đối tượng button; click vào đối tượng *button* và vẽ hình chữ nhật trên giao diện chính của form1.

Sau khi tạo xong *button1* trên form1 bạn đặt thuộc tính như sau: R-click vào *button1* trên form1 chọn *properties*. Trong cửa sổ *properties windows* thiết lập các thuộc tính tùy thích (cẩn thận với thuộc tính *name* – đặt tên không khoảng trắng), bạn chọn thuộc tính *text* thành *Quay số*.

- Tương tự với *button2* bạn chọn thuộc tính *text* là *Kết thúc*. Cả hai nút thuộc tính *Text Align* đều là *Middle Center.*

- Tạo nhãn *label1:* Bạn tạo nhãn bằng nhiều cách như với nút nhưng chọn đối tượng *Label*  từ toolsbox. Bạn đặt con trỏ vào các cạnh của nhãn để chỉnh size cho nó. Nếu không chỉnh được thì bạn nhìn thuộc tính *Auto Size* của nhãn này trên cửa sổ *Properties*, chỉnh nó thành *False* là xong.

Sau khi tạo xong *lable1* trên form1, bạn đặt thuộc tính cho nó như sau: Text – để trống; *TextAlign – Middle center*; Các thuộc tính khác tùy thích.

- Tương tự với các nhãn *lable2, lable3, lable4*. Riêng *lable4* bạn đặt thuộc tính *text* là "Chương trình số 7 may mắn".

- Tạo *Picturebox1* – đối tượng cho phép chứa ảnh: Tạo *picturebox1* tương tự như tạo các đối tượng khác với cách click vào đối tượng  trên Tools box.

Thiết lập thuộc tính cho *Picturebox1*: SizeMode – StretchImage (cho phép ảnh co giãn đúng theo kích cỡ của *Picturebox*); Visible – False (ảnh không hiện trừ khi mã chương trình cho phép); Image – bạn chọn ảnh nào tùy thích.

Bạn có thể kéo vị trí các đối tượng trên form1 sao cho phù hợp.

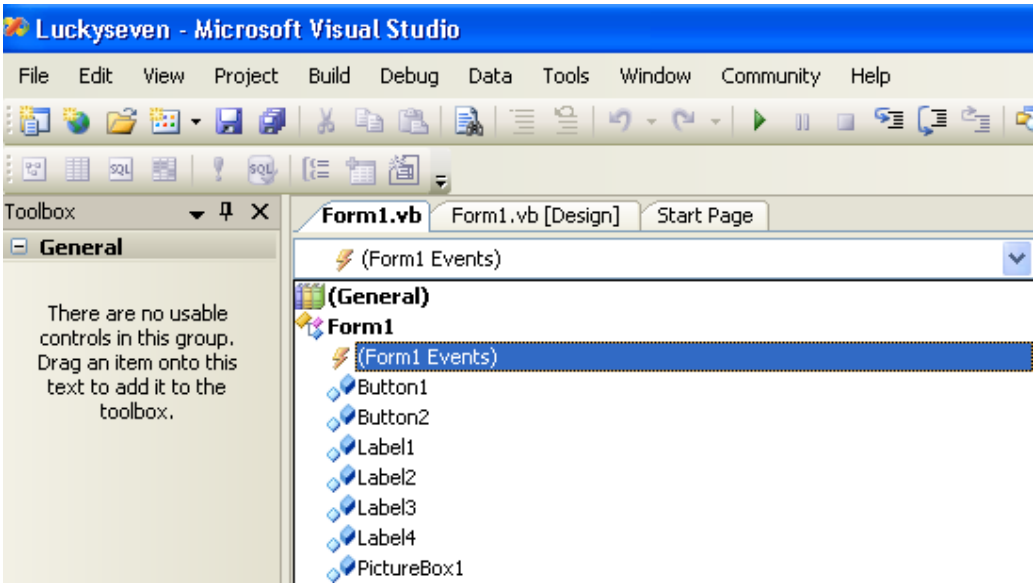
1.3. Viết mã cho chương trình

1.3.1. Sự kiện *Form1_Load*

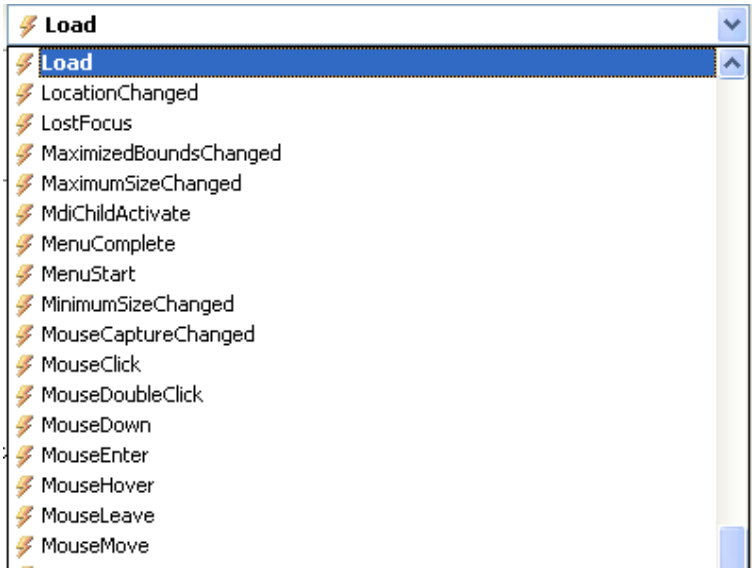
Mã là phần quan trọng và mạnh mẽ nhất dùng để tùy biến và xây dựng chương trình. Để xem mã của form1 ta R-click vào phần Form1 và chọn *ViewCode*. Kết quả:

```
Public Class Form1

End Class
```

Đây là cấu trúc đặc trưng của vb. Ta tiếp tục bàn về thủ tục *form_load*. *Load* là sự kiện triệu gọi một form khi thực thi. Để tạo bạn chọn *form1events* từ danh sách xổ xuống như sau:



Tiếp đó là chọn sự kiện *load* từ danh sách xổ xuống kế bên phần chọn sự kiện:



Và vs.net tự tạo một thủ tục cho bạn như sau:

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

End Sub
```

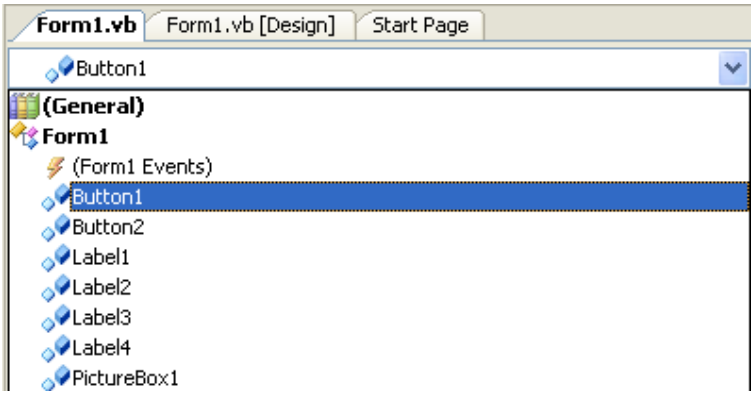
Vì đây là chương trình sinh số ngẫu nhiên nên bạn cần gọi đến hàm *rnd()* – hàm sinh ngẫu nhiên. Cũng theo đó, ta khai báo trong sự kiện *form1_load* hàm `Randomize()`:

```
Private Sub Form1_Load(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles Me.Load
        Randomize()
```

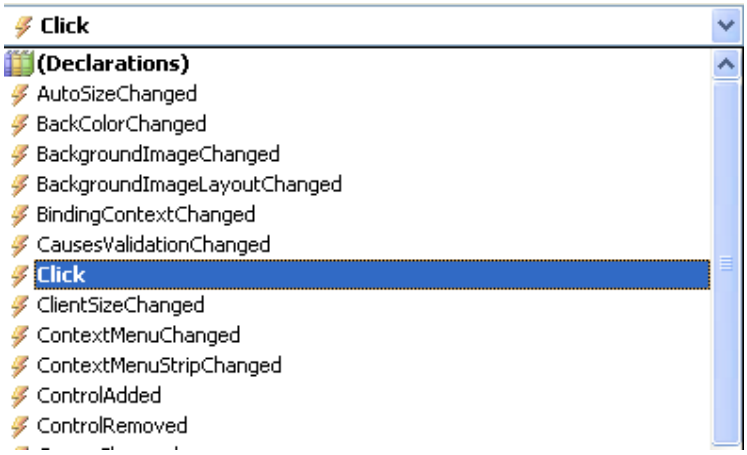
```
End Sub
```

1.3.2. Viết mã cho *nút nhấn quay số - button1*

Ta muốn chương trình thực hiện quay số ngẫu nhiên khi click vào nút *quay số* thì phải viết mã hay chính xác hơn là tạo thủ tục có tên `Button1_Click` xử lý sự kiện.Việc tạo thủ tục này như sau: bạn tiến hành một trong các cách. Thứ nhất, double click vào nút *quay số* trên giao diện thiết kế form. Cách thứ hai chọn đối tượng *button1* từ danh sách xổ xuống:



Tiếp theo chọn sự kiện *click* bên danh sach xổ xuống bên cạnh:



Và nhập chính xác đoạn mã sau vào phần thủ tục tương ứng xuất hiện:

```
Private Sub Button1_Click(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
        PictureBox1.Visible = False
        Label1.Text = CStr(Int(Rnd() * 10))
        Label2.Text = CStr(Int(Rnd() * 10))
        Label3.Text = CStr(Int(Rnd() * 10))
        'Nếu một trong ba nhãn xuất hiện số 7 thì hiện ảnh và kêu beep

        If (Label1.Text = "7") Or (Label2.Text = "7") Or (Label3.Text =
"7") Then
            PictureBox1.Visible = True
            Beep()
        End If
        If (Label1.Text = "7") And (Label2.Text = "7") And (Label3.Text =
"7") Then
            PictureBox1.Visible = True
```

```
            Beep()
            Label4.Text &= "Bạn đã chiến thắng!"
        End If
    End Sub
```

1.3.3. Viết mã cho nút *kết thúc – button2*

Tương tự như *button1*, bạn tạo sự kiện click của button2 và nhập hàm *End()* vào là xong.

1.3.4. Mã đầy đủ của chương trình

```
Public Class Form1
    Private Sub Button2_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button2.Click
        End
    End Sub

    Private Sub Button1_Click(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
        PictureBox1.Visible = False
        Label1.Text = CStr(Int(Rnd() * 10))
        Label2.Text = CStr(Int(Rnd() * 10))
        Label3.Text = CStr(Int(Rnd() * 10))
        'Nếu một trong ba nhãn xuất hiện số 7 thì hiện ảnh và kêu beep

        If (Label1.Text = "7") Or (Label2.Text = "7") Or (Label3.Text =
"7") Then
            PictureBox1.Visible = True
            Beep()
        End If
        If (Label1.Text = "7") And (Label2.Text = "7") And (Label3.Text =
"7") Then
            PictureBox1.Visible = True
            Beep()
            Label4.Text &= "Bạn đã chiến thắng!"
        End If
    End Sub

    Private Sub Form1_Load(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles Me.Load
        Randomize()
    End Sub
End Class
```

2. Thực thi chương trình

2.1. Chạy chương trình

Để chạy chương trình click vào nút start  trên *standard bar* hay chọn *Debug* | *start debugging* từ menu bar.

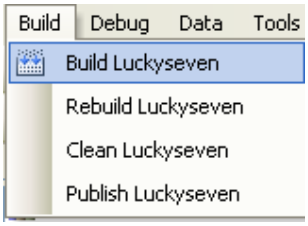
2.2. Biên dịch chương trình ra file .exe

VS.NET hỗ trợ bạn biên dịch chương trình ra file .exe để chạy trên bất kỳ môi trường nào. Nó đóng gói tất cả các thành phần cần thiết và tạo ra file chạy trên tất cả môi trường windows.

Có hai kiểu file chạy: kiểu *Debug build* (gỡ lỗi) và *release build* (xây dựng).

Trên lý thuyết, kiểu *debug build* chạy chậm hơn vì chứa thông tin gỡ lỗi. Trên thực tế thì chúng chạy tương đương nhau.

Để tạo ra file thực thi chọn *Build* | *Build luckyseeven*.



Chương trình sẽ tạo ra một thư mục BIN chứa hai thư mục con là DEBUG và RELEASE có hai *file luckyseven.exe* là hai file thực thi ta cần.

3. Tổng kết

Bạn hãy làm một bảng tổng kết các công việc đã thực hiện và tự làm lại chương trình không cần xem mẫu.

Chương 3:

Làm việc với các điều khiển trên TOOLBOX

--------oOo--------

Nội dung thảo luận:

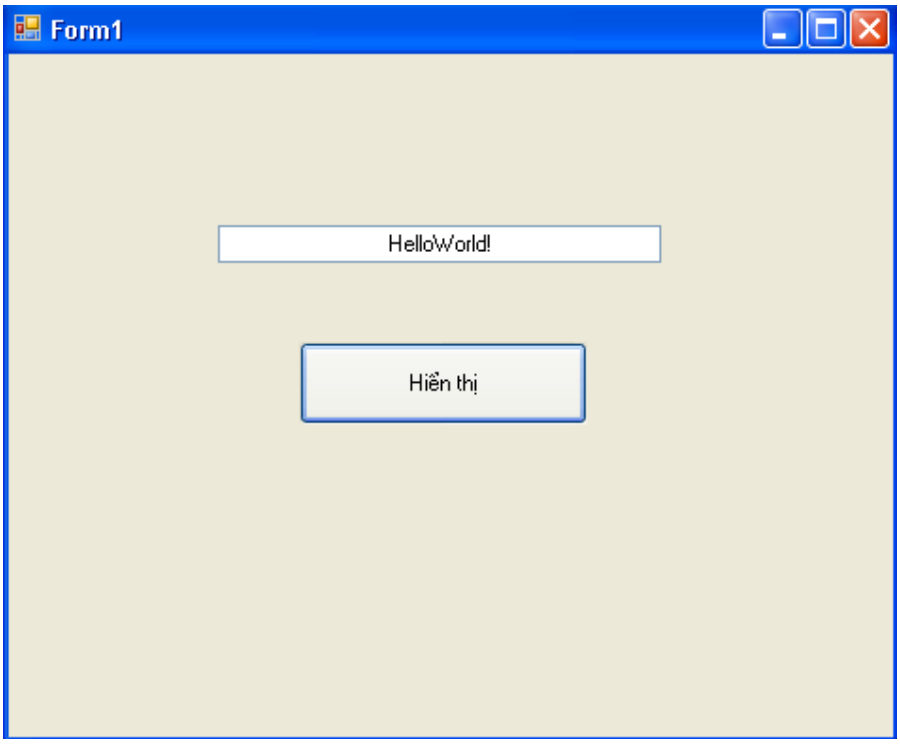
- Sử dụng các điều khiển *Textbox* và *Button* tạo chương trình *Hello World*
- Sử dụng điều khiển *DateTimePicker* hiển thị ngày sinh của bạn
- Sử dụng *combobox, CheckBox, RadioButton, ListBox* để xử lý các nhập liệu của người dùng
- Sử dụng điều khiển *LinkLabel* để hiển thị trang web trên Internet
- Cài đặt điều khiển *ActiveX*

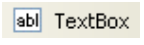
TOOLBOX chứa các điều khiển để thiết kế form chương trình. Bạn có thể sử dụng những điều khiển đã có sẵn hay thêm một vài thành phần vào đó để sử dụng sau này.

1. Xây dựng chương trình *HELLOWORLD*

1.1. Tìm hiểu chương trình

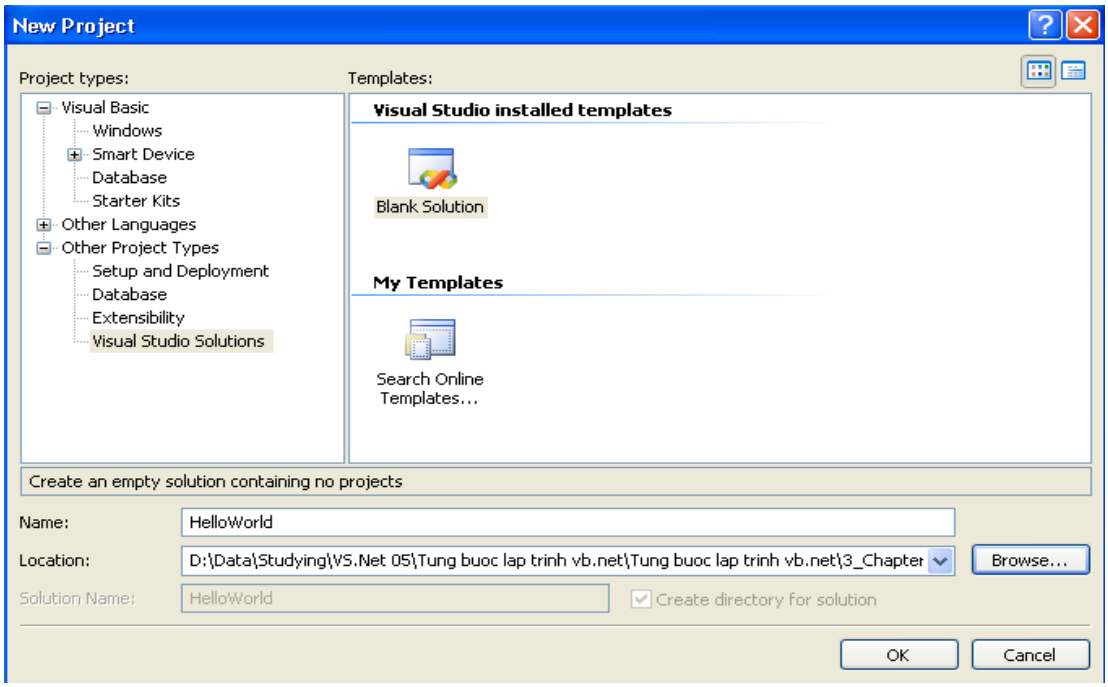
Giao diện chính của chương trình như sau:



Nó bao gồm một textbox – ô cho phép nhập chuỗi ký tự có biểu tượng  trên TOOLBOX và một button. Khi chương trình chạy, click vào button hiển thị để textbox hiện dòng chữ "HelloWorld" – Xin chào thế giới.

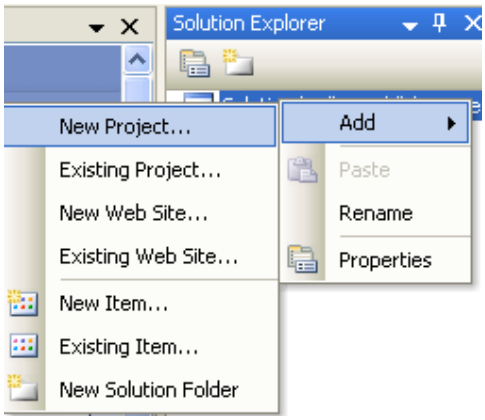
1.2. Thiết kế chương trình

Bạn tạo mới một dự án như đã học. Tại trang *start page* chọn tạo mới một *Visual Studio Solution.* Nhập tên tại ô *Name* là HelloWorld, click vào nút *Browse* để chọn đường dẫn lưu dự án của mình.

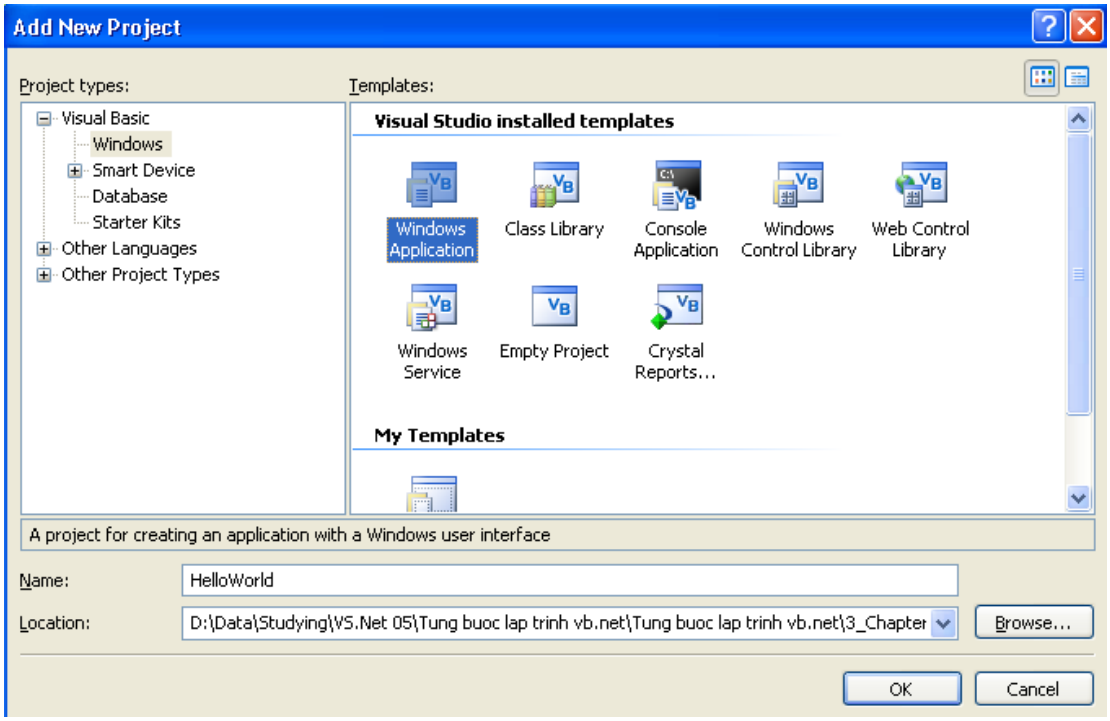


Nhấn OK để tạo.

Bây giờ bạn đã có một giải pháp trắng. Tiếp theo ta tạo mới một dự án từ giải pháp này. Để tạo R-click vào Solution vừa tạo chọn *Add | New Project*



Một cửa sổ hiện ra, click chọn *Windows Application* tại ô *Visual Studio Installed Template*. Nhập tên là *HelloWorld* tại ô Name, đường dẫn như đường dẫn chứa solution mới tạo.



Thiết kế:

- Tạo một Textbox (*textbox1*) và một Button (*Button1*) lên giao diện đồ họa của Form như đã biết

TextBox: điều khiển cho phép nhập một chuỗi các ký tự cho chương trình xử lý

Button: điều khiển cho phép chương trình có một hành động khi người dùng click lên nó khi chạy chương trình.

- Thiết lập thuộc tính cho các điều khiển: *Textbox1 – Text:Rỗng; Button1 – Text:Hiển thị.*

Viết mã:

- Tại giao diện chính của Form double click vào Button1 để chuyển qua chế độ viết mã, viết thủ tục *Button1_Click*

- Nhập đoạn mã sau vào đó:

```
TextBox1.Text = "HelloWorld!"
```

Khi bạn gõ textbox1 và dấu '.' thì chương trình tự xổ xuống một danh sách cho bạn chọn lựa, bạn chọn thuộc tính *text* (Enter).

Chạy chương trình:

Nhắp nút start như ví dụ trước là xong.

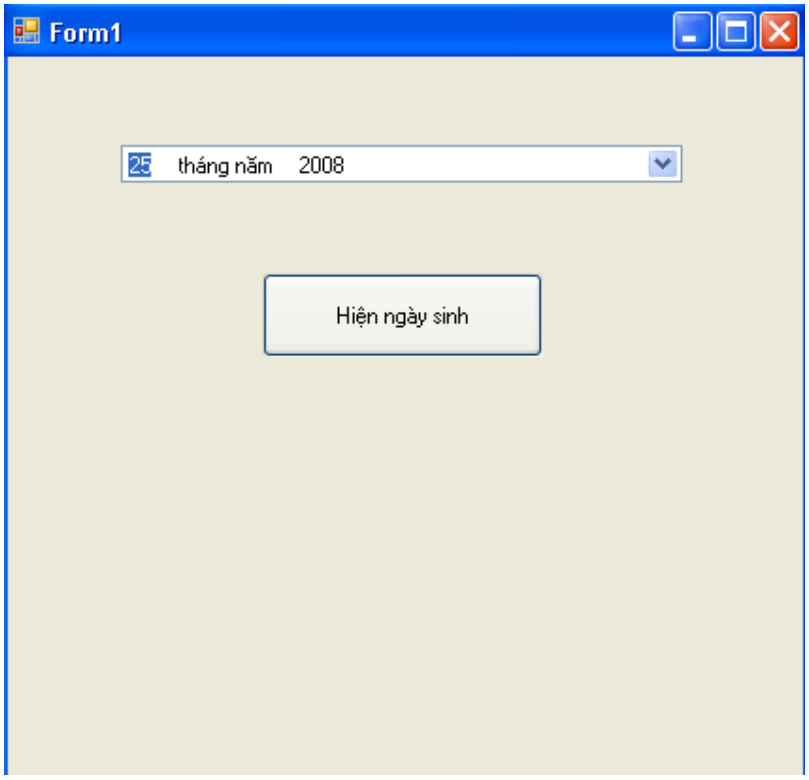
2. Sử dụng điều khiển DATETIMEPICKER

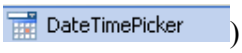
DATETIMEPICKER, điều khiển cho phép người dùng chọn thời gian dưới dạng giao diện của lịch biểu.

2.1. Chương trình Birthday

2.1.1. Tìm hiểu chương trình

Giao diện của chương trình:



Chương trình có một điều khiển *DaeTimePicker* (trên TOOLBOX là điều khiển có hình 

) cho phép người dùng chọn một ngày bất kỳ để chương trình xử lý và một nút *Button1* sẽ thực hiện đưa ra một hộp thông báo bằng *MsgBox* hiển thị ngày mà người dùng đã chọn.

2.1.2. Xây dựng giao diện

Bạn cũng tạo mới một Solution trắng có tên *Birthday* và thêm một Project có tên tương tự ở dạng Windows Application trong ô Visual Studio Installed Template như ví dụ trước.

Tại giao diện thiết kế của form1 bạn thêm hai điều khiển là *DateTimePicker* và *Button1* vào, đặt thuộc tính *Text* cho *Button1* là 'Hiện ngày sinh'. Lưu lại tất cả những thiết đặt bằng cách nhấp chọn *Save All* trên Standard Bar. Nếu chương trình hiện ra một thông báo yêu cầu chọn chế độ lưu thì bạn chọn lưu với mã hóa 65001.

2.1.2. Viết mã cho chương trình

Bạn chỉ cần viết mã cho *Button1* để thực thi hành động hiện ra thông báo khi người dùng đã chọn ngày và click lên nó. Double click vào Button1 tại giao diện thiết kế form1 và nhập mã như sau:

```
MsgBox("Ngày sinh của bạn là: " & DateTimePicker1.Text)
MsgBox("Ngày trong năm: " & DateTimePicker1.Value.DayOfYear.ToString)
MsgBox("Hôm nay là ngày: " & Now.ToString)
```

Đoạn mã này sẽ hiển thị lần lượt ba thông báo có trong dấu ngoặc đơn. Dấu '&' để kết nối chuỗi như "Ngày sinh của bạn là" với nội dung là thuộc tính *Text* của điều khiển *DateTimePicker1*. Các hàm khác các bạn sẽ làm quen dần trong các chương sau.

2.1.3. Thực thi chương trình

Bạn thử chạy chương trình và chọn đúng ngày sinh của mình xem sao.

2.2. Làm quen với các thuộc tính khác của *DateTimePicker*

Bạn click vào đối tượng *DateTimePicker1* trên giao diện chính của form và chọn mở thuộc tính của nó.

Trên *Properties Windows* bạn thử thay đổi các thuộc tính của nó xem sao. Ví dụ, để cho nó hiển thị thông tin về giờ thay vì ngày tháng, bạn thay đổi thuộc tính *Format* của nó từ *long* qua *Time* xem sao.

3. Làm việc với các điều khiển nhập liệu

Trong suốt quá trình lập trình, thực tế ta luôn xuay quanh việc lập trình để xử lý các điều khiển nhập liệu. Các điều khiển nhập liệu gồm *TextBox* cho phép người dùng nhập vào một chuỗi các ký tự, menu thể hiện thông tin dưới dạng chọn lệnh, các loại hộp thoại như *Checkbox, ListBox, RadioButton, ComboBox* thể hiện thông tin dưới dạng tương tự như menu…

3.1. Tìm hiểu *CheckBox*

3.1.1 Sử dụng

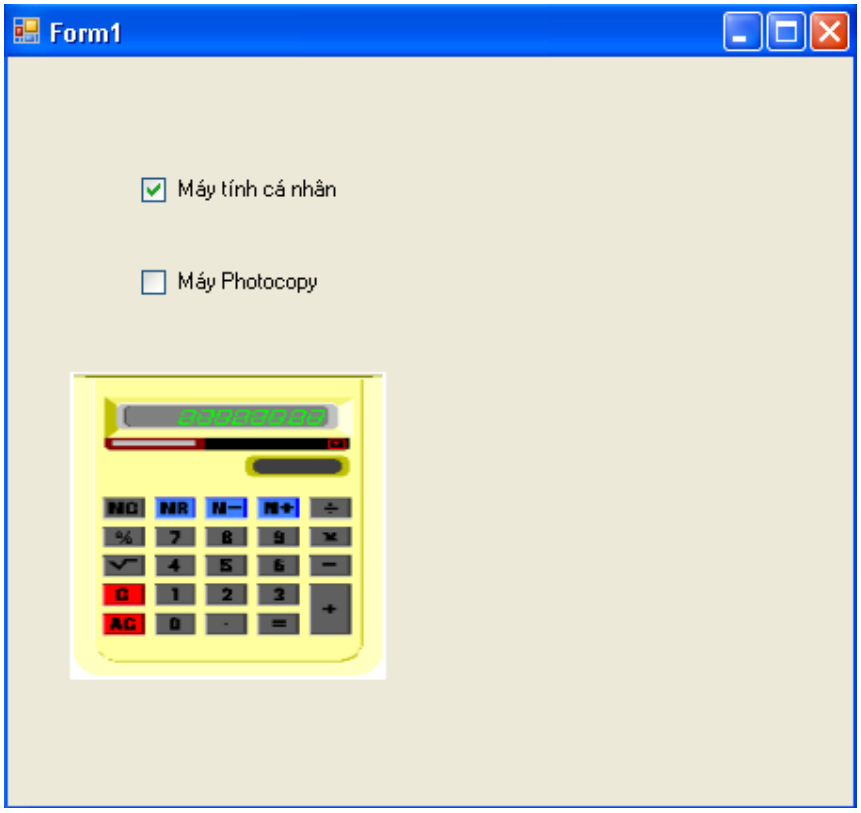
CheckBox là điều khiển cho phép người dùng chọn lựa khả năng xử lý của chương trình. Ta thử tìm hiểu kỹ hơn về điều khiển này qua bài tập sau:

3.1.2. Chương trình MyCheckBox

Tìm hiểu chương trình:

Chương trình này có hai *CheckBox* cho phép click chọn. Nếu click chọn vào *CheckBox* nào thì sẽ hiện một bức ảnh tương ứng với nó.

Giao diện chính của nó như sau:



Thiết kế giao diện:

Tạo một giải pháp mới và thêm vào đó một dự án như đã biết, đặt tên là *MyCheckBox*.

Các điều khiển sử dụng trong form gồm:

- CheckBox1: thuộc tính *Checked – False; Text – Máy tính cá nhân*
- Checkbox2: thuộc tính *Checked – False; Text – Máy photocopy*
- PictureBox1: thuộc tính *Image – None; SizeMode: StretchImage*
- PictureBox2: thuộc tính *Image – None; SizeMode: Stretchimage*

Viết mã chương trình:

Vì ta muốn khi người dùng click vào checkbox thì lập tức có thay đổi ẩn/hiện các ảnh ngay nên ta cần xây dựng thủ tục thể hiện sự thay đổi gắn với các checkbox. Trong vb thủ tục đó là thủ tục `CheckBox1_CheckedChanged` mà ta có thể tạo ra bằng cách nhắp đúp vào điều khiển checkbox từ giao diện thiết kế form hay lựa chọn từ danh sách xổ xuống như đã biết.

- Bạn double click vào điều khiển *Checkbox1* để tạo thủ tục `CheckBox1_CheckedChanged`. Sau đó nhập đoạn mã sau vào:

```
If CheckBox1.CheckState = 1 Then
            'PictureBox1.Visible = True
            PictureBox1.Image = System.Drawing.Image.FromFile _
            ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh vb.net\Tung
buoc lap trinh vb.net\3_Chapter3\Bai
tap\MyCheckBox\MyCheckBox\Images\Calcultr.bmp")
            PictureBox1.Visible = True
```

```
        Else
            PictureBox1.Visible = False
        End If
```

Chú ý: Dấu '_' ở dòng mã thứ 3 tư trên xuống là dấu cho phép xuống dòng khi cảm thấy dòng mã quá dài trong VB. Bức ảnh của các bạn muốn cho vào điều khiển *PictureBox1* không nhất thiết phải giống như trên. Bạn có thể copy một bức ảnh bất kỳ vào thư mục chứa dự án và kéo trực tiếp từ cửa sổ *Solution Explorer* vào trong đoạn mã để lây đường dẫn.

- Tương tự bạn tạo thủ tục `CheckBox2_CheckedChanged` như sau:

```
If CheckBox2.CheckState = 1 Then
            'PictureBox2.Visible = True
            PictureBox2.Image = System.Drawing.Image.FromFile _
            ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh vb.net\Tung
buoc lap trinh vb.net\3_Chapter3\Bai
tap\MyCheckBox\MyCheckBox\Images\CopyMach.bmp")
            PictureBox2.Visible = True
        Else
            PictureBox2.Visible = False
        End If
```

Chạy chương trình:

Bạn thử chạy chương trình xem sao.

3.2. Một số điều khiển khác

3.2.1. Sử dụng

Ta thử tìm hiểu tác dụng của một số điều khiển khác như *RadioButton, ComboBox, ListBox* ... qua ví dụ InputControls xem sao.

3.2.2 Chương trình InputControls

Tìm hiểu chương trình:

Chương trình này có 6 ô hiện ảnh tương ứng với 5 mặt hàng và một hiển thị đơn vị tiền mà người dùng sẽ chi trả khi mua hàng.

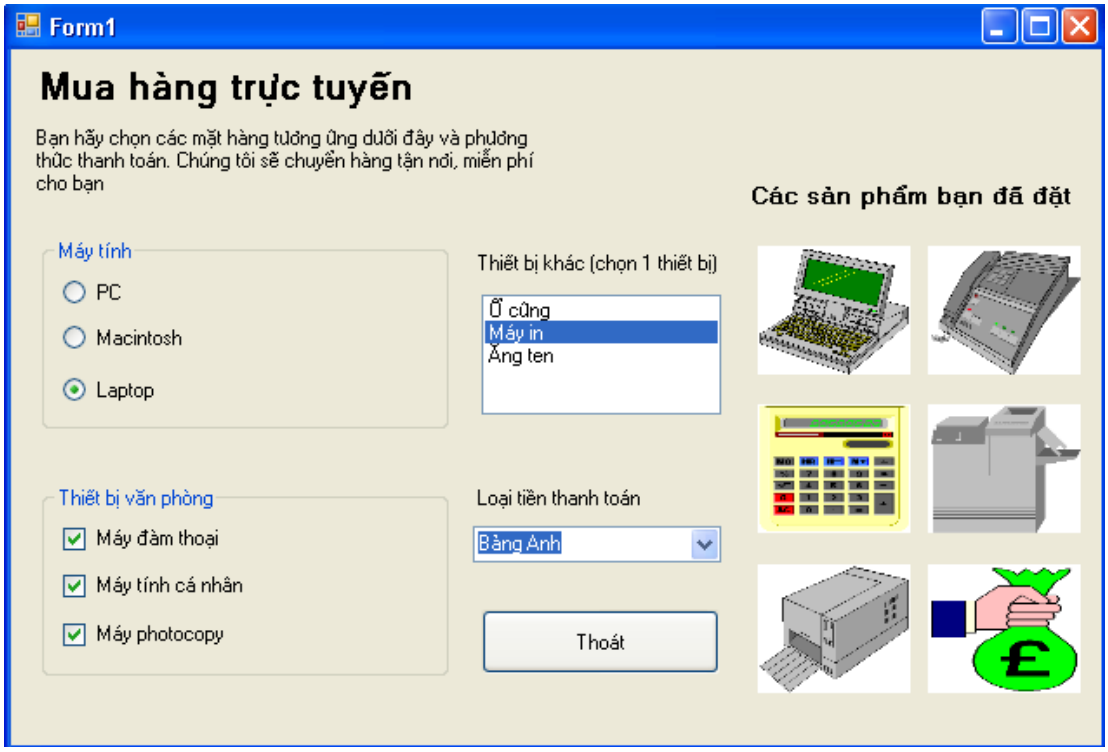
Ô thứ nhất sẽ hiển thị các sản phẩm tương ứng với một trong ba radiobutton đặt trong điêu khiển *GroupBox* – điều khiển cho phép đặt một số điều khiển khác vào (bạn thử tìm xem nó ở đâu trên TOOLBOX)

Ô thứ hai, thứ ba và thứ tư hiển thị các sản phẩm tương ứng với các mặt hàng chọn bởi các *checkbox* đặt trong *GroupBox2*.

Ô thứ 5 hiển thị 1 trong 3 sản phẩm được chọn bởi điều khiển *ListBox1*.

Ô thứ 6 hiển thị ảnh của đơn vị tiền tệ mà người dùng chọn bởi *ComboBox1*.

Sau đây là giao diện của chương trình:



Thiết kế giao diện:

- Tạo hai điều khiển *GroupBox*
- Tạo 3 *radiobox* đặt vào trong điều khiển *GroupBox1*.
- Tạo 3 *CheckBox* đặt vào trong điều khiển *GroupBox2*.
- Tạo 1 điều khiển *ListBox* và không nhập liệu gì cả.
- Tạo một điều khiển *ComboBox*.
- Tạo 6 *PictureBox* và 3 *Label* cùng 1 *Button*.
- Sửa các thuộc tính sao cho phù hợp với hình trên. Riêng hai điều khiển *ListBox* và *ComboBox* thì các dữ liệu sẽ được nhập khi Form được load vào lúc chương trình chạy.

Viết mã chương trình:

Dưới đây là toàn bộ mã của chương trình, bạn có thể tham khảo:

```
Public Class Form1

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

    Private Sub CheckBox1_CheckedChanged(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles CheckBox1.CheckedChanged
        If CheckBox1.CheckState = 1 Then
            PictureBox2.Image = System.Drawing.Image.FromFile _
            ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh vb.net\Tung
buoc lap trinh vb.net\3_Chapter3\Bai
tap\InputControls\InputContorls\Images\AnswMach.bmp")
            PictureBox2.Visible = True
```

```
        Else
            PictureBox2.Visible = False
        End If
    End Sub

    Private Sub CheckBox2_CheckedChanged(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles CheckBox2.CheckedChanged
        If CheckBox2.CheckState = 1 Then
            PictureBox3.Image = System.Drawing.Image.FromFile _
            ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh vb.net\Tung
buoc lap trinh vb.net\3_Chapter3\Bai
tap\InputControls\InputContorls\Images\Calcultr.bmp")
            PictureBox3.Visible = True
        Else
            PictureBox3.Visible = False
        End If
    End Sub

    Private Sub CheckBox3_CheckedChanged(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles CheckBox3.CheckedChanged
        If CheckBox3.CheckState = 1 Then
            PictureBox4.Image = System.Drawing.Image.FromFile _
            ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh vb.net\Tung
buoc lap trinh vb.net\3_Chapter3\Bai
tap\InputControls\InputContorls\Images\CopyMach.bmp")
            PictureBox4.Visible = True
        Else
            PictureBox4.Visible = False
        End If
    End Sub

    Private Sub ListBox1_SelectedIndexChanged(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles ListBox1.SelectedIndexChanged
        Select Case ListBox1.SelectedIndex
            Case 0
                PictureBox5.Image = System.Drawing.Image.FromFile _
                ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh
vb.net\Tung buoc lap trinh vb.net\3_Chapter3\Bai
tap\InputControls\InputContorls\Images\Harddisk.bmp")
                PictureBox5.Visible = True
            Case 1
                PictureBox5.Image = System.Drawing.Image.FromFile _
                ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh
vb.net\Tung buoc lap trinh vb.net\3_Chapter3\Bai
tap\InputControls\InputContorls\Images\Printer.bmp")
                PictureBox5.Visible = True
            Case 2
                PictureBox5.Image = System.Drawing.Image.FromFile _
                ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh
vb.net\Tung buoc lap trinh vb.net\3_Chapter3\Bai
tap\InputControls\InputContorls\Images\SateDish.bmp")
                PictureBox5.Visible = True
        End Select
    End Sub

    Private Sub Form1_Load(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles Me.Load
        ListBox1.Items.Add("Ổ cứng")
        ListBox1.Items.Add("Máy in")
        ListBox1.Items.Add("Ăng ten")
```

```
        ComboBox1.Items.Add("USD")
        ComboBox1.Items.Add("Kiểm tra")
        ComboBox1.Items.Add("Bảng Anh")
    End Sub

    Private Sub RadioButton1_CheckedChanged(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles RadioButton1.CheckedChanged
        PictureBox1.Image = System.Drawing.Image.FromFile _
        ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh vb.net\Tung buoc
lap trinh vb.net\3_Chapter3\Bai
tap\InputControls\InputContorls\Images\PComputr.bmp")
        PictureBox1.Visible = True
    End Sub

    Private Sub RadioButton2_CheckedChanged(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles RadioButton2.CheckedChanged
        PictureBox1.Image = System.Drawing.Image.FromFile _
        ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh vb.net\Tung buoc
lap trinh vb.net\3_Chapter3\Bai
tap\InputControls\InputContorls\Images\Computer.bmp")
        PictureBox1.Visible = True
    End Sub

    Private Sub RadioButton3_CheckedChanged(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles RadioButton3.CheckedChanged
        PictureBox1.Image = System.Drawing.Image.FromFile _
        ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh vb.net\Tung buoc
lap trinh vb.net\3_Chapter3\Bai
tap\InputControls\InputContorls\Images\Laptop1.bmp")
        PictureBox1.Visible = True
    End Sub

    Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles ComboBox1.SelectedIndexChanged
        Select Case ComboBox1.SelectedIndex
            Case 0
                PictureBox6.Image = System.Drawing.Image.FromFile _
                ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh
vb.net\Tung buoc lap trinh vb.net\3_Chapter3\Bai
tap\InputControls\InputContorls\Images\Dollar.bmp")
                PictureBox6.Visible = True
            Case 1
                PictureBox6.Image = System.Drawing.Image.FromFile _
                ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh
vb.net\Tung buoc lap trinh vb.net\3_Chapter3\Bai
tap\InputControls\InputContorls\Images\Check.bmp")
                PictureBox6.Visible = True
            Case 2
                PictureBox6.Image = System.Drawing.Image.FromFile _
                ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh
vb.net\Tung buoc lap trinh vb.net\3_Chapter3\Bai
tap\InputControls\InputContorls\Images\PoundBag.bmp")
                PictureBox6.Visible = True
        End Select
    End Sub
End Class
```

Như ví dụ trước, các ảnh bạn có thể tùy biến.

Chạy chương trình:

Bạn thử chạy chương trình xem sao.

Đề nghị:

Bạn thử thiết kế và viết toàn bộ mã lại, có thể theo ý mình xem.

4. Sử dụng điều khiển LINKLABEL

4.1. Sử dụng

Cho phép mở trình duyệt web IE hay Netscape truy cập một trang web.

Trong ví dụ dưới đây *WebLink* chúng ta sẽ thử tạo chương trình sử dụng *LinkLabel* để hiển thị chuỗi văn bản trong Form, kết hợp sự kiện *Click* với phương thức *Process.Start* để mở trang web.

4.2. Chương trình *WebLink*

4.2.1. Tìm hiểu chương trình

Giao diện chương trình:

Chương trình chỉ có môt điều khiển *LinkLabel* cho phép ta mở một trang web bất kỳ. Trong ví dụ này ta mở trang web cục bộ http://localhost/localstart.asp. Ở đây có thể thay bằng địa chỉ IP là 127.0.0.1.

4.2.2. Thiết kế giao diện

Bạn tạo mới một dự án và tạo form như đã biết.

4.2.3. Viết mã cho chương trình

Bạn mở chế độ Code Editor để viết thủ tục `LinkLabel1_LinkClicked` xử lý khi người dùng click vào *LinkLabel1* bằng cách double click vào điều khiển *LinkLable1* trên form1. Tiếp theo nhập chính xác đoạn mã sau:

```
LinkLabel1.LinkVisited = True
System.Diagnostics.Process.Start _
("http://127.0.0.1/localstart.asp")
```

Vậy là chương trình của chúng ta đã có thể thực thi.

4.2.4. Chạy chương trình.

Chương trình tương đối đơn giản, bạn thử chạy xem. Nếu nhà có nối mạng, bạn có thể cho một trang web nào trực tuyến thay vì trang cục bộ. Bạn cũng cần cấu hình *Default Website* để không có bất cứ trục trặc nào khi chạy chương trình.

4.2.5. Hiểu thêm về mã chương trình

- Dòng 1: `LinkLabel1.LinkVisited = True`

Dòng này có tác dụng đánh dấu màu thể hiện người dùng đã duyệt qua liên kết này nhờ thuộc tính *LinkVisited*.

- Dòng 2: kết hợp sự kiện click với phương thức *Process.Start* để mở trang web.

5. Cài đặt điều khiển ACTIVEX

Visual Studio.NET là một sản phẩm hoàn toàn mới của Microsoft. Các chương trình trước đây thường dựa trên công nghệ COM (Component Object Model). Và .NET không còn dựa vào mô hình COM nữa nhưng nó vẫn cho phép ta tái sử dụng chúng cũng như đưa những đối tượng COM, ACTIVEX cũ vào cửa sổ TOOLBOX để sử dụng như một điều khiển của VS.NET.

Các điều khiển activeX hay đối tượng COM thường chứa trong các file .exe hay .dll. Khi bạn cài đặt một số chương trình, ứng dụng thì thường cũng đăng ký theo chúng vào hệ

thống, ví dụ như khi cài Microsoft Word chẳng hạn, có một điều khiển ActiveX giúp vẽ biểu đồ có tên Microsoft Chart.

Trong bài tập dưới đây chúng ta sẽ đưa ActiveX *Microsoft Chart* vào TOOLBOX của VS.NET để sử dụng.

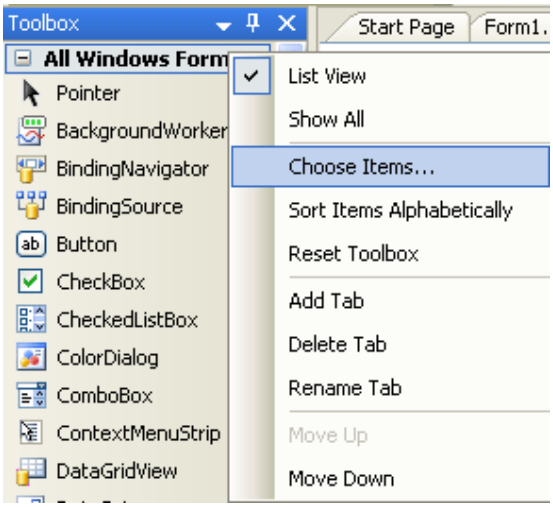
Nếu bạn nào xây dựng diễn đàn bằng ngôn ngữ ASP.NET thì cũng có thể thêm điều khiển *FreeTextBox* đã xây dựng sẵn vào TOOLBOX và sử dụng để các thành viên đăng tải bài viết lên diễn đàn khá tiện ích.

Cài đặt ActiveX :

- Để đưa được một điều khiển ActiveX vào thì trước hết TOOLBOX phải hiện lên tức là phải có một dự án đang mở và mở ở chế độ thiết kế form.

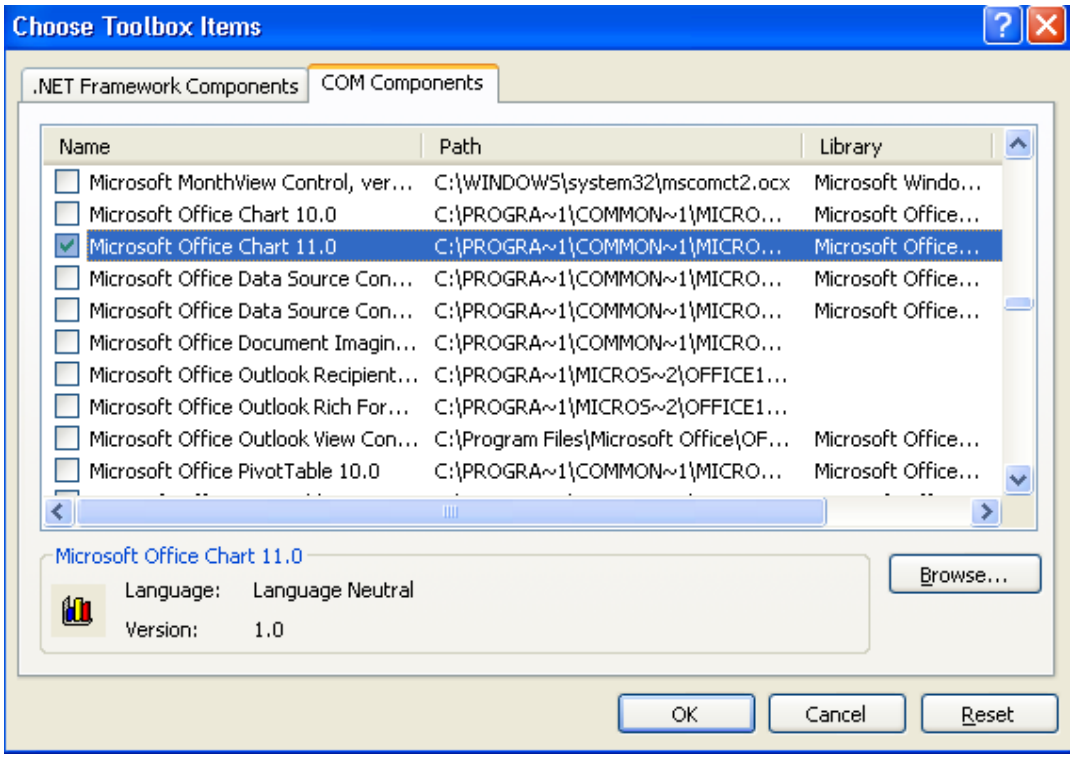
- Khi đã làm hiện TOOLBOX bạn chọn một tab bất kỳ muốn cho thêm ActiveX vào, mình chọn tab chuẩn *Windows Forms*.

- R-Click vào TOOLBOX và chọn *Choose Items*... như hình:

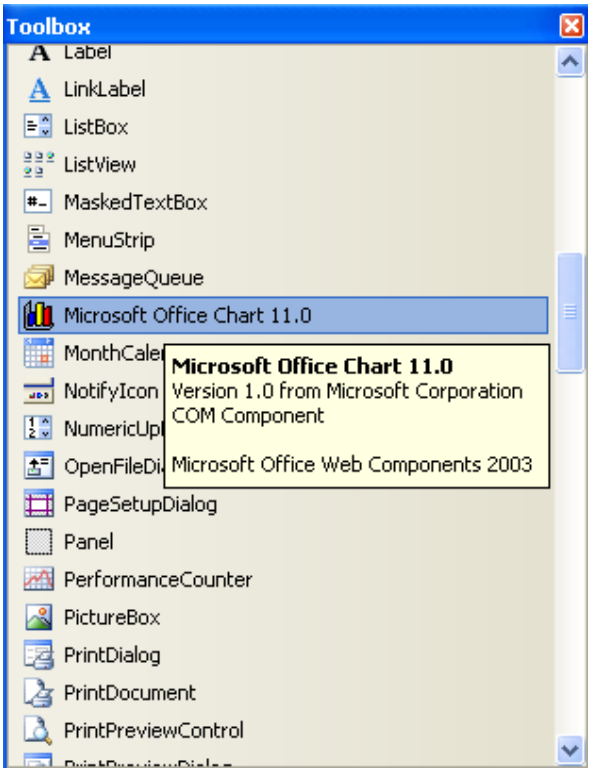


- Một cửa sổ hiện ra cho phép ta chọn các thành phần muốn thêm vào tùy thích. Có thể chọn thành phần của .Net Frame Works, COM Components hay là chọn một điều khiển nào bạn sẵn có bằng click chọn nút *Browse* để duyệt. Ở đây ta chọn .COM Components và duyệt đến điều khiển ActiveX *Microsoft Office Chart 11.0* để thêm vào. Nhấp OK và chờ xem kết quả.

Hình minh họa:



Và bây giờ trên TOOLBOX đã có thêm điều khiển mới cho ta thực hiện:



Chương 13 ta sẽ tìm hiểu thêm về COM/ActiveX cụ thể hơn.

6. Tổng kết

Bạn hãy làm một bảng tổng kết những gì đã làm ở chương này. Nếu có thể, hãy thử xây dựng bất cứ chương trình đơn giản nào theo ý muốn.

Chương 4:

Làm việc với Menu và hộp thoại

--------oOo--------

Nội dung thảo luận:

- Thêm menu vào chương trình với điều khiển *MainMenu*
- Xử lý mục chọn menu bằng mã lệnh
- Sử dụng hộp thoại *OpenFileDialog* và *ColorDialog*

1. Sử dụng điều khiển *MainMenu*

Điều khiển Menu cho phép thêm vào chương trình các thực đơn. Bạn có thể thêm mới, hiệu chỉnh, sắp xếp lại, xóa các menu. Bạn cũng có thể thêm các hiệu ứng như gán phím tắt, thêm dấu chọn *CheckBox*. Bạn có thể tạo sự kiện cho menu bằng mã lệnh như các điều khiển khác.

Dưới đây chúng ta sẽ sử dụng menu qua bài tập *MyMenu*

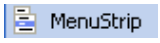
2. Chương trình *MyMenu*

2.1. Tìm hiểu chương trình

Chúng ta sẽ tìm hiểu chương trình thông qua các bước xây dựng.

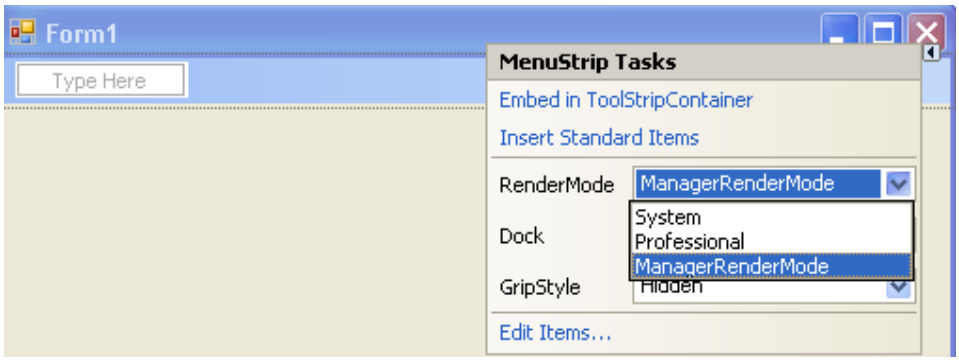
2.2. Thiết kế giao diện và xây dựng chương trình từng bước

Tạo mới một giải pháp mang tên *MyMenu* và thêm vào đó một dự án mới cùng tên như đã biết trong các bài tập trước.

Tại giao diện thiết kế, các bạn đưa điều khiển *MenuStrip* 

vào trong Form bằng cách double click hay kéo thả như đã biết.

Chúng ta không cần quan tâm đến vị trí của menu trên form vì VS sẽ tự động đặt nó sao cho phù hợp. Các bạn có thể thay đổi các thuộc tính sao cho phù hợp bằng cách click mở *Smart Tags* là nút mũi tên tam giác màu đen bên góc phải điều khiển *Menu*.



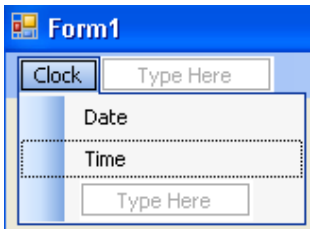
Khi được đặt vào form thì điều khiển menu sẽ được đặt tại một vùng như trên hình gọi là *khay công cụ* - *Component tray* và VS sẽ hiển thị trực quan menu trên đầu cửa sổ Form.

Chuỗi *Type Here* là nơi bạn có thể click chọn và nhập vào các mục chọn cho menu.

Chúng ta sẽ tạo ra menu ngay sau đây.

Nhắp chuột vào chuỗi *Type Here* và gõ vào chuỗi "Clock" và ấn enter.

Nhắp chuột vào chuỗi *Type Here* con ở dưới rồi gõ Date, Time như hình



Để đóng phần thiết kế menu, bạn click vào một vùng nào đó trên form, để hiển thị bạn lại click vào menu *Clock* như trên.

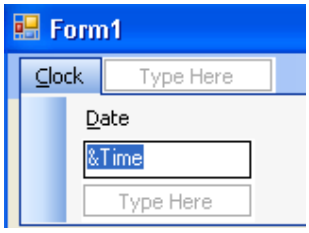
Bây giờ chúng ta sẽ tạo một số tùy biến cho Menu.

2.2.1. Thêm phím truy cập vào các mục chọn lệnh trên menu

Trong một số phần mềm hay ngay trình duyệt Windows Explorer của hệ điều hành các bạn có thể ấn tổ hợp *Alt + phím tắt* để mở nhanh một thực đơn nào đó. Các phím tắt ấy được gọi là phím truy cập – *Access Key*. Phím này có dấu gạch chân ở dưới.

Trong VS, để tạo phím này ở menu khá đơn giản. Bạn chỉ việc gõ thêm dấu '&' trước ký tự nào muốn hiển thị gạch chân trong phần *Type Here*.

Bạn hãy tạo ra các phím tắt cho các mục chọn của menu *Clock* như hình:



2.2.2. Thay đổi thứ tự các mục chọn

Việc thay đổi thứ tự các mục chọn khá đơn giản, bạn mở chế độ thiết kế menu rồi nhắp chọn mục chọn nào đó và kéo nó đến vị trí mong muốn.

Bạn thử kéo mục chọn *Time* lên thay cho vị trí mục chọn *Date* xem.

2.2.3. Xử lý các mục chọn

Bây giờ chúng ta tạo ra sự kiện click cho các mục chọn của menu. Khi bạn click vào *Date* hay *Time* thì một nhãn *Label* sẽ xuất hiện và hiển thị thông tin ngày hay giờ tương ứng.

Để làm được như thế, trước hết bạn tạo ra một Label vào trong form.Tạo thuộc tính cho đối tượng *Label1* như sau: *BorderStyle – FixedSingle; Font – Bold 14; Text – rỗng; TextAlign – MiddleCenter.*

Cài đặt thủ tục sự kiện cho mục chọn menu

Bây giờ chúng ta sẽ tạo sự kiện click cho các mục con trong menu Clock.

Nhắp vào menu Clock trên form1 để hiển thị menu con

Nhắp đôi chuột vào mục chọn Time để mở cửa sổ Code Editor và tạo ra một thủ tục có tên `TimeToolStripMenuItem_Click`. Trong VS.NET 2005 thì khi bạn gõ tên mục chọn là gì thì mặc định khi double click để viết mã thì VS sẽ tạo ra một thủ tục có phần đầu tên trùng với tên mục chọn (phần tên chưa có dấu cách trống phân cách tên mục chọn) menu (ở trên là TimeToolStripMenuItem_Click). Tất nhiên đây là default, bạn có thể thay đổi tên nhờ thuộc tính *Name* ở cửa sổ *Properties*.

Nhập dòng mã sau:

```
Label1.Text = TimeString
```

Tương tự với thủ tục `DateToolStripMenuItem_Click` của mục chọn *Date*

```
Label1.Text = DateString
```

2.2.4. Chạy chương trình MyMenu

Bạn thử chạy chương trình xem. Các thông tin về ngày tháng và thời gian sẽ được hiển thị bên trong lable11 khi bạn click chọn mục chọn tương ứng trên menu. Để thay đổi cách hiển thị thông số ngày tháng và thời gian bạn có thể thao tác trong coltrol panel.

Tìm hiểu các hàm và thuộc tính về thời gian hệ thống:

Thuộc tính, hàm	Mô tả
TimeString	Trả giờ hệ thống
DateString	Trả ngày hệ thống
Now	Trả về ngày giờ hệ thống đã mã hóa
Hour (time)	Trả về giờ dựa trên thời gian của đối số time
Minute (time)	Trả về phút dựa trên thời gian của đối số time
Second (time)	Trả về giây dựa trên thời gian của đối số time
Day (date)	Trả về ngày dựa trêin đối số date (1-31)
Month (date)	Trả về tháng dựa trên đối số date (1-12)
Year (date)	Trả về năm của đối số date
Weekday (date)	Trả về ngày trong tuần của đối số date

3. Sử dụng thành phần điều khiển hộp thoại chuẩn

VS.NET 2005 cung cấp 8 hộp thoại chuẩn. Các bạn có thể tìm thấy các điều khiển này trên TOOLBOX. Bảng sau liệt kê các hộp thoại chuẩn đó:

Hộp thoại	Mô tả
ColorDialog	Cho chọn tên và hiệu chỉnh giá trị màu sắc
FolderBrowserDialog	Cho phép duyệt thư mục
FontDialog	Chọn tên và kiểu font chữ mới
OpenFileDialog	Cho lấy về ổ đĩa, tên file, tên folder
PageSetupDialog	Điều khiển các thiết lập trang in
PrintDialog	Cho thiết lập các tùy chỉnh in ấn
PrintPreviewDialog	Hiển thị xem trước khi in
SaveFileDialog	Cho đặt tên file, folder mới sắp ghi lên đĩa

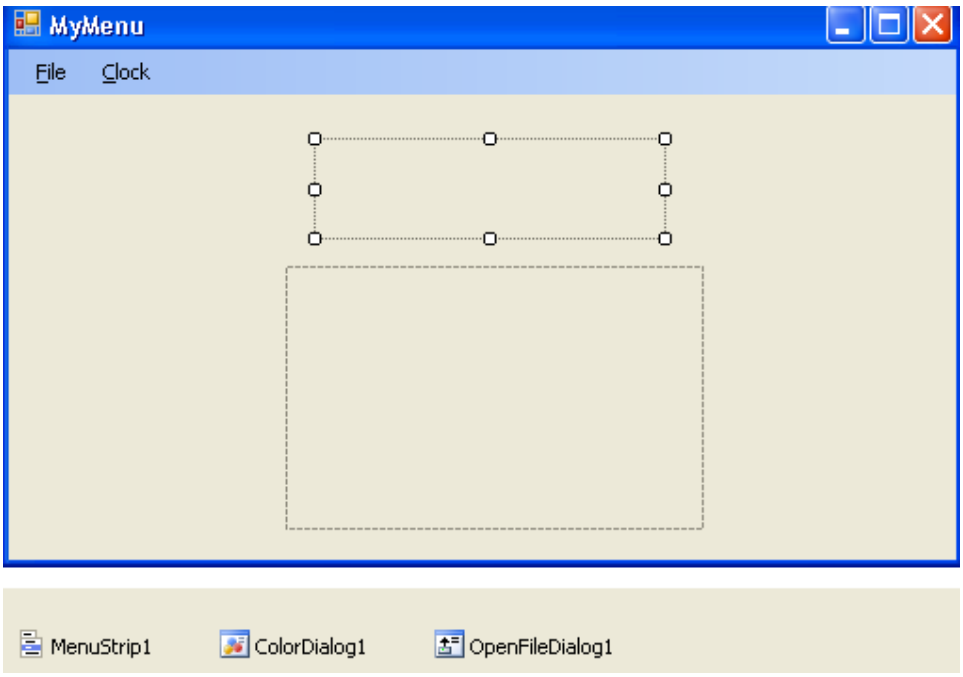
3.1. Thêm vào hộp thoại chuẩn

Ta tiếp tục bổ sung cho dự án *MyMenu* trước đây bằng cách thêm vào các hộp thoại, ở đây ta sẽ thêm hai hộp thoại chuẩn là *OpenFileDialog* để mở một ảnh cho hiển thị trong một điều khiển *PictureBox1* và một hộp thoại *ColorDialog* cho phép chọn màu hiển thị cho *Label1* hiển thị thông tin ngày giờ hệ thống.

Trước hết mở lại solution *MyMenu* và để chế độ thiết kế form (mở file form1.vb[Design])

Tạo hai điều khiển là *OpenFileDialog* và *ColorDialog* vào Form1 bằng cách double click vào hai điều khiển này trên TOOLBOX.

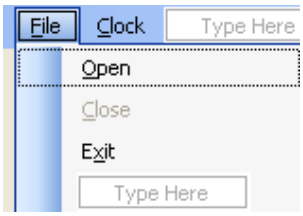
Tạo điều khiển *PictureBox1* vào trong form1. Giao diện thiết kế:



3.2. Thêm mục File vào menu chương trình

Bạn tạo thêm một mục con *Color* vào trong menu *Clock*. Mục này sẽ kích hoạt hộp thoại *ColorDialog1* chọn màu cho *Label1*.

Tạo một Menu *File* bên cạnh menu *Clock* như hình. Đồng thời tạo thêm các mục con *Open, Close, Exit* trong menu này.



Tiếp theo bạn thay đổi tên bằng thuộc tính *Name* trong cửa sổ *Properties* cho các mục chọn: mục *Open* thành mnuOpenItem, *Close* thành mnuCloseItem, *Exit* thành mnuExitItem.

Bạn cũng đặt thuộc tính *Enable* của mục *Close* (giờ là mnuCloseItem) thành False. Thuộc tính này vô hiệu hóa hay làm mờ mục *Close* như hình. Nó chỉ được sáng lên để người dùng click khi mã thực thi chương trình cho phép.

3.3. Viết mã chương trình

3.3.1. Cài đặt thủ tục cho mục *Open* trên menu *File*.

Bạn tạo thủ tục `mnuOpenItem_Click` bằng cách double click vào mục *Open* trên menu *File* và nhập đoạn mã sau:

```
OpenFileDialog1.Filter = "Bitmaps (*.bmp) | *.bmp"
If OpenFileDialog1.ShowDialog() = Windows.Forms.DialogResult.OK
Then
    PictureBox1.Image = System.Drawing.Image.FromFile _
    (OpenFileDialog1.FileName)
    mnuCloseItem.Enabled = True
End If
```

Chú thích mã:

- Đoạn mã thứ nhất giúp lọc ra loại file để mở là file ảnh dạng Bitmap (*.bmp). Bạn có thể mở nhiều loại file bằng câu lệnh:

```
OpenFileDialog1.Filter = _
  "Bitmaps(*.bmp)|*.bmp|JPEG(*.jpg)|*.jpg|All Files(*.*)|*.*"
```

- Phương thức `ShowDialog()` là phương thức mới trong VS.NET, nó có thể dùng được với mọi hộp thoại và cửa sổ Windows Forms. Phương thức này trả về kết quả mang tên `DialogResult` cho biết người dùng đã click vào hộp thoại. Và nếu nút OK được click thì kết quả trả về sẽ bằng với `DialogResult.OK`.

- Khi nút Open được nhấn, nếu hợp lệ thì thuộc tính *FileName* của *OpenFileDialog* sẽ mang đầy đủ đường dẫn và tên file của file đã mở vì thế mà dòng mã thứ 3 sẽ nạp chính xác ảnh vào *PictureBox1*.

3.3.2. Cài đặt thủ tục cho mục *Close*

Tương tự bạn cũng double click vào mục *Close* để tạo thủ tục click cho nó và nhập chính xác đoạn mã sau:

```
PictureBox1.Image = Nothing
mnuCloseItem.Enabled = False
```

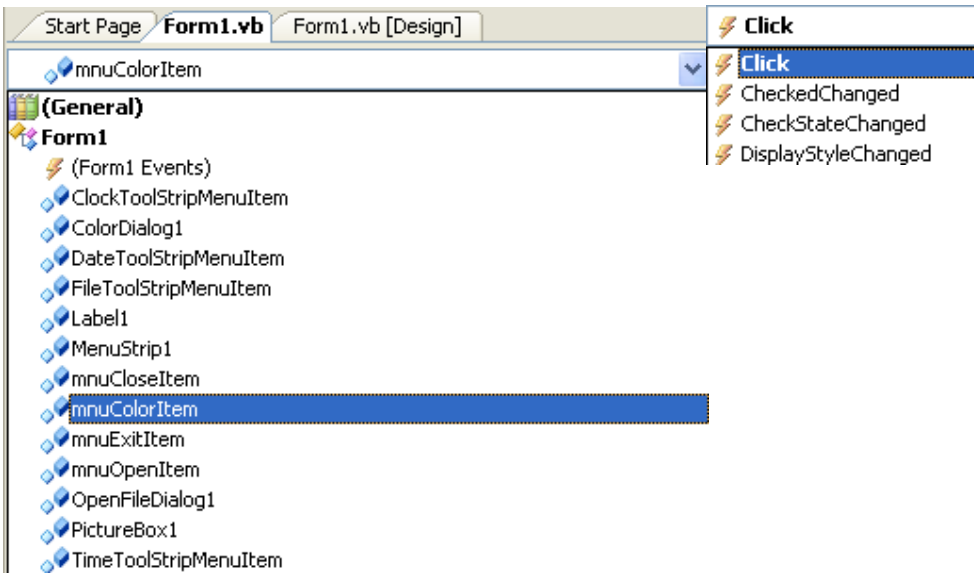
Khi mở ảnh rồi thì mục *Close* sáng lên, khi click vào mục này thì *PictureBox1* không còn ảnh nữa và mục này lại bị vô hiệu hóa.

3.3.3. Cài đặt thủ tục cho mục *Exit*

Nhắp đôi vào mục *Exit* và nhập dòng mã: End

3.3.4. Cài đặt thủ tục cho mục *Color*

Tạo thủ tục `mnuColorItem_Click` bằng cách double click hay chọn từ danh sách xổ xuống như hình



Nhập vào đoạn mã:

```
ColorDialog1.ShowDialog()
Label1.ForeColor = ColorDialog1.Color
```

Chú thích mã:

- Phát biểu đầu tiên gọi *ShowDialog()* để hiển thị hộp thoại *ColorDialog*.

- Phát biểu thứ hai nhận giá trị màu trả về từ hộp thoại *ColorDialog* và gán cho màu chữ *Text* – ForeColor của điều khiển *Label1*. Bạn có thể gán màu cho bất cứ thuộc tính nào như *BackColor*.

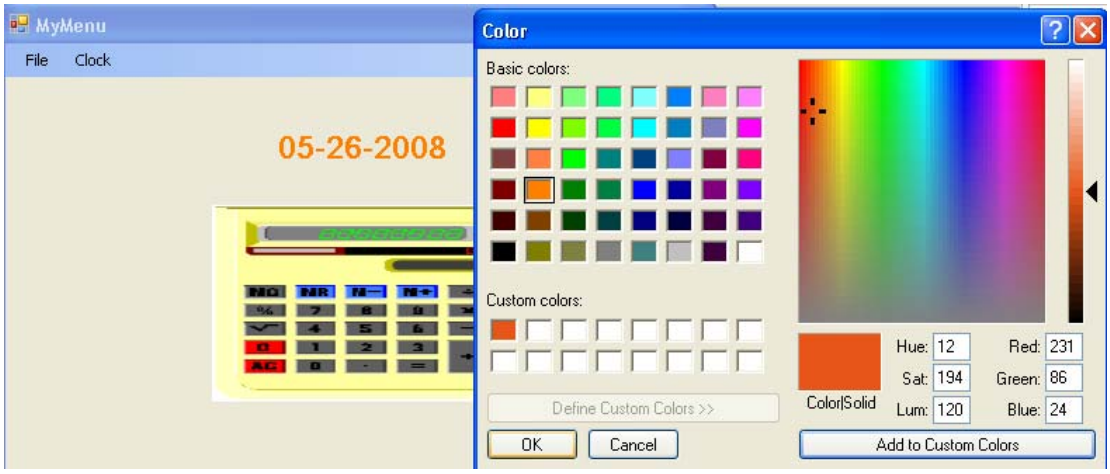
Ngoài ra, bạn cũng có thể thêm các thuộc tính khác cho hộp thoại *ColorDialog* trước khi gọi đến phương thức ShowDialog(). Một số thuộc tính và cách gọi được liệt kê như sau:

```
'ColorDialog1.FullOpen = True :Hiển thị khung tùy biến màu mở rộng
'ColorDialog1.AllowFullOpen = True: hiển thị nút định nghĩa màu tùy biến
'ColorDialog1.AnyColor = True: cho phép chọn tất cả các loại màu
'ColorDialog1.ShowHelp = True: Hiển thị nút nhấn trợ giúp
'ColorDialog1.SolidColorOnly = True: Hiển thị chỉ những màu đặc
```

3.3.5. Chạy chương trình

Bạn hãy chạy chương trình bằng cách nhấn phím F5 hay *Start* trên Standard Bar và thử tất cả các tính năng của chương trình.

Đây là giao diện:



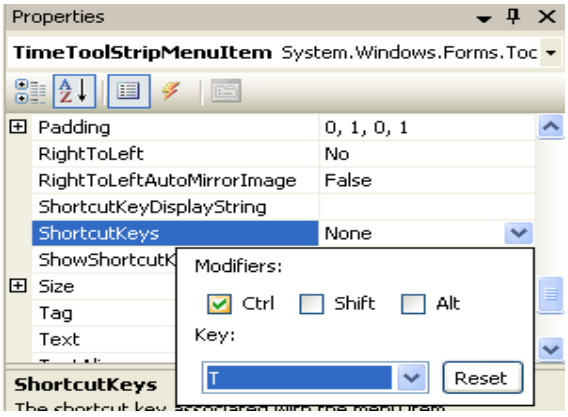
3.4. Gán phím tắt cho MENU

Phím tắt cho phép bạn ấn tổ hợp phím để thực hiện lệnh mà không cần chọn menu. Ví dụ như Ctrl+C để sap chép một đoạn text trong Word.

Chúng ta thử gán các phím tắt cho menu trong chương trình *MyMenu* xem sao.

Trước hết mở giải pháp *MyMenu* ở chế độ thiết kế

Click vào menu Clock trên Form, chọn mục Time và R-Click chọn Properties. Thiết lập thuộc tính *ShortCutKeys* như hình



Tương tự bạn chọn các mục còn lại theo ý thích miễn là các phím nóng không trùng nhau.

3.5. Chạy chương trình hoàn thiện

Bây giờ bạn kiểm tra những gì đã làm bằng cách chạy chương trình một lần nữa. Bạn kiểm tra lần lượt từ việc mở file ảnh cho hiện lên trên *PictureBox1*, hiển thị thông tin giờ hệ thống, gọi các mục menu bằng phím tắt.

4. Tổng kết

Bây giờ như mọi chương chúng ta vẫn làm, bạn hãy làm bảng tổng kết các công việc chúng ta đã thực hiện. Viết lại mã chương trình theo ý chúng ta.

Chương 5:

Biến và toán tử trong VISUAL BASIC.NET

--------oOo--------

Nội dung thảo luận:

- Sử dụng biến để chứa dữ liệu của chương trình
- Nhận dữ liệu nhập bằng cách sử dụng hàm *InputBox*
- Hiển thị thông điệp bằng *MsgBox*
- Làm việc với những biến dữ liệu khác nhau
- Sử dụng các toán tử toán học và hàm trong công thức
- Sử dụng các phương thức toán học trong lớp *System.Math* của .NET

Chúng ta đã biết cách tạo mới một chương trình VS.NET và thực thi chúng như thế nào. Trong 5 chương kế tiếp chúng ta sẽ làm quen với cách viết mã VB – phát biểu, từ khóa, cú pháp – là các yếu tố quan trọng tạo nên một chương trình VB.

Sau phần này chúng ta có thể yên tâm tạo mới một chương trình VB chuyên nghiệp hơn.

Chú ý trước chương:

- Khi muốn sử dụng một biến trong VS.NET bạn phải khai báo trước bằng câu lệnh *Dim*. Nếu muốn dùng mà không khai báo thì phải đặt phát biểu *Option Explicity Off*. Điều này khuyến cáo là không nên.

- Việc chuyển kiểu trong VS.NET rất được xem trọng. Bạn phải thường xuyên sử dụng các hàm chuyển kiểu như *CInt, CLong,CType*...để khiến các biến tương thích với nhau. Việc thực hiện các phép tính giữa các biến cũng phải cùng kiểu.

1. Các phần tử của một phát biểu chương trình VISUAL BASIC

Một phát biểu trong VS.NET là bất cứ thứ gì kết hợp giữa từ khóa, thuộc tính, hàm, toán tử phương thức, các biểu tượng trong VB có thể tạo nên một chỉ thị hợp lệ được nhận dạng và hiểu được bởi trình biên dịch VB.

Ví dụ: *End* là một *phát biểu* để chấm dứt chương trình.

Các nguyên tắc để xây dựng nên phát biểu của chương trình được gọi là *cú pháp – Syntax.*

Trước hết chúng ta sẽ làm quen về cách sử dụng biến, kiểu dữ liệu trong VB.

2. Sử dụng biến để chứa thông tin

Trong VB.NET bạn cần khai báo biến trước khi sử dụng nó.

Việc khai báo được tiến hành bằng câu lệnh *Dim*. Cấu trúc của phát biểu là *Dim + tên biến + As + kiểu dữ liệu*. Phát biểu này có thể đặt ở bất kỳ đâu nhưng thường được đặt ở đầu mỗi thủ tục, nơi cần dùng biến. Ví dụ:

Dim LastName As String

Phát biểu trên khai báo một biến tên là *LastName* có kiểu dl là *String*.

Sau khi đã khai báo biến thì bạn có thể thoải mái gán hay lưu thông tin vào biến, ví dụ:

LastName = "Duc Lap"

Và có thể gán nội dung biến cho thuộc tính của đối tượng, ví dụ:

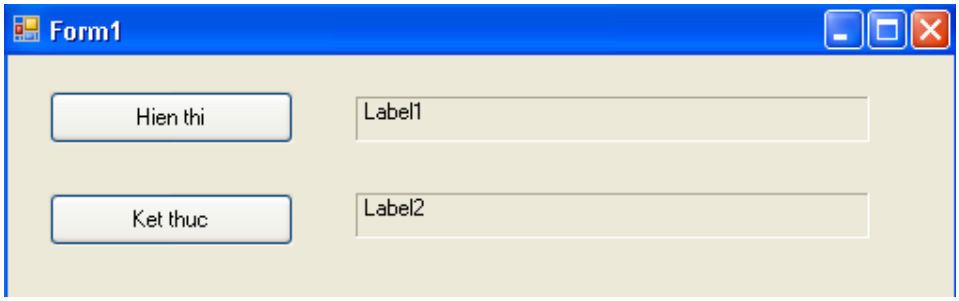
Label1.Text = LastName

3. Sử dụng biến trong chương trình

Chúng ta sẽ làm quen với cách thay đổi giá trị của biến trong chương trình qua bài tập đơn giản *VariableTest*:

Thiết kế giao diện:

Giao diện của form thiết kế như sau



Nó gồm hai nút là *Hien thi – Button1, Ket thuc – Button* và hai nhãn *Label1, Label2.*

Tìm hiểu cách thực thi chương trình:

Chương trình sẽ khai báo một biến có tên *LastName* và tạo cho nó hai giá trị khác nhau là hai chuỗi (String) đồng thời gán giá trị chuỗi đó cho thuộc tính *Text* của hai nhãn khi người dùng click vào nút *Hien thi*.

Viết mã:

Tạo thủ tục `Button1_Click` bằng cách double click vào nút *Hien thi* trên form trong giao diện thiết kế và nhập chính xác đoạn mã sau:

```
Dim LastName As String

LastName = "Đức Lập"
Label1.Text = LastName

LastName = "LVP Office"
Label2.Text = LastName
```

Chú thích mã:

- Phát biểu thứ nhất khai báo một biến có tên *LastName* có kiểu *String*. Bạn không lo lắng nếu có một dòng gạch xanh dưới chân biến. Có dòng này là vì ta chưa khởi tạo giá trị cho biến.

- Phát biểu thứ hai gán giá trị "Đức Lập" cho biến, phát biểu thứ 3 gán giá trị của biến cho thuộc tính Text của nhãn Label1. Tương tự cho các phát biểu còn lại.

Chú thích về qui ước khai báo biến:

- Không có khoảng trắng trong tên biến. Tên biến bắt đầu bằng dấu gạch chân '_' hay chữ cái.

- Tên biến có chiều dài tùy thích nhưng nên đặt tên cho gợi nhớ và không nên dài quá 33 ký tự.

- Không nên đặt tên biến trùng với các từ khóa, tên thuộc tính, phương thức chuẩn của VB để tránh gặp lỗi khi biên dịch.

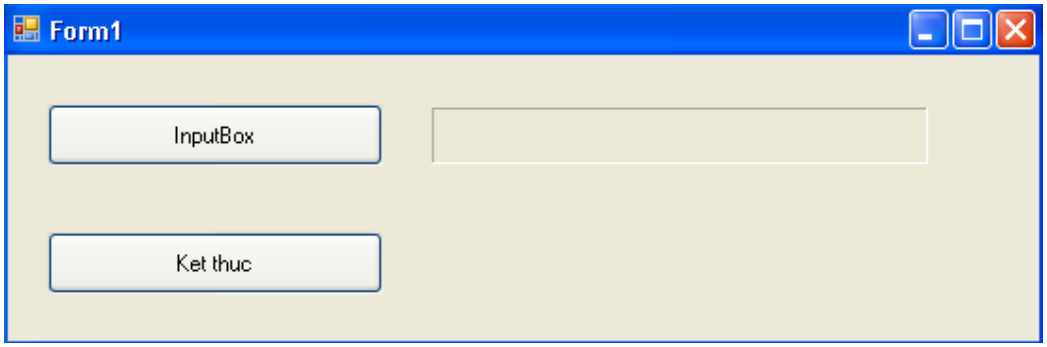
4. Sử dụng biến để chứa dữ liệu nhập từ ngoài vào

Biến thường dùng để nhận thông tin của người dùng nhập vào từ các điều khiển như *TextBox, ListBox, Menu,…*

Trong ví dụ sau đây với bài tập *InputBox* chúng ta sẽ sử dụng một hàm có tên *InputBox*. Khi bạn gọi hàm này chương trình sẽ hiển thị một cửa sổ nhỏ bao gồm một TextBox cho phép nhập liệu. Khi đồng ý, họ ấn nút OK và dữ liệu trong ô sẽ trả về cho nơi gọi hàm.

Thiết kế giao diện:

Bạn tạo mới một giải pháp có tên InputBox và thêm một dự án có cùng tên. Tiếp theo thiết kế giao diện như hình:



Chương trình bao gồm hai nút là *InputBox – Button1, Ket thuc – Button2* và một nhãn có tên *Label1*.

Viết mã:

Tạo thủ tục `Button1_Click` bằng cách double click vào nút *InputBox* trên form và nhập đoạn mã sau:

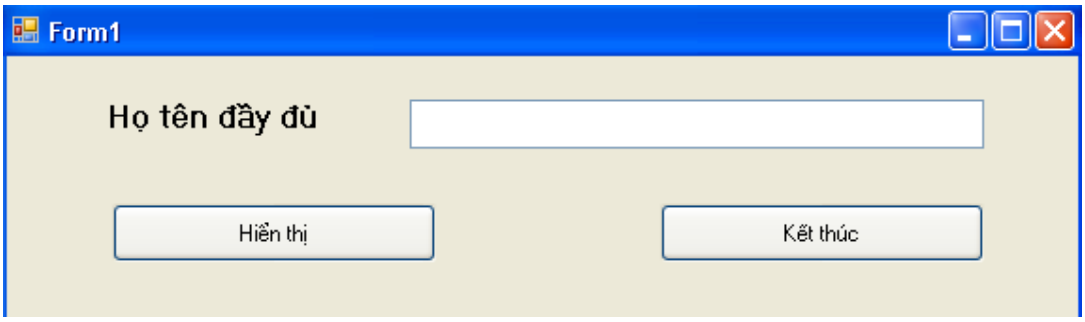
```
Dim prompt, FullName As String
prompt = "Nhập tên đầy đủ:"
'FullName = CStr(Me.IsInputChar(prompt))
FullName = InputBox(prompt)

Label1.Text = FullName
```

Chú ý: Có thể bạn sẽ gặp một thông báo lỗi với hàm *Inputbox* này. Nó báo *Inputbox* là một không gian tên chứ không phải là một hàm nên không thể sử dụng như ta đã gọi nó. Thực ra thì lỗi này phát sinh do bạn đặt tên của dự án cũng như solution là *Inputbox* trùng với tên hàm. Đây cũng là một chú ý khi đặt tên giải pháp và tên của dự án không nên đặt tên trùng với tên hay từ khóa trong VB.NET, có thể gây những lỗi không nên gặp và khó giải quyết. Bây giờ bạn tạo một giải pháp mới có tên *Inputbox1* và làm như trên là xong.

Bây giờ chúng ta sẽ tạo một chương trình mới có chức năng tương tự nhưng có xuất kết quả nhờ biến. Chương trình gồm một ô textBox cho nhập tên. Khi người dùng click vào nút hiển thị thì thông tin về tên trong ô textbox sẽ được hiển thị dưới dạng một thông báo nhờ hàm *MsgBox()*.

5. Sử dụng biến nhận kết quả xuất, ví dụ ImportValueOfVariable

Giao diện chương trình:



Bạn tạo mới dự án *ImportValueOfVariable* và thêm vào một dự án như đã biết. Tạo giao diện như hình.

Double click vào nút *hiển thị* và nhập đoạn mã sau:

```
Dim FullName As String = TextBox1.Text
If FullName = "" Then
    MsgBox("Bạn chưa nhập tên")
Else
    MsgBox(FullName, , "Thông tin nhập")
End If
```

Ghi chú mã:

- Hàm *MsgBox:* có tác dụng hiện một hộp thoại chứa thông điệp. Cấu trúc của hàm như sau: *ButtonClicked=MsgBox(Prompt, Buttons, Tittle)*.

Prompt là thông điệp cần hiển thị

Buttons là con số cho biết những nút nhấn hay biểu tượng sẽ hiển thị trong hộp thoại

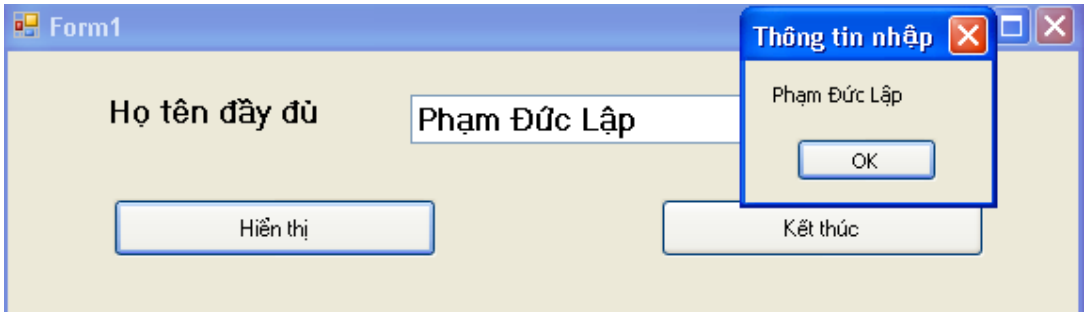
Tittle là tiêu đề hiển thị trên hộp thoại

ButtonClicked được dùng để nhận về kết quả của hàm

Trong trường hợp chỉ muốn hiển thị thì các đối số *Buttons, Tittle* và biến *ButtonClicked* không cần có.

Bây giờ bạn chạy thử chương trình xem.

Ở ví dụ trên, chúng ta đã sử dụng biến FullName để nhận về giá trị thuộc tính Text của *TextBox1* và xuất giá trị biến này nhờ hàm MsgBox.



6. Làm việc với các kiểu dữ liệu đặc thù

VS.NET cung cấp rất nhiều kiểu dữ liệu giúp ta định nghĩa biến. Bảng sau liệt kê chúng:

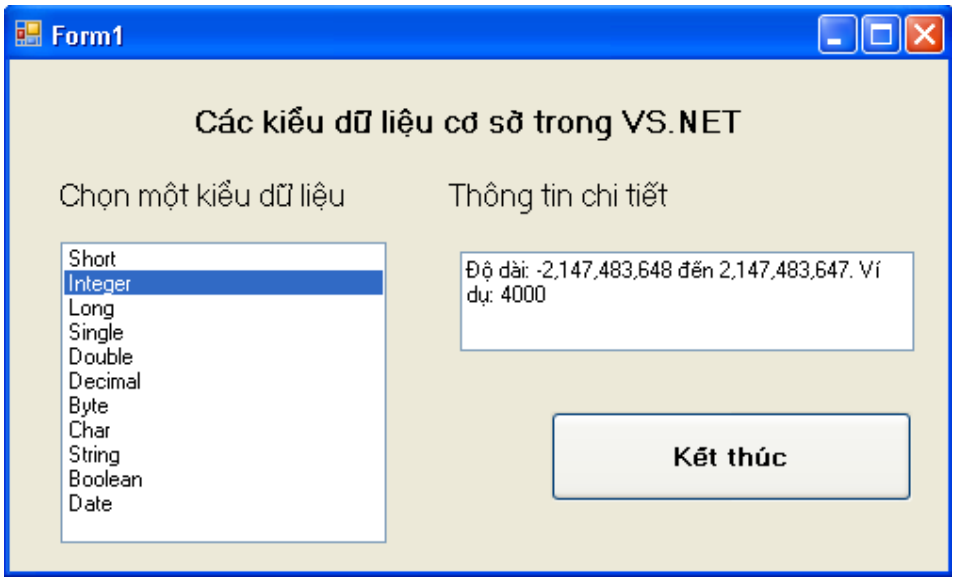
Kiểu dữ liệu	Kích thước	Phạm vi	Ví dụ
Short	16-bit	-32,678 - 32,767	Dim S as Short S = 12500
Integer	32-bit	-2,147,483,648 đến 2,147,483,647	Dim I as Integer S = 4000
Long	64-bit	-9,233,372,036,854,775,808 đến 9,233,372,036,854,775,807	Dim L as Long L = 3988890343
Single	32-bit (dấu phảy động)	-3.402823E38 đến 3.402823E38	Dim Sg as Single Sg = 899.99
Double	64-bit (dấu phảy động)	-1.797631348623E308 đến 1.797631348623E308	Dim D as Double D=3.1.4159265
Decimal	128-bit	Trong khoảng +/-79,228x10^{24}	Dim Dc as Decimal Dc=7234734.5
Byte	8-bit	0-255	Dim B as Byte B=12
Char	16-bit	0-65,536	Dim Ch As Char Ch=”L”
String	Nhiều ký tự	Chứa 0 đến 2 tỷ ký tự	Dim St As String St=”Đức Lập”
Boolean	16-bit	Hai giá trị True hay False	Dim Bl As Boolean

			Bl = True
Date	64-bit	Từ 1/1/1 đến 31/12/9999	Dim Da As Date Da=#16/07/1984
Object	32-bit	Bất kỳ kiểu đối tượng nào	Dim Obj As Object

Sau đây chúng ta sẽ xây dựng một chương trình có tên *MyDataTypes* cho phép người dùng chọn kiểu dữ liệu. Các kiểu dữ liệu sẽ được liệt kê trong một danh sách *ListBox1*. Khi người dùng click chọn kiểu dữ liệu tương ứng nào thì một thông báo về độ dài và một ví dụ được đưa ra trong một ô TextBox.

Sử dụng các kiểu dữ liệu cơ sở:

Giao diện chương trình:



Bạo tạo trong form các điều khiển gồm *ListBox, TextBox* và một *Button* như hình.

Viết mã:

- Khai báo biến: Bạn khai báo các biến (đặt dưới dòng `Public Class Form1`) :

```
Dim Sh As Short
Dim I As Integer
Dim L As Long
Dim Sg As Single
Dim Db As Double
Dim Dc As Decimal
Dim B As Byte
Dim Ch As Char
Dim Str As String
Dim Bo As Boolean
Dim Da As Date
```

- Tạo thủ tục `Form1_Load` để thêm các mục chọn cho ListBox1 như sau:

```
ListBox1.Items.Add("Short")
```

```
        ListBox1.Items.Add("Integer")
        ListBox1.Items.Add("Long")
        ListBox1.Items.Add("Single")
        ListBox1.Items.Add("Double")
        ListBox1.Items.Add("Decimal")
        ListBox1.Items.Add("Byte")
        ListBox1.Items.Add("Char")
        ListBox1.Items.Add("String")
        ListBox1.Items.Add("Boolean")
        ListBox1.Items.Add("Date")
```

- Tạo thủ tục `ListBox1_SelectedIndexChanged` để hiện thông tin trên TextBox1 như sau:

```
      Select Case ListBox1.SelectedIndex
            Case 0
                TextBox1.Text = "Độ dài: -32,678 - 32,767. Ví dụ: "
                Sh = 1000
                TextBox1.Text &= Sh
            Case 1
                I = 4000
                TextBox1.Text = "Độ dài: -2,147,483,648 đến
2,147,483,647. Ví dụ: "
                TextBox1.Text &= I
            Case 2
                L = 3988890343
                TextBox1.Text = "Độ dài: -9,233,372,036,854,775,808 đến
9,233,372,036,854,775,807. " & _
                "Ví dụ: "
                TextBox1.Text &= L
            Case 3
                Sg = 899.99
                TextBox1.Text = "Độ dài: -3.402823E38 đến 3.402823E38 " &
_
                "Ví dụ: "
                TextBox1.Text &= Sg
            Case 4
                Db = 3.14159265
                TextBox1.Text = "Độ dài: -1.797631348623E308 đến
1.797631348623E308. " & _
                "Ví dụ: "
                TextBox1.Text = Db
            Case 5
                Dc = 7234734.5
                TextBox1.Text = "Độ dài: Trong khoảng +/-79,228x1024. " &
_
                "Ví dụ: "
                TextBox1.Text &= Dc
            Case 6
                B = 12
                TextBox1.Text = "Độ dài: 0-255. " & _
                "Ví dụ: "
                TextBox1.Text = B
            Case 7
                Ch = "L"
                TextBox1.Text = "Độ dài: 0-65,536. " & _
                "Ví dụ: "
                TextBox1.Text &= Ch
            Case 8
                Str = "Đức Lập"
                TextBox1.Text = "Chứa 0 đến 2 tỷ ký tự. " & _
```

```
            "Ví dụ: "
            TextBox1.Text &= Str
        Case 9
            Bo = True
            TextBox1.Text = "True hay False. " & _
            "Ví dụ: "
            TextBox1.Text &= Bo
        Case 10
            Da = Now.Date
            TextBox1.Text = "Từ 1/1/1 đến 31/12/9999. " & _
            "Ví dụ: "
            TextBox1.Text &= CStr(Da)
    End Select
```

- Tạo thủ tục `Button1_Click` để kết thúc chương trình:

```
        End
```

Chạy chương trình:

Bạn thử chạy chương trình và click chọn tất cả các mục trong ListBox để hiện thông tin về độ dài cũng như các ví dụ của từng kiểu dữ liệu.

Kiểu dữ liệu tự định nghĩa:

Visual Basic cũng cho phép ta tự định nghĩa kiểu dữ liệu của riêng mình – gọi là kiểu dữ liệu cấu trúc hay kiểu dữ liệu tự định nghĩa bởi người dùng (User – Defind Type hay UDT) bằng phát biểu *Structure*. Phát biểu này phải xuất hiện ở đầu form hay trong đơn thể mã code module giống như các biến khai báo *Public*. Ví dụ:

```
Structure Employee
    Dim name As String
    Dim DateOfBirth As Date
    Dim age As Date
End Structure
```

Sau khi đã tạo mới một UDT bạn có thể sử dụng nó ngay trong thủ tục hay chương trình. Ví dụ:

```
    Dim Worker1 As Employee
    Worker1.name = "Lê Thị Lan"
    Worker1.age = Date.FromOADate(12 / 1 / 1983)
```

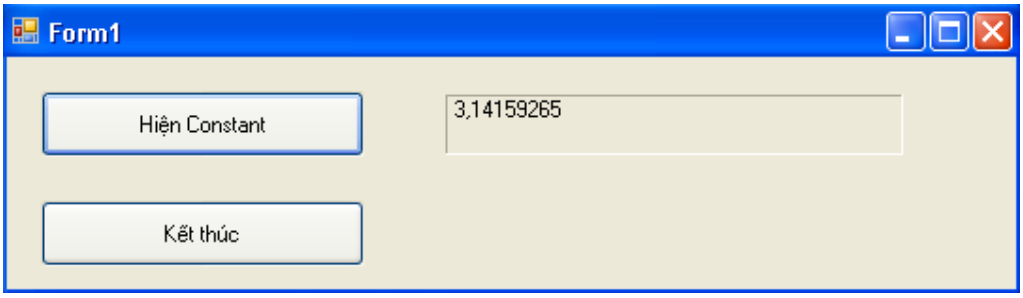
7. Hằng số: Biến không cho thay đổi giá trị

Trong VB cũng như nhiều ngôn ngữ khác tồn tại khái niệm hằng. Hằng là một biến đặc biệt không thay đổi giá trị. Nó cũng giống như biến nhưng không tồn tại khái niệm gán lưu giá trị mới cho hằng số. Hằng số được khai báo bằng từ khóa *Const*. Ví dụ:

```
    Const PI As Double = 3.14159265
    Label1.Text = PI
```

Bạn có thể tạo ví dụ sau: tạo mới một giải pháp và thêm vào một dự án nhỏ cùng tên là *ConstantValue*.

Chương trình sẽ hiển thị giá trị của một hằng được khai báo với từ khóa *Const*. Để làm được việc này, các bạn thiết kế giao diện như hình sau:



Mã chương trình tương đối đơn giản. Bạn tạo thủ tục `Button1_Click` và thêm vào đoạn mã như sau:

```
Const PI As Double = 3.14159265
Label1.Text = PI
```

8. Làm việc với toán tử trong VISUAL BASIC.NET

Visual Basic.NET cung cấp các toán tử cơ bản sau:

Toán tử	Mô tả
+	Phép cộng
-	Phép trừ
*	Nhân
/	Chia
\	Chia lấy phần nguyên
^	Mũ lũy thừa
&	Nối chuỗi

8.1. Các toán tử cơ sở: +, -, *, /

Chúng ta sẽ sử dụng các toán tử đơn giản này để làm ví dụ *BasicMath* sau:

Tìm hiểu chương trình:

Giao diện:

Chương trình gồm hai textbox cho phép nhập hai giá trị để gán cho hai biến *value1* và *value2*, bốn radiobutton cho phép chọn bốn toán tử khác nhau, khi đã nhập đầy đủ hai giá trị thì có thể thực hiện tính bằng cách nhấp chọn nút 'thực hiện tính' và kết quả hiển thị trong ô textbox3 – kết quả.

Xây dựng giao diện:

Bạn tạo một giải pháp và thêm một dự án cùng tên *BasicMath* đồng thời thiết kế giao diện như hình.

Viết mã:

- Khai báo biến: bạn khai báo 2 biến *value1, value2* ở đầu lớp *form1* như sau:

```
Dim value1, value2 As Double
```

- Tạo thủ tục `Button1_Click` bằng cách double click vào nút 'thực hiện tính' và nhập đoạn mã sau:

```
If TextBox1.Text = "" Or TextBox2.Text = "" Then
    MsgBox("Bạn cần nhập đầy đủ hai giá trị")
Else
    value1 = CDbl(TextBox1.Text)
    value2 = CDbl(TextBox2.Text)
    If RadioButton1.Checked = True Then
        TextBox3.Text = value1 + value2
    End If
    If RadioButton2.Checked = True Then
        TextBox3.Text = value1 - value2
    End If
    If RadioButton3.Checked = True Then
        TextBox3.Text = value1 * value2
    End If
    If RadioButton4.Checked = True Then
        TextBox3.Text = value1 / value2
    End If
End If
```

Chú thích mã:

- Hàm `CDbl` là hàm chuyển kiểu sang kiểu Double.

Thực thi chương trình:

Bạn ấn phím F5 hay nút start để chạy chương trình.

8.2. Sử dụng các toán tử : \, Mod, ^, &

Chúng ta tiếp tục sử dụng bốn toán tử khác gồm: chia lây nguyên (\), chia lấy dư (Mod), mũ lũy thừa (^), nối chuỗi (&) trong bài tập *AdvancedMath* sau đây:

Tìm hiểu chương trình:

Chương trình *Advanced* cũng tương tự như chương trình *BasicMath* các bạn vừa xây dựng ở trên. Nó chỉ thay thế các toán tử mà thôi.

Chúng ta sẽ xây dựng chương trình này.

Thiết kế giao diện:

Giao diện chương trình như sau:

Bạn cũng tạo một giải pháp mới và thêm vào một dự án có cùng tên là *AdvancedMath* và thiết kế giao diện như hình.

Viết mã:

- Khai báo biến: bạn khai báo hai biến ở đầu lớp Form1 như sau:

```
Dim value1, value2 As Double
```

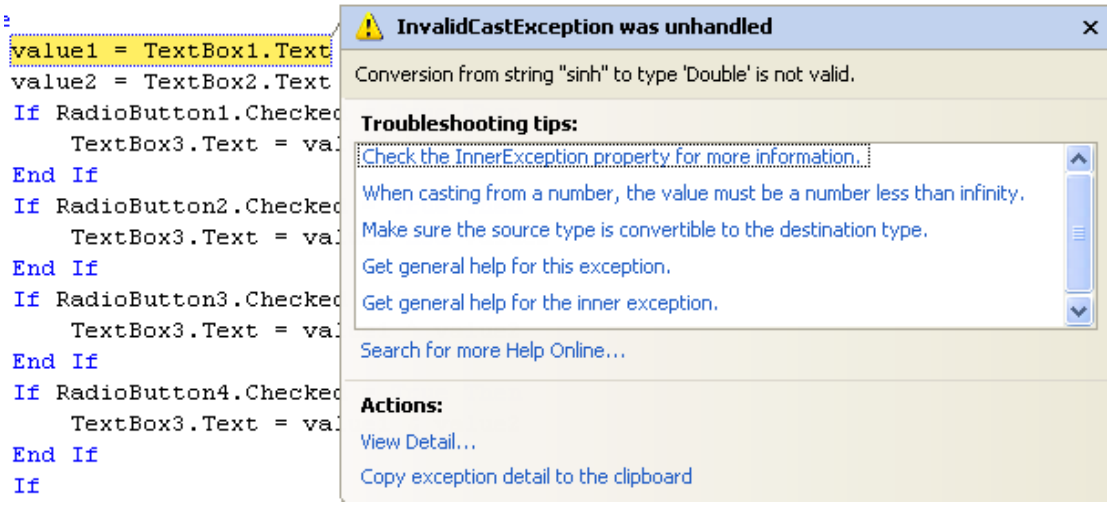
- Tạo thủ tục Button1_Click như sau:

```
If TextBox1.Text = "" Or TextBox2.Text = "" Then
    MsgBox("Bạn phải nhập đầy đủ hai giá trị")
Else
    value1 = TextBox1.Text
    value2 = TextBox2.Text
    If RadioButton1.Checked = True Then
        TextBox3.Text = value1 \ value2
    End If
    If RadioButton2.Checked = True Then
        TextBox3.Text = value1 Mod value2
    End If
    If RadioButton3.Checked = True Then
        TextBox3.Text = value1 ^ value2
    End If
    If RadioButton4.Checked = True Then
        TextBox3.Text = value1 & value2
    End If
End If
```

Chạy chương trình:

Bạn chạy chương trình như đã biết. Tất cả đều suôn sẻ cho đến toán tử thứ tư '&' thì xảy ra lỗi nếu bạn nhập vào hai textbox hai giá trị *value1* hay *value2* là hai chuỗi ví dụ value1 là "sinh" và value2 là "nhật".

Lỗi này gọi là lỗi thực thi – runtime error phát sinh khi chạy chương trình. Màn hình lỗi như sau:



Bạn có thể khắc phục lỗi này một cách không triệt để bằng cách thay đổi kiểu hai biến ở phát biểu khai báo chúng như sau:

```
Dim value1, value2 As String
```

Bạn chạy lại chương trình để xem phương thức thực thi của toán tử '&'. Nhưng nếu bạn nhập giá trị "sinh" "nhật" để thực hiện một trong ba toán tử '\', 'Mod', '^' thì chương trình lại phát sinh lỗi như trên.

Để khắc phục triệt để lỗi này bạn phải sử dụng một số hàm chuyển kiểu sẽ được bàn ở các chương sau.

9. Làm việc với các phương thức trong thư viện .NET

Thư viện .NET cung cấp rất nhiều phương thức hữu ích. Chúng ta sẽ làm quen với các phương thức được cung cấp bởi lớp *Math* trong thư viện .NET.

Bộ khung làm việc .NET FRAMEWORK là một tính năng mới trong VS.NET, nó chia sẻ môi trường lập trình và là nền tảng của hệ điều hành windows trong tương lai. Bộ khung này bao gồm tập hợp các lớp thư viện mà bạn có thể đem vào sử dụng trong dự án của mình bằng phát biểu *Imports*.

Trước hết chúng ta sẽ làm quen với lớp System.Math của .Net FrameWork. Bảng sau sẽ liệt kê danh sách một số phương thức thông dụng trong lớp Math:

Phương thức	Mục đích
Abs(n)	Trả về trị tuyệt đối của n
Atan(n)	Trả về Artang của n (n – radian)
Cos(n)	Trả về cosin của góc n (n – radian)

Exp(n)	Trả về e^n
Sign(n)	Trả về -1 nếu n<0 và 1 nếu n>0, 0 nế n=0
Sin(n)	Trả về sin của góc n (n – radian)
Sqrt(n)	Trả về căn bậc hai của n
Tan(n)	Trả về tang của góc n (n – radian)

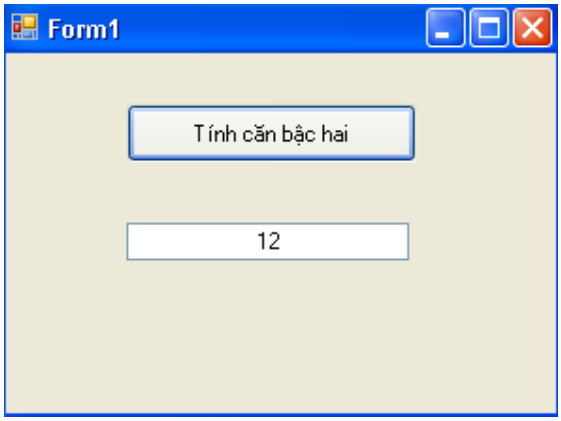
Bây giờ chúng ta làm quen với các phương thức trên thông qua ví dụ *MyFrameWorkMath*.

Tìm hiểu chương trình:

Chương trình này chỉ đơn giản là tính căn bậc hai của số 144 và đưa kết quả ra một ô textbox khi người dùng nhấn vào nút button có tên 'Tính căn bậc hai'.

Xây dựng giao diện:

Giao diện chỉ đơn giản như sau:



Viết mã:

- Khai báo sử dụng lớp *Math* trong chương trình: bạn đặt phát biểu sau ở bên trên phát biểu Public Class Form1:

```
Imports System.Math
```

Lệnh này cho phép bạn sử dụng, gọi đến phương thức, thuộc tính, đối tượng có trong lớp *Math* của thư viện hệ thống *System* dùng xử lý các số học.

- Tạo thủ tục `Button1_Click` bằng cách double click vào nút 'Tính căn bậc hai' và gõ đoạn mã sau:

```
Dim ketqua As Double
ketqua = Sqrt(144)
TextBox1.Text = ketqua
```

Chạy chương trình:

Bạn chạy chương trình như đã biết và xem kết quả.

10. Thứ tự ưu tiên của toán tử

Visual Basic qui ước thứ tự ưu tiên các toán tử như sau:

() → ^ → **-** (dấu âm) → ***/** (toán tử nhân, chia) → \ (phép chia nguyên) → **Mod** (lấy phần dư) → +- (toán tử cộng, trừ) .

11. Tổng kết chương

Như vẫn làm mỗi chương, chúng ta hãy tạo bảng tổng kết các công việc đã làm trong chương 5.

Bài tập:

- Mô phỏng mô hình máy tính đơn giản với các toán tử đã biết. Nhập dữ liệu bằng các nút như Calculator của hệ điều hành Windows.
- Tìm hiểu về phép tính toán trên các số nguyên lớn.

Chương 6:

Sử dụng các phát biểu cấu trúc ra quyết định

--------oOo--------

Nội dung thảo luận:

- Viết các biểu thức điều kiện
- Sử dụng phát biểu *If…Then* rẽ nhánh chương trình dựa vào một điều kiện
- Ước lượng tắt trong phát biểu *If…Then*
- Sử dụng phát biểu *Select…Case* để chọn quyết định trong số nhiều điều kiện
- Phát hiện và quản lý sự kiện chuột

1. Lập trình hướng sự kiện

Các chương trình chúng ta đã xây dựng bao gồm các đối tượng. Chúng ta tạo ra các đối tượng và dặt chúng lên form. Khi người dùng tương tác, họ sẽ quyết định xem sự kiện nào phát sinh trên đối tượng. Nói chung chương trình được tạo ra từ một tập các đối tượng thông minh chờ và phát sinh sự kiện do người dùng tương tác. Đây được gọi là lập trình hướng sự kiện – *Event-drivent Programming*.

Sự kiện có thể phát sinh do người dùng kích hoạt đối tượng hay có thể do hệ thống tự quyết định (như khi có email, chương trình sẽ phát sinh yêu cầu chúng ta xử lý).

VS.NET hỗ trợ sẵn rất nhiều sự kiện cho các đối tượng. Bạn có thể tìm thấy các sự kiện này trong ô thả xuống tại cửa sổ *Code Editor* khi chọn tên lớp ở *Class Name* và tên sự kiện tại *Method Name*.

Bây giờ chúng ta sẽ tìm hiểu về cấu trúc rẽ nhánh của phát biểu chương trình để thực hiện các sự kiện phát sinh.

2. Sử dụng biểu thức điều kiện

Một trong những cách xử lý mạnh mẽ nhất là dựa vào biểu thức điều kiện. Nó quyết định dựa trên kết quả so sánh điều kiện. Ví dụ:

gia < 1000

biểu thức này cho kết quả *True* nếu biến gia < 1000 và *False* nếu gia > 1000. Các toán tử so sánh có thể dùng trong biểu thức điều kiện:

Toán tử so sánh	**Ý nghĩa**
=	Bằng
<>	Khác
<	Nhỏ hơn
>	Lớn hơn

<=	Nhỏ hơn hoặc bằng
>=	Lớn hơn hoặc bằng

3. Phát biểu cấu trúc rẽ nhánh *If…Then*

Dạng đơn giản của một biểu thức rẽ nhánh:

```
If bieuthuc Then Thucthi
```

Trong đó *bieuthuc* là biểu thức điều kiện và *Thucthi* là phát biểu được gọi khi *bieuthuc* nhận giá trị True. Ví dụ

```
If gia <1000 then Label1.Text = "Giá rẻ, mua lăm cái!"
```

3.1. Kiểm tra nhiều điều kiện trong cấu trúc *If…Then*

Biểu thức If…Then còn có thể kiểm tra nhiều điều kiện một lúc và đưa ra nhiều quyết định khác nhau với việc kết hợp với các từ khóa như *ElseIf, Else và EndIf*:

```
If bieuthuc1 then
    Khối lệnh 1
ElseIf bieuthuc2
    Khối lệnh 2
ElseIf bieuthuc3
    Khối lệnh 3
...
Else
    Khối lệnh thực thi nếu không có giá trị bieuthuc(n) nào True
EndIF
```

Trong phát biểu trên, nếu *Bieuthuc1* đúng, thực hiện *Khối lệnh 1*; nếu *bieuthuc2* đúng, thực hiện *Khối lệnh 2*…

Ví dụ sau cho thấy cách sử dụng phát biểu rẽ nhánh này để xem xét số thuế phải nộp trong báo cáo tài chính:

```
Dim thunhap, thuenop As Double
thunhap=Cdbl(Textbox1.Text)

If thunhap <= 27050 Then
    thuenop = thunhap * 0.15
ElseIf thunhap <= 65550 Then
    thuenop = thunhap * 0.28
ElseIf thunhap <= 13675 Then
    thuenop = 132 + thunhap * 0.19
Else
    thuenop = 0
EndIf
```

Trong bài tập *MyUserValidation* dưới đây chúng ta sẽ dùng cấu trúc rẽ nhánh để kiểm tra tính hợp lệ của người dùng đăng nhập.

Chương trình có một ô textbox cho phép người dùng nhập tên. Khi click vào nút *đăng nhập* thì chương trình sẽ kiểm tra xem có đúng hay không để đăng nhập. Mỗi người dùng có một ảnh khác nhau hiển thị trong một *PictureBox*.

3.2. Chương trình *MyUserValidation*

Bây giờ chúng ta sẽ xây dựng chương trình.

Thiết kế giao diện:

Giao diện chương trình như sau:

Form gồm một Label, một textbox và một picturebox như hình. Bạn tạo mới một giải pháp và thêm vào một dự án có cùng tên *MyUserValidation* sau đó thiết kế giao diện như hình.

Từ dự án, R-Click vào *MyUserValidation* chọn Add | New Folder, gõ tên *Images*. Bạn copy hai ảnh bất kỳ vào đây sau đó cho hai ảnh này xuất hiện trong dự án bằng cách R-Click vào thư mục *Images* chọn Add | Existing Item… và chọn hai ảnh vừa copy vào.

Viết mã:

Tạo thủ tục `Button1_Click` và gõ đoạn mã sau:

```
        Dim UserName As String
        If TextBox1.Text = "" Then
            MsgBox("Bạn phải nhập UserName")
        Else
            UserName = TextBox1.Text
            If UserName = "Kelley" Then
                MsgBox("Chào bạn, Kelley!")
                PictureBox1.Image = System.Drawing.Image.FromFile _
                ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh
vb.net\Tung buoc lap trinh vb.net\6_Chapter6\Bai
tap\MyUserValidation\MyUserValidation\Images\Kelley.jpg")
            ElseIf UserName = "Sophie" Then
                MsgBox("Chào bạn, Sophie!")
                PictureBox1.Image = System.Drawing.Image.FromFile _
                ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh
vb.net\Tung buoc lap trinh vb.net\6_Chapter6\Bai
tap\MyUserValidation\MyUserValidation\Images\Sophie.jpg")
```

```
        Else
            MsgBox("Bạn không được phép đăng nhập!")
            End
        End If
    End If
```

Tìm hiểu mã:

Các mã tương đối đơn giản các bạn có thể tự tìm hiểu. Riêng việc lấy đường dẫn ảnh để load vào picturebox1 thì bạn không cần gõ đường dẫn, thay vào đó bạn có thể kéo ảnh từ thư mục *Images* bên cửa sổ dự án Solution Explorer vào dấu ngoặc kép sau phương thức FromFile như trong mã.

Chạy chương trình:

Các bạn chạy chương trình và lần lượt nhập vào các UserName là "Kelley", "Sophie" và một tên nào đó để xem kết quả.

3.3. Sử dụng các toán tử logic trong biểu thức điều kiện

Bạn có thể kiểm tra đồng thời nhiều điều kiện trong cùng một mệnh đề If then hay elseif nhờ các toán tử logic.

Toán tử Logic	**Ý nghĩa**
And	TRUE nếu cả hai cùng *True*.
Or	Nếu chỉ cần một biểu thức *True* thì giá trị TRUE. Nếu cả hai *False* thì kết quả FALSE
Not	Nếu một biểu thức *False* thì kết quả TRUE và ngược lại.
Xor	Nếu có duy nhất một biểu thức *True*, kết quả trả về là TRUE. Nếu cả hai cùng *True* hay cùng *False* thì kết quả trả về là FALSE

Bây giờ chúng ta bổ sung thêm việc đăng nhập vào chương trình *MyUserValidation* trên đây mật khẩu người dùng. Ta sử dụng các toán tử logic trên đây để kiểm tra tính hợp lệ của người dùng và pass nhập vào.

Bạn mở lại dự án trên đây nếu đã đóng lại. Thiết kế lại giao diện bằng cách bổ sung thêm một lable2 thuộc tính text là "Mật khẩu" và thêm một ô textbox thứ hai để nhập pass. Sửa thuộc tính *UseSystemPassWordChar* thành TRUE để giấu ký tự nhập vào.

Viết lại mã chương trình:

Bổ sung khai báo biến *Pass* như sau:

```
Dim UserName, Pass As String
```

Và nhập lại mã như thế này:

```
    If TextBox1.Text = "" Or TextBox2.Text = "" Then
        MsgBox("Bạn phải nhập UserName, PassWord")
    Else
```

```
            UserName = TextBox1.Text
            Pass = TextBox2.Text
            If UserName = "Kelley" And Pass = "kelley" Then
                MsgBox("Chào bạn, Kelley!")
                PictureBox1.Image = System.Drawing.Image.FromFile _
                ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh
vb.net\Tung buoc lap trinh vb.net\6_Chapter6\Bai
tap\MyUserValidation\MyUserValidation\Images\Kelley.jpg")
            ElseIf UserName = "Sophie" And Pass = "sophie" Then
                MsgBox("Chào bạn, Sophie!")
                PictureBox1.Image = System.Drawing.Image.FromFile _
                ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh
vb.net\Tung buoc lap trinh vb.net\6_Chapter6\Bai
tap\MyUserValidation\MyUserValidation\Images\Sophie.jpg")
            Else
                MsgBox("Bạn không được phép đăng nhập!")
                End
            End If
        End If
```

Như bạn thấy, giờ đây biểu thức điều kiện đã được bổ sung thêm các toán tử logic *OR* ở phát biểu If đầu tiên xem người dùng đã nhập đủ UserName và PassWord chưa. And ở phát biểu if thứ hai và thứ ba để kiểm tra đồng thời xem UserName và PassWord nhập vào có hợp lệ hay không.

Chạy lại chương trình:

Bạn chạy lai chương trình và nhập các giá trị UserName trùng PassWord để xem.

3.4. Ước lượng tắt sử dụng toán tử *AndAlso* và *OrElse*

VB.NET cung cấp hai toán tử logic mới là AndAlso và OrElse cho phép ước lượng tắt.

Giả sử phát biểu If có hai biểu thức điều kiện liên kết với nhau bằng toán tử AndAlso. Để phát biểu trong câu trúc được thực thi thì cả hai biểu thức điều kiện đều phải TRUE. Nếu một trong hai biểu thức mà False thì nó ngưng không ước lượng tiếp tiếp các điều kiện khác nữa mà chuyển sang mệnh đề ElseIf tiếp theo.

Toán tử OrElse tương tự. Nếu chỉ cần thấy một biểu thức TRUE thì phép ước lượng sẽ dừng lại. Tác dụng: tăng tốc độ ước lượng biểu thức → tăng tốc chương trình.

4. Phát biểu cấu trúc lựa chọn *Select Case*

Cấu trúc này ta đã biết trong các chương trước. Nó cho phép lựa chọn trường hợp và rẽ nhánh hiệu quả, dễ hiểu hơn If.

Cú pháp:

```
Select case giatri
    Case giatri1
        Khối lệnh 1
    Case giatri2
        Khối lệnh 2
    ...
    Case giatrin
        Khối lệnh n
    Case Else
        Khối lệnh thực thi nếu các khối lệnh trên sai
End Select
```

Bạn có thể xem ví dụ *InputControls* trong chương 3. Đây là mã chương trình:

```
        Select Case ComboBox1.SelectedIndex
            Case 0
                PictureBox6.Image = System.Drawing.Image.FromFile _
                ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh
vb.net\Tung buoc lap trinh vb.net\3_Chapter3\Bai
tap\InputControls\InputContorls\Images\Dollar.bmp")
                PictureBox6.Visible = True
            Case 1
                PictureBox6.Image = System.Drawing.Image.FromFile _
                ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh
vb.net\Tung buoc lap trinh vb.net\3_Chapter3\Bai
tap\InputControls\InputContorls\Images\Check.bmp")
                PictureBox6.Visible = True
            Case 2
                PictureBox6.Image = System.Drawing.Image.FromFile _
                ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh
vb.net\Tung buoc lap trinh vb.net\3_Chapter3\Bai
tap\InputControls\InputContorls\Images\PoundBag.bmp")
                PictureBox6.Visible = True
        End Select
```

Sử dụng các toán tử so sánh trong cấu trúc *Select Case*

Bạn có thể sử dụng các toán tử so sánh để đưa vào một phạm vi các giá trị lựa chọn trong mệnh đề case.

Các toán tử so sánh như <, >, =, <>, >=, <=. Để sử dụng toán tử so sánh, bạn cần thêm vào từ khóa *Is* hoặc *To*. Ví dụ:

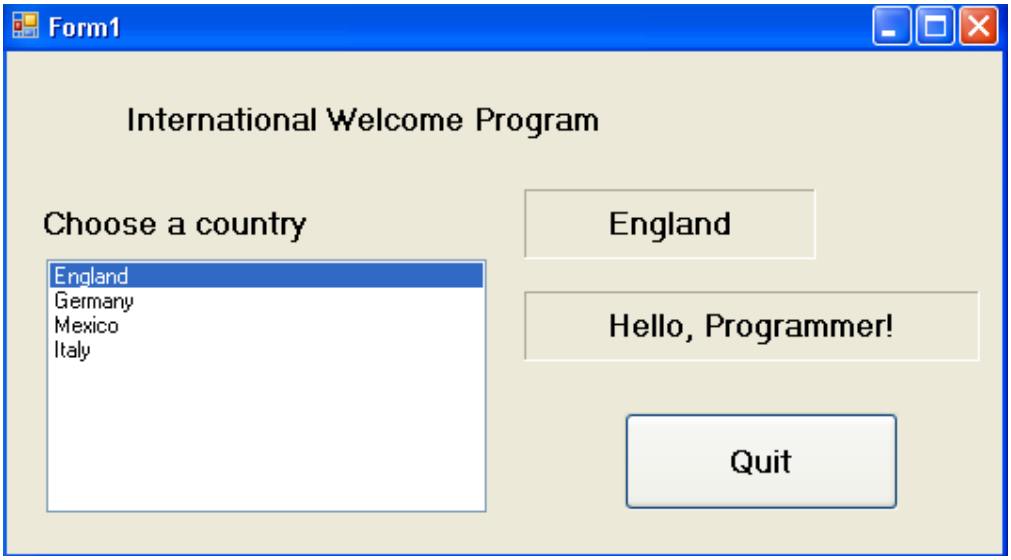
```
Select Case tuoi
    Case Is < 13
        MsgBox("Cần học thêm nhiều")
    Case 13 To 19
        MsgBox("Lứa tuổi teen")
    Case 21
        MsgBox("Bạn có thể kết hôn")
    Case > 100
        MsgBox("Đẹp lão!")
    Case Else
        MsgBox("Mừng thọ")
End Select
```

Nếu bạn có ít hơn 3 phát biểu so sánh thì bạn nên dùng câu lệnh rẽ nhánh If … Then.

Bây giờ chúng ta sẽ tìm hiểu kỹ hơn về cấu trúc này thông qua ví dụ *MyCaseGreeting*.

Tìm hiểu chương trình:

Giao diện chương trình:



Chương trình bao gồm một listbox liệt kê danh sách 4 nước. Khi người dùng click vào một nước nào đó thì tên nước hiện trên một nhãn và thông tin chi tiết hiện trên một nhãn khác.

Thiết kế giao diện:

Bạn tạo một giải pháp mới và add một dự án cùng tên *MyCaseGreeting* và thiết kế giao diện như trên.

Đặt tên đối tượng: trong các ví dụ trước chúng ta để các đối tượng với các tên mặc định vì đó là các chương trình đơn giản. Còn trong một dự án phức tạp, các đối tượng có số lượng tương đối lớn nên người lập trình cần đặt tên sao cho dễ nhớ. Trong ví dụ này, ta đặt tên như sau:

- Lable1: *Name – lbltittle, Text – "Internatinonal Welcome Program"*

- Lable2: *Name – lblchoose, Text – "Choose a country"*

- Lable3: *Name – lblcountry, Text – ""*

- Lable4: *Name – lblinfo, Text – ""*

- ListBox1: *name – lstcountry*

- Button1: *Name – btnquit, Text – "Quit"*

Các thuộc tính còn lại các bạn có thể tùy chọn.

Viết mã:

- Thêm các mục chọn là 4 nước trong lstcountry: các bạn có thể thêm các mục này khi thiết kế hoặc khởi tạo chúng tại sự kiện `Form1_Load` như thế này:

```
lstcountry.Items.Add("England")
lstcountry.Items.Add("Germany")
lstcountry.Items.Add("Mexico")
lstcountry.Items.Add("Italy")
```

- Tạo thủ tục `lstcountry_SelectedIndexChanged` để điền thông tin tên nước và thông tin lời chào tương ứng với ngôn ngữ các nước để chào người lập trình:

```
lblcountry.Text = lstcountry.Text
Select Case lstcountry.SelectedIndex
    Case 0
        lblinfo.Text = "Hello, Programmer!"
    Case 1
        lblinfo.Text = "Hallo, Programmierer!"
    Case 2
        lblinfo.Text = "Hola, Programador!"
    Case 3
        lblinfo.Text = "Ciao, Programmatore!"
End Select
```

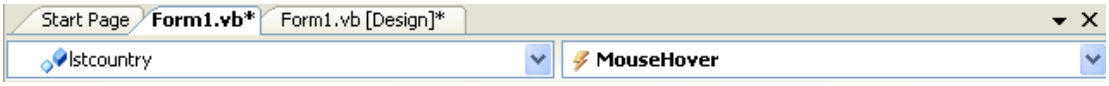
Chạy chương trình:

Bạn chạy chương trình bằng phím F5 hay start và xem các tính năng của chương trình.

5. Thêm bộ quản lý sự kiện chuột vào chương trình

Bây giờ chúng ta thử thêm vào chương trình bộ quản lý sự kiện chuột. Lúc này nếu người dùng click vào lstcountry nhưng nếu không click đúng vào một trong bốn mục thì chương trình lập tức hiện thông báo yêu cầu chọn một trong bốn mục đó.

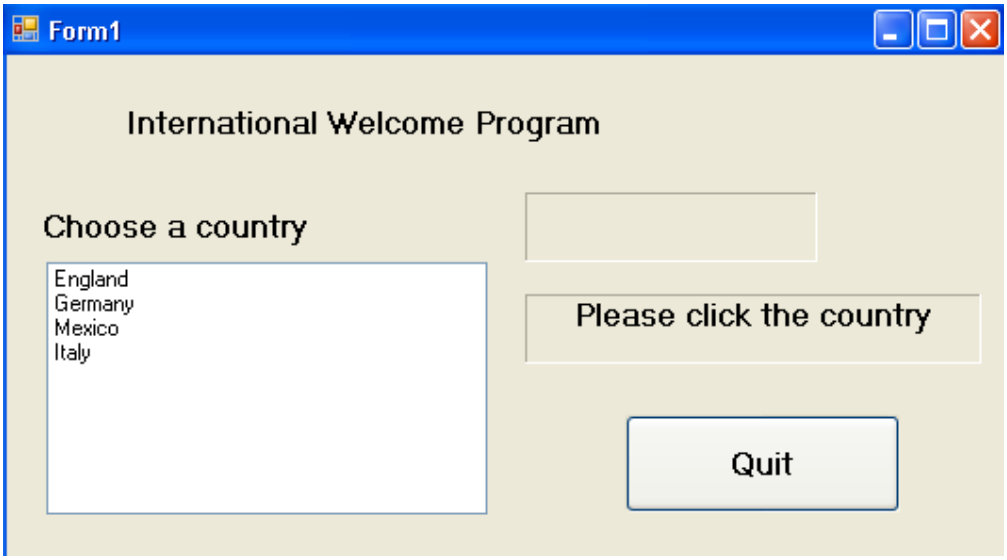
Để làm được điều này chúng ta tạo ra thủ tục `lstcountry_MouseHover` bằng cách sau:



Mở form ở chế độ viết mã code editor, chọn lstcountry ở ô class name và sự kiện MouseHover ở ô Method name và gõ đoạn mã sau:

```
If lstcountry.SelectedIndex < 0 Or _
lstcountry.SelectedIndex > 4 Then
    lblcountry.Text = ""
    lblinfo.Text = "Please click the country name!"
End If
```

Bây giờ bạn chạy lại chương trình xem có gì khác biệt hay không.



6. Tổng kết

Chúng ta lại tạo một bảng tổng kết những gì đã biết trong chương này.

Chương 7:

Sử dụng phát biểu lặp và bộ định thời (TIMER)

--------oOo--------

Nội dung thảo luận:

- Sử dụng vòng lặp *For ... Next*
- Hiển thị kết xuất trong ô TextBox nhiều dòng bằng phép nối chuỗi
- Sử dụng lệnh *Do ... Loop*
- Sử dụng đối tượng định thời *Timer* để thực thi mã lệnh tại một thời điểm
- Tạo chương trình đồng hồ số và công cụ đặt mật khẩu định thời

1. Vòng lặp *For…Next*

Vòng lặp này cho phép bạn thực thi lặp lại một nhóm hay nhiều lệnh trong một số lần nhất đinh.

Cú pháp:

```
For bien = batdau To ketthuc
      Khối lệnh gọi thực thi
Next
```

Ví dụ:

```
For i = 1 to 4
    Beep()
Next i
```

Đoạn mã trên đây sẽ phát ra bốn tiếng bíp bằng một vòng lặp for thay vì viết bốn hàm beep().

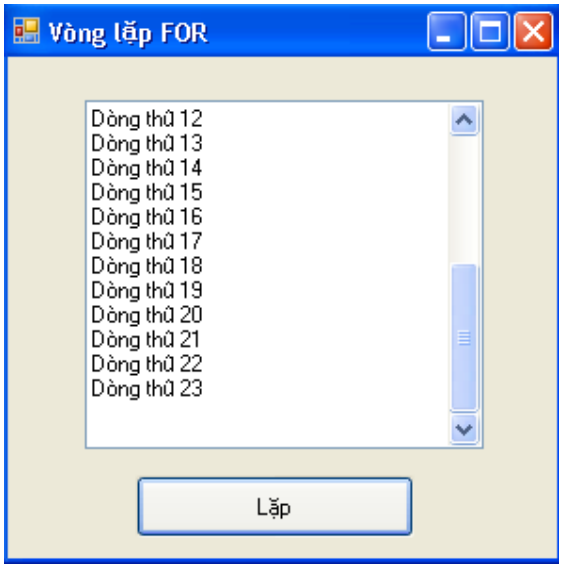
1.1. Hiển thị biến đếm của lệnh FOR trong điều khiển TEXTBOX

Biến đếm của vòng lặp FOR cũng như nhiều biến khác. Ta có thể sử dụng nó để gán hay hiển thị. Ví dụ *MyForLoop* sau đây sẽ hiển thị giá trị của biến đếm trong một ô texbox. Textbox này không chỉ có một dòng mà sẽ có nhiều dòng nhờ thay đổi thuộc tính *MultiLine* thành TRUE. Bạn cũng có thể tạo thanh cuộn đứng cho textbox bằng cách tạo thuộc tính *ScrollsBar* thành VERTICAL.

Tìm hiểu chương trình:

Chương trình khá đơn giản thế này: Chương trình có một ô textbox được thiết kế ở chế độ nhiều dòng và một nút.

Khi người dùng click vào nút trên thì chương trình thực hiện vòng lặp for và điền thông tin "dòng thứ" cùng với chỉ số biến đếm i từng dòng một.



Thiết kế giao diện:

Bạn tạo mới một giải pháp và add một dự án có cùng tên là *MyForLoop*.

Thiết kế giao diện như sau:

Đặt các thuộc tính cho đối tượng: Texbox1: *Name – txtline, MultiLine – TRUE, ScrollsBar – VERTICAL*. Button1: *Name – btnloop, Text – "Lặp"*

Viết mã:

Bạn tạo thủ tục `btnloop_Click` bằng cách double click vào nút "lặp" trên form và nhập vào đoạn mã sau:

```
Dim i As Integer
Dim ch As String
ch = Chr(13) & Chr(10)
txtline.Text = ""
For i = 1 To 23
    txtline.Text = txtline.Text & "Dòng thứ " & i & ch
```

```
Next
```

Tìm hiểu mã:

Chương trình khai báo hai biến: i kiểu số nguyên làm biến đếm cho vòng lặp for và ch kiểu xâu chuỗi – String, biến ch được gán giá trị 13 – canh lề và 10 – dòng mới.

Hàm *Chr()* giúp đổi số thành mã ASCII của một ký tự.

Chạy chương trình:

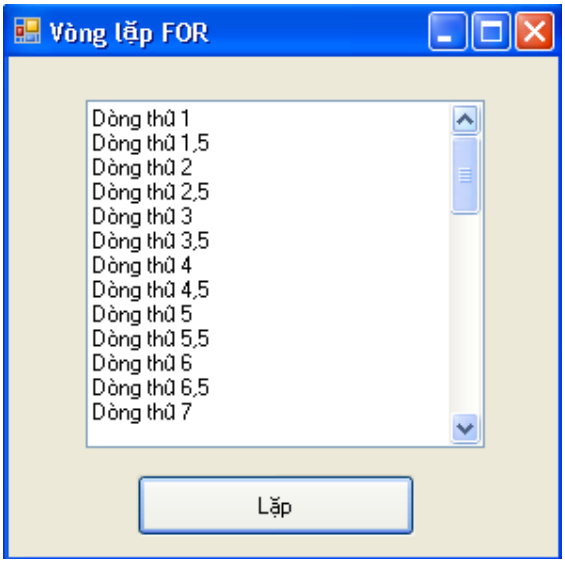
Bạn hãy chạy chương trình để xem vòng lặp FOR hoạt động như thế nào.

Nâng cao:

Bạn có thể tạo biến đếm của vòng lặp FOR kiểu khác, ví dụ DOUBLE như sau:

```
Dim i As Double
Dim ch As String
ch = Chr(13) & Chr(10)
txtline.Text = ""
For i = 1 To 23 Step 0.5
    txtline.Text = txtline.Text & "Dòng thứ " & i & ch
Next
```

Kết quả:



Trong đó `Step 0.5` là bước nhảy của biến đếm.

1.2. Tạo vòng lặp FOR…NEXT phức tạp hơn

Ví dụ sau đây sẽ minh họa việc sử dụng biến đếm để tạo tên file, mở và hiển thị các file .ICO chứa trên đĩa cứng. Chúng ta có thể chỉ dẫn nơi chứa các file .ICO nào chúng ta muốn, ở đây ta dùng các file sẵn có trong thư mục: Icos đã sao chép vào bên trong dự án.

Tìm hiểu chương trình:

Chương trình có một điều khiển PictureBox để hiển thị các ảnh ICO và một nút cho phép người dùng click. Khi người dùng click thì vòng lặp FOR bắt đầu. Nó sẽ duyệt tất cả các ảnh và hiển thị ảnh đó vào trong khung ảnh.

Thiết kế giao diện:

Bạn tạo mới một giải pháp và thêm vào một dự án cùng tên là *MyLoop_Advanced* đồng thời thiết kế giao diện như hình:

Trong đó thuộc tính Name của các đối tượng như sau: PictureBox1 – *ptbBieutuong,* Button1 – *btnhienbieutuong*.

Viết mã:

Tạo thủ tục `btnhienbieutuong_Click` bằng cách double click vào nút "hiện biểu tượng" và nhập vào đoạn mã sau:

```
        Dim i As Integer
        For i = 0 To 6
            ptbBieutuong.Image = System.Drawing.Image.FromFile _
            ("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh vb.net\" & _
            "Tung buoc lap trinh vb.net\7_Chapter7\Bai
tap\MyForloop_Advanced\" & _
            "MyForloop_Advanced\Icos\Face" & i & ".ico")
            MsgBox("Click để thấy hình tiếp theo")
        Next
```

Tìm hiểu mã:

Trong thư mục ICOS các file ảnh để tên trùng nhau phần đầu, chỉ khác nhau chỉ số cuối nên trong khai báo đường dẫn ta để biến I chạy `& i &` .

Vì vòng FOR chạy tương đối nhanh và ta không thể quan sát tất cả các ảnh hiển thị được nên dùng hàm MsgBox() để nhìn được ảnh mỗi khi biến đếm tăng lên 1.

Chạy chương trình:

Bạn chạy chương trình như đã biết.

Và đây là kết quả:

1.3. Sử dụng biến đếm toàn cục (Global)

Ở chương trình trên, để thấy được ảnh tiếp theo thì bạn phải click vào hộp thoại. Điều này gây khó chịu.

Bây giờ chung ta sẽ xây dựng lại chương trình này với biến toàn cục và phát biểu IF để khắc phục rắc rối trên.

Bạn mở lại dự án trên ở chế độ viết mã và khai báo biến *dem* đồng thời khởi tạo giá trị cho nó là 0 đặt ngay dưới dòng `Public Class Form1` như sau:

```
Dim dem As Integer = 1
```

Trong thủ tục `btnhienbieutuong_Click` bạn thay đổi như sau:

```
ptbBieutuong.Image = System.Drawing.Image.FromFile _
("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh vb.net\" & _
"Tung buoc lap trinh vb.net\7_Chapter7\Bai tap\" & _
"MyForloop_Advanced\MyForloop_Advanced\Icos\Face" & dem & ".ico")
dem += 1
If dem = 6 Then
    dem = 1
End If
```

Bạn chạy lại chương trình và thấy sau mỗi lần click vào nút "hiện biểu tượng" thì một ảnh tiếp theo hiện ra không cần click vào hộp thoại như trước nữa.

2. Vòng lặp DO LOOP

Thay vì chạy có giới hạn như FOR NEXT thì DO LOOPS sẽ chạy không giới hạn cho đến khi nào điều kiện ghi trong phần WHILE sai thì mới dừng lại.

Cú pháp:

```
Do while bieuthuc
    Khối lệnh
Loop
```

Với vòng lặp kiểu này thì nếu ngay từ đầu biểu thức sai ngay thì vòng lặp sẽ không thực hiện một lệnh nào trong khối lệnh. Nếu muốn vòng lặp thực thi ít nhất là một lệnh thì bạn dùng cú pháp sau:

```
Do
    Khối lệnh
Loop While Bieuthuc
```

2.1. Tránh vòng lặp vô tận

Nếu bạn thiết lập điều kiện không đúng thì vòng lặp có thể diễn ra vô tận. Ví dụ:

```
Dim i As Double = 12
Do
    i += 1
Loop While i > 12
```

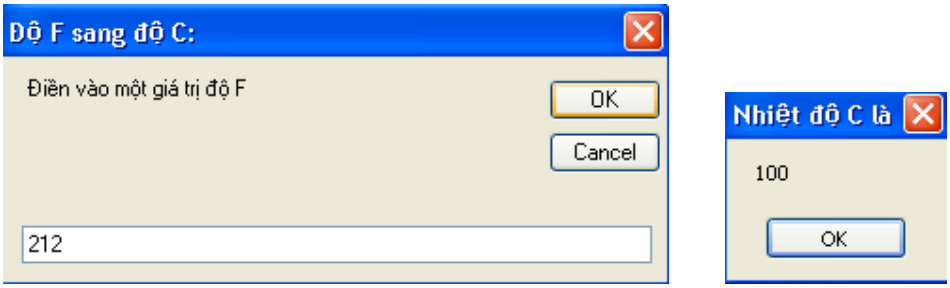
Vòng lặp trên không bao giờ dừng lại được vì điều kiện trên không bao giờ sai.

2.2. Dùng vòng lặp Do Loop viết chương trình chuyển đổi nhiệt độ

Tạo một giải pháp mới và thêm vào một dự án cùng tên *MyDoLoop_Temperature*. Bạn tạo sự kiện Form1_load bằng cách double click vào form và nhập vào đoạn mã sau:

```
Dim FTemp, Celsius As Single
Dim strFtemp As String
Dim Prompt As String = "Điền vào một giá trị độ F"
Do
    strFtemp = InputBox(Prompt, "Độ F sang độ C:")
    If strFtemp <> "" Then
        FTemp = CSng(strFtemp)
        Celsius = Int((FTemp + 40) * 5 / 9 - 40)
        MsgBox(Celsius, , "Nhiệt độ C là")
    End If
Loop While strFtemp <> ""
```

Chương trình này sẽ hiện một hộp thoại cho phép bạn nhập vào giá trị độ F để chuyển sang độ C. Vòng lặp sẽ thực thi liên tục, nếu bạn không nhập gì vào hộp thoại thì vòng lặp sẽ dừng lại.



Sử dụng từ khóa *Until* trong phát biểu Do Loop

Nếu ta dùng từ khóa *Until* thì chương trình se dừng lại khi nào điều kiện ước lượng nhận giá trị TRUE. Ví dụ chúng ta viết lại điều kiện `Loop while strFtemp <> ""` như sau:

```
Loop Until strFtemp = ""
```

Thì chương trình không thay đổi kết quả thực thi.

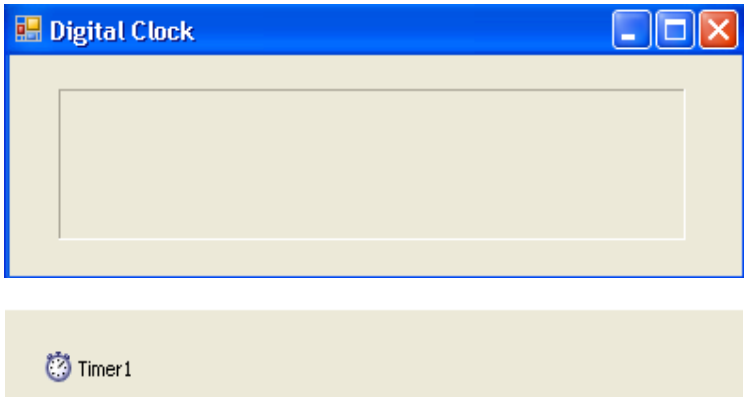
3. Sử dụng bộ định thời TIMER

Chúng ta có thể quy định một khối lệnh nào đó chỉ thực hiện trong một thời gian xác định với bộ định thời *Timer*. Sử dụng thuộc tính *Interval* và đặt *Enalble* của điều khiển Timer. Giá trị Interval bằng 1000 tương ứng với 1 giây. Khi được thiết đặt, timer sẽ phát sinh sự kiện *Time_Tick* để thực thi. Muốn dừng bộ định thời lúc nào thì đặt enable là fasle là được.

Bây giờ chúng ta sẽ tạo ví dụ *DigitalClock* tạo bộ đếm giờ với điều khiển Timer. Ta đặt bộ đếm giây, khi thuộc tính enable là true thì cứ sau 1 giây (giá trị interval là 1000) thì hệ thống phát sinh sự kiện Time_Tick gọi thủ tục bạn thực thi.

Thiết kế giao diện:

Giao diện form gồm một nhãn hiện thông tin ngày và giờ là một đồng hồ điện tử như sau:



Bạn tạo mới một giải pháp và add vào một dự án. Tạo form như trên bằng cách kéo một lable và đặt một điều khiển Timer1 vào form bằng cách nhắp đúp lên điều khiển này trên TOOLBOX. Bạn thiết đặt thuộc tính cho lable1 như hình.

Thuộc tính *Interval* của Timer1 bạn đặt là 1000 (ứng với 1 s), thuộc tính enable bạn chuyển thành TRUE.

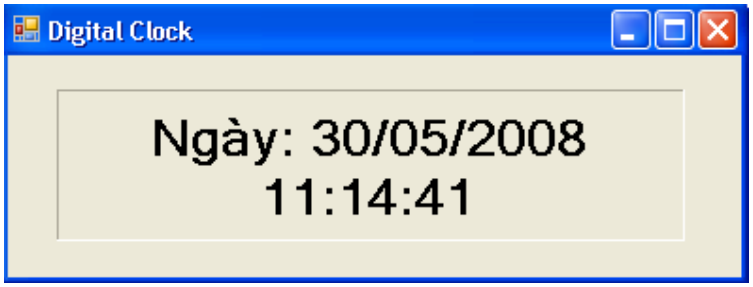
Viết mã:

Bạn tạo thủ tục `Timer1_Tick` bằng cách double click vào điều khiên Timer1 và nhập vào đoạn mã sau:

```
Label1.Text = "Ngày: " & DateTime.Now.Date & Chr(10)
Label1.Text &= TimeString
```

Chạy chương trình:

Bây giờ bạn hãy chạy chương trình và quan sát giao diện. Lúc này trên form có một đồng hồ điện tử như hình:



4. Sử dụng đối tượng TIMER để hạn chế thời gian

Ta có thể sử dụng điều khiển TIMER để khống chế cho một tác vụ nào đó. Bài tập sau đây *MyTimePassWord* sẽ sử dụng điều khiển TIMER để giới hạn thời gian nhập mật khẩu của người dùng.

Tìm hiểu chương trình:

Chương trình có một ô textbox cho phép nhập password. Nếu sau 15 giây mà người dùng không nhập mật khẩu chương trình đưa ra thông báo và kết thúc chương trình. Ngoài ra chương trình còn có một nút để click đăng nhập. Nếu người dùng không nhập pass thì thông báo lỗi chưa nhập pass, nhập đúng chuỗi "matkhau" thì báo thành công, ngược lại thì báo không đăng nhập được.

Thiết kế giao diện:

Tạo một giải pháp và thêm vào một dự án cùng tên là *MyTimePassWord* đồng thời thiết kế giao diện như hình trên.

Đặt thuộc tính *Interval* của Timer1 là 15000 ứng với 15s và enable là TRUE.

Viết mã:

Trước hết ta cài đặt thủ tục `Timer1_Tick` để xử lý khi quá 15 giây mà người dùng chưa đăng nhập. Bạn tạo thủ tục này bằng cách double click vào TIMER1 và nhập mã:

```
MsgBox("Rất tiếc, đã hết 15 giây.")
End
```

Bây giờ tạo thủ tục `Button1_Click` xử lý đăng nhập:

```
If TextBox1.Text = "" Then
```

```
        MsgBox("Bạn chưa nhập mật khẩu!")
    Else
        If TextBox1.Text = "matkhau" Then
            Timer1.Enabled = False
            MsgBox("Bạn đã đăng nhập thành công!")
        Else
            MsgBox("Bạn không có quyền truy cập!")
        End If
    End If
```

Chạy chương trình:

Bạn chạy chương trình và xem các tính năng của ví dụ trên.

5. Tổng kết

Bạn làm bảng tổng kết như mọi chương và làm thêm một số bài tập đơn giản sau:

- Viết lại ví dụ MyUserValidation ở chương 6 sử dụng bộ định thời để giới hạn thờ gian nhập password.

Chương 8:

Gỡ lỗi (DEBUG) trong chương trình VISUAL BASIC.NET

--------oOo--------

Nội dung thảo luận:

- Các kiểu lỗi khác nhau trong chương trình
- Sử dụng công cụ gỡ lỗi trong VS.NET đặt điểm dừng cho chương trình
- Sử dụng cửa sổ Watch kiểm tra các giá trị của các biến khi thực thi chương trình
- Sử dụng cửa sổ Command để thay đổi giá trị biến và thực thi lệnh trực tiếp

Trong các chương trước, chúng ta đã viết nhiều chương trình khác nhau và chưa hoàn thiện vì nó có khá nhiều lỗi có thể xảy ra. Chương này chúng ta sẽ tìm hiểu kỹ hơn về lỗi và cách gỡ lỗi.

1. Tìm kiếm và hiệu chỉnh lỗi

1.1. Các loại lỗi

Có ba loại lỗi thường xảy ra, gồm:

- Lỗi cú pháp (Syntax Error): Còn được gọi là lỗi biên dịch – Compiler error. Lỗi này do bạn gõ sai cấu trúc ngôn ngữ. Lỗi này bộ soạn thảo mã có thể bắt được (xuất hiện dòng gạch chân màu xanh loằng ngoằng dưới dòng mã gây lỗi).
- Lỗi thực thi (Runtime error): Xảy ra bất ngờ khi chương trình đang chạy.
- Lỗi logic: Lỗi do tư duy sai dẫn đến kết quả sai với dự kiến nó phải như thế.

Khi gặp lỗi thực thi thì ta cần chú ý đến việc xử lý dữ liệu động nhập vào đúng và hợp lý.

Có rất nhiều lỗi yêu cầu ta phải có giải pháp thích hợp thông qua bộ xử lý lỗi (error handler). Nó là một đoạn chương trình có khả năng phát hiện các lỗi khác nhau và đưa ra giải pháp thích hợp để xử lý.

1.2. Phát hiện lỗi LOGIC

Để phát hiện lỗi này thì chương trình cần chạy nhiều lần với nhiều kết quả để xem nó có phù hợp hay không.

2. Dò lỗi từng dòng lệnh – sử dụng chế độ ngắt (BREAK MODE)

Một trong những cách gỡ lỗi hiệu quả là chạy từng dòng mã và kiểm tra nội dung của một hay nhiều biến. Để thực hiện điều này, bạn chuyển sang chế độ ngắt. Khi đó chương trình vẫn chạy nhưng ở cửa sổ Code Editor.

Bây giờ với ví dụ *DebugTest* chúng ta sẽ học cách đặt điểm dừng (breakpoint) và chuyển chương trình sang chế độ ngắt để kiểm tra lỗi.

Để cô lập lỗi, bạn sử dụng nút Step into  trên thanh standard bar và cửa sổ Autos để kiểm tra giá trị các biến cũng như thuộc tính chính trong chương trình.

Ví dụ *DebugTest*:

Tạo mới một dự án có tên *DebugTest* như đã biết và thiết kế form như sau:

Viết mã:

Bạn tạo thủ tục Button1_click và nhập vào đoạn mã sau:

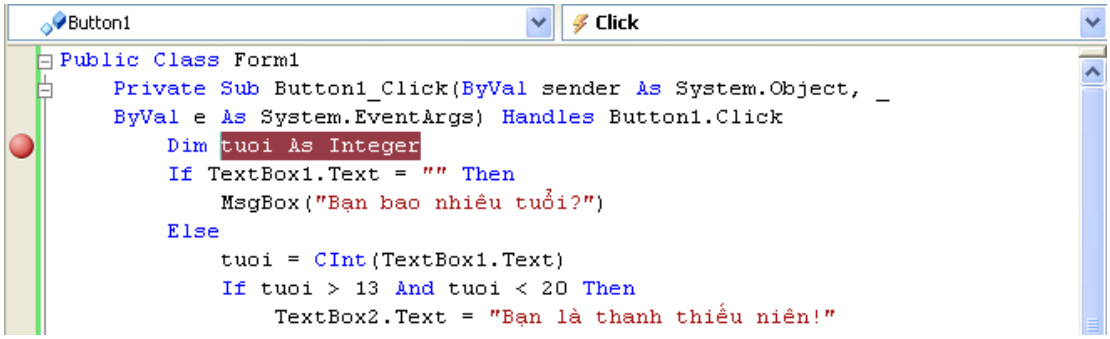
```
Dim tuoi As Integer
If TextBox1.Text = "" Then
    MsgBox("Bạn bao nhiêu tuổi?")
Else
    tuoi = CInt(TextBox1.Text)
    If tuoi > 13 And tuoi < 20 Then
        TextBox2.Text = "Bạn là thanh thiếu niên!"
    Else
        TextBox2.Text = "Bạn không phải là thanh thiếu niên"
    End If
End If
```

Chương trình này sẽ phát sinh lỗi logic: lứa tuổi 13 cũng là thanh thiếu niên nhưng khi người dùng nhập vào tuổi 13 chương trình vẫn xem như đây không phải là thanh thiếu niên.

Bây giờ chúng ta sẽ dùng điểm dừng để kiểm tra xem lỗi này do dòng mã nào gây ra:

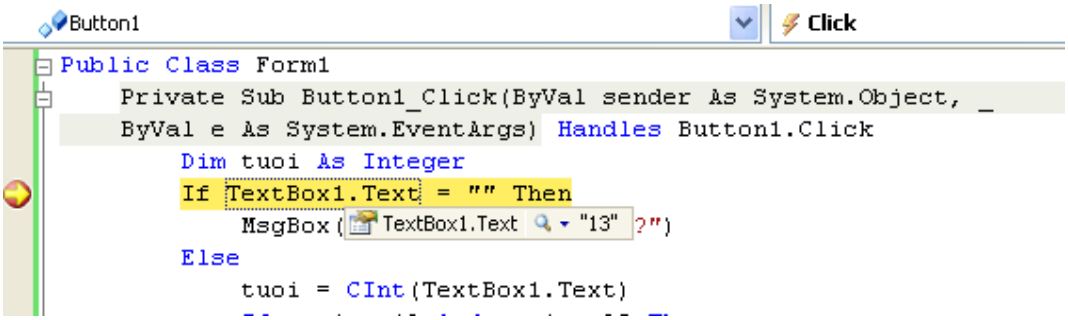
Đặt điểm dừng (BreakPoint):

Bạn mở form ở chế độ soạn thảo mã và click chuột vào lề trái của đoạn mã như hình để làm xuất hiện một dòng sáng:



Nhấn F5 hay nút start để chạy chương trình.

Gõ giá trị 13 vào ô textbox thứ nhất và ấn nút "Kiểm tra". Lúc này chương trình trở về cửa sổ code editor và xuất hiện dòng vàng ở dòng ta đặt điểm dừng như thế này:



Ở chế độ này ta có thể xem rất nhiều thứ mà chương trình đang diễn ra. Bạn có thể xem giá trị hiện hành của biến *tuoi* bằng cách di chuột lên biến *tuoi*, giá trị là 0 tuoi 0. Bạn cũng có thể thay đổi giá trị biến hay giá trị nhập vào của textbox1.

Nhấn nút Step Into hay F11 để chuyển sang dòng lệnh tiếp theo.

Giờ bạn mở cửa sổ Autos bằng cách chọn Debug | Windows | Autos. Cửa sổ này cho phép bạn xem tất cả những gì xảy ra khi chương trình chạy.

Tiếp tục ấn nút Step Into hay F11 ba lần nữa và quan sát. Lúc này phát biểu If thấy điều kiện so sánh sai (13 không nằm trong khoảng 13 →20) và chuyển đến mệnh đề Else. Đây chính là lỗi mà chúng ta cần tìm. Vậy là ta cần xem xét lại biểu thức so sánh trong phát biểu If.

Bạn dừng chạy bằng cách nhấn nút stop và sửa lại toán tử so sánh > thành >=13 rồi lưu lại thay đổi này.

Bỏ điểm dừng:

Bạn bỏ điểm dừng bằng cách click chuột vào điểm màu đỏ của dòng mã đã thiết đặt điểm dừng khi trước là xong.

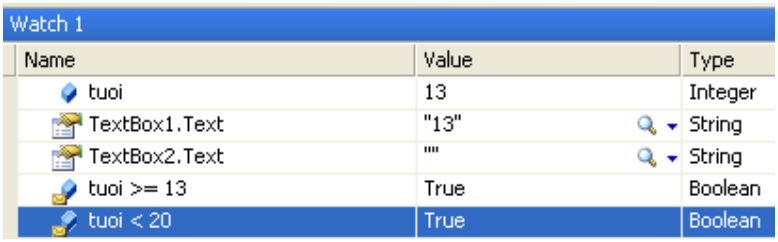
Chạy lại chương trình và kiểm tra xem kết quả đã thay đổi hợp lý hay chưa.

3. Theo dõi các biến bằng cửa sổ WATCH

Bạn có thể theo dõi các biến bằng cửa sổ Autos nhưng cửa sổ này sẽ không hiển thị tất cả các biến trong chương trình, nó chỉ hiển thị biến của dòng lệnh đang thực thi hay hàm đang thực thi mà thôi.

Để xem toàn bộ nội dung cac biến bạn có thể dùng cửa sổ WATCH. Trong VB.NET, bạn có thể mở một lúc tới 4 cửa sổ WATCH. Để mở bạn chọn Debug | Windows | Watch.

Bạn mở lại dự án trên và chạy lại ở chế độ ngắt. Mở Watch1 theo cách trên. Để theo dõi biến hay biểu thức nào bạn chọn nó bằng cách bôi đen và R-Click rồi chọn Add Watch. Kết quả:



Để loại bỏ một biến hay biểu thức ra khỏi watch bạn chọn nó và ấn Delete là xong.

4. Sử dụng cửa sổ COMMAND

Cửa sổ command cho phép ta thay đổi giá trị biến và bổ sung một số lệnh. Nó cho phép tương tác trực tiếp vơi VB. Ở chế độ Imediate (tức thời) cửa sổ cho phép ta tương tác trực tiếp với hầu hết các lệnh của VS như Save, Print…

4.1. Mở cửa sổ COMMAND trong chế độ Imediate

Để mở, bạn chọn Debug | Windows | Imediate.

Kiểm thử bằng cách gõ *tuoi = 18* vào cửa sổ này ấn enter. Bạn đã yêu cầu thay đổi giá trị biến thành 18. Giờ nếu bạn xem trong cửa sổ watch thì giá trị tuoi là 18.

4.2. Chuyển sang chế độ command

Cửa sổ command còn cho phép làm việc ở chế độ command để tương tác trực tiếp với VB như File.SaveAll chẳng hạn.

Để chuyển, bạn có thể gõ >cmd (enter) trong cửa sổ Immediate.

Gõ thử File.SaveAll (Enter)

5. Tổng kết

Bạn làm bảng tổng kết như các chương.

Các chương trình trước đây ta viết có nhiều lỗi có thể xảy ra. Bạn thử chạy lại chúng, nhập nhiều giá trị đặc biệt xem co lỗi nào phát sinh không đồng thời khắc phục thử xem.

Chương 9:

Bẫy lỗi và sử dụng cấu trúc xử lý lỗi

--------oOo--------

Nội dung thảo luận:

- Quản lý các lỗi thực thi chương trình bằng phát biểu *Try…Catch*.
- Kiểm tra một số điều kiện lỗi đặc trưng bằng phát biểu *Catch When*
- Sử dụng thuộc tính Err.Number và Err.Description để xác định các lỗi ngoại lệ
- Sử dụng phát biểu *Try…Catch*
- Sử dụng các bộ xử lý lỗi kết hợp với các kỹ thuật phòng vệ lỗi khác
- Thoát khỏi bộ xử lý lỗi bằng phát biểu *Exit Try*

Chương này chúng ta sẽ xây dựng các khối mã tự xử lý lỗi phát sinh, còn gọi là các ngoại lệ. Ta dùng khối *Try…Catch* để bẫy những lỗi này và làm nó không ảnh hưởng đến luồng thực thi.

Các tính năng mới của bắt lỗi trong VB.NET:

- Phát biểu *Catch When* cho phép kiểm tra một số lỗi đặc trưng ngay trong khối Try…Catch
- Phát biểu *Exit Try* cho phép ta thoát khỏi khối bất cứ lúc nào
- Các đối tượng *Err* và thuộc tính *Err.Number, Err.Description* cho phép xác định mã lỗi. Phương thức mới *Err.GetException* trả về thông tin của lỗi ngoại lệ phát sinh.

1. Xử lý lỗi sử dụng cú pháp *Try…Catch*

Lỗi có thể phát sinh bất cứ lúc nào. Ví dụ như khi bạn nạp một file mà không có thực trong đĩa thì chương trình sẽ gặp lỗi. VB có khả năng xử lý nhưng nhiệm vụ của bạn là phải thông báo cho VB biết. Chính vì thế khối lệnh *Try…Catch* sẽ bao bọc đoạn mã lệnh có khả năng gây ra lỗi cho chương trình. Thông thường có các lỗi xảy ra do nhập xuất dl, phép chia cho 0, thiết bị ngoại vi không sẵn sàng.

1.1. Cú pháp *Try…Catch*

```
Try
    Các phát biểu có thể gây lỗi
Catch
    Các phát biểu xử lý nếu có lỗi phát sinh
Finally
    Các phát biểu được gọi ngay cả khi có hay không có lỗi
End Try
```

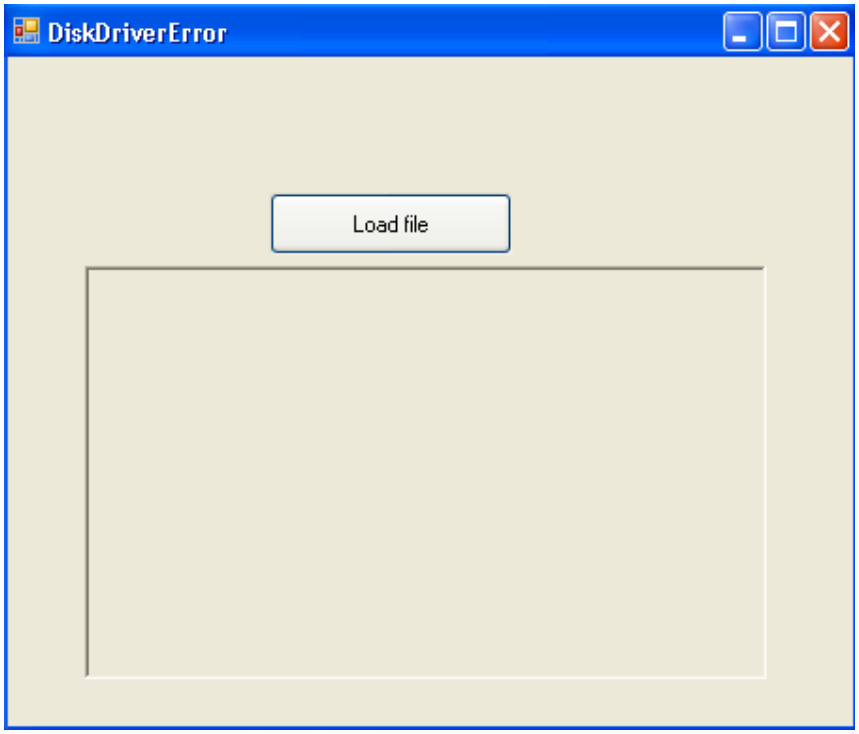
Trong đó *Finally* là tùy chọn, các từ khóa còn lại là bắt buộc.

1.2. Các lỗi về đường dẫn và ổ đĩa

Ví dụ sau *DiskDriverError* sẽ minh họa tình huống xử lý lỗi runtime thường thấy nhất. Chúng ta tạo một form có nút nhấn và một ô ảnh PictureBox. Khi click vào nút thì ảnh trong một đĩa mềm có tên 6_82MELINH.ico sẽ load vào ô ảnh. Nếu bỏ đĩa mềm ra khỏi ổ mềm thì chạy chương trình sẽ báo lỗi không tìm thấy đĩa trong ổ A:\ ngay.

Thiết kế Form:

Bạn mở mới một dự án và thiết kế form như hình:

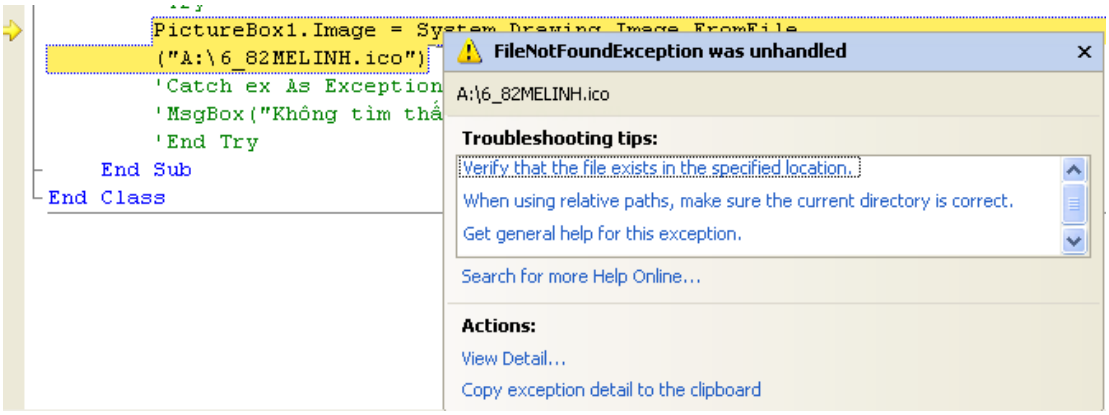


Viết mã:

Tạo thủ tục Button1_Click và gõ mã như sau:

```
PictureBox1.Image = System.Drawing.Image.FromFile _
("A:\6_82MELINH.ico")
```

Lúc này trong ổ mềm không có đĩa nên khi chạy chương trình sẽ có thông báo lỗi xảy ra



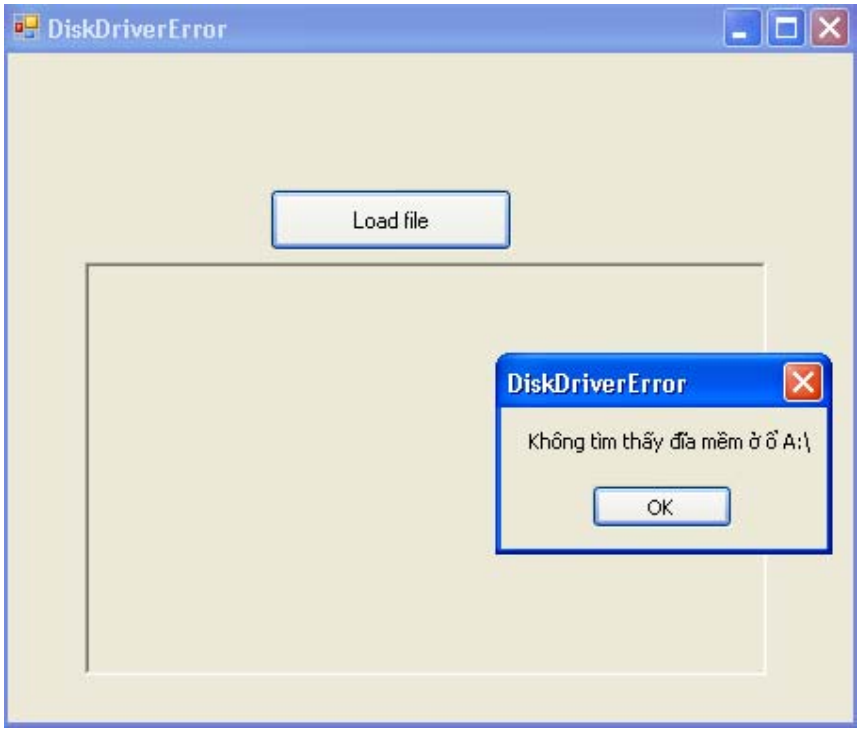
Để khắc phục ta đặt thêm khối try … catch vào như thế này:

2. Cài đặt cơ chế xử lý lỗi đọc đĩa

Bạn sửa lại thủ tục Button1_click như sau:

```
Try
    PictureBox1.Image = System.Drawing.Image.FromFile _
    ("A:\6_82MELINH.ico")
Catch ex As Exception
    MsgBox("Không tìm thấy đĩa mềm ở ổ A:\")
End Try
```

Lúc này phát biểu gây lỗi `PictureBox1.Image = System.Drawing.Image.FromFile _` đã được đặt ở trong khối *Try…Catch* nên khi chạy chương sẽ thực thi hiện thông báo thay vì phát sinh lỗi như trên:



2.1. Sử dụng mệnh đề *Finally* để thực hiện tác vụ dọn dẹp

Mệnh đề này sẽ cho phép dùng các phát biểu sau nó dù có hay không có lỗi xảy ra. Nó thuận tiện khi bạn muốn dọn dẹp lỗi, giá trị của biến, thuộc tính khi bạn thực thi đoạn mã bảo vệ xong.

Trở lại ví dụ trên, ta thêm vào đoạn mã như sau:

```
Try
    PictureBox1.Image = System.Drawing.Image.FromFile _
    ("A:\6_82MELINH.ico")
Catch ex As Exception
    MsgBox("Không tìm thấy đĩa mềm ở ổ A:\")
Finally
    MsgBox("Đã bắt lỗi thành công.")
End Try
```

Và chạy lại chương trình để xem nó hoạt động như thế nào.

2.2. Cài đặt *Try…Catch* phức tạp hơn

Khi chương trình phức tạp thì việc bắt lỗi cũng trở nên phức tạp hơn. Với Try…Catch bạn có thể:

- Đặt một khối hay nhiều khối phát biểu giữa các từ khóa.
- Cho phép sử dụng mệnh đề lọc lỗi *Catch When*
- Cho phép sử dụng khối *Try…Catch* lồng nhau
- Cùng với đối tượng *Err* cho phép xác định lỗi phát sinh

Đối tượng Err:

Đây là đối tượng đặc biệt cung cấp chi tiết thông tin lỗi phát sinh. Các thuộc tính thông dụng Err.Number, Err.Description chứa thông tin mã lỗi, mô tả chi tiết lỗi. Phương thức Err.Clear cho phép xóa bỏ lỗi hiện hành. Bảng sau đây liệt kê các lỗi Runtime thường gặp trong VB:

Mã lỗi (Err.Number)	Mô tả
5	Gọi hàm hay truyền đối số không đúng
6	Tràn
7	Hết bộ nhớ
9	Truy xuất vượt chỉ số mảng
11	Chia cho 0
13	Kiểu không hợp lệ
48	Lỗi nạp thư viện DLL
51	Lỗi nội bộ
52	Tên File hay số không hợp lệ
53	Không tìm thấy File
55	File đang mở
57	Lỗi thiết bị xuất nhập
58	File đã tồn tại
61	Đĩa đầy
62	Con trỏ file vượt quá điểm cuối file
67	File mở quá nhiều
68	Thiết bị chưa sẵn sàng
70	Không cho phép truy xuất
71	Ổ đĩa chưa sẵn sàng
75	Truy cập đường dẫn và file không đúng

76	Không thấy đường dẫn
91	Biến đối tượng thiếu từ khóa truy xuất *With*
321	Định dạng file không hợp lệ
322	Không thể tạo file tạm
380	Giá trị thuộc tính không hợp lệ
381	Chỉ số thuộc tính không hợp lệ
422	Thuộc tính không tìm thấy
423	Thuộc tính hay phương thức không có
424	Yêu cầu về đối tượng
429	Không thể tạo đối tượng ActiveX
430	Lớp đối tượng không hỗ trợ Automation
440	Không thể tạo đối tượng Automation
460	Định dạng trong Clipboard không hợp lệ
461	Phương thức hay biến thành viên không tìm thấy
462	Server không sẵn sàng
463	Lớp không đăng ký trên máy cục bộ
481	Ảnh không hợp lệ
482	Máy in bị lỗi

Bây giờ vẫn dùng ví dụ trên nhưng ta thêm thuộc tính Err.Number, Err.Description đồng thời ta cũng tìm hiểu thêm về mệnh đề đọc lỗi *Catch When*.

Bạn sửa lại thủ tục Button1_Click như sau:

```
Try
    PictureBox1.Image = System.Drawing.Image.FromFile _
    ("A:\6_82MELINH.ico")
Catch When Err.Number = 53 'nếu không thấy file
    MsgBox("Kiểm tra lại đường dẫn và tên file")
Catch When Err.Number = 7  'Hết bộ nhớ
    MsgBox("File ảnh quá lớn - hết bộ nhớ", , Err.Description)
Catch ex As Exception
    MsgBox("Không tìm thấy đĩa mềm ở ổ A:\", , Err.Description)
Finally
    MsgBox("Đã bắt lỗi thành công.")
End Try
```

Trong khối lệnh trên ta sử dụng mệnh đề *Catch When* hai lần, mỗi lần ta sử dụng thêm các thuộc tính Number của đối tượng Err để phát hiện lỗi cụ thể hơn.

Bạn chạy lại chương trình xem nó hoạt động ra sao.

Tự mình phát sinh lỗi:

Trong một số trường hợp bạn có thể tự kiểm tra lỗi trong mệnh đề *Try* và muốn nhảy ngay đến mệnh đề *Catch* để lỗi được xử lý. Khi đó VB.NET cung cấp phương thức *Err.Raise* để làm điều đó. Ví dụ ta có thể tự phát hiện ra lỗi không tìm thấy *File* ở ví dụ trên (lỗi 53) và thực hiện phát biểu trong mệnh đề Catch:

```
Try
    PictureBox1.Image = System.Drawing.Image.FromFile _
    ("A:\6_82MELINH.ico")
    If Err.Number = 53 Then Err.Raise(53)
Catch When Err.Number = 53
    MsgBox("Không tìm File")
End Try
```

2.3. Xác định số lần thử lại

Một trong những đặc sắc của *Try…Catch* là cho phép bạn thử lại một số thao tác gây ra lỗi trước khi đưa ra quyết định không thực hiện thao tác này nữa. Ví dụ ta có thể xem số lần người dùng click vào nút "Load File" bao nhiêu lần, nếu vượt quá số lần cho phép thì không cho người dùng click tiếp nữa:

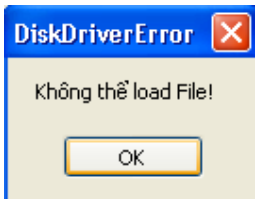
Khai báo thêm biến *dem* ở dưới dòng public class form1:

```
Dim dem As Short = 0
```

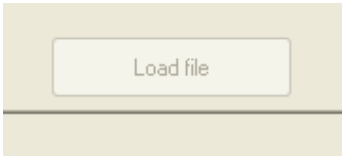
Sửa lại thủ tục Button1_Click như sau:

```
Try
    PictureBox1.Image = System.Drawing.Image.FromFile _
    ("A:\6_82MELINH.ico")
Catch ex As Exception
    dem += 1
    If dem <= 2 Then
        MsgBox("Không tìm thấy đĩa mềm ở ổ A:\")
    Else
        MsgBox("Không thể load File!")
        Button1.Enabled = False
    End If
End Try
```

Và bây giờ khi người dùng click vào nút "Load File" quá hai lần thì thông báo xuất hiện:



Và nút "Load File" sẽ bị mờ đi không cho người dùng click nữa như thế này:



2.4. Sử dụng các khối *Try…Catch* lồng nhau

Bạn có thể sử dụng các khối Try…Catch lồng nhau để kiểm tra kép các thao tác có thể gây lỗi. Ví dụ bây giờ ta sửa lại ví dụ trên để người dùng phải đưa đĩa mềm vào ổ A:\ ngay từ lần thông báo lỗi đầu tiên, nếu không nút "Load File" lập tức sẽ bị vô hiệu hóa. Code:

```
Try
    PictureBox1.Image = System.Drawing.Image.FromFile _
    ("A:\6_82MELINH.ico")
Catch
    MsgBox("Không tìm thấy đĩa mềm ở ổ A:\, cho đĩa mềm vào")

    Try
        PictureBox1.Image = System.Drawing.Image.FromFile _
        ("A:\6_82MELINH.ico")
    Catch ex As Exception
        MsgBox("Không thể load file!")
        Button1.Enabled = False
    End Try
End Try
```

Bạn nên sử dụng việc lồng hai phát biểu Try…Catch lồng nhau trong trường hợp kiểm tra lại lỗi 2 lần. Còn nếu kiểm tra nhiều lần thì bạn nên sử dụng kết hợp với các biến đếm và vòng lặp *For, Do Loop*.

3. So sánh cơ chế xử lý lỗi với các kỹ thuật phòng vệ lỗi

Bạn có thể đoán trước xem lỗi nào có thể xảy ra để phòng trước thay vì xử lý lỗi bằng Try…Catch. Ví dụ trong bài tập trên, thay vì dùng Try ta sẽ dùng phương thức của hệ thống là *File.Exists* kiểm tra xem có tồn tại file hay không rồi mới gọi phương thức nạp ảnh *FromFile*:

Để dùng được phương thức này, bạn cần khai báo sử dụng thư viện IO bằng từ khóa *Imports* ở đầu khối lệnh:

```
Imports System.IO
```

Rồi sửa lại mã lệnh trong thủ tục Button1_Click như sau:

```
'Phòng vệ lỗi
If File.Exists("A:\6_82MELINH.ico") Then
    PictureBox1.Image = System.Drawing.Image.FromFile _
    ("A:\6_82MELINH.ico")
Else
    MsgBox("Không tồn tại file này!")
End If
```

Việc sử dụng phương thức nào là do bạn quyết định và trong hoàn cảnh nào thì dùng phương thức nào cho hợp lý.

4. Sử dụng phát biểu thoát *Exit Try*

Phát biểu này là tùy chọn trong khối Try…Catch. Nó giúp bạn thoát khỏi khối Try…Catch khi muốn.

Tuy nhiên nếu trong khối Try…Catch có phát biểu Finally thì chương trình sẽ thực thi các phát biểu trong phần Finally trước khi thoát khỏi khối Try theo yêu cầu của Ext Try.

Ví dụ như sau:

```
'Thoát Try với Exit Try
Try
    If PictureBox1.Enabled = False Then Exit Try
    PictureBox1.Image = System.Drawing.Image.FromFile _
    ("A:\6_82MELINH.ico")
Catch ex As Exception
    MsgBox("Không tìm thấy File này!")
End Try
```

Trong đoạn mã trên, nếu chương trình kiểm tra xem điều khiển PictureBox1 mà chưa sẵn sàng thì lập tức thoát khỏi khối Try…Catch mà không thực hiện đưa ra thông báo nào.

5. Tổng kết chương

Bạn lập bảng tông kết chương và dùng khối Try…Catch để xử lý các lỗi có thể gây ra trong các bài tập của các chương trước đây.

Chương 10:

Sử dụng các MODULE (đơn thể) và thủ tục (PROCEDURE)

--------oOo--------

Nội dung thảo luận:

- Tạo các module chuẩn
- Khai báo và sử dụng các biến *Public* toàn cục
- Tạo các hàm và thủ tục tự định nghĩa bởi người dùng
- Gọi thực thi hàm và thủ tục do người dùng cài đặt

Module là nơi chứa các biến, các hàm, thủ tục và có thể triệu gọi từ bất cứ nơi nào trong chương trình.

1. Làm việc với MODULE chuẩn

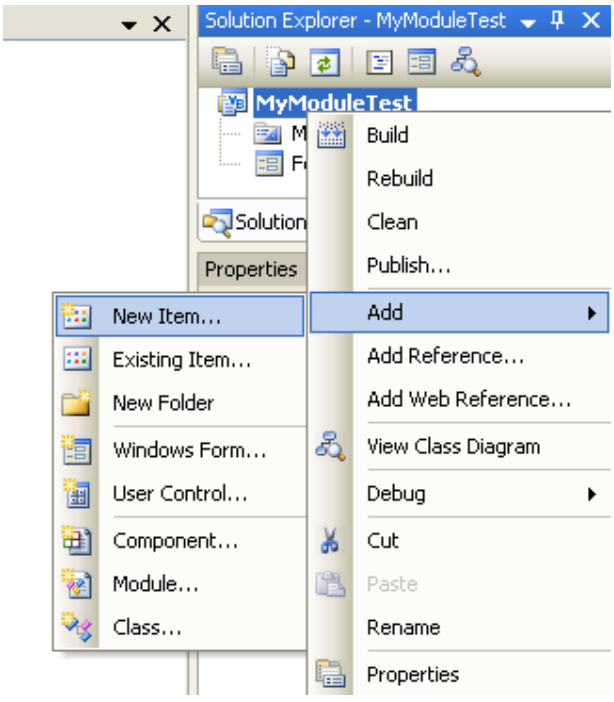
Khi dự án của bạn rất lớn thì việc có nhiều form là điều đương nhiên. Có điều bạn không thể sử dụng những hàm, biến khai báo trong form này cho form kia được.

Để chia sẻ biến và các hàm, thủ tục giữa các form trong dự án thì bạn có thể khai báo chúng trong một module của dự án. Module là một file có đuôi mở rộng .vb chỉ chứa các mã. Bạn có thể lưu module bằng cách chọn Flie | Save Module1 As.

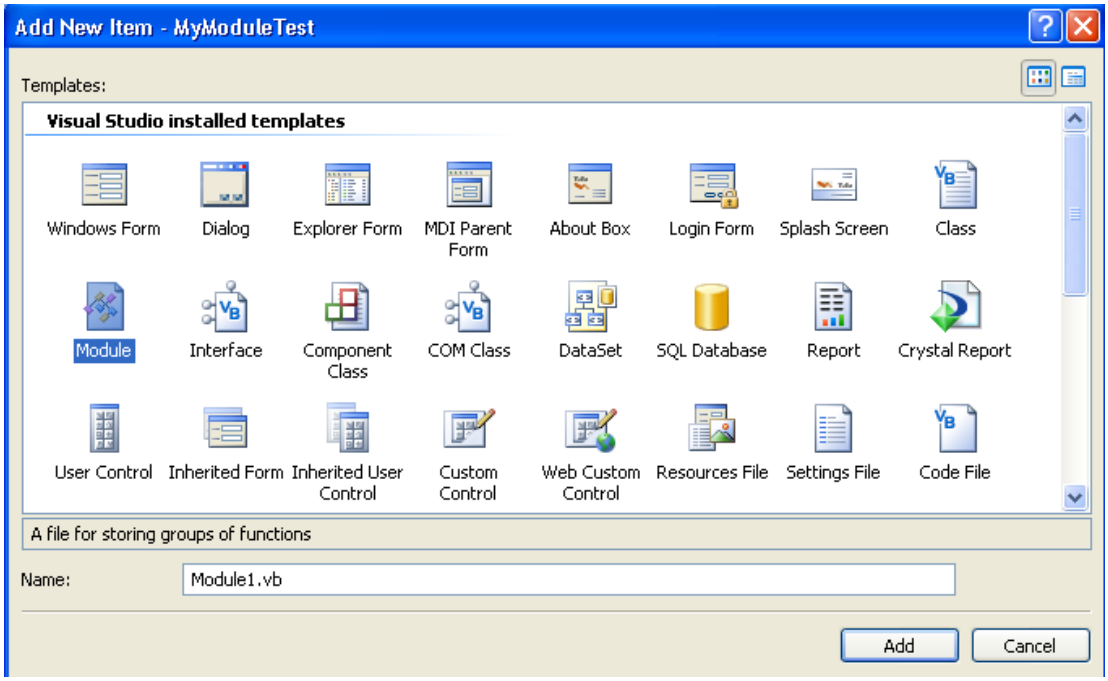
1.1. Tạo và lưu module chuẩn

Bây giờ chúng ta tạo một module với ví dụ *MymoduleTest* sau đây:

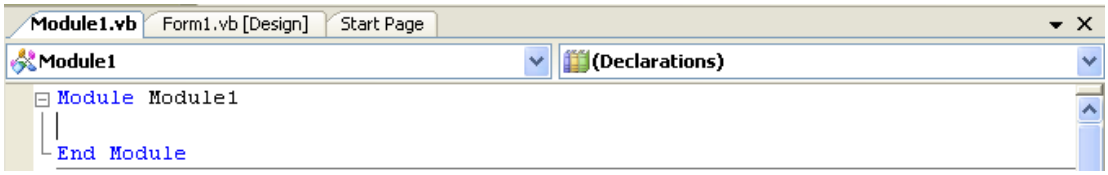
Bạn tạo mới một giải pháp và thêm vào một dự án cùng tên *MyModuleTest* như đã biết. Tại cửa sổ Solution Explorer bạn R-Click vào tên dự án và chọn Add | New Item… như hình:



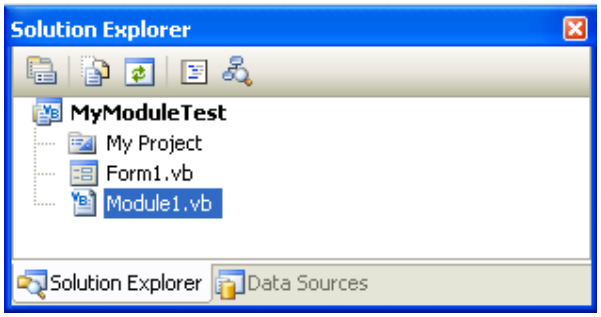
Bạn chọn mẫu *Module* và nhấn Open. Ở đây bạn có thể để tên mặc định là Module1.vb hay có thể nhập tên module luôn. Nếu để tên mặc định thì việc thay đổi tên sau này có thể dùng phương thức File | Save Module1 As như đã bàn:



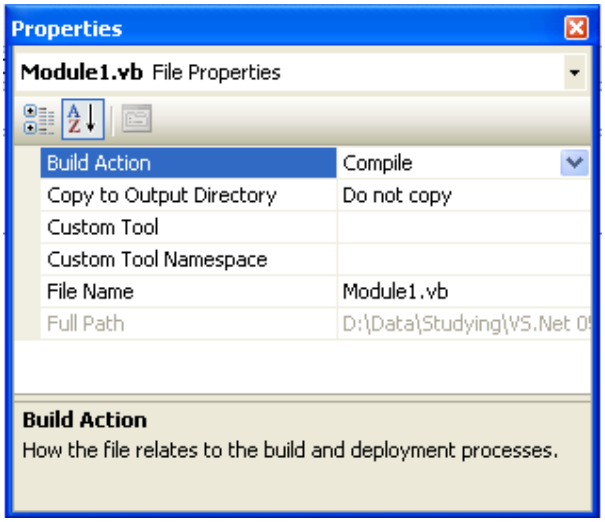
Khi nhấn Open thì một cửa sổ ở chế độ Code Editor hiện ra cho phép ta thao tác mã.



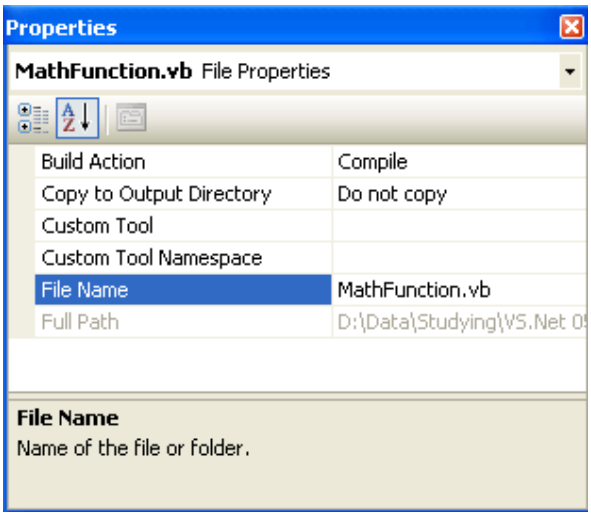
Bạn có thể xem liệt kê các thành phần của dự án ba gồm cả module1 ta vừa tạo bằng cách double click vào phần tiêu đề cửa sổ Solution Explorer:



Để cho cửa sổ này trở về vị trí cũ bạn có thể double click một lần nữa. Để xem thuộc tính của module, bạn R-Click vào module và chọn Properties:



Bạn thử thay đổi tên của module bằng thuộc tính *File Name* xem sao. Ở đây ta thay tên thành *MathFuction*:



Để xóa module, bạn r-clik vào nó và chọn Delete. Để tạm loại bỏ nó ra khỏi dự án bạn R-Click chọn Exclude From Projects (có thể chọn Project | Exclude From Project). Khi nào muốn thêm trở lại bạn chọn Add | Exist Item.

2. Làm việc với các biến Public (biến toàn cục)

Biến toàn cục là biến được khai báo với từ khóa *Public* ở trước. Biến này cho phép bạn triệu gọi xử lý ở bất cứ nơi nào trong chương trình. Ví dụ:

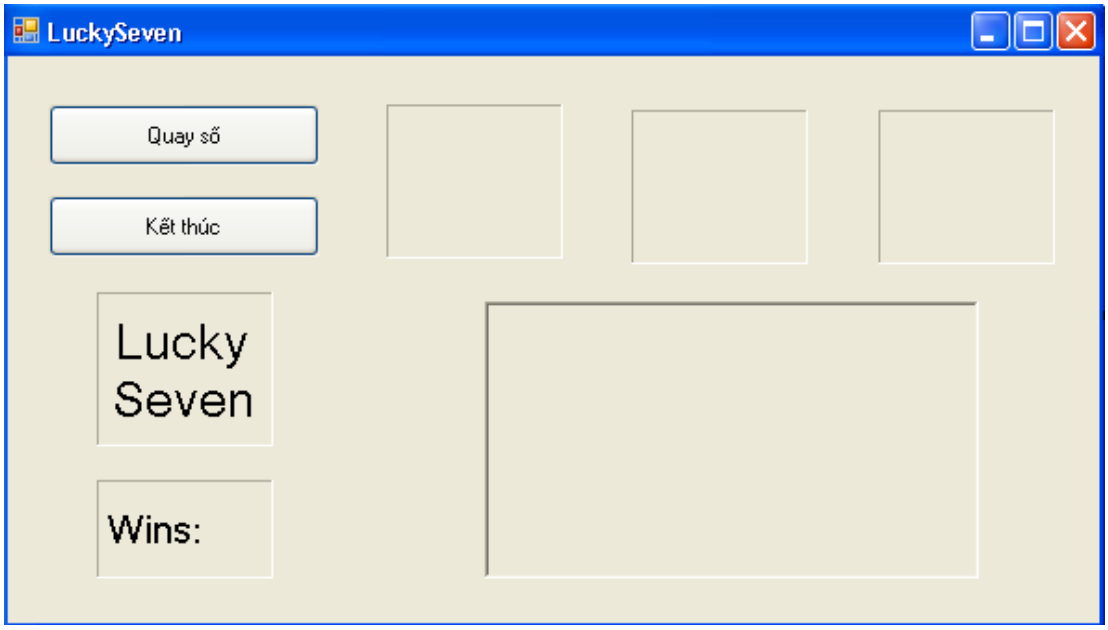
```
Public toancuc As Integer
```

Khai báo này khai báo một biến tên *toancuc* có kiểu dữ liệu là Integer.

Bây giờ ta quay lại chương trình *LuckySeven* đã làm trong các chương trước nhưng trong ví dụ này ta sử dụng một biến toàn cục có tên *solanthang* để lưu lại số lần người chơi chiến thắng và cho hiển thị nó lên trong một nhãn.

Ví dụ LUCKYSEVEN:

Bạn lưu lại dự án trên đây và đóng nó lại. Chọn tạo mới một giải pháp và thêm vào một dự án cùng tên *LuckySeven* như đã biết. Bạn thiết kế Form như hình:



Chương trình như đã biết bao gồm ba nhãn hiển thị 3 số ngẫu nhiên, hai nút cho phép click quay số và kêt thúc chương trình, một ô PictureBox hiển thị ảnh khi chiến thắng, một nhãn ghi tên chương trình *LuckySeven*. Bây giờ ta thiết kế thêm một nhãn nữa (Label5) hiển thị số lần chiến thắng của người chơi.

Bây giờ ta thêm vào một module – module *module1* và gõ vào trong đó một khai báo biến như sau:

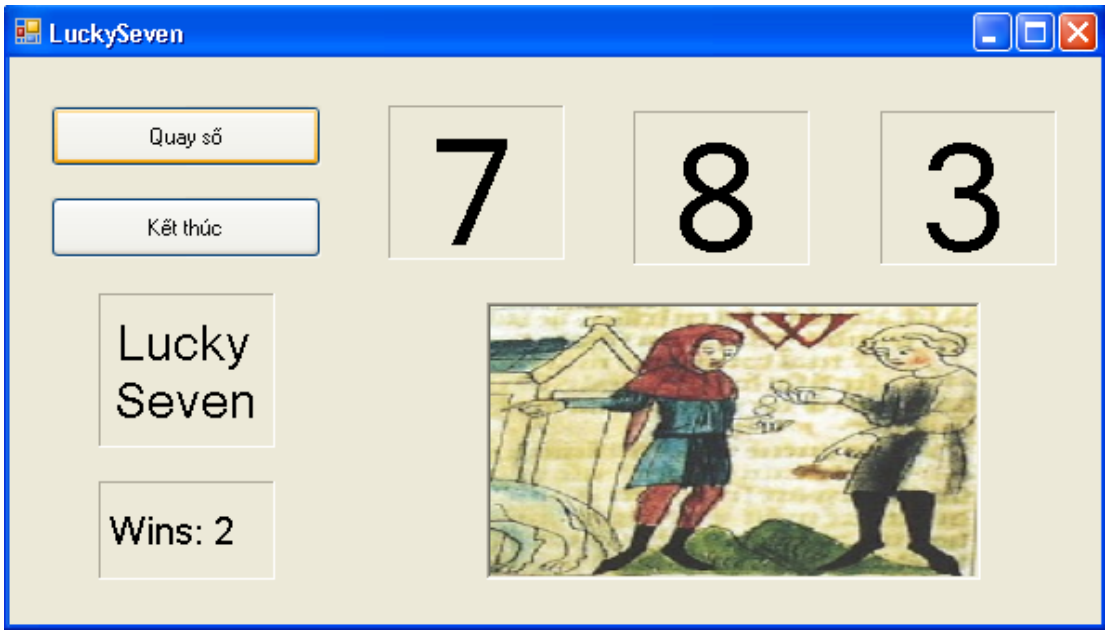
```
Public solanchienthang As Integer
```

Bây giờ chúng ta sẽ sử dụng biến này trong thủ tục Button1_Click như sau:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button1.Click
    PictureBox1.Visible = False
    Label1.Text = CStr(Int(Rnd() * 10))
    Label2.Text = CStr(Int(Rnd() * 10))
    Label3.Text = CStr(Int(Rnd() * 10))

    If (Label1.Text = "7") Or (Label2.Text = "7") _
    Or (Label3.Text = "7") Then
        PictureBox1.Visible = True
        Beep()

        solanchienthang += 1
        Label5.Text = "Wins: " & solanchienthang
    End If
End Sub
```

Bạn cũng dùng hàm `Randomize()` trong sự kiện Form_Load như ví dụ trước.

Như vậy chúng ta đã bước đầu dùng đến module và biến toàn cục. Bây giờ bạn hãy chạy chương trình để xem nó hoạt động như thế nào.



Biến phạm vi form và biến public trong module:

Biến phạm vi form là biến khai báo ở đầu chương trình dưới dòng khai báo form. Với chương trình này thì cả hai cách đều cho kết quả như nhau. Nhưng những biến khai báo ở mức độ form chỉ có thể sử dụng trong các hàm, các thủ tục ở form đó mà thôi.

3. Tạo thủ tục (Procedure)

Thủ tục nhằm để nhóm các phát biểu lệnh liên quan đến nhau thực thi một tác vụ nào đó. Có hai dạng thủ tục là, thủ tục dạng hàm *Function* và thủ tục dạng thuần túy *Procedure*. Thủ tục hàm thường để tính toán và trả về một kết quả nào đó cho nơi gọi hàm. Còn thủ tục thuần túy chỉ để thực hiện một tác vụ nào đó. Cả hai dạng đều có thể nhận đối số để làm nguồn tính toán, thực hiện thao tác xử lý.

Bạn có thể khai báo hàm và thủ tục trong form nhưng thường thì việc khai báo này được đặt trong module. Thủ tục giúp bạn không phải viết lại những đoạn mã trùng lặp nhiều lần chỉ để thực hiện một tác vụ nào đó.

Khi thủ tục đã hoàn thiện bạn có thể biên dịch thành file .dll lưu trong thư viện để sử dụng cho các dự án khác.

4. Xây dựng hàm (FUNCTION)

Hàm được khai báo bằng từ khóa *Function* và kết thúc bằng từ khóa *End Function*. Việc thực thi hay gọi hàm bằng cách dùng tên hàm cùng các đối số trong ngoặc đơn nếu có.

Khi hàm được khai báo trong module được mặc định là hàm toàn cục và có thể được triệu gọi từ bất cứ nơi nào của dự án.

4.2. Cú pháp khai báo hàm

Cú pháp của hàm:

```
Function FunctionName([argument]) As Type
    Function_statements()
    [Return value]
End Function
```

Trong đó:

- FunctionName: tên của hàm mà mình muốn tạo, dùng để gọi hàm sau này

- As Type: định nghĩa kiểu dl trả về của hàm sau khi tính toán xong

- Argument: danh sách đối số truyền cho hàm. Mặc định VB sẽ thêm từ khóa Byval tức truyền theo tham trị, tất cả những thay đổi trong hàm lên đối số không làm thay ảnh hưởng đến đối số truyền vào khi hàm chấm dứt.

- Function_Statement: các khối phát biểu cài đặt cho hàm.

- Return: cho phép trả lại kết quả sau cùng của hàm cho nơi gọi hàm.

Ví dụ:

```
Function TotalTax(ByVal Cost As Single) As Single
    Dim StateTax, CityTax As Single
    StateTax = Cost * 0.05
    CityTax = Cost * 0.015
    TotalTax = StateTax + CityTax
End Function
```

Hàm trên trả về giá trị thuế tổng *TotalTax* bằng câu lệnh ở phát biểu cuối cùng. Bạn có thể dùng phát biểu *Return* như ví dụ sau để trả về kết quả cho hàm:

```
Function TotalTax(ByVal Cost As Single) As Single
    Dim StateTax, CityTax As Single
    StateTax = Cost * 0.05
    CityTax = Cost * 0.015
    Return (StateTax + CityTax)
End Function
```

4.2. Gọi hàm

Việc gọi hàm theo cú pháp sau:

```
Label1.Text = TotalTax(500)
```

Phát biểu trên sẽ tính thuế dựa trên chi phí đầu vào là $500 và gán kết quả cho thuộc tính Text của nhãn label1.

4.3. Sử dụng hàm thực hiện tác vụ tính toán

Trong ví dụ sau ta sẽ quay trở lại ví dụ *LuckySeven* và tính tỷ lệ chiến thắng rồi gán vào thuộc tính của nhãn mới *Label6*.

Bạn thiết kế lại giao diện như sau:

Viết mã:

Trước hết ta khai báo thêm biến public trong module1 như sau:

```
Public solanchienthang As Short
Public Spins As Short
```

Sau đó bạn tạo thủ tục `HitRate` tính số phần trăm chiến thắng. Số phần trăm này được tính bằng tỉ số giữa số lần chiến thắng trên tổng số lần quay:

```
Function HitRate(ByVal Hits As Short, _
ByVal Tries As Short) As String
    Dim percent As Single
    percent = Hits / Tries
    Return Format(percent, "0.0%")
End Function
```

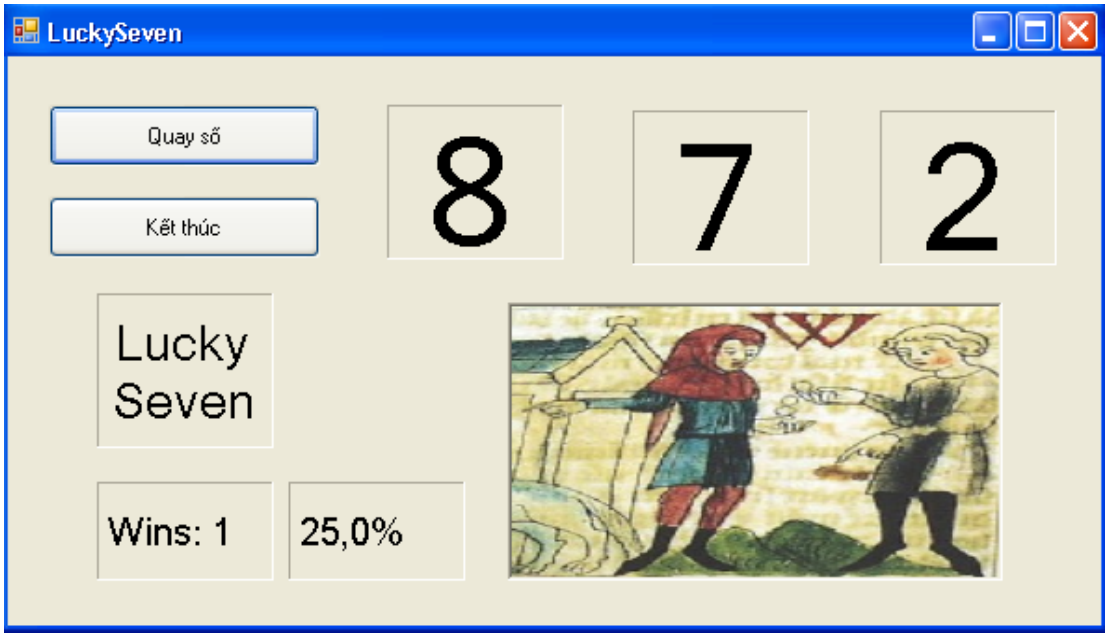
Bây giờ bạn trở lại thủ tục Button1_Click và nhập như sau:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button1.Click
    PictureBox1.Visible = False
    Label1.Text = CStr(Int(Rnd() * 10))
    Label2.Text = CStr(Int(Rnd() * 10))
    Label3.Text = CStr(Int(Rnd() * 10))
    Spins += 1

    If (Label1.Text = "7") Or (Label2.Text = "7") _
    Or (Label3.Text = "7") Then
        PictureBox1.Visible = True
        Beep()

        solanchienthang += 1
        Label5.Text = "Wins: " & solanchienthang
    End If
    Label6.Text = HitRate(solanchienthang, Spins)
End Sub
```

Giá trị của hàm HitRate đã được gán cho thuộc tính Text của nhãn Label6.

Chạy chương trình:

Bạn chọn Save All và chạy bằng cách ấn F5 và theo dõi kết quả.



5. Viết và xây dựng thủ tục SUB

Thủ tục này không trả về giá trị. Nó thiết kế để thực hiện một thao tác nào đó. Nó được khai báo bằng từ khóa *Sub…End Sub*. Tuy không trả về giá trị ở cuối thủ tục nhưng nó cũng có thể trả về giá trị thông qua đối số truyền vào thủ tục.

5.1. Khai báo thủ tục

Cú pháp:

```
Sub ProcedureName([Arguments])
    'Procedure Statement
End Sub
```

Trong đó:

- ProcedureName: tên của thủ tục.
- Arguments: các đối số.
- Procedure Statements: Các phát biểu cài đặt cho phần nội dung của thủ tục.

Ví dụ:

Bạn tạo ví dụ *BestWishesForBirthday* để minh họa cách tạo và dùng hàm Sub. Thiết kế form1 chỉ có một nút "End" và nhập End vào thủ tục Button1_Click.

Bây giờ bạn thêm một module vào chương trình và khai báo trong đó một thủ tục như sau:

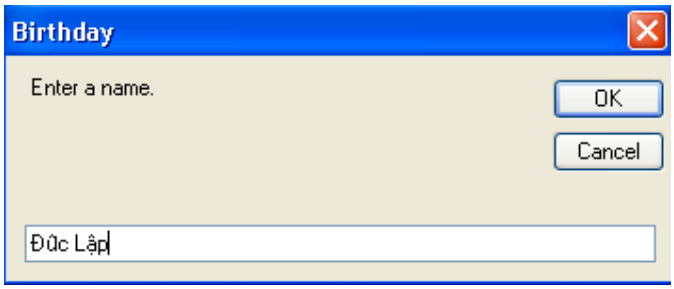
```
Sub BirthdayGreeting(ByVal Person As String)
    Dim msg As String
    If Person <> "" Then
        msg = "Happy Birthday " & Person & "!"
    Else
        msg = "Name no specified"
    End If
    MsgBox(msg, , "BestWhished")
End Sub
```

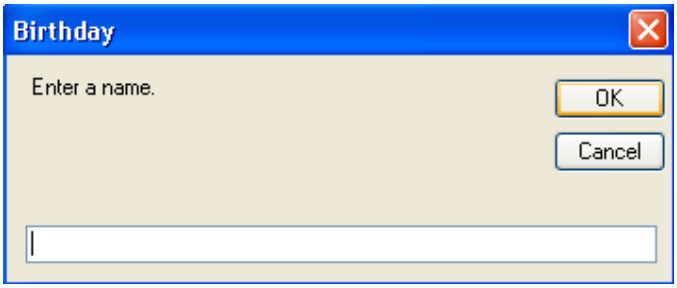
Thủ tục này cho phép hiện một lời chào với người có tên là đối số truyền vào cho biến *Person*.

Bây giờ trở về cửa sổ soạn mã cho form1 và tạo sự kiện Form1_Load nhập đoạn mã sau:

```
Private Sub Form1_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
    Dim name As String
    Do
        name = InputBox("Enter a name.", "Birthday")
        BirthdayGreeting(name)
    Loop Until name = ""
End Sub
```

Đoạn mã này sẽ hiện một hộp thoại để người dùng nhập tên của họ cho chương trình gọi thủ tục trên chúc sinh nhật người đó. Khi không nhập vào thì chương trình hiện form1 với nút "End" cho phép kết thúc chương trình:





5.3. Sử dụng thủ tục SUB quản lý nhập liệu của người dùng

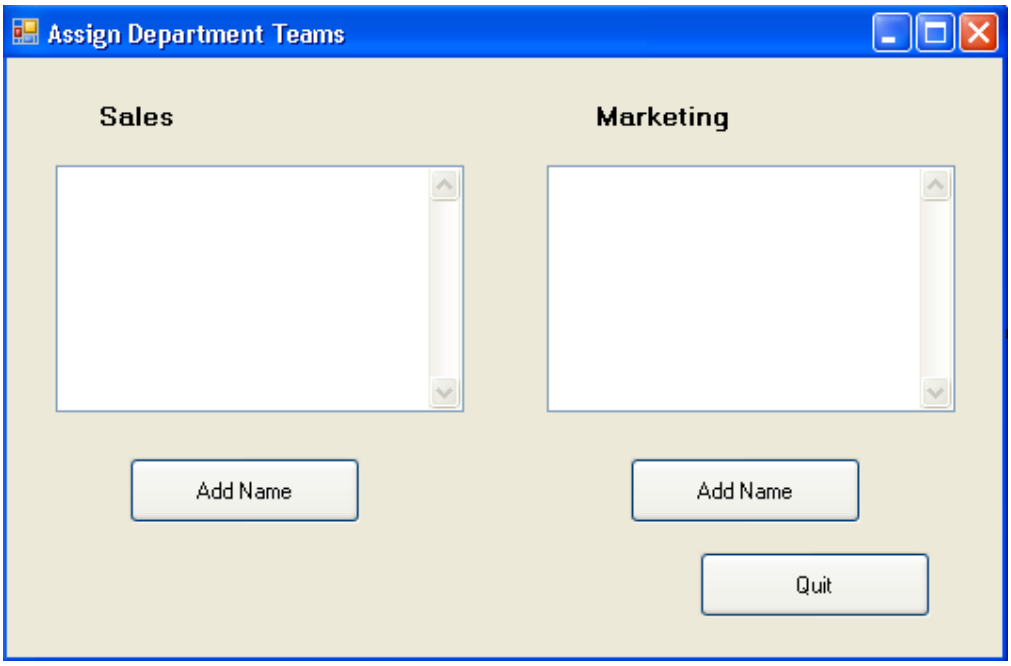
Bài tập sau đây chúng ta sẽ tạo một thủ tục trong một module cho phép người dùng nhập vào tên và hiển thị tên đó trong một ô textbox.

Tìm hiểu chương trình:

Chương trình mô phỏng một form cập nhật tên nhân viên mới vào hai vị trí là *Sales* và *Marketing*. Khi người dùng click vào nút *addname* bên dưới ô textbox *sales* để thêm vào một nhân viên sales mới thì một ô nhập liệu Inputbox hiện lên. Tên mới của nhân viên này sẽ hiển thị vào ô textbox tương ứng. Tương tự như trên với việc nhập nhân viên vào vị trí marketing.

Thiết kế giao diện:

Giao diện chương trình như hình:



Bạn tạo một giải pháp mới có tên là MyTextboxSub và thêm vào một dự án cùng tên rồi thiết kế form như hình. Trong đó các thuộc tính name của các điều khiển như sau:

- Ô textbox dưới nhãn Sales là *txtSales*
- Ô textbox dưới nhãn Marketing là *txtMkt*
- Nút Add Name dưới ô txtSales là *btnSaleAdd*
- Nút Add Name dưới ô txtMkt là *btnMktAdd*

Các thuộc tính khác của hai textbox như tạo chế độ MultiLine và thanh cuộn dọc thì bạn đã biết tạo trong ví dụ trước.

Viết mã :

Trước hết ta tạo một thủ tục để người dùng nhập vào tên của nhân viên mới khi click nút Add Name thêm tên nhân viên – thủ tục có tên AddName.

Bạn add vào dự án một module1.vb và tạo ra hàm AddName như sau :

```
Module Module1
    Sub AddName(ByVal vitri As String, _
    ByRef tenNVmoi As String)
        Dim Prompt, ten, dinhdang As String
        Prompt = "Enter a " & vitri & "employee."
        ten = InputBox(Prompt, "InputBox")
        dinhdang = Chr(13) + Chr(10)
        tenNVmoi = ten & dinhdang
    End Sub
End Module
```

Trong đó, đối số *vitri* là "Sales" hay "Marketing"; *tenNVmoi* là tên của nhân viên khi người dùng nhập vào bằng inputBox;

Bạn tạo thủ tục `btnsaleAdd_Click` bằng cách double click vào nút btnSale và nhập đoạn mã như sau:

```
Dim salesTeam As String
AddName("Sales", salesTeam)
txtSale.Text = txtSale.Text & salesTeam
```

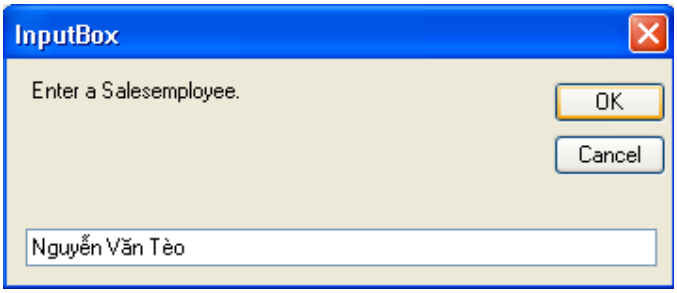
Tiếp tục tạo thủ tục `btnMktAdd_Click` và nhập mã như sau:

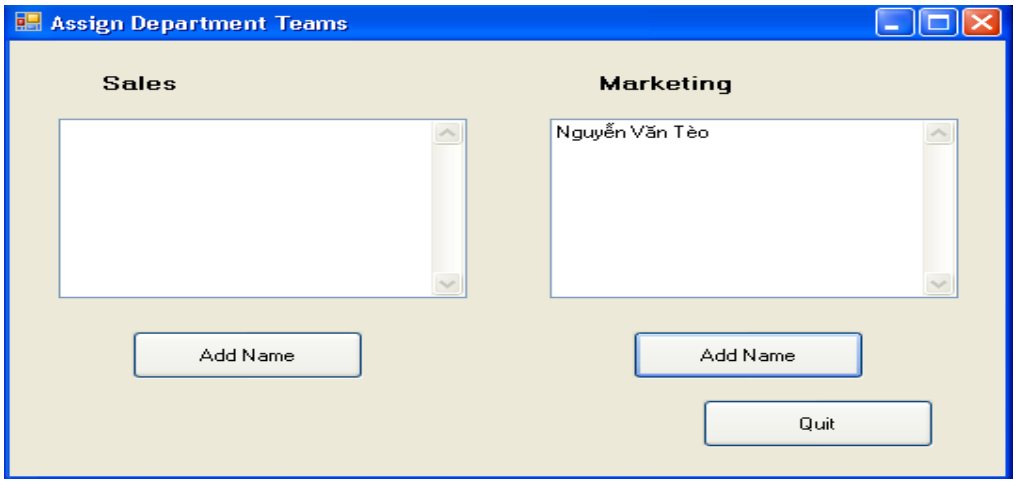
```
Dim MktTeam As String
AddName("Marketing", MktTeam)
txtMkt.Text = txtMkt.Text & MktTeam
```

Ngoài ra bạn cũng tạo thủ tục click của nút btnQuit với phát biểu End để kết thúc chương trình.

Chạy chương trình:

Bây giờ bạn chạy chương trình và thêm vào vài nhân viên mới ở cả hai vị trí.





6. Truyền đối số theo tham trị và tham biến

Trong phần bàn về thủ tục, ta thấy đối số có thể truyền theo tham trị (Byval) hay tham chiếu (Byref). Mặc định khi không khai báo thì nó truyền theo tham trị.

Để truyền theo tham trị ta khai báo đối số trong thủ tục bằng từ khóa *Byval*. Khi đó bất cứ thay đổi nào trong thủ tục cũng không làm thay đổi giá trị của biến sau khi thủ tục chấm dứt.

Ngược lại, để truyền theo tham chiếu, ta khai báo đối số bằng từ khóa *Byref*. Khi đó những thay đổi trong hàm hay thủ tục sẽ làm thay đổi giá trị của biến.

Khi nào bạn chưa biết dùng *Byval* hay *Byref* thì bạn nên dùng *Byval*.

7. Tổng kết chương 10

Chương này đã kết thúc. Bạn hãy làm bảng tổng kết những gì đã học.

Chương 11:

Sử dụng mảng và tập hợp (Collection)

--------oOo--------

Nội dung thảo luận:

- Tổ chức thông tin dữ liệu có chiều dài cố định và mảng động
- Dự trữ mảng dữ liệu khi định nghĩa lại kích thước mảng
- Xử lý tập hợp các đối tượng điều khiển trên form
- Sử dụng vòng lặp *For Each…Next* để duyệt qua danh sách tập hợp các đối tượng
- Tạo tập hợp của riêng bạn để quản lý dữ liệu chuỗi

Khi chương trình trở nên lớn và lượng thông tin và dữ liệu lớn, bạn không thể chỉ dùng biến mà còn cần đến một công cụ lưu trữ, xử lý tốt hơn đó là mảng. Mảng chứa danh sách tuần tự các phần tử cùng định dạng.

Chúng ta cũng có thể nhóm các đối tượng vào thành tập hợp để dễ quản lý, dùng vòng lặp For each…Next để duyệt qua các phần tử của tập hợp.

Các tính năng mới trong VB.NET:

- Mảng có chỉ số cơ sở là 0
- Hàm *Lbound* luôn trả về 0 là giới hạn thấp nhất của mảng. Hàm *Ubound* trả về giới hạn trên của mảng. Số phần tử của mảng là *Ubound()-1*
- VB.NET dùng lớp *System.Collections* để xử lý tập hợp
- Bạn có thể chứa các điều khiển trong mảng nếu mảng của bạn khai báo kiểu Object

1. Làm việc với mảng các biến

Mảng giúp quản lý các dữ liệu lớn hết sức dễ dàng. Việc truy cập các phần tử của mảng thông qua chỉ số.

1.1. Tạo mảng

Việc khai báo mảng tương tự như khai báo biến. Việc khai báo thường chứa các thông tin như:

- Tên mảng: Tên đại diện cho mảng, việc truy cập một phần tử mảng gồm tên mảng và chỉ số mảng.
- Kiểu dữ liệu: Tất cả các phần tử trong mảng phải có cùng kiểu.
- Kích thước mảng: Là số chiều của mảng.
- Số phần tử của mảng: Số phần tử tối đa của mảng

1.2. Khai báo mảng cố định

Cú pháp chung khai báo mảng có kích thước là:

```
Dim ArrayName(Dim1Index, Dim2Index) As DataType
```

Trong đó:

- ArrayName: tên mảng
- Dim1Index và Dim2Index: là hai chiều của mảng
- Datatype: kiểu dữ liệu của mảng. Khi chưa xác định kiểu cụ thể, có thể dùng kiểu Object.

Ví dụ:

Khai báo `Dim Employee(4) As String` khai báo mảng một chiều chứa 5 phần tử có tên là Employee có kiểu String.

Bạn cũng có thể khai báo mảng một cách toàn cục trong module bằng từ khóa Public như sau: `Public Employee(4) As String.`

Mảng một chiều có dạng:

Employee

0
1
2
3
4

Để khai báo mảng hai chiều mang tên ScoreBoard bạn có thể khai báo như sau:

```
Dim ScoreBoard(1, 4) As Short
```

Mảng này gồm 2*5 = 10 phần tử tương ứng với 10 ô vuông gồm hai dòng và 5 cột đánh số từ 0.

1.3. Làm việc với các phần tử trong mảng

Sau khi khai báo, bạn có thể sử dụng mảng. Việc truy cập vào một phần tử của mảng nhờ tên mảng và chỉ số của mảng đặt trong ngoặc đơn, chỉ số là số nguyên, là biến nguyên hay biểu thức có giá trị. Để duyệt qua tất cả các phần tử trong mảng, dùng vòng lặp For…Next.

Ví dụ:

```
employee(3) = "Thanh Van"
```

Phát biểu trên gán cho phần tử có chỉ số thứ 3 (tại ô thứ 4) tên là “Thanh Van”.

```
ScoreBoard(0, 2) = 12
```

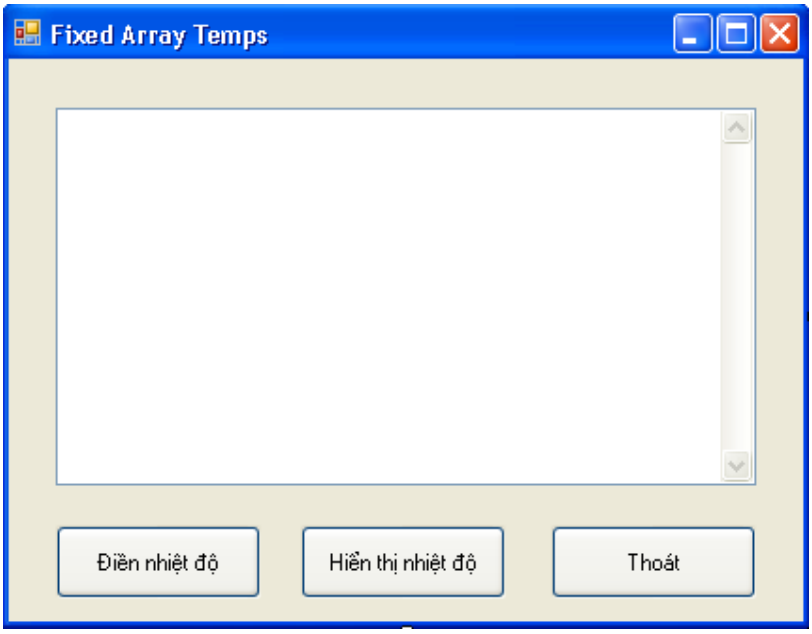
Phát biểu trên gán cho phần tử ở dòng 0, cột 2 giá trị là 12.

1.4. Tạo và sử dụng mảng có kích thước cố định

Bây giờ ta tạo ví dụ *MyFixedArray* sử dụng mảng một chiều có tên *nhietdo* để ghi lại giá trị nhiệt độ cao thấp hàng ngày trong tuần. Mảng này được khai báo ở đầu form và được gán giá trị bằng hàm *InputBox* nhờ vòng lặp For…Next. Toàn bộ nội dung của mảng sau đó lại được hiển thị lại vào một textbox cũng nhờ vòng lặp For…Next.

Thiết kế giao diện:

Tạo mới một giải pháp và thêm vào một dự án có cùng tên là MyFixedArray. Thiết kế giao diện như hình:



Trong đó: nút button1 có text là "Điền nhiệt độ", button2 là "Hiển thị nhiệt độ", button3 là "Thoát".

Viết mã:

Trước hết ta khai báo mảng *nhietdo* ở ngay dưới dòng `Public Class Form1` như sau:

```
Dim nhietdo(6) As Single
```

Khai báo như thế này nghĩa là tất cả các thủ tục, các hàm đều có thể sử dụng mảng này.

Tiếp theo ta tạo ra sự kiện nhập vào các giá trị nhiệt độ trong tuần bằng cách tạo thủ tục `Button1_Click` và nhập mã như sau:

```
Private Sub Button1_Click(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles Button1.Click
    Dim Prompt, tieude As String
    Dim i As Short
    Prompt = "Điền vào nhiệt độ của ngày."
    For i = 0 To UBound(nhietdo)
        tieude = "Ngày " & (i + 1)
        nhietdo(i) = CInt(InputBox(Prompt, tieude))
    Next
End Sub
```

Trong đó, hàm *Ubound(nhietdo)* là hàm lấy về chỉ số trên của mảng *nhietdo*, trong trường hợp này là 6.

Sau đó ta cho hiển thị các giá trị nhiệt độ trong bảy ngày trong tuần cũng như giá trị nhiệt độ trung bình bằng thủ tục `Button2_Click` khi người dùng click vào nút "Hiển thị nhiệt độ" như sau:

```
Private Sub Button2_Click(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles Button2.Click
    Dim ketqua As String
    Dim i As Short
    Dim tong As Single = 0
    ketqua = "Nhiệt độ của tuần: " & vbCrLf & vbCrLf

    For i = 0 To UBound(nhietdo)
        ketqua = ketqua & "Ngày " & (i + 1) & _
        vbTab & nhietdo(i) & vbCrLf
        tong = tong + nhietdo(i)
    Next

    ketqua = ketqua & vbCrLf & _
    "Nhiệt độ trung bình: " & _
    Format(tong / 7, "0.0")

    TextBox1.Text = ketqua
End Sub
```

Thủ tục này lại sử dụng vòng lặp For…Next để duyệt lại các phần tử trong mảng sau khi đã được gán giá trị ở thủ tục button1_Click. Biến *ketqua* được dùng để làm chuỗi kết xuất gộp các giá trị phần tử mảng. Sau mỗi lần gộp ta sử dụng hằng số *vbCrLf* để khiến dấu ngắt dòng và dấu về đầu dòng (tương đương với hai hàm *Chr(13) và Chr(10)*). Hằng *vbTab* để phân cách giữa phần ghi ngày và ghi nhiệt độ.

Bạn tạo thủ thục Button3_Click và nhập phát biểu End để kết thúc chương trình.

Chạy chương trình:

Bạn chạy chương trình và nhập đủ giá trị nhiệt độ 7 ngày rồi cho hiển thị giá trị đó lên xem sao.

1.5. Tạo mảng động

Việc dùng mảng là rất thuận tiện. Tuy nhiên khi bạn chưa biết chính xác số phần tử của mảng là bao nhiêu thì sao? Ví dụ khi bạn muốn để người dùng nhập vào bao nhiêu nhiệt độ tùy thích, nhập càng nhiều thì độ chính xác càng cao.

VB giải quyết việc này bằng mảng động. Kích thước mảng động chỉ được chỉ định khi chương trình thực thi chứ không định trong lúc viết mã. Việc khai báo trước kích thước mảng là không cần thiết nhưng cũng cần dành chỗ trước cho mảng đó.

Các bước tạo mảng động:

- Chỉ định tên và kiểu cho mảng khi thiết kế form, ví dụ `Dim nhietdo() As Single`
- Thêm mã xác định kích thước mảng khi chương trình thực thi. Ví dụ khi chương trình chạy bạn hỏi xem người dùng muốn nhập bao nhiêu ngày, ví dụ:

```
Dim songay As Integer
songay = InputBox("Ban muon nhap bao nhieu ngay?", "Tao mang dong")
```

- Dùng biến songay để định lại kích thước mảng (trừ đi 1 vì mảng tính từ 0). Ví dụ

```
If songay > 0 Then ReDim nhietdo(songay - 1)
```

- Tiếp theo ta dùng hàm Ubound(nhietdo) để xác định số phần tử của mảng.

Bây giờ chúng ta sẽ làm lại ví dụ trên sử dụng mảng động:

- Trước hết, bạn khai báo lại mảng động và khai báo biến songay chứa số ngày người dùng muốn nhập bằng đoạn mã ngay dưới dòng khai báo lớp form1:

```
Dim nhietdo() As Single
Dim songay As Integer
```

- Sau đó sửa lại mã của thủ tục Button1_Click như sau:

```
Dim Prompt, tieude As String
Dim i As Short
Prompt = "Điền vào nhiệt độ của ngày."

