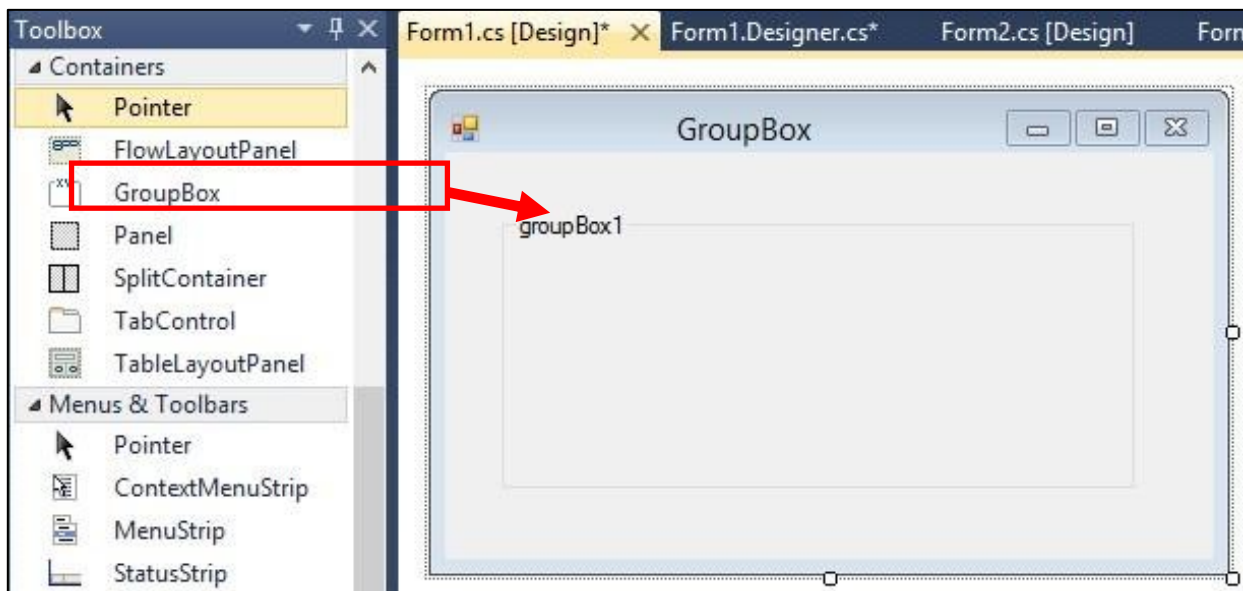


CHƯƠNG 6: ĐIỀU KHIỂN CHỨA CÁC ĐIỀU KHIỂN KHÁC

6.1. Điều khiển GroupBox

GroupBox là dạng điều khiển chứa, có thể chứa các điều khiển khác hiển thị trên form, giúp cho việc thiết kế giao diện của form dễ nhìn và khoa học hơn. *GroupBox* không hỗ trợ thanh trượt (ScrollBar). *GroupBox* có thể có tiêu đề hiển thị, tiêu đề này được thiết lập trong thuộc tính *Text*. Nếu không muốn hiển thị tiêu đề, lập trình viên có thể thiết lập chuỗi rỗng trong thuộc tính *Text*. Điều khiển *GroupBox* được đặt trong nhóm Containers của cửa sổ Toolbox như hình 6.1.



Hình 6.1: Điều khiển GroupBox trong cửa sổ Toolbox

Thông thường *GroupBox* sử dụng để nhóm điều khiển *RadioButton* (xem mục 3.5.2 của chương 3).

- Một số thuộc tính thường dùng của *GroupBox*:

Bảng 6.1: Bảng mô tả các thuộc tính của *GroupBox*

Thuộc tính	Mô tả
<i>Name</i>	Đặt tên cho <i>GroupBox</i>
<i>Text</i>	Chuỗi hiển thị
<i>Font</i>	Thiết lập kiểu chữ, kích thước chữ, ..
<i>ForeColor</i>	Thiết lập màu chữ hiển thị
<i>BackColor</i>	Thiết lập màu nền của <i>GroupBox</i>

<i>Visible</i>	Mang giá trị True hoặc False. <ul style="list-style-type: none"> - Nếu là True: Hiện thị <i>GroupBox</i> - Nếu là False: Không hiện thị <i>GroupBox</i>
<i>AutoSize</i>	Mang giá trị True hoặc False. <ul style="list-style-type: none"> - Nếu là True: <i>GroupBox</i> tự động thay đổi kích thước để có thể hiển thị hết các điều khiển chứa bên trong - Nếu là False: <i>GroupBox</i> có kích thước như lập trình viên thiết lập.
<i>AutoSizeMode</i>	Quy định cách thức điều khiển thay đổi kích thước. <ul style="list-style-type: none"> - <i>GrowAndShrink</i>: <i>GroupBox</i> có thể co và giãn - <i>GrowOnly</i>: Mặc định, chỉ giãn lớn

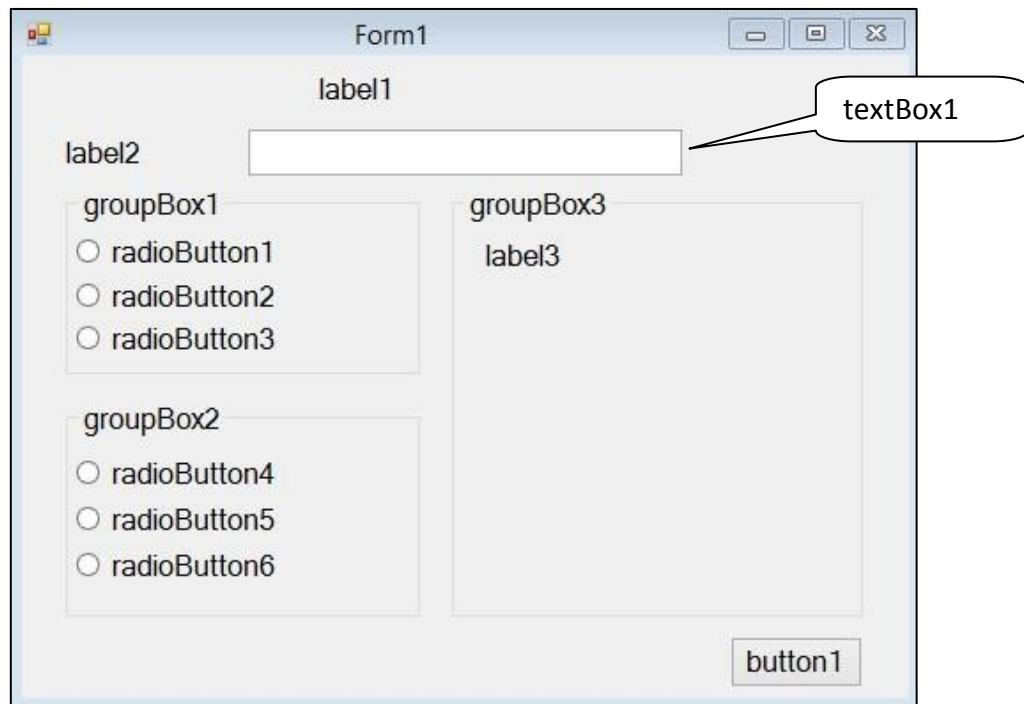
Khi thiết lập giá trị thuộc tính cho *GroupBox* trong bảng 6.1, thì những điều khiển như: *RadioButton*, *TextBox*, *Label*, ... nếu nằm trong *GroupBox* cũng sẽ có những giá trị thuộc tính tương tự như của *GroupBox*.

Ví dụ 6.1: Viết chương trình định dạng chuỗi văn bản, thiết kế giao diện chương trình như hình 6.2.

Hình 6.2: Giao diện form định dạng chuỗi ví dụ 6.1

Yêu cầu: Khi người dùng nhập chuỗi văn bản trong TextBox thì Label “Chuỗi định dạng” sẽ hiển thị chuỗi văn bản vừa nhập. Khi nhấp chuột chọn các Radio trong GroupBox “Màu” và GroupBox “Kiểu hiển thị” thì chuỗi định dạng sẽ thay đổi định dạng tương ứng với lựa chọn của người dùng.

Bước 1: Thiết kế giao diện chương trình. Kéo các điều khiển: Label, GroupBox, RadioButton, TextBox từ cửa sổ Toolbox vào form như hình 6.3.



Hình 6.3: Giao diện form sau khi thêm các điều khiển vào form

Bước 2: Thiết lập giá trị thuộc tính cho các điều khiển trong cửa sổ Properties

- label1:

 - Thuộc tính *Text*: “Định dạng chuỗi”

 - Thuộc tính *Size*: 14

 - Thuộc tính *FontStyle*: Bold

- label2:

 - Thuộc tính *Text*: “Nhập chuỗi muốn định dạng.”

- label3:

 - Thuộc tính *Text*: “Chuỗi định dạng”

 - Thuộc tính *Name*: lblChuoiDinhDang

- textBox1:

 - Thuộc tính *Name*: txtNhapChuoi

- groupBox1:
Thuộc tính Text: “Màu”
- groupBox2:
Thuộc tính Text: “Kiểu hiển thị”
- groupBox3:
Thuộc tính Text: “Chuỗi sau khi định dạng”
- radioButton1:
Thuộc tính Name: radXanh
- radioButton2:
Thuộc tính Name: radDo
- radioButton3:
Thuộc tính Name: radDen
- radioButton4:
Thuộc tính Name: radInDam
- radioButton5:
Thuộc tính Name: radInNghieng
- radioButton6:
Thuộc tính Name: radGachChan
- button1:
Thuộc tính Name: btnThoat
Thuộc tính Text: “Thoát”

Bước 3: Viết mã lệnh cho điều khiển

- Sự kiện *Click* của nút btnThoat:

```
private void btnThoat_Click(object sender, EventArgs e)
{
    Close();
}
```

- Sự kiện *TextChanged* của txtNhapChuoi:

```
private void txtNhapChuoi_TextChanged(object sender, EventArgs e)
{
    lblChuoiDinhDang.Text = txtNhapChuoi.Text;
}
```

- Sự kiện *CheckedChanged* của radXanh:

```
private void radXanh_CheckedChanged(object sender, EventArgs e)
{
    if (radXanh.Checked == true)
    {
        lblChuoiDinhDang.ForeColor = Color.Blue;
    }
}
```

- Sự kiện *CheckedChanged* của radDo:

```
private void radDo_CheckedChanged(object sender, EventArgs e)
{
    if (radDo.Checked == true)
    {
        lblChuoiDinhDang.ForeColor = Color.Red;
    }
}
```

- Sự kiện *CheckedChanged* của radDen:

```
private void radDen_CheckedChanged(object sender, EventArgs e)
{
    if (radDen.Checked == true)
    {
        lblChuoiDinhDang.ForeColor = Color.Black;
    }
}
```

- Sự kiện *CheckedChanged* của radInDam:

```
private void radInDam_CheckedChanged(object sender, EventArgs e)
{
    if (radInDam.Checked == true)
    {
        lblChuoiDinhDang.Font = new
        System.Drawing.Font("Microsoft Sans Serif", 12F,
        System.Drawing.FontStyle.Bold);
    }
}
```

- Sự kiện *CheckedChanged* của radInNghienng:

```
private void radInNghienng_CheckedChanged(object sender,
EventArgs e)
{
    if (radInNghienng.Checked == true)
    {
        lblChuoiDinhDang.Font = new
        System.Drawing.Font("Microsoft Sans Serif", 12F,
        System.Drawing.FontStyle.Italic);
    }
}
```

- Sự kiện *CheckedChanged* của radGachChan:

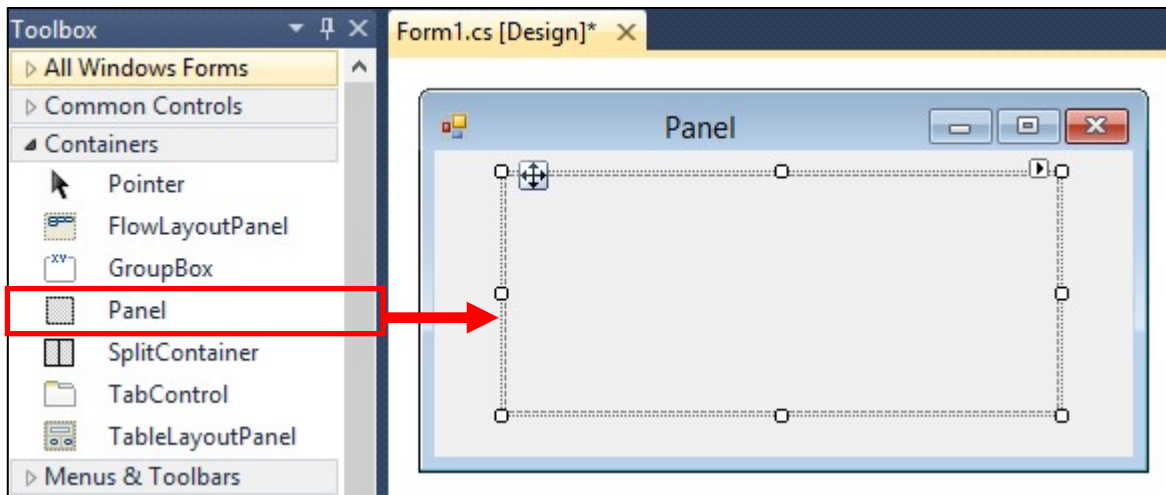
```
private void radGachChan_CheckedChanged(object sender,
EventArgs e)
{
    if (radGachChan.Checked == true)
    {
        lblChuoiDinhDang.Font = new
        System.Drawing.Font("Microsoft Sans Serif", 12F,
        System.Drawing.FontStyle.Underline);
    }
}
```

6.2. Điều khiển Panel

Cũng như GroupBox, Panel là một điều khiển dùng để chứa các điều khiển khác. Panel có các thuộc tính AutoSize, AutoSizeMode như GroupBox và thuộc tính đường viền BorderStyle như Label.

