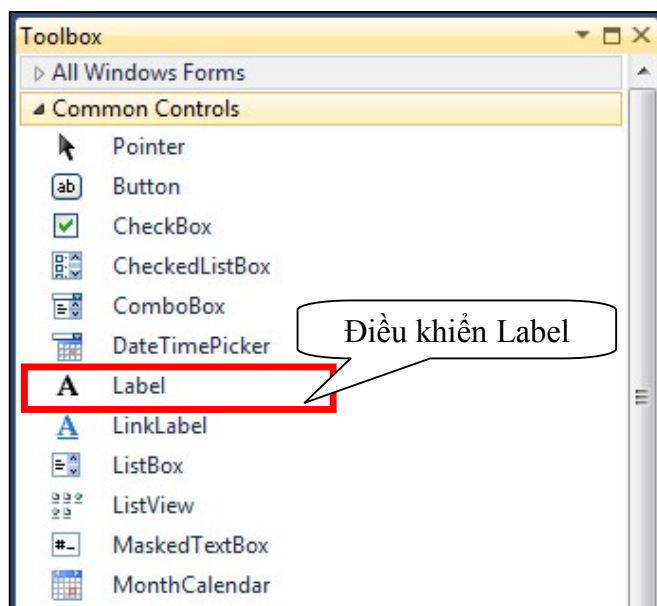


CHƯƠNG 3: CÁC ĐIỀU KHIỂN THÔNG THƯỜNG

3.1. Điều khiển Label

Label thường dùng hiển thị thông tin chỉ đọc và thường sử dụng kèm với các điều khiển khác để mô tả chức năng. *Label* được đặt trong nhóm *Common Controls* của cửa sổ *Toolbox* như hình 3.1.



Hình 3.1: Điều khiển *Label* trong cửa sổ *Properties*

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

Bảng 3.1: Bảng mô tả thuộc tính *Label*

Thuộc tính	Mô tả
<i>BorderStyle</i>	Thiết lập đường viền
<i>Font</i>	Kích thước và kiểu chữ hiển thị trên <i>Label</i>
<i>Text</i>	Nội dung hiển thị trên <i>Label</i>
<i>BackColor</i>	Màu nền của <i>Label</i>
<i>Name</i>	Tên của <i>Label</i>
<i>Locked</i>	Khóa không cho di chuyển
<i>Enabled</i>	Đánh dấu tích là đối tượng sẽ ở trạng thái hoạt động
<i>Cursor</i>	Đổi hình trỏ chuột khi đưa vào <i>Label</i>
<i>ForeColor</i>	Thiết lập màu chữ hiển thị trên <i>Label</i>
<i>TextAlign</i>	Căn lề chữ hiển thị trên <i>Label</i>
<i>Visible</i>	Ẩn hoặc hiện <i>Label</i>

Ví dụ 1: Thiết kế form hiển thị thông tin sinh viên như hình 3.2:

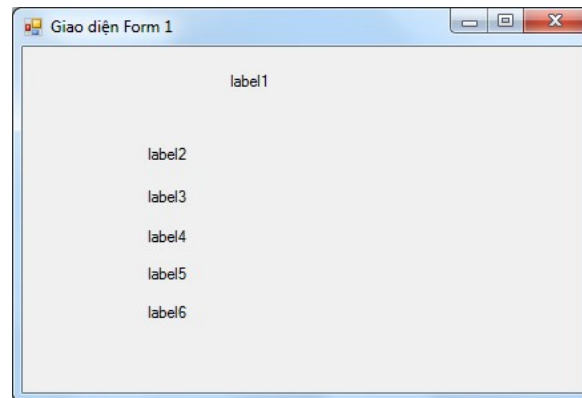


Hình 3.2: Giao diện hiển thị thông tin sinh viên

Hướng dẫn:

Bước 1: Tạo dự án Windows Forms mới, Form1 mặc định tạo sẵn trong dự án đặt tên tiêu đề trên titlebar như sau: “Giao diện Form1”

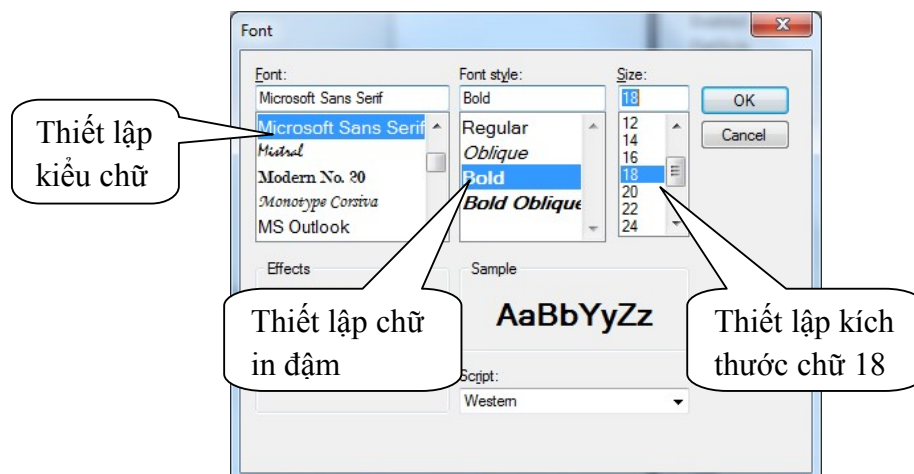
Bước 2: Kéo thả các *Label* trong cửa sổ *Toolbox* vào Form1 như hình 3.3 sau:



Hình 3.3: Giao diện sau khi kéo Label vào Form1

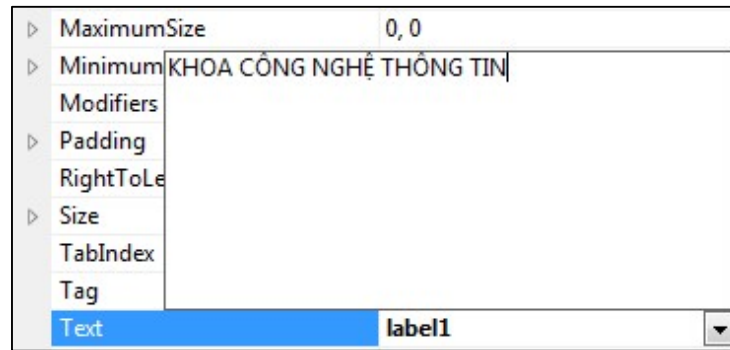
Bước 3: Thiết lập giá trị thuộc tính cho label1

Nhấp chuột trái vào Form1 và thiết lập thuộc tính *Font* trong cửa sổ *Properties* như hình 3.4:



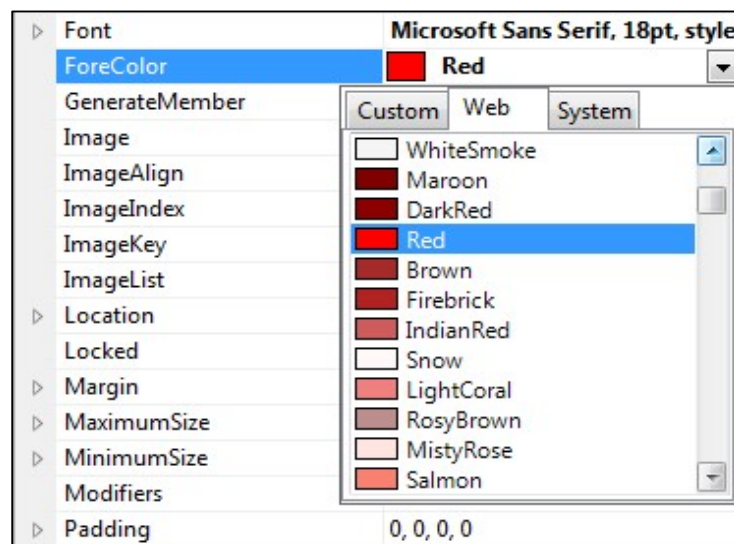
Hình 3.4: Cửa sổ thiết lập thuộc tính Font

Thiết lập thuộc tính *Text* cho label1 như hình 3.5:



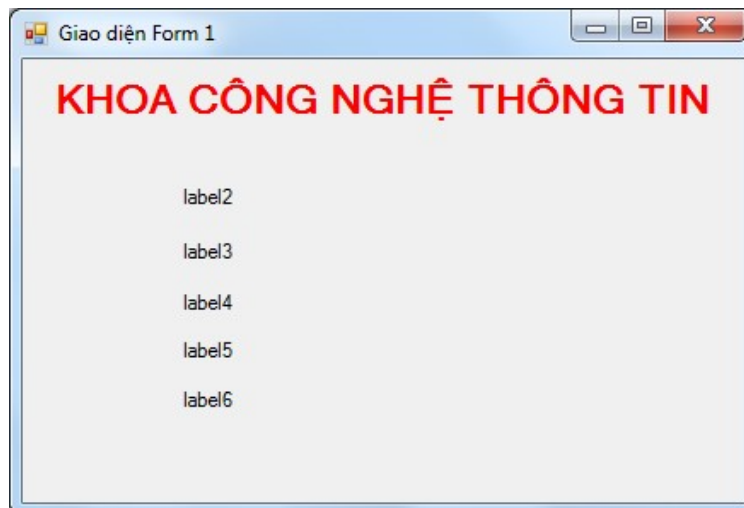
Hình 3.5: Thiết lập thuộc tính *Text* cho label1

Thiết lập thuộc tính *ForeColor* như hình 3.6:



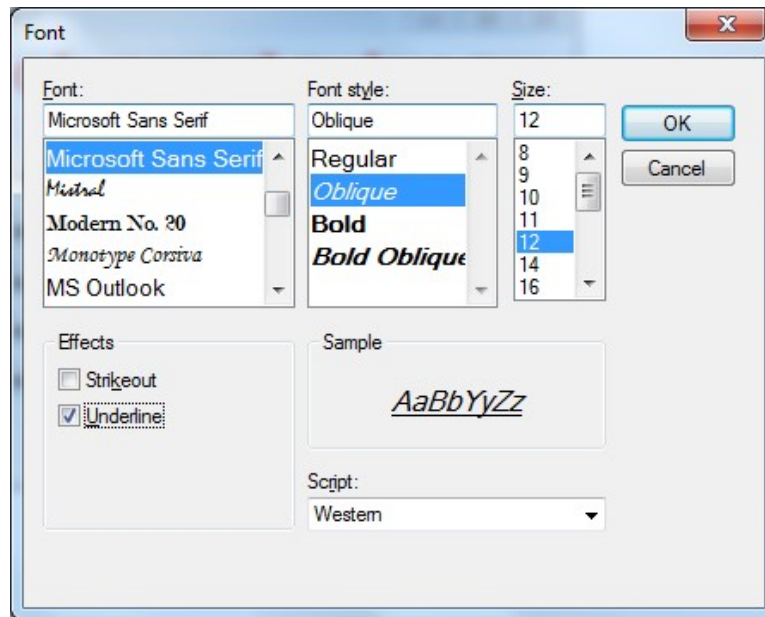
Hình 3.6: Thiết lập thuộc tính *ForeColor*