'Nhap so ngay muon ghi nhiet do
songay = InputBox("Ban muon nhap bao nhieu ngay?", "Tao mang
dong")
If songay > 0 Then ReDim nhietdo(songay - 1)

For i = 0 To UBound(nhietdo)
    tieude = "Ngày " & (i + 1)
    nhietdo(i) = CInt(InputBox(Prompt, tieude))
Next
```

- Tiếp theo thay số 7 trong thủ tục Button2_Click bằng biến *songay*:

```
ketqua = ketqua & vbCrLf & _
"Nhiệt độ trung bình: " & _
Format(tong / songay, "0.0")
```

- Bạn có thể dùng phát biểu *Try…Catch* để bắt lỗi nếu người dùng nhập vào một số nhỏ hơn 0.
- Chạy lại chương trình và kết quả rõ ràng linh động hơn.

2. Dự trữ sẵn mảng bằng phát biểu REDIM

Trong ví dụ trên, chúng ta đã định lại kích thước bằng từ khóa REDIM. Khi định lại kích thước như thế thì dữ liệu cũ của mảng sẽ mất hết. Các phần tử sẽ có giá trị là 0 hay NULL. Để giữ nguyên giá trị các phần tử cũ, bạn có thể dùng từ khóa *Preserve* đi kèm phát biểu *Redim*. Ví dụ:

```
If songay > 0 Then ReDim Preserve nhietdo(songay - 1)
```

Với phát biểu này thì các phần tử mới nhận giá trị là 0 hay NULL.

Ví dụ ta định nghĩa mảng động: Dim mangdong() As string

Sau đó tái định nghĩa kích thước mảng là 200:

Redim mangdong(200) As String

mangdong(200) = 200

Giờ ta mở rộng mảng này với câu lệnh Preserve:

Redim Preserve mangdong(300)

Khi đó giá trị của phần tử mangdong(200) vẫn là 200. Việc tái định nghĩa mảng đa chiều cũng tương tự.

3. Làm việc với tập hợp đối tượng Collection

Tiếp theo chúng ta sẽ làm quen với tập hợp. Tập hợp cũng tương tự như mảng nhưng nó dùng để xử lý các phần tử kiểu đối tượng, có khối lượng và kích thước lớn hơn. Trong VB, các điều khiển trong form cũng được nhóm thành tập hợp gọi là Controls Collection – tập hợp các điều khiển. Mỗi thành phần điều khiển khi bạn cho vào form đều được đưa vào trong tập hợp điều khiển. Mỗi tập hợp trong chương trình đều có một tên riêng để ta tham chiếu đến nó. Bạn có thể dùng Object Browser để xem các tập hợp đối tượng của hệ thống, điều này sẽ được hướng dẫn cụ thể trong chương 13.

3.1. Tham chiếu đến đối tượng trong tập hợp

Bạn có thể tham chiếu đến một đối tượng trong tập hợp thông qua chỉ số như ở mảng. Trong VB thì các đối tượng được đưa vào tập hợp theo thứ tự đảo ngược, tức là phần tử đưa vào sớm nhất có chỉ số cao nhất và ngược lại, phần tử đưa vào sau cùng có chỉ số nhỏ nhất là 0. Ví dụ, để truy cập thuộc tính Text của đối tượng phần tử sau cùng trên form bạn sử dụng chỉ số thứ tự là 0 như sau:

```
Controls(0).Text="phần tử cuối cùng"
```

Muốn truy cập các phần tử khác trên form, bạn sử dụng chỉ số khác như 0, 1,…Ví dụ vòng lặp for…next sau sẽ in ra tiêu đề của 4 đối tượng trên form:

```
For i = 0 To 3
    ketqua &= Controls(i).Text & vbCrLf
Next
```

Bạn có thể dùng vòng lặp For…Next như trên, tuy nhiên hiệu quả và dễ hiểu nhất bạn nên dùng vòng lặp For Each…Next.

3.2. Sử dụng vòng lặp For Each…Next

Mặc dù bạn có thể tham chiếu đến các phần tử trong tập hợp riêng lẻ nhưng hầu như các thao tác trên tập hợp đều duyệt từ đầu đến cuối tập hợp bằng vòng lặp For Each…Next.

Các thao tác thường gặp là: di chuyển các đối tượng, sắp xếp, đổi tên hay thay đổi lại kích thước của toàn bộ tập hợp. Cú pháp của vòng lặp như sau:

```
Dim CtrlVar As Control
...
For Each CtrlVar in controls
    Khối lệnh xử lý đối tượng trong tập hợp
Next
```

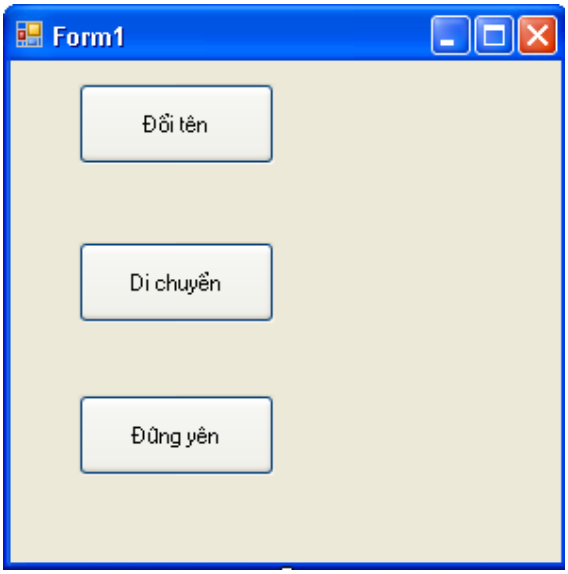
Ví dụ sau ta sử dụng tập hợp đối tượng Controls để xử lý các đối tượng trên form, ví dụ *MyControlsCollection*.

Tìm hiểu chương trình:

Chương trình có một form chính. Trên form chính có ba nút nhấn. Khi người dùng nhấn vào nút thứ nhất có text "Đổi tên" thì nút nhấn này đổi text thành "Đã đổi tên", nút nhấn thứ hai sẽ di chuyển sang trái 40 đơn vị và đổi text từ "Di chuyển" thành "Đã di chuyển", nút nhấn thứ ba có text "Đứng yên" khi người dùng click vào đây thì nút này đổi text thành "Đây đứng yên" còn các nút khác di chuyển sang trái 40.

Thiết kế form:

Form có giao diện như sau:



Các bạn tạo mới một giải pháp và thêm vào một dự án có cùng tên *MyControlsCollection* rồi thiết kế form như hình.

Viết mã:

Khai báo một biến *Ctrl* ngay dưới dòng khai báo lớp form1 như sau:

```
Dim ctrl As Control
```

Tạo thủ tục `btndoiten_Click` và nhập mã như sau (ở đây nút nhấn "Đổi tên" có thuộc tính Name là "btndoiten"):

```
Private Sub btndoiten_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btndoiten.Click
    For Each ctrl In Controls
```

```
            If ctrl.Name = "btndoiten" Then
                ctrl.Text = "Đã đổi tên"
            End If
        Next
    End Sub
```

Trong đoạn mã trên ta sử dụng vòng lặp for each … next để duyệt qua các phần tử trong tập hợp các điều khiển controls của form. Ta cũng dùng phát biểu if để lọc ra những điều khiển có thuộc tính Name là "btndoiten" để thao tác.

Tương tự bạn tạo thủ tục `btnmove_Click` và nhập mã như sau (nút nhấn có text "Di chuyển" có thuộc tính Name là "btnmove"):

```
Private Sub btnmove_Click(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles btnmove.Click
    For Each ctrl In Controls
        If ctrl.Name = "btnmove" Then
            ctrl.Text = "Đã di chuyển"
            ctrl.Left = ctrl.Left + 40
        End If
    Next
End Sub
```

Thủ tục `btnstatic_Click` (btnstatic là tên của nút "Đứng yên"):

```
Private Sub btnstatic_Click(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles btnstatic.Click
    For Each ctrl In Controls
        If ctrl.Name = "btnstatic" Then
            ctrl.Text = "Đây đứng yên"
        Else
            ctrl.Left = ctrl.Left + 40
        End If
    Next
End Sub
```

Chạy chương trình:

Công việc của bạn giờ là ấn F5 để chạy chương trình xem thành quả chúng ta đã làm.

4. Tự tạo tập hợp của người dùng

VB cho phép tạo một tập hợp của người dùng để lưu trữ từ các thành phần điều khiển như nút nhấn, nhãn,… đến các kiểu đơn giản như chuỗi, số nguyên và các kiểu cơ sở khác tương tự như mảng.

Cú pháp khai báo:

```
Dim MyCollection As New Collection()
```

Trong đó MyCollection là tên của tập hợp, phương thức New khởi tạo vùng nhớ cho tập hợp. Sau khi tạo tập hợp bạn có thể dùng phương thức *add* để thêm phần tử dl vào tập hợp. Để duyệt tập hợp bạn dùng vòng lặp *for each next* như đã biết.

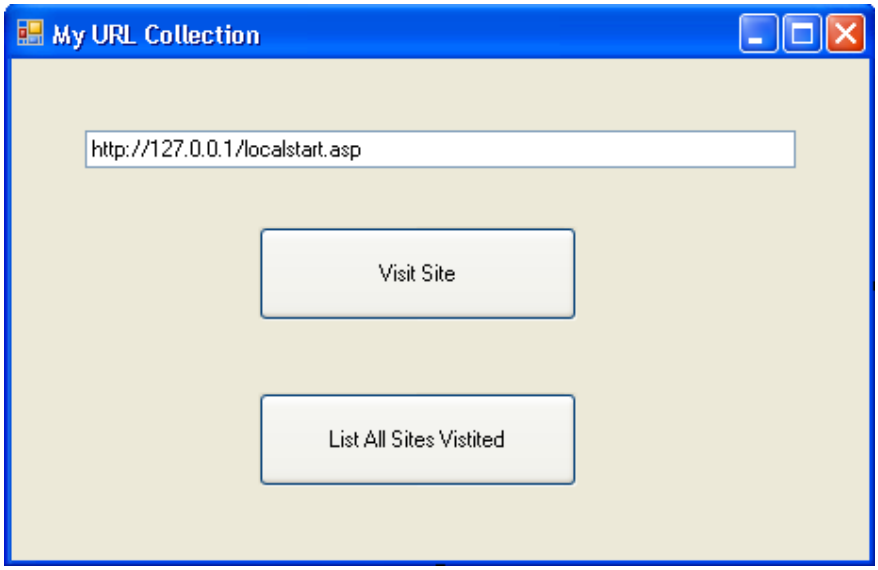
Trong ví dụ sau đây chúng ta sẽ làm quen với cách sử dụng tập hợp giữa danh sách các địa chỉ Internet được người dùng sử dụng gần đây nhất. Để gọi trình duyệt IE, bạn dùng phương thức *System.Dyagnostics.Process.Start*.

Tìm hiểu chương trình:

Chương trình có một form chính gồm một textbox và hai nút nhấn. Ô textbox cho phép nhập vào địa chỉ website, nút nhấn thứ nhất để người dùng click vào duyệt trang web có địa chỉ ghi trong ô textbox, ô nhấn thứ hai để liệt kê tất cả những trang web người dùng đã duyệt bằng cách nhập địa chỉ URL vào trong ô textbox.

Thiết kế giao diện:

Bạn tạo một giải pháp và thêm một dự án mới cùng tên là MyURLCollection và thiết kế giao diện như sau:



Viết mã:

Trước hết ta tạo một tập hợp để chứa các tên địa chỉ web mà người dùng đã thăm bằng phát biểu sau đặt ngay dưới dòng khai báo lớp form1:

```
Dim URLsVisited As New Collection()
```

Với phát biểu này thì tất cả các thủ tục trong form đều có thể truy cập đến tập hợp này.

Tiếp theo tạo thủ tục Button1_Click và nhập mã như sau:

```
Private Sub Button1_Click(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles Button1.Click
    URLsVisited.Add(TextBox1.Text)
    System.Diagnostics.Process.Start(TextBox1.Text)
End Sub
```

Phương thức Add(TextBox1.Text) sẽ thêm vào tập hợp một phần tử có nội dung là thuộc tính text của ô textbox1.

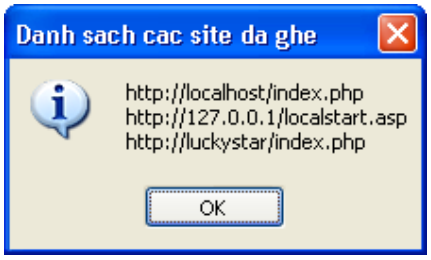
Sau đó ta cũng tạo thủ tục Button2_Click cho phép người dùng liệt kê tất cả các trang web đã duyệt bởi người dùng (các phần tử trong tập hợp):

```
Private Sub Button2_Click(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles Button2.Click
    Dim URLname, AllURLVisited As String
```

```
        For Each URLname In URLsVisited
            AllURLVisited = AllURLVisited & URLname & vbCrLf
        Next
        MsgBox(AllURLVisited, MsgBoxStyle.Information, _
        "Websites Visited")
    End Sub
```

Chạy chương trình:

Ấn F5 chạy chương trình. Bạn có thể nhập các địa chỉ khác địa chỉ của localhost nếu bạn có kết nối Internet.



5. Tập hợp COLLECTION trong ứng dụng VBA

Bạn có thể viết các ứng dụng macro cho ứng dụng văn phòng bằng ngôn ngữ VBA (Visual Basic for Application). Trong các ứng dụng này thì tập hợp đóng vai trò rất quan trọng. Có thể tham khảo thêm trong các quyển sách của trang Appress.Com.

6. Tổng kết chương

Làm bảng tổng kết chương những gì đã học. Đồng thời dùng tập hợp thay thế cho bài tập về mảng trong chương trước.

Chương 12:

Khám phá cách xử lý file TEXT và chuỗi

--------oOo--------

Nội dung thảo luận:

- Hiển thị nội dung file text bằng đối tượng TextBox
- Lưu các thông tin trong file text
- Sử dụng kỹ thuật xử lý chuỗi để sắp xếp và mã hóa file Text

Trong chương này chúng ta học cách xử lý file text đơn giản với các thao tác như mở file, hiển thị nội dung, lưu và các thao tác khác như xử lý chuỗi trong chương trình thông qua lớp chuỗi String. Bạn có thể sắp xếp, ghép nối mã hóa hiển thị từng từ, từng dòng và toàn bộ nội dung văn bản trong file text.

Chú ý:

- Đối tượng *FileSystem* cung cấp các hàm như *FileOpen, LineInput, PrintLine, FileClose* để thao tác với tập tin. Đối tượng này nằm trong không gian tên *Microsoft.VisualBasic*.
- Ngoài ra một số hàm trong không gian System.IO cũng có thể dùng bổ sung.

1. Hiển thị nội dung file Text bằng đối tượng TextBox

Cách đơn giản nhất để hiển thị một file text là dùng điều khiển textbox. Để nạp nội dung file text vào textbox ta dùng 4 hàm sau: *FileOpen* – Mở file để dọc hay ghi, *LineInput* – Đọc một dòng văn bản từ file, *EOF* – Kiểm tra xem con trỏ đã đến cuối file chưa, *FileClose* – Đóng file.

1.1. Mở file Text để đọc nội dung

Bạn có thể cho phép người dùng mở file text bằng cách hiển thị hộp thoại *OpenfileDialog*. Sau khi người dùng đã chọn file, hộp thoại sẽ trả về đường dẫn file đầy đủ thông qua thuộc tính filename.

1.2. Hàm FileOpen

Sau khi đã có tên file, bạn có thể dùng hàm *FileOpen* mở file để đọc hay ghi. Cú pháp hàm FileOpen như sau:

```
FileOpen(filenumber, pathname, mode)
```

Trong đó:

- filenumber: số nguyên từ 1 đến 255
- pathname: đường dẫn hợp lệ trỏ đến file cần mở

- mode: từ khóa cho biết chế độ mở (ví dụ *OpenMode.Input* là mở file để đọc và *OpenMode.Output* là mở file để ghi)

Số nguyên *filenumber* dùng để kết hợp với file khi nó được mở cho mục đích đọc ghi. Bạn dùng nó để tham chiếu đến file trong quá trình xử lý. Lưu ý là các số *filenumber* trong hàm *FileOpen, LineInput, FileClose* và *EOF* phải trùng nhau thì khi mở file mới không gây ra lỗi.

Ví dụ:

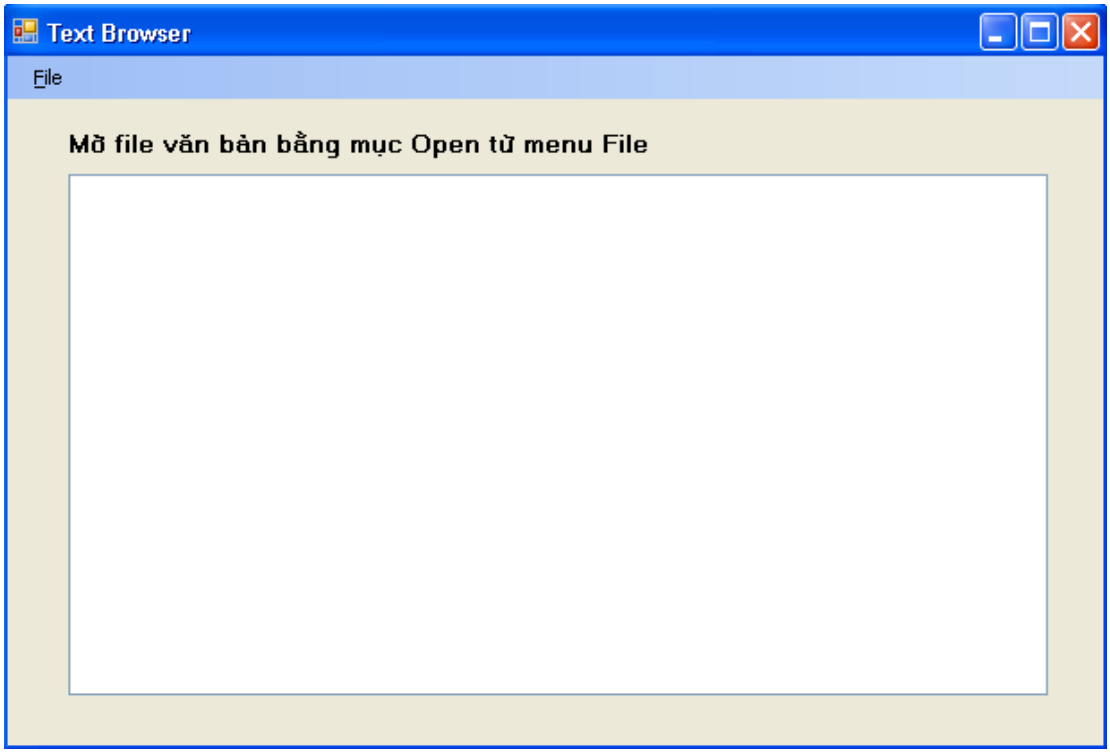
Chương trình *TextBrowser* sau sẽ minh họa cách mở một file text và cho hiển thị nó lên một ô textbox.

Tìm hiểu chương trình:

Chương trình bao gồm một menu File với hai mục chọn là Open cho phép mở file rồi hiển thị nội dung file text đó vào một textbox và mục Close để đóng file.

Thiết kế giao diện:

Bạn tạo một giải pháp mới và thêm vào dự án có tên TextBrowser và thiết kế giao diện như sau:



Trong đó các điều khiển có thuộc tính như sau:

- Textbox1: Enable – False, Multiline – True.
- OpenToolStripMenuItem: Enable – True
- CloseToolStripMenuItem: Enable – False

- Các điều khiển và thuộc tính khác như hình.

Viết mã:

Tạo thủ tục `OpenToolStripMenuItem_Click` như sau:

```
'Khai báo hai biến, một biến lưu toàn bộ văn bản
'một biến lưu từng dòng văn bản
Dim AllText, LineOfText As String
'Tạo bộ lọc file *.txt
OpenFileDialog1.Filter = "Text files (*.txt)| *.txt"
OpenFileDialog1.ShowDialog()
If OpenFileDialog1.FileName <> "" Then
    Try
        'Mở file để đọc
        FileOpen(1, OpenFileDialog1.FileName, OpenMode.Input)
        Do Until EOF(1)
            'Đọc từng dòng đến hết
            LineOfText = LineInput(1)
            'Nối vào biến Alltext
            AllText = AllText & LineOfText & vbCrLf
        Loop
        'Cập nhật nội dung textbox
        Label1.Text = OpenFileDialog1.FileName
        TextBox1.Text = AllText
        ' Loại bỏ đánh dấu chọn cho văn bản
        TextBox1.Select(1, 0)
        'Cho phép soạn thảo
        TextBox1.Enabled = True
        'Cho phép chọn mục Close trên menu
        CloseToolStripMenuItem.Enabled = True
        'Vô hiệu hóa mục Open trên menu
        OpenToolStripMenuItem.Enabled = False
    Catch ex As Exception
        MsgBox("Lỗi mở file")
    Finally
        'Đóng file
        FileClose(1)
    End Try
End If
```

Tiếp theo tạo thủ tục `CloseToolStripMenuItem_Click` như sau:

```
Label1.Text = "Mở file văn bản bằng mục Open từ menu File"
TextBox1.Text = ""
OpenToolStripMenuItem.Enabled = True
CloseToolStripMenuItem.Enabled = False
```

Các bạn có thể đọc các dòng ghi chú màu xanh lá cây để biết công dụng của từng phát biểu của chương trình.

Chạy chương trình:

Các bạn ấn F5 để chạy chương trình và mở một file text bất kỳ để xem chương trình chạy.

Sử dụng lớp StreamReader để mở file Text

Ngoài các hàm mở đọc file như đã biết, chúng ta cũng có thể sử dụng lớp *StreamReader* của VB.NET để thực hiện chức năng tương tự. Để sử dụng lớp này ta cần đặt thêm khai báo `Imports System.IO` ở đầu chương trình.

Sau đây là thủ tục `OpenToolStripMenuItem_Click` đã được viết lại sử dụng lớp StreamReader:

```
Dim StreamReaderToDisPlay As StreamReader
OpenFileDialog1.Filter = "TEXT FILES (*TXT) | *.TXT"
OpenFileDialog1.ShowDialog()
If OpenFileDialog1.FileName <> "" Then
    Try
        StreamReaderToDisPlay = New StreamReader _
        (OpenFileDialog1.FileName)
        Label1.Text = OpenFileDialog1.FileName
        TextBox1.Text = StreamReaderToDisPlay.ReadToEnd
        TextBox1.Enabled = True
        OpenToolStripMenuItem.Enabled = False
        CloseToolStripMenuItem.Enabled = True
    Catch ex As Exception
        MsgBox("Lỗi mở file")
    Finally
        StreamReaderToDisPlay.Close()
    End Try
End If
```

Bạn có thể xem toàn bộ mã chương trình trong giải pháp TextBrowser1 của phần bài tập chương 12.

2. Tạo một file text mới

Tạo file text rất hữu ích khi bạn muốn ghi ra file .log, .ini hay readme. Các bước tổng quát để ghi một file text có thể như sau:

- Nhận dl nhập từ người dùng hay do chương trình tính ra
- Gán dl cho một hay nhiều biến. Ví dụ như gán nội dung textbox1 ra một biến
- Yêu cầu nhập tên file sẽ ghi ra bằng hộp thoại SaveFileDialog
- Sử dụng đường dẫn và tên file do hộp thoại SaveFileDialog trả về và gọi hàm ghi file
- Sử dụng hàm PrintLine để lưu nội dung biến xuống file
- Đóng file khi ghi xong

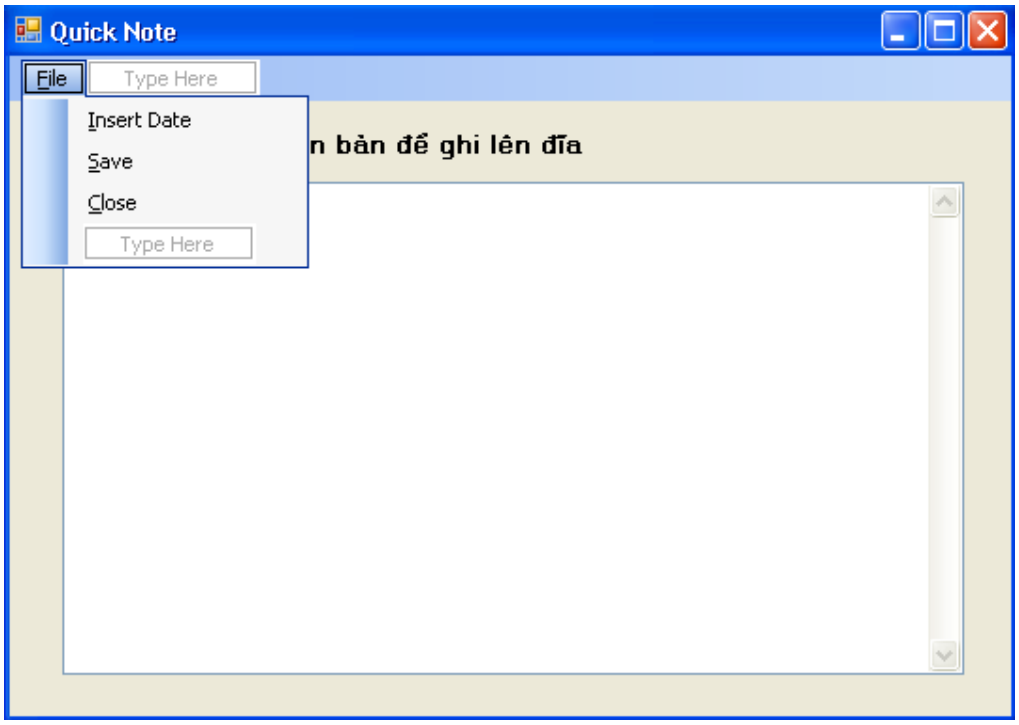
Ví dụ QuickNote sau đây sẽ minh họa việc tạo file text.

Tìm hiểu chương trình:

Chương trình gồm một ô textbox để chế độ multiline hiện scrollbars cả hai chiều (thuộc tính scrollbar – giá trị both), menu File gồm ba mục chọn Insert Date cho phép chèn ngày tháng vào đầu văn bản, mục chọn Save cho phép ghi lại nội dung ô textbox vào một file text với tên do người dùng nhập vào, mục Close đóng chương trình.

Thiết kế giao diện:

Bạn tạo một giải pháp mới và thêm vào một dự án có cùng tên là QuickNote rồi thiết kế giao diện như hình:



Viết mã:

Tạo thủ tục `InsertDateToolStripMenuItem_Click` như sau:

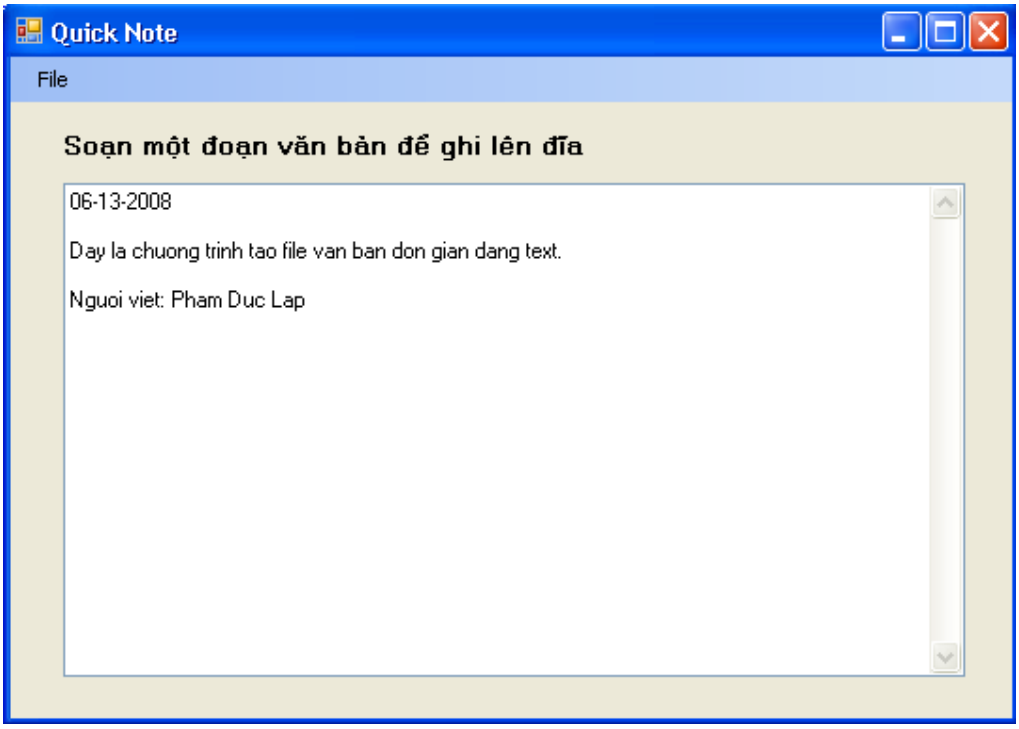
```
TextBox1.Text = DateString & vbCrLf & TextBox1.Text
TextBox1.Select(1, 0)
```

Tạo thủ tục `SaveToolStripMenuItem_Click` như sau:

```
SaveFileDialog1.Filter = "Text File (*.txt) | *.txt"
SaveFileDialog1.ShowDialog()
If SaveFileDialog1.FileName <> "" Then
    Try
        FileOpen(1, SaveFileDialog1.FileName, OpenMode.Output)
        PrintLine(1, TextBox1.Text)
    Catch ex As Exception
        MsgBox("Lỗi khi ghi")
    Finally
        FileClose(1)
    End Try
End If
```

Chạy chương trình:

Bạn hãy chạy chương trình, ghi vào textbox những đoạn văn bản tùy ý hay có thể sao chép từ một file khác, chèn thêm ngày tháng và lưu vào một file nào đó (phải nhập tên file).



3. Xử lý chuỗi trong chương trình

Chuỗi là thông tin cần xử lý nhiều nhất trong mọi ngôn ngữ lập trình. Các thao tác xử lý sẽ học trong chương này bao gồm cắt chuỗi, nối chuỗi, tìm kiếm, sắp xếp, so sánh chuỗi,…

Việc nối chuỗi ta có toán tử & hay có thể dùng phương thức *Concat* của VB ví dụ:

Dim loichao as String

loichao = String.Concat("Hello ", "World, ", "Hi Everyone!")

Bạn có thể dùng các phương thức cũ như Mid, Ucase, Lcase…hay dùng các phương thức mới mà lớp String cung cấp như SubString, ToUpper, ToLower. Thường ta dùng các phương thức mới này hơn.

Bảng sau liệt kê các hàm xử lý chuỗi trong cả .NET và VB truyền thống:

.NET	VB cũ	Chức năng
ToUpper	UCase	Đổi toàn bộ chuỗi sang chữ hoa
ToLower	LCase	Đổi toàn bộ chuỗi sang chữ thường
Length	Len	Trả về chiều dài chuỗi
SubString	Mid	Cắt chuỗi con trong chuỗi cha
IndexOf	InStr	Xác định vị trí chuỗi con trong chuỗi cha
Trim	Trim	Cắt bỏ khoảng trắng trong chuỗi
Remove		Loại bỏ khoảng trắng ở giữa chuỗi

Insert		Chèn chuỗi con vào chuỗi cha
StrComp		So sánh chuỗi

3.1. Sắp xếp chuỗi

Các ký tự ghép lại tạo thành chuỗi. Mỗi ký tự có một mã trong bảng mã ASCII. Bảng mã này có 255 mã. Sau này trong các hệ điều hành như WinXP, NT, 2000 thì chuẩn quốc tế qui định các bảng mã ký tự khác như Unicode. Trong bảng mã này mỗi mã ký tự chiếm 2 byte chứ không phải là 1 byte như trong ASCII. Việc sắp xếp các ký tự là dựa vào mã của ký tự đó. Chữ A mã 65 sẽ đứng trước chữ B có mã 66.

3.2. Làm việc với ký tự ASCII

Để xem một ký tự có mã ASCII là bao nhiêu có thể dùng hàm Asc, ví dụ:

```
Dim AscCode As Short
AscCode = Asc("A")
MsgBox(AscCode)
```

Ngược lại bạn có thể dùng hàm Chr() để biết ký tự nào tương ứng với một mã ASCII cho trước, ví dụ:

```
Dim letter As Char
letter = Chr(65)
MsgBox(letter)
```

Để so sánh chuỗi, bạn có thể dùng các toán tử sau: <>, =, >, <, >=, <=.

Một ký tự coi là lớn hơn nếu mã ASCII của nó lớn hơn, ví dụ "A" < "B" vì mã của "A" là 65 nhỏ hơn 66 là mã của "B".

Khi so sánh hai chuỗi, VB sẽ so sánh lần lượt các ký tự từ đầu tiên đến tiếp theo cho đến khi gặp ký tự khác nhau. Nếu không có ký tự khác nhau thì chuỗi nào dài hơn, chuỗi đó lớn hơn.

3.3. Sắp xếp chuỗi trong ô TextBox

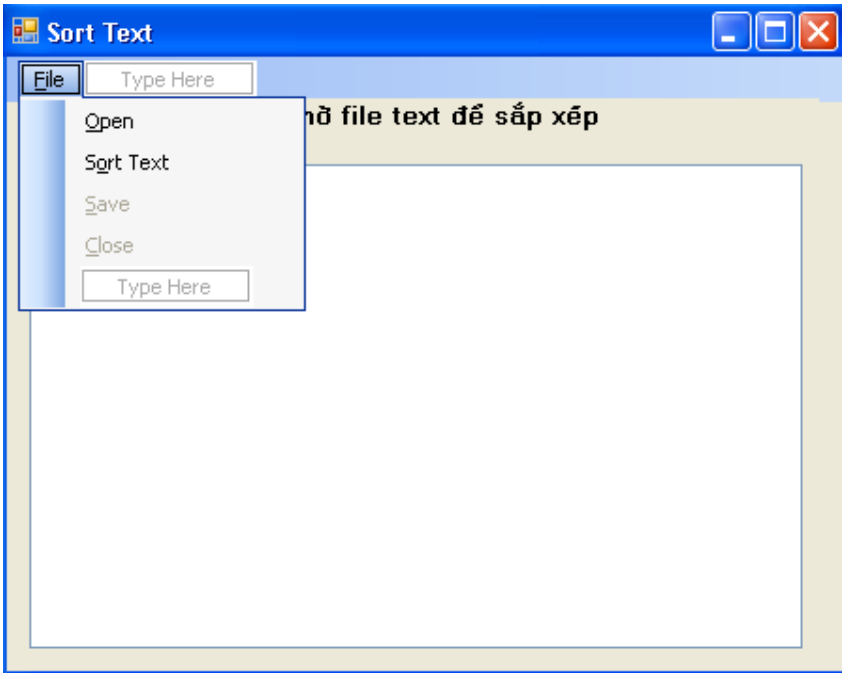
Chương trình SortText sau đây sẽ minh họa việc sắp xếp chuỗi bằng các toán tử so sánh chuỗi trong một ô textbox. Đây là chương trình được nâng cấp từ chương trình QuickNote.

Tìm hiểu chương trình:

Chương trình SortText này có một form chính. Form này có một menu File với 4 mục chọn. Mục chọn Open cho phép mở một file text, mục chọn Sort Text cho phép sắp xếp các dòng trong textbox theo thứ tự tăng dần, mục chọn Save cho phép lưu lại những thao tác đã thực hiện, mục chọn Close để đóng file.

Thiết kế giao diện:

Giao diện nó tương tự như chương trình QuickNote nhưng có thêm các mục chọn như đã liệt kê. Giao diện như hình:



Thuộc tính enable của textbox có giá trị là True để cho phép người dùng nhập văn bản cho chương trình sắp xếp. Nghĩa là chương trình có thể sắp xếp nội dung của một file text được chỉ định hay là người dùng trực tiếp nhập liệu vào.

Viết mã:

Trước hết ta thêm vào dự án một module có tên SortModule. Module này sẽ khai báo một mảng có tên strArr chứa các dòng của văn bản và chứa một hàm `Sapxep()` để sắp xếp các phần tử trong mảng theo thứ tự giảm dần. Nội dung module như sau:

```
Public strArr() As String
Sub Sapxep(ByRef mang() As String, ByVal sophantumang As Short)
    Dim tam As String
    Dim i, j, trungbinh As Short
    'Sắp xếp các phần tử trong mảng mang()
    'Mảng sắp xếp nhị phân theo thứ tự giảm dần
    trungbinh = sophantumang \ 2
    Do While trungbinh > 0
        For i = trungbinh To sophantumang - 1
            j = i - trungbinh + 1
            For j = (i - trungbinh + 1) To 1 Step -trungbinh
                If mang(j) <= mang(j + trungbinh) Then Exit For
                tam = mang(j)
                mang(j) = mang(j + trungbinh)
                mang(j + trungbinh) = tam
            Next j
        Next i
        trungbinh = trungbinh \ 2
    Loop
End Sub
```

Ta có thể lưu lại nội dung module này để sử dụng trong các ví dụ sau.

Tiếp theo ta tạo thủ tục `OpenToolStripMenuItem_Click` như sau:

```
Dim All, line As String
OpenFileDialog1.Filter = "TEXT FILES (*.TXT) | *.TXT"
OpenFileDialog1.ShowDialog()
If OpenFileDialog1.FileName <> "" Then
    Try
        FileOpen(1, OpenFileDialog1.FileName, OpenMode.Input)
        Do Until EOF(1)
            line = LineInput(1)
            All = All & line & vbCrLf
        Loop
        TextBox1.Text = All
        TextBox1.Select(1, 0)
        TextBox1.Enabled = True
        OpenToolStripMenuItem.Enabled = False
        CloseToolStripMenuItem.Enabled = True
        SaveToolStripMenuItem.Enabled = True
    Catch ex As Exception
        MsgBox("Lỗi mở File!")
    Finally
        FileClose(1)
    End Try
End If
```

Thủ tục `CloseToolStripMenuItem_Click` như sau:

```
TextBox1.Text = ""
CloseToolStripMenuItem.Enabled = False
OpenToolStripMenuItem.Enabled = True
SaveToolStripMenuItem.Enabled = False
TextBox1.Enabled = False
```

Thủ tục `SaveToolStripMenuItem_Click`:

```
OpenFileDialog1.Filter = "TEXT FILES (*.TXT) | *.TXT"
OpenFileDialog1.ShowDialog()
If OpenFileDialog1.FileName <> "" Then
    Try
        FileOpen(1, OpenFileDialog1.FileName, OpenMode.Output)
        PrintLine(1, TextBox1.Text)
        OpenToolStripMenuItem.Enabled = True
        SaveToolStripMenuItem.Enabled = False
    Catch ex As Exception
        MsgBox("Lỗi ghi file!")
    Finally
        FileClose(1)
    End Try
End If
```

Bây giờ ta tạo thủ tục `SortTextToolStripMenuItem_Click`:

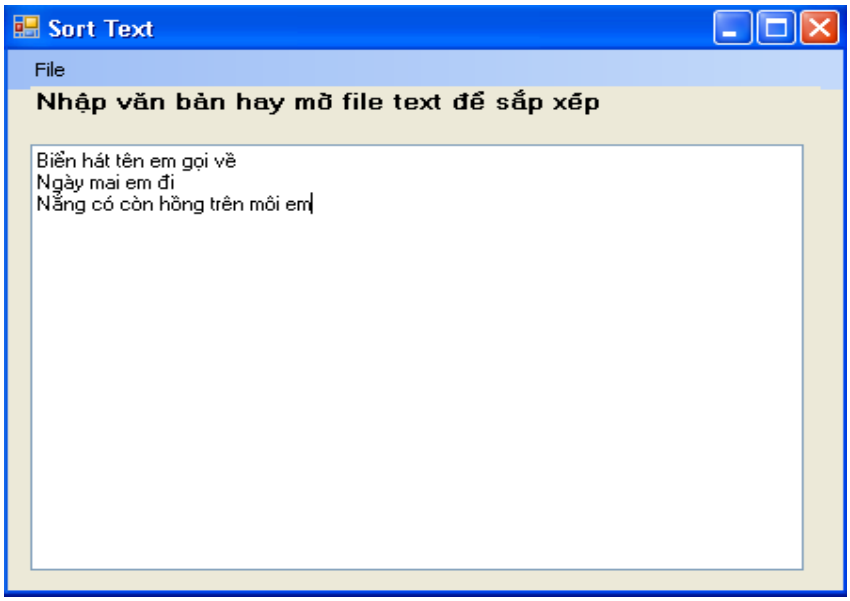
```
Dim ln, curline, letter As String
Dim i, charsInFile, lineCount As Short
'Kiểm tra số dòng trong chuỗi
lineCount = 0
charsInFile = TextBox1.Text.Length 'Lấy chiều dài chuỗi
For i = 0 To charsInFile - 1 'Đếm từng ký tự
    'Lấy nội dung
    letter = TextBox1.Text.Substring(i, 1)
    'Nếu là ký tự xuống dòng
    If letter = Chr(13) Then
        lineCount += 1
        i += 1 'Bỏ qua ký tự 10 (10, 13 thường đi kèm)
    End If
```

```
Next i

'Tạo mảng chứa các dòng văn bản
ReDim strArr(lineCount)
curline = 1
ln = ""

'Duyệt qua ký tự trong chuỗi
For i = 0 To charsInFile - 1
    letter = TextBox1.Text.Substring(i, 1)
    If letter = Chr(13) Then
        curline += 1
        i += 1
        ln = ""
    Else
        'Đưa nội dòng vào mảng
        ln = ln & letter
        strArr(curline) = ln
    End If
Next i

'Sắp xếp mảng
Sapxep(strArr, lineCount)
'Hiển thị mảng đã sắp xếp trở lại TextBox
TextBox1.Text = ""
curline = 1
For i = 1 To lineCount
    TextBox1.Text = TextBox1.Text & _
    strArr(curline) & vbCrLf
    curline += 1
Next i
```

Chạy chương trình:

Bạn có thể chạy chương trình, mở một file text có sẵn hay tự mình nhập vào nội dung văn bản và tiến hành lưu lại.

Bạn có thể thấy chương trình vẫn còn một lỗi nhỏ. Bạn thử tìm ra và khắc phục xem sao.



Trong module trên, với thủ tục sapxep bạn có thể dùng giải thuật sắp xếp khác nếu muốn. Các thủ tục sắp xếp hay tìm kiếm đều có thể tìm thấy trong môn cấu trúc dữ liệu và giải thuật, các bạn có thể tham khảo thêm.

Mở rộng:

Bây giờ nhằm phục vụ các bạn ôn lại các thuật toán trong môn cấu trúc dữ liệu và giải thuật phần sắp xếp, mình sẽ cài đặt các thủ tục sắp xếp khác nhau như sắp xếp chọn (SelectionSort), sắp xếp chèn trực tiếp (Insertion Sort), sắp xếp nổi bọt (Bubble Sort)…

Sắp xếp chọn:

Sắp xếp chọn trong một mảng là chạy một vòng lặp từ đầu đến cuối mảng, chọn ra phần tử nhỏ nhất tiến hành đổi vị trí hai phần tử đó (thứ k) và phần tử thứ nhất. Vòng lặp lại tiếp tục như vậy từ phần tử thứ hai trở đi.

Thủ tục SelectionSort được cài đặt trong module sortmodule như sau:

```
'Sap xep theo phuong phap sap xep chon
Sub SelectionSort(ByRef mang() As String, _
ByVal sophantumang As Short)
    Dim i, j, k As Short
    For i = 1 To sophantumang - 1
        k = i
        For j = i + 1 To sophantumang
            If (mang(j) < mang(k)) Then k = j
        Next j
        doicho(mang(i), mang(k))
    Next i
End Sub
```

Trong đó thủ tục doicho() cài đặt:

```
'Thu tuc doi cho hai phan tu
Sub doicho(ByRef x As String, ByRef y As String)
    Dim tam As String
    tam = x
    x = y
    y = tam
End Sub
```

Để thực thi thủ tục sắp xếp theo phương pháp chọn này, trong chương trình thay vì câu gọi `Sapxep(strArr, lineCount)` thì bạn gọi `SelectionSort(strArr, lineCount)` và chạy chương trình. Kết quả không có gì thay đổi.

Sắp xếp chèn trực tiếp:

Sắp xếp chèn trực tiếp dựa trên ý tưởng như sau: coi mảng đó có đoạn đầu (i-1 phần tử) đã sắp xếp. Ta chạy vòng lặp từ phần tử tiếp theo (phần tử thứ i) và chèn các phần tử tiếp theo đó vào đoạn đầu sao cho theo trật tự qui định (tăng hay giảm). Vậy nếu muốn sắp xếp một mảng thì đoạn đầu tiên sẽ gồm một phần tử duy nhất a[0] và tiến hành chèn các phần tử từ a[1].

Việc chèn tiến hành như sau: lưu a[i] vào biến x. Cho biến j chạy từ đầu mảng (từ 1) và xét xem x < a[j]. Nếu đúng thì đẩy a[j] ra sau một vị trí và giảm j đi 1. Quá trình tiếp tục khi x>=a[j] hay j = 0 và đặt x vào vị trí j+1.

Thủ tục InsertionSort() được cài đặt trong SortModule như sau:

```
'Sap xep theo phuong phap chen truc tiep
Sub InsertionSort(ByRef mang() As String, _
ByVal sophantumang As Short)
    Dim i, j As Short
    Dim tam As String
    For i = 1 To sophantumang
        tam = mang(i)
        j = i - 1
        Do While ((tam < mang(j)) And (j > 0))
            mang(j + 1) = mang(j)
            j -= 1
        Loop
        mang(j + 1) = tam
    Next
End Sub
```

Lúc này cũng tương tự như cách gọi trên, bạn gọi thủ tục này thay cho lời gọi thủ tục sắp xếp `SelectionSort`. Và kết quả không có gì thay đổi.

Sắp xếp nổi bọt:

Sắp xếp nổi bọt là sắp xếp bằng cách đi từ trái qua phải, nếu thấy hai phần tử liền kề nhau không đúng trật tự thì đổi chỗ. Quá trình đó cứ lặp đi lặp lại như vậy cho đến khi thu được dãy có trật tự tăng hay giảm theo ý muốn.

Thủ tục BubbleSort() được cài đặt trong SortModule như sau:

```
'Sap xep theo phuong phap noi bot
Sub BubbleSort(ByRef mang() As String, _
ByVal sophantumang As Short)
    Dim i, j As Short
    i = sophantumang
    Do While i > 0
        For j = 1 To i - 1
            If mang(j) > mang(j + 1) Then
                doicho(mang(j), mang(j + 1))
            End If
        Next j
        i -= 1
    Loop
End Sub
```

Thủ tục đổi chỗ đã được khai báo trong module và trình bày trong phần trên. Bạn cũng thay lời gọi InsertiontSort bằng lời gọi thủ tục BubbleSort và xem kết quả có gì thay đổi không.

*Sắp xếp nhanh **QuickSort**:*

*Sắp xếp trộn **MergeSort**:*

4. Bảo vệ nội dung văn bản bằng cách mã hóa

Bây giờ chúng ta thử mã hóa những gì có trong ô textbox để chỉ mình bạn là người có thể đọc được. Ta dùng một giải thuật mã hóa làm xáo trộn văn bản và một thuật toán giải mã để đưa văn bản trở về trạng thái ban đầu.

4.1. Mã hóa tài liệu bằng cách thay đổi mã ASCII của các ký tự

Bây giờ chúng ta tạo chương trình mã hóa và tiến hành giải mã một file văn bản. Ta làm ví dụ *EncriptionText*.

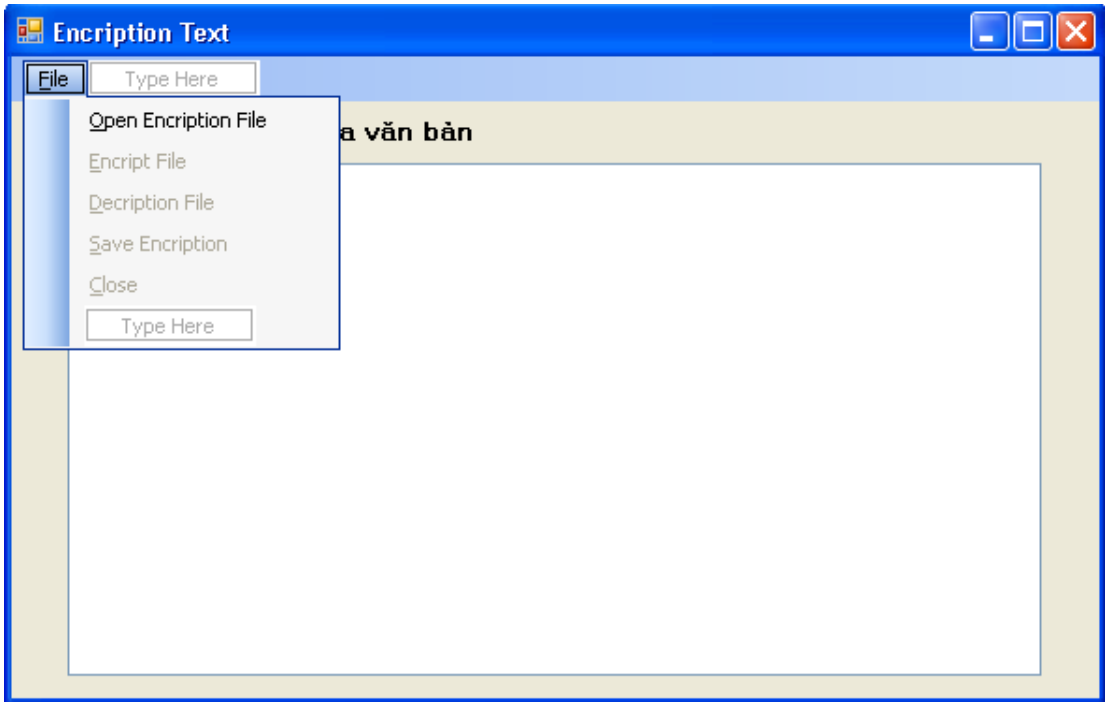
4.2. Chương trình EncriptionText

Tìm hiểu chương trình:

Chương trình gồm một ô textbox để hiển thị nội dung văn bản. Một menu File với các mục chọn: Open Encription File dùng để mở các file đã mã hóa; Encript File dùng để mã hóa nội dung văn bản có trong ô textbox; Decription File giải mã nội dung văn bản đã bị mã hóa; Save Encription lưu lại nội dung văn bản đã mã hóa vào một file; Close để đóng file và đưa ô textbox về trạng thái trắng.

Thiết kế giao diện:

Giao diện thiết kế như hình:



Các bạn tạo các điều khiển như hình và kéo các điều khiển khác gồm OpenFileDialog1 để mở file, MenuStrip1, SaveFileDialog1 để lưu file.

Các thuộc tính thay đổi như sau:

- Open Encription File: name là mnuOpenEncriptionFile
- Encript File: name là mnuEncriptionFile
- Decription File: name là mnuDecription
- Save Encription: name là mnuSaveEncription
- &Close: name là mnuClose
- TextBox: name là txtDocument

Viết mã:

Tạo thủ tục mở file:

```
Private Sub mnuOpenEncriptionFile_Click(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles mnuOpenEncriptionFile.Click
    Dim all, line As String
    OpenFileDialog1.Filter = "TEXT FILES (*.TXT) | *.TXT"
    OpenFileDialog1.ShowDialog()
    If OpenFileDialog1.FileName <> "" Then
        Try
            FileOpen(1, OpenFileDialog1.FileName, OpenMode.Input)
            Do Until EOF(1)
                line = LineInput(1)
                all = all & line
            Loop
            txtDocument.Text = all
            mnuClose.Enabled = True
            mnuDecription.Enabled = True
            mnuOpenEncriptionFile.Enabled = False
        Catch ex As Exception
            MsgBox("Lỗi mở File!")
        Finally
            FileClose(1)
        End Try
    End If
End Sub
```

Thủ tục lưu file:

```
Private Sub mnuSaveEncription_Click(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles mnuSaveEncription.Click
    SaveFileDialog1.Filter = "TEXT FILES (*.TXT) | *.TXT"
    SaveFileDialog1.ShowDialog()
    If SaveFileDialog1.FileName <> "" Then
        Try
            FileOpen(1, SaveFileDialog1.FileName, OpenMode.Output)
            PrintLine(1, txtDocument.Text)
        Catch ex As Exception
            MsgBox("Lỗi mở file!")
        Finally
            FileClose(1)
        End Try
        mnuClose.Enabled = True
        mnuSaveEncription.Enabled = False
```

```
        End If
    End Sub
```

Thủ tục đóng file:

```
    Private Sub mnuClose_Click(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles mnuClose.Click
        txtDocument.Text = ""
        mnuOpenEncriptionFile.Enabled = True
        mnuClose.Enabled = False
        mnuDecription.Enabled = False
        mnuEncriptionFile.Enabled = False
        mnuSaveEncription.Enabled = False
    End Sub
```

Thủ tục mã hóa:

```
    Private Sub mnuEncriptionFile_Click(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles mnuEncriptionFile.Click
        Dim Encript As String = ""
        Dim letter As Char
        Dim i, charsInFile As Short

        charsInFile = txtDocument.Text.Length
        For i = 0 To charsInFile - 1
            letter = txtDocument.Text.Substring(i, 1)
            Encript = Encript & Chr(Asc(letter) + 1)
        Next

        txtDocument.Text = Encript
        mnuClose.Enabled = True
        mnuSaveEncription.Enabled = True
        mnuDecription.Enabled = True
    End Sub
```

Trong thủ tục trên ta mã hóa bằng cách tăng mã của ký tự trong bảng mã ASCII lên một bằng dòng lệnh:

```
Encript = Encript & Chr(Asc(letter) + 1)
```

Thủ tục giải mã:

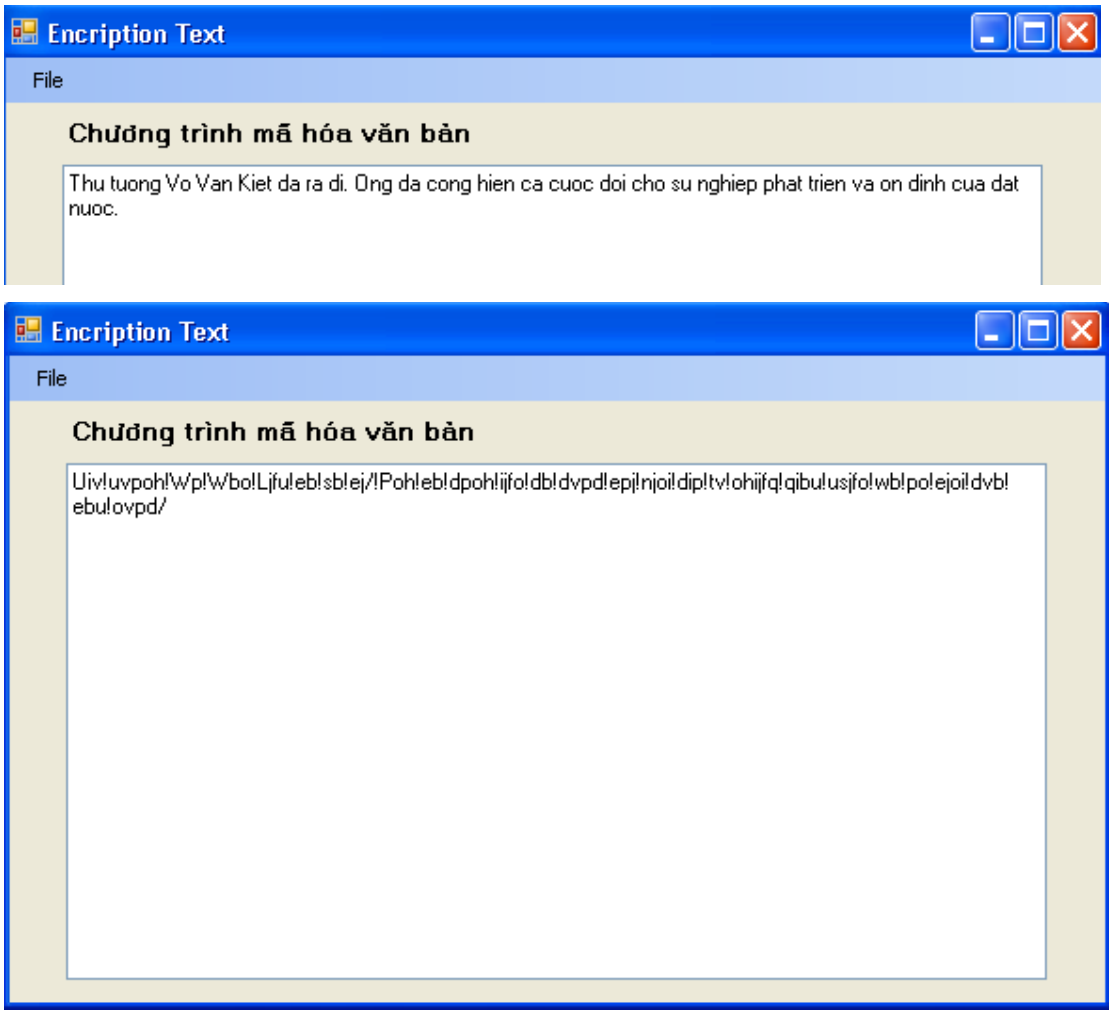
```
    Private Sub mnuDecription_Click(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles mnuDecription.Click
        Dim i, charsInFile As Short
        Dim letter As Char
        Dim Decript As String = ""

        charsInFile = txtDocument.Text.Length
        For i = 0 To charsInFile - 1
            letter = txtDocument.Text.Substring(i, 1)
            Decript = Decript & Chr(Asc(letter) - 1)
        Next

        txtDocument.Text = Decript
        mnuClose.Enabled = True
        mnuSaveEncription.Enabled = False
        mnuEncriptionFile.Enabled = True
        mnuDecription.Enabled = True
    End Sub
```

Việc giải mã là tiến hành trừ mã của ký tự trong bảng mã ASCII đi 1 với dòng mã:

```
Decript = Decript & Chr(Asc(letter) - 1)
```

Chạy chương trình:

Bạn ấn F5 chạy chương trình và có thể mở file văn bản hay gõ nội dung vào textbox để mã hóa và giải mã. Điều này thật thú vị.



5. Sử dụng toán tử XOR trong mã hóa

Với cách mã hóa trên đây có rất nhiều hạn chế. Hạn chế đầu tiên đó là mã ASCII chỉ giới hạn từ 0-255, nếu cộng ra ngoài khoảng này thì chương tình sẽ gặp lỗi ngay.

Cách khắc phục là ta dùng toán tử XOR, khi XOR một số với một giá trị nào đó hai lần thì bạn sẽ nhận lại chính số đó.

Chúng ta hãy làm rõ điều này thông qua ví dụ **XorEccriptionTextFile**:

Tìm hiểu chương trình:

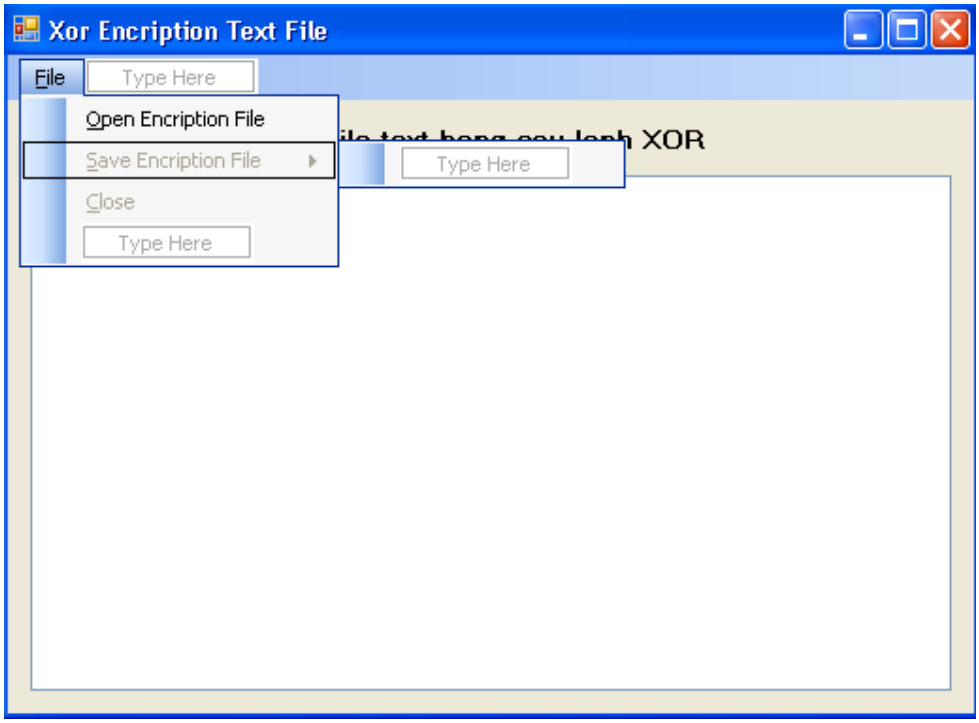
Chương trình có một textbox cho phép hiển thị cũng như nhập liệu văn bản; một menu File với ba mục chọn là Open Encription File cho phép mở file đã mã hóa. Khi mở file này chương trình sẽ yêu cầu người dùng nhập chính xác khóa đã dùng để mã hóa khi mã hóa.

Mục chọn thứ hai là Save Encription File. Mục này cho phép người dùng mã hóa nội dung văn bản gõ vào trong ô textbox. Chương trình sẽ đưa ra thông báo yêu cầu người dùng nhập vào một khóa dùng để mã hóa. Người dùng cần nhớ chính xác từ khóa này để giải mã file sau này.

Mục chọn Close sẽ đưa textbox về dạng trắng và hiện mục Open Encription File lên cho người dùng mở file, hoặc gõ văn bản vào ô textbox để làm sáng mục chọn Save.

Thiết kế giao diện:

Giao diện của chương trình như hình:



Bạn tạo giải pháp và thêm vào dự án cùng tên rồi thiết kế giao diện như hình.

Viết mã:

Tạo thủ tục `mnuSaveEncriptionFile_Click` mã hóa văn bản và lưu vào file:

```
Private Sub mnuSaveEncriptionFile_Click(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles mnuSaveEncriptionFile.Click
    Dim letter As Char
    Dim key As String
    Dim i, charsInFile, code As Short

    SaveFileDialog1.Filter = "TEXT FILES (*.TXT) | *.TXT"
    SaveFileDialog1.ShowDialog()
    If SaveFileDialog1.FileName <> "" Then
```

```
            key = InputBox("Nhap khoa ma hoa:")
            If key = "" Then Exit Sub

            code = CShort(key)
            charsInFile = txtDocument.Text.Length
            FileOpen(1, SaveFileDialog1.FileName, OpenMode.Output)
            For i = 0 To charsInFile - 1
                letter = txtDocument.Text.Substring(i, 1)
                Print(1, Asc(letter) Xor code)
            Next
            FileClose(1)
            mnuClose.Enabled = True
        End If
    End Sub
```

Trước hết, chương trình sẽ yêu cầu nhập vào một khóa bằng phát biểu:

```
key = InputBox("Nhap khoa ma hoa:")
```

Sau đó nó tiến hành XOR mã ASCII của ký tự với khóa và ghi vào file bằng phát biểu:

```
Print(1, Asc(letter) Xor code)
```

Thủ tục `mnuOpenEncriptionFile_Click` giải mã:

```
Private Sub mnuOpenEncriptionFile_Click(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles mnuOpenEncriptionFile.Click
    Dim ch As Char
    Dim key As String
    Dim code, number As Short
    Dim DeEncript As String = ""

    OpenFileDialog1.Filter = "TEXT FILE (*.TXT) | *.TXT"
    OpenFileDialog1.ShowDialog()
    If OpenFileDialog1.FileName <> "" Then
        Try
            key = InputBox("Nhap dung khoa da ma hoa:")
            If key = "" Then Exit Sub
            code = CShort(key)
            FileOpen(1, OpenFileDialog1.FileName, OpenMode.Input)
            Do Until EOF(1)
                Input(1, number)
                ch = Chr(number Xor code)
                DeEncript = DeEncript & ch
            Loop
            txtDocument.Text = DeEncript

            txtDocument.Enabled = True
            mnuClose.Enabled = True
            mnuOpenEncriptionFile.Enabled = False
        Catch ex As Exception
            MsgBox("Loi mo File!")
        Finally
            FileClose(1)
        End Try
    End If
End Sub
```

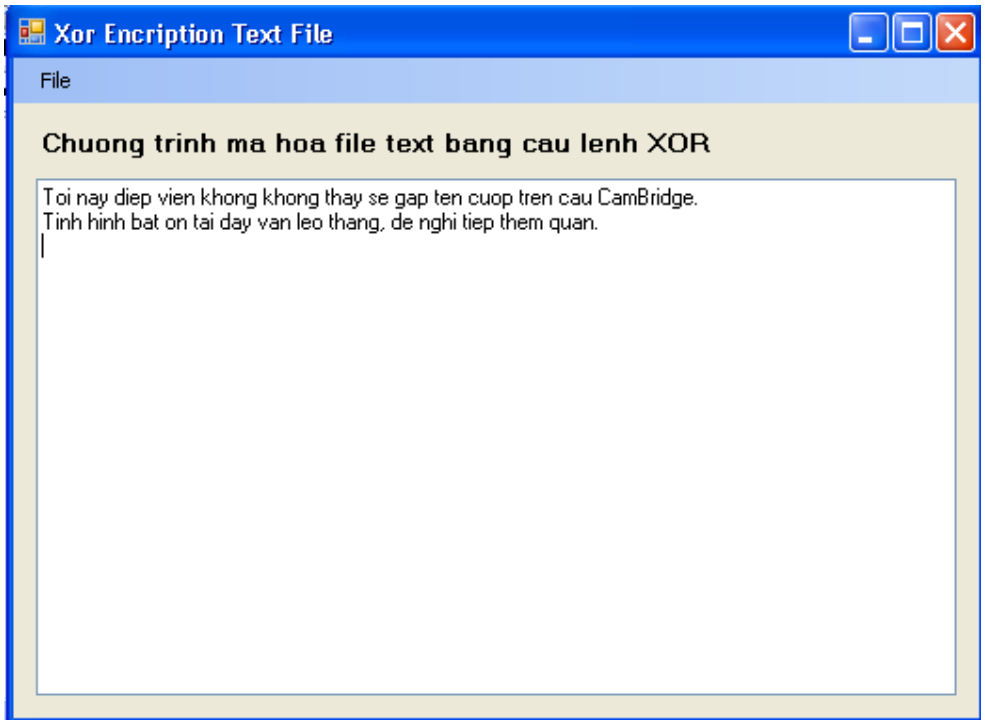
Thủ tục này cũng tương tự. Trước hết yêu cầu nhập vào khóa đã dùng để mã hóa và sau đó tiến hành giải mã bằng cách XOR lại một lần nữa để thu được mã ASCII ban đầu và chuyển trở lại ký tự bằng hàm Chr:

```
ch = Chr(number Xor code)
```

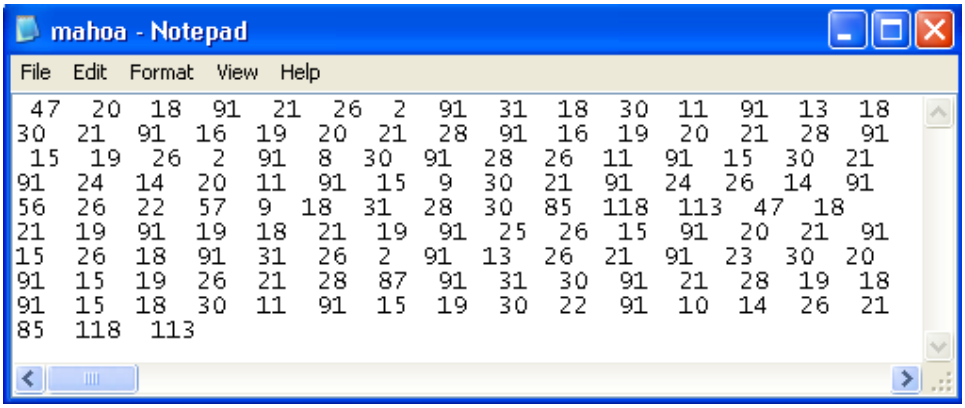
Vậy là chương trình đã hoàn thành.

Chạy chương trình:

Bạn chạy chương trình, nhập vào textbox một đoạn văn bản bất kỳ:



Tiến hành mã hóa và lưu lại vào file có tên mahoa.txt với từ khóa là 123. File đó sẽ có nội dung như sau:



Bây giờ bạn đóng lại bằng cách chọn mục Close và mở lại file trên xem sao.

6. Tổng kết chương

Bạn có thể làm lại các bài tập đã thực hiện trong chương này.

Chương 13:

Tự động hóa trong ứng dụng Microsoft và quản lý tiến trình

--------oOo--------

Nội dung thảo luận:

- Sử dụng Object Browser để tìm hiểu đối tượng
- Sử dụng Microsoft Excel để tính toán trong chương trình VB
- Xử lý bảng tính Excel từ trong chương trình VB.NET
- Khởi động và chấm dứt tiến trình Windows bằng thành phần quản lý tiến trình

Trong chương này chúng ta sẽ thử quản lý các ứng dụng văn phòng của Microsoft từ chương trình VB.NET, xem các đối tượng với thuộc tính, phương thức bằng Object Browser.

Chú ý:

- Dù giờ đây tính năng ActiveX dựa trên mô hình công nghệ COM không còn được thiết kế trong VB.NET nữa tuy nhiên bạn vẫn có thể sử dụng chúng như đã nói trong chương trước.
- Các ứng dụng văn phòng vẫn có thể truy xuất theo đặc tả COM và có thể điều khiển ngay trong ứng dụng VB.NET thông qua kỹ thuật Automation. Tuy nhiên các tham chiếu nên sử dụng lúc biên dịch thay cho lúc chương trình thực thi.
- Giờ đây trong VB.NET lớp đối tượng Process được đặt ngay trong TOOLBOX để phục vụ cho công việc khởi tạo, điều khiển một ứng dụng bên ngoài.

1. Lập trình điều khiển các đối tượng ứng dụng bằng Atomation

Automation là kỹ thuật dựa trên công nghệ và nền tảng của mô hình thành phần đối tượng COM (Component Object Model). COM cho phép các đối tượng của ứng dụng khác nhau có thể chạy chung như trong một môi trường. Ví dụ, bạn có thể tận dụng các ứng dụng soạn thảo văn bản trong chương trình của bạn. Với ASP.NET thì bạn có thể dùng một số đối tượng soạn thảo như FreeTextBox 2.0…. Với VB.NET, để có thể sử dụng COM thì bạn cần làm theo một số bước sẽ được giới thiệu sau đây.

Hiện tại bạn có thể truy xuất các ứng dụng văn phòng Microsoft Office sau theo mô hình COM:

- Microsoft Visual Studio.NET, Microsoft Visual 6.
- MS.World 2003, 2002, 2000, 97
- MS.Excel 2003, 2002, 2000, 97
- MS.Powerpoint 2003, 2002, 2000, 97

- MS.Outlook 2003, 2002, 2000, 97 – 98

Hiện nay Microsoft đã chính thức cho phép VB là ngôn ngữ lập trình chính thức cho các ứng dụng nên hầu hết các ứng dụng windows hỗ trợ Automation đều cung cấp cho bạn cách tương tác và viết lệnh rất giống với chương trình VB.

1.1. Sử dụng Automation trong VB.NET

Trong VB bạn vừa có thể tạo ra ứng dụng dạng đối tượng dạng server, client. Trong phần này chúng ta sẽ sử dụng các đối tượng server có sẵn.

Trong ứng dụng microsoft office đều có một tập hợp đối tượng với phương thức hỗ trợ riêng. Bạn có thể xem chúng qua tài liệu hướng dẫn hay nhờ đối tượng Object Browser.

1.2. Công cụ Visual Studio Object Browser

Công cụ này cho phép bạn thực hiện quan sát nội dung đối tượng như:

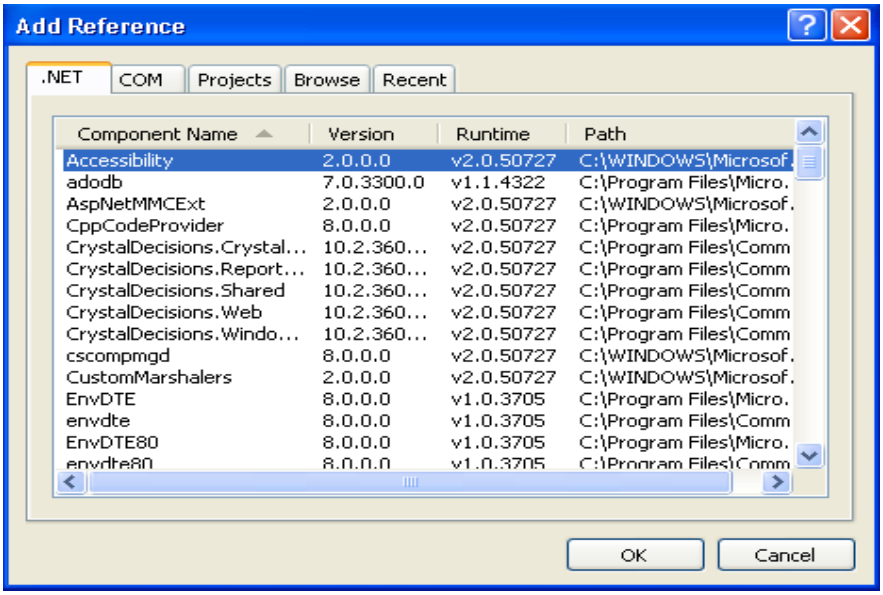
- Xem tất cả các thuộc tính, phương thức và tập đối tượng con của VB.NET. Bạn có thể biết tham số, kiểu dữ liệu của thuộc tính và mục đích sử dụng của chúng.
- Xem tất cả các thuộc tính, phương thức và tập con của các ứng dụng Automation server đang cài trên hệ thống của bạn.

Bây giờ chúng ta sẽ tạo chương trình MyExcelAutomation để dùng đối tượng Object Browser xem các đối tượng trong ứng dụng MS.Excel 2003.

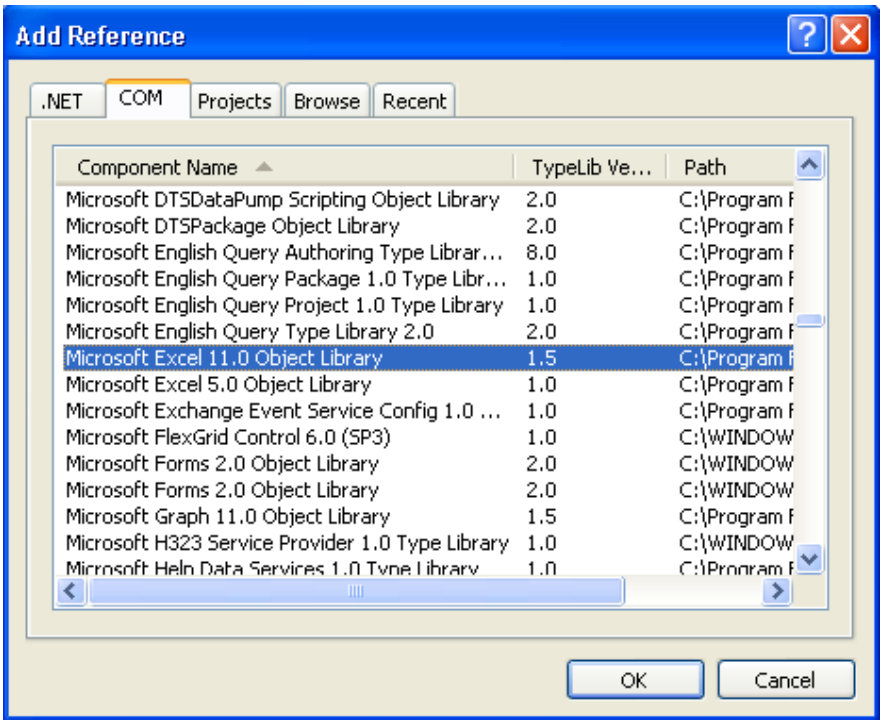
Chương trình MyExcelAutomation:

Bạn tạo mới một giải pháp và thêm vào dự án cùng tên MyExcelAutomation như đã biết. Sau đó làm theo các bước sau:

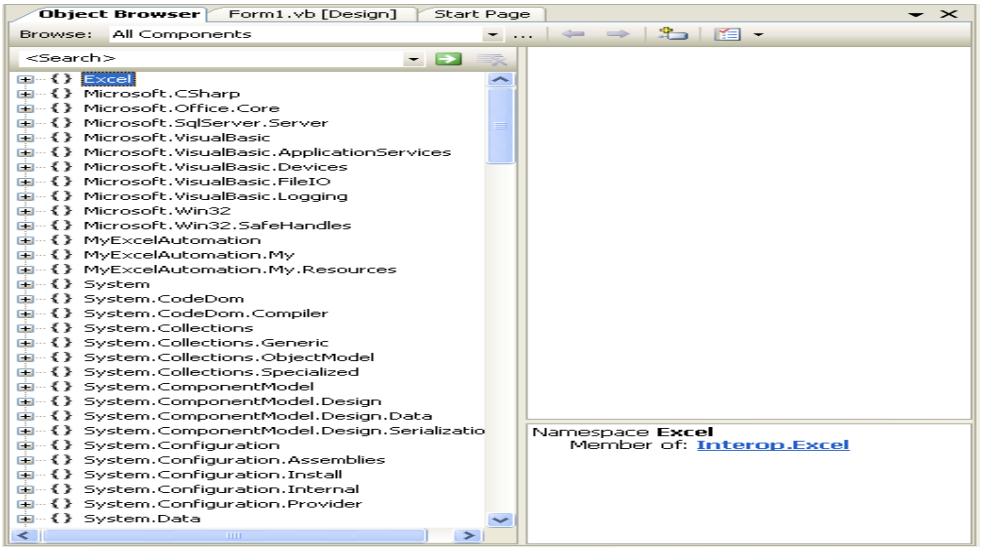
- Chọn Project | Add Reference làm xuất hiện một hộp thoại như hình:



- Co 5 tab như hình, mỗi tab chứa một tập các đối tượng dùng theo các chức năng riêng biệt.
- Nhấn chọn tab COM, các đối tượng trong tab này được hệ thống đăng ký sẵn trong Registry của windows (bạn có thể xem Registry bằng cách mở hộp thoại run và gõ lệnh *Regedit*)
- Bạn cuộn xuống và tìm Microsoft Excel 11.0 Object Library, nhấn nút OK như hình



- Chọn View | Object Browser hay ấn phím F2 để mở cửa sổ công cụ Object Browser như hình:



- Bạn nhấn vào nút (+) kế bên Excel và xem chi tiết về thông tin của các đối tượng Automation mà Excel cung cấp.
- Nếu cần bạn có thể ấn tiếp nút (+) nhánh Application để xem danh sách các phương thức, sự kiện và thuộc tính dành cho đối tượng sẽ xuất hiện trong khung Member of: **Excel** bên phải. Chúng là những phương thức cho phép bạn truy cập bảng tính Excel.
- Bạn có thể xem bất kỳ phương thức nào trong phần được liệt kê trong khung member. Ví dụ ta có thể xem phương thức *Quit* là phương thức đóng ứng dụng Excel cũng như các ứng dụng Automation.

Bây giờ sau khi đã xem xét các phương thức, đối tượng với Object Browser chúng ta sẽ sử dụng Atomation Excel trong chương trình của mình.

2. Tự động hóa bằng Atomation EXCEL trong chương trình Visual Basic

Sau đây là các bước tổng quát để sử dụng các lệnh Excel cũng như các đối tượng Automation nói chung:

- B1. Thêm vào dự án tham chiếu trỏ đến thư viện đối tượng mà bạn muốn truy xuất bằng lệnh Add Reference như đã biết trong phần 1.
- B2. Viết chương trình VB. Dùng phát biểu Dim để khai báo sử dụng đối tượng Atomation. Tiếp theo dùng hàm Ctype và CreatObject để tạo một thể hiện cho đối tượng:

```
Dim xlApp As Excel.Application
xlApp = CType(CreateObject("Excel.Application"), _
Excel.Application)
```

Trong VB.NET sử dụng cơ chế ràng buộc sớm (early binding). Sau khi tạo đối tượng nên dùng hàm Ctype chuyển đổi đối tượng về một kiểu tường minh để khi biên dịch VB.NET biết đây là kiểu gì.

- B3. Sử dụng các phương thức, thuộc tính của đối tượng Atomation trong chương trình của mình. Nếu chưa nắm rõ bạn có thể xem lại bằng Object Browser như đã biết. Ví dụ:

```
Dim loanPayment As Single
loanPayment = xlApp.WorksheetFunction.Pmt _
(txtInterest.Text / 12, txtMonths.Text, txtPrincipal.Text)
```

- B4. Khi đã hoàn thành việc sử dụng đối tượng Automation thì bạn gọi phương thức Quit để chấm dứt sử dụng chúng:

```
xlApp.Quit()
```

Trong bài tập ExcelPayment sau đây chúng ta sẽ dùng tính toán tự động của Excel để tính số tiền phải trả khi thuê nhà.

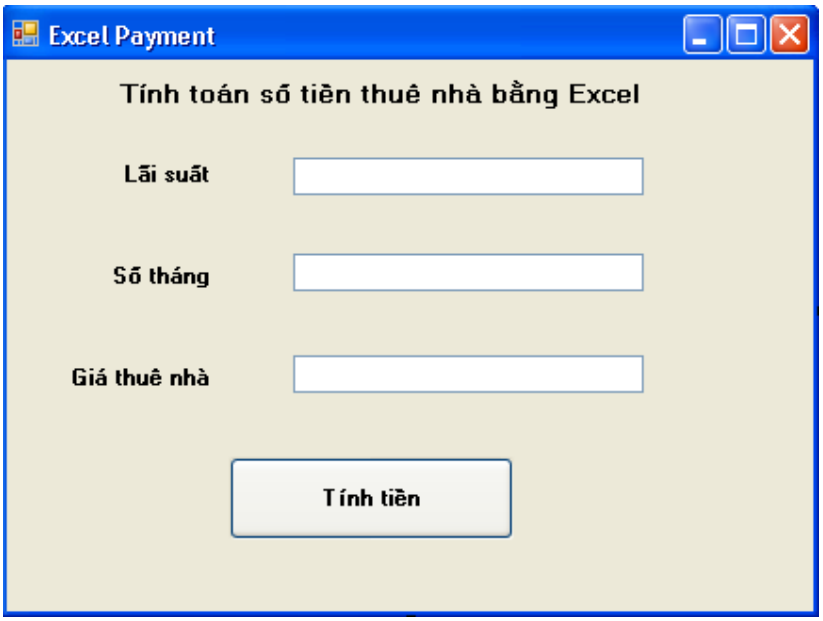
2.1. Chương trình ExcelPayment

Tìm hiểu chương trình:

Chương trình có một form, form có ba textbox cho phép nhập vào mức lãi suất, số tháng và số tiền đơn giá thuê nhà. Khi người dùng click vào nút tính tiền thì chương trình sẽ tính số tiền hàng tháng phải trả.

Thiết kế giao diện:

Giao diện chương trình như hình:



Trong đó thuộc tính name của các textbox như sau: "Lãi suất" – txtLaisuat, "Số tháng" – txtSothang, "Giá thuê nhà" – txtGia. Button "Tính tiền" - btnTinhtien

Viết mã:

Do chương trình có sử dụng một số hàm toán học như Abs nên ta cần khai báo Imports lớp Math ở đầu form như sau:

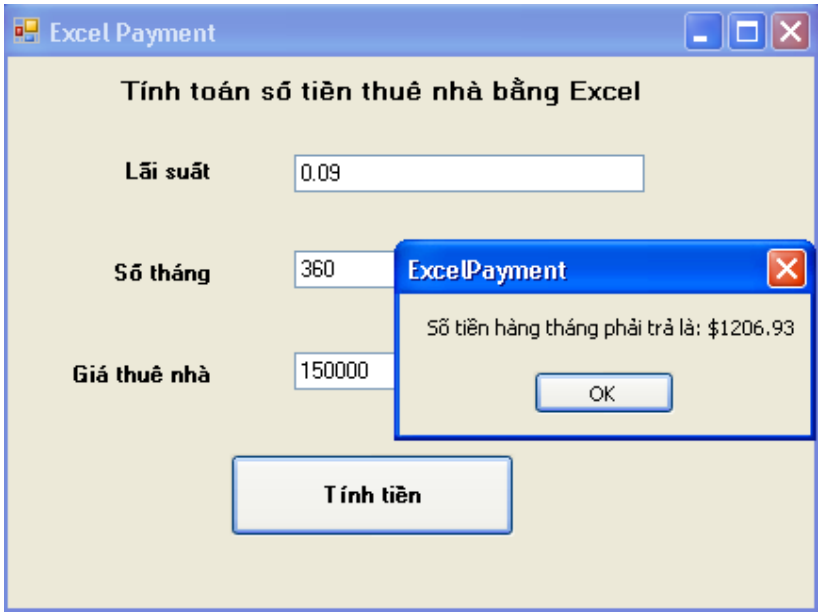
```
Imports System.Math
```

Tạo thủ tục `btnTinhtien_Click` như sau:

```
Dim xlApp As Excel.Application
Dim LoanPayment As Single
xlApp = CType(CreateObject("Excel.Application"),   Excel.Application)
LoanPayment = xlApp.WorksheetFunction.Pmt _
(txtLaisuat.Text / 12, txtSothang.Text, txtGia.Text)
MsgBox("Số tiền hàng tháng phải trả là: " & _
Format(Abs(LoanPayment), "$#.##"), , "ExcelPayment")
xlApp.Quit()
```

Chạy chương trình:

Bạn chạy chương trình và nhập vào các ô textbox các giá trị lần lượt là: 0.09, 360, 150000 và ấn nút Tính tiền để xem kết quả.



2.2. Xử lý bảng tính Excel

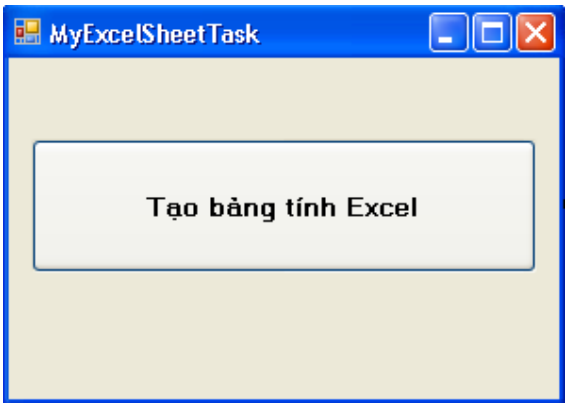
Bây giờ chúng ta sẽ xử lý bảng tính Excel thông qua ví dụ MyExcelSheetTask sau đây.

Tìm hiểu chương trình:

Chương trình khá đơn giản, nó chỉ có một nút nhấn cho phép tạo một bảng tính Excel với một hàm tính tổng, một dạng định dạng font và lưu vào một thư mục trong ổ cứng.

Thiết kế giao diện:

Giao diện như hình:



Viết mã:

Trước hết bạn cần tạo tham chiếu đến Automation Excel như đã biết.

Tiếp theo tạo thủ tục `Button1_Click` như sau:

```
        Dim xlApp As Excel.Application
        Dim xlBook As Excel.Workbook
        Dim xlSheet As Excel.Worksheet
        xlApp = CType(CreateObject("Excel.Application"),
Excel.Application)
        xlBook = CType(xlApp.Workbooks.Add, Excel.Workbook)
        xlSheet = CType(xlBook.Worksheets.Add, Excel.Worksheet)

        'Chèn dữ liệu
        xlSheet.Cells(1, 2) = 5000
        xlSheet.Cells(2, 2) = 75
        xlSheet.Cells(3, 1) = "Tổng"

        'Chèn công thức tính tổng
        xlSheet.Range("B3").Formula = "=Sum(R1C2:R2C2)"

        'Định dạng các ô
        xlSheet.Range("B3").Font.Bold = True
        'Hiển thị bảng tính
        xlSheet.Application.Visible = True
        'Lưu bảng tính vào thư mục nào đó
        xlSheet.SaveAs("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh
vb.net\Tung buoc lap trinh vb.net\13_Chapter13\Bai
tap\MyExcelSheetTask\MyExcelSheetTask")
```

Chạy chương trình:

Bạn nhấn F5 để chạy chương trình.

3. Khởi động và dừng một chương trình Windows sử dụng đối tượng *Process*

Như ta đã biết phương thức Process có thể khởi động hay dừng bất kỳ ứng dụng hay tài liệu nào đã đăng ký trong Registry của hệ thống. Process.Start có thể tự tìm đường dẫn của chương trình để khởi động. Nhược điểm của nó là ta vẫn phải tạo ra bằng hàm CreatObject như các đối tượng Automation khác.Việc chấm dứt chương trình chúng ta phải dựa vào thành phần đối tượng Process trên bảng công cụ Component của ToolBox thực hiện.

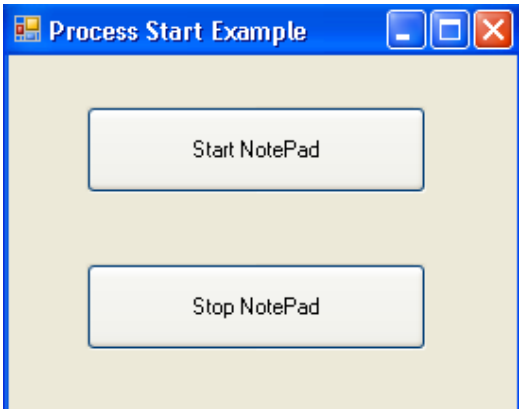
Trong bài tập **MyStarApplication** sau đây chúng ta sẽ dùng Process để khởi động và dừng chương trình NotePad của windows. Bạn có thể dùng Process để khởi động hay chấm dứt bất kỳ chương trình nào của windows.

Tìm hiểu chương trình:

Chương trình gồm hai nút nhấn, nút "Start NotePad" cho phép khởi động chương trình NotePad. Nút nhấn "Stop NotePad" cho phép dừng chương trình NotePad giống như khi bạn click vào nút (X) bên góc phải trên của chương trình như vẫn thấy.

Thiết kế giao diện:

Bạn tạo mới một giải pháp và thêm vào một dự án có cùng tên là MyStartApplication, tiếp theo thiết kế giao diện như hình:



Viết mã:

Bạn thêm vào form một đối tượng Process trên ToolBox bằng cách double click vào đối tượng đó hay kéo thả. Đổi thuộc tính Name của nó thành noteProcess. Để cho noteProcess khởi động chương trình NotePad bạn có thể chỉ định ngay chương trình NotePad.exe trong thuộc tính FileName của phần StartInfo (click vào dấu + bên trái) hay chỉ định trong lúc chương trình thực thi.

Để sử dụng lớp đối tượng Process ta cần khai báo yêu cầu sử dụng lớp này bằng phát biểu Imports ở đầu mã như sau:

```
Imports System.Threading
Imports System.Diagnostics
```

Ở đây mình chỉ định sẵn trong thuộc tính FileName của nó là NotePad.Exe.

Tiếp theo tạo thủ tục `btnStartNote_Click` (btnStartNote là thuộc tính Name của nút nhấn "Start NotePad") như sau:

```
noteProcess.Start()
```

Tạo thủ tục `btnStopNote_Click` (btnStopNote là thuộc tính Name của nút nhấn "Stop NotePad") như sau:

```
noteProcess.CloseMainWindow()
```

Phương thức CloseMainWindow() tương đương với việc người dùng click vào nút nhẫn close trên góc phải thanh tiêu đề cửa sổ.

Bạn có thể dùng phương thức Kill() nhưng phương thức này không thân thiện, nó giống như việc bạn dùng End Task của Task Manager vậy.

Chạy chương trình:

Ấn F5 để thực thi chương trình.

4. Tổng kết chương 13

Bạn tạo bảng liệt kê những gì đã biết trong chương này và làm lại các ví dụ có thể mở rộng thêm như xuất các tính toán ra một bảng tính excel.

Chương 14:

Phân phối và đóng gói ứng dụng Visual Basic.NET

--------oOo--------

Nội dung thảo luận:

- Thêm vào dự án phân phối (Deployment Project)
- Chạy chương trình Setup Winzard để tạo chương trình cài đặt cho từng ứng dụng của chúng ta
- Tùy biến trình cài đặt thông qua các thiết lập và thuộc tính khi xây dựng bộ cài
- Kiểm tra việc cài đặt và gỡ bỏ ứng dụng khỏi hệ thống

Khi đã hoàn tất chương trình ta có thể đóng gói ứng dụng bằng cách tạo bộ cài đặt để đem đi cài trên máy khác được. Chúng ta sẽ học cách tạo thêm dự án đóng gói cài đặt với các thư viện cần thiết.

Chú ý:

- Trong ứng dụng VB.NET, phân phối ứng dụng bằng cách đưa thêm vào dự án phân phối cho giải pháp hiện hành. Các tùy chọn được thiết lập trực tiếp trong dự án phân phối này.
- Visual Studio loại bỏ việc dùng mô hình COM bằng cách cài thêm bộ khung .NET Framework phía máy khách (nếu chưa có), đóng gói ứng dụng thành từng gói và không cần dùng đến các file DLL nữa.
- Các ứng dụng VB.NET có thể cài đặt mà không động chạm gì đến Registry của hệ điều hành nữa.

1. Hoạch định triển khai đóng gói dự án

Cài đặt theo mô hình COM trước kia có rất nhiều hạn chế như: các đăng ký DLL bị trùng lắp, không phù hợp phiên bản, xung đột. Một số chương trình cài đặt được nhưng không thực thi được, cài đặt xong không gỡ được hay gỡ nhưng vẫn để lại rác…Giờ đây VB.NET đã khắc phục những nhược điểm đó vì nó có thể cài đặt mà không cần đăng ký vào Registry của hệ thống. Nó chủ yếu dựa trên bộ khung .NET Framework thay vì yêu cầu đối tượng COM gọi đến hàm API của Windows. Chương trình sẽ được biên dịch thành các gói (Assembly) và chương trình cài đặt sẽ ráp các phần lại cùng với các file thư viện cần thiết.

Một gói Assembly của ứng dụng .NET gồm 4 thành phần: ngôn ngữ diễn dịch trung gian MSIL (Microsoft Intermediate Language), mã (MSIL code), dữ liệu mô tả (metadata) và file chứa thông tin (manisfest file), các file hỗ trợ (support files) và tài nguyên (resource) dành cho chương trình.

Hình sau minh họa chương trình Luckyseven.exe được xem là một gói Assembly với 4 thành phần cơ bản:

LuckySeven.EXE
MSIL Code
Metadata - Định nghĩa kiểu dữ liệu, phương thức - Các tham chiếu về kiểu dữ liệu, đối tượng, phương thức
Manifest - Tên và phiên bản của chương trình - Danh sách các file - Các tham chiếu đến gói (Assembly) khác - Các thông tin bảo mật - Các kiểu xuất (Exported) cho môi trường sử dụng - Các thông tin về tài nguyên
Resource - Các tài nguyên chương trình cần dùng

2. Các cách đóng gói và triển khai ứng dụng khác nhau

Khi bắt đầu chuẩn bị cho việc phân phối sản phẩm của mình, hãy xem xét các phương thức cài đặt khác nhau:

- Cài dặt ứng dụng lên máy tính và đăng ký nó với Windows Registry
- Tạo chương trình cài đặt cho phép chương trình có thể cài từ mạng LAN hay Internet
- Đóng gói ứng dụng cho phép cài từ CD
- Đóng gói ứng dụng thành các file .CAB (một dạng file tự bung) có thể dùng cho các trình duyệt download về hay sao chép đi cài ở bất kỳ đâu.

Trong VS có thể nhanh chóng tạo ra dự án đóng gói Deployment bằng trình Setup Winzard. Việc tùy biến dự án là hoàn toàn dễ dàng bằng cách thay đổi các tùy chọn. Nếu muốn đóng gói lên CD thì cần có đầu ghi CD.

Trên mỗi máy chạy chương trình .NET cần có một bộ khung .NET FrameWork. Bạn có thể download bộ cài là file Dotnetfx.exe (20MB) về cài đặt. Sau này trong các phiên bản hệ điều hành sẽ chứa sẵn .NET Framework Runtime. Nếu máy đã có sẵn thì việc cài đặt chỉ đơn giản là sao chép và chạy.

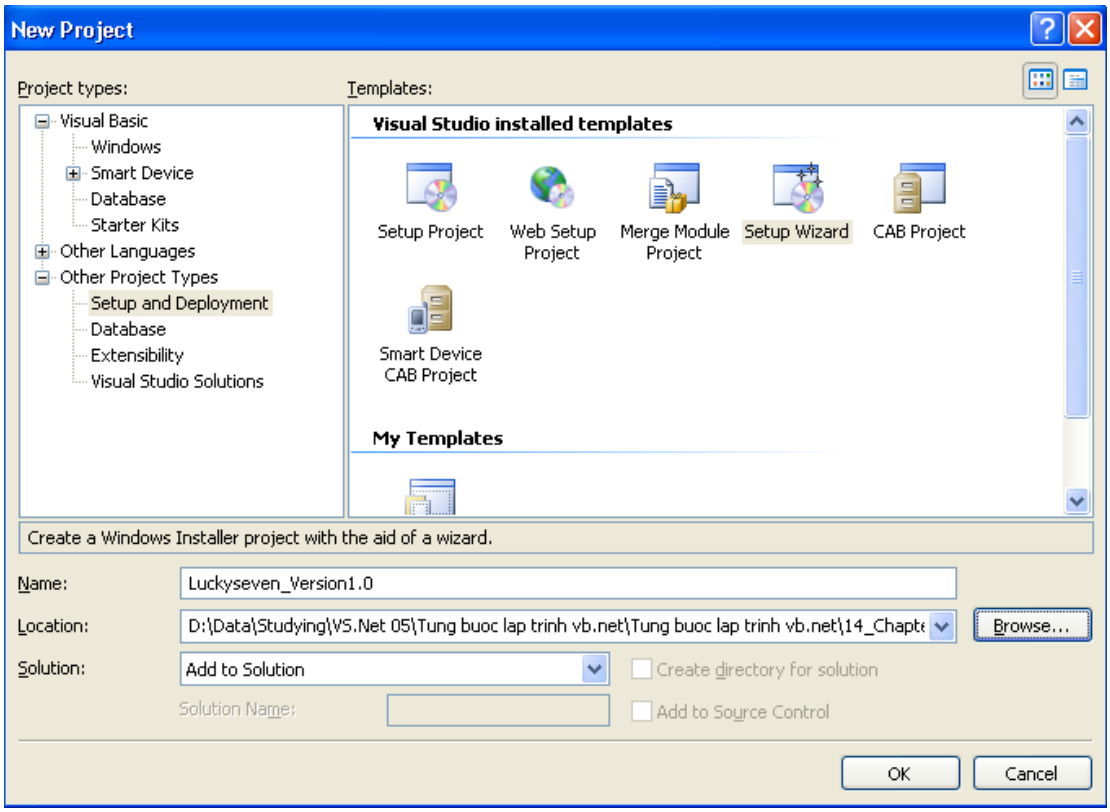
Tuy nhiên hầu như khi đóng gói VS đã nhúng luôn thư viện .NET runtime kèm theo chương trình nên nếu dung lượng bộ cài có lớn hơn nhiều so với chương trình thì bạn cũng đừng ngạc nhiên.

3. Tạo dự án Deployment

Bây giờ chúng ta sẽ tạo dự án Deployment. Chương trình setup của chúng ta sẽ tạo một short cut icon cho chương trình trên menu Start | Programs của windows, thêm file readme vào thư mục C:\Program Files \ microsoft press\Lucky Seven, đăng ký với Registry để cuối chương tiến hành gỡ bằng chức năng Add/Remove Programs.

Bạn làm theo các bước sau đây:

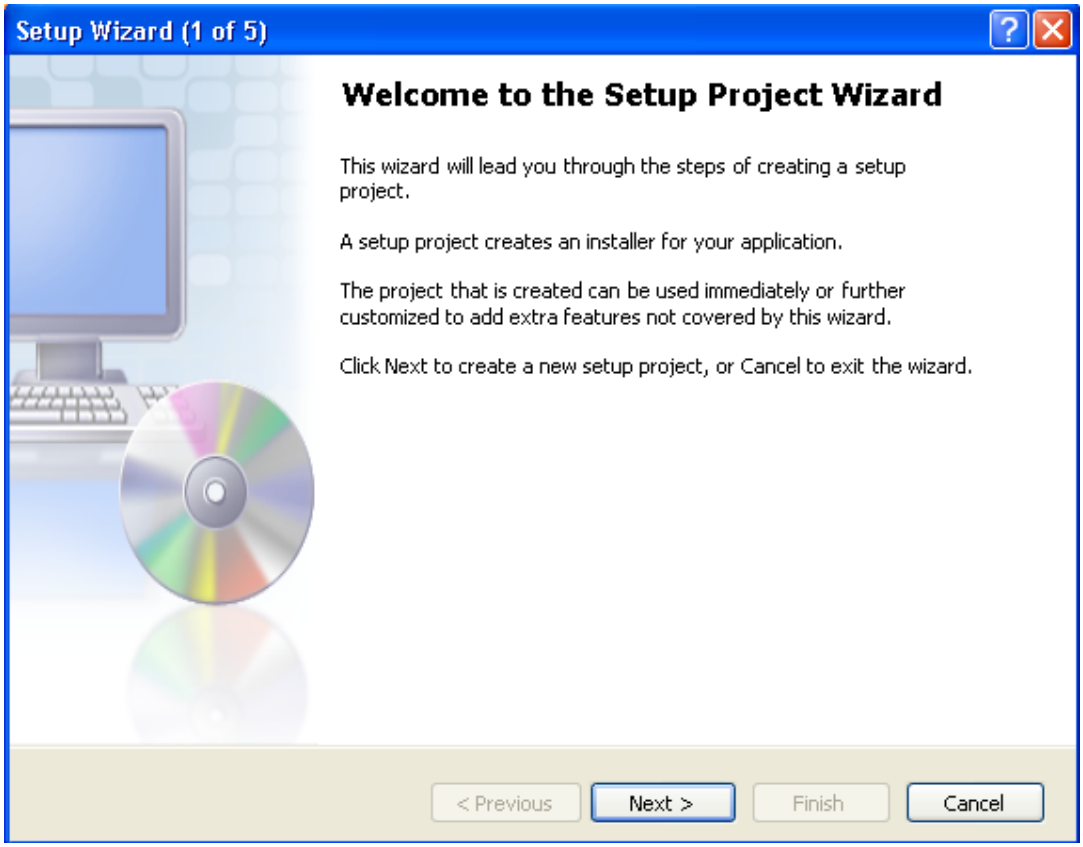
- Sao chép bài tập Lukyseven chúng ta đã hoàn thiện trong chương 10 sang thư mục Bai tap của chương 14.
- Mở nó bằng Visual Studio và chọn File | New | Project từ menu File.



- Chọn Other Project Types và chọn Setup and Deployment. Có 6 mẫu phía bên phải khung cho phép lựa chọn. Mẫu *CAB project* cho phép đóng gói chương trình thành nhiều gói .cab. Mẫu *Merge Module Project* cho phép đóng gói chung các dự án khác nhau (tạo file .msm có thể trộn tiếp vào các gói cài đặt khác). Mẫu *Setup Project* tạo bộ đóng gói cài đặt bởi Windows Installer. Mẫu *Web Installer* cài đặt

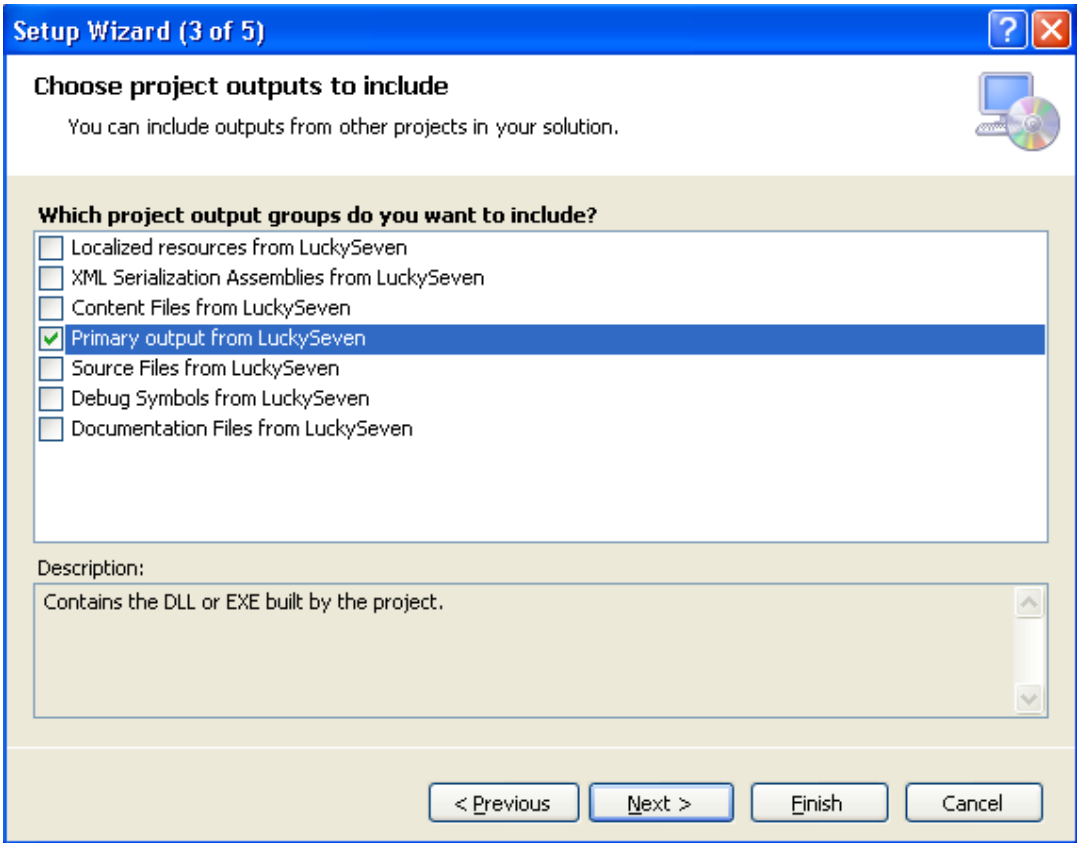
chương trình từ một Webserver…Có lẽ chúng ta sẽ chọn mẫu *Setup Winzard*, mẫu này đóng gói chương trình thông qua các câu hỏi và yêu cầu thực hiện một số bước cấu hình đơn giản.

- Nhấn vào biểu tượng Setup Wizard và gõ vào chuỗi LuckySeven_Version1.0 và chọn đường dẫn đến giải pháp LuckySeven.
- Chọn *Add To Solution* từ danh sách Solution để kết hợp hai dự án cùng lúc vào giải pháp hiện hành.
- Nhấn OK để làm xuất hiện trình Setup Winzard.

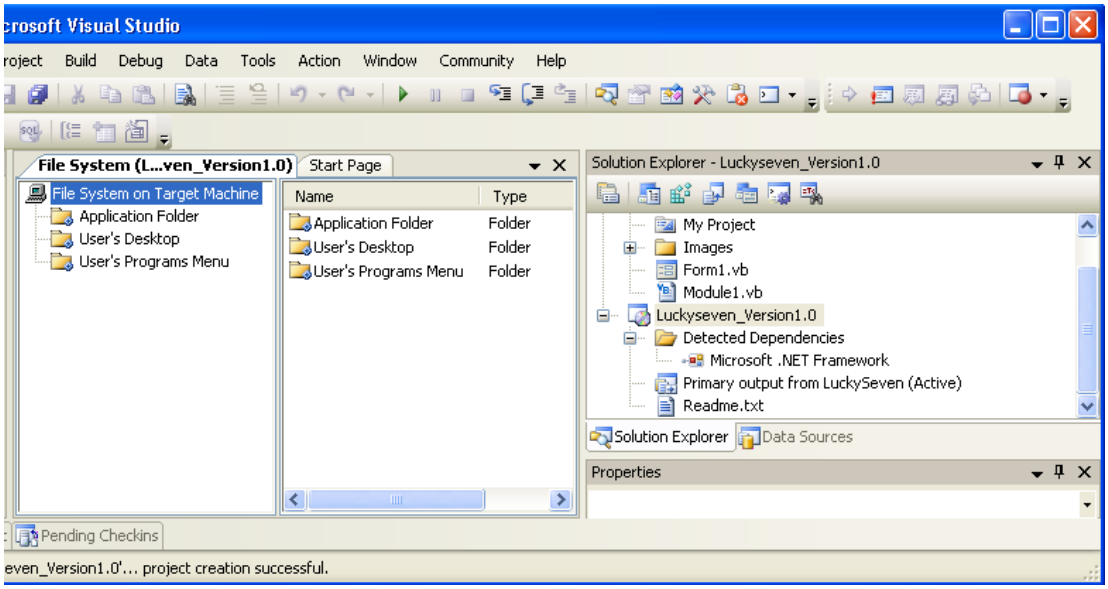


3.1. Sử dụng trình Setup Winzard

- 1. Nhấn nút Next ở màn hình giới thiệu
- 2. Hộp thoại *Choose a Project Type* xuất hiện. Hộp thoại này cho phép ta kiểm soát cách đóng gói và phân phối dự án chương trình. Trong bài tập này ta để mặc định là lựa chọn đầu tiên "Creat a setup for a windows application"
- 3. Nhấn nút Next. Hộp thoại *Choose Project Output To Include* hiển thị cho phép chỉ định file kèm theo trình cài đặt và các thông tin về chương trình.
- 4. Nhấn chọn Primary Output như hình:



- 5. Nhấn Next. Hộp thoại *Choose Files To Include* hiển thị cho phép chọn các files dữ liệu, file hướng dẫn…
- 6. Nhấn nút Add và chọn file readme.txt trong dự án.
- 7. Nhấn Next để hiển thị hộp thoại *Creat Project*. Tại đây các thông tin về dự án đóng gói đã chọn. Bạn có thể nhấn nút Previous để hiệu chỉnh thông tin trước khi hoàn thiện.
- 8. Nhấn nút Finish để tạo dự án Deployment cho chương trình Luckyseven. Lúc này dự án mới đóng gói sẽ được liệt kê trong cửa sổ Solution Explorer. Bộ soạn thảo *File System Editor* cũng xuất hiện. Bạn có thể sử dụng *File System Editor* để thêm vào các file kết xuất, cùng các mục khác để xác định dự án có cần cài thêm những file phục lên máy khách trong quá trình cài đặt hay không. File System Object hiển thị danh sách các folder chuẩn sẽ được sử dụng hay tạo ra khi cài dặt. Bạn có thể tùy biến những folder này và thêm vào những folder đặc biệt khác tùy theo mục đích của chương trình. Bạn cũng có thể yêu cầu File System Editor tạo short cut cho chương trình. Bạn hãy xem qua các file chúng ta đã đính kèm và các file phụ thuộc vào thư viện thực thi .Net Framework trong folder *Detected Dependencies*, nơi chứa file .exe (Primary Output) như hình:



- 9. Tùy biến các lựa chọn đóng gói. Phần này chúng ta sẽ tách riêng trong mục 4.

3.2. Tạo dự án đóng gói sử dụng mẫu Setup Winzard

Tương tự như phần trên, các bạn co thể tham khảo thêm trong các tài liệu khác.

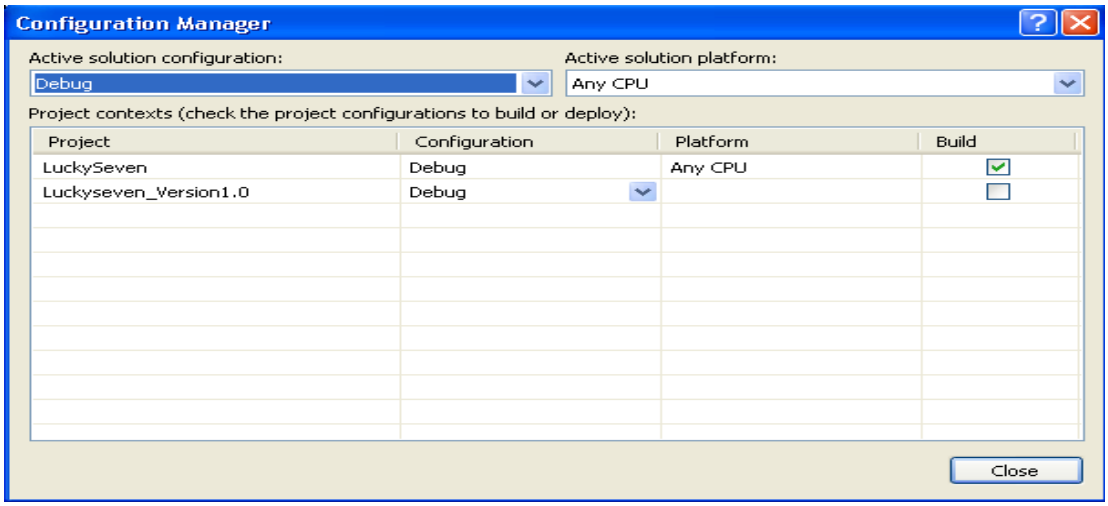
4. Tùy biến các lựa chọn đóng gói

Bây giờ dự án của chúng ta đã sẵn sàng đóng gói. Khi có yêu cầu đóng gói, các file sẽ kết xuất ra thư mục chúng ta đã chọn và lưu trong file .msi (Microsoft Installer). Bạn có thể cài đặt từ file này.

Tuy nhiên chúng ta sẽ tạo thêm một số tùy chọn cho chương trình cài đặt trước khi yêu cầu đóng gói như tạo short cut, tạo tên công ty, phiên bản chương trình…

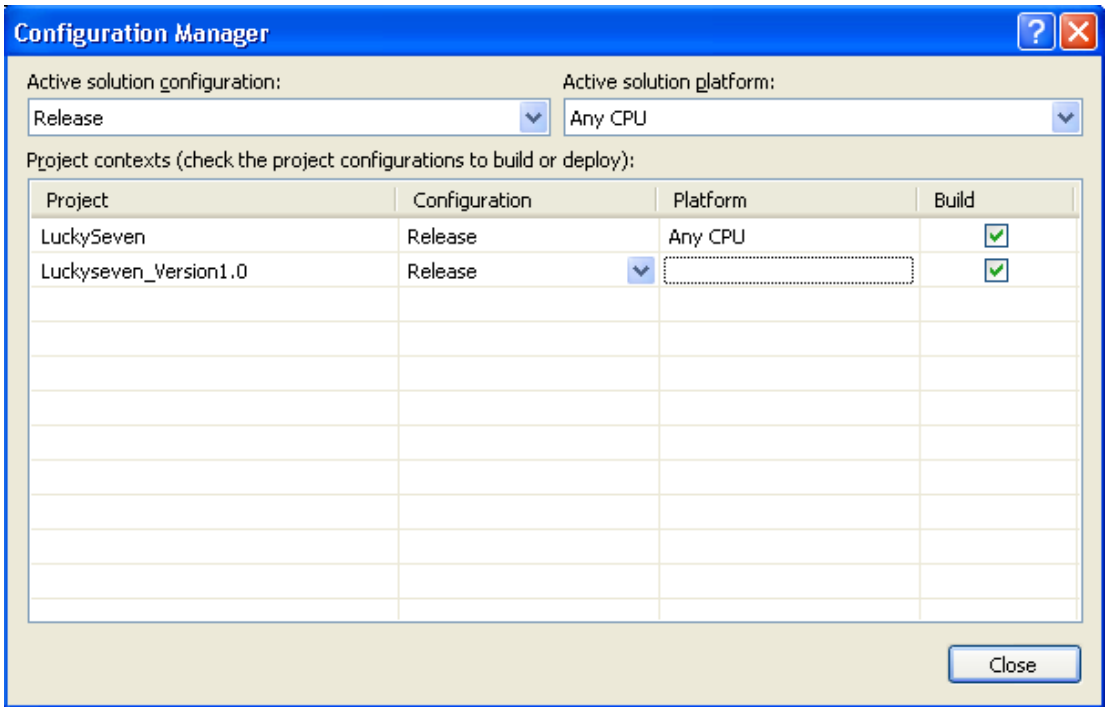
4.1. Cấu hình các thiết lập

- 1. Chọn Build | Configuration Manager từ menu làm xuất hiện hộp thoại sau:



Hộp thoại này cho phép hiển thị chế độ biên dịch hiện hành cho các dự án đang có trong Solution. Lúc này chúng ta nên chọn là Release – biên dịch tối ưu thay cho Debug.

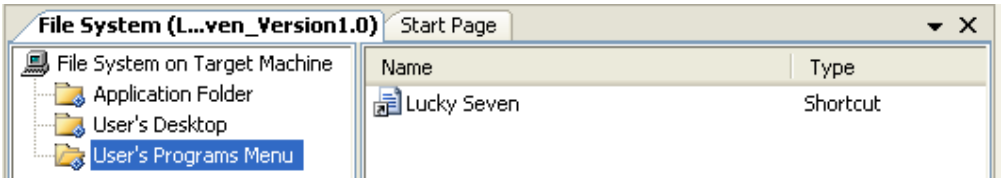
- 2. Nhấn chọn danh sách Active Solution Configuration và chọn là Release
- 3. Nhấn chọn danh sách Configuration cho cả hai dự án LuckySeven và LuckySeven_Version1.0 và chọn Release.
- 4. Đánh dấu chọn mục Buil cho cả hai dự án.
- 5. Nhấn nút Close.



Tiếp theo chúng ta sử dụng *File System Editor* để yêu cầu trình cài đặt tạo short cut cho chương trình sau khi cài đặt thành công.

4.2. Tạo shortcut cho ứng dụng cài đặt

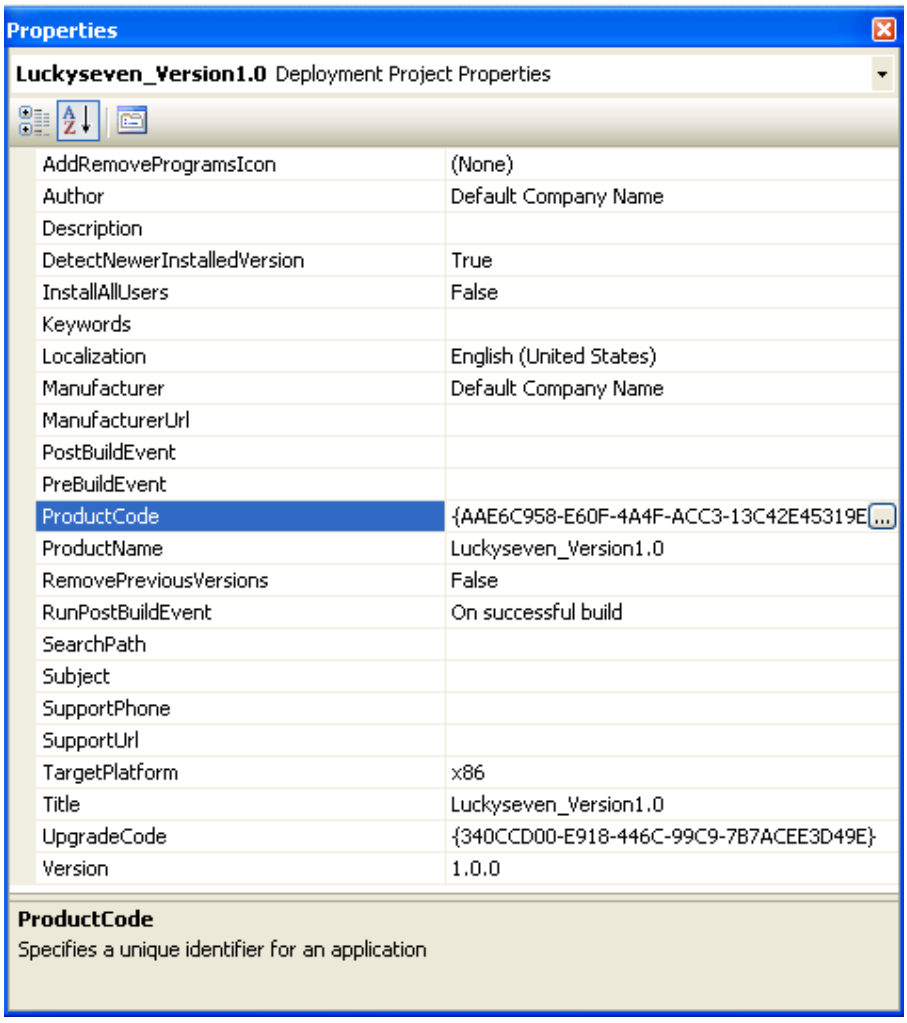
- 1. Chọn *Application Folder* trong khung trái cửa sổ *File System Editor*. Nếu File System Editor chưa xuất hiện bạn có thể cho nó xuất hiện bằng cách chọn dự án trong Solution Explorer và chọn *View* | *Editor* | *File System*.
- 2. Bên khung phải bạn nhấp chuột phải vào mục Primary Output From LuckySeven và chọn Creat Shortcut To Primary Output From Luckyseven từ menu.
- 3. Đổi tên thành Lucky Seven và ấn Enter.
- 4. Kéo Shortcut Lucky Seven vào trong thư mục User's Programs Menu Folder bên khung trái như hình:



Tiếp theo chúng ta sẽ tìm hiểu thêm về cách tùy biến thông tin về tên công ty và phiên bản chương trình.

4.3. Thiết lập tên công ty và phiên bản chương trình

- 1. Chọn dự án Luckyseven_Version1.0 từ cửa sổ Solution Explorer.
- 2. Mở cửa sổ Properies của nó.



Thuộc tính Author thường dùng đặt tên cho tác giả hay nhà sản xuất. Tên này cũng được dùng làm thư mục cho chương trình đặt trong Program Files. Khi chương trình Setup thực thi, thông tin về tác giả sẽ được đặt trong trường Contact của hộp thoại Support Info, hộp thoại này bạn có thể truy xuất sau đó thông qua mục Add/Remove Programs trong Control Panel.

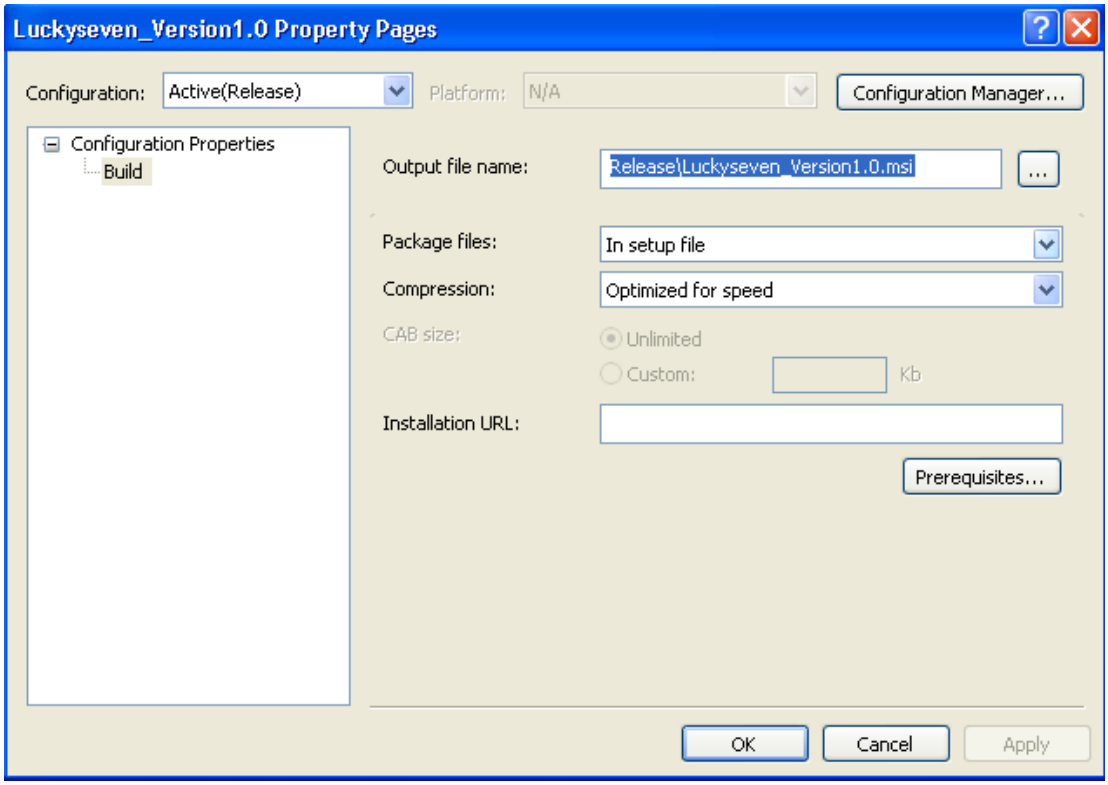
Thuộc tính Tiile chứa tên chương trình setup, thuộc tính Version chứa số hiệu phiên bản cho chương trình.

- 3. Thay đổi thuộc tính Author thành Luckystar Programming
- 4. Đổi Version thành 1.0
- 5. Ấn Yes khi có hộp thoại xuất hiện
- 6. Xem lại các thuộc tính khác nữa trước khi đóng gói thực sự.

Bây giờ hãy thử mở hộp thoại Property Pages bằng cách R-Click vào dự án chọn Properties xem các thông tin thiết lập có đúng hay không.

4.4. Đặt các thuộc tính cho gói ứng dụng

Bạn làm xuất hiện cửa sổ Property Pages như hình:



Hộp thoại này cho phép quyết định xem lại những gì đã tùy chọn trong trình Setup Winzard và tùy biến số ít các thiết lập bổ sung không có trong winzard.

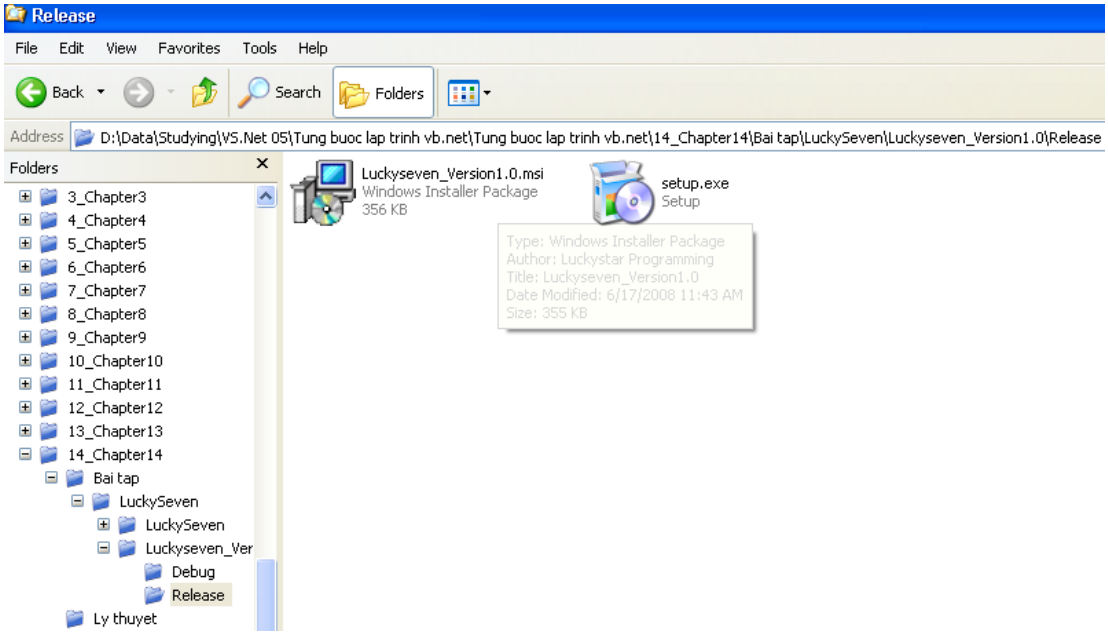
- Output File name cho phép đặt tên cho file cài đặt.
- Danh sách Package files: có 3 tùy chọn là As Loose Uncompress Files – tạo ra các file không nén trong cùng thư mục với file .msi; mục In Setup File lúc này chọn mặc định do lựa chọn của chúng ta trong quá trình dùng Setup Winzard; mục In Cabinet Files – tạo ra một hay nhiều file .CAB là các file chứa ứng dụng của bạn trong đó.

- Chọn mục In Cabinet Files, bạn có thể tùy chỉnh các thông số tương ứng
- Nhấn chuột trở lại và lại chọn In Setup File. Chúng ta sẽ tạo ra một file cài đặt đơn chứa tất cả các file yêu cầu của chương trình.
- Nhấn OK để lưu lại các thay đổi của bạn trong hộp thoại Property Pages.

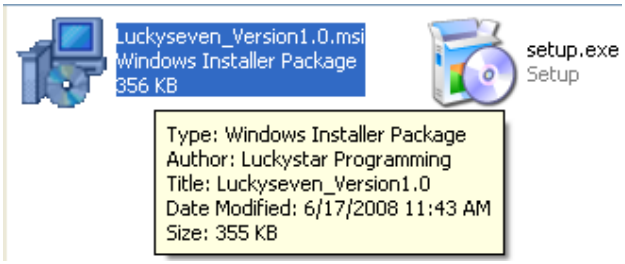
5. Biên dịch và đóng gói dự án – kiểm tra việc cài đặt

Bây giờ dự án đã sẵn sàng để biên dịch, các bước thực hiện như sau:

- Biên dịch bằng lệnh Build | Build Solution. Quá trình diễn ra hơi lâu. Bạn để ý thanh theo dõi tiến trình  xem quá trình biên dịch dự án. Nếu biên dịch thành công thì một thông báo hiện ở cuối góc trái màn hình Build succeeded.
- Chạy chương trình Setup để cài đặt ứng dụng. Bạn mở thư mục chứa dự án đã chọn và tìm file cài đặt:



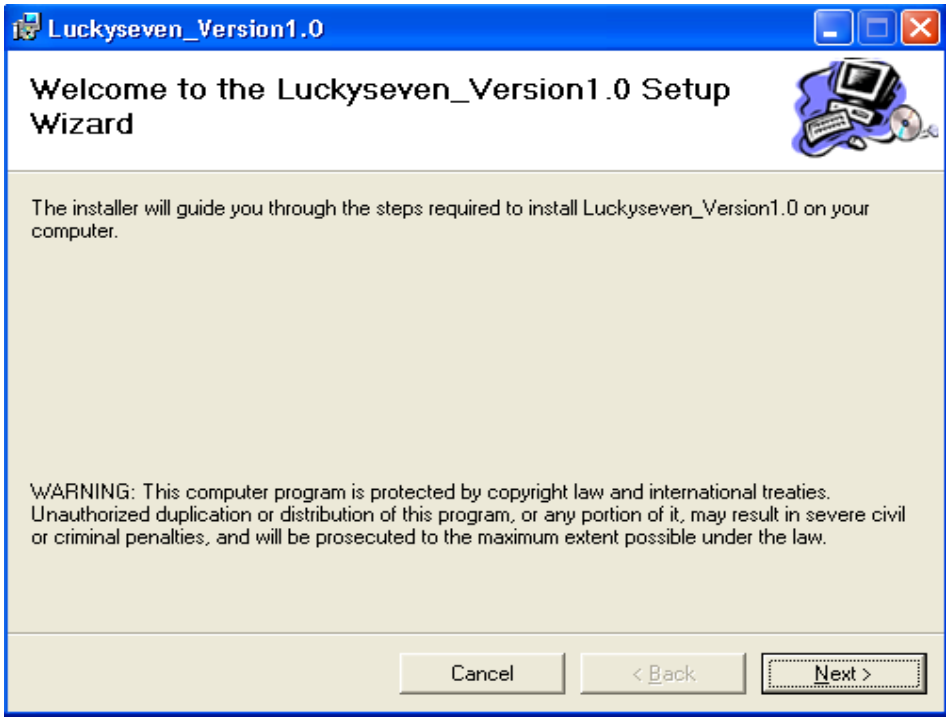
Khi bạn chọn file LuckySeven_Version1.0 thì thông tin cấu hình trong phần trước sẽ hiện lên như thế này:



Kiểm tra quá trình cài đặt xem các file có được cài đủ và đúng vị trí hay không:

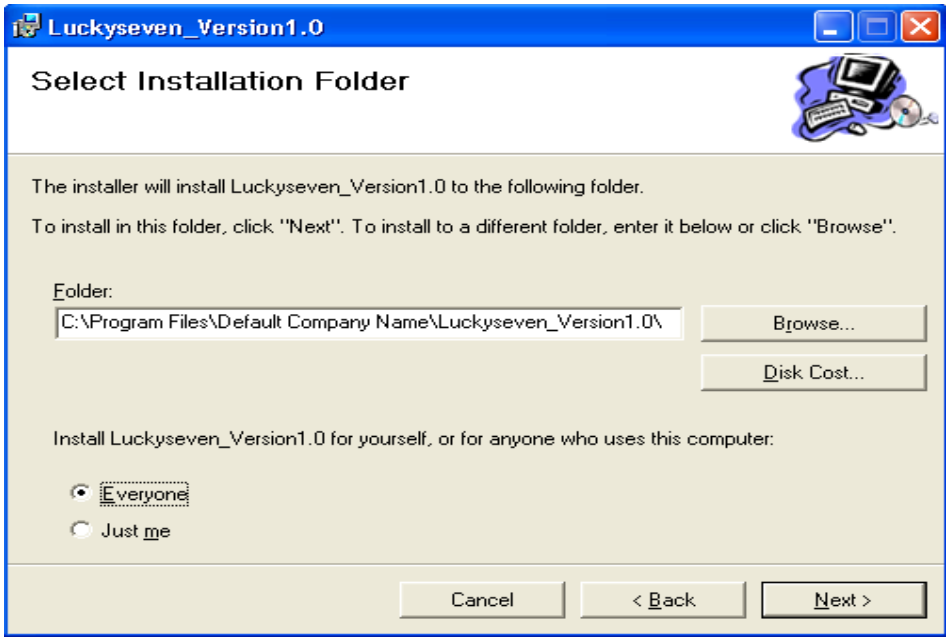
5.1. Chạy chương trình cài đặt Setup

Nhắp đôi vào file setup.exe để khởi động chương trình cài đặt, sau một lúc màn hình chào mừng hiện ra như sau:

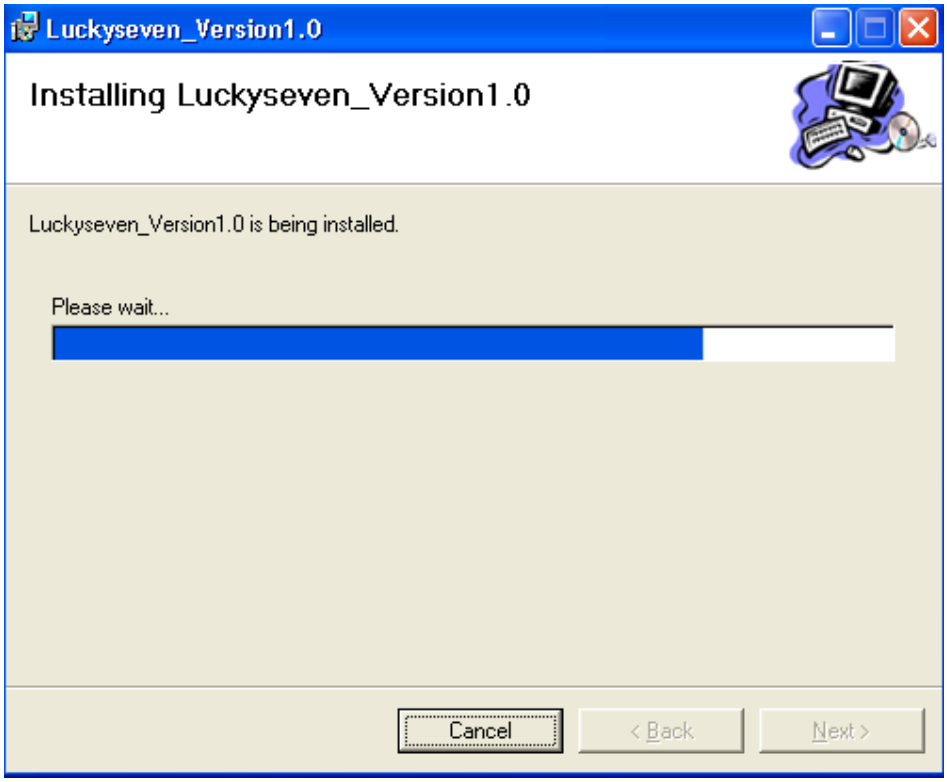


Nhấn Next để tiếp tục cài đặt. Hộp thoại Select Installation Folder yêu cầu nhập thông tin về thư mục cài đặt và các tùy chọn bổ sung.

Để mặc định các tùy chọn trừ tùy chọn người sử dụng thì chọn Everyone. Nhấn Next



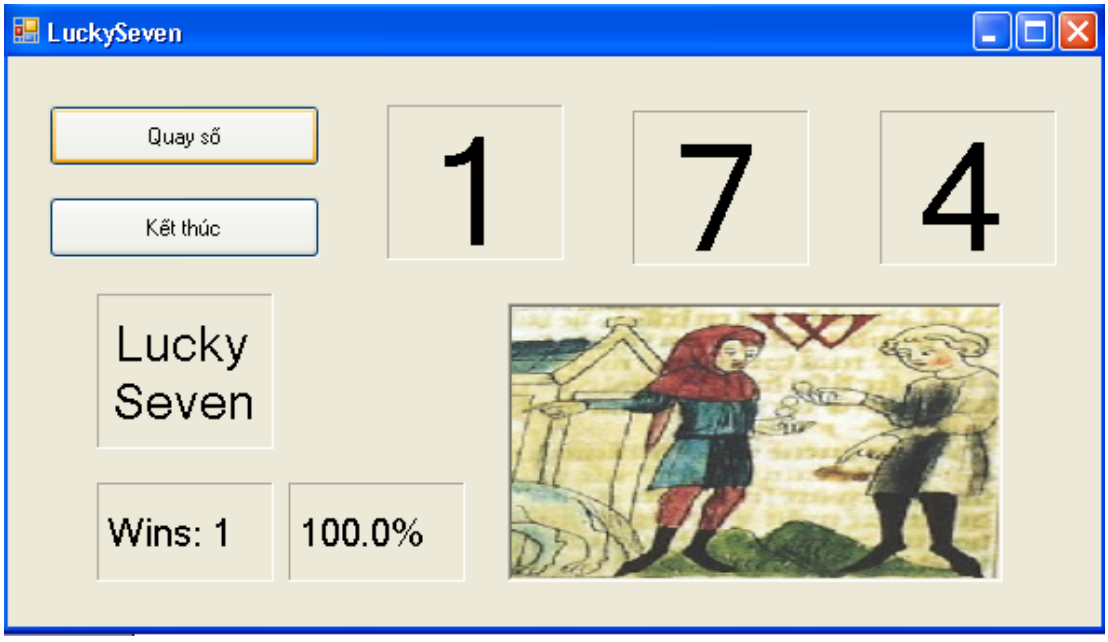
Nhấn nút Next để bắt đầu cài đặt.



Nhấn Close để hoàn tất cài dặt.

5.2. Chạy chương trình LuckySeven sau khi cài đặt

Chọn Start | Program và chọn Shortcut LuckySeven. Chương trình khởi động thành công. Bạn chạy thử chương trình xem có gặp lỗi gì không.

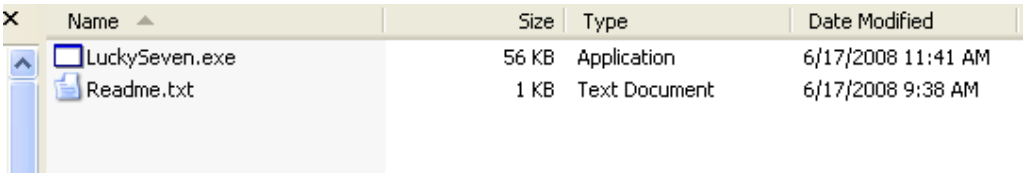


Như vậy là chúng ta đã đóng gói thành công một ứng dụng VB.

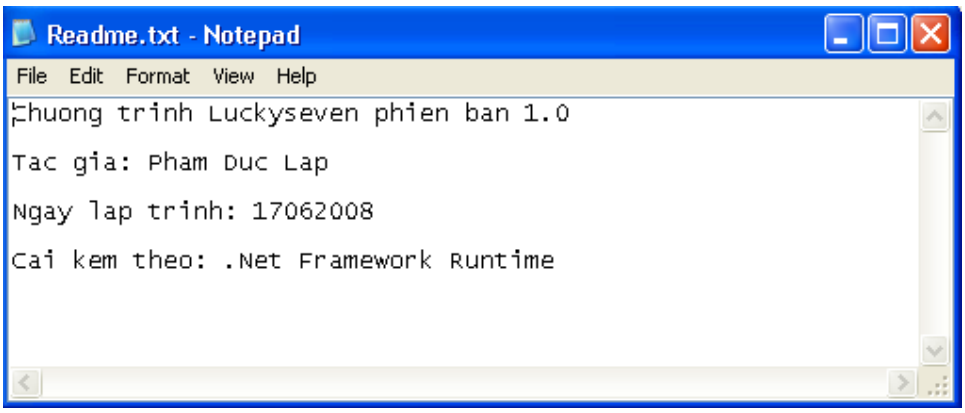
6. Tìm hiểu các file Setup và gỡ chương trình

6.1. Kiểm tra file cài đặt

Bạn mở thư mục chứa phần cài đặt trong Program Files và xem các file mà chương trình đã cài vào. Như hình ta có hai file là Luckyseven.exe và Readme.txt:

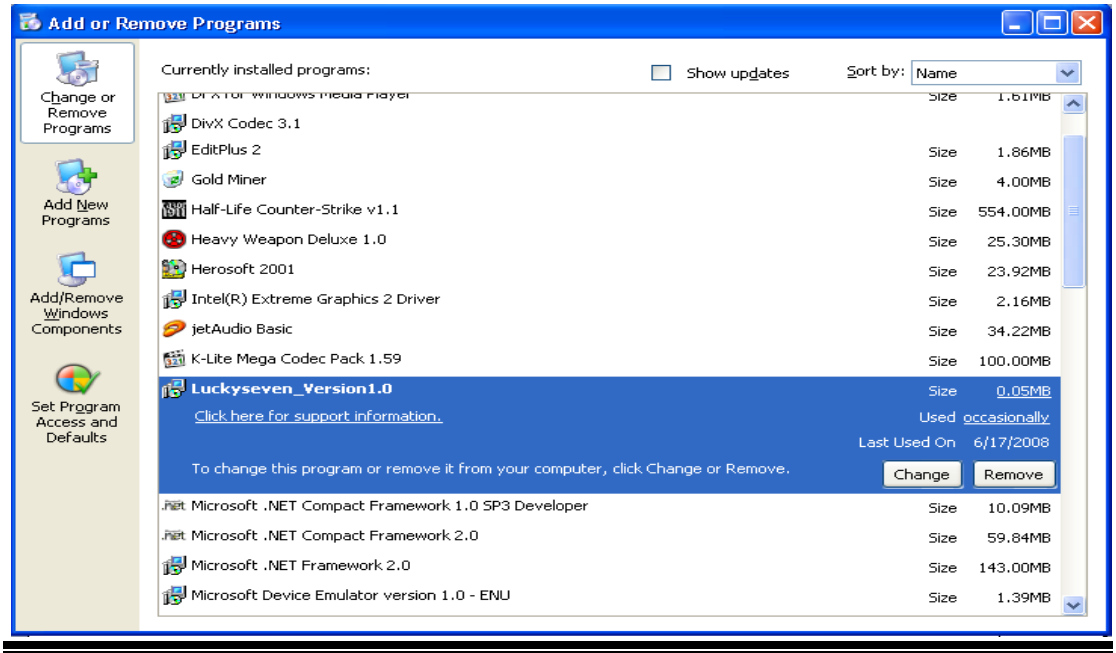


Bạn có thể mở xem nội dung file Readme.txt xem nội dung:

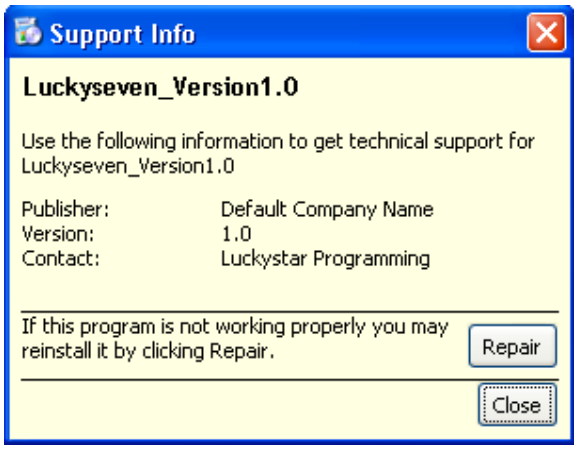


6.2. Tháo gỡ chương trình

Bạn tháo gỡ chương trình như mọi chương trình khác.



Có thể xem thông tin bằng cách click vào nút Support Info để xem thông tin về chương trình:



Đóng thông tin lại và click vào Remove và chọn Yes khi có thông báo hiện ra. Vậy là ta đã tạo dự án phân phối, cài đặt và tháo gỡ thành công một chương trình VB.NET.

7. Tổng kết chương 14

Bạn làm bảng tổng kết những gì đã học trong chương 14. Bạn có thể tạo dự án phân phối các chương trình chúng ta đã tạo.

Chương 15:

Quản lý Windows Forms

--------oOo--------

Nội dung thảo luận:

- Thêm vào một form mới cho chương trình
- Thay đổi vị trí của form trên màn hình Windows Desktop
- Thêm một điều khiển vào form khi chương trình đang chạy
- Thay đổi canh lề của các đối tượng trên form
- Chỉ định đối tượng khởi động chương trình

Trong chương này chúng ta sẽ học cách thêm nhiều form vào dự án để xử lý nhập, xuất và hiển thị các thông điệp đặc biệt. Ta cũng sử dụng thuộc tính DesktopBounds để định vị trí thay đổi kích thước form, thêm vào các thành phần điều khiển khi chương trình đang chạy…

Chú ý:

- Trong VB.NET muốn đặt thuộc tính cho form thứ hai trong dự án cần phải có tham chiếu đến thể hiện (instance) của biến form đó
- Có thể đặt và định lại vị trí, kích thước form lúc chương trình đang chạy bằng cách sử dụng cửa sổ Form Layout. Tuy nhiên bạn cũng có thể sử dụng thuộc tính DesktopBound mới do VB.NET không còn hỗ trợ Form Layout.
- Thuộc tính mới Anchor cho phép xác định kích thước giới hạn tối đa và tối thiểu mà người dùng được phép thay đổi lên form. Thuộc tính Dock cho phép Form hay đối tượng có thể neo vào một cạnh cửa sổ hay form khác.
- Trong VB.NET form MDI cha chỉ là một form bình thường có thuộc tính IsMdiContainer đặt là TRUE. Các form con có thuộc tính MdiParent trỏ đến tên của form MDI cha.

1. Thêm một Form mới vào chương trình

Ta có thể thêm rất nhiều form vào chương trình VB.NET. Mỗi form thêm vào được coi là một đối tượng kế thừa từ lớp *System.Windows.Forms.Form*. Các form thêm vào có thứ tự lần lượt và tên tương ứng là Form1.vb, Form2.vb,…Bạn có thể thay đổi tên mặc định bằng cách chỉ định tên lúc Add NewItem hay tại cửa sổ Solution Explorer.

Cách sử dụng form:

Bạn có thể cho tất cả các form trong chương trình hiển thị cùng lúc hay chỉ hiển thị khi cần thiết. Khi cho hiển thị lớn hơn một form thì bạn có thể kiểm soát thứ tự form hay cho người dùng hoán chuyển giữa các form.

2. Làm việc với các dự án có nhiều form

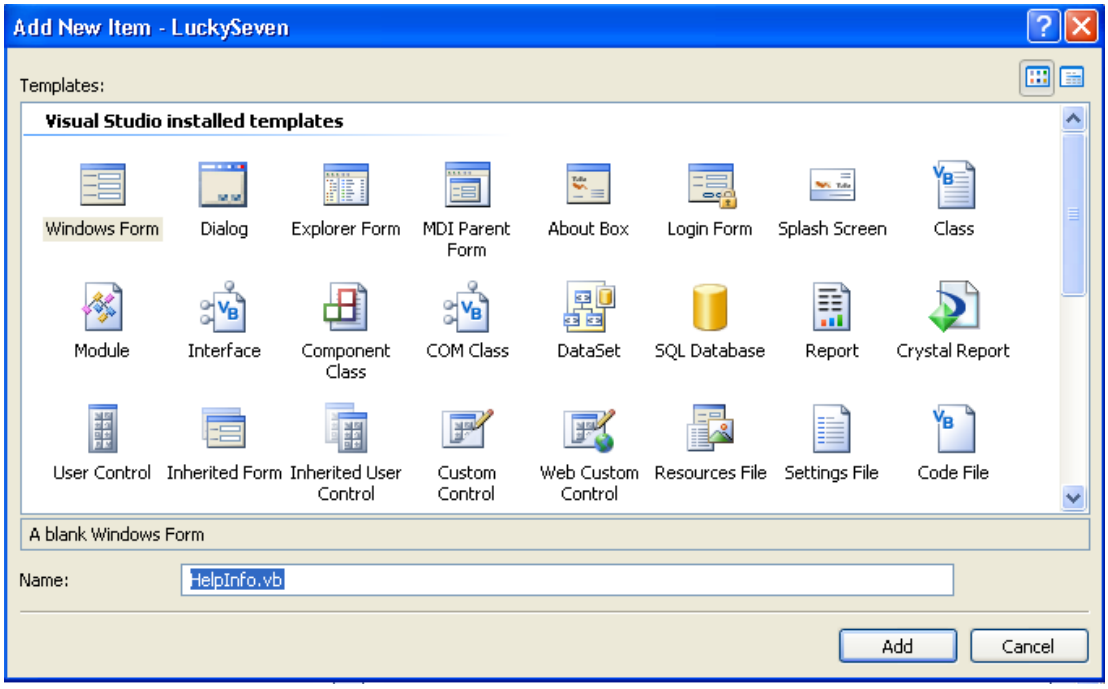
Bài tập sau đây chúng ta sẽ sử dụng một dự án với hai form. Bạn sao chép bài Luckyseven trong chương 10 vào thư mục bài tập của chương 15. Chúng ta sẽ thêm một form thể hiện trợ giúp cho chương trình.

2.1. Thêm form vào dự án

Bạn khởi động giải pháp Luckyseven ta vừa sao chép.

Nhắp đôi vào form1.vb trong cửa sổ Solution Explorer để hiển thị form chính.

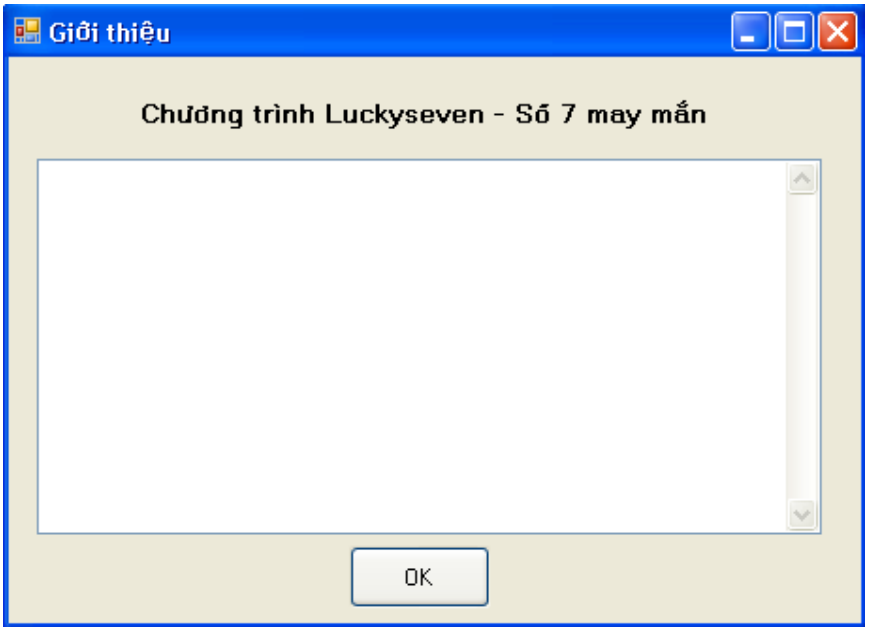
Bạn R-Click vào dự án Luckyseven và chọn Add | New Item. Bạn chọn thêm vào một Windows Form và gõ tên là HelpInfo.vb:



Thêm một số điều khiển vào trong form mới này:

- Thêm vào một nhãn đặt thuộc tính text là "Chương trình Luckyseven – Số 7 may mắn"
- Thêm một TextBox ngay dưới nhãn, thuộc tính MultiLine là True, Scrollbar là Both
- Thêm một nút Button1, thuộc tính Text là OK

Giao diện như hình:



Viết mã:

Form này ta sẽ thể hiện nội dung file Readme.txt trong chương trước chúng ta đã biết. Trước hết ta dùng lớp StreamReader để đọc thông tin của file text và gán cho thuộc tính Text của TextBox1.

Trước hết, khai báo sử dụng lớp này ở đầu form:

```
Imports System.IO
```

Sau đó tạo sự kiện form `HelpInfo_Load` bằng cách nhấp đôi chuột vào form hay chọn từ danh sách thả xuống như đã biết. Chúng ta nhập đoạn mã sau:

```
Dim StreamToDisplay As StreamReader
StreamToDisplay = New StreamReader _
("D:\Data\Studying\VS.Net 05\Tung buoc lap trinh vb.net\" & _
"Tung buoc lap trinh vb.net\15_Chapter15\Bai tap\LuckySeven\" & _
"LuckySeven\Readme.txt")
TextBox1.Text = StreamToDisplay.ReadToEnd
StreamToDisplay.Close()
```

Việc dùng lớp StreamReader để điền nội dung một file văn bản vào textbox chúng ta đã biết trong chương học về xử lý file text và chuỗi. Ở đây thay vì gõ đường dẫn của file Readme.txt chúng ta có thể kéo thả nó từ trong dự án của mình. Để kéo thả thì file đó phải hiện lên trong cửa sổ Solution Explorer. Muốn nó hiện lên thì bạn có thể chép nó vào thư mục chứa dự án, trở về cửa sổ Solution Explorer nhấp vào nút Refresh  hay copy trực tiếp vào cửa sổ Solution Explorer.

Tạo thủ tục `Button1_Click` để người dùng click vào nút OK thì đóng form trợ giúp:

```
Me.DialogResult = Windows.Forms.DialogResult.OK
```

Bây giờ làm thế nào để hiện thị form thứ hai này vì dự án của chúng ta có tới hai form?

2.2. Hiển thị Form thứ hai sử dụng thủ tục sự kiện

Để làm được điều này ta sẽ thêm một nút ở form thứ nhất Form1.vb và thêm thủ tục triệu gọi form thứ hai.

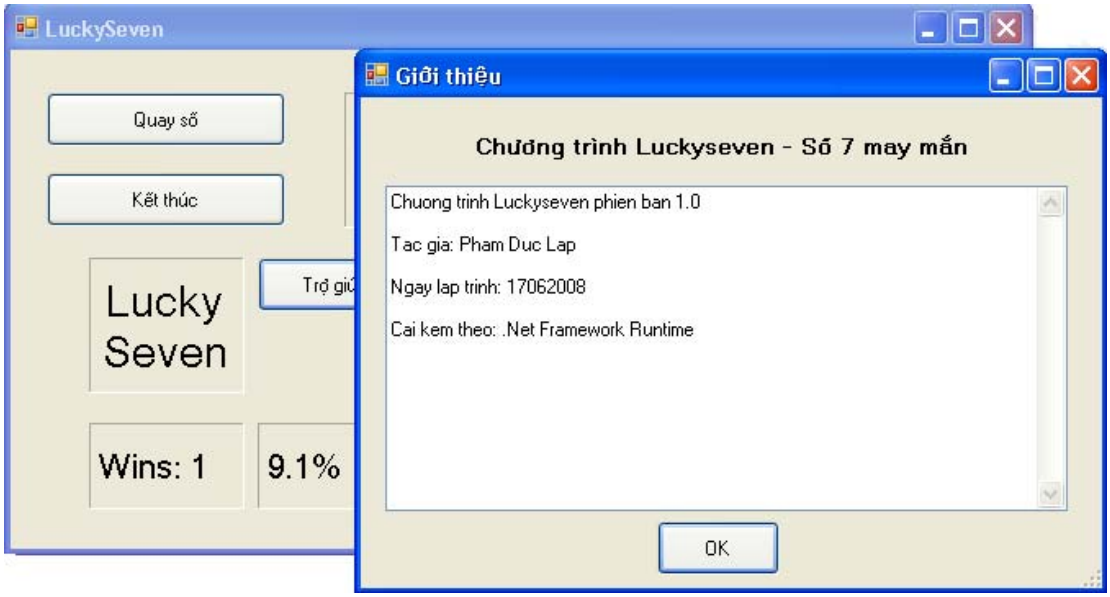
Bạn mở form1.vb và thêm vào nút nhấn đặt thuộc tính Text cho nó là "Trợ giúp", thuộc tính name là btnHelp.

Tạo thủ tục `btnHelp_Click` bằng cách double click vào nút Trợ giúp và nhập đoạn mã sau:

```
Dim frmtrogiup As New HelpInfo()
frmtrogiup.ShowDialog()
```

Hai phát biểu này cho phép triệu gọi form thứ hai. Như đã nói trước, để tham chiếu đến form thứ hai bạn cần tường minh form đó. Ở đây chúng ta khai báo biến *frmtrogiup* có kiểu *HelpInfo* nhờ phát biểu *New HelpInfo()*. Sau khi đã khởi tạo chúng ta có thể hiển thị form bằng cách gọi đến phương thức ShowDialog(). Nếu ở đây bạn gọi form này bằng phương thức Show() thì trong thủ tục Button1_Click của form HelpInfo bạn cần gọi phương thức *Me.Close* thay cho phương thức `DialogResult.OK` chúng ta đã dùng.

Bạn chạy chương trình bằng cách ấn F5 và ấn nút Trợ giúp để hiển thị form thứ hai:



3. Định vị form trên màn hình Desktop

Bạn có thể định vị form trên màn hình desktop khi nó xuất hiện bằng thuộc tính DesktopBounds. Nó cho phép định vị trí của form với góc phải dưới và góc trái trên. Đơn vị tính là pixel.

Ngoài ra bạn còn có thể sử dụng thuộc tính StartPosition với các đặc điểm: Manual – bằng tay, CenterScreen – giữa màn hình, WindowsDefaultLocation – vị trí mặc định, WindowsDefaultBound – kích thước mặc định.

3.1. Sử dụng thuộc tính StartPosition

Bây giờ chúng ta sẽ dùng thuộc tính StartPosition và DestopBounds để định vị trí form qua bài tập *MyDesktopBound* sau đây.

Bạn tạo mới giải pháp và thêm vào một dự án cùng tên MyDesktopBound và làm như sau:

- Mở properties của form1.vb.
- Thay thuộc tính StartPosition thành CenterScreen và chạy thử. Form sẽ xuất hiện ở chính giữa màn hình.
- Đóng chương trình, đặt thuộc tính StarPosition thành Manual. Với thuộc tính này bạn cần đặt lại thuộc tính Location, ta đặt thuộc tính này là 100, 50.
- Chạy thử chương trình. Form sẽ hiển thị theo tọa độ ta đã đặt.

3.2. Sử dụng thuộc tính DestopBounds

Đặt thêm nút nhấn lên form1, đặt text là "Tạo form mới".

Tạo thủ tục `Button1_Click` và nhập mã như sau:

```
'Tạo form thứ hai có tên Form2
Dim form2 As New Form()
'Định nghĩa thuộc tính Text và đường viền cho form
form2.Text = "Form mới"
form2.FormBorderStyle = Windows.Forms.FormBorderStyle.FixedDialog
'Chỉ định vị trí của form được đặt thủ công
form2.StartPosition = FormStartPosition.Manual
'Khai báo cấu trúc Rectangle nắm giữ kích thước mới
'Góc trái trên (200,100)
'Chiều dài và cao (300,250)
Dim rectangle_form2 As New Rectangle(200, 100, 300, 250)
'Định kích thước của form sử dụng đối tượng rectangle trên
form2.DesktopBounds = rectangle_form2
'Hiển thị form
form2.ShowDialog()
```

Bạn chạy chương trình này bằng cách ấn F5. Nhấn vào nút "tạo form mới" để tạo form thứ hai. Form này có vị trí như ta đã định. Form này không cho phép bạn kéo lại kích thước như các form trước đây do ta đã đặt thuộc tính `FormBorderStyle` của form thành `FixedDialog`.

3.3. Phóng to, thu nhỏ và khôi phục lại cửa sổ chương trình

Ngoài ra bạn cũng có thể phóng to, thu nhỏ hay khôi phục lại vị trí mặc định của form. Bạn có thể thực hiện điều này khi thiết kế hay khi chương trình đang chạy.

Để làm điều này trước hết bạn cần cho hiện hai nút Maximize và minimize ở góc phải trên chương trình bằng hai thuộc tính:

```
MaximizeBox = True
MinimizeBox = True
```

Tiếp đến trong mã chương trình hay trong cửa sổ thuộc tính bạn đặt thuộc tính WindowState như sau:

```
WindowState = FormWindowState.Minimized
```

Nếu bạn muốn kiểm soát kích thước phóng to, thu nhỏ cho phép của form bạn đặt thuộc tính `MinimumSize, MaximumSize`. Hai thuộc tính này có kiểu cấu trúc Size giống như cấu trúc Rectangle, ví dụ:

```
Dim Formsize As New Size(400, 300)
MaximumSize = Formsize
```

4. Thêm vào các điều khiển lúc form đang chạy

Ta thường đưa các điều khiển trên Toolbox khi thiết kế form. Bạn cũng có thể đưa chúng vào trong form khi chương trình đang chạy – tạo điều khiển động. Quy trình để đưa như sau:

Khai báo biến đối tượng có kiểu lớp của phần tử giao diện mà bạn muốn đưa vào, ví dụ:

```
Dim btnOK As New Button()
```

Thiết lập thuộc tính cho các nút nhấn sau khi đã khai báo như trên:

```
'Đặt thuộc tính cho nút nhấn
btnOK.Text = "OK"
btnOK.Location = New Point(110, 100)
```

Đưa đối tượng vào form. Để thực hiện điều này, bạn đưa các đối tượng vào tập hợp Controls của form bằng phương thức Add:

```
form2.Controls.Add(btnOK)
```

Bài tập *MyAddControls* sau đây sẽ minh họa cụ thể hơn:

Bạn tạo một giải pháp mới và thêm vào một dự án có cùng tên như trên. Thiết kế form1 có một nút nhấn với thuộc tính text là "Hiển thị ngày". Khi người dùng click vào đây thì một form mới sẽ được tạo ra. Khi form này tạo ra thì đồng thời mã chương trình sẽ tạo hai điều khiển là nhãn lblNgay ghi ngày hiện hành và nút nhấn btnOK để đóng form thứ hai này lại. Bạn tạo thủ tục Button1_click và nhập mã như sau:

```
'Khai báo form và các đối tượng điều khiển
Dim form2 As New Form()
Dim lblNgay As New Label()
Dim btnOK As New Button()

'Đặt thuộc tính nhãn
lblNgay.Text = "Hôm nay là: " & DateString
lblNgay.Size = New Size(150, 50)
lblNgay.Location = New Point(80, 50)
```

```
'Đặt thuộc tính cho nút nhấn
btnOK.Text = "OK"
btnOK.Location = New Point(110, 100)

'Đặt thuộc tính cho form mới
form2.Text = "Ngày hiện hành"
form2.CancelButton = btnOK
form2.StartPosition = FormStartPosition.CenterScreen

'Đưa các đối tượng mới vào tập hợp Controls
form2.Controls.Add(lblNgay)
form2.Controls.Add(btnOK)


'Gọi hiển thị form2
form2.ShowDialog()
```

Chạy chương trình và chúng ta sẽ thấy hiệu quả.

5. Tổ chức sắp xếp các điều khiển trên form

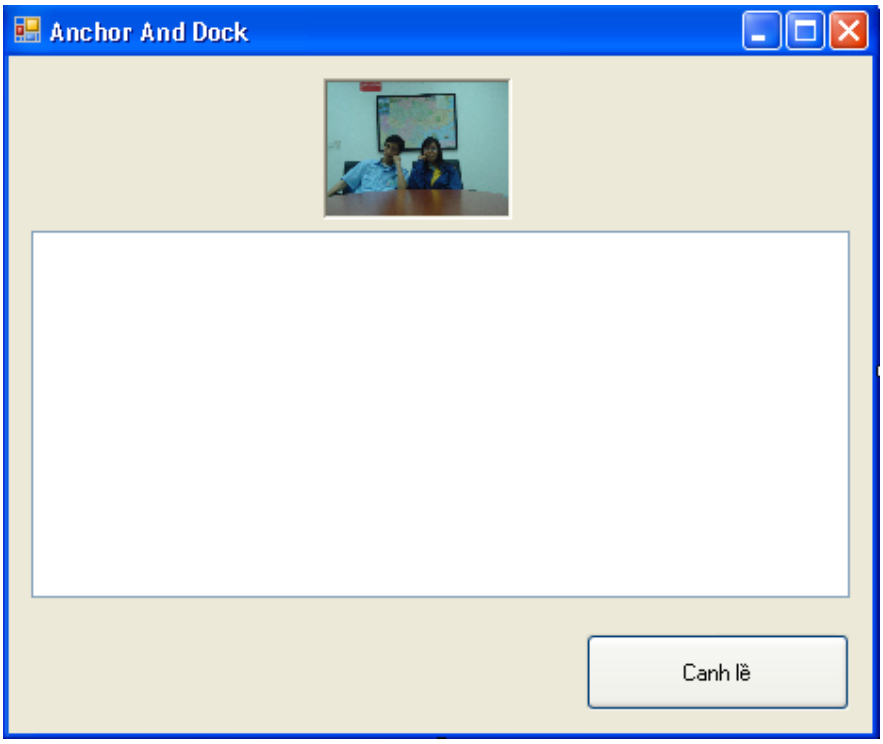
Việc thêm các điều khiển bằng mã chương trình gặp khó khăn trong việc căn chỉnh vị trí các đối tượng do không có công cụ nhìn trực quan. Chúng ta chỉ có thể định kích thước và vị trí thông qua hai thuộc tính là Size và Location. Để khắc phục điều này, VB.NET cung cấp một số thuộc tính mới như **Anchor** – định phạm vi ràng buộc tương đối giữa các đối tượng, **Dock** – neo dính đối tượng này vào cạnh một đối tượng khác. Chúng ta sẽ làm quen với hai thuộc tính này trong bài tập **MyAnchorAndDock** sau đây:

Tìm hiểu chương trình:

Chương trình gồm một PictureBox cho hiển thị một ảnh, một TextBox và một nút nhấn. Khi người dùng click vào nút này thì tiến hành định vị các điều khiển trong form.

Thiết kế giao diện:

Giao diện chính của form như hình:



Thuộc tính của các đối tượng:

- PictureBox1: Image – các bạn có thể cho một ảnh bất kỳ nào (dung lượng nhỏ thôi) để hiển thị; sizemode – StretchImage.
- Button1: Text – "Canh lề".

Viết mã:

Bạn tạo thủ tục `Button1_Click` bằng cách double click vào nút "Canh lề" và nhập đoạn mã sau:

```
PictureBox1.Dock = DockStyle.Top
TextBox1.Anchor = AnchorStyles.Bottom Or _
AnchorStyles.Left Or _
AnchorStyles.Right Or AnchorStyles.Top
Button1.Anchor = AnchorStyles.Bottom Or AnchorStyles.Right
```

Chạy chương trình:

Ấn F5 để chạy chương trình. Khi ấn canh lề thì ảnh sẽ được canh lề theo mép trên của form. Bạn có thể kéo form rộng ra theo ý muốn và quan sát. Phóng to form ra thì thấy vị trí các đối tượng trên form cũng không thay đổi vị trí.

6. Chỉ định thủ tục hay đối tượng thực thi chương trình khởi động

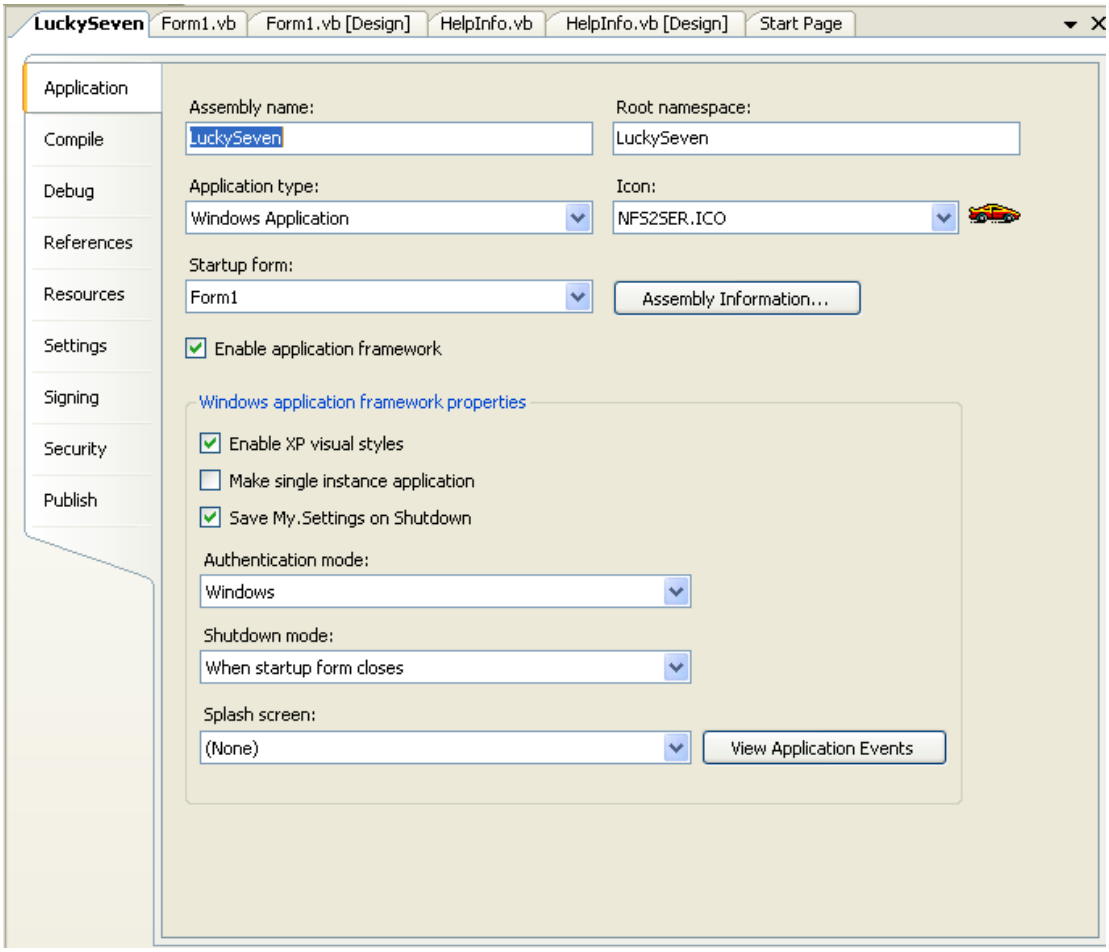
Khi dự án có nhiều form bạn sẽ phải chỉ định xem form nào sẽ khởi động trước form nào.

Bạn có thể làm điều này nhờ hộp thoại Properties của dự án hay yêu cầu VB thực thi thủ tục mang tên Sub Main, trong thủ tục này bạn có trách nhiệm tạo và hiển thị form khác.

6.1. Thay đổi form khởi động

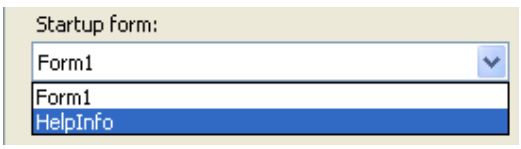
Bạn mở lại Solution Luckyseven của chương này chúng ta vừa thao tác. Ta thấy dự án Luckyseven có hai form là Form1.vb và HelpInfo.vb. Bây giờ chúng ta sẽ chỉ định xem form nào sẽ khởi động trước.

Bạn R-Click vào tên dự án Luckyseven và chọn Properties. Cửa sổ thuộc tính Properties của dự án xuất hiện như hình:

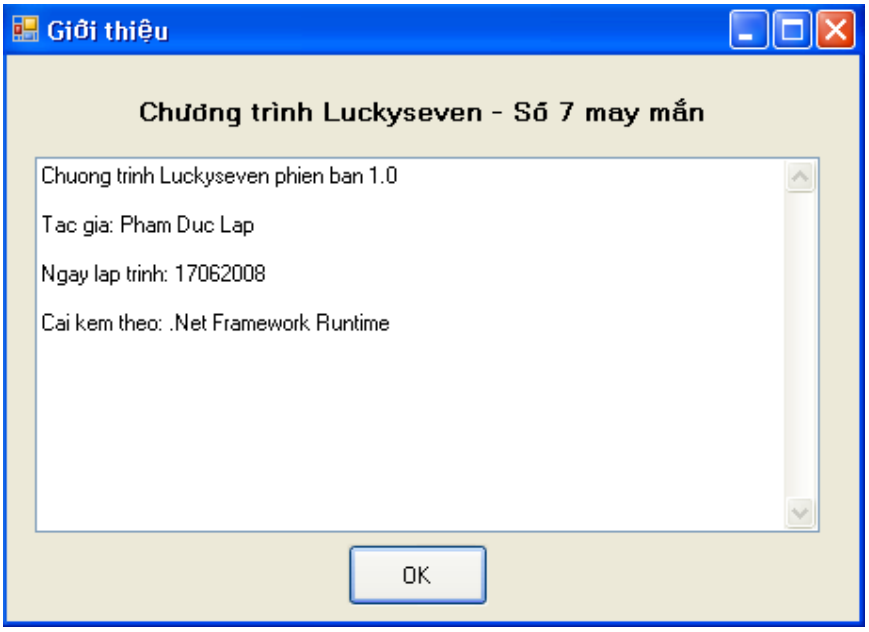


Hộp thoại này cho phép bạn tùy chỉnh lại một số thiết lập cho dự án. Để thay đổi thứ tự form khởi động, bạn dùng combobox Startup Form trong tab Application như hình.

Bạn hãy chọn form HelpInfo thay vì form1 và chạy thử chương trình.



Lúc này form khởi động không phải là form1 mà là form HelpInfo:



Đóng chương trình và chọn lại form1 trong danh sách Startup Form và chạy lại chương trình một lần nữa. Khi đó form1 sẽ khởi động trước tiên.

6.2. Thực thi chương trình từ thủ tục Sub Main

Bây giờ thay vì yêu cầu chương trình hiển thị form HelpIfo trước chúng, ta sẽ yêu cầu chương trình thực thi thủ tục Sub Main. Thủ tục này thường được khai báo trong Module.

Bạn R-Click vào dự án LuckySeven và chọn Add | New Item và thêm vào một module có tên SubMainModule.

Bạn nhập vào khai báo như sau:

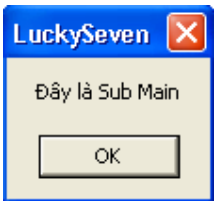
```
Public MyForm1 As New Form1()
Public MyForm2 As New HelpInfo()

Public Sub Main()
    MsgBox("Đây là Sub Main")
    'Có thể đặt thêm các mã khởi tạo tại đây
    'trước khi hiển thị Form chính
    '...

    'Hiển thị Form chính
    MyForm1.ShowDialog()
End Sub
```

Khi bạn thêm thủ tục Sub Main vào dự án, bạn phải đặt nó trong module và khai báo thủ tục này có phạm vi toàn cục Public.

Bây giờ chúng ta cần cho chương trình gọi đến thủ tục này trước tiên. Bạn cũng mở trang Properties của dự án LuckySeven như đã làm. Muốn để sub Main khởi động thì bạn lại phải chọn lại kiểu của ứng dụng trong danh sách Application Type. Ta có thể chọn là

Console Application hay Windows Service. Trong trường hợp này là **Windows Service**, sau đó chọn Sub Main trong danh sách Start Object.

Bây giờ bạn chạy chương trình và sẽ thấy thủ tục Sub Main được triệu gọi trước tiên. Nó đưa ra thông báo "Đây là thủ tục Sub Main" và gọi đến form1 sau khi người dùng ấn OK của hộp thoại:



7. Tổng kết chương 15

Chúng ta đã hoàn thành chương 15 – chương viết về form và quản lý form. Như các chương trước các bạn tự mình tổng kết những gì đã học.

Chương 16:

Xử lý đồ họa và các hiệu ứng ảnh động

--------oOo--------

Nội dung thảo luận:

- Sử dụng thư viện hay không gian tên System.Drawing để vẽ ảnh đồ họa
- Tạo hiệu ứng ảnh động trên form
- Co giãn đối tượng trên form
- Tạo một form trong suốt (transparency)

VB.NET cung cấp đủ công cụ và thư viện để khai thác các hiệu ứng đồ họa. Trong chương này chúng ta sẽ khám phá việc tạo các form mang dáng dấp đồ họa, tạo hiệu ứng ảnh động dựa vào PictureBox và bộ định thời Timer, co giãn các đối tượng dựa vào thuộc tính Height và Width.

Chú ý:

- VB.NET sử dụng các hàm đồ họa trong thư viện GDI+ chứa trong System.Drawing để vẽ các hình đơn giản như đường thẳng, đường tròn…
- Hệ thống đồ họa trong VB.NET chỉ là điểm Pixel
- VB.NET không hỗ trợ phương thức Move, thay vào đó bạn sẽ sử dụng các thuộc tính Left, Top hay Location, SetBound.
- Có thể làm việc với nhiều khuôn dạng ảnh như BMP, GIF, JPEG, WMF, TIFF…

1. Thêm vào hình ảnh bằng cách sử dụng thư viện System.Drawing

Chương này chúng ta sẽ sử dụng các hàm API trong thư viện GDI+ để vẽ ảnh. Ta có thể tự vẽ ảnh, thay đổi màu nền, màu cọ, kiểu chữ vẽ và tất cả mọi thứ.

Hệ thống tọa độ của form

Trong VB, mỗi form có một hệ thống tọa độ riêng. Gốc tọa độ bắt đầu từ góc trái trên của form (dưới thanh tiêu đề). Đơn vị được tính bằng pixel.

Có hai trục, trục ngang là trục hoành – trục x, chiều hướng qua phải. Trục dọc, chiều hương xuống dưới là trục tung – trục y. Một điểm trên form được xác định bởi cặp tọa độ (x, y).

2. Lớp xử lý đồ họa System.Drawing.Graphics

Lớp Graphics trong thư viện System.Drawing chứa các phương thức và thuộc tính để vẽ hình ảnh lên form. Các lớp khác bạn có thể tham khảo trong Help của VB.NET.

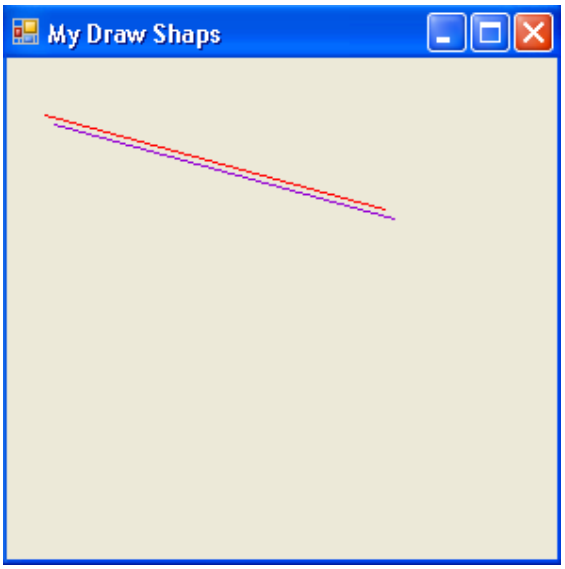
Sau đây là các phương thức dùng vẽ đường hình học cơ bản có trong lớp Graphics:

Đường hình học	Phương thức	Mô tả
Đường thẳng	Line	Đường thẳng nối hai điểm
Hình chữ nhật	DrawRectangle	Hình chữ nhật với 4 điểm
Cung tròn	DrawArc	Đường cong nối dây cung hai điểm
Vòng tròn/ Elipse	DrawEllipse	Vẽ hình Elip hay hình tròn
Đa giác	DrawPolygon	Đa giác được vẽ từ một tập các điểm
Đường cong	DrawCurve	Đường cong tự nhiên nối thành từ mảng các điểm
Đường cong bezier	DrawBezier	Đường cong Bezier

Ngoài ra còn có một số hàm tô đầy như là FillRectangle, FillEllipse, FillPolygon.

Khi sử dụng các phương thức của system.Drawing.Graphics bạn cần tạo ra một thể hiện của biến lớp Graphics. Tiếp theo tạo ra các đối tượng bút vẽ (Pen), chổi vẽ (Brush) để xác định nét vẽ hình học sẽ dùng vẽ và tô. Đối tượng vẽ Pen được truyền như tham số cho các phương thức vẽ không cần đến tô màu. Đối tượng Brush được truyền như tham số cho các phương thức vẽ yêu cầu đến tô màu. Ví dụ như phương thức DrawLine sau sẽ vẽ đường thẳng nối hai điểm (20, 30) và (100, 80). Biến đối tượng GraphicsFun được khai báo có kiểu Graphics và biến đối tượng Pen mang tên PenColor được dùng để chỉ định nét vẽ và màu để vẽ đường thẳng:

```
Dim GraphicsFun As Graphics
Dim PenColor As New System.Drawing.Pen(System.Drawing.Color.Red)
GraphicsFun = Me.CreateGraphics
GraphicsFun.DrawLine(PenColor, 20, 30, 200, 80)
GraphicsFun.DrawLine(Pens.DarkViolet, 25, 35, 205, 85)
```



2.1. Sử dụng sự kiện Paint của Form

Nếu bạn đặt đoạn mã trên vào sự kiện Click của một button nào đó thì khi click nút đó sẽ có một đường thẳng được vẽ ra. Tuy nhiên nếu ta di chuyển một cửa sổ khác đè lên chương trình hay thay đổi kích thước của form thì đường thẳng sẽ biến mất. Muốn nó hiện diện thường xuyên thì bạn phải biết khi nào cần vẽ lại đường thẳng. VB cung cấp sự kiện Paint để thực hiện công việc này. Bất kỳ khi nào chương trình bị Windows xóa nội dung cửa sổ và yêu cầu vẽ lại, nó sẽ gọi đến phương thức Paint, vì thế muốn tất cả các hình ảnh trên form hiển thị thường trực bạn cần đặt nó trong sự kiện Paint này.

Trong bài tập MyDrawShaps sau đây, chúng ta sẽ thực hành vẽ các đường cơ bản lên form sử dụng sự kiện Paint này. Bạn có thể di chuyển cửa sổ khác đè lên, thay đổi kích thước mà các hình không hề mất đi.

2.2. Chương trình MyDrawShaps vẽ hình chữ nhật, đường thẳng và Ellipse

Tạo mới một Solution và Add vào một dự án cùng tên MyDrawShaps

Thay đổi kích thước Form lớn hơn, đặt thuộc tính Text của Form là My Draw Shaps

Tạo thủ tục Form1_Paint bằng cách chọn Form1 Events trong danh sách Class Name của cửa sổ Code Editor, chọn Paint trong danh sách Method Name

Nhập vào đoạn mã sau:

```
'Chuẩn bị biến cho phương thức đồ họa
Dim GraphicsFun As Graphics
GraphicsFun = Me.CreateGraphics

'Sử dụng bút vẽ màu đỏ để vẽ đường thẳng và Ellipse
Dim PenColor As New System.Drawing.Pen(Color.Red)

GraphicsFun.DrawLine(PenColor, 20, 30, 100, 80)
GraphicsFun.DrawEllipse(PenColor, 10, 120, 200, 160)

'Sử dụng chổi vẽ màu xanh vẽ hình chữ nhật
Dim BrushColor As New SolidBrush(Color.Green)
GraphicsFun.FillRectangle(BrushColor, 150, 10, 250, 100)
```

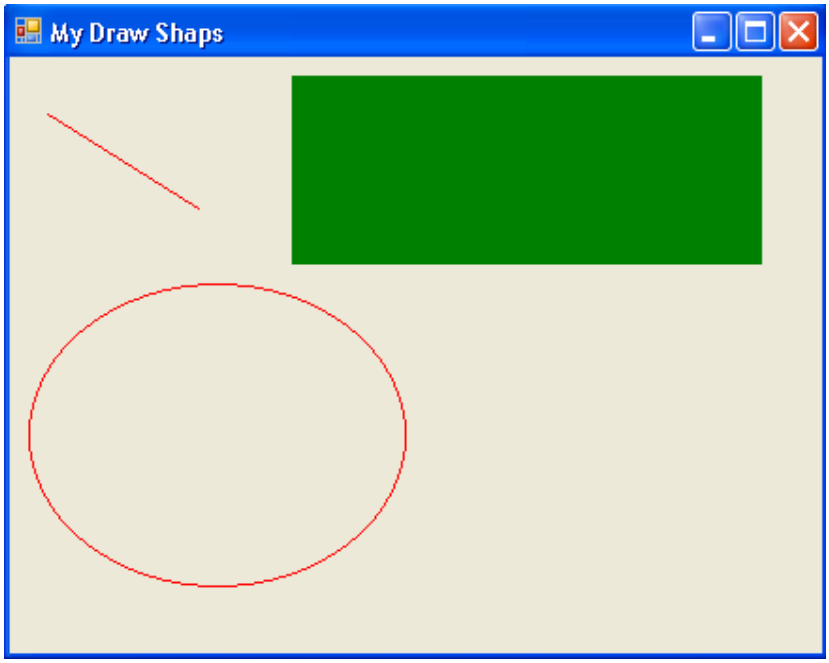
Ghi chú mã:

- Đoạn mã trên vẽ ba hình cơ bản là hình chữ nhật, đường thẳng và ellipse.
- Hàm vẽ hình Ellipse yêu cầu nhập hình chữ nhật cơ bản chứa Ellipse với 4 điểm góc.

Chạy chương trình:

Bạn ấn F5 để chạy chương trình.

Khi chương trình chạy, thủ tục Form1_Paint được triệu gọi và các hình xuất hiện. Bạn có thể thay đổi kích thước hay cho một cửa sổ khác đè lên mà không làm mất đi các hình trên đã vẽ. Kết quả:



3. Thêm hoạt hình cho chương trình

Trong phần này chúng ta sẽ khám phá một số hiệu ứng đơn giản như di chuyển vị trí ảnh trong PictureBox, phóng to, thu nhỏ ảnh kết hợp với bộ định thời Timer.

3.1. Di chuyển một đối tượng trên form

Như đã nói VB.NET không còn hỗ trợ phương thức Move như VB6. Thay vào đó bạn sử dụng thuộc tính Left, Top hay phương thức SetBounds để thay đổi vị trí, di chuyển và định lại kích thước cho đối tượng

Thuộc tính / phương thức	Mô tả
Left	Cho phép định tọa độ đỉnh góc trái trên cùng của đối tượng theo hướng ngang
Top	Định tọa độ đỉnh góc trái trên cùng của đối tượng theo hướng dọc
Location	Kết hợp của Left và Top
SetBounds	Thiết lập phạm vi (kích thước và vị trí cho đối tượng)

Thuộc tính Left và Top được dùng nhiều nhất khi muốn thay đổi vị trí của đối tượng. Để thay đổi vị trí đối tượng theo chiều ngang, ta thay đổi hay gán lại giá trị cho Left. Ngược lại theo chiều dọc, thay đổi hay gán lại giá trị cho Top. Ví dụ:

Để di chuyển đối tượng PictureBox1 sang ngang 300 bạn cộng Left của nó lên 300:

```
PictureBox1.Left = PictureBox1.Left + 300
```

Để di chuyển đối tượng PictureBox1 sang trái 300, trừ Left của nó đi 300:

```
PictureBox1.Left = PictureBox1.Left - 300
```

Để di chuyển theo chiều dọc lên trên 300, cộng Top lên 300:

```
PictureBox1.Top = PictureBox1.Top + 300
```

Để di chuyển xuống dưới 300, trừ Top đi 300:

```
PictureBox1.Top = PictureBox1.Top - 300
```

Nếu muốn định vị chính xác thuộc tính Top và Left bạn có thể gán như sau:

```
PictureBox1.Top = 20
PictureBox1.Left = 30
```

3.2. Thuộc tính Location

Bạn cũng có thể sử dụng thuộc tính Location để định vị trí của đối tượng như sau:

```
Dim p As New Point(20, 30)
PictureBox1.Location = p
```

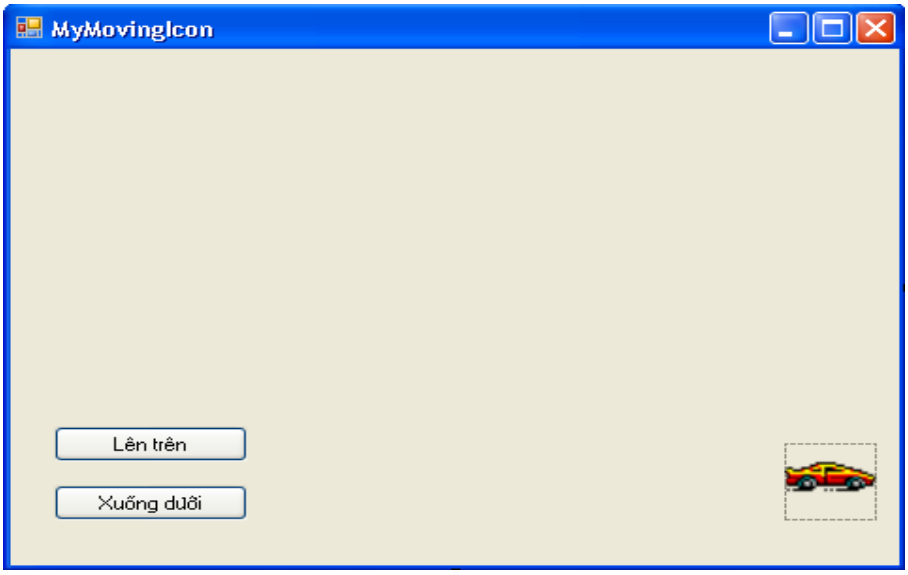
3.3. Tạo hiệu ứng hoạt hình dựa vào đối tượng Timer

Hiệu ứng hoạt hình dựa vào mẹo là, cứ sau một khoảng thời gian nào đó rất ngắn ta lại thay đổi vị trí của đối tượng ảnh. Trong bài tập MyMovingIcon sau đây chúng ta sẽ sử dụng đối tượng Timer để định thời gian di chuyển cho đối tượng ảnh chiếc ô tô (bạn có thể lấy bất cứ ảnh nào mình thích, miễn là dung lượng đủ nhỏ để chương trình chạy không quá chậm).

Tìm hiểu chương trình:

Chương trình có hai nút là "Lên trên" và "Xuống dưới" cùng một PictureBox. Khi người dùng click vào một trong hai nút thì ảnh chiếc ô tô sẽ tự động di chuyển theo chiều đó.

Thiết kế giao diện:



Bạn tạo mới một Solution và add vào một dự án cùng tên là MyMovingIcon rồi thiết kế giao diện như hình. Trong đó các đối tượng có thuộc tính như sau:

- Form1: Text – "MyMovingIcon"
- Button1: Name – btnLentren, Text – "Lên trên"
- Button2: Name – btnXuongduoi, Text – "Xuống dưới"
- PictureBox1: SizeMode – StretchImage, Image – một ảnh Icon bất kỳ mà bạn thích
- Timer1: Enable – False

Viết mã:

Trước hết chúng ta sẽ khai báo một biến có tên là ***lentren*** kiểu Boolean ở dưới dòng khai báo Form1 để nhận lệnh lên trên hay xuống dưới. Nếu người dùng click vào nút "Lên trên" thì biến *letren* = *true* và ngược lại khi click vào nút "Xuống dưới" thì *lentren* = *False*:

```
Dim lentren As Boolean
```

Tiếp theo ta tạo thủ tục `Timer1_Tick` để tạo hiệu ứng di chuyển ảnh. Bạn double – click vào đối tượng Timer1 để tạo thủ tục này và nhập vào đoạn mã như sau:

```
If lentren = True Then
    'Di chuyển ảnh lên trên
    If PictureBox1.Top > 10 Then
        PictureBox1.Location = New Point _
        (PictureBox1.Location.X - Int(Rnd() * 5), _
        PictureBox1.Location.Y - Int(Rnd() * 5))
    End If
Else
    'Di chuyển ảnh xuống dưới
    If PictureBox1.Top < (Me.Size.Height - 75) Then
        PictureBox1.Location = New Point _
        (PictureBox1.Location.X + 10, _
        PictureBox1.Location.Y + 10)
    End If
End If
```

Khi Timer hoạt động (thuộc tính Enable = True) thì sau khoảng 75 / 1000 giây, thủ tục Timer_Tick sẽ được gọi và nó tiến hành kiểm tra vị trí của PictureBox1 để thực thi hành động dịch chuyển tương ứng.

Bây giờ ta tạo thủ tục `btnLentren_Click` như sau:

```
lentren = True
Timer1.Enabled = True
```

Thủ tục `btnXuongduoi_Click`:

```
lentren = False
Timer1.Enabled = True
```

Để có thể sinh số ngẫu nhiên ta tạo thủ tục Form1_Load và nhập dòng mã sau:

```
Randomize()
```

Chạy chương trình:

Bạn ấn F5 để chạy chương trình. Ấn nút "Lên trên" và xem ảnh di chuyển lên trên. Ấn "Xuống dưới" để xem ảnh di chuyển xuống dưới:

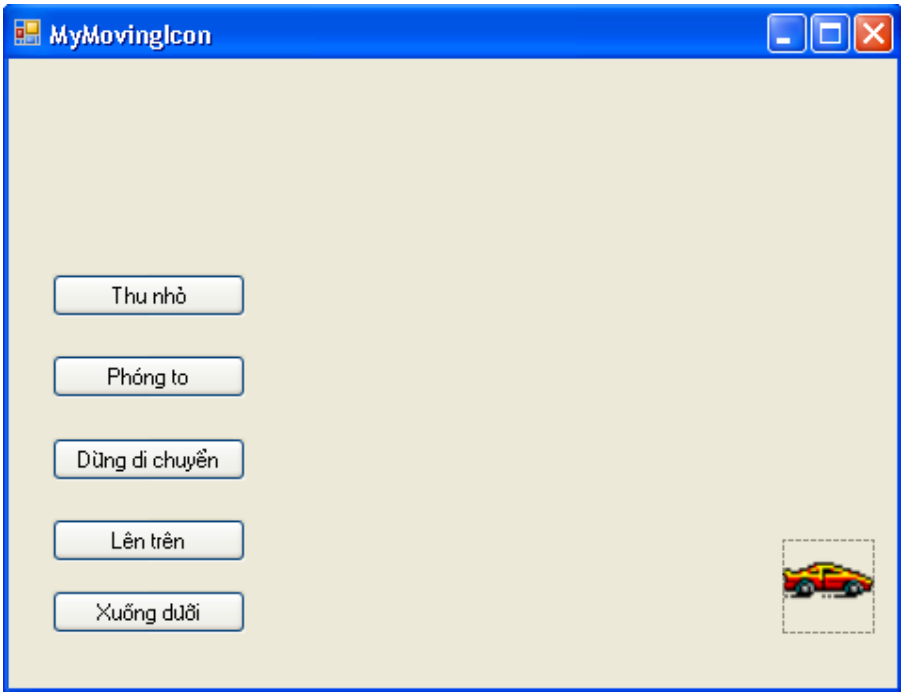
4. Phóng to, thu nhỏ đối tượng khi chương trình đang thực thi

Bạn cũng có thể sử dụng thuộc tính Height và Width để thay đổi chiều cao, chiều rộng của đối tượng để đối tượng có thể phóng to, thu nhỏ hay co giãn được. Bây giờ chúng ta sẽ chỉnh sửa bài tập MyMovingIcon để có thể minh họa cách phóng to, thu nhỏ của đối tượng PictureBox1.

Làm theo các bước sau đây:

- Mở lại Solution MyMovingIcon nếu đã đóng nó.
- Trở lại cửa sổ thiết kế giao diện và tạo thêm ba nút nhấn mới là btnDungdichuyen – Nút nhấn tạm dừng di chuyển nếu Icon đang đi chuyển theo lệnh; btnPhongto – Nút nhấn cho phép phóng to ảnh; btnThunho – Nút nhấn cho phép thu nhỏ ảnh.
- Ta cũng sẽ tạo thêm thủ tục PictureBox1_Click cho phép phóng to ảnh khi người dùng click lên ảnh khi chương trình đang thực thi.

Giao diện của Solution sau khi chỉnh sửa sẽ như sau:



Viết mã:

Tạo thủ tục `btnDungdichuyen_Click`:

```
Timer1.Enabled = False
```

Thủ tục này sẽ thiết lập lại thuộc tính Enable của Timer1 là False để dừng việc di chuyển ảnh.

Tạo thủ tục `PictureBox1_Click` phóng to ảnh nếu người dùn click vào ảnh:

```
PictureBox1.Height = PictureBox1.Height + 15
PictureBox1.Width = PictureBox1.Width + 15
```

Thủ tục này tăng các giá trị chiều cao (Height) và chiều rộng (Width) của PictureBox1 lên để ảnh trong nó co giãn theo.

Thủ tục `btnPhongto_Click` phóng to ảnh:

```
PictureBox1.Height = PictureBox1.Height + 15
PictureBox1.Width = PictureBox1.Width + 15
```

Thủ tục `btnThunho_Click` thu nhỏ ảnh:

```
PictureBox1.Height = PictureBox1.Height - 15
PictureBox1.Width = PictureBox1.Width - 15
```

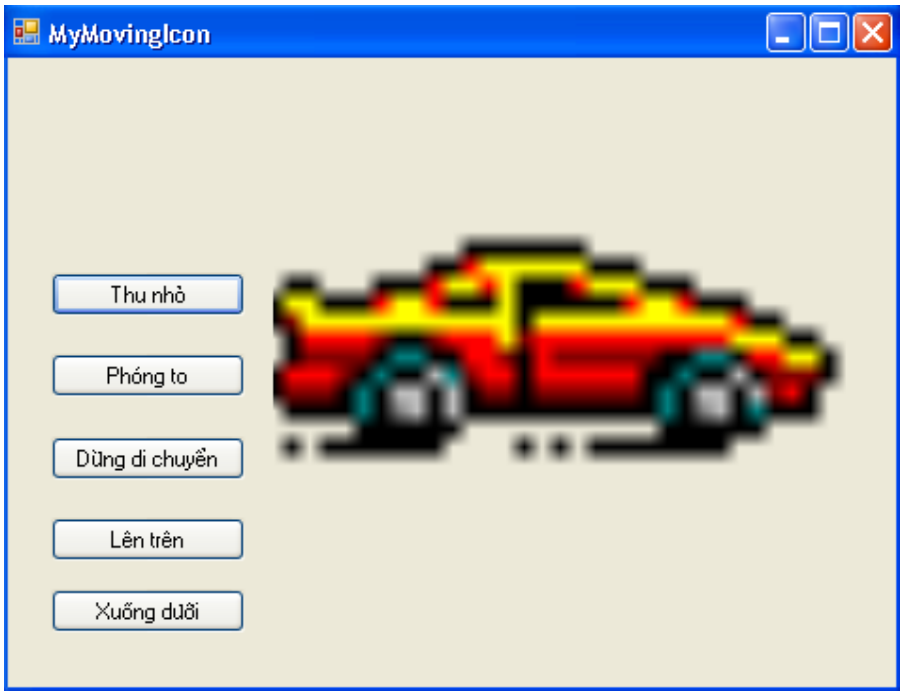
Thủ tục này tiến hành ngược lại với thủ tục phóng to, nó sẽ trừ giá trị chiều cao và chiều rộng của PictureBox1 đi để ảnh co lại.

Bây giờ chương trình đã sẵn sàng để bạn chạy thử.

Chạy chương trình:

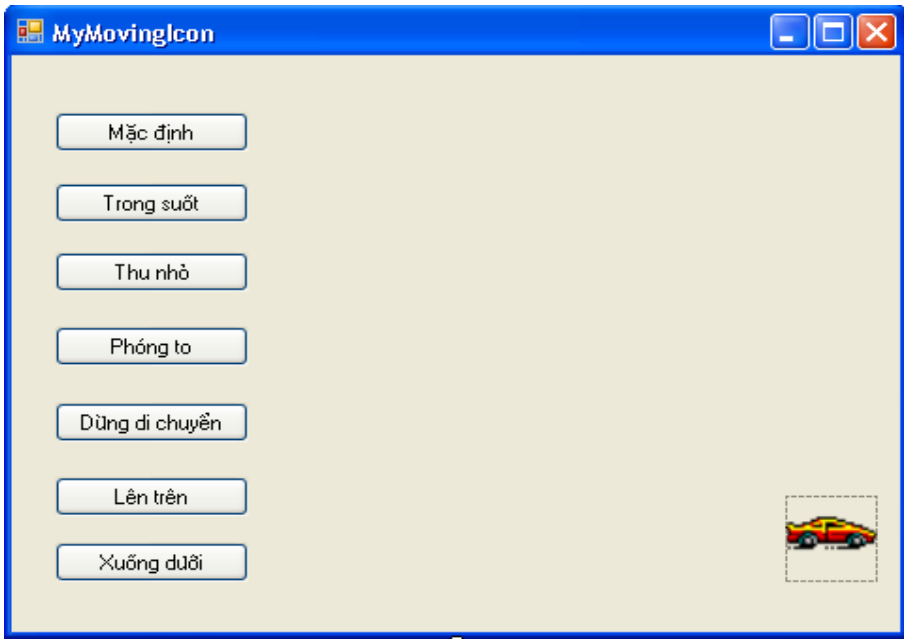
Ấn F5 để chạy chương trình. Bạn có thể ấn đồng thời hai nút lên trên và xuống dưới để xem ảnh di chuyển theo hai phương khác nhau.

Khi ảnh di chuyển đến giữa form, bạn click vào nút dừng di chuyển để ảnh định vị giữa form và cho phóng to, thu nhỏ ảnh. Kết quả:



5. Tạo cửa sổ Form trong suốt

Sau đây chúng ta sẽ tạo hiệu ứng trong suốt (transparency) cho cửa sổ form. Bạn mở lại Solution MyMovingIcon nếu đã đóng nó. Bạn tạo thêm hai nút nhấn là btnTrongsuot – nút nhấn tạo độ trong suốt cho Form và nút btnMacdinh – nút nhấn đưa form trở về trạng thái ban đầu.



Viết mã:

Tạo thủ tục `btnTrongsuot_Click` tạo độ trong suốt:

```
Me.Opacity = 0.5
```

Thủ tục `btnMacdinh_Click` đưa trạng thái form trở về như cũ:

```
Me.Opacity = 1
```

Chạy chương trình:

Ấn F5 để chương trình chạy. Bạn ấn nút nhấn "Trong suốt" để form trong suốt. Kết quả:

6. Tổng kết chương 16

Bạn làm bảng tổng kết những gì đã học trong chương 16 này và làm lại các bài tập theo ý thích của bạn.

Bạn có thể kết hợp với môn xử lý ảnh hay đồ họa máy tính để có những bài tập thực sự hữu ích. Trong thời gian tới tôi sẽ đưa mã nguồn một số bài tập xử lý ảnh cũng như mô phỏng chương trình Paint của Windows, chương trình vẽ giúp bé học tập. Mong các bạn đón đọc.

Chương 17:

Kế thừa Form và tạo các lớp cơ sở

--------oOo--------

Nội dung thảo luận:

- Sử dụng Inheritance Picker để kết hợp các form hiện có vào dự án
- Tự tạo các lớp cơ sở của bạn với các thuộc tính và phương thức tùy biến
- Dẫn xuất một lớp mới từ lớp cơ sở bằng từ khóa kế thừa Inherits

Trong các phiên bản trước đây, VB vẫn chưa được coi là ngôn ngữ lập trình mạnh do nó không có tính kế thừa để hoàn thiện mô hình lập trình hướng đối tượng. VB.NET đã cung cấp đầy đủ khả năng kế thừa. Bạn có thể kế thừa một form hay những lớp cơ sở để tạo ra những lớp đối tượng con.

Chú ý:

- Kế thừa form trong môi trường VB.NET sử dụng đối tượng Inheritance Picker.
- Lớp người dùng giờ đây có thể chứa trong một file.
- Các từ khóa *Property Get, Property Set và Property Let* không còn được sử dụng nữa.
- Có thể tạo lớp con kế thừa lớp cha thông qua từ khóa Inherits.

1. Kế thừa và sử dụng lại form đã có bằng INHERITANCE PICKER

Kế thừa là ta tận dụng lại những gì đã có, những ưu điểm của form hay lớp cha. Trong thiết kế form thì thực ra lớp form của ta kế thừa từ lớp cha là System.Windows.Forms.Form. Việc kế thừa một form ta có hai cách làm, đó là bạn có thể viết bằng mã chương trình như sau:

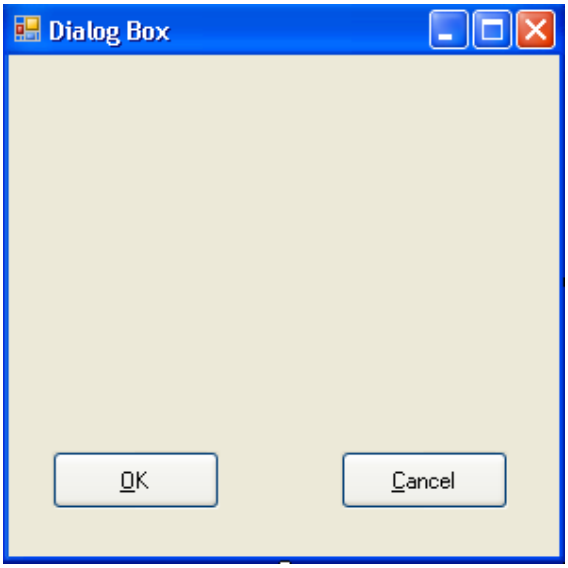
```
Public Class Form1 : Inherits System.Windows.Forms.Form
```

Cách thứ hai là dựa vào bộ công cụ Inheritance Picker để thực hiện việc kế thừa ngay trong khi thiết kế. Bạn truy xuất công cụ này nhờ việc chọn lệnh Project | Add New Item và chọn mục Inherited Form trong danh sách mục mới. Nhưng dự án cần được Build trước khi việc kế thừa có thể thực hiện.

1.1. Kế thừa một form hộp thoại đơn giản

Bây giờ chúng ta làm ví dụ MyFormInheritance để kế thừa một form hộp thoại đơn giản sau đây:

- Tạo mới một Solution và add vào một dự án cùng tên là MyFormInhertiance và thiết kế Form1 đơn giản với hai nút nhấn như sau:



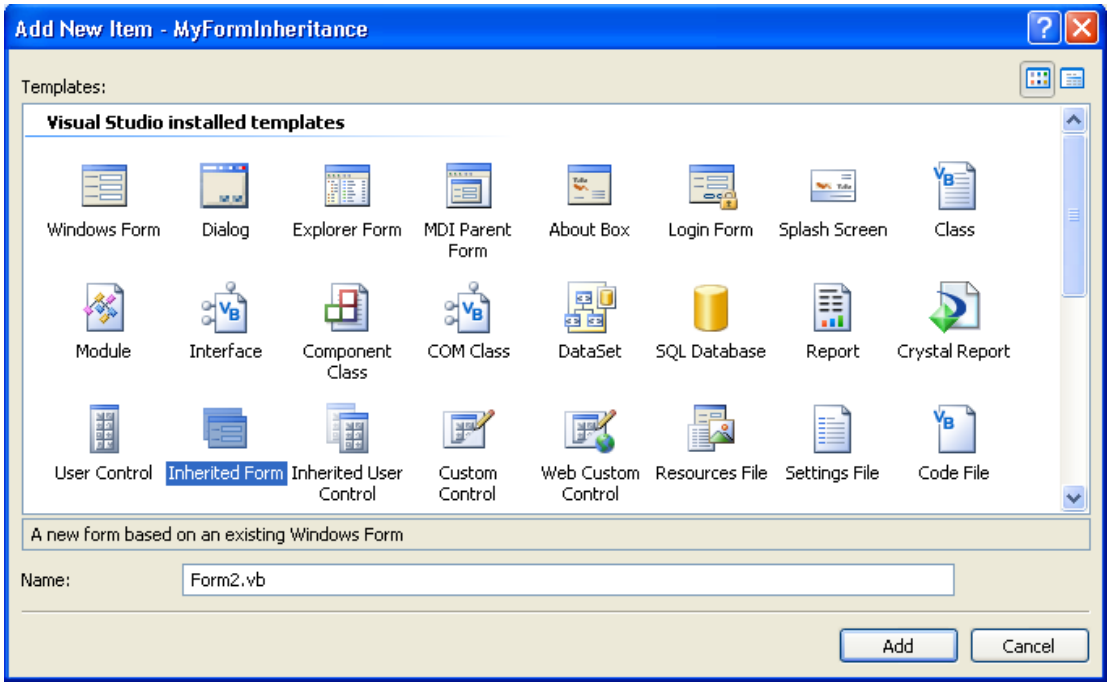
- Bạn tạo thủ tục Button1_Click và nhập vào dòng mã:

```
MsgBox("Bạn click nút OK")
```

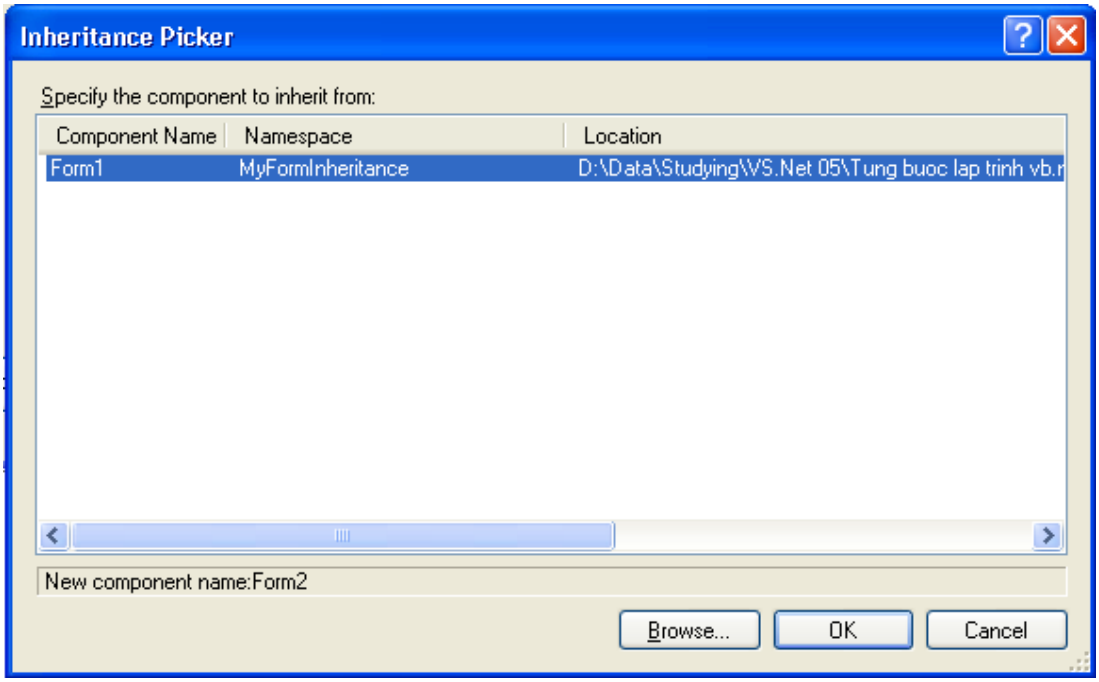
- Tương tự thủ tục Button2_Click với dòng mã:

```
MsgBox("Bạn click vào nút Cancel")
```

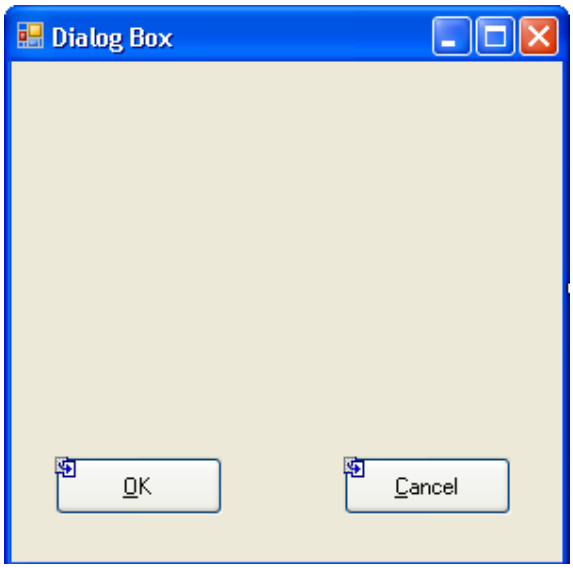
- Tiến hành Build Solution vì bạn chỉ có thể kế thừa các form khi chúng đã biên dịch ra File .DLL hay .EXE.
- Bây giờ ta sẽ tạo form2 kế thừa form1. Bạn chọn Project | Add | New Item rồi chọn mục Inherited Form trong danh sách:



- Nhấn nút Add để hiện hộp thoại Inheritance Picker như hình:



- Hộp thoại này liệt kê tất cả danh sách form có trong dự án hiện hành. Bạn có thể tìm các form khác trên đĩa cứng đã biên dịch bằng cách nhấp vào nút Browse.
- Chọn Form1 và nhấn nút OK. Lúc này một form mới xuất hiện với hai nút nhấn và thuộc tính Text "Dialog Box" kế thừa từ form1 như hình:

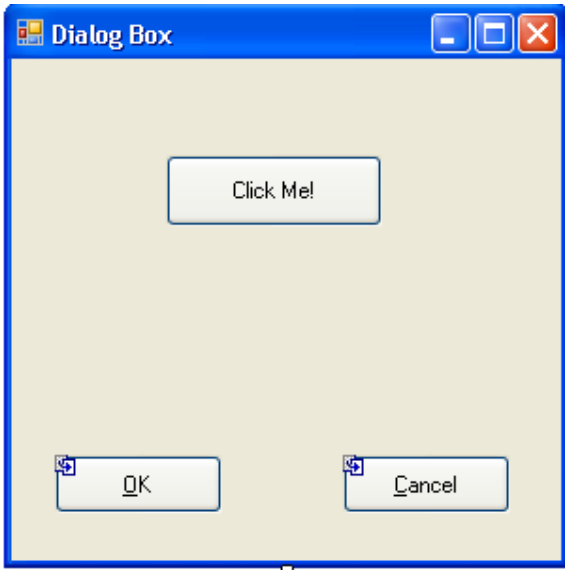


1.2. Tùy biến form kế thừa

Bạn đặt thêm một nút nhấn thứ ba nên form2, đặt thuộc tính Text của nó là "Click Me!"

Tạo thủ tục Button3_Click với dòng mã:

```
MsgBox("Đây là một Form kế thừa!")
```



Trở lại cửa sổ thiết kế form2 và thử double click vào nút OK hay Cancel ta thấy không thể chỉnh sửa thủ tục này. Điều này nghĩa là, bạn không thể chỉnh sửa thành phần kế thừa nhưng có thể thêm mới thành phần vào.

Đặt lại form2 là form khởi động theo cách đã học trong chương trước.

Chạy chương trình:

Chương trình đã hoàn thành và chúng ta có thể kiểm thử. Ấn F5 để chạy chương trình. Ấn nút OK và Cancel để xem hộp thoại thông báo.

2. Tự tạo các lớp cơ sở của riêng mình

Để biên dịch form2, Inheritance Picker sẽ tạo một liên kết đến dự án và form1 cùng form mới. Nội dung của form mới sẽ như sau (trong cửa sổ code editor của form1.vb bạn sẽ không nhìn thấy những khai báo này. Để xem bạn có thể dùng một trình soạn thảo nào đó như Edit Plus mở file tương ứng là Form2.Designer.vb):

```
Partial Class Form2
    Inherits MyFormInheritance.Form1
    …
```

Ngoài những gì kế thừa của VB.NET, chúng ta cũng có thể tạo ra những lớp của riêng mình. Lớp này cũng có thuộc tính, phương thức giống như của VB.NET. Để tạo chúng ta chọn Project | Add Class rồi định nghĩa lớp trong cửa sổ Code Editor.

Bài tập MyPersonClass sau đây sẽ hướng dẫn chúng ta cách tạo ra lớp cơ sở Person yêu cầu người dùng nhập vào tên nhân viên, ngày sinh. Thông tin này lưu trong lớp đối tượng. Ta cũng tạo một phương thức cho phép tính tuổi nhân viên.

2.1. Xây dựng lớp Person

- Bạn đóng dự án hiện hành lại. Tạo mới một solution và add vào một dự án cùng tên là MyPersonClass.
- Thiết kế form như hình:

Form gồm một nhãn Label1, một nút nhấn Button1 (Hiển thị), hai textBox như hình.

- Tạo lớp Person bằng cách chọn Project | Add Class (có thể R-Click vào dự án và chọn Add Class trong dach sách). Thay tên lớp là Person.vb.

Bây giờ chúng ta sẽ tạo lớp bằng cách viết mã cho lớp. Có ba bước chung để tạo lớp đó là khai báo biến của lớp, tạo các thuộc tính, tạo các phương thức. Chúng ta sẽ lần lượt tìm hiểu.

Khai báo biến lớp:

Chúng ta khai báo hai biến chứa First Name và Last Name ngay sau khai báo Public Class Person như sau:

```
Private FName, LName As String
```

Từ khóa Private cho biết biến này chỉ dùng để truy xuất trong phạm vi khai báo nó. Ở đây là truy xuất trong lớp Person.

Tạo thuộc tính:

Ta tạo thuộc tính FirstName cho lớp để trả về First Name của nhân viên. Bạn gõ vào dòng phát biểu sau:

```
Public Property FirstName() As String
```

Ấn Enter, VS.NET sẽ tự tạo ra cấu trúc đầy đủ của thuộc tính như thế này:

```
Public Property FirstName() As String
    Get

    End Get
    Set(ByVal value As String)

    End Set
End Property
```

Các từ khóa Get sẽ trả về giá trị cho thuộc tính khi người dùng muốn đọc nó và Set sẽ đặt giá trị cho thuộc tính khi gán giá trị cho nó.

Thêm vào mã cài đặt đầy đủ cho thuộc tính như sau:

```
Public Property FirstName() As String
    Get
        Return FName
    End Get
    Set(ByVal value As String)
        FName = value
    End Set
End Property
```

Khi người cùng đọc thông tin từ thuộc tính của đối tượng thì thuộc tính trả về giá trị biến Fname, còn khi gán giá trị thì nó sẽ gán giá trị của biến Lname bằng Value trong phần Set. Trong các thuộc tính phức tạp thì bạn có thể cài thêm các câu lệnh xử lý trong khối lệnh Get…End Get, Set…End Set. Nhưng việc trả về giá trị và gán giá trị là bắt buộc phải có.

Tương tự ta xây dựng thuộc tính LastName() như sau:

```
Public Property LastName() As String
    Get
        Return LName
    End Get
    Set(ByVal value As String)
        LName = value
    End Set
End Property
```

Tạo phương thức cho đối tượng:

Bây giờ chúng ta tạo phương thức Tinhtuoi để tính số tuổi của nhân viên dựa trên ngày sinh của họ.

Bên dưới khai báo thuộc tính LasName bạn khai báo một hàm (hàm Function vì nó trả lại cho nơi gọi số tuổi của nhân viên) như sau:

```
Public Function Tinhtuoi(ByVal NS As Date) As Integer
    Return Int(Now.Subtract(NS).Days / 365.25)
End Function
```

Từ khóa Public cho phép phương thức có thể truy cập bởi người dùng khi sử dụng lớp Person. Hàm trên sử dụng hàm Subtract trừ ngày hiện hành cho ngày sinh của nhân viên và chia cho 365.25 để tính ra số tuổi.

Vậy là lớp Person đã định nghĩa xong. Chúng ta sẽ sử dụng lớp này trong form1.

2.2. Tạo đối tượng dựa trên lớp định nghĩa

Trở lại cửa sổ thiết kế của form1.vb. Tạo thủ tục Button1_Click bằng cách nhấp đôi vào nút nhấn "hiển thị" và nhập vào đoạn mã sau:

```
Dim nhanvien As New Person
Dim NgaySinh As Date
nhanvien.FirstName = TextBox1.Text
nhanvien.LastName = TextBox2.Text
NgaySinh = DateTimePicker1.Value.Date

MsgBox("Nhân viên " & nhanvien.FirstName & " " & _
nhanvien.LastName _
& " đã " & nhanvien.Tinhtuoi(NgaySinh) & " tuổi")
```

Trong đoạn mã trên, trước hết ta khai báo biến đối tượng *nhanvien* có kiểu Person. Từ khóa New dùng để tạo vùng nhớ và cấp phát một đối tượng Person thật sự cho *nhanvien*. Ta khai báo biến NgaySinh để chứa ngày tháng nhập vào. Dữ liệu trong hai TextBox được gán cho thuộc tính FirstName và LastName.

Chạy chương trình:

Bạn nhấn F5 để chạy chương trình. Nhập vào tên đầy đủ ở cả hai ô textbox và chọn một ngày sinh phù hợp rồi ấn nút hiển thị. Kết quả:

3. Kế thừa lớp tự tạo

Ta cũng có thể kế thừa lớp tương tự như kế thừa form. Để kế thừa ta chọn Project | Add Class và cùng với từ khóa Inherits để tạo lớp kế thừa. Ta có thể thêm vào những phương thức cũng như thuộc tính mới đồng thời giữ lại những ưu điểm của lớp cha.

Bài tập sau sẽ minh họa việc tạo thêm lớp kế thừa của lớp Person ta vừa tạo. Chúng ta sẽ tạo thêm lớp mới có tên *kisu*.

Lớp này kế thừa lớp Person, nó có hai thuộc tính là Firstname và LastName cùng phương thức Tinhtuoi() nhưng cũng có thêm phương thức BacTho lưu lại cấp bậc tay nghề của kỹ sư đó.

Bạn add thêm một class tên là KiSu và đặt dưới câu khai báo lớp đó từ khóa Inherits như sau:

```
Public Class KiSu
    Inherits Person
```

Như vậy là lớp KiSu đã kế thừa lớp Person. Bạn bổ sung thêm mã cho lớp này như sau:

```
Public Class KiSu
    Inherits Person
    Private CapBac As Short
    Public Property BacTho() As Short
        Get
            Return CapBac
        End Get
        Set(ByVal value As Short)
            CapBac = value
        End Set
    End Property
End Class
```

Bây giờ chúng ta sẽ sử dụng lớp này. Bạn trở lại cửa sổ Code Editor của form1 và mở thủ tục Button1_Click rồi sửa như sau:

```
        Dim nhanvien As New KiSu
        Dim NgaySinh As Date
        nhanvien.FirstName = TextBox1.Text
        nhanvien.LastName = TextBox2.Text
        NgaySinh = DateTimePicker1.Value.Date
        nhanvien.BacTho = InputBox("Bậc thợ của nhân viên:")

        MsgBox("Nhân viên " & nhanvien.FirstName & " " & _
        nhanvien.LastName _
        & " đã " & nhanvien.Tinhtuoi(NgaySinh) & " tuổi." & _
        vbCrLf & "Bậc thợ: " & nhanvien.BacTho)
```

Trong đoạn mã này chúng ta cho biến nhanvien có kiểu KiSu để có thêm những thuộc tính và phương thức mới bên cạnh những thuộc tính có sẵn của lớp Person.

Chạy chương trình:

Nhấn F5 để chạy chương trình. Bạn nhập vào tên đầy đủ, số tuổi và nhập vào cấp bậc thợ của nhân viên để hiển thị như sau:

4. Tổng kết

Bạn làm bảng tổng kết những gì đã biết trong chương này. Bạn có thể tìm hiểu thêm về kế thừa trong các ngôn ngữ khác như C, C++, $C^{\#}$...

Chương 18:

Làm việc với máy in

--------oOo--------

Nội dung thảo luận:

- In đồ họa trong chương trình VB
- In văn bản trong chương trình VB
- In tài liệu nhiều trang trong chương trình VB
- Sử dụng hộp thoại Print, Page Setup, Print Preview

Trong chương này chúng ta sẽ học cách tích hợp chức năng in ấn vào trong form cũng như cách xử lý in ấn đồ họa, in văn bản và tài liệu nhiều trang.

Chú ý:

- Lớp đối tượng PrintDocument cung cấp nhiều chức năng in ấn hơn
- Chúng ta sử dụng các điều khiển hộp thoại PrintDialog, PrintPreViewDialog và PageSetupDialog để thực thi tác vụ in ấn.
- Để in tài liệu nhiều trang, chúng ta phải tạo ra thủ tục sử lý sự kiện PrintPage xử lý thao tác in mỗi lần từng trang tài liệu. Tuy nhiên hầu như các chức năng in ấn chúng ta đều được hỗ trợ trong thư viện System.Drawing.Printing.

1. Sử dụng lớp Printdocument

Printdocument là lớp bạn thường xuyên sử dụng khi in ấn. Việc sử dụng lớp này bằng cách chọn nó trên toolbox và thêm nó vào dự án như các điều khiển khác. Lớp này nằm trong thư viện System.Drawing.Printing. Bạn cần imports nó ra ở đầu chương trình khi muốn sử dụng. Bài tập MyPrintsGraphics sau đây sẽ hướng dẫn cách sử dụng đối tượng này.

1.1. Sử dụng điều khiển PrintDocument

Tìm hiểu chương trình:

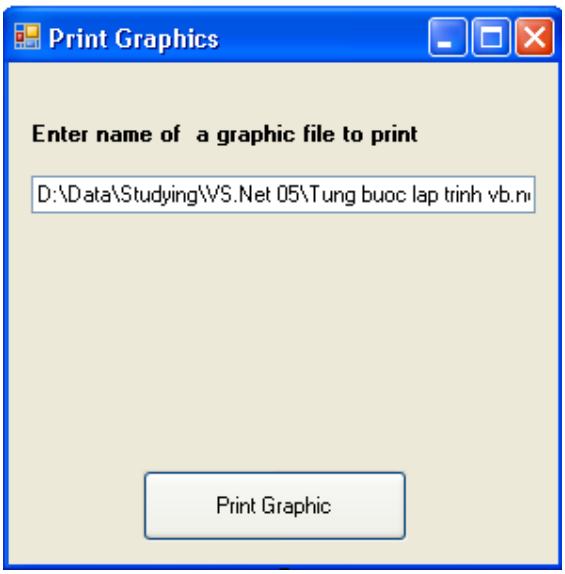
Chương trình sẽ bao gồm một textbox cho phép nhập đường dẫn của file graphic cần in và một nút nhấn cho phép in ra. Vì ta không nối máy in nên ta sẽ lưu ra một file định dạng *.mdi.

Thiết kế giao diện:

Bạn tạo mới một Solution và thêm vào một dự án cùng tên là MyPrintsGraphics và thiết kế giao diện như hình dưới đây. Trong đó các đối tượng có thuộc tính như sau:

- TextBox1: thuộc tính text là đường dẫn đến file graphics của bạn. Ở đây là file NFS2SER.ICO tôi đã chép vào thư mục chứa dự án.

- Label1 và nút nhấn có text như hình.



Bạn cũng thêm vào điều khiển PrintDocument1 từ ToolBox.

Viết mã:

Trước hết ta tạo mới một module có tên ModulePrinting.vb chứa hàm PrintGraphic là hàm phục vụ in ấn sẽ được triệu gọi trong thủ tục Button1_Click. Nội dung của module như sau:

```
Imports System.Drawing.Printing
Module ModulePrinting
    Public Sub PrintGraphic(ByVal sender As Object, _
    ByVal ev As PrintPageEventArgs)
        'Vẽ ảnh đồ họa bằng DrawImage
        ev.Graphics.DrawImage _
        (Image.FromFile(Form1.TextBox1.Text), _
        ev.Graphics.VisibleClipBounds)
        'Xác định không còn trang khác
        ev.HasMorePages = False
    End Sub
End Module
```

Thủ tục này là hạt nhân của tác vụ in ấn từng trang khi sự kiện PrintPage của điều khiển PrintDocument1 xảy ra sau lời gọi PrintDocument1.Print(). Trong đó, *ev* dùng để tham chiếu đến các thông số mà PrintDocument truyền vào. Ta dùng *ev* để lấy về đối tượng Graphics dùng vẽ ra máy in đồng thời sử dụng phương thức DrawImage để vẽ ảnh. Thuộc tính HasMorePage đặt là false để chỉ in trong một trang.

Trở lại cửa sổ thiết kế form1. Bạn vào chế độ Code Editor và đặt khai báo sau ở đầu Form1:

```
Imports System.Drawing.Printing
```

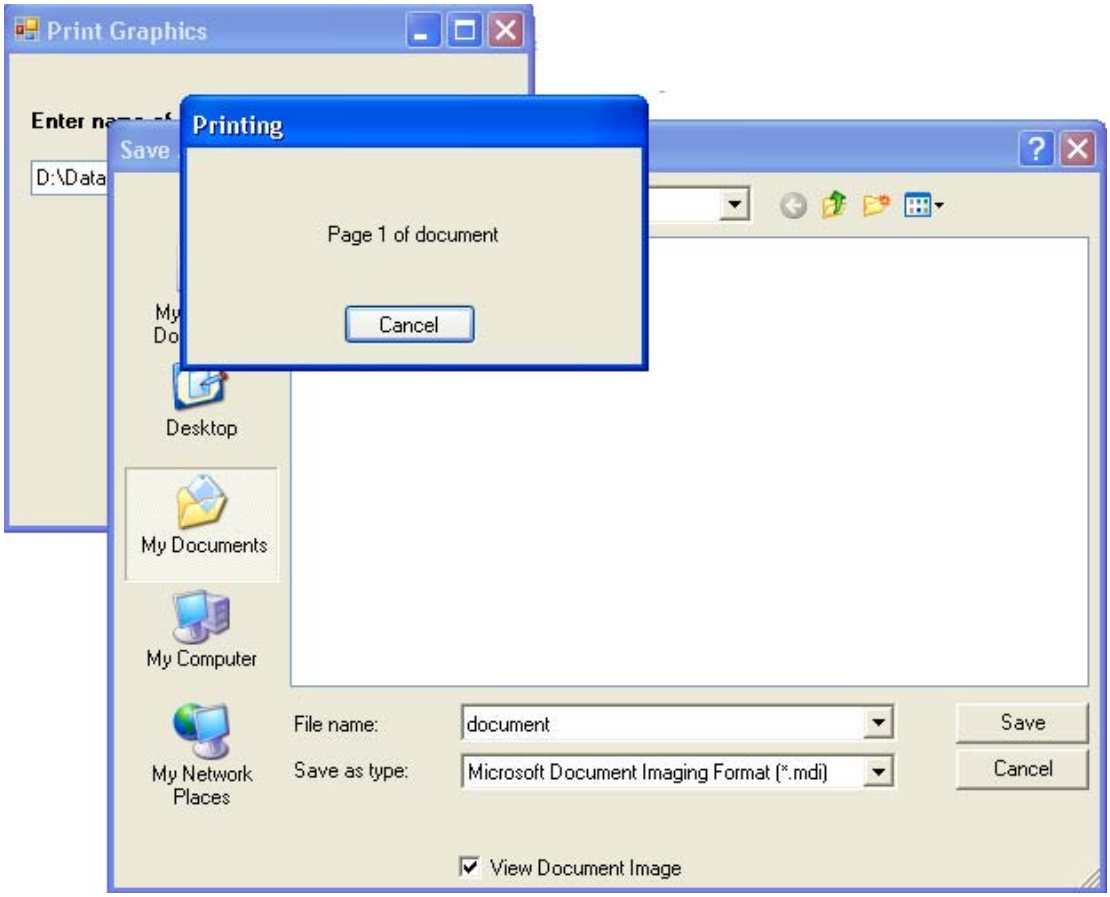
Tiếp theo ta tạo thủ tục Button1_Click như sau:

```
Private Sub Button1_Click(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles Button1.Click
    Try
        AddHandler PrintDocument1.PrintPage, _
        AddressOf ModulePrinting.PrintGraphic
        PrintDocument1.Print()
    Catch ex As Exception
        MessageBox.Show _
        ("Sorry-there is problem printing", ex.ToString)
    End Try
End Sub
```

Phát biểu AddHandler yêu cầu thủ tục xử lý PrintGraphic trong Module sẽ được triệu gọi khi sự kiện PrintPage của điều khiển PrintDocument xảy ra. Hàm AddressOf dùng để chỉ định địa chỉ của thủ tục PrintGraphic trong ModulePrint.vb. Đây là kỹ thuật chuyển giao (delegate) trong các ngôn ngữ của nền .NET. Bạn chuyển địa chỉ của hàm cho đối tượng và đối tượng sẽ triệu gọi hàm.

Chạy chương trình:

Ấn F5 để chạy chương trình. Bạn có thể nhập mới một đường dẫn đến đối tượng in hay click vào nút nhấn để in luôn. Kết quả:



Bạn chọn nơi lưu file *.mdi. và ấn nút save để lưu. Bạn cũng có thể dừng tác vụ in bằng cách ấn nút Cacel.

1.2. In văn bản từ đối tượng TextBox

Bài tập sau chúng ta sẽ in đoạn văn bản trong một ô textbox. Ta sẽ không dùng trực tiếp đối tượng PrintDocument nữa mà tạo nó bằng mã chương trình. Ta cũng dùng phương thức Graphics.DrawString thay cho phương thức DrawImage trước đây.

Bài tập MyTextPrinting:

Tìm hiểu chương trình:

Chương trình gồm một ô textbox và một nút nhấn. Ô textbox cho phép nhập đoạn văn bản và nút nhấn sẽ tiến hành in đoạn văn bản có trong textbox đó ra.

Thiết kế giao diện:

Bạn tạo mới một Solution và thêm vào một dự án cùng tên là MyTextPrinting, thiết kế giao diện như hình:

Trong đó các thuộc tính của các điều khiển như sau: đối tượng TextBox1: ScrollBars – Both, MultiLine – True. Các đối tượng và thuộc tính của nó như hình.

Viết mã:

Trước hết muốn sử dụng in ấn ta cần đặt khai báo sau ở đầu chương trình:

```
Imports System.Drawing.Printing
```

Khai báo một hàm phục vụ in đoạn văn bản trong ô textbox. Bạn tạo ngay trong cửa sổ code editor của form1.vb dưới dòng khai báo form1 như sau:

```
Private Sub PrinText(ByVal sender As Object, _
ByVal ev As PrintPageEventArgs)
    'sử dụng DrawString để vẽ chuỗi cần in
    ev.Graphics.DrawString(TextBox1.Text, _
```

```
        New Font("Arial", 11, FontStyle.Regular), _
        Brushes.Black, 120, 120)
        'Cho biết không còn trang tiếp theo
        ev.HasMorePages = False
    End Sub
```

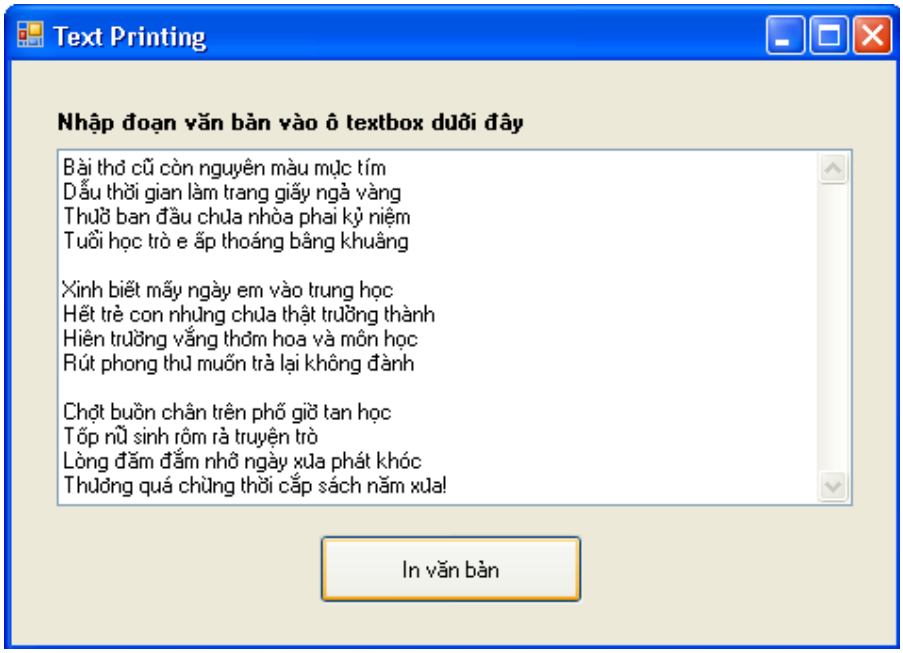
Tạo thủ tục Button1_Click như sau:

```
        Try
            'Khai báo biến PrintDoc có kiểu PrintDocument
            Dim PrintDoc As New PrintDocument()
            AddHandler PrintDoc.PrintPage, AddressOf Me.PrinText
            'Bắt đầu in
            PrintDoc.Print()
        Catch ex As Exception
            MessageBox.Show("Sorry-There is a problem printing", _
            ex.ToString())
        End Try
```

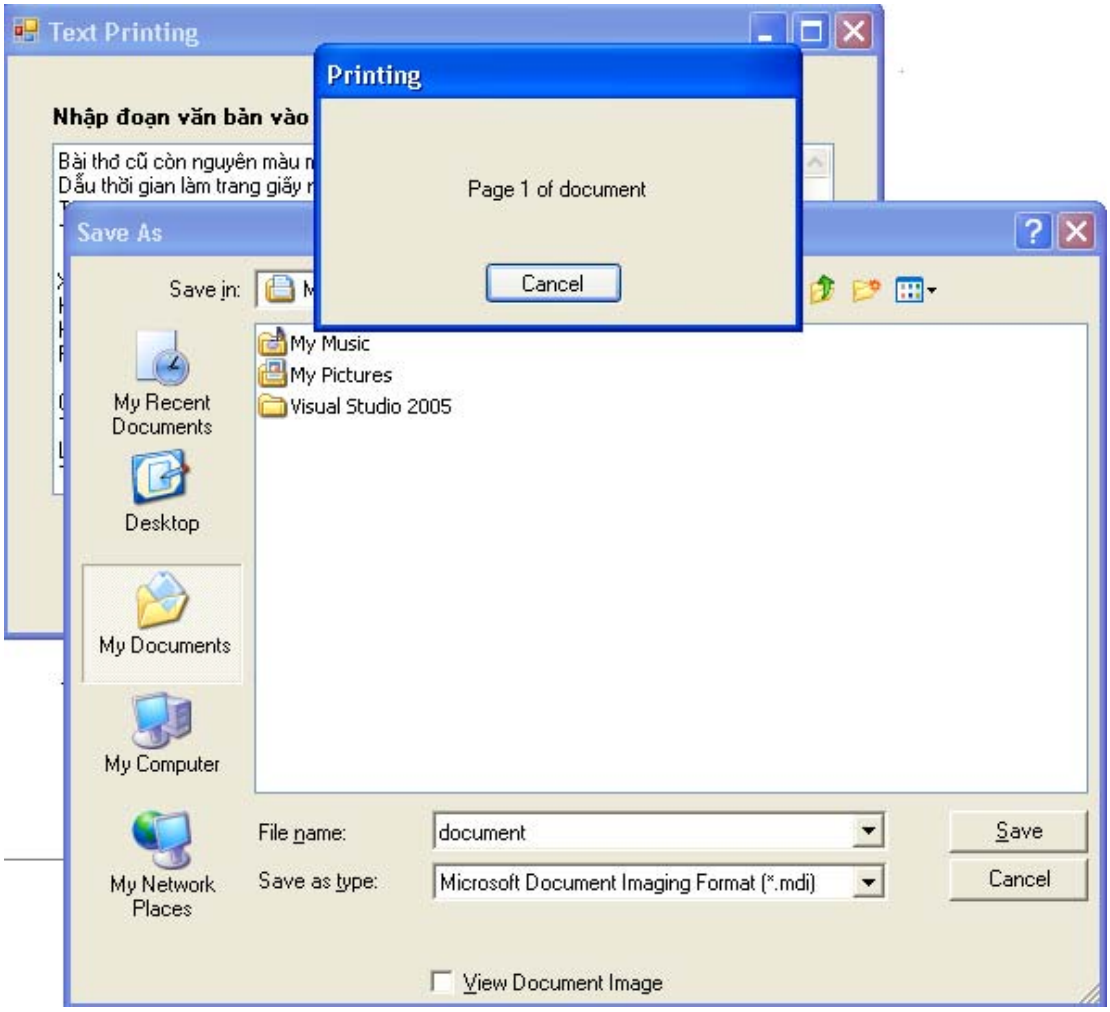
Trong thủ tục này ta khai báo một biến có kiểu PrintDocument thay vì sử dụng trực tiếp đối tượng PrintDocument từ ToolBox. Tiếp theo ta cũng sử dụng kỹ thuật chuyển giao để chỉ định hàm thực thi cho biến PrintDoc, hàm này là hàm PrinText ta đã khai báo trước đó. Cuối cùng là gọi phương thức PrintDoc.Print() để in trang in.

Chạy chương trình:

Bạn ấn F5 và nhập vào một đoạn văn bản bất kỳ:



Cũng giống như bài tập trước, bạn có thể ấn vào nút In văn bản và chờ một cửa sổ in xuất hiện để lưu vào một file định dạng .MDI như hình. File .MDI bạn có thể xem trong cùng thư mục chứa dự án nếu muốn.



2. In file văn bản nhiều trang

Ta đã sử dụng đối tượng PrintDocument để in đồ họa cũng như in văn bản đơn giản nhưng cũng còn rất nhiều vấn đề cần giải quyết. Đó là, PrintDocument không có ký tự ngắt dòng (10 và 13), ta cần tự viết mã xử lý ngắt dòng. Để in một văn bản có nhiều trang ta cũng cần viết mã ngắt trang. PrintDocument chỉ đơn giản là gửi dữ liệu đến máy in.

Có hai cách giải quyết vấn đề này. Thứ nhất, sử dụng đối tượng ev có kiểu PrintPageEventArgs để xác định trạng thái in sau mỗi trang được đưa ra máy in (kích thước trang, số ký tự có thể in trên một trang,…). Thứ hai, sử dụng phương thức Graphics.MeasureString. Phương thức này dùng để đo xem có chuỗi in ra sẽ chiếm kích thước dài và rộng bao nhiêu trên trang giấy. Chúng ta sẽ sử dụng kỹ thuật này để in tài liệu nhiều trang.

Bài tập MyFilePrinting sau đây sẽ minh họa việc in đó. Chương trình có thể mở một file với chiều dài bất kỳ và in nội dung của file đó ra nhiều trang. Ta cũng học cách sử dụng

các hộp thoại cấu hình máy in như PrintDialog cùng cách sử dụng đối tượng RichTextBox, OpenFileDialog. Trong đó RichTextBox là ô TextBox mở rộng có thể in và hiển thị được rất nhiều loại tài liệu định dạng font chữ, hình ảnh, kiểu tài liệu của MS Word.

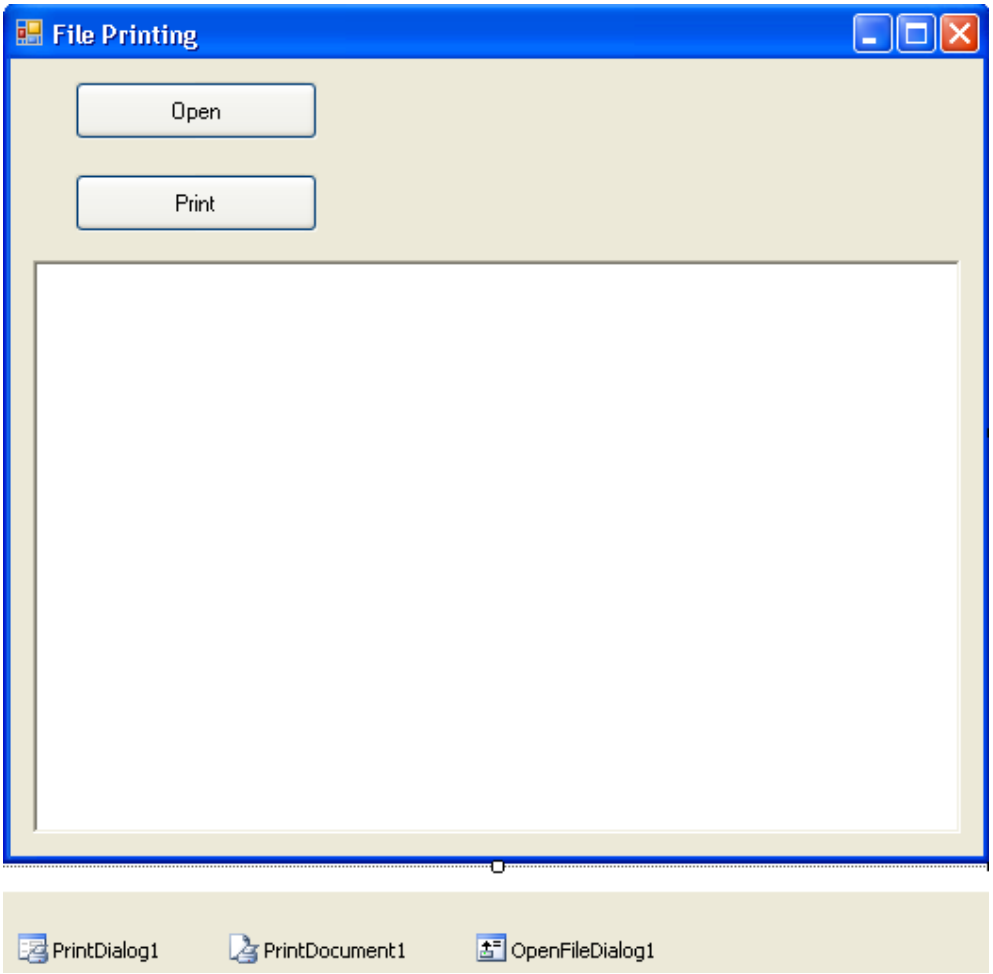
Bài tập MyFilePrinting:

Tìm hiểu chương trình:

Chương trình gồm một RichTextBox để hiển thị nội dung văn bản cần in. Hai nút nhấn, một nút Open cho phép mở file, một nút Print cho phép in văn bản.

Thiết kế giao diện:

Bạn tạo thêm các điều khiển phục vụ mở file, in file như OpenFileDialog1, PrintDocument và PrintDialog. Giao diện của chương trình như hình:



Trong đó thuộc tính của các đối tượng như sau:

- Button1: Text – "Open", Name – btnOpen
- Button2: Text – "Print", Name – btnPrint, Enable – False
- RichTextBox1: Name – rxtDocument

Viết mã:

Khai báo hai thư viện sau trước khai báo lớp Form1:

```
Imports System.IO 'Dùng để xử lý File
Imports System.Drawing.Printing
```

System.IO cho phép sử dụng lớp FileStream mở và đọc file, System.Drawing.Printing cho phép sử dụng các đối tượng in ấn.

Tiếp theo khai báo các biến dành cho việc in ấn, khai báo đặt ngay sau khai báo lớp form1:

```
Private PrintPageSetting As New PageSettings()
Private StringToPrint As String
Private PrintFont As New Font("Arial", 10)
```

Các biến này dùng khi in trang, biến PrintPageSetting để thiết đặt trang in, biến StringToPrint để đọc dòng in, biến PrintFont để thiết đặt font chữ.

Tạo thủ tục btnOpen_Click để mở file ra đọc:

```
Dim FilePath As String
'Hiển thị hộp thoại mở file
OpenFileDialog1.Filter = "Text Files (*.txt) | *.txt"
OpenFileDialog1.ShowDialog()
If OpenFileDialog1.FileName <> "" Then
    FilePath = OpenFileDialog1.FileName
    Try
        'Đọc nội dụng file vào rxtDocument
        Dim myFileStream As New FileStream _
        (FilePath, FileMode.Open)

        rxtDocument.LoadFile(myFileStream, _
        RichTextBoxStreamType.PlainText)

        myFileStream.Close()
        'Chuỗi để in
        StringToPrint = rxtDocument.Text
        'Enable nút nhấn Print
        btnPrint.Enabled = True
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End If
```

Khi ấn vào nút Open, đoạn mã trên sẽ hiển thị một hộp thoại cho phép chọn file. Khi chọn file xong, đường dẫn và tên file được lưu vào FileName của OpenFileDialog1. Ta gán thuộc tính filename của OpenFileDialog cho biến FilePath. Tiếp theo định nghĩa biến đối tượng FileStream để mở và đọc nội dung file. Toàn bộ nội dung file sau đó được tải vào RichTextBox rxtDocument nhờ phương thức LoadFile. Toàn bộ nội dung của rxtDocument sau đó lại được gán cho biến chuỗi StringToPrint. Nút nhấn Print lúc này được hiển thị cho phép click vào nó.

Bây giờ chúng ta cài đặt đoạn mã cho phép in ấn. Tạo thủ tục btnPrint_Click bằng cách double – click vào nút Print trên form1 và nhập vào đoạn mã sau:

```
Try
    'Chỉ định các thiết lập in mặc định
    PrintDocument1.DefaultPageSettings = PrintPageSetting
    'Lấy dữ liệu để in
    StringToPrint = rxtDocument.Text
    PrintDialog1.Document = PrintDocument1
    Dim result As DialogResult = PrintDialog1.ShowDialog()
    'In tài liệu nếu ấn OK
    If result = Windows.Forms.DialogResult.OK Then
        PrintDocument1.Print()
    End If
Catch ex As Exception
    MsgBox(ex.Message)
End Try
```

Trong đoạn mã trên, ta gán các thiết lập in mặc định của đối tượng PrintDocument1 là biến đối tượng PrintPageSetting, gán nội dung in trong ô rxtDocument vào biến StringToPrint. Tiếp theo, thuộc tính Document của đối tượng PrintDialog1 được gán bằng với đối tượng tài liệu muốn in hay PrintDocument1. Khi hộp thoại cấu hình máy in hiển thị, mọi thiết lập của người dùng trong hộp thoại sẽ được áp đặt cho tài liệu này. Tiếp theo ta phải cho hiển thị hộp thoại chọn cấu hình máy in để người dùng cấu hình bằng phương thức ShowDialog của PrintDialog1. Khi người dùng ấn OK trong hộp thoại cấu hình máy in thì PrintDocument1 sẽ gọi phương thức Print() để in.

Bây giờ ta tạo thủ tục PrintDocument1_PrintPage để thiết đặt các thông số in như ngắt dòng, ngắt trang,…bằng cách double click vào đối tượng PrintDocument1 trong cửa sổ thiết kế form hay chọn trong danh sách class name và method name ở cửa sổ code editor. Lúc trước ta dùng phương thức AddHandler để chuyển giao việc gọi hàm thì giờ đây ta sử dụng trực tiếp sự kiện có sẵn của PrintDocument1. Thủ tục này có nội dung như sau:

```
Dim sokytu, sodong As Integer
Dim chuoiin As String
Dim chuoidinhdang As New StringFormat()

'Định nghĩa vùng có thể in ra dựa trên thiết lập trang
Dim vungin As New RectangleF(e.MarginBounds.Left, _
e.MarginBounds.Top, e.MarginBounds.Width, _
e.MarginBounds.Height)

Dim kichthuoc As New SizeF(e.MarginBounds.Width, _
e.MarginBounds.Height - PrintFont.GetHeight(e.Graphics))

'Nếu in một chuỗi dài, yêu cầu ngắt xuống dòng
chuoidinhdang.Trimming = StringTrimming.Word

'Tính xem độ dài chuỗi bao nhiêu thì có thể vừa 1 dòng in
e.Graphics.MeasureString(StringToPrint, PrintFont, _
kichthuoc, chuoidinhdang, sokytu, sodong)

chuoiin = StringToPrint.Substring(0, sokytu)

'In chuỗi trên trang hiện hành
e.Graphics.DrawString(chuoiin, PrintFont, _
```

```
Brushes.Black, vungin, chuoidinhdang)

'Nếu vẫn còn chuỗi để in
If sokytu < StringToPrint.Length Then
    'Loại bỏ chuỗi đã in xong
    StringToPrint = StringToPrint.Substring(sokytu)
    e.HasMorePages = True
Else
    e.HasMorePages = False
    'Tất cả chuỗi đã in xong, khôi phục lại dữ liệu
    StringToPrint = rxtDocument.Text
End If
```

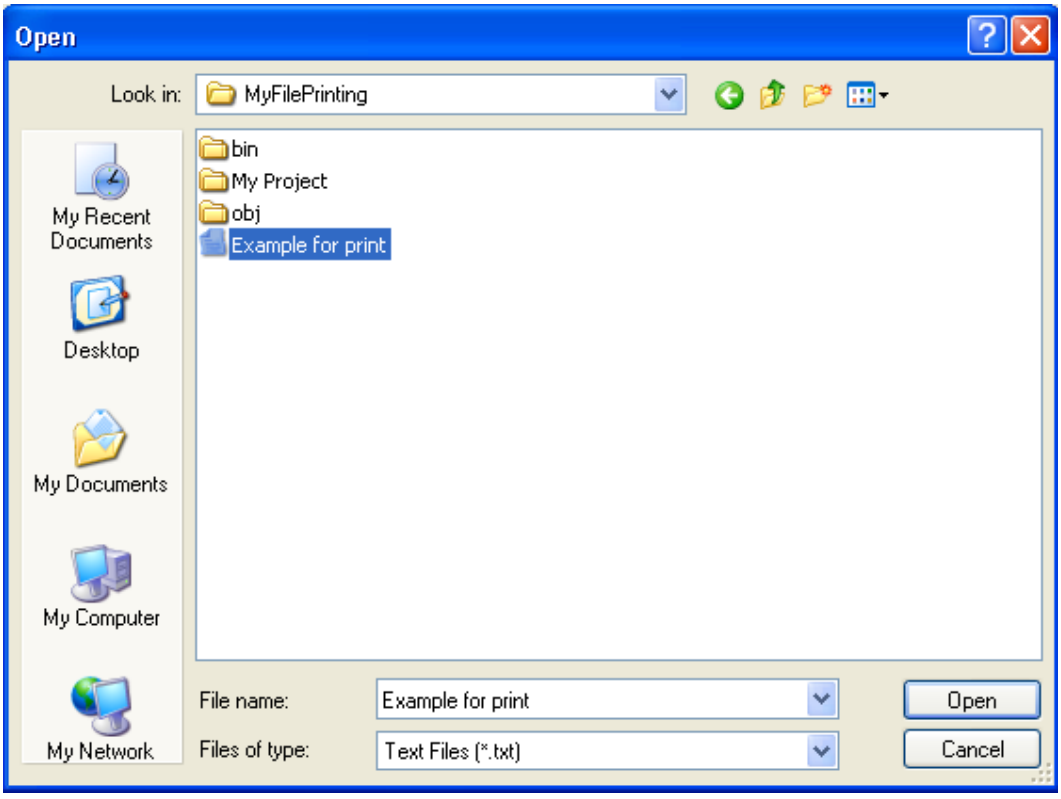
Thủ tục này thực sự tính toán khổ giấy và đẩy dl ra máy in. Trước hết, ta định nghĩa vùng in trong một khung hình chữ nhật:

```
Dim vungin As New RectangleF(e.MarginBounds.Left, _
e.MarginBounds.Top, e.MarginBounds.Width, _
e.MarginBounds.Height)
```

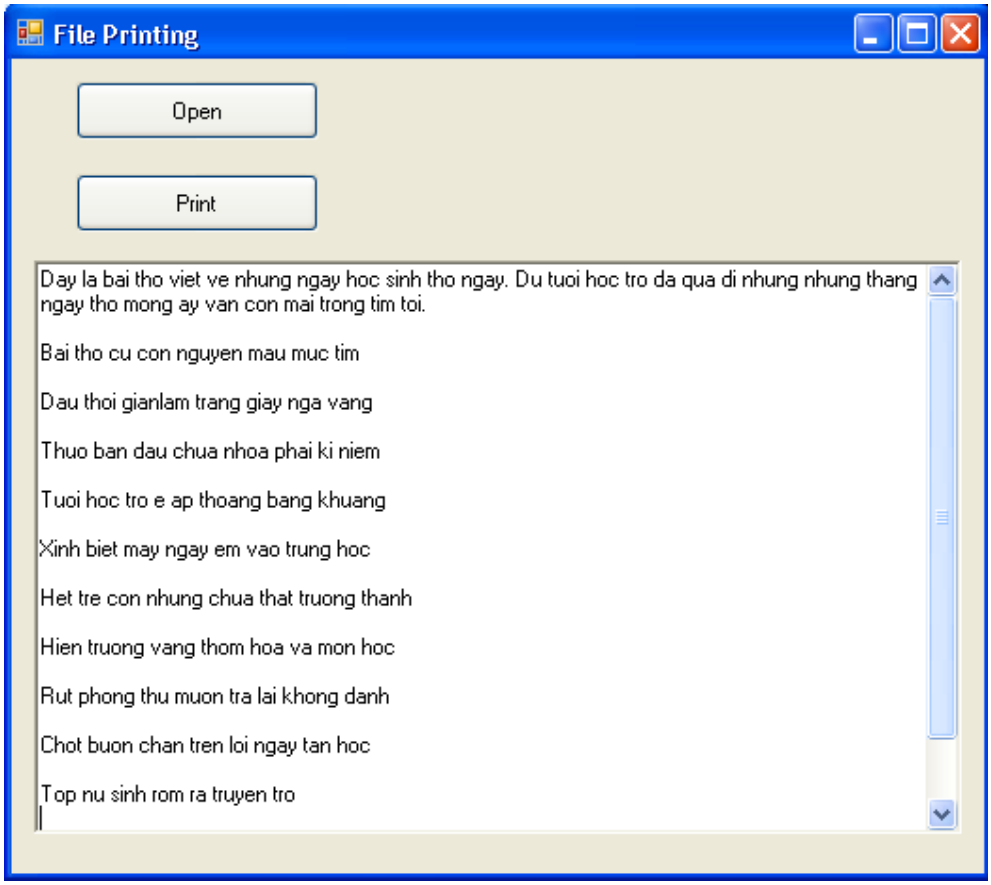
Vùng in này được thiết lập nhờ các thông số do biến e truyền vào(biến e này tương đương với biến ev trong thủ tục PrinText và PrintGraphics trước đây). Thông số mà biến e có được là do hộp thoại PrintDialog cung cấp. Tất cả văn bản trong vùng này đều in ra bình thường, phần dài hơn sẽ được đẩy ra in dòng khác. Dữ liệu sau đó được đẩy ra máy in bằng phương thức DrawString.

Chạy chương trình:

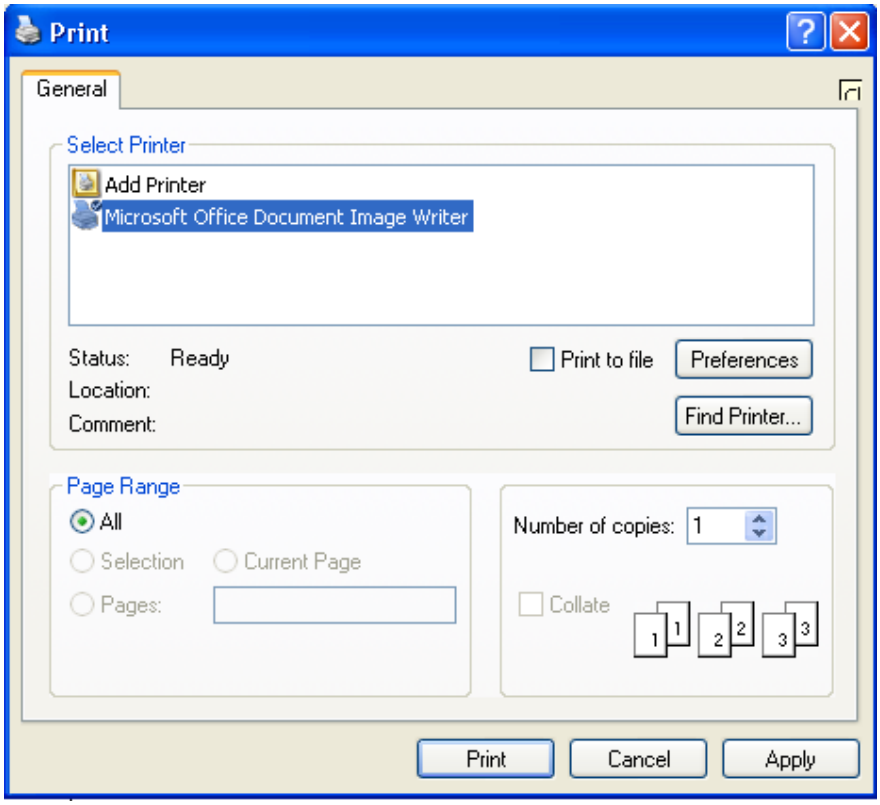
Ấn F5 để chạy chương trình. Bạn mở một file text nào đó.



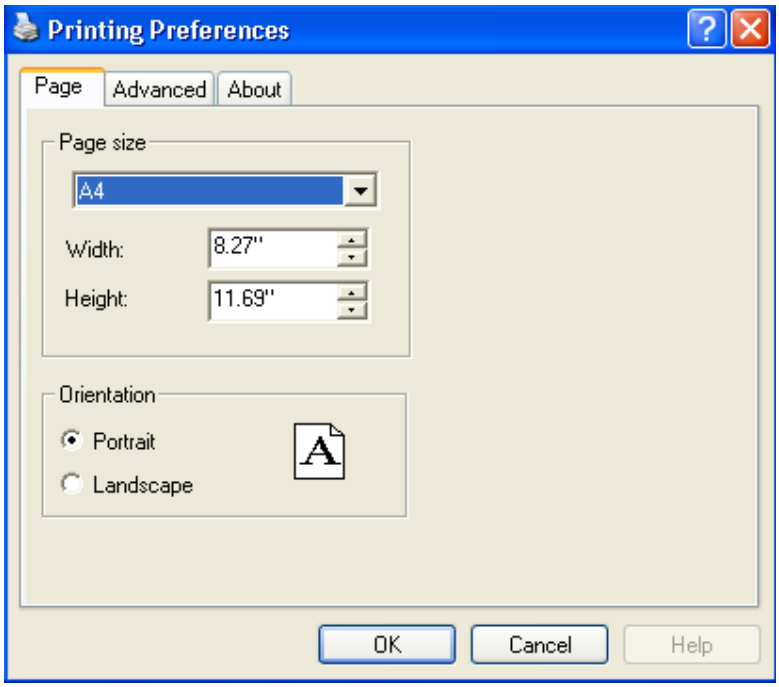
Chương trình sẽ hiển thị nội dung file vào trong rxtDocument:



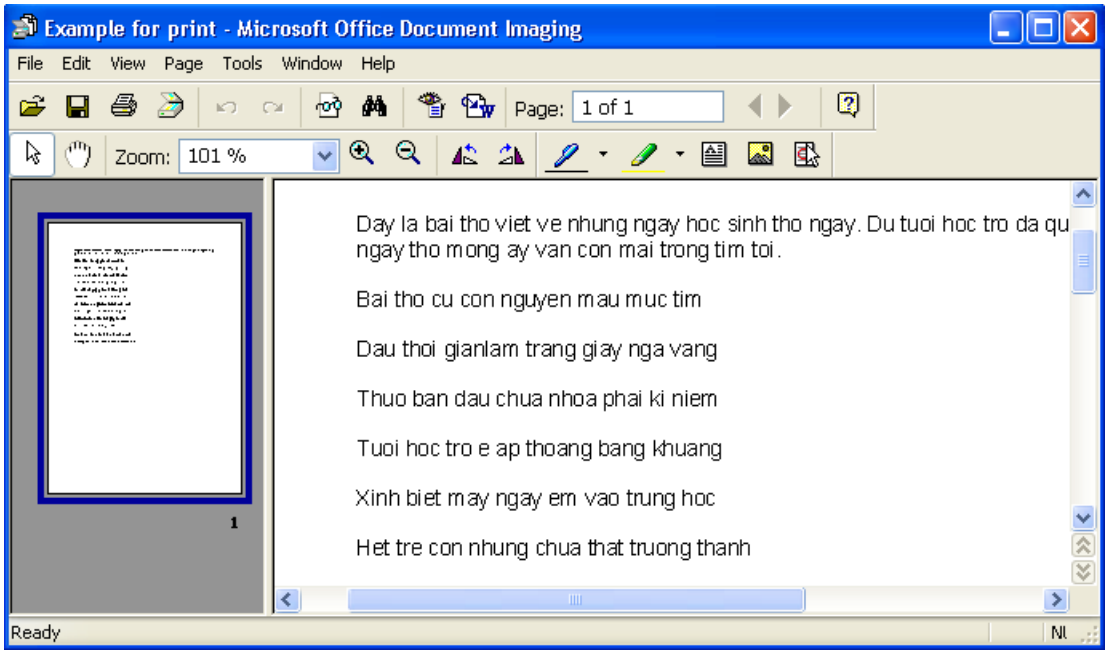
Nút Print hiện lên, bạn click vào nó để chương trinh hiện hộp thoại thiết lập trang in:



Tôi chưa kết nối máy in nên danh sách máy in không hiện lên ở đây. Bạn có thể thiết lập trang in để truyền thông số vào thủ tục ta đã xây dựng:



Bạn ấn OK, Print để chương trình in ấn. Ở đây tôi lưu vào file Example For Print.mdi bạn có thể xem:

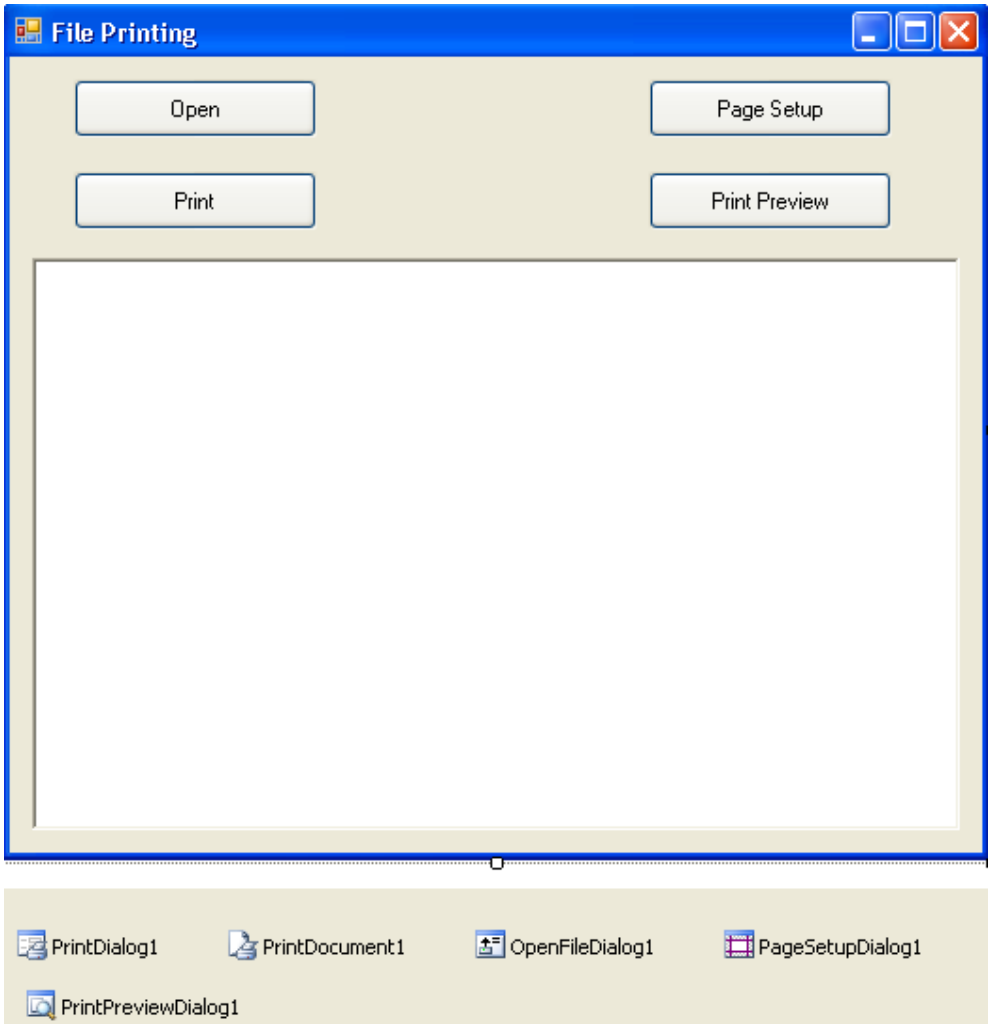


Như vậy là ta đã hoàn thành việc thiết lập trang in cho việc in văn bản nhiều trang. Tiếp sau đây chúng ta sẽ dùng hộp thoại PrintPreviewDialog và PageSetupDialog vào việc thiết đặt trang in.

3. Sử dụng hộp thoại PrintPreviewDialog và PageSetupDialog

Chương trình in ấn của chúng ta đã hoàn chỉnh, giờ chúng ta sẽ cải tiến chương trình MyFilePrinting cho phép định dạng trang in, chọn khổ giấy và xem trước trang in bằng hộp thoại PageSetupDialog và hộp thoại PrintPreviewDialog.

Bạn mở lại Solution MyFilePrinting ta đã làm và đặt thêm hai điều khiển PrintPreViewDialog và PageSetupDialog cùng hai nút nhấn vào form, kết quả như hình:



Trong đó các điều khiển mới có thuộc tính như sau:

- Button1: Name – btnPageSetup, Text – "Page Setup", Enable – False.
- Button2: Name – btnPreview, Text – "Print Preview", Enable – False.

Khi người dùng click vào nút Page Setup thì một hộp thoại cho phép người dùng chọn khổ giấy và thiết lập các thông số cho trang in. Khi click vào nút Print Preview thì hộp thoại cho phép người dùng xem trước trang in hiện ra.

Bây giờ ta sẽ tạo thủ tục btnPageSetup_Click bằng cách double click vào nút Page Setup và nhập vào đoạn mã như sau:

```
Try
    'Nạp các thiết lập và hiển thị hộp thoại PageSetup
    PageSetupDialog1.PageSettings = PrintPageSetting
    PageSetupDialog1.ShowDialog()
Catch ex As Exception
    MsgBox(ex.Message)
End Try
```

Đoạn mã này sẽ tạo ra một hộp thoại Page Setup. Biến PrintPageSetting đã được khai báo ở đầu chương trình. Biến PrintPageSetting sẽ nắm giữ các thông số mà hộp thoại PageSetupDialog trả về.

Tiếp theo ta tạo thủ tục btnPreview_Click và nhập đoạn mã như sau:

```
Try
    'Chỉ định các thiết lập trang in hiện hành
    PrintDocument1.DefaultPageSettings = PrintPageSetting

    'Chỉ định tài liệu hiển thị cho hộp thoại PrintPreview
    StringToPrint = rxtDocument.Text
    PrintPreviewDialog1.Document = PrintDocument1
    PrintPreviewDialog1.ShowDialog()
Catch ex As Exception
    MsgBox(ex.Message)
End Try
```

Thủ tục này sẽ gán biến PrintPageSetting cho thuộc tính DefaultPageSettings của đối tượng PrintDocument1, copy văn bản trong ô rxtDocument vào biến StringToPrint và mở hộp thoại PrintPreviewDialog1. Thuộc tính Document của hộp thoại PrintPreviewDialog1 được gán bằng PrintDocument1 để chương trình biết bạn đang cần xem tài liệu nào.

Sau cùng ta cần enable hai nút nhấn mới thêm là Page Setup và Print Preview bằng hai dòng mã sau trong thủ tục btnOpen_Click:

```
btnPageSetup.Enabled = True
btnPreview.Enabled = True
```

Bạn cũng có thể cho phép người dùng nhập nội dung vào ô RichTextBox bằng cách bổ sung thêm thủ tục `rxtDocument_TextChanged` với nội dung như sau:

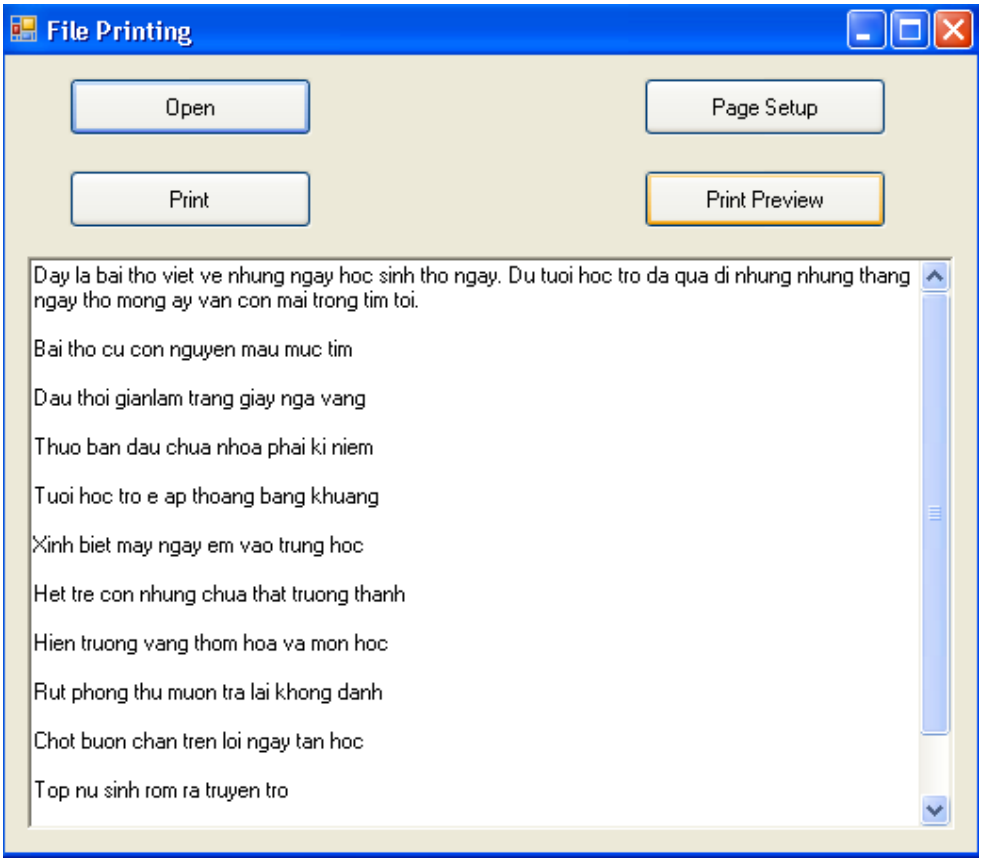
```
Private Sub rxtDocument_TextChanged(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles rxtDocument.TextChanged
    btnPrint.Enabled = True
    btnPageSetup.Enabled = True
    btnPreview.Enabled = True
End Sub
```

Bây giờ chương trình đã hoàn chỉnh, bạn hãy lưu lại tất cả những thay đổi của chương trình bằng cách nhấn vào nút Save All  trên thanh Standard Bar.

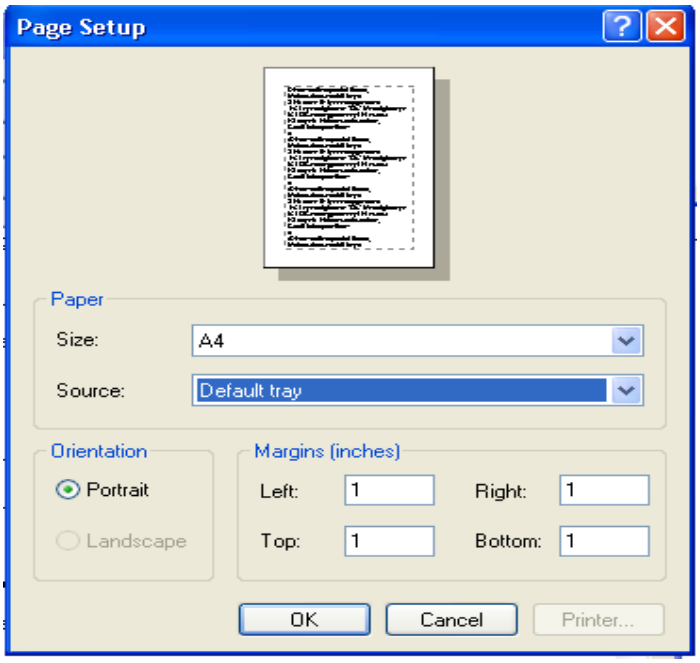
Chúng ta sẽ tiến hành chạy và kiểm tra chương trình.

Chạy chương trình:

Bạn chạy chương trình bằng cách nhấn nút F5, bạn hãy thử mở lại file text nào đó hay có thể tự nhập nội dung văn bản vào trong ô RichTextBox.

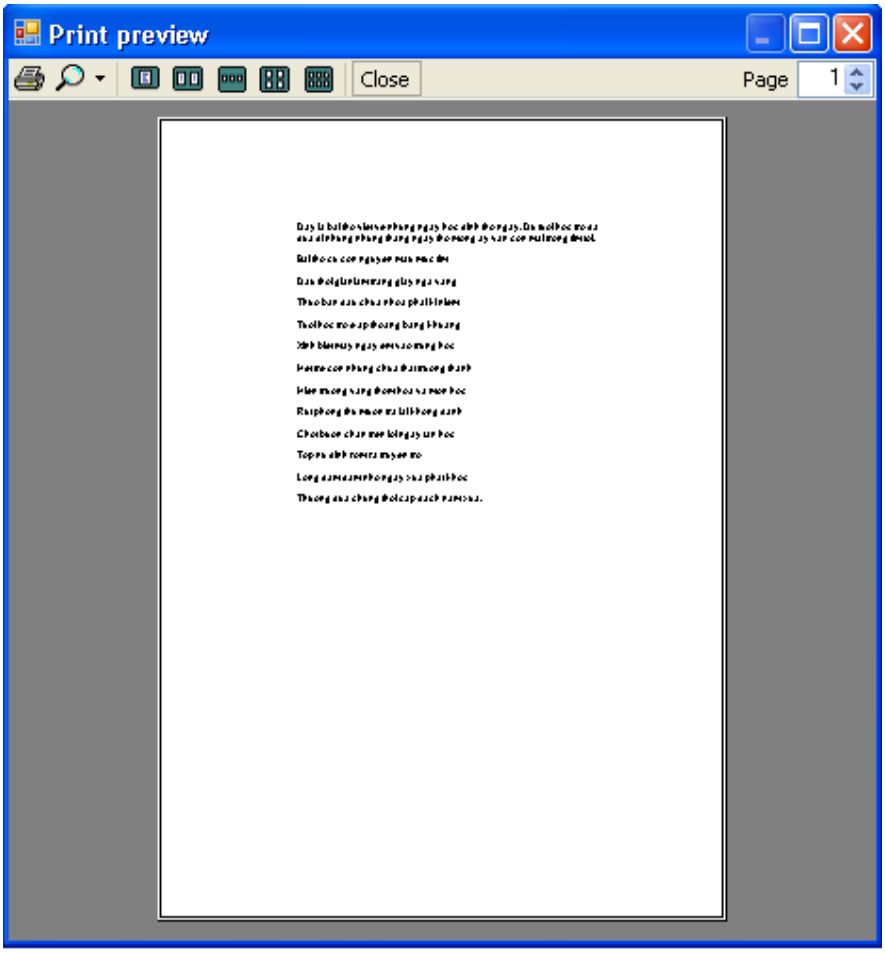


Bạn nhấn vào nút Page Stup để hiển thị hộp thoại chọn khổ giấy và kích thước trang in:



Bạn có thể thay đổi kích thước các lề in. Tôi thay như sau: Left – 2.5, ba thông số còn lại là 1.5. Nhấn OK để lưu lại thông số này.

Nhấn nút Print Preview để xem trước trang in:



Bạn có thể xem trong nhiều chế độ. Nhấn Close để đóng cửa sổ này lại. Vậy là chúng ta đã hoàn thành việc in ấn cho một tài liệu văn bản đơn giản.

4. Tổng kết chương 18

Bạn hãy làm bảng tổng kết những gì đã học trong chương 18. Bạn có thể áp dụng những ví dụ trong chương này cho các bài tập cần in ấn sau này của bạn.

Chương 19:

Làm quen với ADO.NET

--------oOo--------

Nội dung thảo luận:

- Sử dụng Server Explorer để thiết lập kết nối đến cơ sở dữ liệu
- Tạo bộ điều phối dữ liệu (data adapter) trích xuất thông tin trong csdl
- Sử dụng TextBox, Label và nút nhấn để hiển thị thông tin trong csdl
- Tạo tập dữ liệu dataset trình bày dl của một hay nhiều bảng trong csdl
- Sử dụng điều khiển duyệt các thông tin trong csdl

Trong chương này chúng ta sẽ sử dụng cách lập trình với ADO.NET để thao tác với csdl. Ta cũng tìm hiểu cách thao tác vói csdl bằng các điều khiển như TextBox, data adapter, label, button,…

Chú ý:

- ADO.NET là mô hình lập trình truy xuất dl chung cho tất cả các ngôn ngữ và chương trình Windows.
- Chúng ta dùng các thành phần như DataSet, DataAdapter để thao tác với csdl thay cho các thành phần cũ như Data Control và ADO Data Control.
- Định dạng dữ liệu trong ADO.NET tuân theo chuẩn XML nên dễ tích hợp với các ứng dụng web.

1. Lập trình với ADO.NET

Cơ sở dữ liệu rất quan trọng trong việc lưu trữ thông tin. Dữ liệu có rất nhiều nguồn và đa dạng. VB.NET được thiết kế với mục đích truy xuất, hiển thị, phân tích csdl. Với ADO.NET, bạn có thể truy xuất đến mọi hệ csdl theo cùng cách thức và mã chương trình như nhau.

1.1. Thuật ngữ về cơ sở dữ liệu

Chúng ta hãy làm quen với một số thuật ngữ về csdl trước khi thực sự thao tác với nó.

Csdl là một file tổ chức thông tin thành các bảng gọi là Table.

Mỗi bảng lại bao gồm nhiều hàng và cột. Cột thường được gọi là trường (field) và dòng được gọi là mẩu tin (record).

Mô hình truy xuất csdl trong ADO.NET có thể nói như sau: trước hết là thiết lập kết nối đến csdl. Tiếp theo đối tượng điều phối (data adapter) được tạo ra để truy vấn dl từ các bảng. Sau đó tạo các đối tượng DataSet chứa bảng dl bạn muốn trích dl.

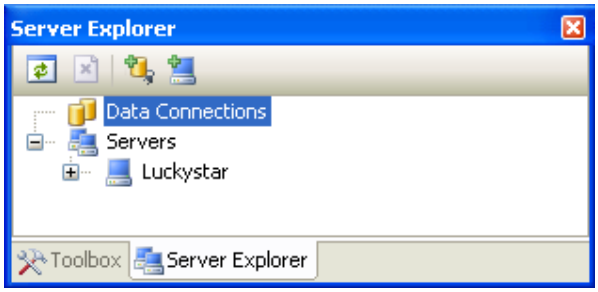
DataSet chỉ tạo bản sao của bảng dl mà thôi. Cuối cùng là gán thông tin trong DataSet vào các đối tượng hiển thị trên Form như TextBox, Label, Button, DataGrid,…

1.2. Làm việc với cơ sở dữ liệu Access

Trong phần tiếp theo chúng ta sẽ sử dụng Server Explorer để thiết lập kết nối đến csdl của MS Access có tên Students.mdb. Sau khi đã biết cách kết nối và đưa dữ liệu vào dataset, chúng ta sẽ bắt đầu xây dựng và tích hợp chúng vào giao diện của form.

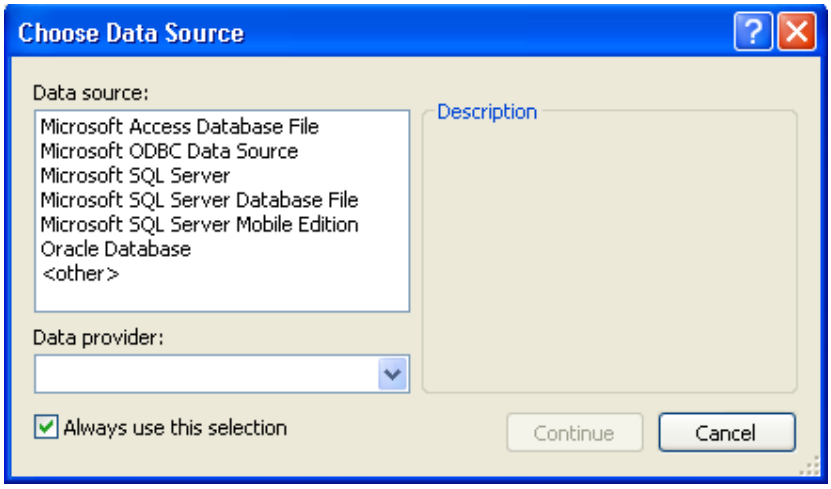
Bạn tạo mới một Solution có tên MyADOForm và thêm vào một dự án cùng tên.

Bạn chọn View | Server Explorer từ menu để hiện cửa sổ Server Explorer như hình:

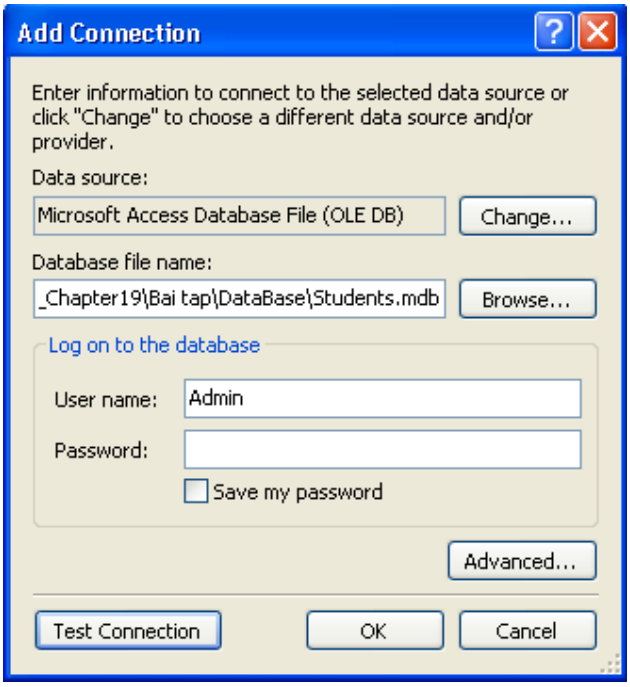


Đây là công cụ đồ họa cho phép kết nối đến csdl cục bộ, trên server theo mô hình client – server. Ta cũng có thể sử dụng nó để xem cấu trúc trong csdl, xem thuộc tính của bảng, kiểu dl của trường và mẩu tin trong csdl. Bạn có thể nắm kéo các kết nối và bảng dl trong cửa sổ này để tạo ra đối tượng dl cho chương trình.

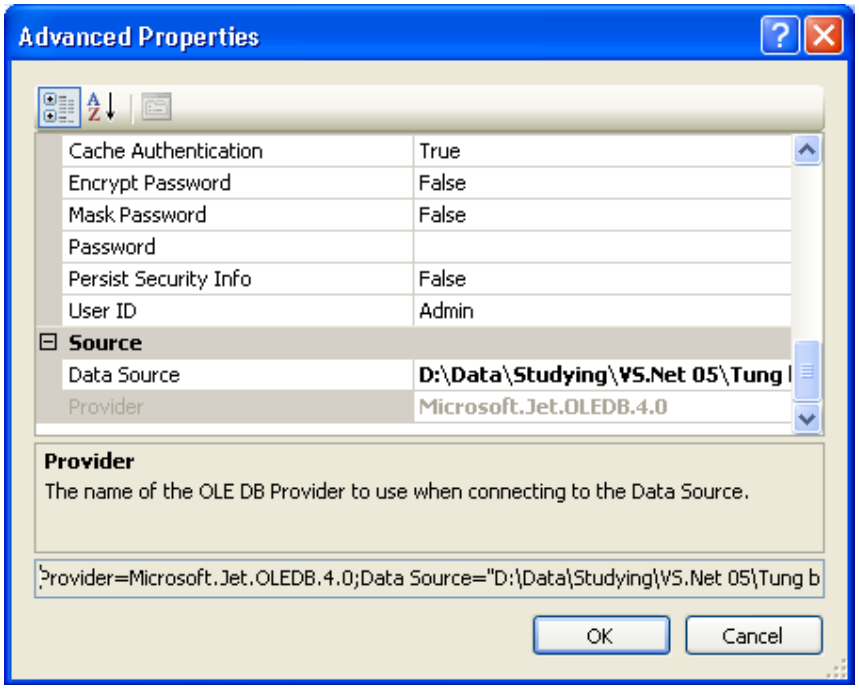
Tiếp theo bạn tạo kết nối đến csdl bằng cách click vào nút Connect To DataBase  trong cửa sổ Server Explorer. Một hộp thoại Choose Data Source hiện ra cho phép ta chọn nguồn dl như hình:



Bạn chọn Microsoft Access DataBase File và nhấn vào nút Continue để làm xuất hiện hộp thoại Add Connection như hình:



Bạn chọn đường dẫn đến csdl bằng cách nhắp vào nút Browse… và chọn csdl Students.mdb như hình. Bạn có thể kiểm tra xem kết nối có thành công không bằng cách click vào nút Test Connection, bạn cũng có thể tùy chỉnh kết nối bằng cách click vào nút Advanced:

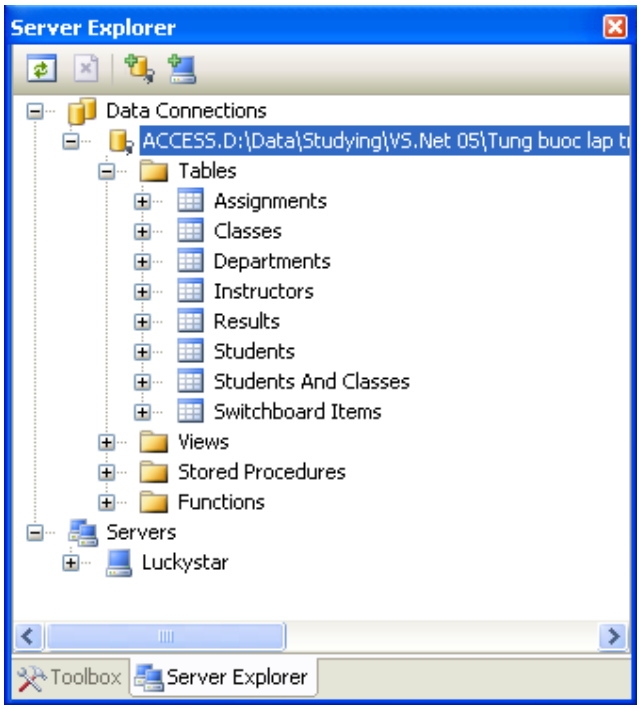


Bạn có thể thấy dòng mã kết nối ở ô cuối cùng như hình, dòng mã có nội dung: "**Provider=Microsoft.Jet.OLEDB.4.0;Data Source="D:\Data\Studying\VS.Net 05\Tu_**

ng buoc lap trinh vb.net\Tung buoc lap trinh vb.net\19_Chapter19\Bai tap\DataBase\Students.mdb"”

Nhấn OK để thêm kết nối vào Server Explorer.

Bạn có thể mở rộng tất cả các mục bằng cách click vào dấu (+) bên cạnh để mở rộng như hình:



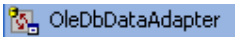
1.3. Tạo bộ điều phối dữ liệu Data Adapter

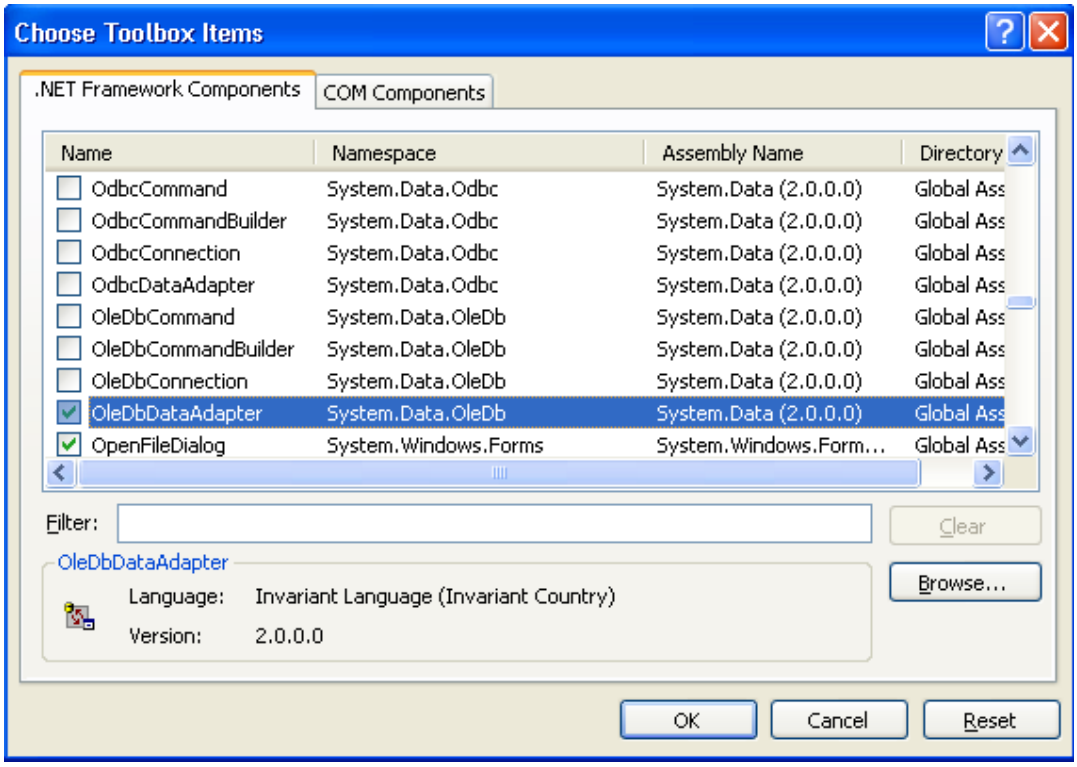
Bước hai trong thao tác csdl như ta đã biết đó là tạo bộ điều phối Data Adapter. Data Adapter sẽ định nghĩa chính xác những thông tin mà bạn muốn lấy trong csdl, là nền tảng để tạo DataSet.

VB.NET cung cấp rất nhiều cách tạo bộ điều phối. Cách đơn giản nhất là ta kéo các biểu tượng bảng trong Server Explorer vào cửa sổ form trong chế độ thiết kế. Ta cũng có cách thứ hai là dùng công cụ Data Adapter Configuration Winzard. Ta gọi đến công cụ này bằng cách chọn đối tượng OledbDataAdapter trên tab Data của ToolBox và đặt nó lên form. Trong bài tập này chúng ta sẽ sử dụng cách thứ hai này.

1.4. Sử dụng đối tượng điều khiển OleDbDataAdapter

Chọn tab Data trong cửa sổ ToolBox. Tab này chứa các điều khiển để thao tác với csdl. Trong tab này có hai đối tượng OleDbConnection và sqlConnection đều cho phép tạo kết nối đến csdl. Nhưng chúng ta đã kết nối bằng Server Explorer nên không cần hai đối tượng này nữa.

Kéo đối tượng OleDbDataAdapter  vào trong form. Nếu đối tượng này không xuất hiện, bạn có thể thêm nó vào bằng cách R-Click vào tab Data chọn Choose Item… để làm xuất hiện cửa sổ Choose ToolBox Items. Chọn tab .Net Framework Components và chọn OleDbAdapter như hình:



Nhắp OK để hoàn thiện việc thêm Item này cho ToolBox. Bạn cũng có thể làm tương tự với các đối tượng khác.

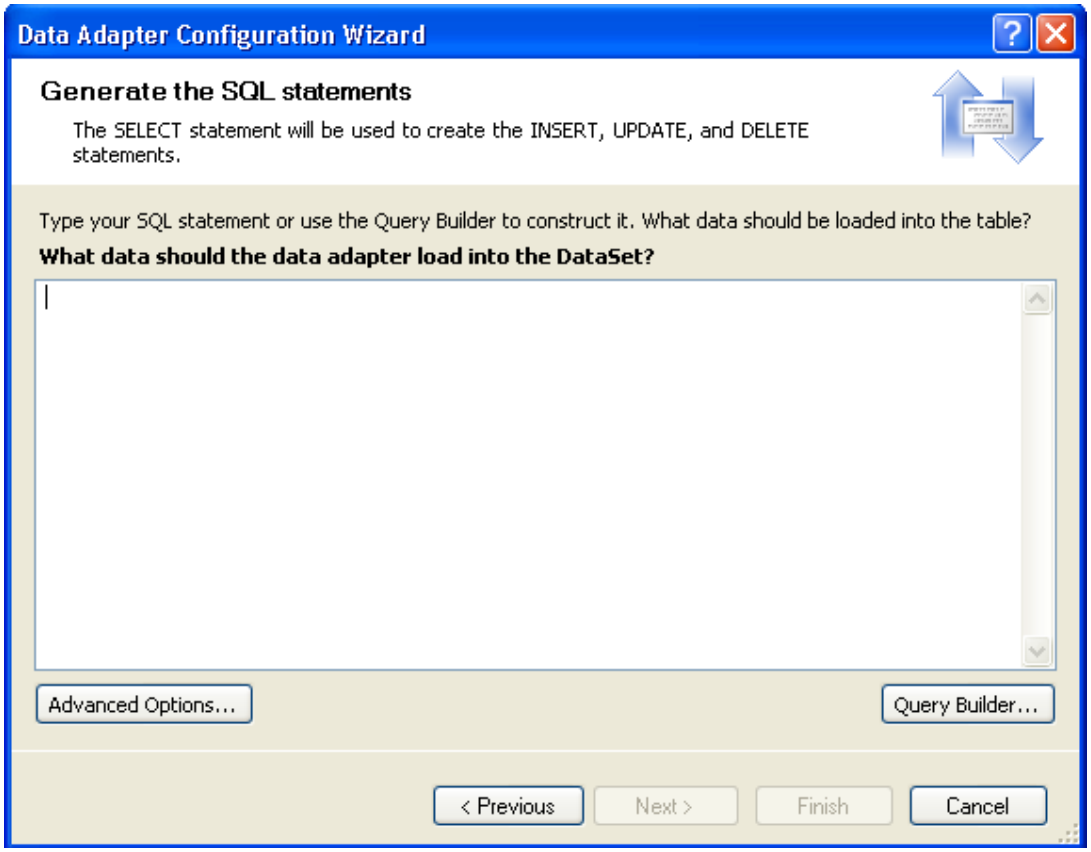
OleDbAdapter được thiết kế để kết nối đến csdl Access. Khi kéo thả đối tượng này vào form thì VS.NET sẽ tao trình Data Adapter Configuration Winzard. Một màn hình khởi đầu, bạn nhấn Next để chuyển sang màn hình thứ hai:

Bạn nhấn Next hai lần để xuất hiện màn hình soạn thảo câu lệnh SQL như hình H.1 dưới.

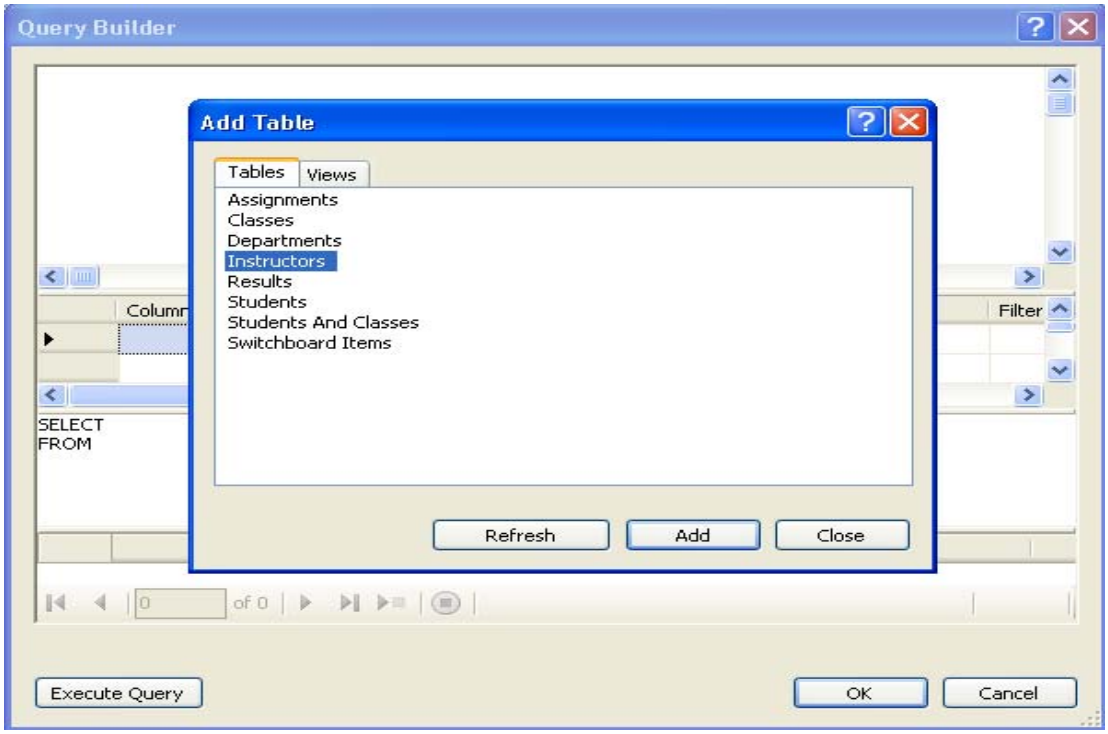
Nếu bạn chưa biết đến các câu lệnh SQL, có thể nhấn vào nút Query Builder… để VS liệt kê các bảng của csdl để bạn chọn.

Bạn hãy nhấn vào bảng Instructors như hình H.2 và nhấn Add, rồi ấn Close để đóng cửa sổ này lại.

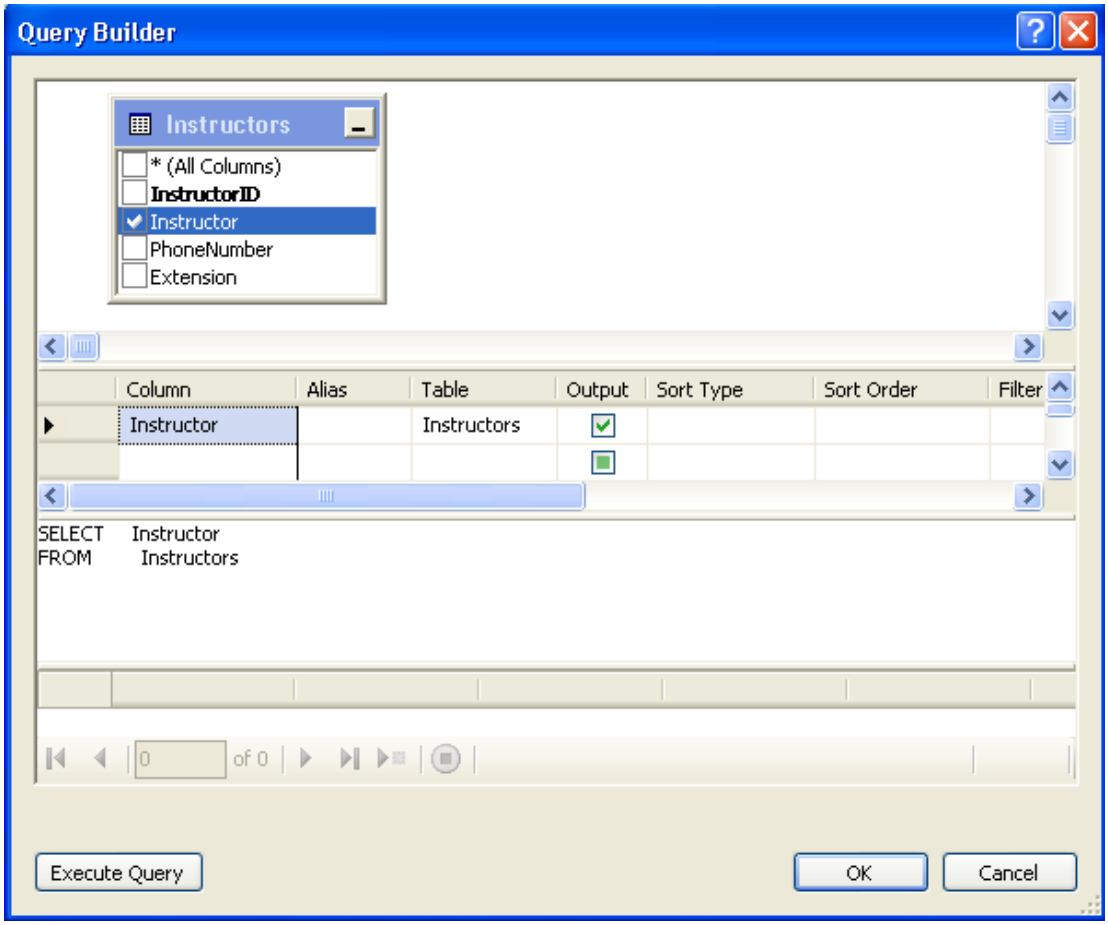
Bạn thấy trong bảng Instructors có các ô CheckBox tương ứng với các trường. Query sẽ tạo câu lệnh tương ứng để rút thông tin của bảng. Trong bài tập này chúng ta chỉ rút thông tin từ một cột trong bảng. Bạn nhấn vào cột Instructor để chọn nó như hình H.3 và nhấn OK. Chúng ta đã tạo xong câu lệnh SQL để rút dữ liệu.



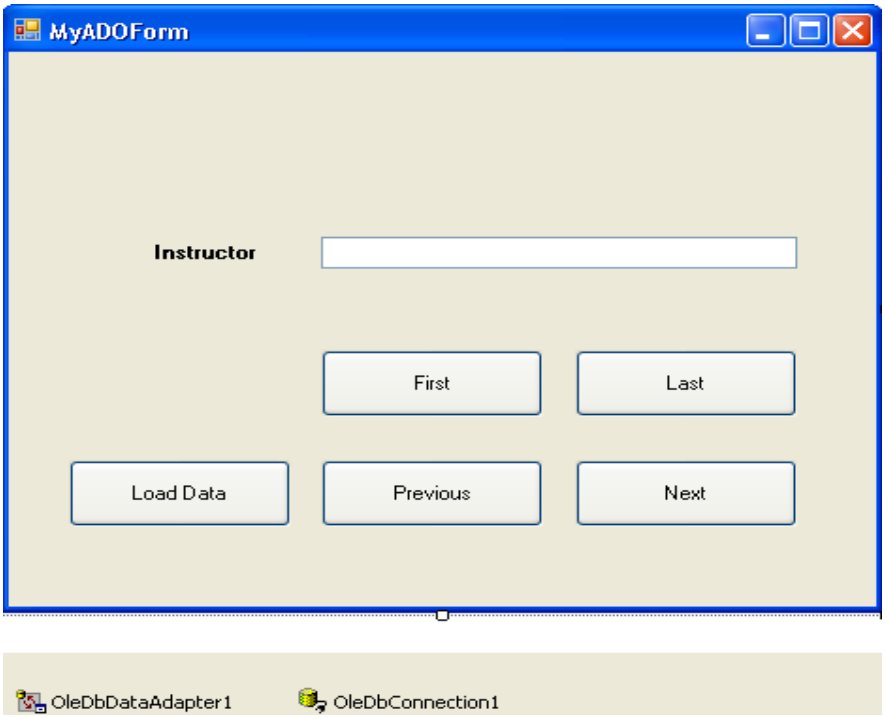
H.1. Cửa sổ soạn thảo mã SQL



H.2. Query Builder



H.3. Chọn trường để xây dựng câu lệnh SQL



H.4. Giao diện Form

Sau khi nhấn OK, một cửa sổ Generate The SQL Statement hiện ra hiển thị câu lệnh SQL ta vừa tạo. Bạn nhấn Finish đê hoàn thành việc tạo đối tượng điều phối. Lúc này giao diện có dạng như hình H.4

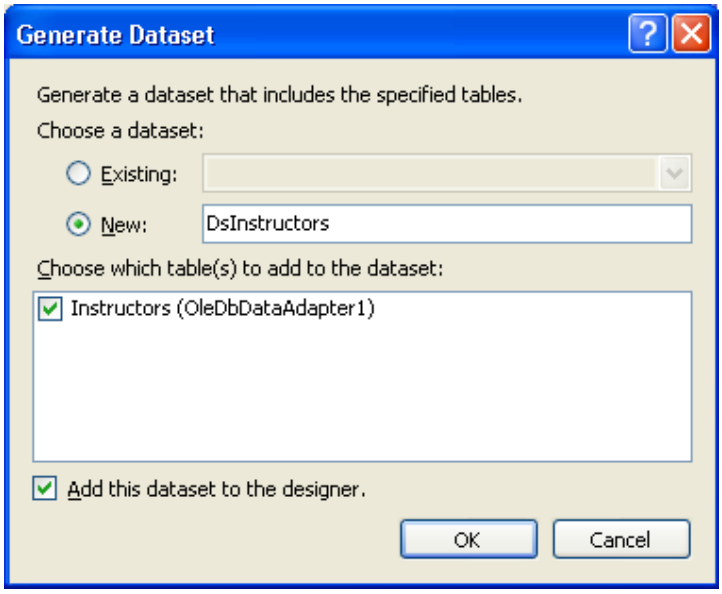
1.5. Làm việc với DataSet

Tiếp theo ta tạo ra đối tượng trình diễn dữ liệu cho người dùng thao tác. Đối tượng này là DataSet. Nó là hình ảnh có được từ DataAdapter. Nó chỉ là ảnh của csdl nên mọi thao tác của người dùng sẽ chưa ảnh hưởng đến csdl cho đến khi có yêu cầu cập nhật.

Trong phần tiếp theo của bài tập này chúng ta sẽ tạo đối tượng DataSet trình diễn thông tin trong cột Instructor của bảng Instructors trong csdl Students.mdb.

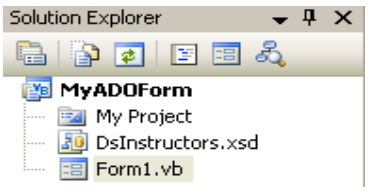
Bạn nhấp chuột lên form1 để chọn nó. Nếu không chọn nó thì các lệnh tạo DataSet sẽ không hiển thị trên menu.

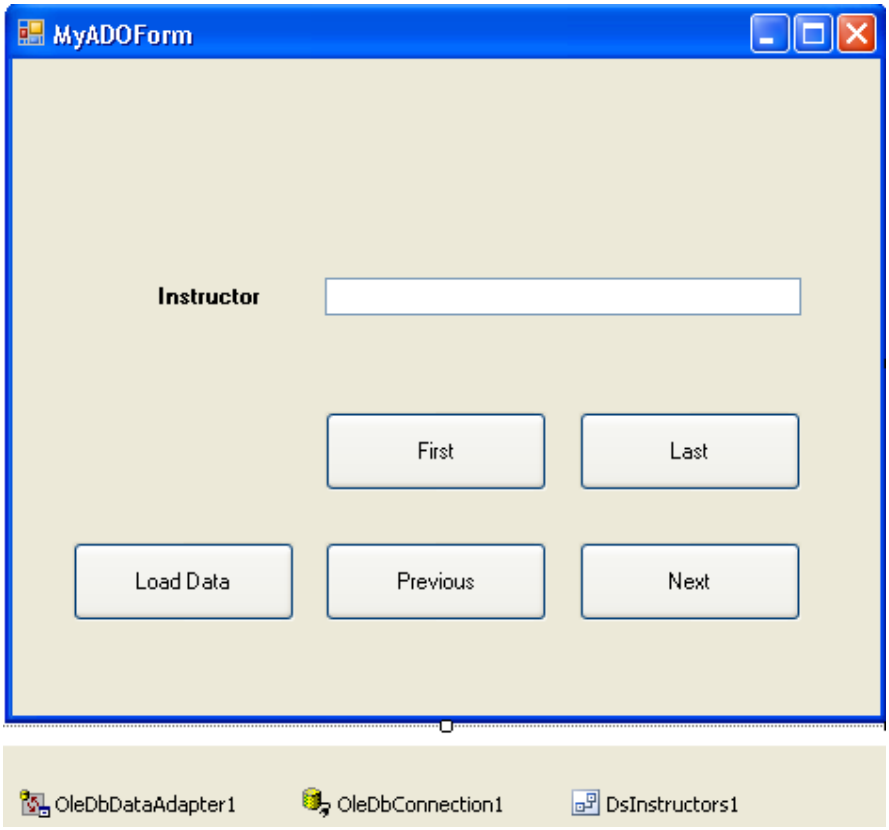
Chọn Data | Generate DataSet từ menu để làm xuất hiện hộp thoại Generate DataSet như hình:



Bạn đặt tên nào tùy thích tại ô New, mình chọn là DsInstructors. Chọn ô checkBox Add this dataset to the designer để VS đưa dataser vào khay công cụ.

Nhấn OK và đối tượng DataSet DsINstructors được tạo trên khay công cụ. Lúc này VB.NET sẽ tự thêm vào một file có tên DsInstructors.xsd trong cửa sổ Solution Explorer. File này chứa các thông tin về dữ liệu theo khuôn dạng XML:





H.5. Đối tượng DsInstructors1 được tạo trên khay hệ thống

Tiếp theo ta sẽ học cách để hiển thị dữ liệu trong dataset này lên form.

2. Sử dụng các điều khiển ràng buộc dữ liệu

Tiếp theo ta sẽ dùng các điều khiển quen thuộc như Textbox, Label, Button để trình bày cơ sở dữ liệu lên form. Để trình bày được như thế ta cần phải làm một thao tác gọi là ràng buộc dữ liệu (data binding), nghĩa là dữ liệu hiển thị lên trong các điều khiển sẽ phụ thuộc vào nguồn dữ liệu có trong DataSet hay DataAdapter.

Bạn có thể ràng buộc dữ liệu với các điều khiển sau: TextBox, Label, ListBox, ComboBox, RadioButon, DataGrid và PictureBox. Trong đó đặc biệt và hữu ích nhất có lẽ là DataGrid vì nó cho phép bạn hiển thị toàn bộ nội dung của DataSet.

Trong bài tập này, chúng ta sẽ ràng buộc dữ liệu vào TextBox để hiển thị thông tin trong bảng Instructors của csdl Students.mdb.

Bạn thiết kế giao diện form như hình trên. Trong đó thuộc tính của các điều khiển như sau:

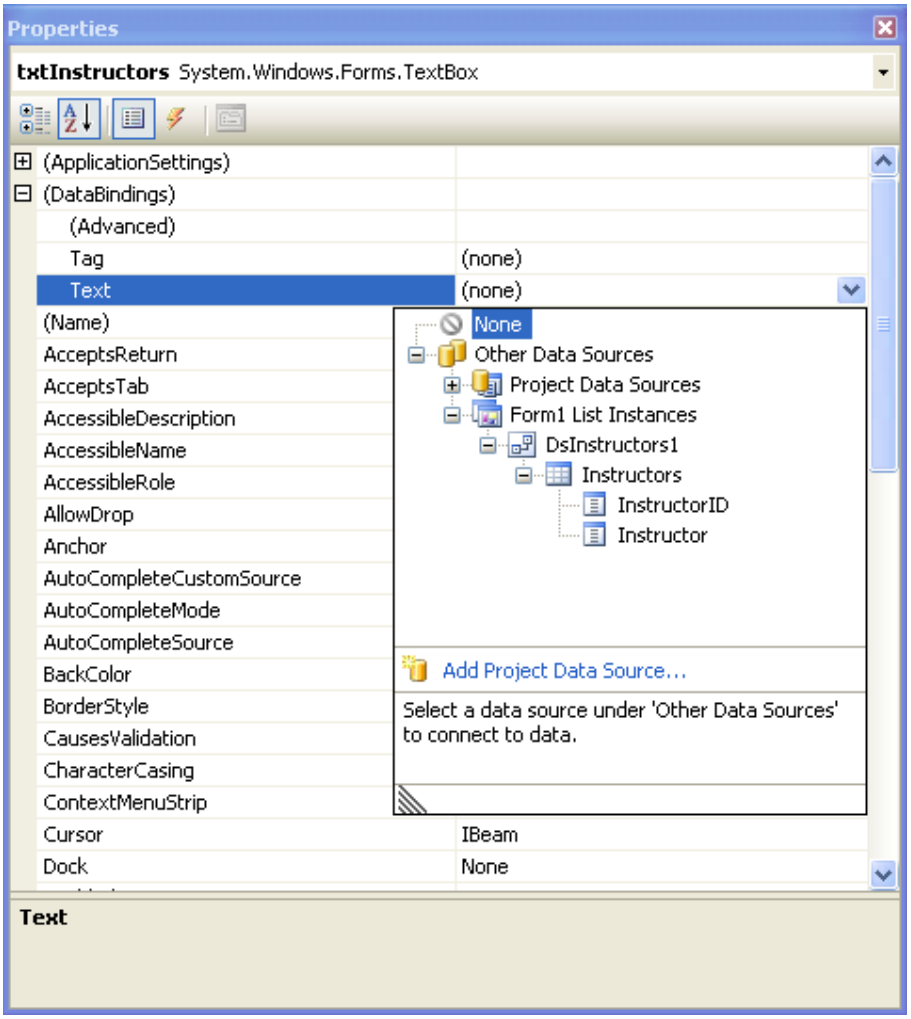
- Button First: Name – btnFirst, enable – False
- Button Last: Name – btnLast, enable – False
- Button Next: Name – btnNext, enable – False
- Button Previous: Name – btnPrevious, enable – False

- Button Load Data: Name – btnLoadData
- TextBox1: Name - txtInstructors

Các điều khiển còn lại có thuộc tính như hình.

Bây giờ ta sẽ tiến hành ràng buộc dữ liệu là các trường (cột dữ liệu – field) vào textbox txtInstructors.

Để làm điều này, bạn chọn ô textbox và mở Properties của nó ra. Click vào dấu (+) bên cạnh nhánh thuộc tính DataBindings và chọn ô text, Click vào nút mũi tên đi xuống và bạn có thể nhìn thấy nguồn dữ liệu DsInstructors1 hiển thị trong danh sách:



Nhấn chọn cột Instructor để chỉ định trường này sẽ hiển thị trong ô textbox txtInstructor. Như vậy ta đã ràng buộc xong, tiếp theo cần viết mã để xuất dữ liệu khi chương trình thực thi.

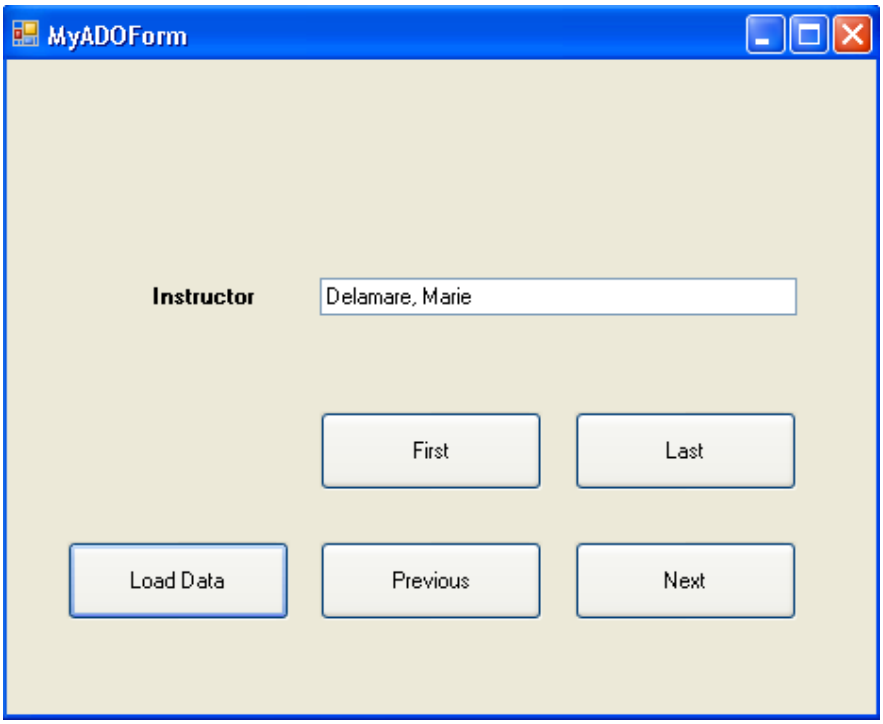
Để làm được điều đó, chúng ta tạo thủ tục btnLoadData_Click bằng cách trở lại cửa sổ thiết kế form và double click vào nút Load Data rồi nhập đoạn mã sau:

```
DsInstructors1.Clear()
OleDbDataAdapter1.Fill(DsInstructors1)

btnFirst.Enabled = True
btnLast.Enabled = True
btnNext.Enabled = True
btnPrevious.Enabled = True
```

Chúng ta viết mã để xóa sạch dữ liệu mà DataSet DsInstructors1 nắm giữ trước đây. Tiếp theo khiến bộ điều phối DataAdapter1 điền dữ liệu vào đối tượng DataSet DsInstructors1 mà chúng ta đã tạo ra ở bước 3 bằng phương thức Fill().

Chạy thử chương trình:

Bạn nhấn F5 để kiểm thử chương trình. Khi chương trình chạy, bạn nhắp vào nút Load Data để chương trình hiển thị bản ghi đầu tiên của trường Instructor trong bảng dữ liệu:



Tiếp theo chúng ta sẽ mở rộng một số chức năng khác của ứng dụng cơ sở dữ liệu thuần túy như duyệt qua các bản ghi, đếm và hiển thị số bản ghi hiện hành.

Như vậy quá trình thao tác csdl có thể tóm tắt như sau: thứ nhất, tạo kết nối đến csdl cần truy xuất; thứ hai tạo đối tượng điều phối DataAdapter; thứ ba, tạo đối tượng trình diễn DataSet; cuối cùng là ràng buộc dữ liệu vào các điều khiển cho phép ràng buộc. Nếu trong các bài tập yêu cầu cập nhật, thống kê, tìm kiếm, … thì còn có bước nữa là tiến hành xử lý các thao tác cập nhật, thống kê, tìm kiếm, … theo yêu cầu của bài.

3. Tạo các điều khiển duyệt xem dữ liệu

Trong bài tập này, chúng ta mới chỉ dừng lại ở việc ràng buộc dữ liệu và hiển thị được bản ghi đầu tiên vào ô textbox mà thôi. Trong phần tiếp theo chúng ta sẽ tạo ra các nút cho phép duyệt qua các bản ghi khác nhau, xem bản ghi đầu tiên cũng như cuối cùng.

ADO.NET cho phép quản lý và duyệt qua các bản ghi (record) bằng đối tượng CurrentManager. Với đối tượng này bạn có thể biết được vị trí hiện hành, đi đến mẩu tin sau cùng hay trở về mẩu tin đầu tiên cũng như đến mẩu tin kế tiếp hay ở trước. Mỗi DataSet đều có sẵn đối tượng CurrentManager và mỗi đối tượng form đều có thuộc tính BindingContext theo dõi tất cả đối tượng CurrentManager trên form.

Bây giờ trở lại bài tập của chúng ta. Trong phần trước chúng ta đã tạo ra bốn nút nhấn mang tên First, Last, Next, Previous. Giờ chúng ta sẽ viết mã cho chúng sử dụng đối tượng BindingContext, CurrentManager để duyệt qua các bản ghi.

Trước hết tạo thủ tục btnFirst_Click với nội dung như sau:

```
Me.BindingContext(DsInstructors1, _
"Instructors").Position = 0
btnFirst.Enabled = False
btnNext.Enabled = True
btnLast.Enabled = True
```

Cú pháp này hiển thị bản ghi đầu tiên của DsInstructors1 sử dụng đối tượng BindingContext. Nó gán giá trị 0 cho thuộc tính Position để con trỏ hiện hành của dữ liệu chuyển đến bản ghi đầu tiên.

Tạo thủ tục btnLast_Click và nhập đoạn mã sau:

```
'Đếm tổng số bản ghi
Dim tongsobanghi As Integer = Me.BindingContext _
(DsInstructors1, "Instructors").Count
'Chuyển con trỏ đến bản ghi cuối cùng
Me.BindingContext(DsInstructors1, _
"Instructors").Position = tongsobanghi - 1
btnLast.Enabled = False
btnFirst.Enabled = True
btnPrevious.Enabled = True
btnNext.Enabled = False
```

Tạo thủ tục btnNext_Click và nhập vào đoạn mã sau:

```
'Đếm số bản ghi hiện hành
Dim tongsobanghi As Integer = Me.BindingContext _
(DsInstructors1, "Instructors").Count
'Nếu chưa phải là bản ghi cuối thì next lên 1
If Me.BindingContext(DsInstructors1, _
"Instructors").Position < tongsobanghi - 1 Then
    Me.BindingContext(DsInstructors1, _
    "Instructors").Position += 1
    btnFirst.Enabled = True
    btnPrevious.Enabled = True
    btnLast.Enabled = True
```

```
    Else
        btnNext.Enabled = False
        btnLast.Enabled = False
        btnFirst.Enabled = True
        btnPrevious.Enabled = True
    End If
```

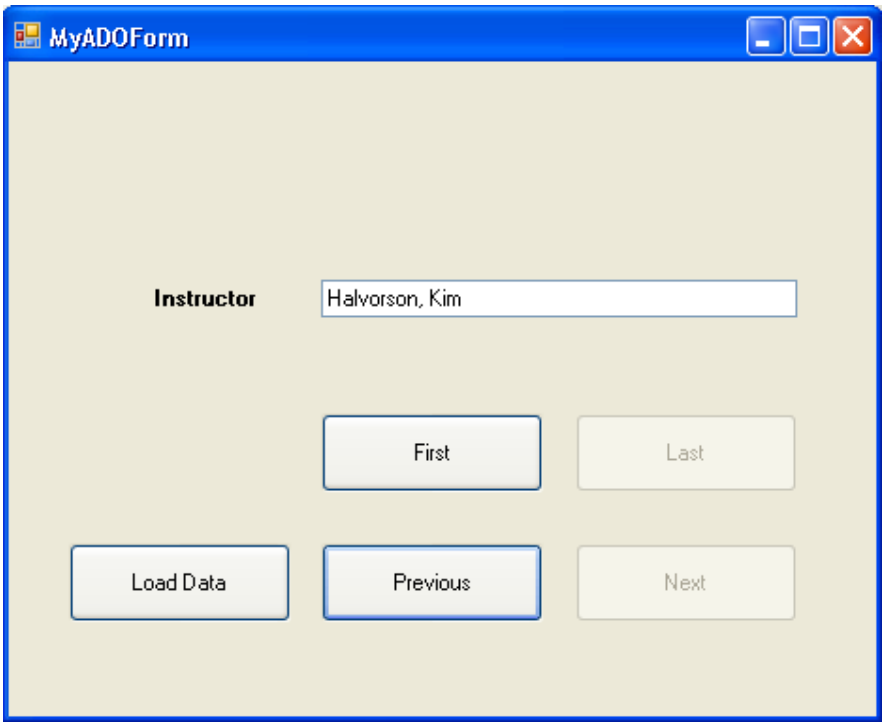
Thủ tục btnPrevious_Click:

```
    'Nếu chưa phải là bản ghi đầu thì lùi lại 1
    If Me.BindingContext(DsInstructors1, _
    "Instructors").Position > 0 Then
        Me.BindingContext(DsInstructors1, _
        "Instructors").Position -= 1
        btnFirst.Enabled = True
        btnLast.Enabled = True
        btnNext.Enabled = True
    Else
        btnFirst.Enabled = False
        btnPrevious.Enabled = False
    End If
```

Vậy là chúng ta đã tạo xong các nút cho phép duyệt qua các bản ghi. Bây giờ chúng ta chạy thử chương trình.

Chạy chương trình:

Bạn nhấn F5 để chạy chương trình. Ấn nút Load Data để hiển thị dữ liệu vào textbox. Ấn các phím để duyệt qua các bản ghi trong cơ sở dữ liệu.



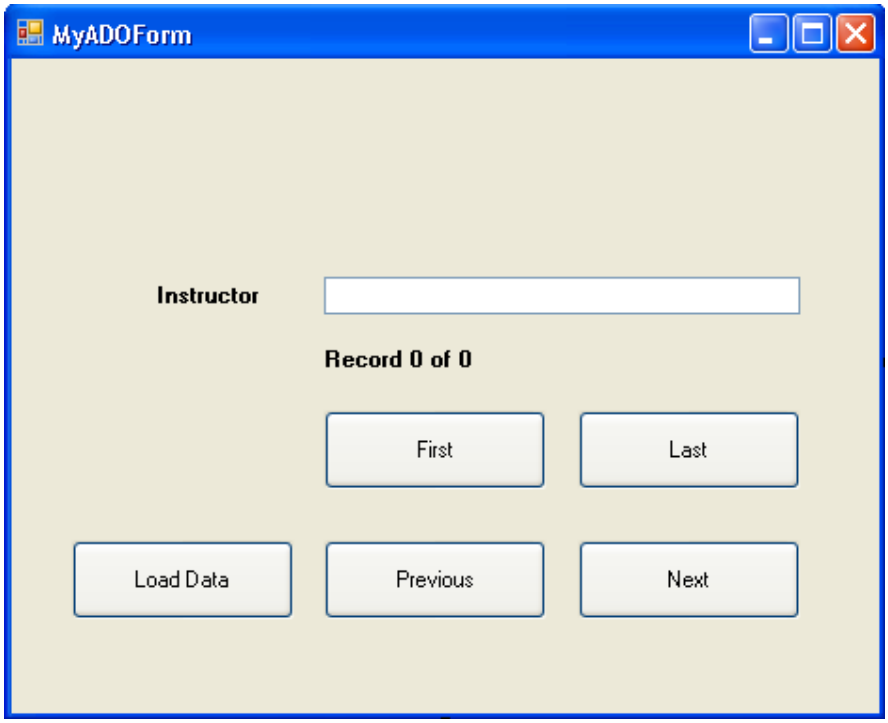
Bạn nhấn nút Close ở góc phải trên của form để đóng chương trình lại.

Bây giờ để cụ thể hơn nữa chúng ta sẽ tạo điều khiển label cho hiển thị vị trí bản ghi hiện hành để người dùng tiện quan sát.

4. Hiển thị vị trí của bản ghi hiện hành

Ngoài việc cung cấp cơ chế duyệt xem các bản ghi, ta cũng cần cho người dùng biết đó là bản ghi thứ mấy. Bây giờ chúng ta sẽ thêm một nhãn Label để hiển thị thứ tự của bản ghi.

Bạn mở thiết kế form và thêm vào một nhãn label1 có thuộc tính Name là lblIndexOfRecord, thuộc tính Text của nhãn là "Record 0 of 0". Giao diện như hình:



Ta tạo một thủ tục có tên count() ở ngay dưới phát biểu khai báo form1 như sau:

```
Private Sub Count()
    Dim tongsobanghi, banghihienhanh As Integer
    tongsobanghi = Me.BindingContext _
    (DsInstructors1, "Instructors").Count
    banghihienhanh = Me.BindingContext _
    (DsInstructors1, "Instructors").Position + 1
    lblIndexOfRecord.Text = "Record " & _
    banghihienhanh.ToString & "Of " & tongsobanghi.ToString
End Sub
```

Thủ tục này sẽ gán thuộc tính count của đối tượng BindingContext vào biến *tongsobanghi* và thuộc tính *Position* của nó cho biến *banghihienhanh* nhưng cộng thêm 1 vì thứ tự bản ghi trong bảng dữ liệu được tính từ 0. Sau đó hai giá trị của hai biến trên được gán cho thuộc tính Text của điều khiển Label lblIndexOfRecord.

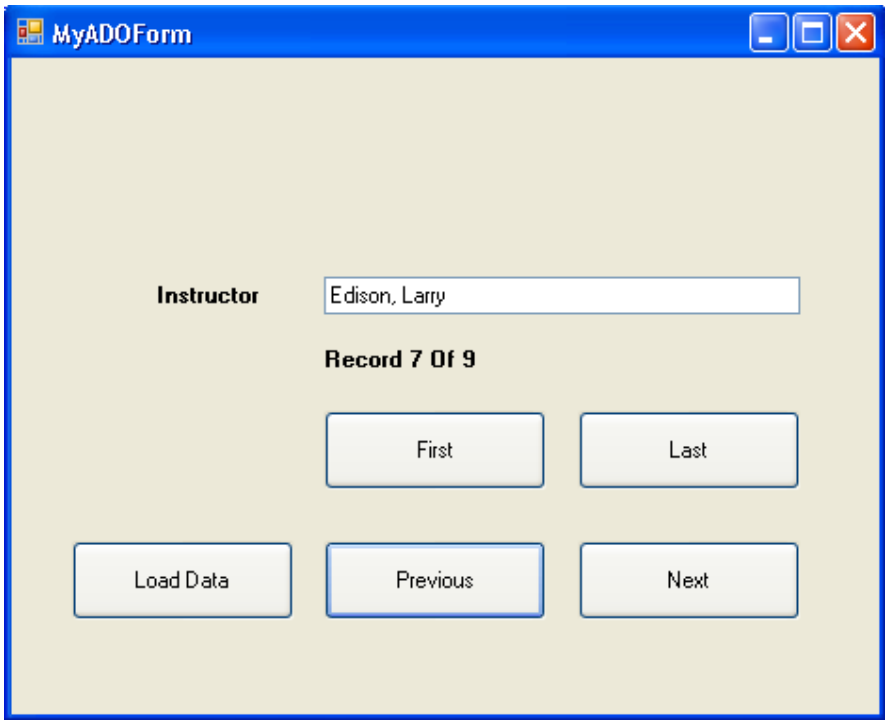
Để thủ tục này phát huy tác dụng thì bạn sẽ thêm lời gọi thủ tục này trong các thủ tục khác như btnFirst_Click, btnLast_Click, btnPrevious_Click, btnNext_Click như sau:

```
Count()
```

Chương trình của chúng ta đến đây là hoàn thiện. Bạn có thể chạy thử để kiểm tra.

Chạy chương trình:

Bạn nhấn F5 để chạy chương trình. Ấn nút Load Data để hiển thị dữ liệu. Sau đó bạn hãy nhấn các nút di chuyển để duyệt qua các bản ghi và xem thứ tự của bản ghi đó trong bảng dữ liệu, kết quả:



5. Tổng kết chương 19

Bạn làm bảng tổng kết những gì đã học. Tổng kết một lần nữa các bước để có thể trình diễn dữ liệu trong form.

Đây mới chỉ là kỹ thuật lập trình đơn giản nhất của ADO.NET, trong phần sắp tới chúng ta sẽ học về DataGrid để trình diễn dữ liệu ở mức độ cao hơn.

Chương 20:

Trình diễn dữ liệu sử dụng điều khiển DataGrid

--------oOo--------

Nội dung thảo luận:

- Tạo đối tượng DataGrid trên form và sử dụng để hiển thị các bản ghi trong csdl
- Sắp xếp dữ liệu các bản ghi theo cột
- Thay đổi định dạng và màu sắc của các ô trong khung lưới dữ liệu DataGrid

DataGrid là đối tượng trình diễn dữ liệu rất hiệu quả. Nó có dạng khung lưới cho phép tình diễn toàn bộ nội dung của tập dữ liệu DataSet.

Chú ý:

- Đối tượng DataGrid cho phép trình diễn dữ liệu theo dạng khung lưới như excel.
- Bạn không cần thêm các lệnh xử lý phụ với DataGrid. Tất cả dữ liệu được quản lý bởi DataAdapter và DataSet ở tầng dưới.

1. Sử dụng DataGrid để hiển thị dữ liệu trong bảng

Trong phần này chúng ta sẽ dùng DataGrid để hiển thị dữ liệu của bảng trong csdl Students.mdb. Ta sẽ điền đầy đủ nội dung khung lưới bằng dữ liệu của bảng ở dạng chuỗi sau đó thực hiện một số thao tác định dạng, sắp xếp và ghi lại những thay đổi trong DataGrid trở lại csdl.

Cũng giống như TextBox, bạn có thể ràng buộc dữ liệu trong DataSet vào DataGrid. Việc ràng buộc này thông qua hai thuộc tính là DataSource và DataMember.

Trong bài tập MyDataGridBinding sau chúng ta sẽ đưa toàn bộ nội dung của bảng Instructors có trong DsInstructors1 hiển thị trong khung lưới DataGrid.

Bài tập MyDataGridBinding:

Bạn tạo mới một Solution và thêm vào một dự án cùng tên là MyDataGridBinding.

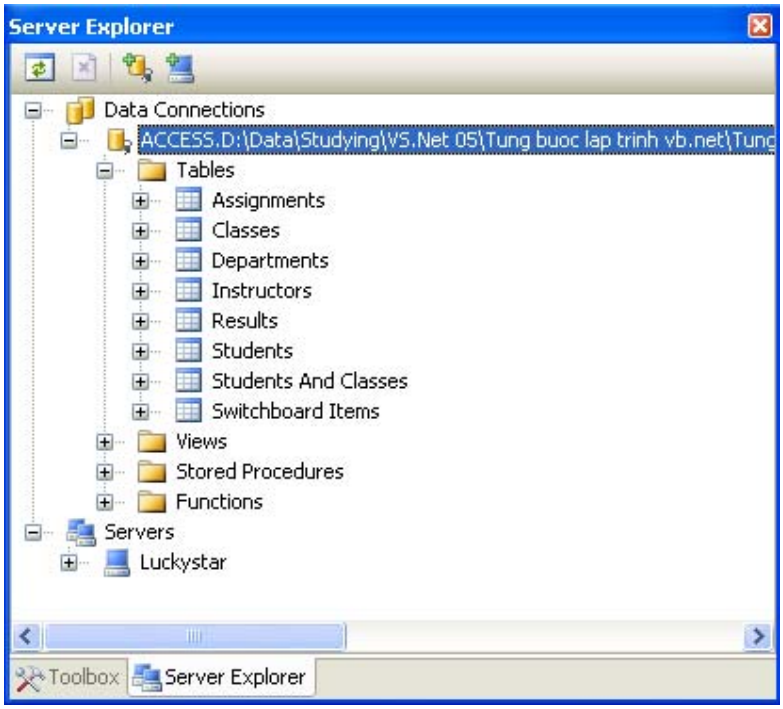
Kết nối cơ sở dữ liệu:

Nếu trong bài trước chúng ta đã hoàn thành kết nối với csdl thì bây giờ trong cửa sổ Server Explorer sẽ có một kết nối đến csdl đó nhưng có thêm một gạch đỏ ở kết nối đó. Nếu muốn sử dụng lại kết nối này bạn chỉ việc ấn vào nút Refresh  là xong. Trong bài tập này tôi chép file csdl Students.mdb vào cùng thư mục với dự án để tiện thao tác.

Bạn chọn nút  để thực hiện kết nối đến csdl như đã biết. Chọn csdl mà chúng ta vừa chép vào thư mục chứa dự án.

Nhấn OK để hoàn thành kết nối.

Bạn có thể xem chi tiết các bảng có trong csdl này bằng cửa sổ Server Explorer:



Tạo đối tượng điều phối DataAdapter:

Bạn tạo thêm đối tượng OleDbDataAdapter vào trong form bằng cách kéo nó từ ToolBox ở tab data vào trong form. Khi đó một cửa sổ Data Adapter Configuration xuất hiện.

Nhấn Next hai lần để hiện cửa sổ Generate SQL Statements. Tại đây bạn có thể tự gõ câu lệnh SQL hay sử dụng nút nhấn Query Builder… Ở đây mình dùng cách nhập trực tiếp câu lệnh SQL. Bạn nhập câu lệnh sau:

```
SELECT    Extension, PhoneNumber, Instructor, InstructorID
FROM      Instructors
```

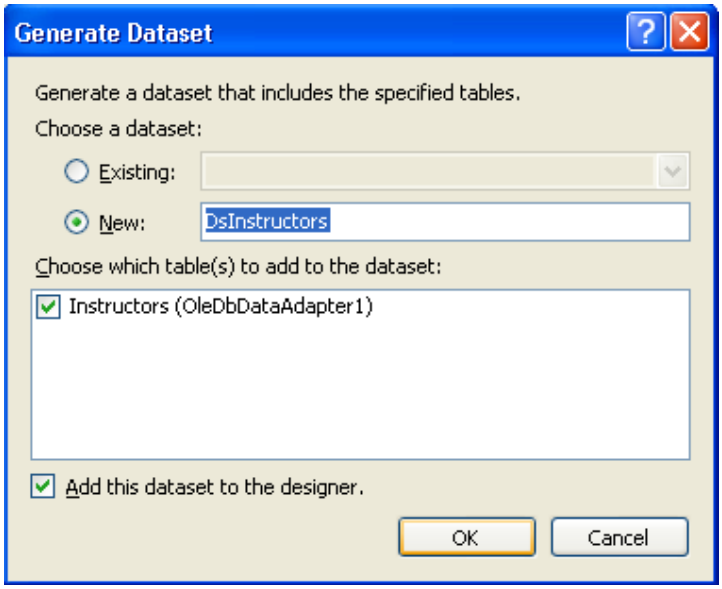
Phát biểu này sẽ trích rút dữ liệu ở cả bốn trường trong bảng Instructors. Bạn nhấn Next để xem kết quả của Winzard. Lúc này, trình Winzard tự tạo ra các câu lệnh khác là Update (cập nhật), Select, Insert (chèn), Delete (xóa).

Nhấn Finish để kết thúc quá trình xây dựng tạo đối tượng điều phối DataAdapter có tên OleDbDataAdapter1.

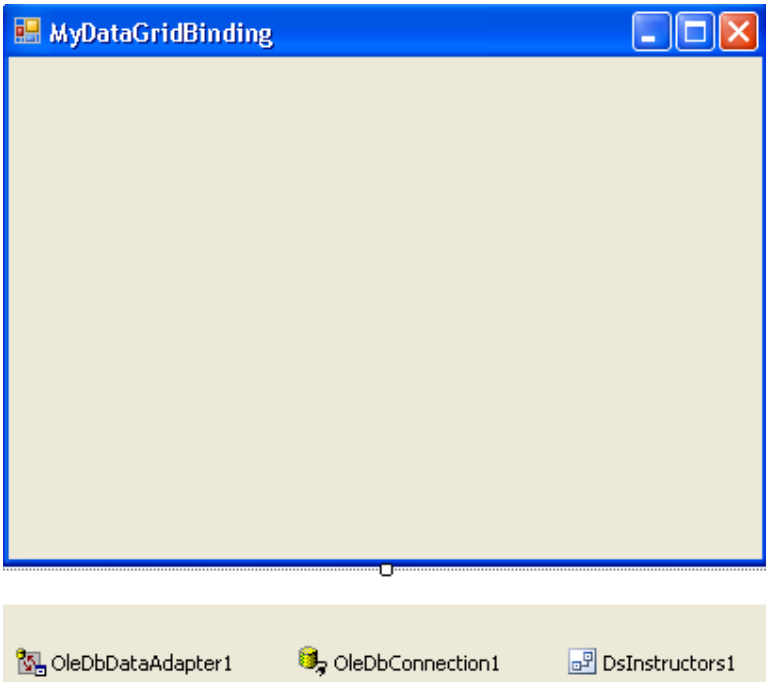
Tạo đối tượng trình diễn DataSet:

Nhấn Form để chọn nó.

Chọn Data | Generate DataSet từ menu làm hiện hộp thoại Generate DataSet như đã biết. Tại ô New bạn nhập vào tên DsInstructors và đánh dấu vào ô checkBox Add this DataSet To The Designer để VS tạo ra đối tượng DataSet và đưa nó vào khay hệ thống như hình:



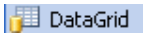
Nhấn OK để VS tạo đối tượng DataSet cho bảng Instructors trong csdl Students.mdb. Lúc này cửa sổ form có thêm các đối tượng như hình:



Chúng ta đã hoàn thành ba bước đầu của thao tác với csdl. Bây giờ chúng ta sử dụng DataGrid để trình bày dữ liệu.

Tạo đối tượng DataGrid:

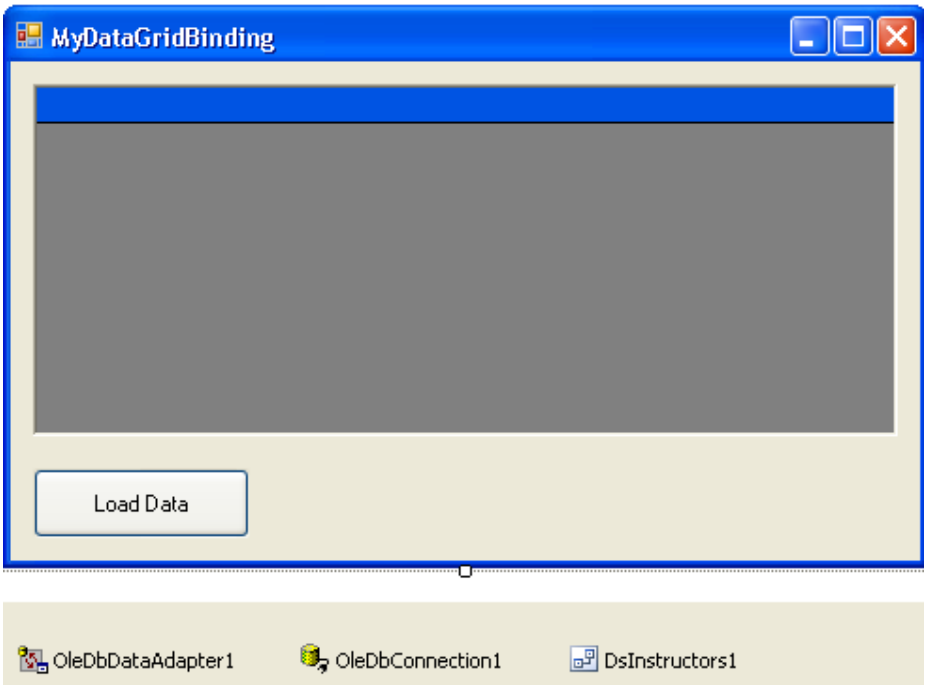
Kéo form cho kích thước rộng ra để chứa đủ khung lưới DataGrid với 4 cột và 10 dòng.

Đưa điều khiển DataGrid 

trên ToolBox vào trong form. Kéo chiều dài của nó cho phù hợp với chiều kích thước của form.

Tạo thêm một nút nhấn nữa vào form. Đặt thuộc tính Name là btnLoad và text là "Load Data".

Mở Properties của DataGrid và đặt thuộc tính Anchor của nó là cả Left, Right, Top, Bottom. Giao diện của form lúc này như hình:



Tiếp theo ta sẽ dùng thuộc tính DataSource và DataMember để ràng buộc dữ liệu trong DsInstructors1 vào khung lưới DataGrid.

Bạn cho hiển thị các tùy chọn của thuộc tính DataSource trong cửa sổ Properties. Một chương trình có thể có rất nhiều DataSet nhưng tại một thời điểm khung lưới chỉ có thể thể hiện một DataSet mà thôi. Bạn chọn DsInstructors1 như hình H.1.

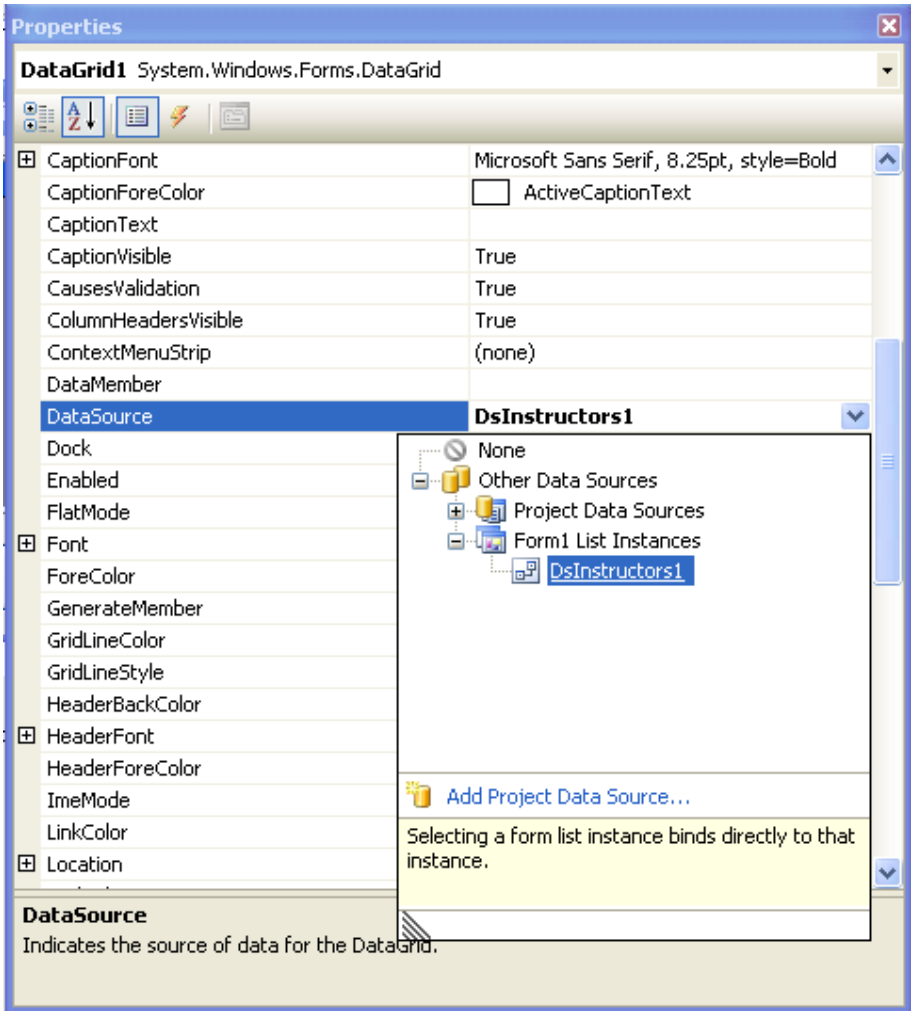
Tiếp theo bạn chọn thuộc tính DataMember là Instructors như hình H.2.

Ngay sau khi bạn chọn xong hai thuộc tính DataSource và DataMember thì khung lưới sẽ hiển thị các cột dữ liệu dù chưa có dòng dữ liệu nào hiển thị. Dữ liệu sẽ được đưa vào khung lưới khi chương trình thực thi.

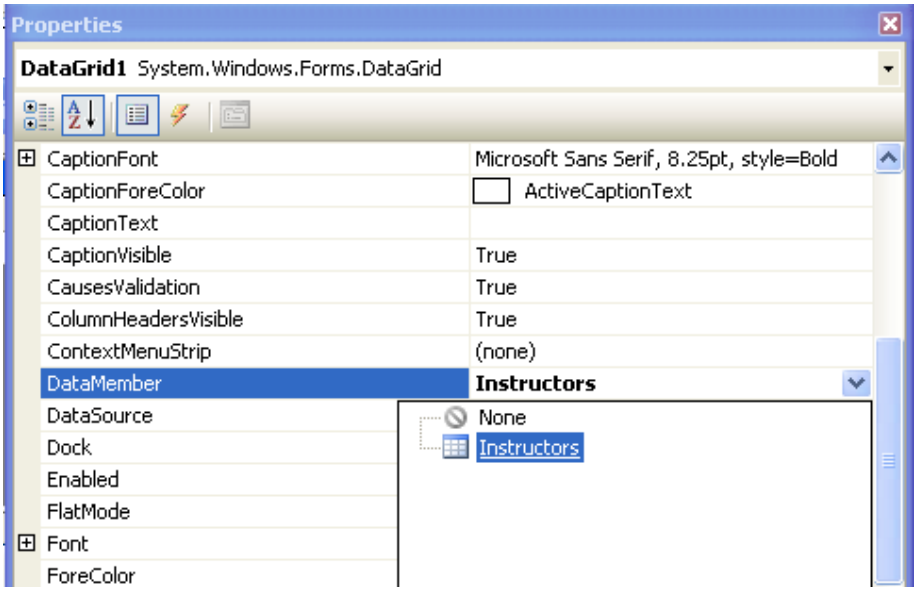
Bạn chọn nút Load Data và đặt thuộc tính Anchor của nó là Bottom, Left.

Lúc này giao diện form thiết kế sẽ như hình H.3.

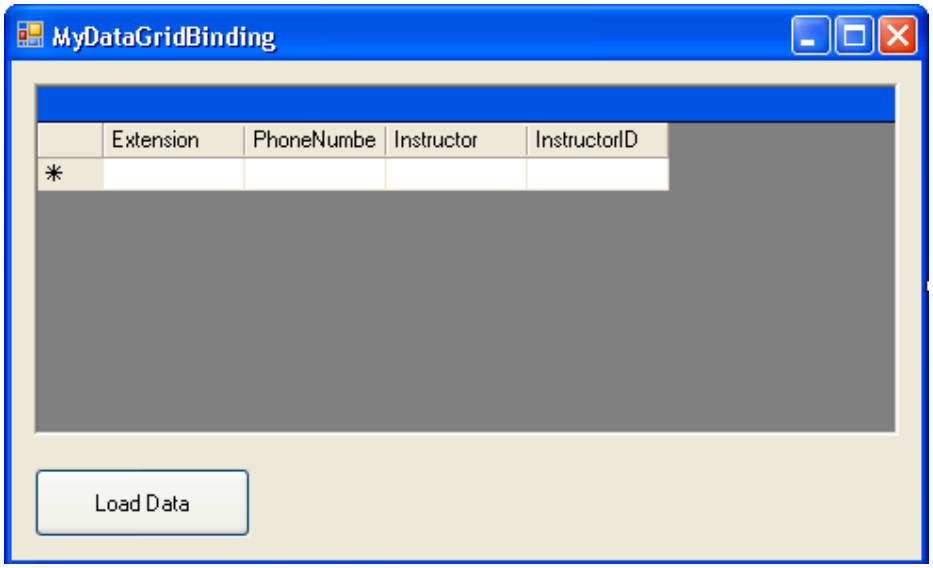
Tiếp theo chúng ta cần viết mã để đổ dữ liệu vào khung lưới bằng phương thức Fill như bạn đã biết trong chương trước.



H.1. Chọn DsInstructors1 cho thuộc tính DataSource



H.2. Chọn Instructors cho thuộc tính DataMember



H.3. Cửa sổ form khi thiết kế xong

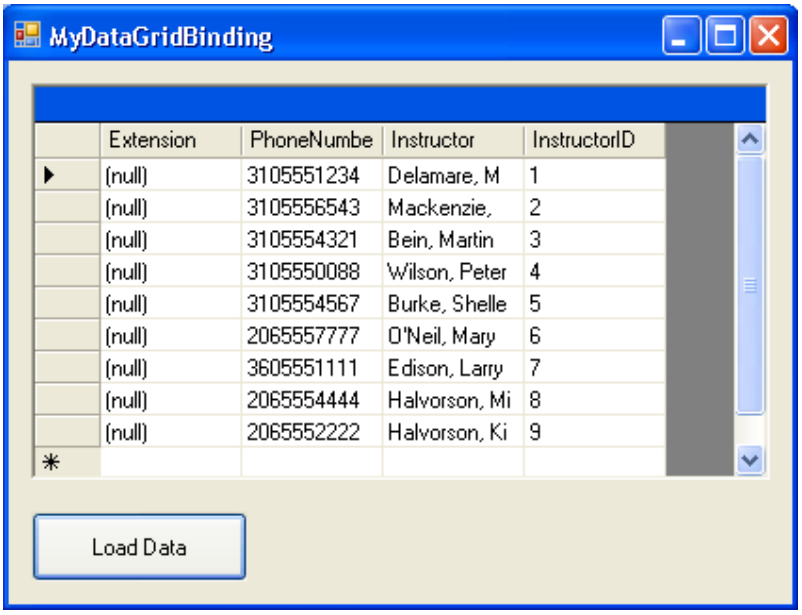
Tạo thủ tục btnLoad_Click bằng cách double click vào nó và nhập đoạn mã sau:

```
DsInstructors1.Clear()
OleDbDataAdapter1.Fill(DsInstructors1)
```

Nhấn nút Save All để lưu lại các thay đổi và chạy thử chương trình.

Chạy chương trình:

Nhấn nút F5 để chạy chương trình. Nhấn nút Load Data để nạp dữ liệu vào trong khung lưới DataGrid:



Bạn có thể kéo để thay đổi kích thước form sao cho các thông tin về csdl xuất hiện đầy đủ. Bạn cũng có thể sắp xếp dữ liệu trong khung lưới bằng cách click vào tiêu đề một cột nào đó. Nhấn nút Close để đóng chương trình.

2. Định dạng các ô lưới trong DataGrid

Bạn có thể định dạng các thành phần trong DataGrid thông qua thuộc tính của nó lúc thiết kế hay khi thực thi chương trình.. Chúng ta sẽ làm điều này với bài tập trên.

Bạn trở lại cửa sổ thiết kế form và mở thuộc tính Properties của khung lưới DataGrid.

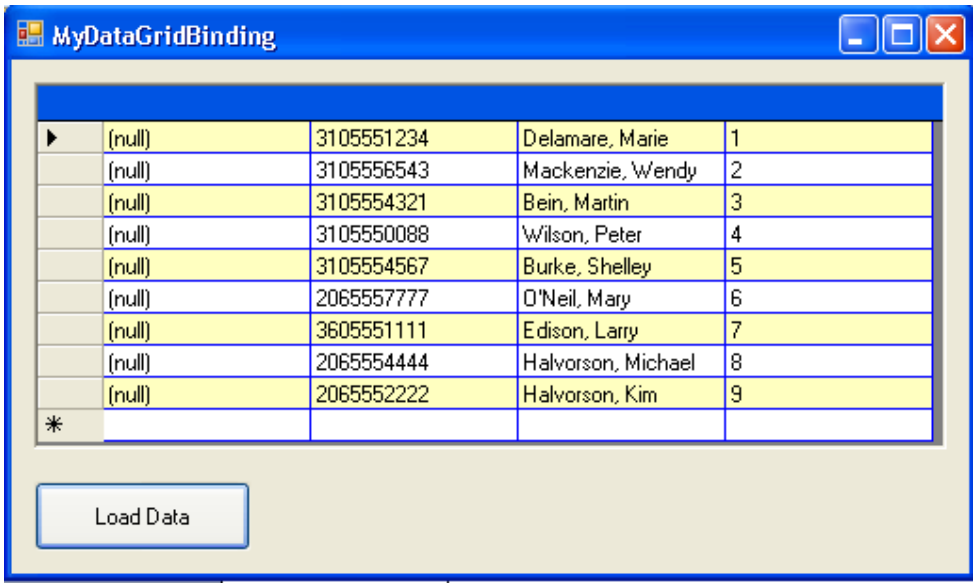
Đặt thuộc tính PreferredColumnWidth là 110 (rộng 110 đơn vị đo Pixel).

Bạn đặt thuộc tính ColumnHeadersVisible là False. Với thiết lập này thì phần tiêu đề của các cột sẽ không hiển thị.

Nhấn chọn thuộc tính BackColor, chọn màu vàng nhạt hiển thị cho nội dung chuỗi chứa trong ô lưới tạo các dòng xen kẽ nhau.

Đặt thuộc tính GridLineColor, chọn màu xanh.

Còn rất nhiều thuộc tính khác bạn có thể tìm hiểu thêm. Giờ bạn hãy chạy chương trình để xem những thay đổi:



3. Cập nhật cơ sở dữ liệu trở lại bảng

DataSet chỉ tiến hành sao chép bảng của csdl chứ không làm thay đổi nội dung csdl cho đến khi có yêu cầu cập nhật bằng phương thức UpDate. Cùng với thuộc tính ReadOnly của DataSet sẽ cho phép có thay đổi hay không với csdl.

Bây giờ chúng ta sẽ tiến hành tìm hiểu những điều đó.

Trở lại cửa sổ thiết kế form và mở thuộc tính properties của DataGrid và thiết lập giá trị TRUE đối với thuộc tính ReadOnly cho phép có những thay đổi dữ liệu trong khung lưới.

Tiến hành đặt một nút nhấn nữa lên form. Thuộc tính như sau: Name – btnUpdate, Text – “Update”.

Nút nhấn Update sẽ hiển thị khi có những thay đổi trong DataGrid và tiến hành cập nhật trở lại cơ sở dữ liệu khi người dùng click vào nó.

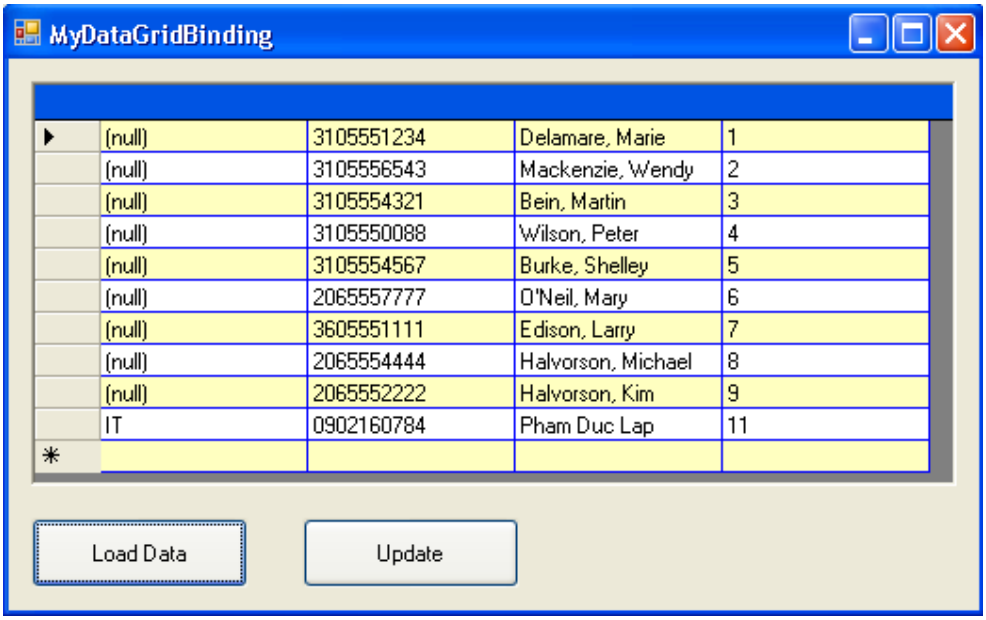
Tạo thủ tục btnUpdate_Click và nhập nội dung như sau:

```
Try
    OleDbDataAdapter1.Update(DsInstructors1)
Catch ex As Exception
    MsgBox(ex.ToString)
End Try
```

Thủ tục này sử dụng phương thức Update của OleDbDataAdapter1 để yêu cầu các thay đổi trong tập DataSet DsInstructors1 trở lại bảng csdl.

Chạy chương trình:

Nhấn F5 để chạy chương trình. Bạn thay đổi nội dung một cột nào đó hay có thể thêm một bản ghi nữa và click vào nút Update để cập nhật vào csdl. Sau đó lại click vào nút Load Data để xem csdl có thay đổi gì không.



4. Tổng kết chương 20

Bạn có thể làm lại những ví dụ trên hay tự tạo cho mình những bài khác có liên quan đến truy xuất csdl.

Trong thời gian tới mình sẽ đưa đến cho các bạn những chương trình xử lý nâng cao với đầy đủ phân tích thiết kế hệ thống, csdl cũng như mã nguồn để các bạn tham khảo. Các bài toán đó mình tổng hợp được như: bài toán vé máy bay, quản lý khách sạn, quản lý sách, phần mềm bán hàng, …. Mời các bạn đón đọc.