Điểm khác biệt của Panel với GroupBox là điều khiển Panel không có tiêu đề mô tả (không có thuộc tính Text) và có thanh trượt ScrollBar ngang và ScrollBar dọc (có thuộc tính AutoScroll).

Điều khiển Panel đặt trong nhóm Containers của cửa sổ Toolbox như hình 6.4



Hình 6.4: Điều khiển Panel trong cửa sổ Toolbox

- Một số thuộc tính thường dùng của *Panel*:

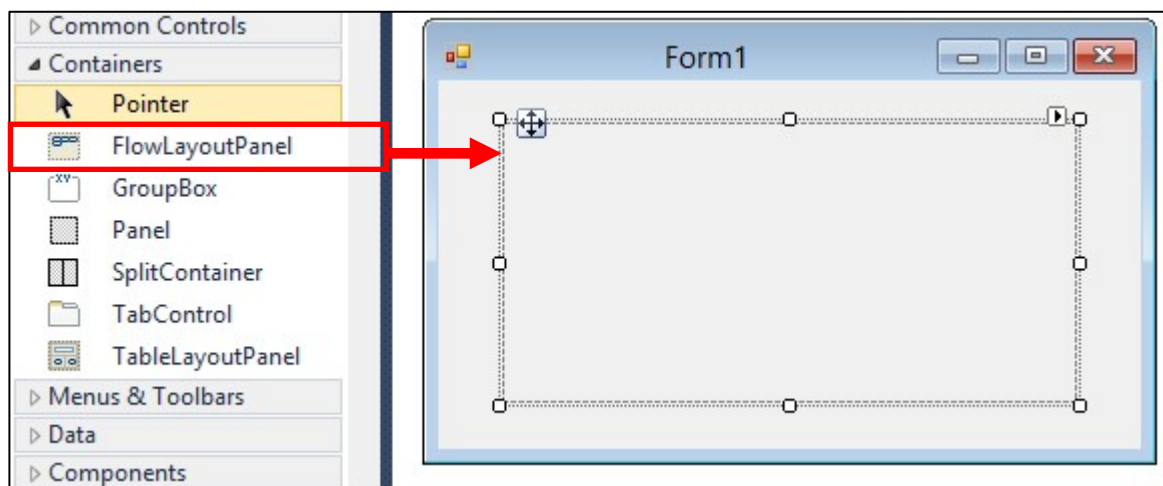
Bảng 6.2: Bảng mô tả các thuộc tính của *Panel*

Thuộc tính	Mô tả
<i>AutoScroll</i>	Mang giá trị True hoặc False. <ul style="list-style-type: none"> - Nếu là True: <i>Panel</i> tự động xuất hiện thanh trượt khi kích thước <i>Panel</i> không thể hiển thị hết các điều khiển chứa bên trong - Nếu là False: <i>Panel</i> sẽ không hiển thị thanh trượt
<i>BorderStyle</i>	Kiểu đường viền của <i>Panel</i> khi hiển thị. Có 3 giá trị: None, FixedSingle và Fixed3D <ul style="list-style-type: none"> - None: Không hiển thị đường viền - FixedSingle:Quanh <i>Panel</i> sẽ hiển thị một đường viền đơn. - Fixed3D: Hiển thị đường viền của <i>Panel</i> dạng 3 chiều.

6.3. Điều khiển FlowLayoutPanel

FlowLayoutPanel là lớp con của điều khiển *Panel*, do đó có thể chứa các điều khiển khác như *Panel*. Mục đích chính của *FlowLayoutPanel* là giúp bố trí các điều khiển trên

form một cách có tổ chức và khoa học. Khi thêm một điều khiển nào đó vào *FlowLayoutPanel* thì *FlowLayoutPanel* sẽ tự động sắp xếp các điều khiển đặt bên trong theo quy tắc định trước và đồng thời cũng thay đổi kích thước của các điều khiển bên trong cho phù hợp với kích thước của *FlowLayoutPanel*. Vì vậy có thể nói điều khiển *FlowLayoutPanel* là điều khiển hỗ trợ tuyệt vời trong việc thiết kế giao diện người dùng. Điều khiển *FlowLayoutPanel* nằm trong nhóm Containers của cửa sổ Toolbox như hình 6.5.

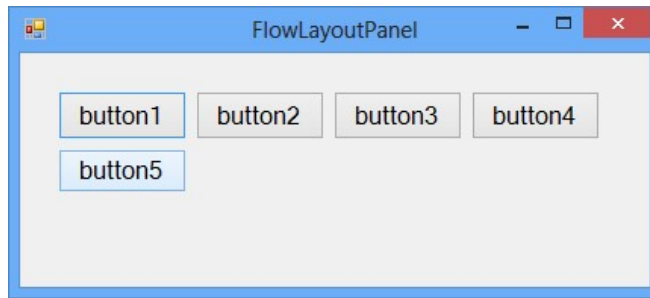


Hình 6.5: Điều khiển *FlowLayoutPanel* trên cửa sổ Toolbox

Điều khiển *FlowLayoutPanel* cũng hỗ trợ dạng thanh trượt (*ScrollBar*) như *Panel*, khi thuộc tính *AutoScroll* được thiết lập là *True* thì khi kích thước các điều khiển được chứa vượt ngoài kích thước *FlowLayoutPanel*, thì *FlowLayoutPanel* sẽ hiển thị thanh trượt.

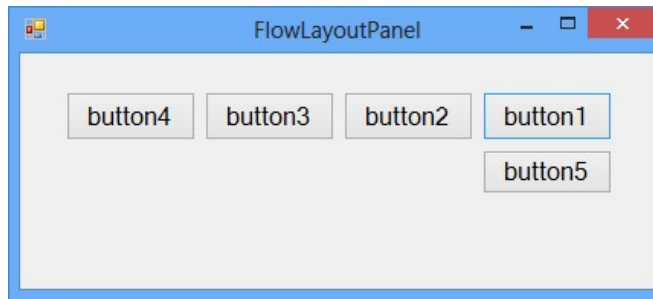
Việc bố trí các điều khiển khi thêm vào *FlowLayoutPanel* như thế nào là do thuộc tính *FlowDirection* quy định. Thuộc tính này mang 4 giá trị cho phép các điều khiển lần lượt thêm vào theo 4 hướng: từ trái qua phải (*LeftToRight*), từ phải qua trái (*RightToLeft*), từ trên xuống (*TopDown*) và từ dưới lên (*BottomUp*). Các điều khiển được thêm vào đến khi vượt ngoài phạm vi của *FlowLayoutPanel*, nếu muốn các điều khiển tự động bố trí xuống dòng mới hoặc sang một cột mới như các hình 6.6, hình 6.7, hình 6.8 và hình 6.9, thì cần phải thiết lập thuộc tính *WrapContents* là *True*. Còn nếu thuộc tính *WrapContents* là *False* thì *FlowLayoutPanel* sẽ hiển thị thanh trượt (thuộc tính *AutoScroll* là *True*) để hiển thị các điều khiển nằm ngoài phạm vi.

Thuộc tính *FlowDirection* là *LeftToRight*:



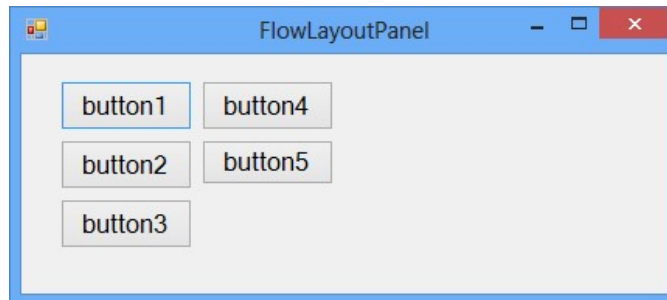
Hình 6.6: Bố trí điều khiển từ trái sang phải với *WrapContents* là *True*

Thuộc tính *FlowDirection* là *RightToLeft*:



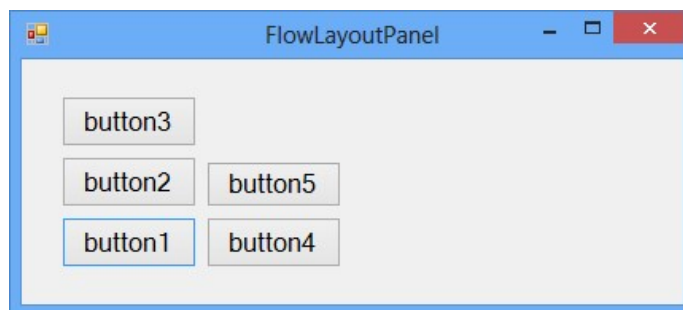
Hình 6.7: Bố trí điều khiển từ phải sang trái với *WrapContents* là *True*

Thuộc tính *FlowDirection* là *TopDown*:



Hình 6.8: Bố trí điều khiển từ trên xuống dưới với *WrapContents* là *True*

Thuộc tính *FlowDirection* là *BottomUp*:



Hình 6.9: Bố trí điều khiển từ dưới lên trên với *WrapContents* là *True*

- Một số thuộc tính thường dùng của *FlowLayoutPanel*:

Bảng 6.3: Bảng mô tả các thuộc tính của *FlowLayoutPanel*

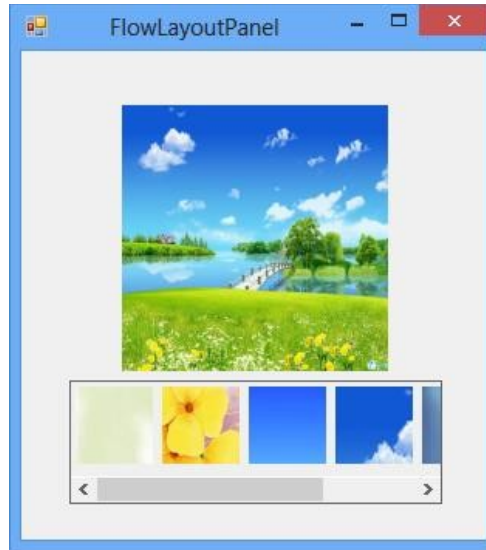
Thuộc tính	Mô tả
<i>AutoScroll</i>	Mang giá trị True hoặc False. <ul style="list-style-type: none"> - Nếu là True: <i>FlowLayoutPanel</i> tự động xuất hiện thanh trượt khi kích thước Panel không thể hiển thị hết các điều khiển chứa bên trong - Nếu là False: <i>FlowLayoutPanel</i> sẽ không hiển thị thanh trượt
<i>BorderStyle</i>	Kiểu đường viền của <i>FlowLayoutPanel</i> khi hiển thị. Có 3 giá trị: None, FixedSingle và Fixed3D <ul style="list-style-type: none"> - None: Không hiển thị đường viền - FixedSingle: Quanh <i>FlowLayoutPanel</i> sẽ hiển thị một đường viền đơn. - Fixed3D: Hiển thị đường viền của <i>FlowLayoutPanel</i> dạng 3 chiều.
<i>FlowDirection</i>	Cách thức bố trí các điều khiển khi các điều khiển nằm ngoài phạm vi của <i>FlowLayoutPanel</i> . Bao gồm 4 giá trị: <i>LeftToRight</i> , <i>RightToLeft</i> , <i>TopDown</i> , <i>BottomUp</i>
<i>WrapContents</i>	Mang giá trị True hoặc False. <ul style="list-style-type: none"> - Nếu là True: Các điều khiển vượt ngoài kích thước <i>FlowLayoutPanel</i> sẽ tự động bố trí trên một dòng mới hoặc một cột mới - Nếu là False: <i>FlowLayoutPanel</i> sẽ xuất hiện thanh trượt để hiển thị các điều khiển ngoài kích thước của <i>FlowLayoutPanel</i> (với thuộc tính <i>AutoScroll</i> là True). Nếu thuộc tính <i>AutoScroll</i> là False thì các điều khiển nằm ngoài kích thước sẽ bị ẩn đi.