Sau khi thiết lập xong thuộc tính *Text*, giao diện Form1 được như hình 3.7:



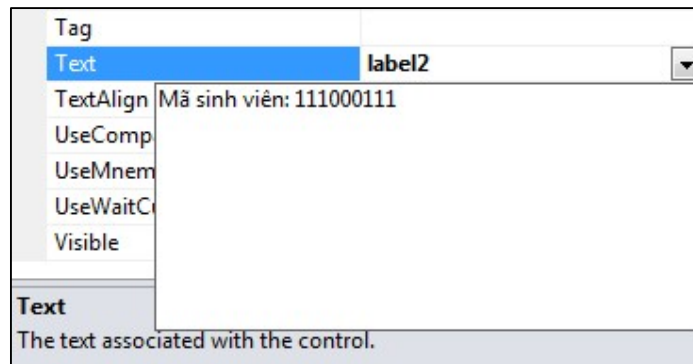
Hình 3.7: Giao diện Form1 sau khi thiết lập xong thuộc tính cho label1

Bước 4: Thiết lập các thuộc tính *Font* cho label2, label3, label4, label5, label6 như hình 3.8:



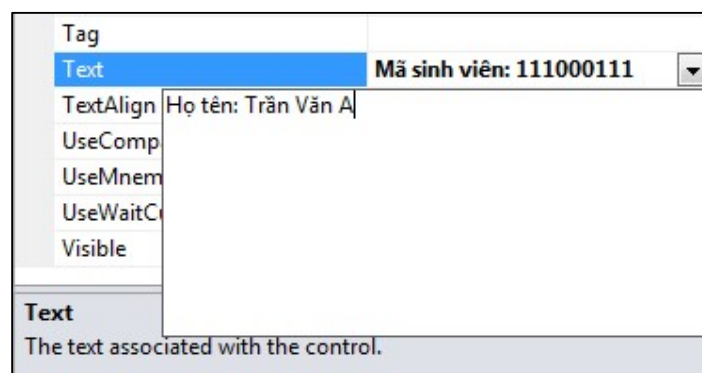
Hình 3.8: Thiết lập thuộc tính *Font* cho label2, label3, label4, label5, label6

Thiết lập thuộc tính *Text* cho label2 như hình 3.9:



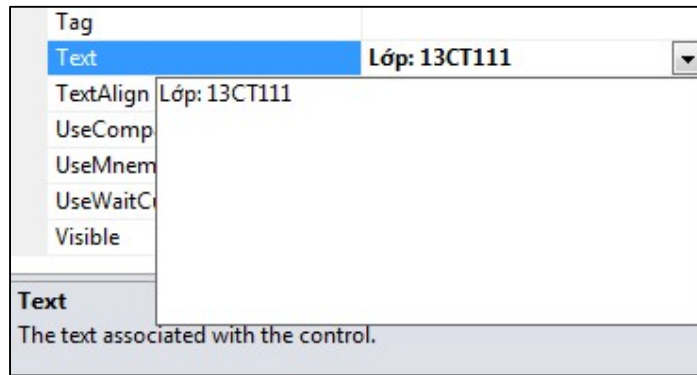
Hình 3.9: Thiết lập thuộc tính *Text* cho label2

Thiết lập thuộc tính *Text* cho label3 như hình 3.10:



Hình 3.10: Thiết lập thuộc tính *Text* cho label3

Thiết lập thuộc tính *Text* cho label4 như hình 3.11:



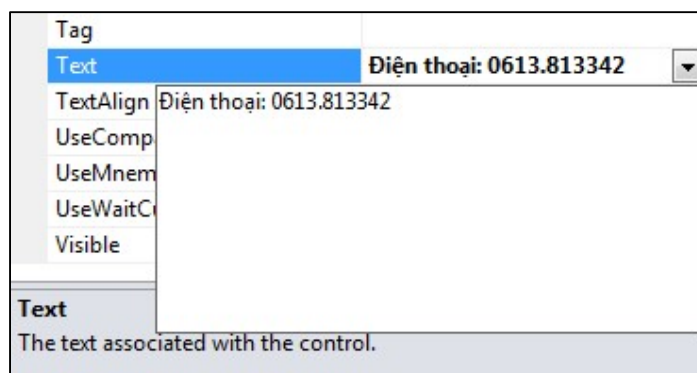
Hình 3.11:Thiết lập thuộc tính *Text* cho label4

Thiết lập thuộc tính *Text* cho label5 như hình 3.12:



Hình 3.12:Thiết lập thuộc tính *Text* cho label5

Thiết lập thuộc tính *Text* cho label6 như hình 3.13:



Hình 3.13: Thiết lập thuộc tính *Text* cho label5

Bước 5: Nhấn F5 thực thi chương trình sẽ được form như hình 3.2.

3.2. Điều khiển Button

Button là điều khiển tạo giao diện nút lệnh trên form, khi người dùng nhấn chuột vào nút lệnh thì chương trình sẽ thực hiện một hành động nào đó.*Button* được đặt trong nhóm *Common Controls* của cửa sổ *Toolbox* như hình 3.14.



Hình 3.14: Điều khiển Button trong cửa sổ Properties

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

Bảng 3.2: Bảng mô tả thuộc tính Button

Thuộc tính	Mô tả
<i>Name</i>	Đặt tên cho nút lệnh
<i>Text</i>	Nội dung hiển thị trên nút nhấn
<i>Visible</i>	Ấn/ hiển nút nhấn
<i>Enable</i>	Cho phép/ không cho phép tương tác với nút lệnh
<i>Font</i>	Chỉ định kiểu chữ, kích cỡ chữ hiển thị
<i>ForeColor</i>	Màu chữ hiển thị trên nút lệnh
<i>Image</i>	Hình ảnh hiển thị trên nút lệnh
<i>BackColor</i>	Màu của nút lệnh
<i>TabIndex</i>	Chỉ định thứ tự tab của các <i>Button</i> trên form

- Một số sự kiện thường dùng của *Button*:

Bảng 3.3: Bảng mô tả sự kiện Button

Sự kiện	Mô tả
<i>Click</i>	Sự kiện nhấn chuột vào <i>Button</i>
<i>MouseEnter</i>	Chuột nằm trong vùng thấy được của <i>Button</i>
<i>MouseHover</i>	Rê chuột vào vùng của <i>Button</i>
<i>MouseLeave</i>	Rê chuột ra khỏi vùng của <i>Button</i>
<i>MouseMove</i>	Chuột được di chuyển trên <i>Button</i> .

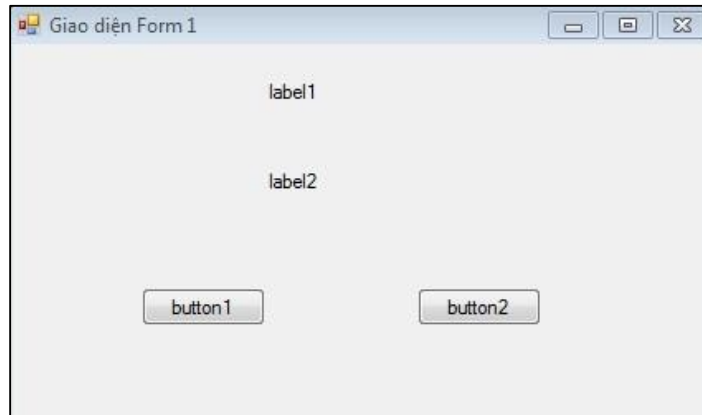
Ví dụ 2: Thiết kế form xử lý nút lệnh như hình 3.15. Yêu cầu: Khi nhấp chuột vào nút lệnh “**Hiển thị**” thì in sẽ hiển thị chữ “**Xin chào**” tại vị trí “---nội dung hiển thị---”, khi nhấp chuột vào nút lệnh “**Thoát**” sẽ đóng chương trình.



Hình 3.15: Giao diện form xử lý nút lệnh

Hướng dẫn:

Bước 1: Kéo các điều khiển từ cửa sổ *Toolbox* vào form như hình 3.16



Hình 3.16: Giao diện ban đầu của form xử lý nút lệnh

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

- label1:
Thuộc tính *Font*: kích cỡ chữ 18;
Thuộc tính *ForeColor*: đỏ;
Thuộc tính *Text*: “KHOA CÔNG NGHỆ THÔNG TIN”
- label2:
Thuộc tính *Name*: lblNoiDung;
Thuộc tính *Font*: kích cỡ chữ 20;
Thuộc tính *Text*: “-----nội dung hiển thị-----”
- button1:
Thuộc tính *Name*: btnHienThi
Thuộc tính *Text*: “Hiển thị”
- button2:
Thuộc tính *Name*: btnThoat
Thuộc tính *Text*: “Thoát”

Bước 3: Viết mã lệnh cho các nút lệnh:

- Sự kiện *Click* của nút lệnh btnHienThi:

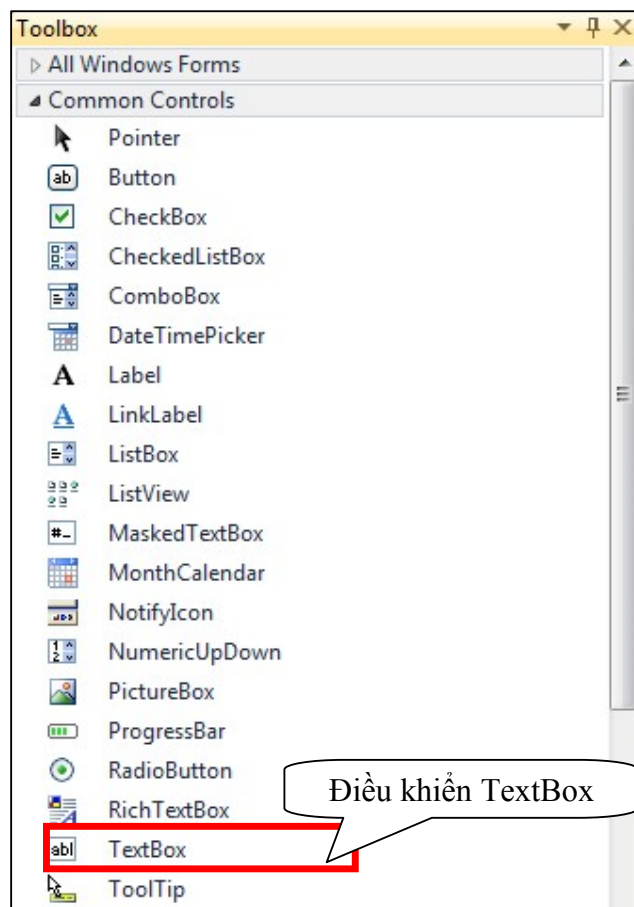
```
private void btnHienThi_Click(object sender, EventArgs e)
{
    lblNoiDung.Text = "Xin chào"
}
```

- Sự kiện *Click* của nút lệnh btnThoat:

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

3.3. Điều khiển TextBox

TextBox là điều khiển dùng để nhập chuỗi làm dữ liệu đầu vào cho ứng dụng hoặc hiển thị chuỗi. Người dùng có thể nhập nhiều dòng, hoặc tạo mặt nạ để nhập mật khẩu. *TextBox* được đặt trong nhóm *Common Controls* của cửa sổ *Toolbox* như hình 3.14.