- Phương thức thường dùng của *FlowLayoutPanel*:

Bảng 6.4: Bảng mô tả các phương thức của *FlowLayoutPanel*

Phương thức	Mô tả
<i>Controls.Add</i>	Phương thức có chức năng thêm điều khiển vào

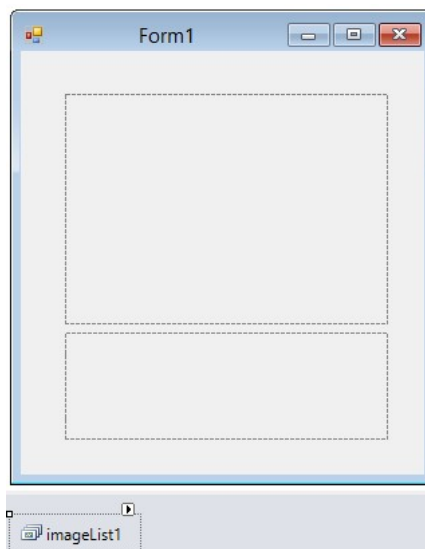
Ví dụ 6.2: Thiết kế giao diện chương trình gồm có 1 PictureBox và 1 FlowLayoutPanel như hình 6.10. Với FlowLayoutPanel chứa danh sách các hình. Khi người dùng nhấp chuột chọn hình nào trong FlowLayoutPanel thì hình đó sẽ hiển thị trên PictureBox.



Hình 6.10: Giao diện form hiển thị hình ví dụ 6.2

Hướng dẫn:

Bước 1: Thiết kế giao diện chương trình. Kéo các điều khiển FlowLayoutPanel, PictureBox và ImageList và form như hình 6.11.



Hình 6.11: Giao diện form sau khi thêm PictureBox, FlowLayoutPanel và ImageList

Bước 2: Thiết lập giá trị thuộc tính cho điều khiển trong cửa sổ Properties

- Form1:

Thuộc tính Text: “FlowLayoutPanel”

- pictureBox1:

Thuộc tính Name: myPictureBox

- flowLayoutPanel1:

Thuộc tính Name: myFlowLayoutPanel

Thuộc tính BorderStyle: FixSingle

Thuộc tính AutoScroll: True

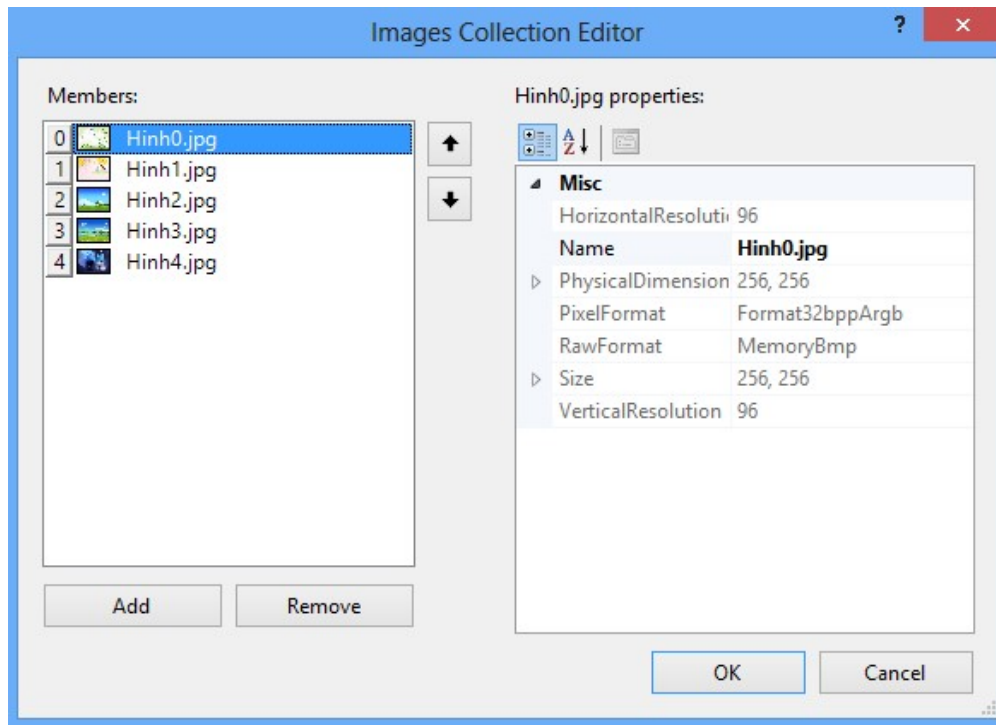
Thuộc tính WrapContents: False

- imageList1:

Thuộc tính Name: myImageList

Thuộc tính ImageSize: 256, 256

Thuộc tính Images: Thêm một số hình ảnh vào myImageList như hình 6.12



Hình 6.12: Cửa sổ thêm hình cho myImageList

Bước 3: Viết mã lệnh cho điều khiển

- Sự kiện *Click* của Form1:

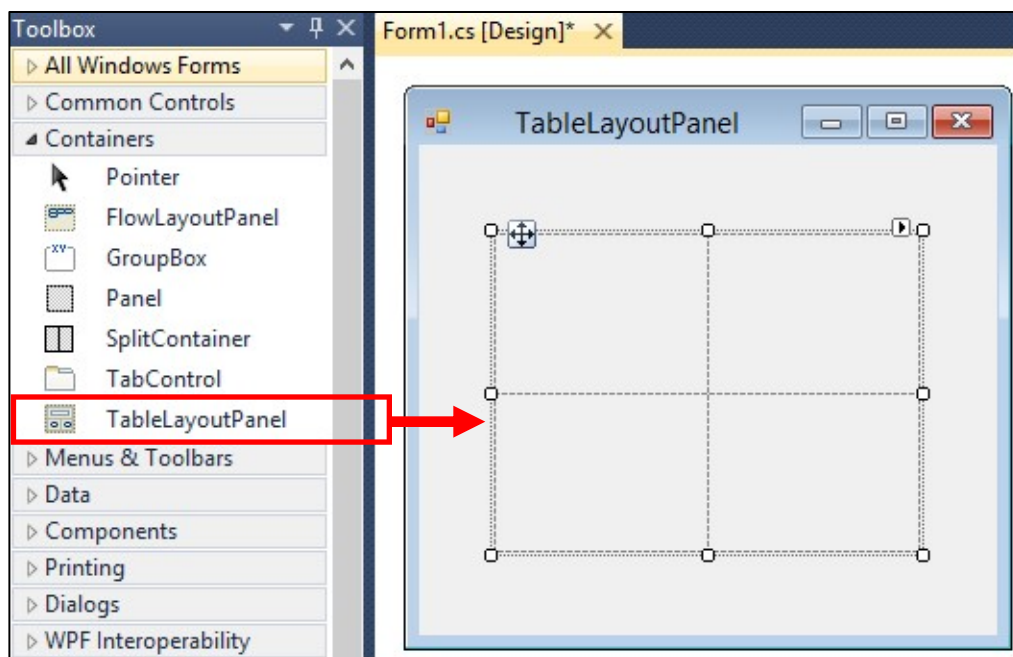
```
private void Form1_Click(object sender, EventArgs e)
{
    try
    {
        PictureBox pic = (PictureBox)sender;
        myPictureBox.Image = pic.Image;
    } catch (Exception ex) { }
}
```

- Sự kiện *Load* của Form1:

```
private void Form1_Load(object sender, EventArgs e)
{
    for (int i = 0; i < myImageList.Images.Count; i++)
    {
        PictureBox pic = new PictureBox();
        pic.Image = myImageList.Images[i];
        pic.Size = new Size(50, 50);
        pic.Click += new EventHandler(Form1_Click);
        myFlowLayoutPanel.Controls.Add(pic);
    }
}
```

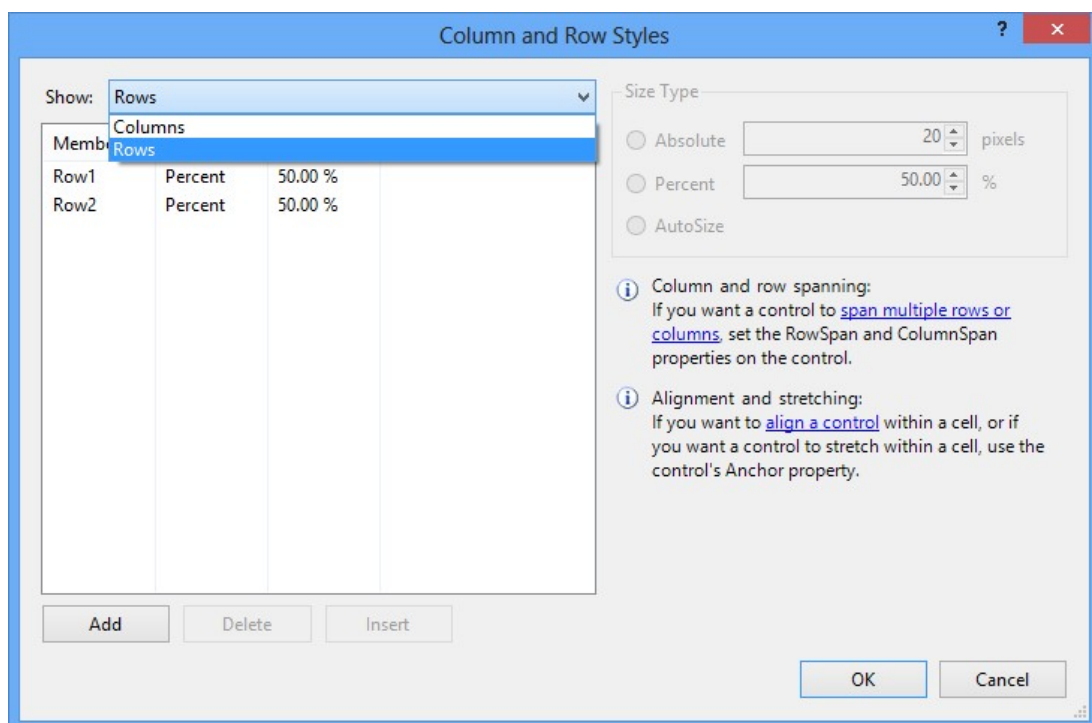
6.4. Điều khiển *TableLayoutPanel*

Cũng như điều khiển *FlowLayoutPanel*, *TableLayoutPanel* là điều khiển dẫn xuất từ điều khiển *Panel* và được dùng cho mục đích thiết kế giao diện form. *TableLayoutPanel* nằm trong nhóm Containers của cửa sổ Toolbox như hình 6.13.

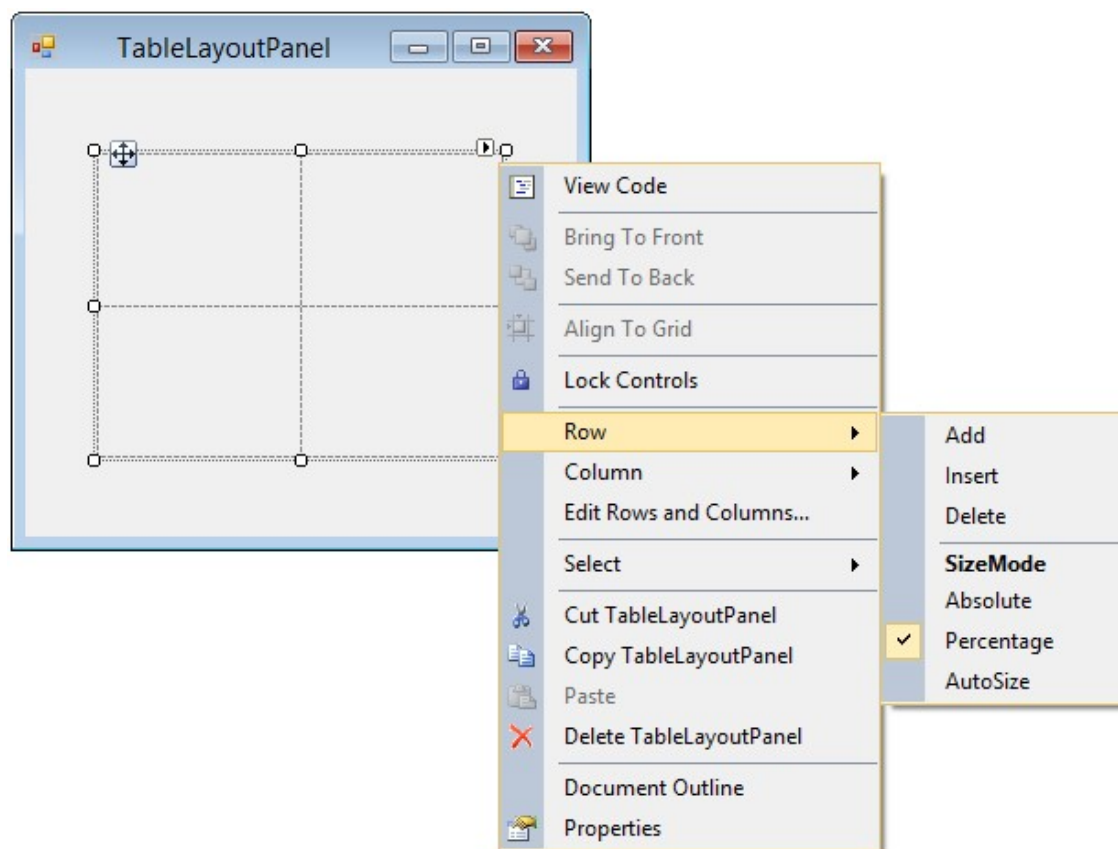


Hình 6.13: Điều khiển *TableLayoutPanel* trong cửa sổ Toolbox

TableLayoutPanel bao gồm các ô theo dòng và cột để thêm điều khiển vào. Lập trình viên có thể thêm các dòng và các cột cho *TableLayoutPanel* qua các thuộc tính *Columns* và *Rows* trong cửa sổ Properties như hình 6.14, hoặc thêm dòng và cột qua *ContextMenuStrip* khi nhấp chuột phải vào *TableLayoutPanel* như hình 6.15.