Hình 3.14: Điều khiển *TextBox* trong cửa sổ *Properties*

- Một số phương thức thường dùng của *TextBox*:

Bảng 3.4 : Bảng mô tả các phương thức của TextBox

Phương thức	Mô tả
<i>Clear()</i>	Xóa tất cả chuỗi hiển thị trong <i>TextBox</i>
<i>Cut()</i>	Di chuyển phần nội dung bôi đen của chuỗi
<i>Past()</i>	Dán phần nội dung được chọn của chuỗi
<i>Copy()</i>	Sao chép phần nội dung bôi đen của chuỗi
<i>Undo()</i>	Khôi phục thao tác trước
<i>Select()</i>	Chọn một phần nội dung của chuỗi trong <i>TextBox</i>
<i>SelectAll()</i>	Chọn tất cả nội dung của chuỗi trong <i>TextBox</i>
<i>DeselectAll()</i>	Bỏ chọn chuỗi trong <i>TextBox</i>

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

Bảng 3.5: Bảng mô tả các thuộc tính của TextBox

Thuộc tính	Mô tả
<i>Name</i>	Tên của <i>TextBox</i> để gọi viết lệnh
<i>Text</i>	Nội dung hiển thị ban đầu khi chương trình chạy
<i>Font</i>	Chọn kiểu chữ, kích thước chữ cho <i>TextBox</i>
<i>ForeColor</i>	Chọn màu chữ cho <i>TextBox</i>
<i>BackColor</i>	Chọn màu nền cho <i>TextBox</i>
<i>Enable</i>	Cho phép/Không cho phép thao tác trên <i>TextBox</i>
<i>Multiline</i>	Cho phép <i>TextBox</i> có nhiều dòng hay không
<i>PasswordChar</i>	Ký tự thay thế khi nhập vào <i>TextBox</i>
<i>ReadOnly</i>	Cho phép/Không cho phép sửa dữ liệu trên <i>TextBox</i>
<i>Visible</i>	Ẩn/ hiển <i>TextBox</i>
<i>TextAlign</i>	Canh lề dữ liệu trong <i>TextBox</i>
<i>MaxLength</i>	Quy định chuỗi Max sẽ nhập vào <i>TextBox</i> , mặc định là 32767
<i>ScrollBars</i>	Các loại thanh cuộn (dọc, ngang, cả hai)
<i>WordWrap</i>	Cho phép văn bản tự động xuống dòng khi độ dài vượt quá chiều ngang của <i>TextBox</i>
<i>BorderStyle</i>	định kiểu đường viền cho <i>TextBox</i>
<i>TabIndex</i>	Chỉ định thứ tự tab của các <i>TextBox</i> trên form

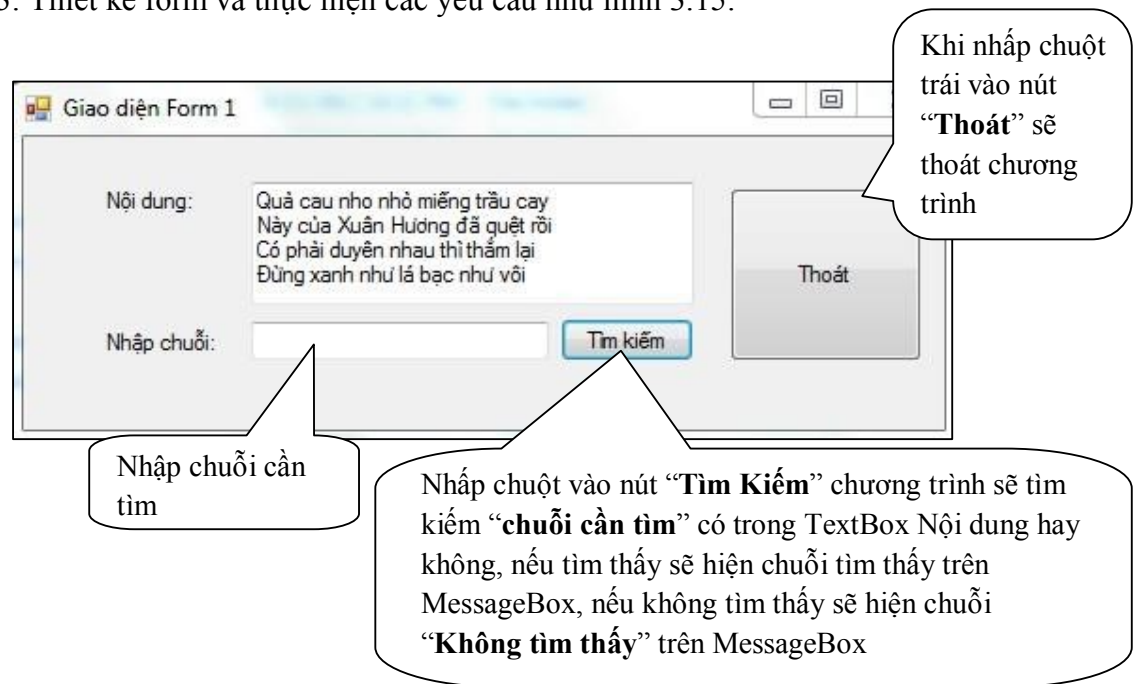
<i>Focus</i>	<i>TextBox</i> sẵn sàng được tương tác bởi người sử dụng
<i>CanUndo</i>	Mang giá trị True hoặc False, nếu là True thì cho phép thực hiện phương thức <i>Undo()</i> , mang giá trị True khi <i>TextBox</i> đã thực hiện thao tác thêm, sửa hay xóa nội dung.
<i>SelectedText</i>	Lấy ra phần nội dung chuỗi được bôi đen
<i>SelectionStart</i>	Vị trí bắt đầu chọn nội dung của chuỗi
<i>SelectionLength</i>	Chiều dài chuỗi sẽ chọn trong <i>TextBox</i>
<i>HideSelection</i>	Mang giá trị True hoặc False, nếu là True thì không cho phép sử dụng thuộc tính <i>SelectionStart</i> , nếu là giá trị False thì cho phép sử dụng <i>SelectionStart</i>

- Một số sự kiện thường dùng của *TextBox*:

Bảng 3.6: Bảng mô tả các sự kiện của *TextBox*

Sự kiện	Mô tả
<i>KeyDown</i>	Thực hiện công việc nào đó khi một phím được nhấn xuống
<i>KeyUp</i>	Thực hiện công việc nào đó khi một phím được nhả ra
<i>KeyPress</i>	Xảy ra khi người sử dụng nhấn một phím và nhả ra, ta dùng sự kiện này để lọc các phím không muốn nhận như cho nhập số (0 đến 9) không cho nhập chuỗi. Mỗi sự kiện <i>KeyPress</i> cho ta một cặp sự kiện <i>KeyDown</i> và <i>KeyUp</i>
<i>Click</i>	Nhấp chuột vào <i>TextBox</i>
<i>DoubleClick</i>	Nhấp đúp chuột vào <i>TextBox</i>
<i>MouseEnter</i>	Chuột nằm trong vùng thấy được của <i>TextBox</i>
<i>MouseHover</i>	Chuột nằm trong vùng hiển thị một quãng thời gian
<i>MouseLeave</i>	Chuột ra khỏi vùng nhập liệu của <i>TextBox</i>
<i>MouseMove</i>	Chuột được di chuyển trên <i>TextBox</i>
<i>TextChanged</i>	Giá trị của thuộc tính <i>Text</i> bị thay đổi

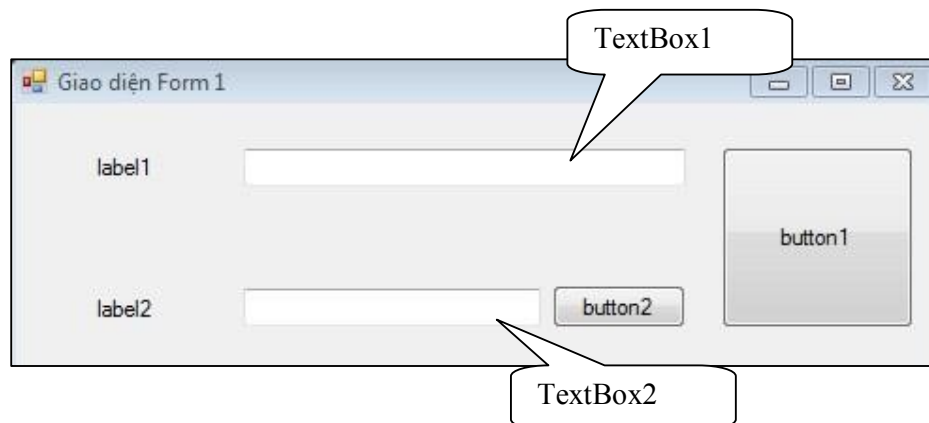
Ví dụ 3: Thiết kế form và thực hiện các yêu cầu như hình 3.15:



Hình 3.15: Giao diện form xử lý TextBox

Hướng dẫn:

Bước 1: Thiết kế form ban đầu như hình 3.16



Hình 3.16: Giao diện ban đầu của form xử lý TextBox

Bước 2: Thiết lập các thuộc tính của điều khiển trong cửa sổ *Properties*:

- label1:
Thuộc tính *Text* = “Nội dung.”
- label2:
Thuộc tính *Text* = “Nhập chuỗi:”
- textbox1:
Thuộc tính *Name* = txtNoiDung
Thuộc tính *Text*= “Quả cau nho nhỏ miếng trầu cay

Này của Xuân Hương đã quệt rồi
Có phải duyên nhau thì thăm lại
Đừng xanh như lá bạc như vôi”

Thuộc tính *Multiline* = true; Sử dụng chuột điều chỉnh kích thước textbox1 như hình 3.15.

- textbox2:
Thuộc tính *Name* = txtTimKiem
- button1:
Thuộc tính *Name* = btnTimKiem
Thuộc tính *Text* = “Tìm kiếm”
- button2:
Thuộc tính *Name* = btnThoat
Thuộc tính *Text* = “Thoát”

Bước 3: Viết mã lệnh cho các nút lệnh:

- Sự kiện *Click* của nút lệnh btnThoat:

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

- Sự kiện *Click* của nút lệnh btnTimKiem:

```
private void btnTimKiem_Click(object sender, EventArgs e)
{
    int i;
    i = txtNoiDung.Text.IndexOf(txtchuoitk.Text);
    if (i >= 0)
    {
        txtNoiDung.SelectionStart = i;
        txtNoiDung.SelectionLength =
            txtchuoitk.Text.Length;
        MessageBox.Show(txtNoiDung.SelectedText);
    }
    else
        MessageBox.Show("Không tìm thấy");
}
```

3.4. Điều khiển ComboBox và ListBox

ComboBox và ListBox là hai điều khiển có nhiều điểm tương đồng, đều sử dụng để chứa dữ liệu cho phép người dùng lựa chọn.