Hình 6.14: Thêm dòng và cột trên cửa sổ *Column and Row Styles*



Hình 6.15: Thêm dòng hoặc cột trên ContextMenuStrip

Trên cửa sổ Column and Row Styles như hình 6.14, các dòng và cột thêm vào có thể xác định kích thước bằng pixel (mục chọn *Absolute*), phần trăm (mục chọn *Percent*) hoặc tự động điều chỉnh kích thước (mục chọn *AutoSize*).

Với mỗi ô của *TableLayoutPanel*, chỉ có thể chứa được một điều khiển. Tuy nhiên lập trình viên có thể thêm nhiều điều khiển trong một ô bằng cách thêm một điều khiển loại Containers như: *GroupBox*, *Panel*, ... vào ô của *TableLayoutPanel*, khi đó lập trình viên có thể thêm nhiều điều khiển vào điều khiển loại Containers nằm trong ô của *TableLayoutPanel*.

TableLayoutPanel cũng được cung cấp thanh trượt (*ScrollBar*) khi thuộc tính *AutoScroll* là *True*.

TableLayoutPanel là điều khiển dạng bảng chia thành các ô (cell) do đó không có thuộc tính *BorderStyle* mà thay vào đó có thuộc tính *CellBorderStyle*. Thuộc tính *CellBorderStyle* chỉ định kiểu đường viền cho *TableLayoutPanel*. Thông thường, thuộc tính *TableLayoutStyle* có giá trị mặc định là *None*, nghĩa là không hiển thị đường viền quanh các ô của *TableLayoutPanel*. Lập trình có thể thiết lập các loại đường viền được hỗ

trợ cho thuộc tính *CellStyle* như: *Single*, *Inset*, *InsetDouble*, *Outset*, *OutsetDouble*, hoặc *OutsetPartial*.

- Một số thuộc tính thường dùng của *TableLayoutStyle*:

Bảng 6.5: Bảng mô tả các thuộc tính của TableLayoutStyle

Thuộc tính	Mô tả
<i>AutoScroll</i>	Mang giá trị True hoặc False. <ul style="list-style-type: none"> - Nếu là True: <i>TableLayoutStyle</i> tự động xuất hiện thanh trượt khi kích thước Panel không thể hiển thị hết các điều khiển chứa bên trong - Nếu là False: <i>TableLayoutStyle</i> sẽ không hiển thị thanh trượt
<i>CellStyle</i>	Kiểu đường viền của <i>TableLayoutStyle</i> khi hiển thị. Có 3 giá trị: <i>None</i> , <i>Single</i> và <i>Inset</i> , <i>InsetDouble</i> , <i>Outset</i> , <i>OutsetDouble</i> , <i>OutsetPartial</i> .
<i>ColumnCount</i>	Số cột của <i>TableLayoutPanel</i> . Lập trình viên có thể thêm hoặc giảm số cột bằng cách thay đổi giá trị thuộc tính <i>ColumnCount</i>
<i>Columns</i>	Thuộc tính này giúp hiển thị bảng Column and Row Styles để thêm, sửa hoặc xóa cột.
<i>GrowStyle</i>	Thuộc tính chỉ định việc thay đổi kích thước của <i>TableLayoutPanel</i> như thế nào. Gồm các giá trị: <ul style="list-style-type: none"> - <i>FixedSize</i>: Cố định kích thước, không thay đổi. - <i>AddRows</i>: Thêm dòng mới - <i>AddColumns</i>: Thêm cột mới
<i>RowCount</i>	Số dòng của <i>TableLayoutPanel</i> . Lập trình viên có thể thêm hoặc giảm số dòng bằng cách thay đổi giá trị thuộc tính <i>RowCount</i>
<i>Rows</i>	Thuộc tính này giúp hiển thị bảng Column and Row Styles để thêm, sửa hoặc xóa dòng.

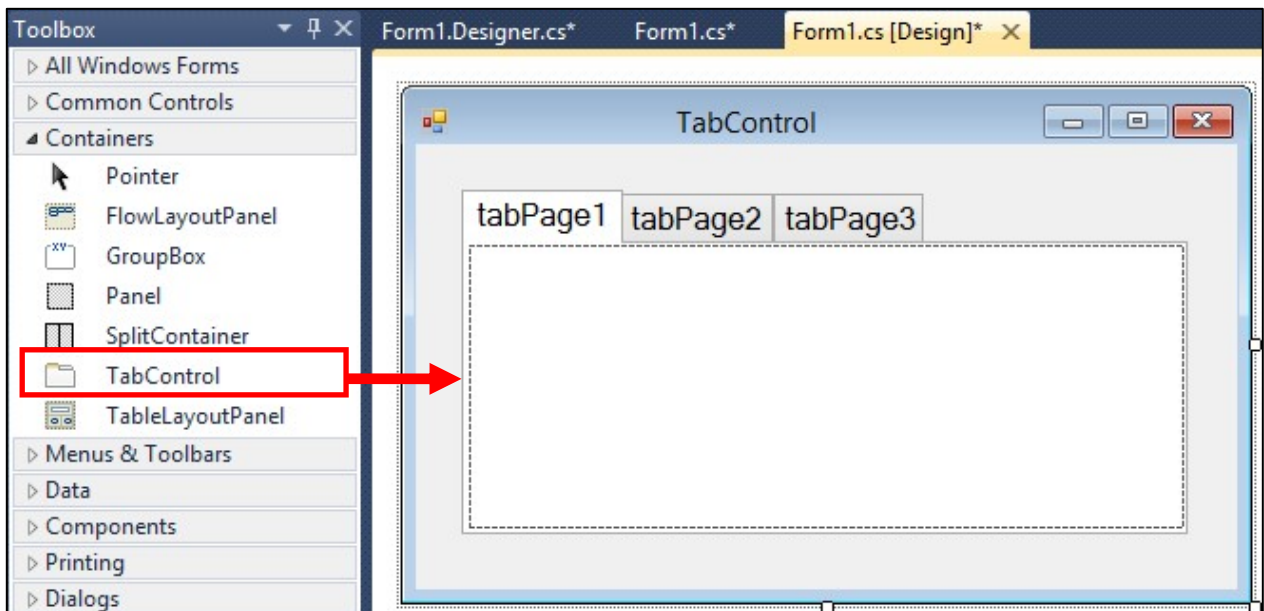
- Phương thức thường dùng của *TableLayoutStyle*:

Bảng 6.6: Bảng mô tả các phương thức của *TableLayoutStyle*

Phương thức	Mô tả
<i>Controls.Add</i>	Phương thức có chức năng thêm điều khiển vào <i>TableLayoutStyle</i>

6.5. Điều khiển TabControl

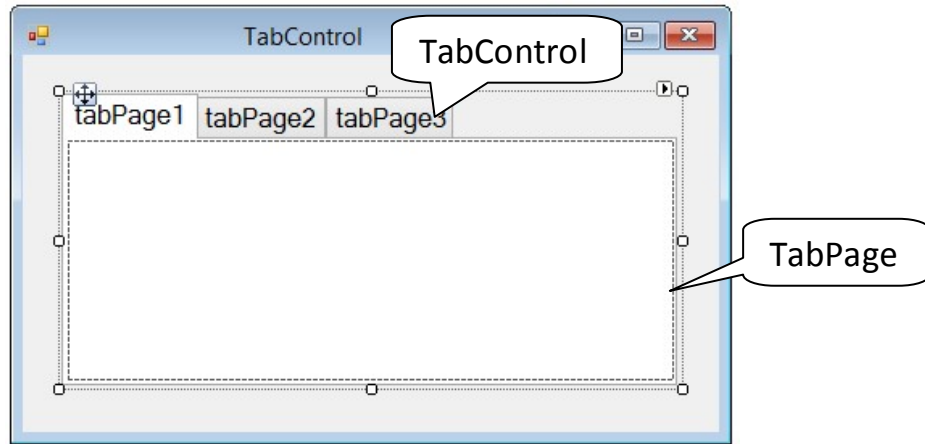
TabControl là điều khiển dạng Containers, do đó có thể chứa các điều khiển khác. Điểm đặc biệt của TabControl là cho phép thể hiện nhiều page trên một form duy nhất. Mỗi page có thể chứa nhiều điều khiển khác bên trong. Điều khiển TabControl nằm trong nhóm Containers của cửa sổ Toolbox như hình 6.16.



Hình 6.16: Điều khiển TabControl trong cửa sổ Toolbox

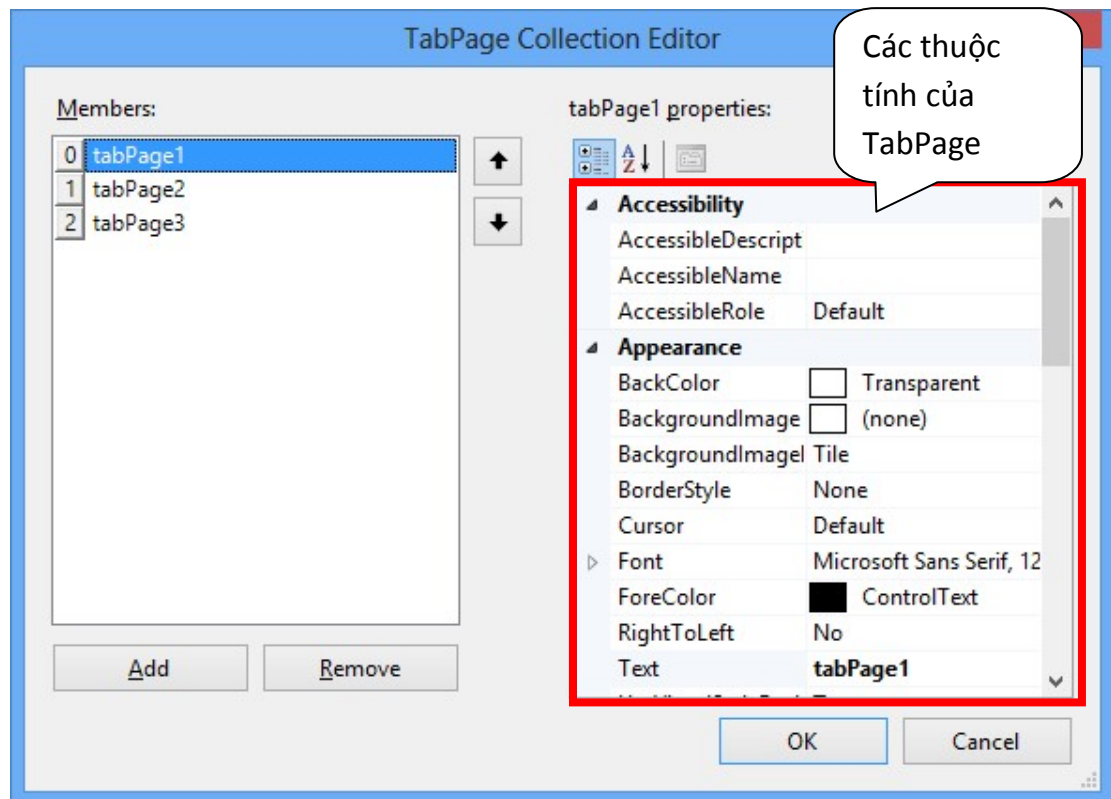
- TabPage:

Thuộc tính quan trọng nhất của *TabControl* là *TabPage*. Một *TabControl* có thể có nhiều *TabPage* như hình 6.17. Người dùng có thể nhấp vào các tab để chuyển đổi qua lại giữa các *TabPage* với nhau



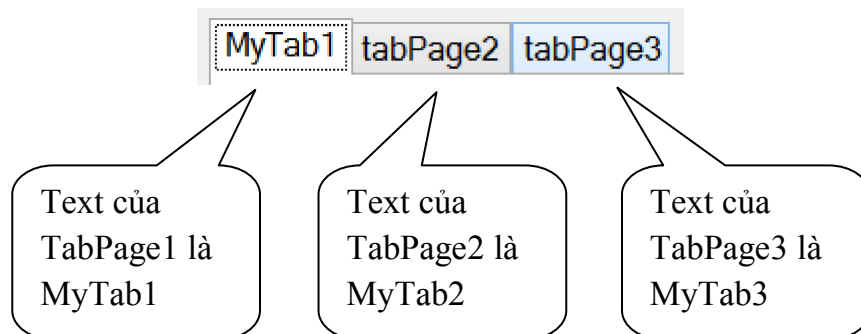
Hình 6.17: TabControl chứa nhiều TabPage

TabPage là điều khiển dạng Container nằm trong TabControl và có thể chứa các điều khiển khác bên trong. Mỗi TabPage có các thuộc tính riêng, lập trình viên có thể thiết lập giá trị thuộc tính khác nhau trên mỗi TabPage của TabControl bằng cách nhấp chuột trái chọn thuộc tính TabPages trên cửa sổ Properties. Khi đó một cửa sổ tabPage Collection Editor sẽ hiển thị như hình 6.18. Tại cửa sổ này, lập trình viên cũng có thể thêm hoặc xóa các TabPage bằng cách nhấn nút Add hoặc Remove.