3.4.1. ListBox

Tại mỗi thời điểm có thể chọn một hoặc nhiều dòng dữ liệu, không cho nhập mới. Điều khiển *ListBox* được đặt trong nhóm *Common Controls* của cửa sổ *Toolbox* như hình 3.17



Hình 3.17: Điều khiển *ListBox* trong cửa sổ *Properties*

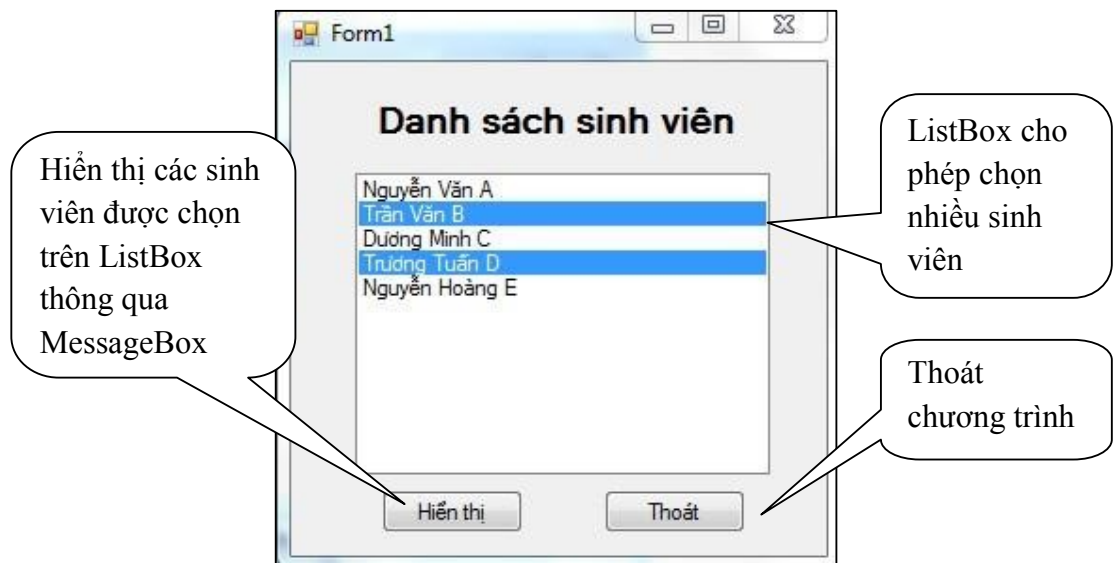
➤ Một số thuộc tính thường dùng của *ComboBox*:

Bảng 3.7: Bảng mô tả các thuộc tính của *ComboBox*

Thuộc tính	Mô tả
<i>SelectionMode</i>	Cho phép chọn một hoặc nhiều dòng dữ liệu trên <i>ListBox</i> , bao gồm: <ul style="list-style-type: none">▪ One: chỉ chọn một giá trị.▪ MultiSimple: cho phép chọn nhiều, chọn bằng cách click vào mục chọn, bỏ chọn bằng cách click vào mục đã chọn.▪ MultiExtended: chọn nhiều bằng cách nhấn kết hợp với Shift hoặc Ctrl
<i>SelectedItems</i>	Được sử dụng khi <i>SelectionMode</i> là <i>MultiSimple</i> hoặc <i>MultiExtended</i> . Thuộc tính <i>SelectedItems</i> chứa các dòng dữ liệu được chọn.
<i>SelectedIndices</i>	Được sử dụng khi <i>SelectionMode</i> là <i>MultiSimple</i> hoặc <i>MultiExtended</i> . Thuộc tính <i>SelectedItems</i> chứa các chỉ số của các dòng dữ liệu được chọn.
<i>FormatString</i>	Chuỗi định dạng. Tất cả các mục chọn trong <i>ListBox</i> sẽ được định dạng theo chuỗi này. Việc

	định dạng này chỉ thực hiện khi thuộc tính <i>FormattingEnabled</i> được thiết lập là <i>True</i>
<i>FormatingEnable</i>	Mang hai giá trị <i>True</i> và <i>False</i> . Nếu là <i>True</i> thì cho phép các mục chọn trong <i>ListBox</i> có thể định dạng lại. Nếu là <i>False</i> thì không thể định dạng.
<i>Items</i>	Trả về các mục chứa trong <i>ListBox</i>
<i>DataSource</i>	Chọn tập dữ liệu điền vào <i>ListBox</i> . Tập dữ liệu có thể là mảng chuỗi, <i>ArrayList</i> , ..
<i>SelectedIndex</i>	Lấy chỉ số mục được chọn, chỉ số mục chọn đầu tiên là 0
<i>SelectedItem</i>	Trả về mục được chọn
<i>SelectedValue</i>	Trả về giá trị của mục chọn nếu <i>ListBox</i> có liên kết dữ liệu. Nếu không liên kết dữ liệu hoặc thuộc tính <i>ValueMember</i> không được thiết lập thì giá trị thuộc tính <i>SelectedValue</i> là giá trị chuỗi của thuộc tính <i>SelectedItem</i>
<i>ValueMember</i>	Thuộc tính này chỉ định dữ liệu thành viên sẽ cung cấp giá trị cho <i>ListBox</i>

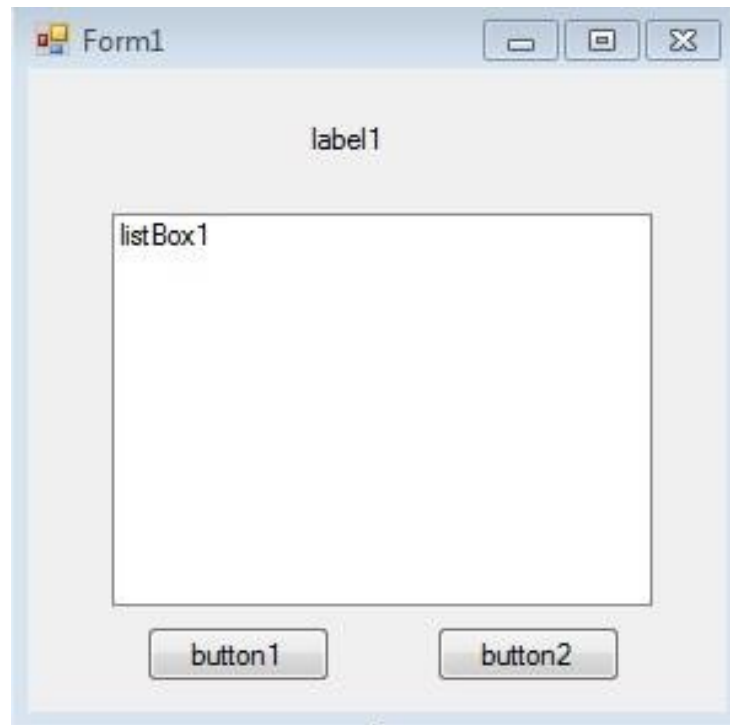
Ví dụ 4: Xây dựng chương trình với *ListBox* chứa danh sách tên của các sinh viên như hình 3.18



Hình 3.18: Giao diện ví dụ 4

Hướng dẫn:

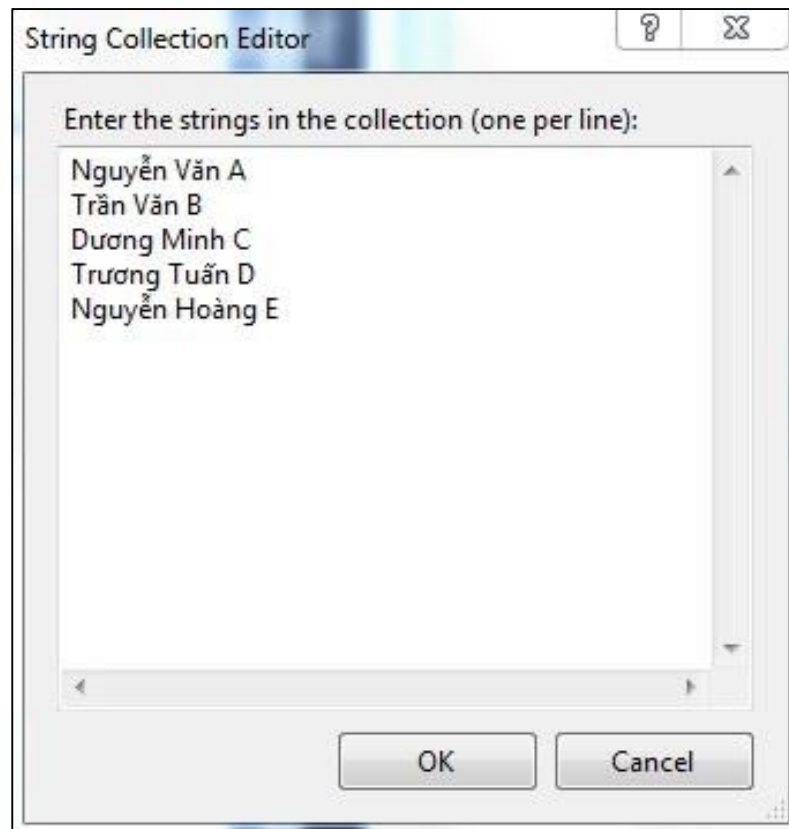
Bước 1: Thiết kế giao diện ban đầu như hình 3.19



Hình 3.19: Giao diện ban đầu ví dụ 4

Bước 2: Thiết lập giá trị các thuộc tính cho điều khiển

- label1:
Thuộc tính *Text* = “Danh sách sinh viên”
Thuộc tính *Size* = 16
- button1:
Thuộc tính *Text* = “Hiển thị”
Thuộc tính *Name* = btnHienThi
- button2:
Thuộc tính *Text* = “Thoát”
Thuộc tính *Name* = btnThoat
- listBox1: listBox1 chứa danh sách tên sinh viên, danh sách tên sinh viên có thể đưa vào listBox1 bằng cách thiết lập thuộc tính *Items* của listBox1 trong cửa sổ *Properties* như hình 3.20



Hình 3.20: Thiết lập thuộc tính Items cho listBox1

Bước 3: Viết mã lệnh cho các nút lệnh:

- Sự kiện *Click* của nút lệnh btnHienThi:

```
private void btnHienThi_Click(object sender, EventArgs e)
{
    string str = "";
    foreach (string item in listBox1.SelectedItems)
    {
        str = str + item + "; ";
    }
    MessageBox.Show(str);
}
```

- Sự kiện *Click* của nút lệnh btnThoat:

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


3.4.2. ComboBox

Mỗi lần chỉ có thể chọn một giá trị, có thể nhập mới dữ liệu vào. Điều khiển *ComboBox* được đặt trong nhóm *Common Controls* của cửa sổ *Toolbox* như hình 3.21



Hình 3.21: Điều khiển *ComboBox* trong cửa sổ *Properties*

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

Bảng 3.8: Bảng mô tả thuộc tính của *ComboBox*

Thuộc tính	Mô tả
<i>Text</i>	Trả về nội dung dòng dữ liệu đang hiển thị trên <i>ComboBox</i>
<i>DropDownStyle</i>	Quy định dạng của <i>ComboBox</i> , nhận một trong các giá trị: <ul style="list-style-type: none">▪ <i>DropDown</i>: giá trị mặc định, có thể chọn hoặc nhập mới mục dữ liệu vào <i>ComboBox</i>.▪ <i>Simple</i>: hiển thị theo dạng <i>ListBox</i> + <i>TextBox</i> có thể chọn dữ liệu từ <i>ListBox</i> hoặc nhập mới vào <i>TextBox</i>.▪ <i>DropDownList</i>: chỉ cho phép chọn dữ liệu trong <i>ComboBox</i>
<i>DropDownHeight</i>	Thiết lập chiều cao tối đa khi sổ xuống của <i>ComboBox</i>
<i>DropDownWidth</i>	Thiết lập độ rộng của mục chọn trong <i>ComboBox</i>
<i>FormatString</i>	Chuỗi định dạng. Tất cả các mục chọn trong <i>ComboBox</i> sẽ được định dạng theo chuỗi này. Việc định dạng này chỉ thực hiện khi thuộc tính <i>FormattingEnabled</i> được thiết lập là <i>True</i>
<i>FormatingEnable</i>	Mang hai giá trị <i>True</i> và <i>False</i> . Nếu là <i>True</i> thì cho phép các mục chọn trong <i>ComboBox</i> có thể định dạng lại. Nếu là <i>False</i> thì không thể định dạng.
<i>Items</i>	Trả về các mục chứa trong <i>ComboBox</i>

<i>DataSource</i>	Chọn tập dữ liệu điền vào <i>ComboBox</i> . Tập dữ liệu có thể là mảng chuỗi, ArrayList, ..
<i>SelectedIndex</i>	Lấy chỉ số mục được chọn, chỉ số mục chọn đầu tiên là 0
<i>SelectedItem</i>	Trả về mục được chọn
<i>SelectedText</i>	Lấy chuỗi hiển thị của mục chọn
<i>SelectedValue</i>	Trả về giá trị của mục chọn nếu <i>ComboBox</i> có liên kết dữ liệu. Nếu không liên kết dữ liệu hoặc thuộc tính <i>ValueMember</i> không được thiết lập thì giá trị thuộc tính <i>SelectedValue</i> là giá trị chuỗi của thuộc tính <i>SelectedItem</i>
<i>ValueMember</i>	Thuộc tính này chỉ định dữ liệu thành viên sẽ cung cấp giá trị cho <i>ComboBox</i>

Ví dụ 5: Xây dựng chương trình với giao diện như hình 3.22 gồm *ComboBox* chứa danh sách 4 màu (vàng, xanh, đỏ, đen) và một Label hiển thị dòng chữ “**Hoàng Sa, Trường Sa là của Việt Nam**”. Yêu cầu khi chọn màu trong *ComboBox* thì màu của dòng chữ trên Label sẽ thay đổi tương ứng.



Hình 3.22: Giao diện ví dụ 5

Hướng dẫn:

Bước 1: Thiết kế giao diện ban đầu như hình 3.23



Hình 3.23: Giao diện ban đầu ví dụ 5

Bước 2: Thiết lập giá trị thuộc tính cho điều khiển

- label1:

Thuộc tính Text = “Chọn màu:”

- button1:

Thuộc tính Name = btnDoiMau

Thuộc tính Text = “Đổi màu”

- button2:

Thuộc tính Name = btnThoat

Thuộc tính Text = “Thoát”

- label2:

Thuộc tính Name = lblHienThi

Thuộc tính Text = “Trường Sa, Hoàng Sa là của Việt Nam”

- comboBox1: comboBox1 chứa danh sách 4 màu gồm: Vàng, xanh, đỏ, đen. Danh sách màu này có thể đưa vào comboBox1 bằng cách thiết lập thuộc tính *Items* của comboBox1 trong cửa sổ *Properties* như hình 3.24



Hình 3.24: Thiết lập thuộc tính *Items* của *comboBox1*

Bước 3: Viết mã lệnh cho các nút lệnh:

- Sự kiện *Click* của nút lệnh btnDoiMau:

```
private void btnDoiMau_Click(object sender, EventArgs e)
{
    if (comboBox1.Text == "Vàng")
        lblHienThi.ForeColor = Color.Yellow;
    if (comboBox1.Text == "Đỏ")
        lblHienThi.ForeColor = Color.Red;
    if (comboBox1.Text == "Đen")
        lblHienThi.ForeColor = Color.Black;
    if (comboBox1.Text == "Xanh")
        lblHienThi.ForeColor = Color.Blue;
}
```

- Sự kiện *Click* của nút lệnh `btnThoat`:

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

3.4.3. Phương thức và sự kiện của `ComboBox` và `ListBox`

Một số phương thức và thuộc tính thường dùng của `ComboBox.Items` hoặc `ListBox.Items`:

Bảng 3.9: Bảng mô tả các phương thức và thuộc tính của `ComboBox.Item` và `ListBox.Item`

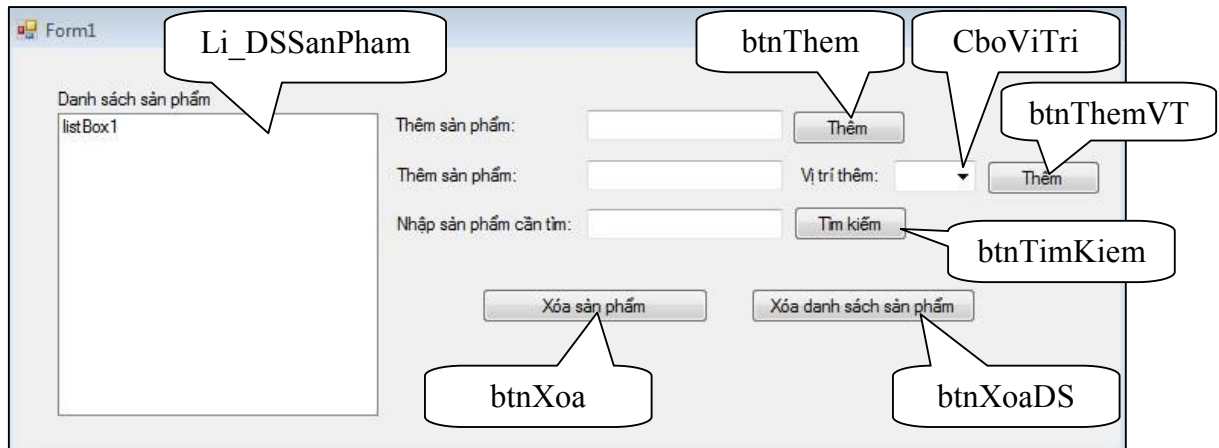
Phương thức	Mô tả
<i>Add(<Mục mới>)</i>	Thêm một mục <Mục mới> vào cuối danh sách <i>ComboBox</i> hoặc <i>ListBox</i>
<i>AddRange(<Mảng mục chọn>)</i>	Thêm một mảng các mục
<i>Insert(i, <Mục mới>)</i>	Chèn thêm mục chọn <Mục mới> vào vị trí <i>i</i>
<i>Count</i>	Trả về số mục chọn hiện đang có
<i>Item(i)</i>	Trả về mục chọn ở vị trí thứ <i>i</i>
<i>Remove(<Mục cần xóa>)</i>	Xóa mục chọn <Mục cần xóa> khỏi <i>ComboBox</i> hoặc <i>ListBox</i>
<i>RemoveAt(i)</i>	Xóa mục chọn có chỉ số <i>i</i> khỏi <i>ComboBox</i> hoặc <i>ListBox</i>
<i>Contains(<Mục cần tìm>)</i>	Trả về <code>True</code> nếu có mục chọn <Mục cần tìm> trong danh sách, trả về <code>False</code> nếu không có mục chọn trong <i>ComboBox</i> hoặc <i>ListBox</i>
<i>Clear()</i>	Xóa tất cả các mục chọn
<i>IndexOf(<Mục cần tìm>)</i>	Trả về chỉ số mục chọn <Mục cần tìm> trong <i>ComboBox</i> hoặc <i>ListBox</i> , nếu không tìm thấy sẽ trả về -1

- Sự kiện thường dùng của `ComboBox` và `ListBox`:

Cả `ComboBox` và `ListBox` đều sử dụng sự kiện `SelectedIndexChanged` để kiểm tra sự thay đổi mục chọn của người dùng.

Ví dụ 5: Thiết kế form quản lý danh sách sản phẩm như hình 3.25. Với `CboViTri` là điều khiển `ComboBox` chứa danh sách chỉ số của danh sách sản phẩm trong `ListBox Li_DSSanPham`. Yêu cầu chức năng:

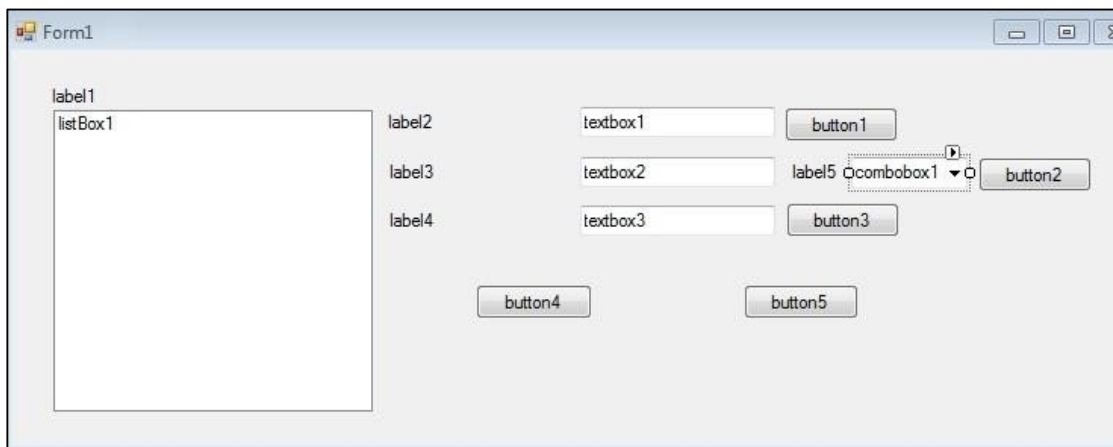
- Khi nhấn nút btnThem thì sẽ thêm sản phẩm mới vào cuối danh sách trong *ListBox* Li_DSSanPham.
- Khi nhấn nút btnThemVT thì sẽ thêm sản phẩm mới vào danh sách tại vị trí như CboVitrì chỉ định.
- Khi nhấn nút btnTimKiem sẽ hiển thị một *MessageBox* thông báo có tìm thấy sản phẩm trong danh sách không.
- Khi nhấn nút btnXoa sẽ xóa sản phẩm được chọn trong danh sách.
- Khi nhấn nút btnXoaDS sẽ xóa tất cả sản phẩm trong danh sách.