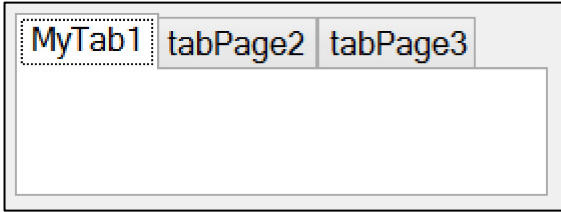
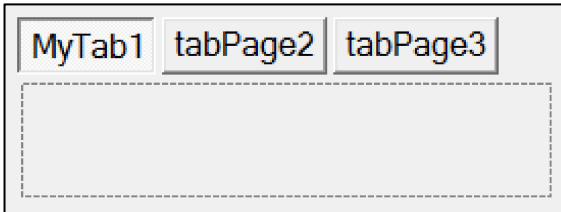
Hình 6.18: Cửa sổ thiết lập giá trị thuộc tính cho TabPage

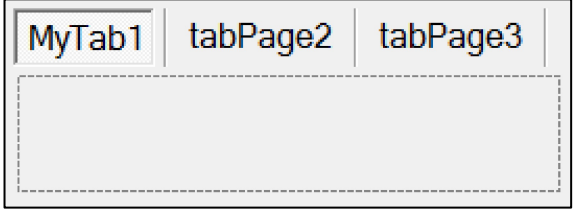
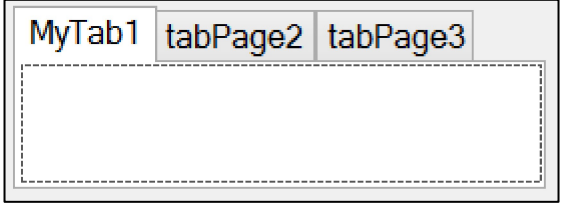



Điều khiển *TabPage* có nhiều điểm giống với điều khiển *Panel*. *TabPage* cũng hỗ trợ thanh trượt khi cần nếu như thuộc tính *AutoScroll* được thiết lập là *True*, có thuộc tính *BorderStyle* để thiết lập đường viền quanh *TabPage* với 3 giá trị: *None*, *FixedSingle*, *Fixed3D*. Tuy nhiên có điểm khác biệt với *Panel* là *TabPage* hỗ trợ thuộc tính *Text*, chuỗi mô tả được thiết lập trong thuộc tính *Text* sẽ hiển thị trên tab của *TabPage*:

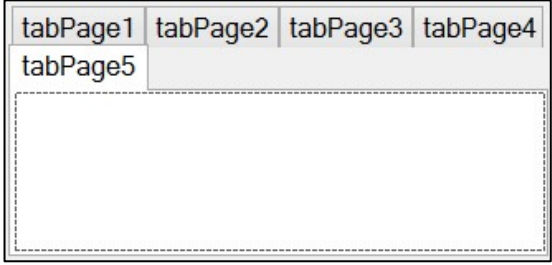
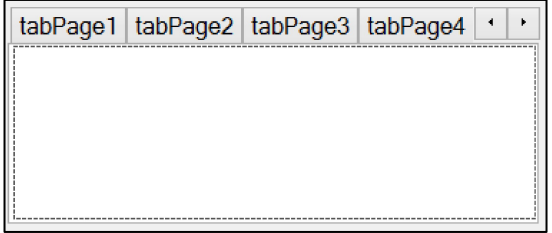


➤ Các thuộc tính thường dùng của *TabControl*:

Bảng 6.7: Bảng mô tả các thuộc tính của *TabControl*

Thuộc tính	Mô tả
<i>Appearance</i>	<p>Thuộc tính chỉ định <i>TabPage</i> sẽ hiển thị ở hình dạng nào. Có 3 giá trị:</p> <ul style="list-style-type: none"> - <i>Normal</i>:  - <i>Button</i>:  - <i>FlatButtons</i>:

	
<i>Alignment</i>	<p>Thuộc tính xác định các tab sẽ hiển thị ở trên, dưới, trái hay phải của <i>TabControl</i>. Gồm các giá trị:</p> <ul style="list-style-type: none"> - <i>Top</i>:  - <i>Bottom</i>:  - <i>Left</i>:  - <i>Right</i>: 
<i>Multiline</i>	<p>Mang hai giá trị True hoặc False.</p> <ul style="list-style-type: none"> - Nếu là True: Cho phép hiển thị nhiều dòng để chứa các tab nếu số lượng các tab vượt ngoài phạm vi kích thước của <i>TabControl</i>.

	 <p>- Nếu là False: Chỉ cho phép tab hiển thị trên một dòng.</p> 
<i>TabPages</i>	Chứa tập các các <i>TabPage</i> có trong <i>TabControl</i>
<i>TabCount</i>	Trả về số lượng <i>TabPage</i> mà <i>TabControl</i> có
<i>SelectedTab</i>	Trả về điều khiển <i>TabPage</i> được chọn
<i>SelectedIndex</i>	Trả về vị trí của <i>TabPage</i> được chọn

➤ Các sự kiện thường dùng của *TabControl*:

Bảng 6.8: Bảng mô tả các sự kiện của TabControl

Sự kiện	Mô tả
<i>SelectedIndexChanged</i>	Phát sinh khi người dùng chọn một <i>TabPage</i> khác trên <i>TabControl</i>

Ví dụ 6.3: Viết chương trình quản lý nhân sự như hình 6.19 và 6.20. Chương trình gồm 2 *TabPage*: *TabPage* Quản lý nhân viên và *TabPage* Quản lý giáo viên.

- *TabPage* quản lý nhân viên: Cho phép thêm, sửa và xóa nhân viên. Thông tin nhân viên cần quản lý bao gồm: Họ tên nhân viên, chức vụ của nhân viên, hệ số lương và lương cơ bản.
- *TabPage* Quản lý giáo viên: Cho phép thêm sửa xóa giáo viên. Thông tin giáo viên cần quản lý gồm: Họ tên giáo viên, chức vụ của giáo viên, tiền giảng một tiết, số tiết dạy và học vị của giáo viên.

Quản lý nhân sự

Quản lý nhân viên | Quản lý giáo viên

Họ tên: Chức vụ:

Lương cơ bản: Hệ số lương:

Thêm nhân viên | Cập nhật | Xóa nhân viên

Họ tên	Chức vụ	Hệ số lương	Lương cơ bản

Thoát

Hình 6.19: Giao diện TabPage Quản lý nhân viên

Quản lý nhân sự

Quản lý nhân viên | Quản lý giáo viên

Họ tên: Học vị:

Tiền dạy một tiết: Số tiết dạy:

Thêm giáo viên | Cập nhật | Xóa giáo viên

Họ tên	Học vị	Tiền dạy một tiết	Số tiết dạy

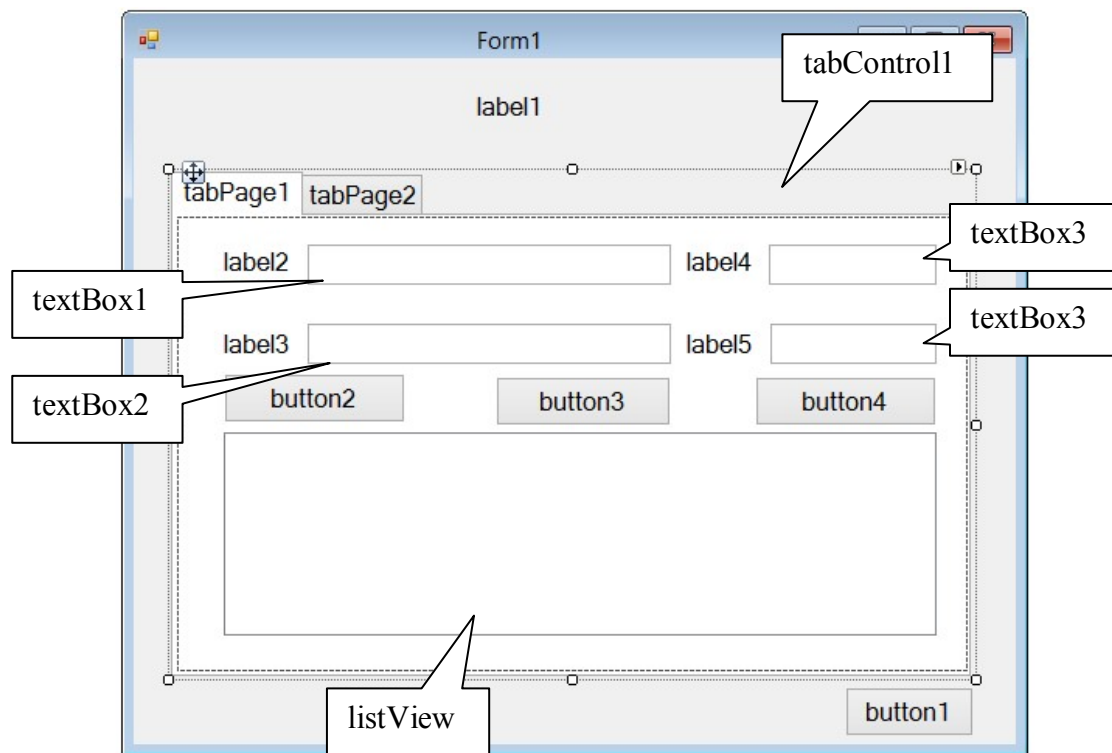
Thoát

Hình 6.20: Giao diện TagPage Quản lý giáo viên

Hướng dẫn:

Tạo *TabPage* Quản lý nhân viên:

Bước 1: Thiết kế giao diện ban đầu. Thêm các điều khiển *Label*, *TextBox*, *TabControl* và *ListView* vào form như hình 6.21.