Hình 3.25: Giao diện form quản lý sản phẩm

Hướng dẫn:

Bước 1: Thiết kế giao diện ban đầu như hình 3.26



Hình 3.26: Giao diện ban đầu form quản lý sản phẩm

Bước 2: Thiết lập thuộc tính cho điều khiển trong cửa sổ *Properties* như sau:

- listBox1:
Thuộc tính *Name*: Li_DSSanPham
- label1:
Thuộc tính *Text*: “Danh sách sản phẩm:”

- label2:
Thuộc tính *Text*: “Thêm sản phẩm:”
- label3:
Thuộc tính *Text*: “Thêm sản phẩm:”
- label4:
Thuộc tính *Text*: “Nhập sản phẩm cần tìm:”
- label5:
Thuộc tính *Text*: “Vị trí thêm:”
- textbox1:
Thuộc tính *Name*: txtThemSP
- textbox2:
Thuộc tính *Name*: txtThemSPViTri
- textbox3:
Thuộc tính *Name*: txtTimSP
- button1:
Thuộc tính *Text*: “Thêm”
Thuộc tính *Name*: btnThem
- button2:
Thuộc tính *Text*: “Thêm”
Thuộc tính *Name*: btnThemVT
- button3:
Thuộc tính *Text*: “Tìm kiếm”
Thuộc tính *Name*: btnTimKiem
- button4:
Thuộc tính *Text*: “Xóa sản phẩm”
Thuộc tính *Name*: btnXoa
- button5:
Thuộc tính *Text*: “Xóa danh sách sản phẩm”
Thuộc tính *Name*: btnXoaDS
- combobox1:
Thuộc tính *Name*: cboViTri

Bước 3: Viết mã lệnh

- Viết mã lệnh cho hàm ThietLapViTriComboBox:

```
private void ThietLapViTriComboBox()
{
    cboViTri.Items.Clear();
    int chiso = DSSanPham.Items.Count;
    for (int i = 0; i < chiso; i++)
        cboViTri.Items.Add(i.ToString());
}
```

- Sự kiện *Form_Load* cho Form1 như sau:

```
privatevoid Form1_Load(object sender, EventArgs e)
{
    ThietLapViTriComboBox();
}
```

- Sự kiện *Click* nút lệnh btnThem

```
privatevoid btnThem_Click(object sender, EventArgs e)
{
    if (txtThemSP.Text.Trim() != "")
    {
        DSSanPham.Items.Add(txtThemSP.Text);
        txtThemSP.Text = "";
        ThietLapViTriComboBox();
    }
    else
        MessageBox.Show("Phải nhập tên sản phẩm");
}
```

- Sự kiện *Click* nút lệnh btnThemVT

```
privatevoid btnThemVT_Click(object sender, EventArgs e)
{
    if (txtThemSPViTri.Text.Trim() != "" )
    {
        if (cboViTri.Text != "")
        {
            DSSanPham.Items.Insert(Convert.ToInt32(cboViTri.Text),
                                    txtThemSPViTri.Text);
            txtThemSPViTri.Text = "";
            ThietLapViTriComboBox();
        }
        else
            MessageBox.Show("Phải chọn vị trí thêm hợp lệ");
    }
    else
        MessageBox.Show("Phải nhập tên sản phẩm ");
}
```

- Sự kiện *Click* nút lệnh btnXoaDS

```
privatevoid btnXoaDS_Click(object sender, EventArgs e)
{
    if (DSSanPham.Items.Count > 0)
        DSSanPham.Items.Clear();
    else
        MessageBox.Show("Danh sách sản phẩm chưa có gì");
}
```

- Sự kiện *Click* nút lệnh btnXoa

```
privatevoid btnXoa_Click(object sender, EventArgs e)
{
    if (DSSanPham.SelectedIndex < 0)
        MessageBox.Show("Chọn sản phẩm muốn xóa ");
    else
    {
        DSSanPham.Items.Remove(DSSanPham.SelectedItem);
        MessageBox.Show("Xóa sản phẩm thành công");
    }
}
```

- Sự kiện *Click* nút lệnh btnTimKiem

```
privatevoid btnTimKiem_Click(object sender, EventArgs e)
{
    if(txtTimSP.Text != "")
    {
        if(DSSanPham.Items.Contains(txtTimSP.Text) == true)
            MessageBox.Show("Tìm thấy sản phẩm");
        else
            MessageBox.Show("Không Tìm thấy sản phẩm ");
    }
    else
        MessageBox.Show("Nhập tên sản phẩm cần tìm");
}
```

3.5. Điều khiển CheckBox và RadioButton

3.5.1. CheckBox

CheckBox là điều khiển cho phép trình bày các giá trị để lựa chọn trên form, người dùng có thể chọn một hoặc nhiều giá trị cùng lúc. Điều khiển *CheckBox* được đặt trong nhóm *Common Controls* của cửa sổ *Toolbox* như hình 3.27



Hình 3.27: Điều khiển *CheckBox* trên cửa sổ *Toolbox*

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

Bảng 3.10: Bảng mô tả các thuộc tính của *CheckBox*

Thuộc tính	Mô tả
<i>AutoCheck</i>	Mang giá trị True hoặc False, nếu là giá trị True thì cho phép người dùng nhấp chuột để chọn, nếu là False thì không cho phép người dùng nhấp chuột chọn.
<i>Checked</i>	Thiết lập cho điều khiển <i>CheckBox</i> được lựa chọn hoặc không. Thiết lập giá trị True là điều khiển được chọn, nếu thiết lập False là điều khiển không được chọn.
<i>CheckState</i>	Thường dùng để kiểm tra tình trạng <i>CheckBox</i> có được chọn hay không. Mang 3 giá trị <i>Unchecked</i> , <i>Checked</i> , và <i>Indeterminate</i> . <i>Checked</i> : Điều khiển đang được chọn <i>UnChecked</i> : Điều khiển không được chọn <i>Indeterminate</i> : Điều khiển ở trạng thái không hoạt động
<i>Text</i>	Chuỗi văn bản hiển thị bên cạnh <i>CheckBox</i>
<i>ThreeState</i>	Mang giá trị True hoặc False; Nếu là True thì cho phép <i>CheckBox</i> có 3 trạng thái: <i>Checked</i> , <i>UnChecked</i> , <i>Indeterminate</i> .
<i>RightToLeft</i>	Mang giá trị Yes hoặc No; Cho biết chuỗi văn bản hiển thị (thuộc tính <i>Text</i>) nằm bên trái hay bên phải của <i>CheckBox</i>

- Sự kiện thường sử dụng của *CheckBox*:

Sự kiện quan trọng và thường xuyên sử dụng của *CheckBox* là sự kiện *Click* và *CheckedChange*. Hai sự kiện này của *CheckBox* cho biết khi nhấp chuột chọn thì *CheckBox* sẽ ở trạng thái chọn (*Checked*) hay ở trạng thái không chọn (*UnChecked*).

Lập trình viên có thể kiểm tra *CheckBox* đang ở trạng thái nào bằng cách kiểm tra thuộc tính *Checked* hoặc *UnChecked* của *CheckBox*.

Ví dụ 6: Thiết kế chương trình như hình 3.28 và thực hiện các yêu cầu chức năng sau:

Hình 3.28: Giao diện form Thông tin sinh viên ví dụ 6

Yêu cầu:

- Khi nhấn nút “Lưu” sẽ xuất hiện MessageBox hiển thị thông tin gồm: Họ tên sinh viên, lớp, và các môn học sinh viên chọn.
- Khi nhấn nút “Thoát” sẽ đóng chương trình.

Hướng dẫn:

Bước 1: Thiết kế giao diện form như hình 3.29

Hình 3.29: Giao diện ban đầu form Thông tin sinh viên ví dụ 6

Bước 2: Thiết lập thuộc tính điều khiển cho form trong cửa sổ *Properties*

- label1:
Thuộc tính Text: “Thông tin sinh viên”
Thuộc tính Size: 16
- label2:
Thuộc tính Text: “Nhập họ tên:”
- label3:
Thuộc tính Text: “Lớp:”
- label4:
Thuộc tính Text: “Danh sách môn học tự chọn:”
- textBox1:
Thuộc tính Name: txtHoTen
- textBox2:
Thuộc tính Name: txtLop
- checkBox1:
Thuộc tính Text: “Lập trình C#”
Thuộc tính Name: chkCSharp
- checkBox2:
Thuộc tính Text: “Mạng máy tính”
Thuộc tính Name: chkMang
- checkBox3:
Thuộc tính Text: “Xử lý ảnh”
Thuộc tính Name: chkXLAnh
- checkBox4:
Thuộc tính Text: “Lập trình Web”
Thuộc tính Name: chkWeb
- button1:
Thuộc tính Text: “Luu”
Thuộc tính Name: btnLuu
- button2:
Thuộc tính Text: “Thoát”
Thuộc tính Name: btnThoat
- Sự kiện *Click* của nút lệnh btnThoat:

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

Bước 3: Viết mã lệnh cho các điều khiển

- Sự kiện *Click* nút lệnh `btnLuu`

```
private void btnLuu_Click(object sender, EventArgs e)
{
    string str = "";
    str = str + "Họ tên: " + txtHoTen.Text + "\nLớp: "
        + txtLop.Text + "\nDanh sách môn: ";
    if (chkCSharp.Checked == true)
        str = str + "Lập trình C#, ";
    if (chkMang.Checked == true)
        str = str + "Mạng máy tính, ";
    if (chkWeb.Checked == true)
        str = str + "Lập trình Web, ";
    if (chkXLANh.Checked == true)
        str = str + "Xử lý ảnh, ";
    MessageBox.Show(str);
}
```

3.5.2. RadioButton

RadioButton là điều khiển cho phép trình bày các giá trị để lựa chọn trên form. Điểm khác biệt của *RadioButton* với *CheckBox* là người dùng chỉ có thể có một sự lựa chọn với các *RadioButton* cùng nhóm, nghĩa là *RadioButton* có thể phân vào những nhóm khác nhau; Với những *RadioButton* cùng nhóm, khi người dùng nhấp chọn một *RadioButton* thì đồng thời các *RadioButton* khác trong nhóm sẽ lập tức bỏ chọn. Điều khiển *RadioButton* được đặt trong nhóm *Common Controls* của cửa sổ *Toolbox* như hình 3.30



Hình 3.30: Điều khiển *RadioButton* trên cửa sổ *Toolbox*

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

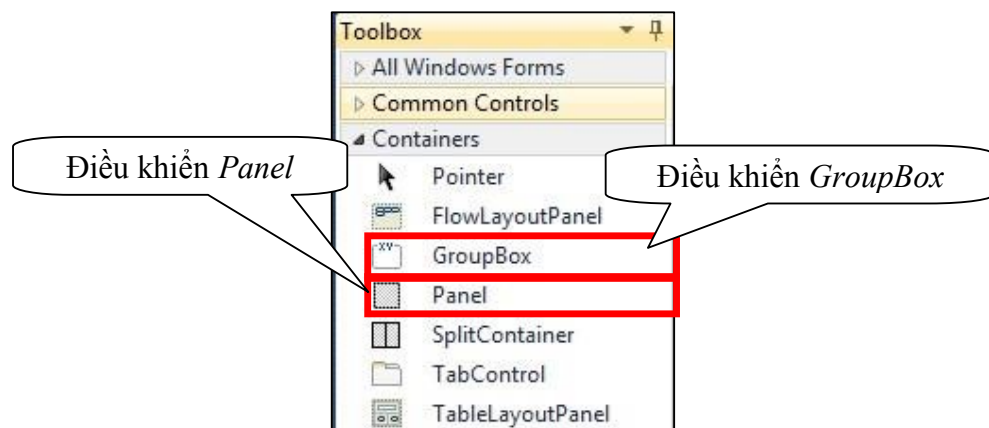
Bảng 3.11 Bảng mô tả các thuộc tính của *RadioButton*

Thuộc tính	Mô tả
<i>Checked</i>	Thiết lập cho điều khiển <i>RadioButton</i> được lựa

	chọn hoặc không. Thiết lập giá trị True là điều khiển được chọn, nếu thiết lập False là điều khiển không được chọn.
<i>Text</i>	Chuỗi văn bản hiển thị bên cạnh <i>RadioButton</i>
<i>RightToLeft</i>	Mang giá trị Yes hoặc No; Cho biết chuỗi văn bản hiển thị (thuộc tính <i>Text</i>) nằm bên trái hay bên phải của <i>RadioButton</i>

Ngoài việc thiết lập thuộc tính *Checked* để thay đổi tình trạng của *RadioButton*, lập trình viên cũng có thể sử dụng để kiểm tra tình trạng của *RadioButton*. Nếu *RadioButton* được chọn thì thuộc tính *Checked* sẽ là *True*, nếu *RadioButton* không được chọn sẽ là *False*.

Tất cả *RadioButton* được chứa trong điều khiển thuộc nhóm *Containers* để tạo thành một nhóm, thông thường sử dụng điều khiển *Panel* hay *GroupBox* như hình 3.31. Điều này cho phép người dùng chỉ có thể chọn duy nhất một *RadioButton* trong nhóm.



Hình 3.31: Điều khiển *GroupBox* trên cửa sổ *Toolbox*

➤ Sự kiện thường sử dụng của *RadioButton*:

Tương tự như *CheckBox*, sự kiện thường dùng của *RadioButton* cũng là hai sự kiện *Click* và *CheckedChange*. Sự kiện *Click* thực hiện khi người dùng nhấp chuột vào *RadioButton*, còn sự kiện *CheckedChange* thực hiện khi tình trạng của *RadioButton* bị thay đổi từ trạng thái chọn (*Checked*) sang trạng thái không chọn (*Unchecked*) và ngược lại.

Ví dụ 7: Thiết kế form Thông tin sinh viên như hình 3.32 và thực hiện yêu cầu chức năng

Hình 3.32: Giao diện form thông tin sinh viên ví dụ 7

Yêu cầu:

- Khi nhấn nút “Đăng ký” thì nội dung người dùng nhập sẽ hiển thị trong điều khiển *TextBox* có tên là *txtNoiDung*. Ví dụ: nhập họ tên là “Nguyễn Văn A”, Khoa là “Công nghệ thông tin”, giới tính chọn là “Nam”, chuyên ngành học là “Công nghệ phần mềm” thì sẽ hiển thị trên *TextBox* là:

Họ tên: Nguyễn Văn A

Khoa: Công nghệ thông tin

Giới tính: Nam

Chuyên ngành: Công nghệ phần mềm

- Khi nhấn nút “Thoát” chương trình sẽ đóng lại

Hướng dẫn:

Bước 1: Thiết kế giao diện ban đầu như hình 3.33



Hình 3.33: Giao diện đầu tiên form Thông tin sinh viên ví dụ 7

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

- label1:
Thuộc tính *Text*: “Thông tin sinh viên”
Thuộc tính *Size*: 16
- label2:
Thuộc tính *Text*: “Họ tên:”
- label3:
Thuộc tính *Text*: “Khoa:”
- textBox1:
Thuộc tính *Name*: txtHoTen
Thuộc tính *Text*: “”
- textBox2:
Thuộc tính *Name*: txtKhoa
Thuộc tính *Text*: “”
- textBox3:
Thuộc tính *Name*: txtNoiDung
Thuộc tính *Text*: “”
Thuộc tính *Multiline*: True
- groupBox1:
Thuộc tính *Text*: “Giới tính”
- groupBox2:
Thuộc tính *Text*: “Chọn ngành học”
- groupBox3:

Thuộc tính *Text*: “Nội dung sinh viên đăng ký”

- button1:

Thuộc tính *Name*: btnDangKy

- button2:

Thuộc tính *Name*: btnThoat

- radioButton1:

Thuộc tính *Name*: radNam

Thuộc tính *Checked*: True

- radioButton2:

Thuộc tính *Name*: radNu

- radioButton3:

Thuộc tính *Name*: radCNPM

Thuộc tính *Checked*: True

- radioButton4:

Thuộc tính *Name*: radKTMT

- radioButton5:

Thuộc tính *Name*: radMMTTT

- radioButton6:

Thuộc tính *Name*: radHTTT

Bước 3: Viết mã lệnh cho các điều khiển

- Sự kiện *Click* nút lệnh btnDangKy:

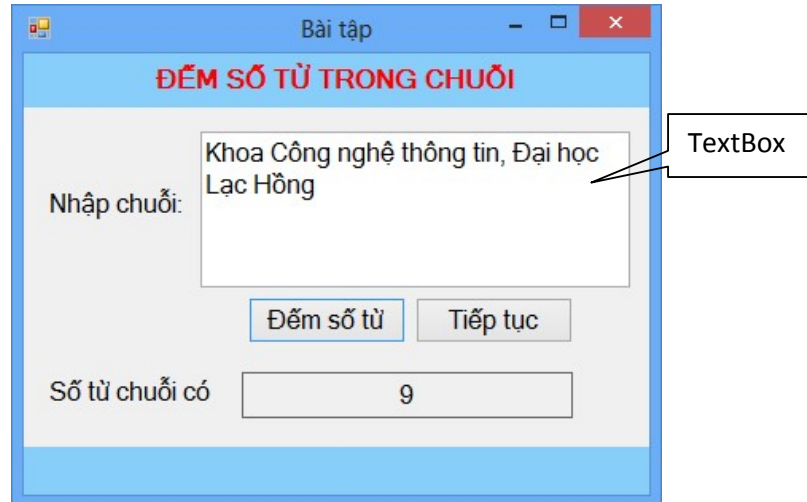
```
private void btnDangKy_Click(object sender, EventArgs e)
{
    txtNoiDung.Text = "";
    string str = "";
    str = "Họ tên: " + txtHoTen.Text + "\r\nKhoa: " +
        txtKhoa.Text + "\r\n";
    if (radNam.Checked == true)
        str = str + "Giới tính: Nam\r\n";
    if (radNu.Checked == true)
        str = str + "Giới tính: Nu\r\n";
    str = str + "Chuyên ngành: ";
    if (radKTMT.Checked == true)
        str = str + " Kỹ thuật máy tính";
    if (radMMTTT.Checked == true)
        str = str + " Mạng máy tính và truyền thông";
    if (radHTTT.Checked == true)
        str = str + " Hệ thống thông tin";
    if (radCNPM.Checked == true)
        str = str + " Công nghệ phần mềm";
    txtNoiDung.Text = str;
}
```


- Sự kiện *Click* của nút lệnh btnThoat:

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

3.6. Bài tập cuối chương

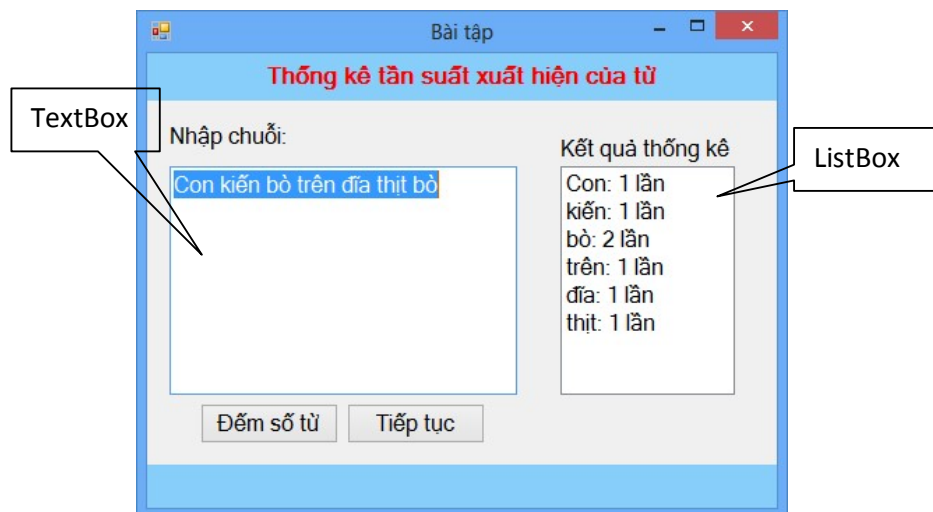
Câu 1: Viết chương trình đếm số từ của chuỗi trong TextBox có giao diện như hình 3.34.



Hình 3.34: Giao diện chương trình đếm từ

- Button “Đếm số từ” sẽ cho biết chuỗi người dùng nhập vào có bao nhiêu từ (xem các từ cách nhau bởi một khoảng trắng).
- Button “Tiếp tục” xóa trắng các textbox về trạng thái như lúc đầu.

Câu 2: Viết chương trình có giao diện như hình 3.35, giúp thống kê tần suất xuất hiện các từ nhập vào.



Hình 3.35: Giao diện chương trình đếm tần suất xuất hiện của từ

- Button “Đếm số từ” sẽ hiển thị tần suất xuất hiện của từ bên ListBox (xem các từ cách nhau bởi một khoảng trắng).
- Button “Tiếp tục” xóa trắng các TextBox và ListBox về trạng thái như lúc đầu.

Câu 3: Viết chương trình giải phương trình bậc hai bao gồm các điều khiển: Label, TextBox và Button như giao diện như hình 3.36.