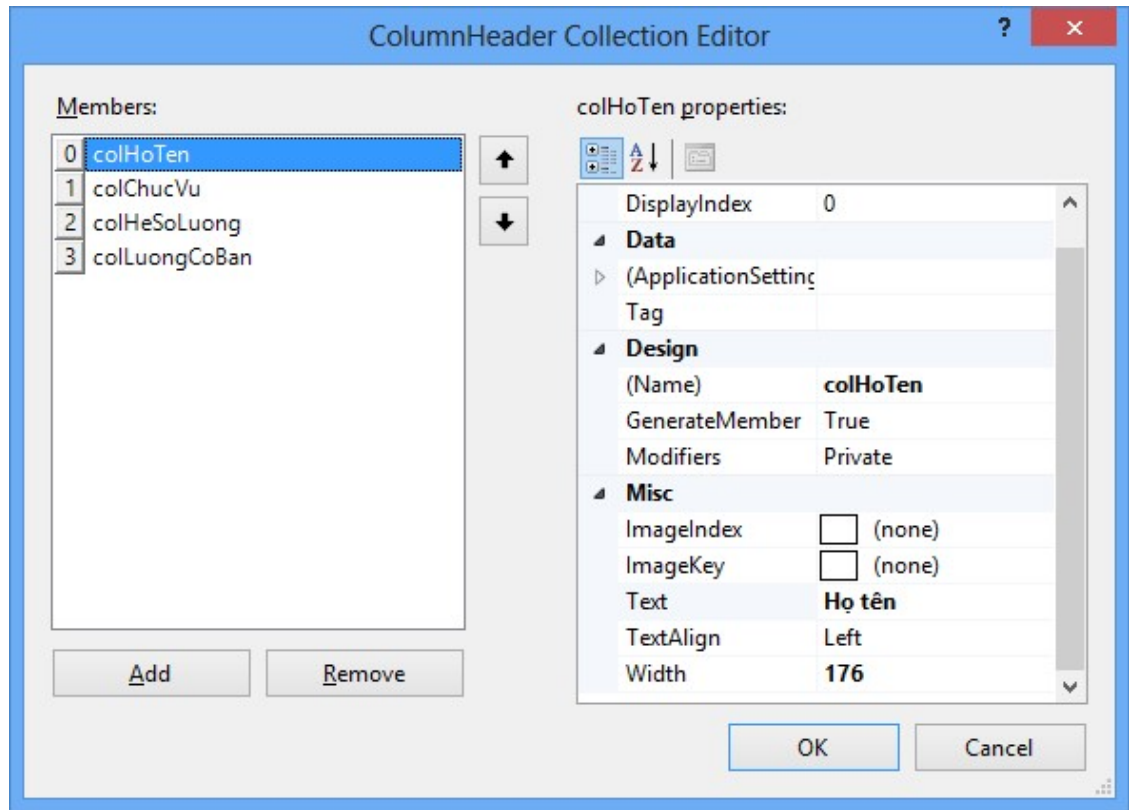


Hình 6.21: Giao diện *TabPage* Quản lý nhân viên sau khi thêm điều khiển

Bước 2: Thiết lập giá trị thuộc tính cho điều khiển trong cửa sổ Properties

- Form1:
Thuộc tính *Text*: “TabControl”
- label1:
Thuộc tính *Text*: “Quản lý nhân sự”
Thuộc tính *Size*: 14
- label2:
Thuộc tính *Text*: “Họ tên:”
- label3:
Thuộc tính *Text*: “Lương cơ bản:”
- label4:
Thuộc tính *Text*: “Chức vụ:”
- label5:

- Thuộc tính *Text*: “Hệ số lương:”
- textBox1:
 - Thuộc tính *Name*: txtHoTenNV
- textBox2:
 - Thuộc tính *Name*: txtLuongCBNV
- textBox3:
 - Thuộc tính *Name*: txtChucVuNV
- textBox4:
 - Thuộc tính *Name*: txtHeSoLuongNV
- button1:
 - Thuộc tính *Name*: btnThoat
 - Thuộc tính *Text*: “Thoát”
- button2:
 - Thuộc tính *Name*: btnThemNV
 - Thuộc tính *Text*: “Thêm nhân viên”
- button3:
 - Thuộc tính *Name*: btnCapNhatNV
 - Thuộc tính *Text*: “Cập nhật”
- button4:
 - Thuộc tính *Name*: btnXoaNV
 - Thuộc tính *Text*: “Xóa nhân viên”
- listView1:
 - Thuộc tính *Name*: listNhanVien
 - Thuộc tính *View*: Detail
 - Thuộc tính *FullRowSelect*: True
 - Thuộc tính *MultiSelect*: False
 - Thuộc tính *Columns*: Mở cửa sổ ColumnHeader Collection Editor, thêm 4 cột: colHoTen, colChucVu, colHeSoLuong, colLuongCoBan như hình 6.22.
 - Cột colHoTen: Thiết lập thuộc tính *Text* là “Họ tên”
 - Cột colChucVu: Thiết lập thuộc tính *Text* là “Chức vụ”
 - Cột colHeSoLuong: Thiết lập thuộc tính *Text* là “Hệ số lương”
 - Cột colLuongCoBan: Thiết lập thuộc tính *Text* là “Lương cơ bản”



Hình 6.22: Thêm cột cho listView1 trong cửa sổ ColumnHeader Collection Editor

- tabControl1:

Thuộc tính Name: myTabControl

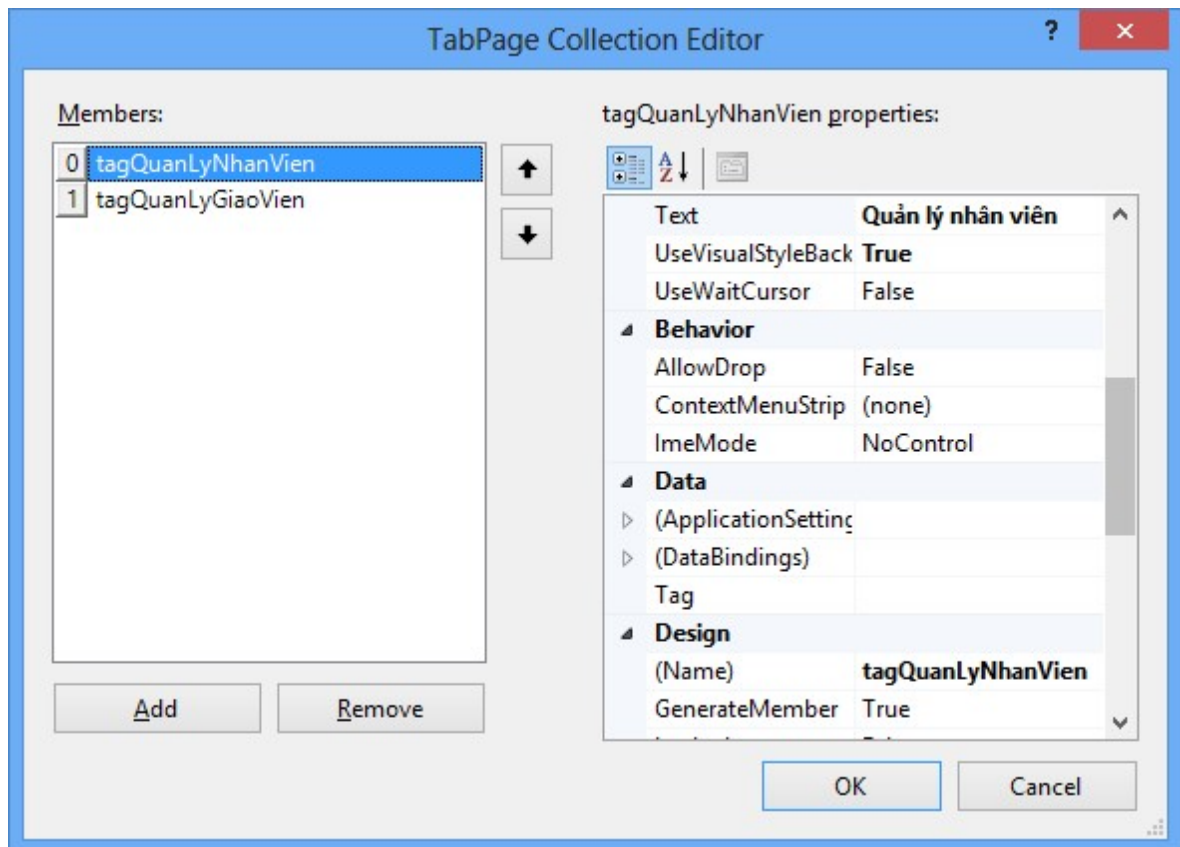
Thuộc tính *TabPage*: Mở cửa sổ TabPage Collection Editor, thêm 2 *TabPage*: Quản lý nhân viên và Quản lý giáo viên như hình 6.23.

Trên tabQuanLyNhanVien:

- Thiết lập thuộc tính *Text*: “Quản lý nhân viên”

Trên tabQuanLyGiaoVien:

- Thiết lập thuộc tính *Text*: “Quản lý giáo viên”



Hình 6.23: Thêm TabPage cho TabControl trên cửa sổ TabPage Collection Editor

Bước 3: Viết mã lệnh cho các điều khiển

- Sự kiện *Click* của btnThoat:

```
private void btnThoat_Click(object sender, EventArgs e)
{
    Close();
}
```

- Sự kiện *Click* của btnCapNhatNV:

```
private void btnCapNhatNV_Click(object sender, EventArgs e)
{
    foreach (ListViewItem lvi in listNhanVien.SelectedItems)
    {
        lvi.SubItems[0].Text = txtTenNV.Text;
        lvi.SubItems[1].Text = txtChucVuNV.Text;
        lvi.SubItems[2].Text = txtHeSoLuongNV.Text;
        lvi.SubItems[3].Text = txtLuongCBNV.Text;
    }
}
```

- Sự kiện *Click* của btnThemNV:

```
private void btnThemNV_Click(object sender, EventArgs e)
{
    if (txtLuongCBNV.Text != "" && txtTenNV.Text != "" &&
        txtHeSoLuongNV.Text != "" && txtChucVuNV.Text != "")
    {
        ListViewItem LVItem = new ListViewItem(txtTenNV.Text);
        ListViewItem.ListViewSubItem LVItemCV = new
            ListViewItem.ListViewSubItem(LVItem,
                txtChucVuNV.Text);
        ListViewItem.ListViewSubItem LVItemHSL = new
            ListViewItem.ListViewSubItem(LVItem,
                txtHeSoLuongNV.Text);
        ListViewItem.ListViewSubItem LVItemLCB = new
            ListViewItem.ListViewSubItem(LVItem,
                txtLuongCBNV.Text);
        LVItem.SubItems.Add(LVItemCV);
        LVItem.SubItems.Add(LVItemHSL);
        LVItem.SubItems.Add(LVItemLCB);
        listNhanVien.Items.Add(LVItem);
        txtLuongCBNV.Text = "";
        txtTenNV.Text = "";
        txtHeSoLuongNV.Text = "";
        txtChucVuNV.Text = "";
    }
    else
        MessageBox.Show("Phải nhập đầy đủ thông tin nhân viên");
}
```

- Sự kiện *Click* của btnXoaNV:

```
private void btnXoaNV_Click(object sender, EventArgs e)
{
    if (listNhanVien.SelectedIndices.Count > 0)
    {
        listNhanVien.Items.RemoveAt(listNhanVien.FocusedItem.Index);
    }
    else
        MessageBox.Show("Phải chọn nhân viên muốn ");
}
```

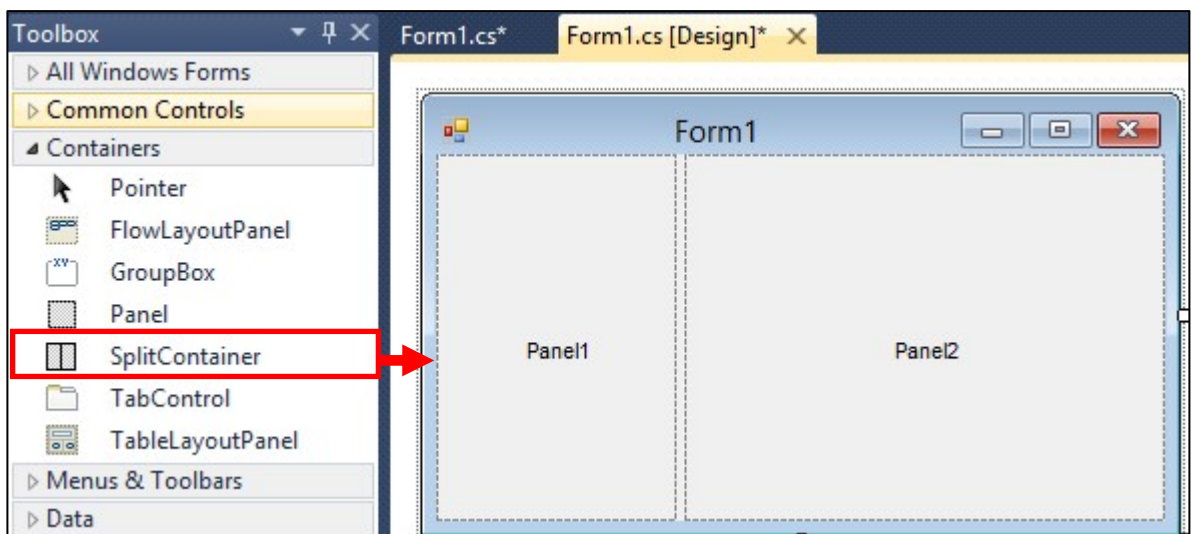
- Sự kiện *SelectedIndexChanged* của *listNhanVien*:

```
private void listNhanVien_SelectedIndexChanged(object sender,
EventArgs e)
{
    foreach (ListViewItem lvi in listNhanVien.SelectedItems)
    {
        txtTenNV.Text = lvi.SubItems[0].Text;
        txtChucVuNV.Text = lvi.SubItems[1].Text;
        txtHeSoLuongNV.Text = lvi.SubItems[2].Text;
        txtLuongCBNV.Text = lvi.SubItems[3].Text;
    }
}
```

Tạo TagPage Quản lý giáo viên: Thiết kế và viết mã lệnh tương tự như TagPage Quản lý nhân viên.

6.6. Điều khiển SplitContainer

Điều khiển *SplitContainer* giúp phân chia form thành hai phần. Cụ thể hơn *SplitContainer* được cấu tạo bởi hai điều khiển *Panel*, mỗi *Panel* trong điều khiển *SplitContainer* đều có chức năng như một điều khiển *Panel* thông thường. Khi thêm điều khiển *SplitContainer* từ cửa sổ Toolbox vào form thì mặc định *SplitContainer* có thuộc tính *Dock* mang giá trị *Fill*. Kích thước của hai *Panel* trong *SplitContainer* có thể thay đổi nhờ *Splitter*, *Splitter* là một vạch phân cách hai *Panel*. Điều khiển *SplitContainer* nằm trong nhóm Containers của cửa sổ Toolbox như hình 6.24.

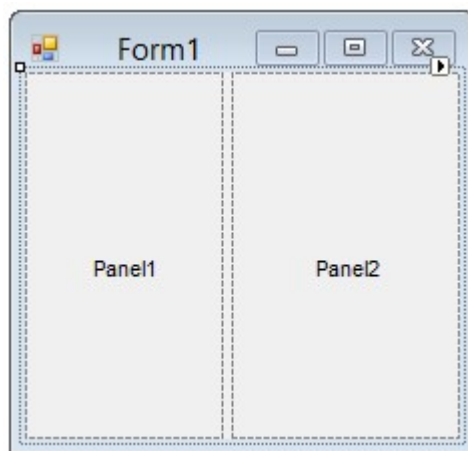


Hình 6.24: Điều khiển *SplitContainer* trong cửa sổ Toolbox

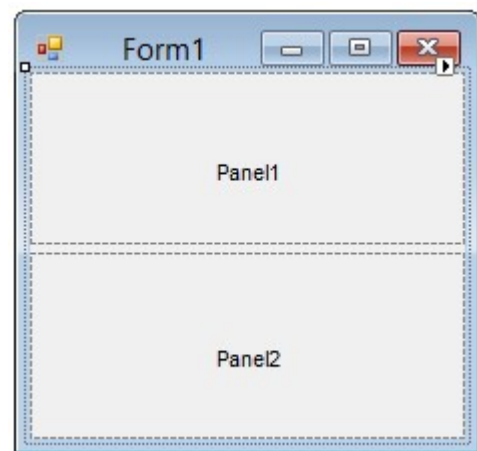
Các *Panel* đều hỗ trợ thanh trượt (ScrollBar) khi thuộc tính *AutoScroll* được thiết lập là *True*.

Tuy nhiên *Panel* trong *SplitContainer* không có thuộc tính *BorderStyle* để thiết lập đường viền, vì là điều khiển chứa trong *SplitContainer*, do đó thuộc tính *BorderStyle* được đặt ở điều khiển *SplitContainer*. Các giá trị của thuộc tính *BorderStyle* của *SplitContainer* cũng gồm 3 giá trị như *Panel*: *None*, *Fixed3D*, *FixedSingle*.

Vạch phân cách *Splitter* có thể phân cách theo chiều dọc hoặc chiều ngang tùy thuộc vào thuộc tính *Orientation*. Thuộc tính *Orientation* mang hai giá trị để thiết lập *SplitContainer* là: *Vertical* và *Horizontal* như hình 6.25, hình 6.26.



Hình 6.25: Thuộc tính *Orientation* là *Vertical*



Hình 6.26: Thuộc tính *Orientation* là *Horizontal*

Nếu không muốn cho người dùng dịch chuyển vạch phân cách *Splitter* để thay đổi kích thước của hai *Panel*, lập trình viên có thể thiết lập thuộc tính *IsSplitterFixed* của *SplitContainer* là *True*. Ngoài ra một điểm đặc biệt là có thể chỉ định không cho phép thay đổi kích thước của *Panel1* hoặc *Panel2* bằng cách kết hợp thuộc tính *FixedPanel* và thuộc tính *IsSplitterFixed* như bảng 6.9.

Bảng 6.9: Bảng mô tả thuộc tính *IsSplitterFixed* và *FixedPanel*

Thuộc tính	Tác dụng
<i>IsSplitterFixed</i> = False	Thuộc tính <i>FixedPanel</i> không có hiệu lực. Người dùng có thể thay đổi kích thước của <i>Panel1</i> và cả <i>Panel2</i>
<i>IsSplitterFixed</i> = True	Thiết lập thuộc tính <i>FixedPanel</i> : <ul style="list-style-type: none"> - <i>FixedPanel</i> = None: Người dùng không thể sử dụng vạch phân cách <i>Splitter</i> để thay đổi kích thước của cả

	<p><i>Panel1</i> và <i>Panel2</i>. Nhưng kích thước cả hai <i>Panel</i> sẽ thay đổi khi <i>SplitContainer</i> có thuộc tính <i>Dock</i> là <i>Fill</i> và người dùng thay đổi kích thước form.</p> <ul style="list-style-type: none"> - <i>FixedPanel</i> = <i>Panel1</i>: Người dùng không thể thay đổi kích thước <i>Panel1</i> (Khi thay đổi kích thước form thì kích thước <i>Panel2</i> thay đổi, kích thước <i>Panel1</i> là không đổi). - <i>FixedPanel</i> = <i>Panel2</i>: Người dùng không thể thay đổi kích thước <i>Panel2</i> (Khi thay đổi kích thước form thì kích thước <i>Panel1</i> thay đổi, kích thước <i>Panel2</i> là không đổi).
--	---

Một trong hai *Panel* của *SplitContainer* có thể ẩn đi bằng cách thiết lập thuộc tính *Panel1Collapsed* và *Panel2Collapsed* là *True*. Việc ẩn hai *Panel* chỉ có tác dụng với một *Panel*. Nghĩa là chỉ có thể thiết lập một trong hai thuộc tính *Panel1Collapsed* và *Panel2Collapsed* là *True*. Khi *Panel1Collapsed* là *True* thì *Panel2Collapsed* là *False* và ngược lại.

➤ Các thuộc tính thường dùng của *SplitContainer*:

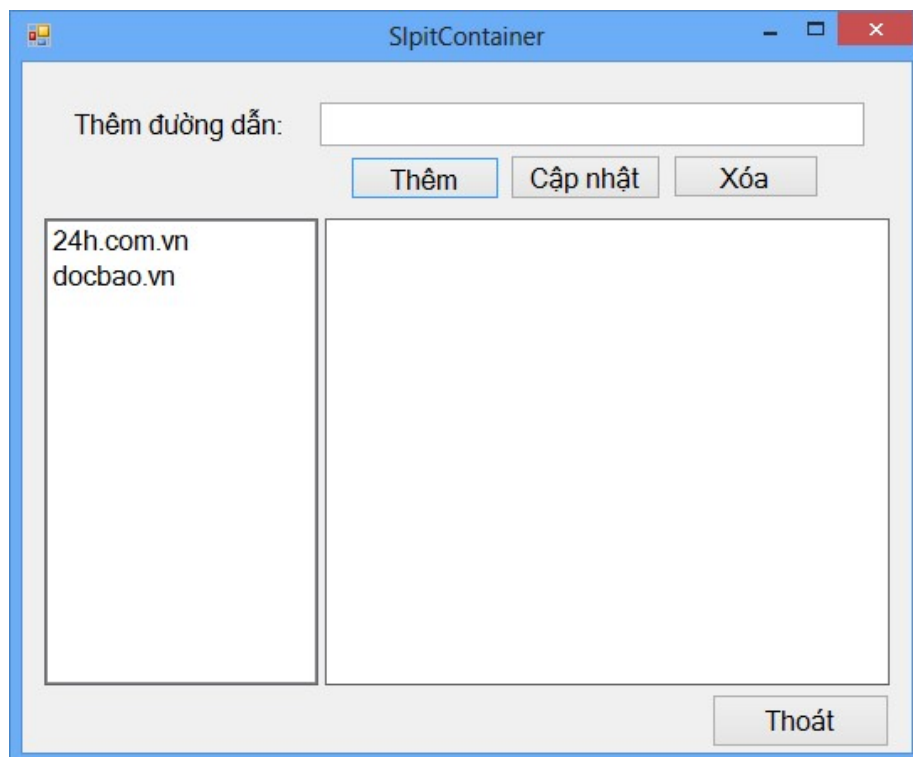
Bảng 6.10: Bảng mô tả các thuộc tính của *SplitContainer*

Thuộc tính	Mô tả
<i>BorderStyle</i>	Thiết lập đường viền cho <i>SplitContainer</i> . Gồm 3 giá trị: <i>None</i> , <i>FixedSingle</i> , <i>Fixed3D</i> .
<i>FixedPanel</i>	Cố định kích thước của các <i>Panel</i> trong <i>SplitContainer</i> . Gồm 3 giá trị: <i>None</i> , <i>Panel1</i> , <i>Panel2</i> .
<i>IsSplitterFixed</i>	Mang hai giá trị <i>True</i> và <i>False</i> . Nếu là <i>True</i> , cố định vạch phân cách <i>Splitter</i>
<i>Orientation</i>	Xác định vạch phân cách <i>Splitter</i> sẽ phân cách theo chiều ngang hay dọc. Gồm 2 giá trị: <i>Vertical</i> , <i>Horizontal</i> .
<i>Panel1Collapsed</i>	Mang hai giá trị <i>True</i> hoặc <i>False</i> . Nếu là <i>True</i> sẽ ẩn <i>Panel1</i>
<i>Panel1MinSize</i>	Lấy kích thước nhỏ nhất hoặc thiết lập kích thước

	nhỏ nhất cho Panel1
<i>Panel2Collapsed</i>	Mang hai giá trị True hoặc False. Nếu là True sẽ ẩn Panel2
<i>Panel2MinSize</i>	Lấy kích thước nhỏ nhất hoặc thiết lập kích thước nhỏ nhất cho Panel2
<i>SplitterDistance</i>	Trả về khoảng cách bằng pixel từ Splitter đến cạnh bên trái (nếu hai Panel nằm dọc) hay đến cạnh trên (nếu hai Panel nằm ngang)
<i>SplitterWidth</i>	thiết lập độ rộng của vạch phân cách Splitter

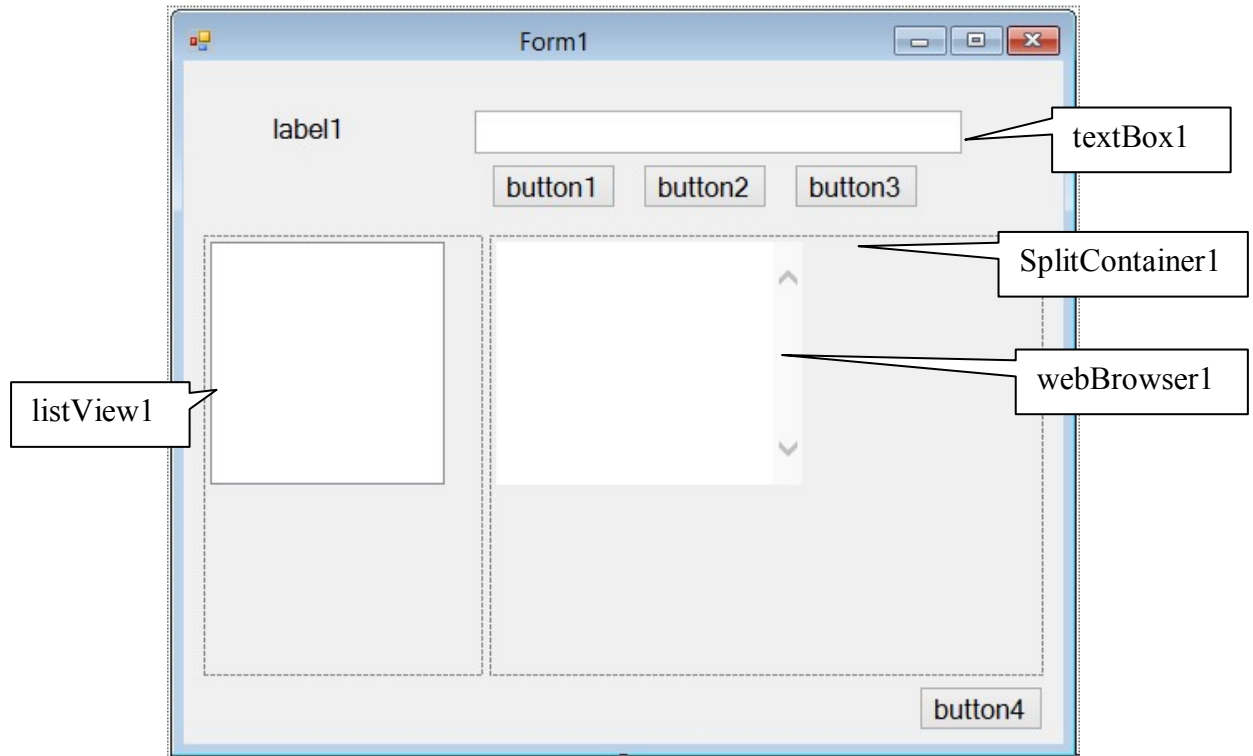
Ví dụ 6.4: Viết chương trình có giao diện như hình 6.27. Gồm: 1 điều khiển SplitContainer với Panel1 chứa 1 điều khiển ListView và Panel2 chứa 1 điều khiểnWebBrowser.

Yêu cầu: Người dùng nhập đường dẫn website vào TextBox và nhấn nút thêm. Đường dẫn website vừa nhập sẽ được đưa vào ListView. Người dùng có thể hiển thị bất cứ website nào trên WebBrowser bằng cách nhấp chuột vào đường dẫn chứa trong ListView.



Hình 6.27: Giao diện quản lý địa chỉ website

Bước 1: Thiết kế giao diện ban đầu cho form. Thêm các điều khiển Label, TextBox, Button, SplitContainer và ListView, WebBrowser từ cửa sổ Toolbox vào form như hình 6.28.



Hình 6.28: Giao diện form sau khi thêm điều khiển

Bước 2: Thiết lập giá trị thuộc tính cho các điều khiển trong cửa sổ Properties

- Form1:
Thuộc tính Text: “SplitContainer”
- label1:
Thuộc tính Text: “Thêm đường dẫn:”
- textBox1:
Thuộc tính Name: txtLink
- button1:
Thuộc tính Name: btnThem
Thuộc tính Text: “Thêm”
- button2:
Thuộc tính Name: btnCapNhat
Thuộc tính Text: “Cập nhật”
- button3:

Thuộc tính Name: btnXoa

Thuộc tính Text: “Xóa”

- button4:

Thuộc tính Name: btnThoat

Thuộc tính Text: “Thoát”

- listView1:

Thuộc tính Name: listLinkWebsite

Thuộc tính Dock: Fill

- webBrowser1:

Thuộc tính Name: myWebsite

Thuộc tính Dock: Fill

- splitContainer1:

Thuộc tính BorderStyle: FixedSingle

Thuộc tính Dock: None

Thuộc tính Orientation: Vertical

Thuộc tính IsSplitterFixed: True

Bước 3: Viết mã lệnh cho điều khiển

- Sự kiện *Click* của btnThoat:

```
private void btnThoat_Click(object sender, EventArgs e)
{
    Close();
}
```

- Sự kiện *Load* của Form1:

```
private void Form1_Load(object sender, EventArgs e)
{
    myWebsite.Navigate("www.google.com");
}
```

- Sự kiện *Click* của nút btnThem:

```
private void btnThem_Click(object sender, EventArgs e)
{
    ListViewItem lvi = new ListViewItem(txtLink.Text);
    listLinkWebsite.Items.Add(lvi);
    txtLink.Text = "";
}
```

- Sự kiện *Click* của nút btnCapNhat:

```
private void btnCapNhat_Click(object sender, EventArgs e)
{
    listLinkWebsite.FocusedItem.Text = txtLink.Text;
}
```

- Sự kiện *Click* của nút btnXoa:

```
private void btnXoa_Click(object sender, EventArgs e)
{
    int i = listLinkWebsite.FocusedItem.Index;
    listLinkWebsite.Items.RemoveAt(i);
    txtLink.Text = "";
    myWebsite.Navigate("www.google.com");
}
```

- Sự kiện *MouseClicked* của listLinkWebsite:

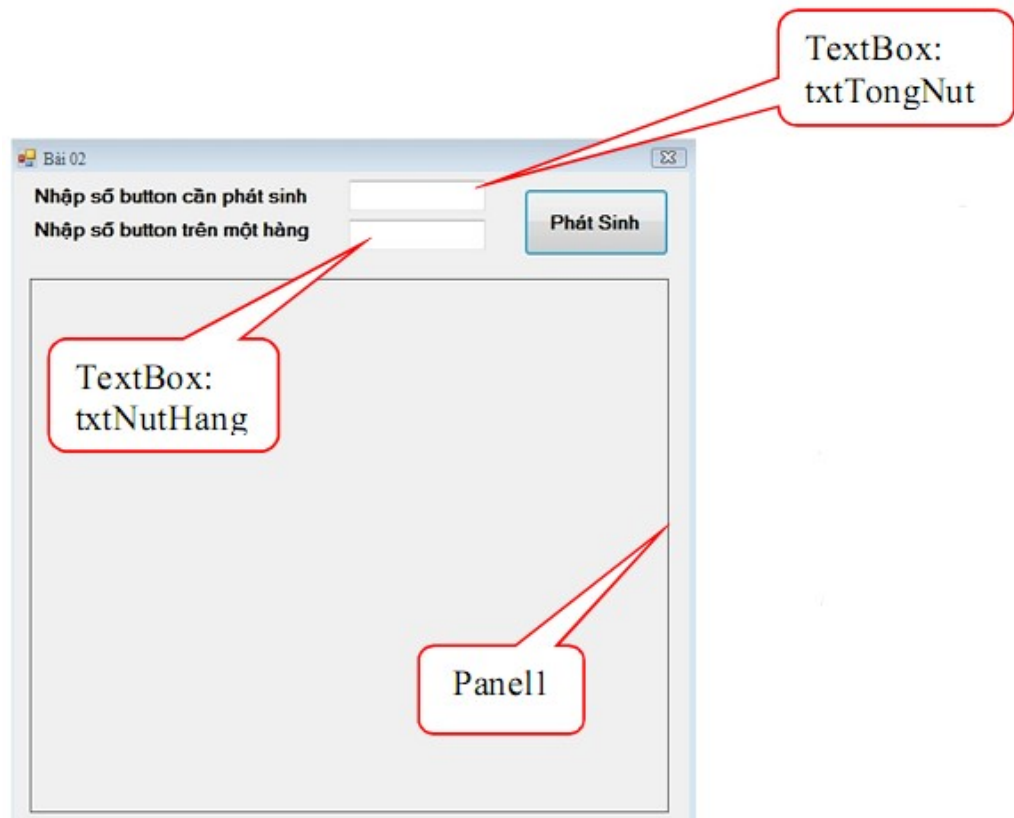
```
private void listLinkWebsite_MouseClick(object sender,
MouseEventArgs e)
{
    txtLink.Text = listLinkWebsite.FocusedItem.Text;
    if (e.Button == MouseButton.Left)
        myWebsite.Navigate(listLinkWebsite.FocusedItem.Text);
}
```

6.7. Bài tập cuối chương

Câu 1: Thiết kế chương trình tạo Button có giao diện như hình 6.29.

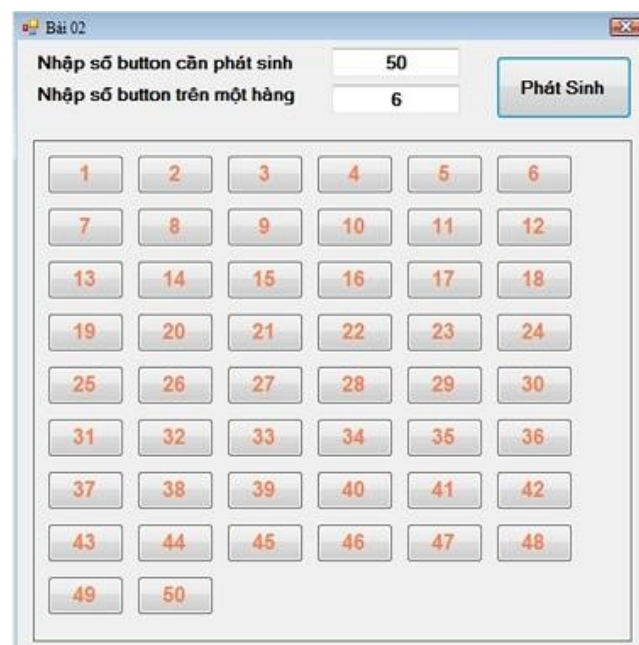
Yêu cầu:

- Khi nhấn F5 chạy chương trình xuất hiện giao diện như hình 6.29.
- Người dùng nhập tổng số Button cần phát sinh trong TextBox txtTongNut; nhập số Button muốn phát sinh trên một hàng trong TextBox txtNutHang.



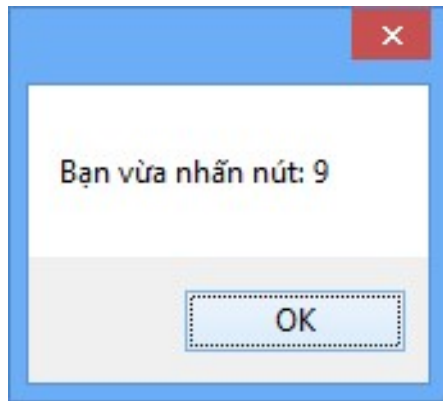
Hình 6.29: Giao diện chương trình tạo Button

- Khi người dùng nhấn nút Phát Sinh sẽ phát sinh các Button như yêu cầu vào Panel1 như hình 6.30.

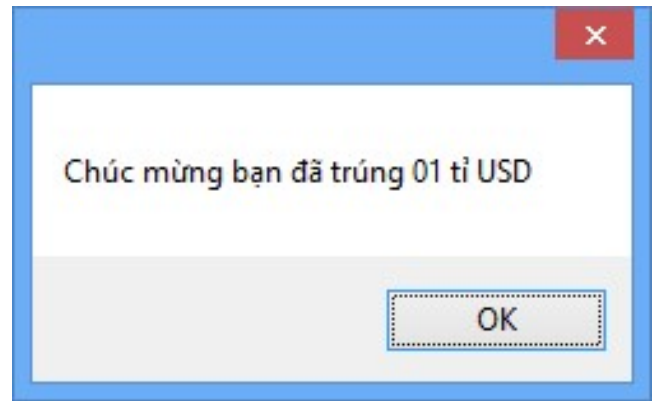


Hình 6.30: Giao diện chương trình sau khi phát sinh Button

- Khi người dùng nhấn vào một nút nhấn xuất thông báo “Bạn vừa nhấn nút: _” như hình 6.31. Riêng nút số 10 khi người dùng nhấn sẽ xuất thông báo “Chúc mừng bạn đã trúng 01 tỉ USD” như hình 6.32.



Hình 6.31: MessageBox thông báo nút người dùng vừa nhấn



Hình 6.32: MessageBox thông báo người dùng vừa nhấn nút 10

Câu 2: Viết chương trình tạo câu hỏi trắc nghiệm như hình 6.33.

 A screenshot of a Windows application titled "Bài tập" (Exercise). The main window contains a form titled "Tạo câu hỏi trắc nghiệm" (Create multiple-choice question). The form has two main sections: "Nhập nội dung câu hỏi" (Enter question content) and "Nhập đáp án" (Enter answers). The first section contains a text box with the text "Trường nào đạt giải nhất cuộc thi Robocon toàn quốc năm 2014". The second section contains a text box for "Nội dung đáp án" (Answer content), a checkbox labeled "Đáp án đúng" (Correct answer) with the label "checkBox1" next to it, and a "Thêm" (Add) button. Below this is a table with two columns: "Nội dung đáp án" and "Đáp án đúng". The table contains four rows of data. At the bottom of the form are two buttons: "Hoàn thành" (Finish) and "Thoát" (Exit). Callout boxes point to various controls: "textBox1" points to the question text box, "textBox2" points to the answer text box, "checkBox1" points to the "Đáp án đúng" checkbox, and "listView1" points to the table.

Nội dung đáp án	Đáp án đúng
Đại học Duy Tân	0
Đại học Lạc Hồng	1
Đại học Bách Khoa Tp HCM	0
Đại học Sư phạm Kỹ thuật Hà Nội	0

Hình 6.33: Giao diện tạo câu hỏi trắc nghiệm

Yêu cầu:

- Người dùng nhập nội dung câu hỏi trong textBox1
- Người dùng nhập đáp án trả lời trong textBox2. Nếu đáp án đúng thì checkBox1 sẽ được chọn tương ứng với giá trị 1. Nếu đáp án sai thì checkBox1 sẽ không được chọn tương ứng với giá trị 0.
- Người dùng nhấn Button “Thêm” để thêm đáp án vào listView1
- Nhấn Button “Thoát” để đóng chương trình.
- Nhấn Button “Hoàn thành” để hoàn tất việc tạo 1 câu hỏi trắc nghiệm. Khi đó sẽ hiển thị form như hình 6.34.

Hình 6.34: Giao diện trả lời trắc nghiệm

Trên giao diện trả lời trắc nghiệm như hình 6.34:

- Nội dung câu hỏi nhập trong textBox1 ở hình 6.33 sẽ hiển thị trên label1.
- Các đáp án trả lời trong listView1 ở hình 6.33 tương ứng sẽ tạo thành các radioButton đáp án để người dùng chọn trong flowLayoutPanel1.

- Nhấn Button “Trả lời” để hoàn thành việc chọn đáp án đúng, nếu chọn đúng thì MessageBox với nội dung “Bạn đã trả lời đúng” được hiển thị, nếu chọn sai thì MessageBox với nội dung “Bạn đã trả lời sai” được hiển thị.