Hình 3.36: Giao diện chương trình giải phương trình bậc hai

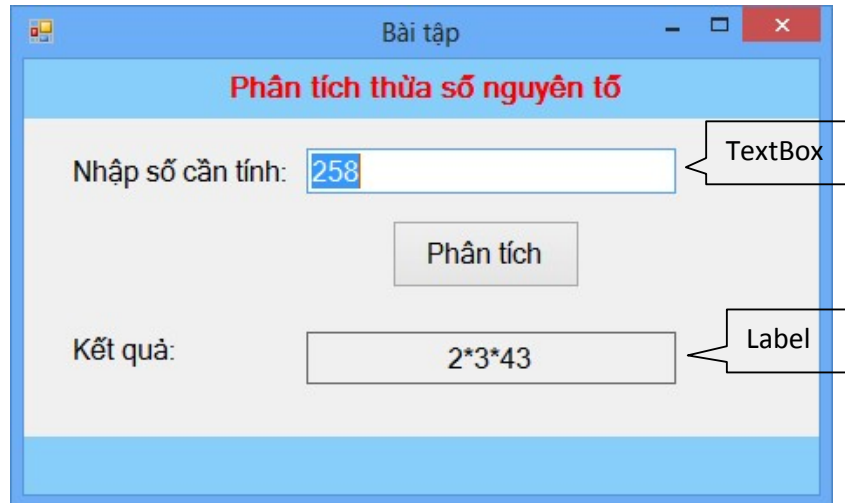
- Button “Giải” sẽ hiển thị kết quả của phương trình tại Label “-----”. Sau khi hiển thị kết quả các TextBox sẽ được thiết lập rỗng để người dùng có thể nhập hệ số của phương trình mới.
- Button “Thoát” sẽ thoát chương trình.

Câu 4: Viết chương trình tìm kiếm số chính phương trong một dãy số nhập trong TextBox (các số cách nhau bởi khoảng trắng). Giao diện gồm các điều khiển: TextBox, Label, Button như hình 3.37.

Hình 3.37: Giao diện chương trình tìm kiếm số chính phương

- Button “Tìm” sẽ hiển thị các số chính phương có trong dãy số đã nhập trong TextBox.
- Button “Thoát” sẽ thoát chương trình.

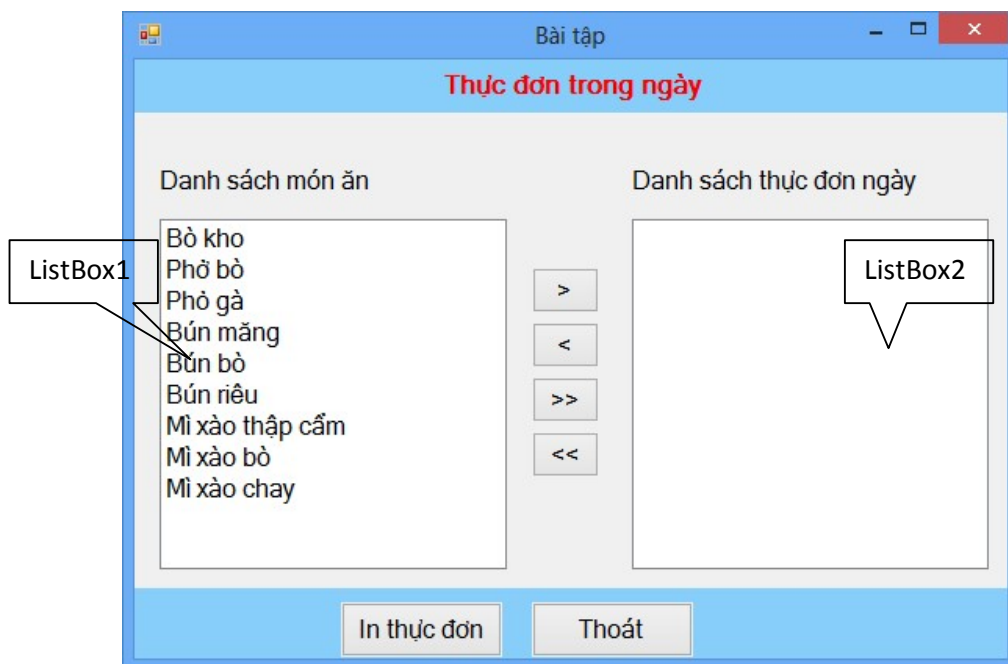
Câu 5: Viết chương trình phân tích một số thành các thừa số nguyên tố. Giao diện gồm các điều khiển: TextBox, Label, Button như hình 3.38.



Hình 3.38: Giao diện chương trình phân tích thừa số nguyên tố

- Button “Phân tích” sẽ phân tích số người dùng trong TextBox thành thừa số nguyên tố hiển thị trong Label.

Câu 6: Viết chương cho phép người dùng lựa chọn các món ăn để làm thành thực đơn các món ăn trong ngày mà cửa hàng thức ăn có bán. Giao diện gồm các điều khiển: ListBox, Label, Button như hình 3.39.



Hình 3.39: Giao diện chương trình chọn thực đơn thức ăn trong ngày

- ListBox1: Hiển thị danh sách tất cả các món ăn
- Button “>”: chuyển một món được chọn từ ListBox1 qua ListBox2
- Button “>>”: chuyển tất cả các món từ ListBox1 qua ListBox2
- Button “<”: xóa món ăn được chọn trong ListBox2
- Button “<<”: xóa tất cả món ăn trong ListBox2
- Button “In thực đơn” sẽ hiển thị hộp thoại MessageBox với nội dung là các món ăn đã chọn hiển thị trong ListBox2
- Button “Thoát” : đóng chương trình.

Câu 7: Viết chương trình tạo form đăng nhập có giao diện như hình 3.40.

Hình 3.40: Giao diện đăng nhập người dùng

Yêu cầu:

Người dùng nhập tên tài khoản và mật khẩu trên TextBox txtTen và txtMatkhaiu.

- Nếu CheckBox chkHienThi không được chọn thì ở TextBox txtMatkhaiu sẽ hiện dấu * với mỗi ký tự người dùng gõ.
- Nếu CheckBox chkHienThi được chọn thì TextBox txtMatkhaiu sẽ hiển thị đúng các ký tự người dùng đã gõ.

Sau khi nhập xong tên đăng nhập và mật khẩu. Người dùng nhấn Button “Đăng nhập”:

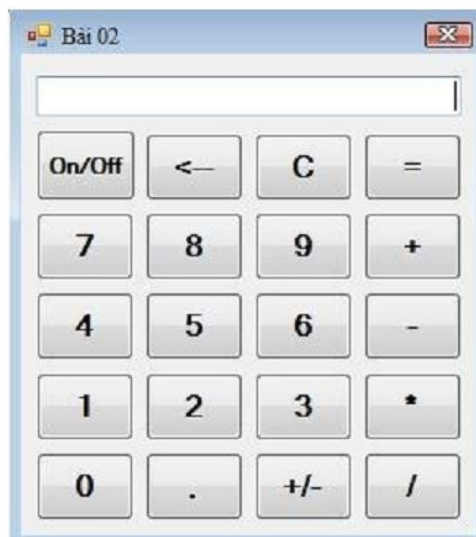
- Nếu tên tài đăng nhập là “admin” và mật khẩu là “123456” thì xuất hiện MessageBox với nội dung: “Đăng nhập thành công”.
- Nếu tên tài khoản và mật khẩu không phải là “admin” và “123456” thì xuất hiện MessageBox với nội dung: “Không đăng nhập thành công”.

Người dùng nhấn Button “Thoát” để đóng chương trình

Câu 8: Viết chương trình tạo Calculator như hình 3.41.

Yêu cầu:

- Khi nhấn F5 để bắt đầu thực thi chương trình, hiển thị messageBox chứa thông tin: Họ tên sinh viên – mã số sinh viên. Khi nhấn chọn Button “OK” trên messageBox thì hiển thị form Máy tính. Khi nhấn chọn Button “Cancel” trên messageBox thì đóng chương trình.
- Xử lý Button “ON/OFF”
 - o Khi chưa nhấn chọn Button “ON/OFF” (hoặc số lần nhấn chọn là chẵn) thì tắt máy, nghĩa là, không cho tác động lên các thành phần còn lại trên form Máy tính.
 - o Nhấn chọn một lần Button “ON/OFF” (hoặc số lần nhấn chọn là lẻ) thì mở máy, nghĩa là cho tác động lên các thành phần còn lại trên form Máy tính, đồng thời xuất hiện cursor nhấp nháy ở lề phải khung hiển thị sẵn sàng làm toán.
- Xử lý khung hiển thị
 - o Không hiển thị số được gõ từ bàn phím của máy tính mà chỉ hiển thị số khi nhấn chọn vào các Button số trên form Máy tính. Nhấn chọn Button nào thì hiển thị số tương ứng với nhãn của Button đó, trong đó, Button “.” để nhập dấu cách phần thập phân của số thực A và B; nút “-” để gõ dấu âm đứng trước số thực A và B.
- Xử lý tính toán + , - , x , /
 - o Thực hiện phép toán giữa 2 số thực A và B. Số A là số nhập trước khi nhấn chọn Button phép toán, nếu khung hiển thị đang để trống thì hiểu số A có giá trị 0. Số B là số nhập sau khi nhấn chọn Button phép toán.
- nhấn chọn Button “=” để xuất kết quả phép toán lên khung hiển thị.
- Mỗi lần nhấn chọn Button “←” để xóa một kí số, xóa từ bên phải sang.
- nhấn chọn Button “C” để xóa tất cả trong khung hiển thị.



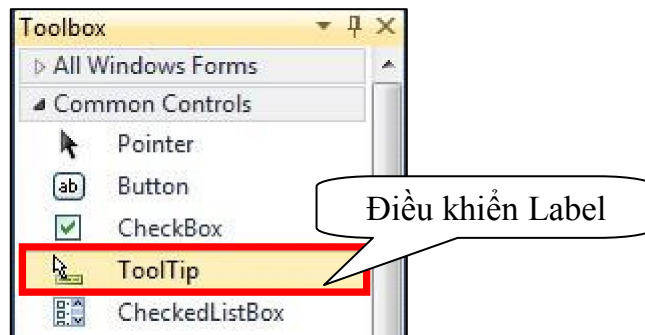
Hình 3.41: Giao diện Calculator

CHƯƠNG 4: CÁC ĐIỀU KHIỂN ĐẶC BIỆT

4.1. Điều khiển Tooltip, HelpProvider, ErrorProvider

4.1.1. Điều khiển Tooltip

Điều khiển *Tooltip* là điều khiển cho phép hiển thị các thông tin chú thích khi người dùng đưa chuột qua các điều khiển có thiết lập *Tooltip*. Điều khiển *Tooltip* được đặt trong nhóm Common Controls của cửa sổ Toolbox như hình 4.1.



Hình 4.1: Điều khiển Tooltip

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

Bảng 4.1: Bảng mô tả các thuộc tính của *Tooltip*

Thuộc tính	Mô tả
Active	Mang giá trị True hoặc False, nếu thiết lập True thì Tooltip có hiệu lực hiển thị thông báo, nếu mang giá trị False thì Tooltip không hiển thị được thông báo.
AutomaticDelay	Thiết lập thời gian xuất hiện Tooltip khi vừa đưa chuột đến điều khiển, thời gian tính bằng mili giây
AutoPopDelay	Thời gian hiển thị Tooltip cho đến khi kết thúc khi người dùng đã đưa chuột đến điều khiển, thời gian tính bằng mili giây
IsBalloon	Quy định kiểu hiển thị của Tooltip. Nếu thiết lập False kiểu hiển thị Tooltip: