

Chương 2. CÁC CONTROL CƠ BẢN VÀ SỰ KIỆN

Mục tiêu:

Sau khi học xong chương này, sinh viên có thể:

- + Hiểu các thành phần chính cấu trúc nên ứng dụng như: điểm vào chương trình, Class Application, phần mã “ẩn” do Forms Designer sản sinh, phần mã “hiện” người lập trình viết bổ sung vào.
- + Hiểu vai trò của Forms Designer trong việc sinh mã cho các Form được thiết kế thông qua “kéo và thả”.
- + Hiểu cách dùng Class Form, các thuộc tính và sự kiện thông thường của nó.
- + Biết cách thiết lập, sử dụng sử dụng các lệnh: lấy dữ liệu, thêm, xóa, cập nhật dữ liệu cho từng Control: TextBox, Lable, Button, ListBox, ComboBox, ListView, TreeView, NumericUpdown, CheckedListBox,...
- + Biết các thuộc tính và sử dụng các lệnh Message Box.
- + Biết thiết lập phím tắt, thứ tự Tab, Focus.
- + Biết nguyên tắc khi thiết kế Form và các Control trên Form.



1. Forms Designer và cấu trúc ứng dụng

1.1. Dùng Forms Designer của Visual Studio.NET

Tạo khuôn (template) ứng dụng Windows: khởi động Microsoft Visual C#. Vào menu File → New → Project... → **chọn template Windows Application** → Nhập Name, nhập Location VS.NET tự động tạo sẵn:

- Mã lệnh của bộ khung chương trình trong các file *.cs
- Form chính của chương trình.

Thiết kế Form theo yêu cầu của ứng dụng: sửa đổi thuộc tính của Form như hình thức → Font → biểu tượng ...

Chọn Form (ví dụ tab Form1.cs [design]). Menu View → **Properties Windows** → Nhấp nút Categorized () hay nút Alphabetical () → Chọn thuộc tính muốn đổi và gán giá trị mới.

Ví dụ 1: để đổi tiêu đề của Form và đổi màu nền của Form, nhấp chuột phải Form → Properties, rồi:

- + Chọn thuộc tính Text, gõ vào “Hello World” (trong textbox ngay bên phải).
- + Chọn thuộc tính BackColor, nhấp xô xuống mũi tên bên phải, chọn Custom, chọn một màu tùy ý.

Đặt thêm các điều khiển (Control) lên Form, sửa đổi thuộc tính của điều khiển:

- Hiện thị **Toolbox**: Menu View → Toolbox.

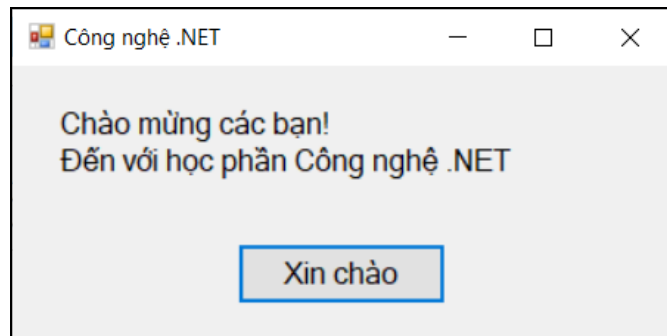
- Để đặt điều khiển lên Form: kéo Control (chọn theo yêu cầu thiết kế) từ Toolbox thả lên Form.

- Để sửa đổi thuộc tính của điều khiển: nhấp chọn điều khiển, nhấp phải, chọn Properties để hiện Properties Windows của điều khiển.

Ví dụ 2: thêm vào Form1 trên hai điều khiển như hình:

+ Thêm 1 Label (lbXinchao) rồi sửa thuộc tính Text: Chào mừng các bạn! Đến với học phần Công nghệ .NET

+ Thêm 1 Button rồi sửa thuộc tính như sau: (Name): btXinchao, Text: Xin chào.



Viết mã xử lý từng sự kiện (event) phát sinh trên điều khiển hay trên Form: Khi người dùng thao tác trên điều khiển hay trên Form (bằng chuột hay bàn phím) sẽ phát sinh sự kiện tương ứng với thao tác đó. Chương trình sẽ đáp ứng với sự kiện bằng phương thức xử lý sự kiện (event handler method) chứa các câu lệnh thực hiện xử lý sự kiện theo yêu cầu ứng dụng.

Ví dụ 3: khi người dùng nhấp lên nút “Xin chào” sẽ phát sinh sự kiện Click của điều khiển tên btXinchao. Với yêu cầu thiết kế khi nút “Xin chào” được nhấp, ứng dụng sẽ hiển thị Label (lbXinchao), viết phương thức xử lý sự kiện này như sau: nhấp kép lên điều khiển btXinchao trong tab Form1.cs [design], một cửa sổ code sẽ mở ra (Form1.cs) với phương thức xử lý sự kiện được tạo sẵn dưới dạng hàm rỗng, gõ vào trong hàm rỗng câu lệnh sau (chữ đậm):

```
public partial Class Form1 : Form
{
    // ...
    private void btXinchao_Click(object sender, EventArgs e)
    {
        lbXinchao.Visible = true;
    }
}
```

1.2. Sự kiện – hàm xử lý sự kiện

Sự kiện là kỹ thuật cơ bản để thông báo qua lại giữa các thành phần khác nhau. Khi nhấp nút trái chuột lên một Control, Control đó sẽ phát sinh ra sự kiện Click và chuyển cho ứng dụng xử lý. Các sự kiện này cho phép ứng dụng

đáp ứng lại các tác động được sinh ra bởi người dùng hay hệ thống. Các Control phát sinh ra các sự kiện để thông báo cho ứng dụng biết có một vài tác động đã xảy ra và yêu cầu ứng dụng giải quyết.

Chương trình Windows thực thi bằng sự xuất hiện một cửa sổ chứa các đối tượng đồ họa như Menu, Toolbar, các điều khiển (Textbox, Listbox, Button ...). Chương trình tiếp tục chạy tùy theo sự dẫn dắt của người dùng thông qua thao tác trên các đối tượng đồ họa này cho đến khi kết thúc (khi người dùng nhấp nút Close hay Exit).

Mỗi thao tác của người dùng lên các đối tượng đồ họa như nhấp chuột lên Button, gõ phím vào Textbox ... sẽ sản sinh một sự kiện. Tùy theo ý nghĩa và nội dung của sự kiện, chương trình sẽ có đáp ứng tương ứng bằng một ***hàm xử lý sự kiện (Event handler)***.

Ngoài ra sự kiện cũng có thể phát sinh từ bản thân chương trình như khi một timer (bộ định thời gian) báo một khoảng thời gian đã trôi qua, khi một file đã được đọc vào bộ nhớ...

1.2.1. Publish and Subscribe

Trong .NET, các điều khiển xuất bản (Publish) một tập các sự kiện mà các Class khác có thể đặt hàng trước (Subscribe). Khi Class xuất bản phát sinh sự kiện (mỗi điều khiển ứng với một Class), tất cả các Class đặt hàng được thông báo.

Thiết lập mối quan hệ publish-subscribe là xác định mối phụ thuộc một-nhiều giữa các đối tượng để khi đối tượng thay đổi trạng thái, các đối tượng phụ thuộc nó được thông báo và tự động cập nhật.

Ví dụ: một Button có thể thông báo các Class quan tâm đến nó khi nó được người dùng nhấp lên (click). Button được gọi là Class xuất bản (Publishing Class) vì nó xuất bản sự kiện Click. Các Class quan tâm được gọi là Class đặt hàng (Subscribing Class) vì chúng nhận thông báo về sự kiện Click.

1.2.2. Delegate

Để cung cấp một hệ thống các sự kiện mạnh mẽ hơn, .NET giới thiệu khái niệm về Delegate. Trong C/C++ thì Delegate giống như con trỏ hàm. Trong VB, Delegate giống như truyền địa chỉ của một hàm tới một hàm khác có sử dụng toán tử AddressOf. Với sự kiện được cài đặt với Delegate, Class xuất bản định nghĩa một Delegate đóng gói một phương thức mà Class đặt hàng sẽ cài đặt. Khi sự kiện phát sinh, phương thức của lớp đặt hàng (phương thức xử lý sự kiện) được kích hoạt thông qua Delegate.

1.3. Xử lý sự kiện

Xử lý sự kiện trong ứng dụng được tạo với text editor: xử lý sự kiện gồm hai bước: cài đặt phương thức xử lý sự kiện và Liên kết (đăng ký) phương thức xử lý sự kiện với sự kiện.

1.3.1. Cài đặt phương thức xử lý sự kiện

Mẫu chung của phương thức xử lý sự kiện như sau:

```
private void EventHandler (object sender, EventArgs e)
{
    // các câu lệnh thực hiện công việc đáp ứng với sự kiện phát sinh.
}
```

Theo qui ước, hàm trả về void và tên của đối số thứ nhất là “sender”, tên của đối số thứ hai là “e”.

Đối số thứ nhất là nguồn của sự kiện, tức đối tượng xuất bản sự kiện. Đối số thứ hai là một đối tượng từ Class EventArgs trong đó EventArgs là lớp cơ sở cho tất cả dữ liệu của sự kiện.

- Nếu dữ liệu cần quan tâm, như vị trí nhấp chuột, phím gõ vào,... đối số thứ hai là một đối tượng dẫn xuất từ Class EventArgs với các properties để dữ liệu được truyền.

- Ngược lại nếu dữ liệu không có ý nghĩa, đối số thứ hai là một đối tượng từ chính Class EventArgs và không có các Properties, chỉ đóng vai giữ chỗ mà không truyền dữ liệu gì.

Tên phương thức (tạm đặt ở trên là EventHandler) có thể đặt tùy ý. Tuy nhiên tên này thường đặt theo cách mặc định là tên bao gồm tên nhận dạng của điều khiển và tên sự kiện nối nhau bằng dấu _

Phương thức là một thành viên của Class của Form chứa điều khiển. Ví dụ trong chương trình HW_win1, phương thức là thành viên của Class HelloWorld.

```
private void btn_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

```
}
```

1.3.2. Liên kết phương thức xử lý sự kiện với sự kiện

Việc liên kết này nhằm móc nối **tên phương thức xử lý sự kiện** với **tên sự kiện** và thực hiện bằng cách tạo một thể hiện của delegate EventHandler để đóng gói tên phương thức xử lý sự kiện, rồi dùng toán tử += để Add thể hiện đó vào Properties của điều khiển ứng với tên sự kiện phát sinh.

Câu lệnh liên kết được đặt trong phương thức tạo của Form chứa điều khiển, ví dụ trong chương trình HW_win1, câu lệnh được đặt trong hàm tạo của Class HelloWorld

```
btn.Click += new System.EventHandler(btn_Click);
```

Xử lý sự kiện trong ứng dụng được tạo dùng Forms Designer

1.3.3. Xử lý sự kiện mặc định

Mỗi điều khiển có một sự kiện mặc định là sự kiện thường phát sinh nhất của nó. Ví dụ với button thì sự kiện mặc định của nó là Click.

Trong Form Designer, để xử lý sự kiện mặc định của điều khiển, nhấp kép lên điều khiển trong cửa sổ Design. (Để di chuyển đến cửa sổ Form1.cs [Design]: trong cây thư mục cửa sổ Solution Explorer, nhấp kép lên Form1.cs, ví dụ Form1 chứa điều khiển đang xét). Khi đó Visual Studio.NET hiểu rằng người lập trình muốn cài đặt hàm xử lý sự kiện và tự động thực hiện các việc sau:

- Tạo phương thức xử lý sự kiện (phương thức rỗng, tức chưa có code trong thân phương thức). Tên phương thức theo mặc định.


- Liên kết phương thức xử lý sự kiện. Câu lệnh liên kết là mã “ẩn” được chứa trong phương thức InitializeComponent, đặt trong vùng *Windows Form Designer generated code (Mã lệnh sinh ra bởi Bộ thiết kế Windows Form)*, trong cửa sổ Form1.Designer.cs (Để xem mã ẩn tham chiếu đến phương thức, đặt điểm chèn giữa tên phương thức, nhấp phải, chọn Find All References; xem định nghĩa phương thức: Go to Definition).

- Di chuyển đến cửa sổ code và đặt con trỏ trong thân phương thức xử lý sự kiện rỗng.


Tới đây, nhiệm vụ người lập trình là gõ code vào thân phương thức để thực hiện hành động nào đó nhằm đáp ứng sự kiện.

1.3.4. Xử lý các sự kiện khác

Để xử lý các sự kiện khác, dùng danh sách Event trong cửa sổ Properties của điều khiển.

Trong cửa sổ Design, chọn điều khiển, vào menu View → Properties window (hay nhấp phải điều khiển, chọn Properties) → nhấp nút Events .

Trong danh sách các sự kiện của điều khiển, di chuyển đến ô bên phải của tên sự kiện muốn xử lý.

- Nếu ô còn trống, tức chưa định nghĩa phương thức xử lý cho sự kiện, nhấp kép lên ô (hay đặt điểm chèn trong ô và nhấn phím Enter), Visual Studio.NET sẽ tự động tạo liên kết và di chuyển đến thân phương thức xử lý như trường hợp xử lý sự kiện mặc định đã trình bày trên. Cũng có thể đặt điểm chèn trong ô và nhấp nút  bên phải ô để chọn một phương thức xử lý đã có sẵn trong danh sách xổ xuống. Như vậy có thể gán cùng phương thức để xử lý cho nhiều sự kiện khác nhau của cùng một điều khiển hay của các điều khiển khác nhau.

- Nếu ô có tên, tức đã định nghĩa phương thức xử lý sự kiện, nhấp kép lên ô sẽ di chuyển đến thân phương thức xử lý. Có thể xóa phương thức xử lý cho sự kiện bằng cách xóa tên trong ô rồi nhấn phím Enter, Visual Studio.NET sẽ tự động xóa câu lệnh liên kết phương thức (có thể chỉ xóa gán phương thức cho sự kiện có tên đang xét).

2. Quan hệ giữa các Control

Windows Form định nghĩa 2 kiểu quan hệ giữa các Control:

- Quan hệ cha con (parent-child) hay quan hệ chứa (containment) qui định Control chứa (Form, Panel, Group box, ...) và Control bị chứa (nằm trong Control chứa).

- Quan hệ sở hữu (ownership) giữa các Control (cửa sổ) cùng mức đỉnh (top-level). Control mức đỉnh là Control không có cha, ví dụ các Form. Quan hệ này lỏng lẻo hơn quan hệ cha-con.

2.1. Quan hệ cha con

Một Control con là một Control được hiển thị hoàn toàn trong cửa sổ cha của nó. Ví dụ mọi Control đặt trên Form là con của Form đó. Vị trí của con được qui định tương đối đối với cha nó.. Form cũng có thể là con, ví dụ các cửa sổ tài liệu trong một ứng dụng MDI (multi document interface) là con của cửa sổ chính của MDI. Một Control có thể vừa là cha vừa là con, ví dụ một group box là con của Form chứa nó, đồng thời là cha của các Control trong nó.

Quan hệ cha-con giữa 2 Control có thể thiết lập bằng một trong hai cách: Vd để gắn đk btn lên Form:

- Dùng thuộc tính (property) **Parent** của con: gán Property này đến tên cha: `btn.Parent = this;`

- Dùng thuộc tính **Controls** của cha: Controls là một Collection mà có các phương thức **Add()** và **AddRange()** để thêm các Control con vào Control cha. Đây là cách Form Designer dùng, xem minh họa trong phương thức **InitializeComponent()**: `Controls.Add(btn);`

Một Control có thể không có cha, khi đó thuộc tính Parent của nó là null. Các Control này được gọi là cửa sổ mức đỉnh (top-level windows), ví dụ như các Form. Mỗi cửa sổ mức đỉnh được chứa trực tiếp bởi desktop và luôn luôn có một biểu tượng trên Taskbar.

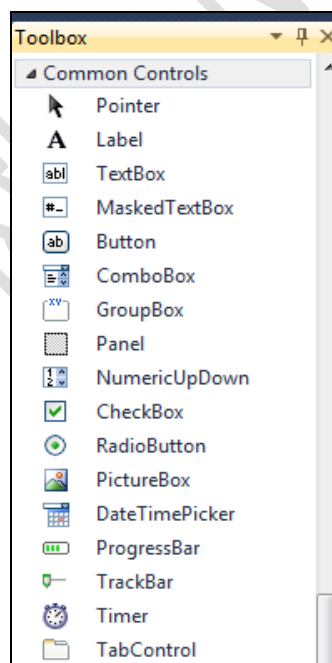
2.2. Quan hệ sở hữu (ownership)

Quan hệ sở hữu (ownership): chỉ tồn tại giữa các Form mức đỉnh, dùng để gom nhóm các Form có liên hệ với nhau trong các tác vụ như minimize, activation, ... và không có ý nghĩa “chứa” như quan hệ cha con. Ví dụ khi Form sở hữu (owner Form) minimize thì các Form bị sở hữu (owned Form) cũng minimize theo.

Thứ tự Z (Z-order) của các Form: khi có nhiều Form thì Form nào được hiển thị chồng trên Form khác được qui định bởi thứ tự Z, là giá trị dùng để xác định vị trí của Form theo chiều thứ ba (Form trên che phủ Form dưới), trong khi X và Y xác định vị trí Form trên mặt phẳng màn hình.

3. Một số thuộc tính cơ bản của các Control

Control là phương tiện để người dùng tương tác với ứng dụng. Control gồm các điều khiển được thiết kế trên Form như: Label, TextBox, Button, RadioButton, CheckBox,... Mỗi Control có các thuộc tính, phương thức và sự kiện với ý nghĩa sử dụng khác nhau.



Hình 2.1. Minh họa các điều khiển trên Toolbox

Một số thuộc tính chung của các Control:

Bảng 3.1. Mô tả ý nghĩa các thuộc tính chung của các Control

Thuộc tính	Ý nghĩa
Name	Tên của đối tượng (không khoảng trắng, không dấu tiếng Việt,

Thuộc tính	Ý nghĩa
	không ký tự đặc biệt).
Text	Nội dung thể hiện của đối tượng tương ứng trên Form.
ReadOnly	Thuộc tính chỉ đọc, không thể hiệu chỉnh được.
Visible	Trạng thái ẩn/hiện của đối tượng.
ForceColor	Màu chữ của đối tượng.
BackColor	Màu nền của đối tượng.
AutoSize	Thuộc tính tự định dạng kích thước cho đối tượng.
Dock	Trạng thái canh vị trí đối tượng trên Form.
TextAlign	Chế độ canh cho nội dung thể hiện.
Locked	Dùng để khóa vị trí của đối tượng trên Form trong quá trình Design.
Enabled	Trạng thái khóa hay không khóa của đối tượng.
Font	Lựa chọn Font chữ cho đối tượng.

Có rất nhiều Control được cung cấp trong Visual Studio, trong phạm vi tập bài giảng này, chỉ giới thiệu một số Control thường hay được sử dụng nhất.

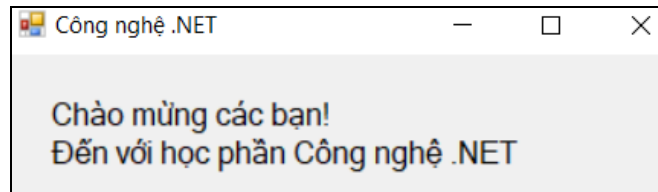
3.1. Label: dùng để hiển thị những thông tin mang tính mô tả.

Thuộc tính thường sử dụng:

Bảng 3.2. Mô tả các thuộc tính thường gặp của Label

Thuộc tính	Ý nghĩa
AutoSize	Cho phép tự động thay đổi kích thước của Label để vừa với nội dung.
BoderStyle	Xác định hình dáng đường viền của Label.
Font	Tên Font, kiểu và kích thước văn bản được hiển thị.
Location	Vị trí của Label trên Form tương ứng với góc trên bên trái của Form.
Name	Tên sử dụng để xác định Label.
Size	Chiều cao và chiều rộng của Label.
Text	Văn bản được hiển thị trên Label.
TextAlign	Chỉ ra cách văn bản được canh chỉnh trong phạm vi Label.

Ví dụ 4: khi chương trình thực thi sẽ hiển thị chuỗi như hình bên dưới lên Label (lbXinchao):



Viết code trong sự kiện Form1_Load như sau:

```
private void Form1_Load(object sender, EventArgs e)
{
    lbXinchao.Text = "Chào mừng các bạn! \nĐến với học phần Công nghệ .NET";
}
```

3.2. Textbox: dùng để nhập dữ liệu hoặc hiển thị kết quả tính toán khi thực thi chương trình.

Thuộc tính thường sử dụng:

Bảng 3.3. Mô tả ý nghĩa các thuộc tính thường gặp của TextBox.

Thuộc tính	Ý nghĩa
Enabled	Cho phép sáng/mờ textbox (True/False)
Multiline	Cho phép textbox có nhiều dòng
PasswordChar	Ký tự thay thế khi nhập giá trị vào TextBox
BackColor	Định màu nền cho textbox
MaxLength	Quy định chiều dài cực đại chuỗi sẽ nhập vào TextBox, mặc định là 32767 byte
Wordwrap	Cho phép văn bản dài tự động xuống dòng (True, False)
ReadOnly	Thuộc tính chỉ đọc
CharacterCasing	Normal (mặc định), Upper (tự động đổi thành chữ hoa), Lower (tự động đổi thành chữ thường)
Scrollbars	None (mặc định), Horizontal (thanh cuộn ngang), Vertical (thanh cuộn dọc)
BorderStyle	Xác định kiểu đường viền bao quanh TextBox
TextAlign	Vị trí hiển thị chuỗi trên TextBox
Font, ForeColor	Quy định Font và màu chữ

Phương thức thường sử dụng:

Bảng 3.4. Mô tả ý nghĩa các phương thức thường sử dụng của TextBox.

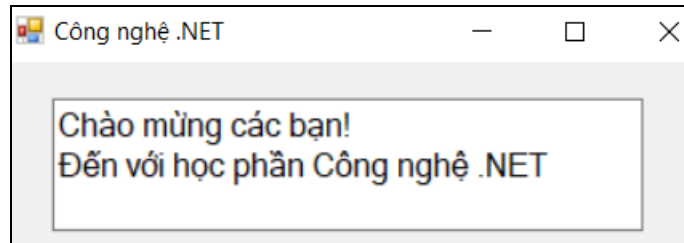
Phương thức	Ý nghĩa
Location	Di chuyển khung đến tọa độ X, Y.
Focus	Con trỏ hiện hành tại TextBox.

Sự kiện thường sử dụng:

Bảng 3.5. Mô tả ý nghĩa các sự kiện thường sử dụng của TextBox.

Sự kiện	Ý nghĩa
TextChanged	Xảy ra khi dữ liệu trên TextBox thay đổi
Validated	Kiểm tra dữ liệu có thỏa điều kiện nào đó khi kết thúc nhập dữ liệu.

Ví dụ 5: thiết kế Form khi chương trình thực thi sẽ hiển thị chuỗi như hình ở bên dưới lên Textbox (txtXinchao).



Viết code trong sự kiện Form1_Load như sau:

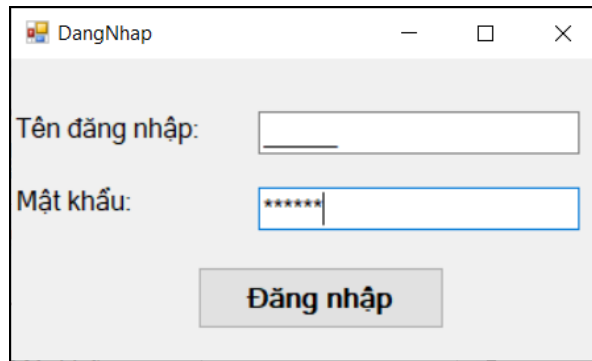
```
private void Form1_Load(object sender, EventArgs e)
{
    txtXinchao.Text="Chào mừng các bạn!\r\nĐến với học phần Công nghệ .NET";
}
```

3.3.MaskedTextBox: giống TextBox nhưng có thuộc tính **Mask** dùng định dạng dữ liệu nhập vào.

Bảng 3.6. Mô tả ý nghĩa định dạng dữ liệu nhập của MaskedTextBox.

Ký tự	Ý nghĩa
0	Chỉ được nhập vào số, nếu không nhập vào sẽ nhận khoảng trắng.
9	Nhập số và khoảng trắng, không thêm khoảng trắng nếu không nhập.
#	Nhập số, khoảng trắng, ký tự +, -, nếu không nhập trả về khoảng trắng.
L	Nhập vào ký tự a→z, A→Z, nếu không nhập vào sẽ nhận khoảng trắng.
?	Nhập vào ký tự a→z, A→Z không thêm khoảng trắng nếu không nhập.
>	Chuyển các ký tự thành chữ hoa.
<	Chuyển các ký tự thành chữ thường.
C	Nhập ký tự bất kỳ.

Ví dụ 6: thiết kế Form như hình bên dưới, với MaskedTextBox tên đăng nhập cho phép nhập 5 ký tự là chuỗi, MaskedTextBox mật khẩu hiển thị ký hiệu * khi người dùng nhập mật khẩu vào: thuộc tính Mask của tên đăng nhập nhập vào: ?????, thuộc tính PasswordChar của mật khẩu nhập vào: *



3.4. Button: là điều khiển cho phép người dùng nhấn để thực hiện các xử lý của chương trình. Sự kiện thường sử dụng với điều khiển này là click.

Thuộc tính thường sử dụng:

Bảng 3.7. Mô tả ý nghĩa các thuộc tính thường sử dụng của Button.

Thuộc tính	Ý nghĩa
Name	Tên của Button
Text	Nội dung văn bản hiển thị trên Button
TextAlign	Canh vị trí hiển thị nội dung trên Button
BackColor	Màu nền cho nút
Enabled	Khóa/mở cho phép sử dụng Button
Image	Chọn Icon cho Button
ImageAlign	Vị trí hiển thị hình nền trên Button
Locked	Khóa/không khóa điều khiển Button
BackgroundImage	Chọn hình nền cho Button
Visible	Có giá trị mặc định là True, dùng để ẩn/hiện Button

Ví dụ 7: Thiết kế Form giống ví dụ 4, khi người dùng nhấp vào nút Đăng nhập nếu chưa nhập đủ thông tin Tên đăng nhập và mật khẩu sẽ xuất hiện thông báo “Vui lòng nhập đầy đủ thông tin!”. Code sự kiện click của nút Đăng nhập như sau:

```
private void btDangnhap_Click(object sender, EventArgs e)
{
    if ((txtMatkhau.Text=="") || (txtTendangnhap.Text==""))
    {
        MessageBox.Show("Vui lòng nhập đầy đủ thông tin!");
    }
}
```

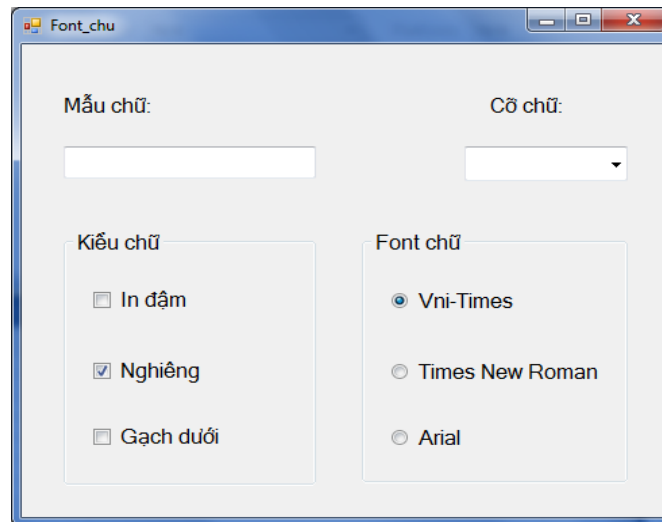
3.5. GroupBox: là điều khiển dùng để chứa một nhóm các điều khiển cùng tính chất, giúp việc bố trí giao diện của Form một cách khoa học và làm nổi bật vai trò của chúng. Có thể hiển thị tiêu đề trên thuộc tính Text.

Thuộc tính thường dùng:

Bảng 3.8. Mô tả ý nghĩa các thuộc tính thường sử dụng của GroupBox.

Thuộc tính	Ý nghĩa
Name	Tên của GroupBox
Text	Tiêu đề của GroupBox

Ví dụ 8: Minh họa GroupBox:



Hình 2.2. Minh họa GroupBox

3.6. Panel: là điều khiển chứa nhóm các Control.

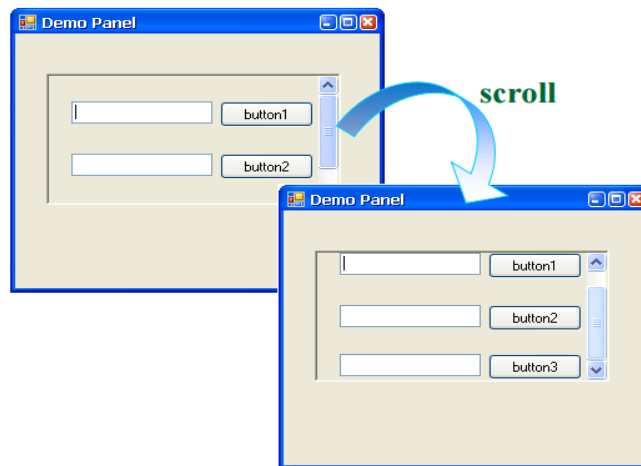
Không có tiêu đề.

Có thanh cuộn (ScrollBars): xem nhiều Control khi kích thước Panel giới hạn.

Bảng 3.9. Mô tả ý nghĩa các thuộc tính thường sử dụng của Panel.

Thuộc tính	Ý nghĩa
Name	Tên của Panel
AutoScroll	Xuất hiện khi Panel quá nhỏ để hiển thị hết các Control, mặc định là False
BorderStyle	Biên của Panel, mặc định là None, các tham số khác như: Fixed3D, FixedSingle

Ví dụ 9: Minh họa Panel:



Hình 2.3. Minh họa Panel.

3.7. NumericUpDown: là điều khiển chứa các số nguyên có giá trị từ nhỏ đến lớn, người sử dụng chỉ có thể chọn giá trị trong điều khiển này.

Thuộc tính thường sử dụng:

Bảng 3.10. Mô tả ý nghĩa các thuộc tính thường sử dụng của NumericUpDown.

Thuộc tính	Ý nghĩa
Minimum	Giá trị nhỏ nhất
Maximum	Giá trị lớn nhất
Value	Giá trị chọn trên điều khiển

Ví dụ 10: Thiết kế Form tính chu vi hình chữ nhật như hình bên dưới, với điều kiện giá trị chiều dài và chiều rộng là số nguyên dương ≤ 100 , giá trị mặc định là 1.

→ Thiết kế điều khiển NumericUpDown cho Chiều dài và chiều rộng với các thuộc tính: Minimum: 1, Maximum: 100, Value: 1. Để tính chu vi, code sự kiện click của nút Tính:

```
private void btTinh_Click(object sender, EventArgs e)
{
    txtChuvi.Text
    =((nuChieudai.Value+nuChieurong.Value)*2).ToString();
}
```

}

3.8. **CheckBox, RadioButton**: là các điều khiển chỉ có 2 trạng thái chọn hoặc không chọn.

CheckBox: có thể chọn đồng thời nhiều checkbox trong 1 groupbox, panel

RadioButton: chỉ chọn một radio button trong cùng cùng 1 groupbox, panel

Thuộc tính thường sử dụng:

Checked: cho biết điều khiển có được chọn hay không, True: được chọn.

Sự kiện thường sử dụng:

CheckedChanged: xảy ra khi điều khiển thay đổi trạng thái.

Ví dụ 11: Xem ví dụ 6.

3.9. **PictureBox**: là điều khiển dùng để hiển thị hình ảnh trên Form

Thuộc tính thường sử dụng:

Bảng 3.11. Mô tả ý nghĩa các thuộc tính thường sử dụng của PictureBox.

Thuộc tính	Ý nghĩa
Image	Hình hiển thị trên điều khiển, có thể chọn hình trong khi thiết kế bằng cách click vào nút ... trên thuộc tính Image → chọn đường dẫn
SizeMode	Định cách hiển thị của hình ảnh trong khung, nhập một trong các giá trị sau: - Zoom: hình luôn vừa khung. - CenterImage: luôn hiển thị hình giữa khung. - StretchImage: kéo hình cho vừa khung.
Dock	Định vị trí của điều khiển trên Form, nhận giá trị: - Fill: hình luôn chiếm tất cả phần trống của Form - Top: chiếm phần trên.

Để hiển thị hình ảnh: **Ten_PictureBox.Load**("tên đường dẫn chứa hình")

Ví dụ 12: người dùng nhập vào đường dẫn chứa file ảnh và nhấn nút **Hiển thị**, hình ảnh sẽ hiển thị trên PictureBox.



Hình 2.4. Minh họa PictureBox.

Code trong sự kiện click của nút Hiển thị:

```
picHinhAnh.Load(txtduongdan.Text);
```

3.10. DateTimePicker: là điều khiển hiển thị giá trị kiểu Date.

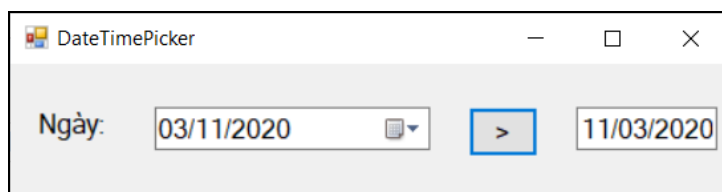
Thuộc tính thường sử dụng:

Bảng 3.12. Mô tả ý nghĩa các thuộc tính thường sử dụng của DateTimePicker.

Thuộc tính	Ý nghĩa
Value	Trả về giá trị được chọn trên điều khiển (đọc, ghi)
CustomFormat	Chuỗi quy định dạng hiển thị dữ liệu trên điều khiển. Ví dụ: muốn hiển thị dữ liệu theo dạng dd/MM/yyyy, trong thuộc tính CustomFormat nhập vào chuỗi dd/MM/yyyy (MM: ghi hoa)
Format	Quy định cách hiển thị dữ liệu, muốn dữ liệu hiển thị theo dạng được định trên CustomFormat, phải chọn giá trị cho thuộc tính này là Custom

Ví dụ 13: Khi người dùng chọn giá trị ngày trên điều khiển DateTimePicker (dtNgay) hiển thị dạng MM/dd/yyyy và nhấn nút > thì giá trị này sẽ hiển thị trên TextBox (txtNgay), code của sự kiện Click nút >:

```
txtNgay.Text = Convert.ToString(dtNgay.Value);
```



3.11. Timer: là điều khiển dùng để tính thời gian.

Thuộc tính	Ý nghĩa
------------	---------

Name	Tên của điều khiển
Interval	Định khoảng thời gian cho sự kiện Tick của điều khiển (1000 = 1s)
Enable	True. Kích hoạt điều khiển Timer, tương đương Timer.Start

Ví dụ 14: Thiết kế Form tạo đồng hồ hẹn giờ như hình bên dưới:

Code xử lý sự kiện Form_Load:

```
int thoigian;
private void Timer_Load(object sender, EventArgs e)
{
    timer_thoigian.Start();
}
```

Code xử lý sự kiện Tick của Timer:

```
private void timer_thoigian_Tick(object sender, EventArgs e)
{
    if(thoigian>0)
    {
        thoigian--;
        lbThoiGianConLai.Text= "Còn lại " + thoigian.ToString() +
        " giây";
    }
}
```

Code xử lý sự kiện Click của nút Bắt đầu:

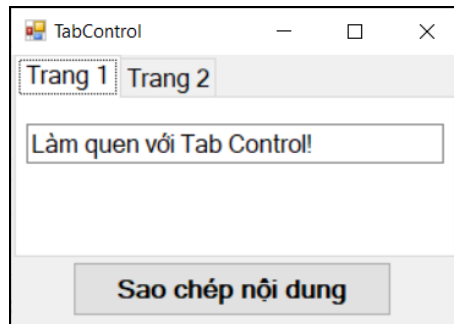
```
private void btBatdau_Click(object sender, EventArgs e)
{
    thoigian = ((int)nuGio.Value) * 3600 + ((int)nuPhut.Value)
    * 60 + ((int)nuGiay.Value);
    lbThoiGianConLai.Text = "Còn lại " + thoigian.ToString() +
    " giây";
}
```

3.12. TabControl

TabControl là điều khiển có nhiều trang, mỗi trang có thể chứa các điều khiển để tiết kiệm không gian trên giao diện của ứng dụng.

Mặc định khi thiết kế trên Form TabControl có 2 trang, muốn thêm trang hoặc xóa trang: nhấn phải vào TabControl → Add/RemoveTab.

Ví dụ 15: Thiết kế Form như hình bên dưới, gồm 2 Tab là: Trang 1 và Trang 2. Tab Trang 1 chứa TextBox với nội dung: “Làm quen với Tab Control!”, khi người dùng nhấp vào nút *Sao chép nội dung* thì nội dung trong Trang 1 sẽ được sao chép qua Trang 2.



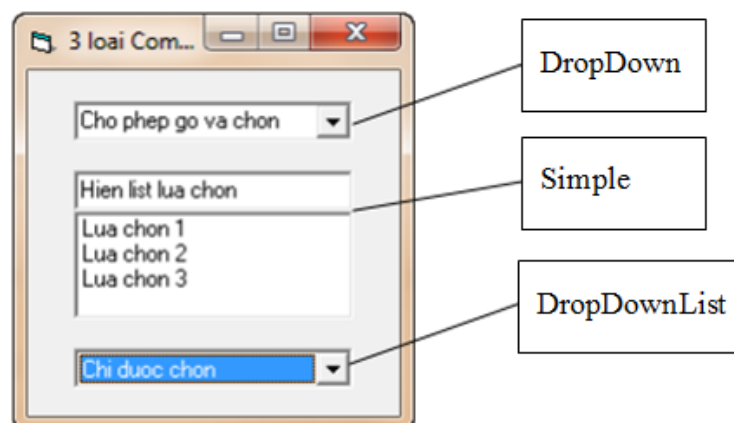
```
private void btSaochep_Click(object sender, EventArgs e)
{
    tabPage2.Controls.Clear();
    tabPage2.Controls.Add(new RichTextBox() {Text =
txtNoidung.Text});
}
```

3.13. ComboBox, ListBox

ComboBox, ListBox: là điều khiển chứa dữ liệu cho phép người dùng chọn lựa.

ListBox: có thể chọn nhiều, không nhập mới.

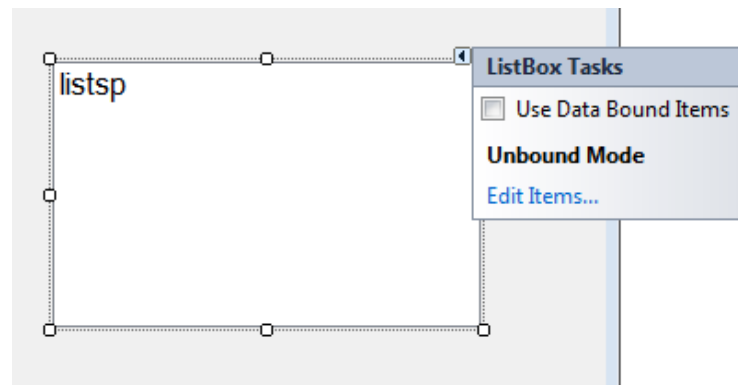
ComboBox: chỉ chọn 1, có thể nhập mới dữ liệu vào. Có 3 dạng hiển thị ComboBox: thuộc tính DropDownStyle quy định cách hiển thị ComboBox.



Hình 6.1. Minh họa các loại ComboBox

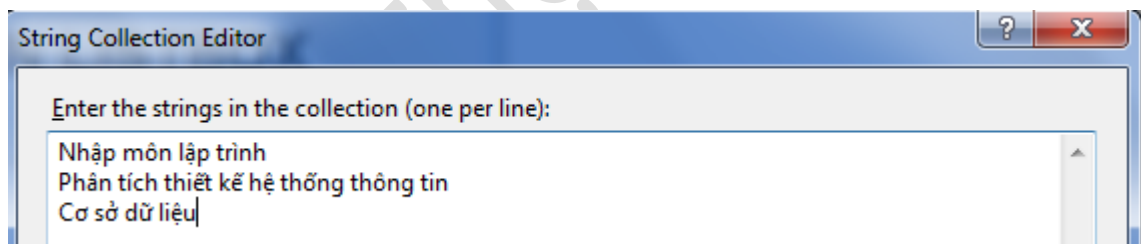
Thuộc tính	Ý nghĩa
Simple	Hiện danh sách các lựa chọn. Lưu ý: phải chỉnh lại chiều cao của ComboBox mới thấy được danh sách lựa chọn.
DropDown	Hiện thị theo dạng ListBox + TextBox, có thể chọn dữ liệu từ ListBox hay nhập mới vào TextBox.
DropDownList	Chỉ được chọn trong các giá trị có sẵn, không thể gõ giá trị khác.

Nhập phần tử vào Listbox lúc Design-time, thuộc tính *Items* cho phép thêm Item vào ListBox.



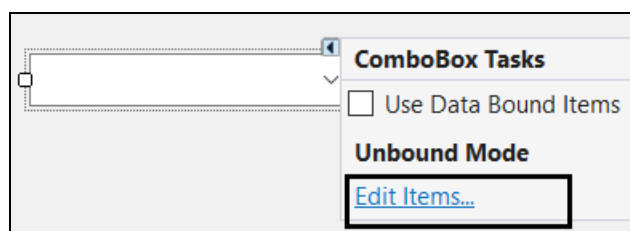
Hình 6.2. Minh họa cách nhập phần tử vào ListBox

Nhấp vào góc trên bên phải của ListBox chọn **Edit Items** hoặc nhấn phải vào ListBox chọn **Items** sẽ xuất hiện hộp thoại cho phép thêm các dòng dữ liệu vào ListBox:



Minh họa cách nhập phần tử vào ListBox (tt)

Nhập phần tử vào ComboBox lúc Design-time, thuộc tính *Edit Items* cho phép thêm Item vào ComboBox.



Code xử lý dữ liệu trên ComboBox và ListBox:

- Đưa dữ liệu vào trong ComboBox, ListBox:

```
Ten_DK.Items.Add("dữ liệu cần thêm");
```

- Chèn dữ liệu vào trong ComboBox, ListBox:

Ten_DK.Item.Insert(vị trí, giá trị);

- Xóa dữ liệu ra khỏi ComboBox, ListBox:

Ten_DK.Items.RemoveAt(chỉ số dòng cần xóa);

Ten_DK.Items.Remove(“giá trị cần xóa”);

Trong trường hợp các đối tượng Item là kiểu chuỗi, truyền vào một chuỗi để xóa. Nếu có nhiều giá trị giống nhau trong ListBox thì mục chọn đầu tiên sẽ bị xóa.

- Xóa toàn bộ dữ liệu trong ComboBox, ListBox:

Ten_DK.Items.Clear();

- Cập nhật dữ liệu trong ComboBox, ListBox:

Ten_DK.Items[chỉ số pt cần cập nhật] = “Giá trị cần cập nhật”;

- Kiểm tra chuỗi đã có trong ComboBox, ListBox:

Ten_DK.Items.Contains(“chuỗi”)=true;

- Số phần tử trong ComboBox, ListBox:

Ten_DK.Items.Count;

- SelectedIndex: trả về vị trí của phần tử được chọn, bắt đầu từ 0
- SelectedValue: trả về giá trị của phần tử được chọn.
- Text: chuỗi được chọn.

Ví dụ 16: Thiết kế giao diện gồm Combobox có các giá trị: Việt Nam, Trung Quốc, Nhật Bản, Hàn Quốc. Khi người dùng nhập nội dung vào TextBox và nhấp nút Thêm nội dung sẽ được thêm vào ComboBox. Khi chọn 1 dòng trên ComboBox nội dung sẽ được hiển thị trên TextBox, nếu người dùng chỉnh sửa nội dung trên Textbox và nhấp nút Cập nhật thì nội dung sẽ được cập nhật vào ComboBox, nếu chọn Xóa thì dòng nội dung sẽ được xóa khỏi ComboBox.

Code xử lý sự kiện Form_Load:

```
private void ComboBox_Load(object sender, EventArgs e)
```

```
{
    cboNoidung.Items.Add("Việt Nam");
    cboNoidung.Items.Add("Nhật Bản");
    cboNoidung.Items.Add("Hàn Quốc");
    cboNoidung.Items.Add("Trung Quốc");
}
```

Code xử lý nút THÊM:

```
private void btThem_Click(object sender, EventArgs e)
{
    cboNoidung.Items.Add(txtNoidung.Text);
}
```

Code xử lý nút XÓA (xóa dòng đang được chọn trên ComboBox):

```
private void btXoa_Click(object sender, EventArgs e)
{
    cboNoidung.Items.RemoveAt(cboNoidung.SelectedIndex);
}
```

Code xử lý sự kiện click của Combobox, lấy giá trị đang được chọn trên Combobox gán vào TextBox:

```
txtNoidung.Text = cboNoidung.SelectedItem.ToString();
```

Code xử lý nút CẬP NHẬT (cập nhật giá trị được chỉnh sửa trên TextBox lưu vào ComboBox):

```
private void btCapnhat_Click(object sender, EventArgs e)
{
    cboNoidung.Items[cboNoidung.SelectedIndex] =
    txtNoidung.Text;
}
```

Code xử lý nút THOÁT:

```
private void btThoat_Click(object sender, EventArgs e)
{
    this.Close();
}
```

Xử lý chọn nhiều phần tử trong ListBox: thuộc tính SelectionMode (Listbox): cho phép chọn nhiều giá trị trên ListBox hay không

Thuộc tính	Ý nghĩa
One	Chỉ chọn một giá trị.
MultiSimple	Cho phép chọn nhiều, chọn bằng cách nhấn vào mục chọn, bỏ chọn nhấn vào mục đã chọn.
MultiExtended	Chọn nhiều bằng cách nhấn nút Shift hoặc Ctrl.

Để lấy giá trị các dòng dữ liệu được chọn trên Listbox, dựa vào các thuộc tính sau:

Thuộc tính	Ý nghĩa
SelectedItems	Chứa các giá trị được chọn.
SelectedIndices	Chứa các chỉ số index của mục chọn.
SelectedItems.Count SelectedIndices.Count	Số dòng đã chọn.

Ví dụ 17: Thiết kế giao diện như hình bên dưới

Yêu cầu:

- Thiết kế Form như trên
- Load danh sách tên các môn học lên ListBox.
- Nhập tên môn học vào TextBox và vị trí sau đó nhấn nút **THÊM** tên môn học sẽ được thêm vào ListBox môn học.
- Khi người sử dụng nhấn chọn các môn học trên ListBox thì danh sách các môn học này sẽ được chọn trên TextBox **Tên MH được chọn**, và danh sách chỉ số các môn học được chọn sẽ được hiển thị trên TextBox **Chỉ số MH được chọn**.
- Nếu nhấn vào nút **XÓA** sẽ xuất hiện hộp thoại xác nhận việc XÓA dữ liệu.

Code xử lý sự kiện Form_Load:

```

1stMonhoc.Items.Add("Công nghệ .NET");
1stMonhoc.Items.Add("Cơ sở dữ liệu");
1stMonhoc.Items.Add("Cấu trúc dữ liệu");
1stMonhoc.Items.Add("Mạng máy tính");
1stMonhoc.Items.Add("Nhập môn lập trình");
1stMonhoc.Items.Add("Tin học văn phòng");

```

Code xử lý sự kiện Click của ListBox chứa tên các môn học (lstMonhoc), lấy nội dung và chỉ số dòng các môn học đã chọn gán vào các TextBox tương ứng (txtMonhocDachon, txtCSdongDachon):

```
private void lstMonhoc_Click(object sender, EventArgs e)
{
    int sodongdachon = lstMonhoc.SelectedItems.Count;
    string s="";
    string cs = "";
    for (int i = 0; i < sodongdachon; i++)
    {
        s += lstMonhoc.SelectedItems[i].ToString() + "\r\n";
        cs += lstMonhoc.SelectedIndices[i]+ " ";
    }
    txtMonhocDaChon.Text = s;//Lấy nội dung các dòng đã chọn
    txtCSdongDaChon.Text = cs;//Lấy chỉ số các dòng đã chọn
}
```

Code xử lý nút THÊM:

```
private void btThem_Click(object sender, EventArgs e)
{
    lstMonhoc.Items.Insert(Convert.ToInt32(nuVitri.Value),
    txtMonhoc.Text);
}
```

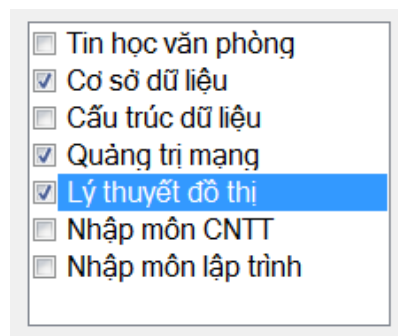
Code xử lý nút XÓA:

```
private void btXoa_Click(object sender, EventArgs e)
{
    int sodongdachon = lstMonhoc.SelectedItems.Count;
    for(int i=sodongdachon-1; i>=0; i--)

    lstMonhoc.Items.RemoveAt(lstMonhoc.SelectedIndices[i]);
}
```

3.14. CheckedListBox

Giống ListBox nhưng bên trái có thêm CheckBox, chọn dữ liệu bằng cách click vào Checkbox này, một thời điểm có thể chọn nhiều giá trị.



Hình 6.3. Điều khiển CheckedListBox

Thuộc tính:

Thuộc tính	Ý nghĩa
CheckedItems	Chứa các giá trị được chọn.
CheckedIndices	Chứa các chỉ số index của các giá trị chọn.
CheckedItems.Count CheckedIndices.Count	Số dòng đã chọn.

Ví dụ 18: Thiết kế Form như hình bên dưới, với yêu cầu xử lý các điều khiển tương tự ví dụ 15.

Code xử lý sự kiện Form_Load:

```
clbMonhoc.Items.Add("Công nghệ .NET");
clbMonhoc.Items.Add("Cơ sở dữ liệu");
clbMonhoc.Items.Add("Cấu trúc dữ liệu");
clbMonhoc.Items.Add("Mạng máy tính");
clbMonhoc.Items.Add("Nhập môn lập trình");
clbMonhoc.Items.Add("Tin học văn phòng");
```

Code xử lý sự kiện Click của ListBox chứa tên các môn học (lstMonhoc), lấy nội dung và chỉ số dòng các môn học đã chọn gán vào các TextBox tương ứng (txtMonhocDachon, txtCSdongDachon):

```
private void clbMonhoc_Click(object sender, EventArgs e)
{
    int sodongdachon = clbMonhoc.CheckedItems.Count;
    string s = "";
    string cs = "";
    for (int i = 0; i < sodongdachon; i++)
    {
        s += clbMonhoc.CheckedItems[i].ToString() + "\r\n";
        cs += clbMonhoc.CheckedIndices[i] + " ";
    }
}
```

```

    }
    txtMonhocDaChon.Text = s;//Lấy nội dung các dòng đã chọn
    txtCSdongDaChon.Text = cs;//Lấy chỉ số các dòng đã chọn
}

```

Code xử lý nút THÊM:

```

private void btThem_Click(object sender, EventArgs e)
{
    clbMonhoc.Items.Insert(Convert.ToInt32(nuVitri.Value),
    txtMonhoc.Text);
}

```

Code xử lý nút XÓA:

```

private void btXoa_Click(object sender, EventArgs e)
{
    int sodongdachon = clbMonhoc.CheckedItems.Count;
    for(int i=sodongdachon-1; i>=0; i--)
        lstMonhoc.Items.RemoveAt(lstMonhoc.CheckedIndices[i]);
}

```

3.15. ListView

ListView: là điều khiển dùng để hiển thị nhiều đối tượng, mỗi đối tượng trong ListView là một Item, mỗi Item là đối tượng của lớp ListViewItem. Mỗi Item có thuộc tính Text, các SubItem hiển thị ở các cột tiếp theo.

Thuộc tính thường sử dụng:

Bảng 8.1. Mô tả ý nghĩa các thuộc tính thường sử dụng của ListView

Thuộc tính	Ý nghĩa
Items	Danh sách dữ liệu hiển thị trên ListView
SelectedItems	Trả về danh sách các dữ liệu được chọn trên Listview
MultiSelect	Cho phép chọn một lúc nhiều dòng dữ liệu trên danh sách hay không

Phương thức:

Clear: xóa toàn bộ dữ liệu trên ListView

Sự kiện: SelectIndexChanged: xảy ra khi thay đổi dòng chọn.

Hiển thị tiêu đề:

- Đặt thuộc tính View = Detail.
- Trong màn hình properties của Window, click vào nút (...) tại thuộc tính Columns. Xuất hiện màn hình columnHeader Collection Editor appears. Nhấn Add để chọn thêm mới 1 cột, nhập vào giá trị tại các thuộc tính:

- + Text: tiêu đề
- + Text Alignment: canh lề
- + Width: độ rộng

- Đưa dữ liệu vào ListView:

Thêm dữ liệu cột đầu tiên:

```
Ten_ListView.Add("chuỗi cần thêm");
```

Thêm dữ liệu vào các cột kế tiếp:

```
Ten_ListView.Item[dòng].SubItems.Add("chuỗi cần thêm");
```

- Tổng số phần tử trong danh sách được chọn:

```
Ten_ListView.SelectedItems.Count
```

```
Ten_ListView.SelectedIndices.Count
```

- Cập nhật một dòng đang được chọn trên ListView (giả sử 1 dòng dữ liệu có 3 cột):

```
Ten_LV.SelectedItems[0].SubItems[0].Text = "giá trị cần cập nhật";
```

```
Ten_LV.SelectedItems[0].SubItems[1].Text = "giá trị cần cập nhật";
```

```
Ten_LV.SelectedItems[0].SubItems[2].Text = "giá trị cần cập nhật";
```

- Xóa 1 dòng trên ListView:

```
Ten_LV.Items.RemoveAt(chỉ số dòng cần xóa);
```

- Xóa tất cả các phần tử đang được chọn trên ListView:

```
Ten_LV.Items.Clear();
```

Ví dụ 19: Thiết kế Form như hình bên dưới, khi Form thực thi sẽ hiển thị thông tin các môn học trên ListView. Khi chọn một dòng trên ListView thông tin chi tiết sẽ hiển thị trên các TextBox tương ứng, nếu người dùng chỉnh sửa thông tin trên các TextBox và nhấp nút CẬP NHẬT thông tin sẽ được cập nhật trên ListView, nếu người dùng chọn XÓA thì dòng dữ liệu đang chọn sẽ xóa khỏi ListView. Người dùng nhập thông tin vào các TextBox và nhấp nút THÊM dữ liệu sẽ được thêm vào ListView.

Mã học phần	Tên học phần	Số tín chỉ
30073	Công nghệ .NET	3
30083	Công nghệ Web và ứng dụng	2
31323	Nhập môn lập trình	3

Học phần

Mã HP: 30073

Tên HP: Công nghệ .NET

Số TC: 3

THÊM XÓA CẬP NHẬT THOÁT

Code xử lý sự kiện Form_Load:

```
//Thêm môn học thứ 1
```

```
lvMonhoc.Items.Add("30073");
```

```
lvMonhoc.Items[0].SubItems.Add("Công nghệ .NET");
```

```
lvMonhoc.Items[0].SubItems.Add("3");
//Thêm môn học thứ 2
lvMonhoc.Items.Add("30083");
lvMonhoc.Items[1].SubItems.Add("Công nghệ Web và ứng dụng");
lvMonhoc.Items[1].SubItems.Add("2");
//Thêm môn học thứ 3
lvMonhoc.Items.Add("31323");
lvMonhoc.Items[2].SubItems.Add("Nhập môn lập trình");
lvMonhoc.Items[2].SubItems.Add("3");
```

Code xử lý sự kiện Click của ListView:

```
private void lvMonhoc_Click(object sender, EventArgs e)
{
    txtMaHP.Text = lvMonhoc.SelectedItems[0].SubItems[0].Text;
    txtTenHP.Text= lvMonhoc.SelectedItems[0].SubItems[1].Text;
    nuSoTC.Value=
    Convert.ToInt32(lvMonhoc.SelectedItems[0].SubItems[2].Text);
}
```

Code xử lý nút THÊM:

```
private void btThem_Click(object sender, EventArgs e)
{
    //Đếm số dòng đang có trên ListView
    //Thêm vào dòng cuối cùng của ListView
    int soptu = lvMonhoc.Items.Count;
    lvMonhoc.Items.Add(txtMaHP.Text);
    lvMonhoc.Items[soptu].SubItems.Add(txtTenHP.Text);

    lvMonhoc.Items[soptu].SubItems.Add(nuSoTC.Value.ToString());
}
```

Code xử lý nút CẬP NHẬT:

```
private void btCAPNHAT_Click(object sender, EventArgs e)
{
    lvMonhoc.SelectedItems[0].SubItems[0].Text = txtMaHP.Text;
    lvMonhoc.SelectedItems[0].SubItems[1].Text = txtTenHP.Text;
    lvMonhoc.SelectedItems[0].SubItems[2].Text =
    nuSoTC.Value.ToString();
}
```

Code xử lý nút XÓA (xóa dòng đang chọn):

```
private void btXoa_Click(object sender, EventArgs e)
{
    lvMonhoc.Items.RemoveAt(lvMonhoc.SelectedIndex[0]);
}
```

Xử lý dữ liệu khi người dùng chọn nhiều dòng dữ liệu trên ListView:

Thuộc tính	Ý nghĩa
SelectedItems	Chứa các giá trị của các dòng dữ liệu đang được

	chọn.
SelectedIndices	Chứa các chỉ số index của các dòng dữ liệu đang được chọn.
SelectedItems.Count SelectedIndices.Count	Số dòng dữ liệu đang được chọn.

3.16. TreeView

TreeView: dữ liệu hiển thị theo dạng cây (giống như cây thư mục)

Thuộc tính thường sử dụng:

Bảng 8.2. Mô tả ý nghĩa các thuộc tính thường sử dụng của TreeView

Thuộc tính	Ý nghĩa
BorderStyle	Chọn dạng viền cho TreeView
CheckBox	Có hiển thị CheckBox để chọn hay không
SelectionNode	Trả về hoặc gán nút được chọn
ImageList	Danh sách hình hiển thị trên Image List
ImageIndex	Hình hiển thị trên các nút của TreeView
SelectImageIndex	Hình hiển thị khi nút được chọn

Nodes: tập hợp các nút thuộc cây, để thêm nút vào cây:

- Thêm Node gốc vào TreeView (treeName):

```
treeName.Nodes.Add("Giá trị cần thêm");
```

- Thêm các Node con vào Node gốc:

```
treeName.Nodes[0].Nodes.Add("Giá trị cần thêm ");
treeName.Nodes[0].Nodes.Add("Giá trị cần thêm ");
```

- Xóa node:

```
treeName.Nodes.Remove(Node cần xóa);
```

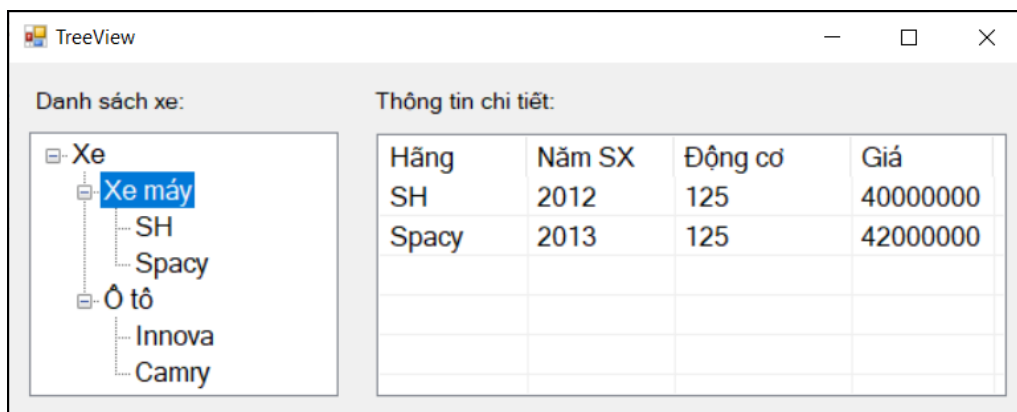
- Các phương thức:

ExpandAll: hiển thị tất cả các thành phần của cây

CollapseAll: thu gọn cây lại chỉ còn nút gốc

- Sự kiện: AfterSelect: xảy ra khi nút được chọn trên cây thay đổi, treeName.SelectedNode.Text để biết giá trị nút được chọn.

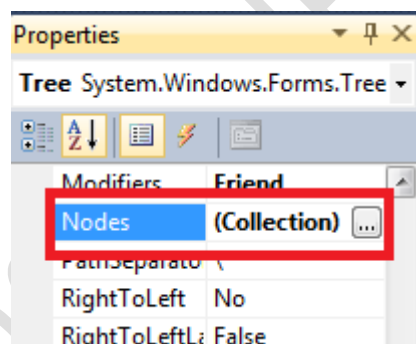
Ví dụ 20: Thiết kế màn hình như sau, khi chương trình thực thi: danh sách xe sẽ hiển thị trên TreeView, khi người dùng chọn Xe máy hoặc Ô tô thì thông tin chi tiết các xe sẽ hiển thị trên ListView tương ứng:



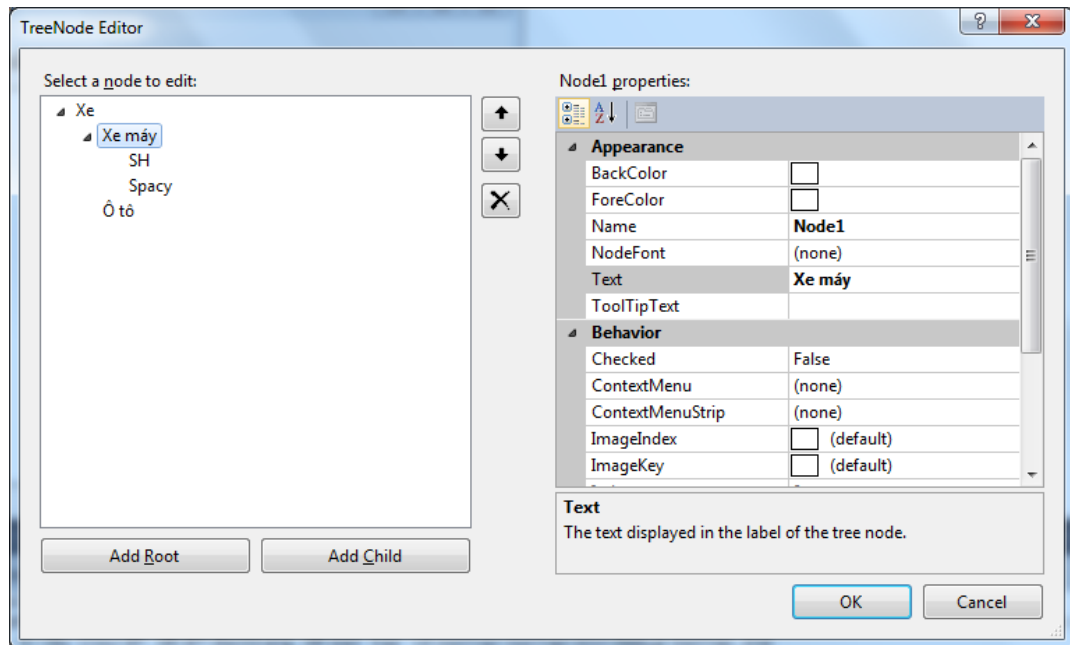
Thông tin chi tiết các phương tiện:

Loại xe	Hãng	Năm SX	Động cơ	Giá
Xe máy	SH	2012	125	40000000
	Spacy	2013	125	42000000
Ô tô	Innova	2014	150	52000000
	Camry	2013	100	150000000

Dùng lệnh hoặc vào nhấn phải vào TreeView → Properties → Nodes



Hình 8.1. Minh họa thiết kế TreeView



Hình 8.2. Minh họa thiết kế TreeView (tt)

Ngoài ra còn có thể xử lý code sự kiện Form_Load

```
TreeNode tnode;
tnode = treeXe.Nodes.Add("Xe");
treeXe.Nodes[0].Nodes.Add("Xe máy");
treeXe.Nodes[0].Nodes.Add("Ô tô");
treeXe.Nodes[0].Nodes[0].Nodes.Add("SH");
treeXe.Nodes[0].Nodes[0].Nodes.Add("Spacy");
treeXe.Nodes[0].Nodes[1].Nodes.Add("Innova");
treeXe.Nodes[0].Nodes[1].Nodes.Add("Camry");
```


3.17. ImageList

ImageList: là điều khiển chứa tập hợp các hình ảnh dùng để gán cho thuộc tính ImageList của các điều khiển khác, khi thực thi điều khiển này không hiển thị trên Form.

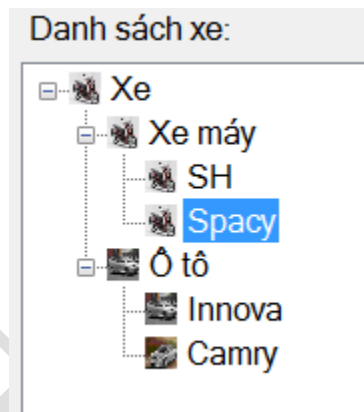
- Để đưa hình ảnh vào ImageList: vẽ điều khiển vào Form, trong thuộc tính Images click vào nút ... → xuất hiện hộp thoại, chọn Add chọn hình cần đưa vào ImageList.

- Trong thuộc tính ImageList của các điều khiển khác chọn tên điều khiển Image này và trong thuộc tính **ImageIndex** nhập vào vị trí hình sẽ hiển thị, hình đầu tiên có giá trị =0.

- Có thể thay đổi giá trị **ImageIndex** trong lệnh để thay đổi hình hiển thị trên điều khiển.

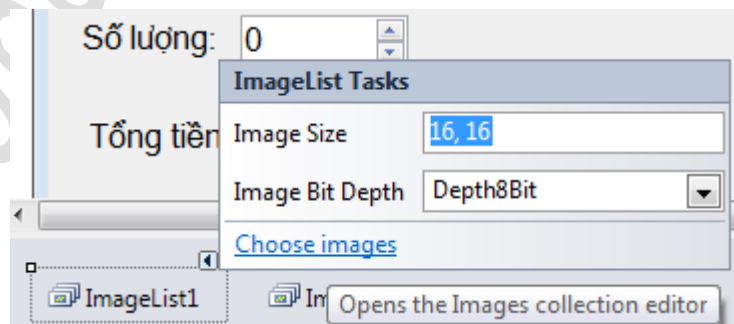
- **SelectedImageIndex**: hình hiển thị trên nút khi click vào.

Ví dụ: gán hình vào điều khiển TreeView như hình bên dưới:



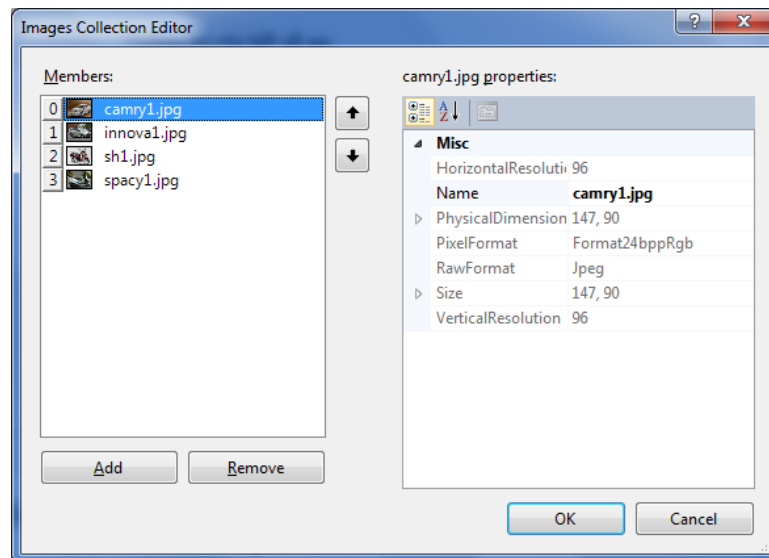
Hình 8.3. Minh họa gán hình vào TreeView

Kéo điều khiển ImageList vào Form, ở smartag, chọn *Choose Images* để thêm danh sách các hình vào:



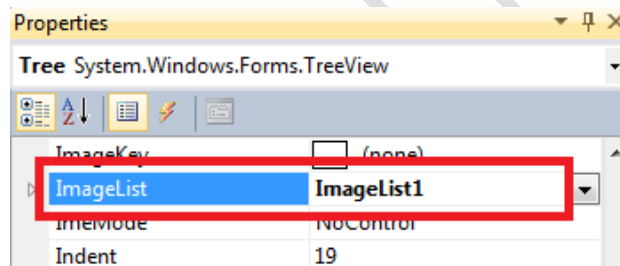
Hình 8.4. Minh họa gán hình vào TreeView (tt)

Nhấn nút Add chọn đường dẫn lưu hình để thêm hình vào ImageList:



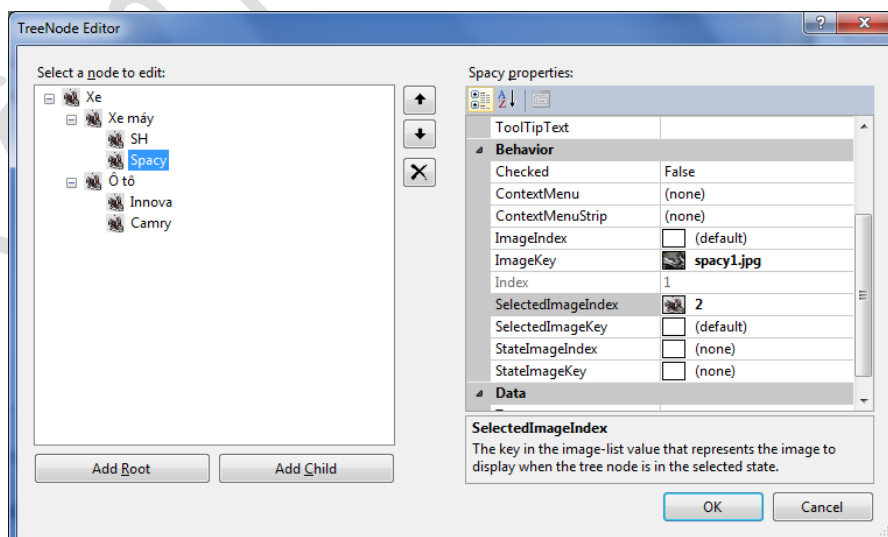
Hình 8.5. Minh họa gắn hình vào TreeView (tt)

Thuộc tính ImageList của Treeview, chọn tên ImageList để hiển thị hình ảnh:



Hình 8.6. Minh họa gắn hình vào TreeView (tt)

Vào thuộc tính *Nodes* của điều khiển TreeView để chọn hình hiển thị vào hình lúc nhấn vào từng Nodes của TreeView:

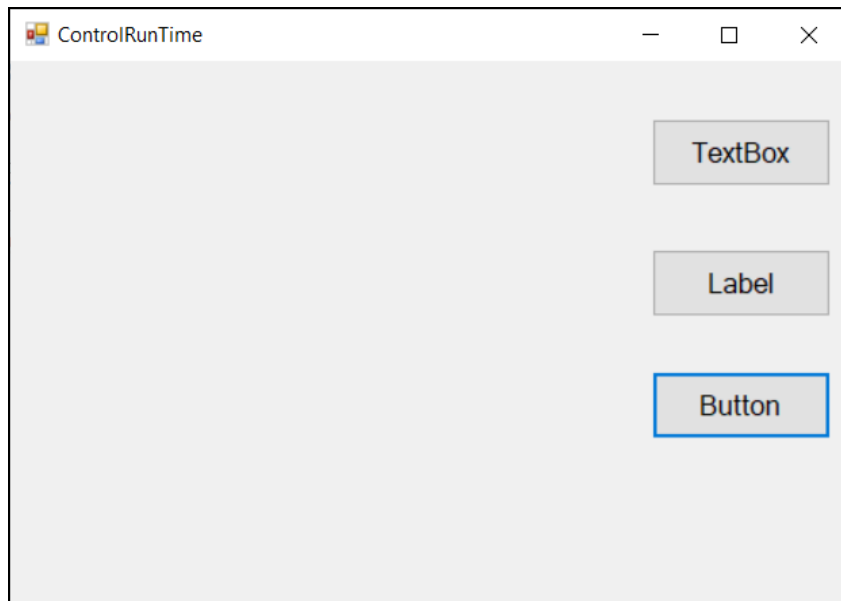


Hình 8.7. Minh họa gắn hình vào TreeView (tt)

4. Tạo Control động

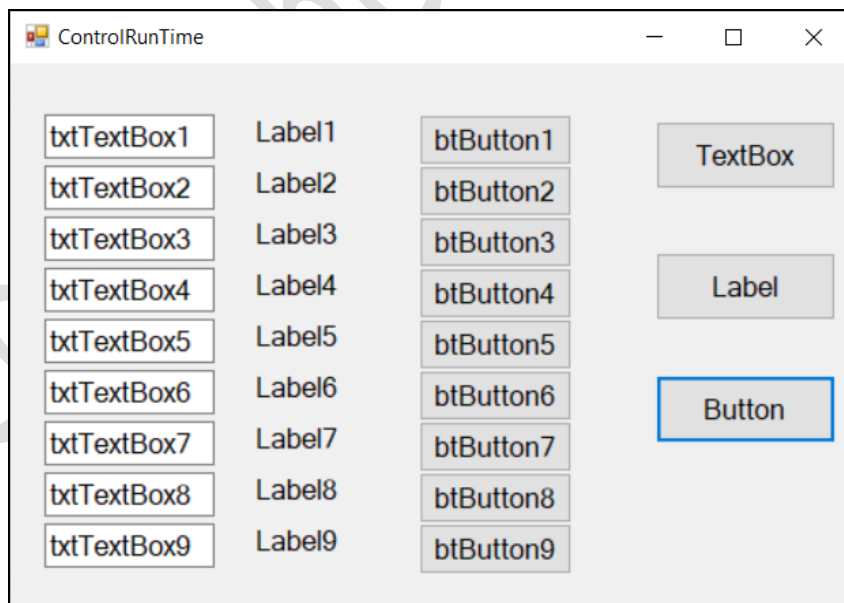
Control động là những Control không được thiết kế từ những Control có sẵn từ thanh Toolbox mà thiết kế chúng bằng Code, khi chương trình thực thi các Control sẽ xuất hiện.

Ví dụ tạo Texbox, Label, Button động như sau:



Hình 7.1. Minh họa chương trình phát sinh Control

Mỗi khi người dùng nhấn **TextBox** sẽ phát sinh ra một TextBox (mỗi lần nhấn phát sinh một TextBox), tương tự như vậy cho **Lable** và **Button**:



Hình 7.2. Minh họa chương trình phát sinh Control (tt)

- Code phát sinh TextBox:

```
int a = 1;  
int b = 1;  
private void btTextBox_Click(object sender, EventArgs e)
```

```

{
    TextBox txtTextBox = new TextBox();
    this.Controls.Add(txtTextBox);
    txtTextBox.Top = a * 30;
    txtTextBox.Left = 20;
    txtTextBox.Name = "txtTextBox" + b;
    txtTextBox.Text = "txtTextBox" + b;
    a += 1;
    b += 1;
}

```

- Code phát sinh Label:

```

int c = 1;
int d = 1;
private void btLabel_Click(object sender, EventArgs e)
{
    Label lbLabel = new Label();
    this.Controls.Add(lbLabel);
    lbLabel.Top = c * 30;
    lbLabel.Left = 140;
    lbLabel.Name = "lbLabel" + d;
    lbLabel.Text = "Label" + d;
    c += 1;
    d += 1;
}

```

- Code phát sinh Button:

```

int n = 1;
int m = 1;
private void btButton_Click(object sender, EventArgs e)
{
    Button btButton = new Button();
    this.Controls.Add(btButton);
    btButton.Top = n * 30;
    btButton.Left = 240;
    btButton.Name = "btButton" + m;
    btButton.Text = "btButton" + m;
    btButton.Height = 30;
    btButton.Width = 90;
    n += 1;
    m += 1;
}

```

5. Một số sự kiện thường dùng của các Control

Một số sự kiện thường dùng của điều khiển: Trong chương trình **Windows Form**, các *Form* và điều khiển đều là các *Class* dẫn xuất từ *Class System.Windows.Forms.Control*. Nhờ đó chúng kế thừa tất cả (hơn 50) các sự kiện public chứa trong đối tượng **Control**.

Tuy vậy mỗi kiểu điều khiển chỉ dùng các sự kiện có ý nghĩa đối với nó, ví dụ kiểu điều khiển button chỉ dùng sự kiện Click và không dùng sự kiện Double Click.

Sau đây là một số sự kiện thường dùng của điều khiển:

Sự kiện	Đối số	Phát sinh khi
Sự kiện chung		
Enter	EventArgs	Focus đi vào điều khiển
TextChanged	EventArgs	Giá trị thuộc tính Text của điều khiển thay đổi
Leave	EventArgs	Focus rời khỏi điều khiển
Validating	CancelEventArgs	Điều khiển đang được kiểm tra hợp lệ dữ liệu nhập
Validated	EventArgs	Điều khiển đã hoàn thành kiểm tra hợp lệ dữ liệu nhập
Paint	PaintEventArgs	Điều khiển được vẽ
Sự kiện liên quan chuột		
Click	EventArgs	Điều khiển được nhấp
Double Click	EventArgs	Điều khiển được nhấp kép
MouseEnter	EventArgs	Con trỏ chuột đi vào trong biên của điều khiển
MouseHover	EventArgs	Con trỏ chuột dừng và nghỉ trên điều khiển
MouseMove	MouseEventArgs	Con trỏ chuột di chuyển trong phạm vi điều khiển
MouseLeave	EventArgs	Con trỏ chuột đi ra khỏi biên của điều khiển
MouseDown	MouseEventArgs	Con trỏ chuột trên điều khiển và một nút chuột được nhấn
MouseUp	MouseEventArgs	Con trỏ chuột trên điều khiển và một nút chuột được thả
Sự kiện liên quan bàn phím		
KeyDown	KeyEventArgs	Một phím được nhấn xuống khi điều khiển đang có focus
KeyPress	KeyPressEventArgs	Một phím được nhấn khi điều khiển đang có focus
KeyUp	KeyEventArgs	Một phím được thả lên khi điều khiển đang

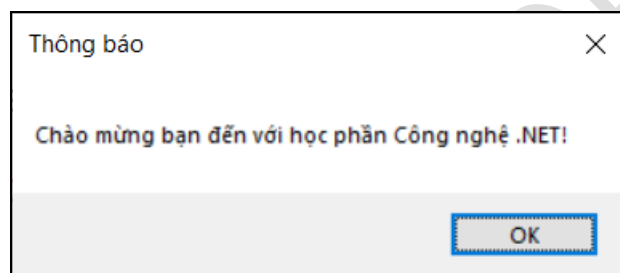
		có focus
--	--	----------

Ví dụ:

6. Message Box (hộp thông điệp)

MessageBox là một lớp nằm trong System.Windows.Forms có phương thức Show để hiển thị thông báo. Có rất nhiều kiểu thông báo, người dùng có thể điều chỉnh nội dung thông báo, tiêu đề, các nút (OK, Cancel, Yes, No,...) biểu tượng. MessageBox thường sử dụng trong các trường hợp: thông báo cho người dùng biết không thể sử dụng chức năng hay lưu ý một số vấn đề trong chương trình, cần sự xác nhận của người dùng với một số lệnh như: xóa dữ liệu, cập nhật dữ liệu, tắt ứng dụng,...

Ví dụ: Viết code xử lý khi Form thực thi sẽ xuất hiện hộp thoại thông báo: Chào mừng bạn đến với học phần Công nghệ .NET!



Code xử lý:

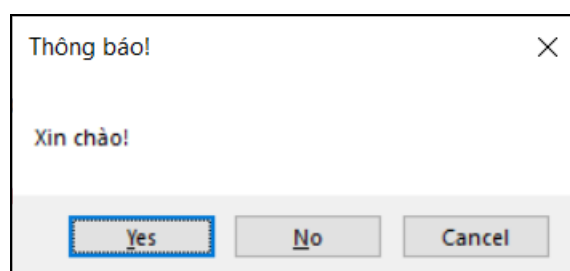
```
MessageBox.Show("Chào mừng bạn đến với học phần Công nghệ .NET!", "Thông báo");
```

Để cài đặt nút chọn, thêm 1 tham số kiểu enum là MessageBoxButtons.<loại nút>. Các loại nút có sẵn bao gồm AbortRetryIgnore, OK, OKCancel, YesNo, YesNoCancel...

Ví dụ code xử lý:

```
MessageBox.Show("Xin chào!", "Thông báo!", MessageBoxButtons.YesNoCancel);
```

Hộp thoại hiển thị:

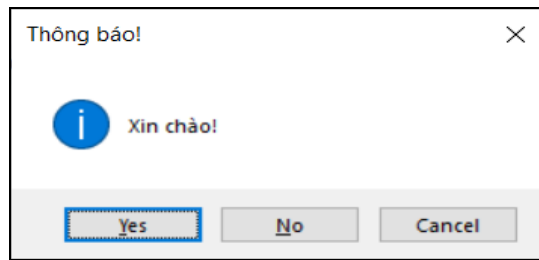


Để thêm vào icon, thêm tham số kiểu enum là MessageBoxIcon.<loại icon>, thường sử dụng là Warning, Error, Information, Question,...

Ví dụ code xử lý:

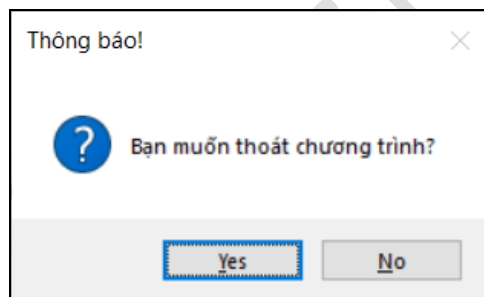
```
MessageBox.Show("Xin chào!", "Thông báo!", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Information);
```

Hộp thoại hiển thị:



Để xử lý sự kiện khi nhấp vào một Button trên MessageBox: dùng một biến kiểu DialogResult để lưu lại kết quả trả về của phương thức MessageBox.Show(), sau đó xét giá trị của biến kiểu DialogResult để xử lý.

Ví dụ: khi nhấp vào nút Thoát thì sẽ xuất hiện thông báo hỏi người dùng có muốn thoát chương trình không, nếu chọn nút Yes: chương trình sẽ thoát, chọn No: chương trình sẽ không thoát.



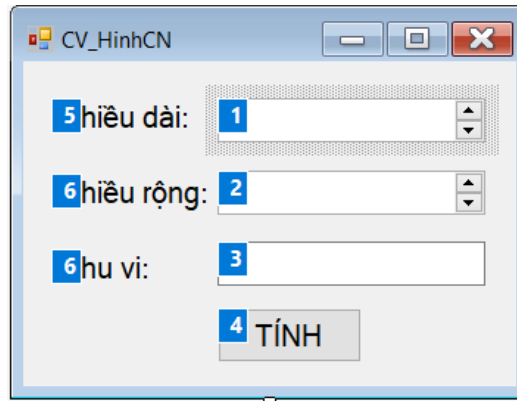
Code xử lý:

```
DialogResult kq = MessageBox.Show("Bạn muốn thoát chương trình?",  
"Thông báo!", MessageBoxButtons.YesNo, MessageBoxIcon.Question);  
if (kq==DialogResult.Yes)  
{  
    this.Close();  
}
```

7. Phím tắt, thứ tự Tab và thiết lập Focus

Trong giao diện window, muốn di chuyển focus từ Control này sang Control khác có thể sử dụng phím Tab. Thứ tự giữa các Control được xác định bằng thuộc tính Tabindex. Để xem chế độ hiển thị trị số Tabindex của mỗi Control, chọn View → Tab Order, nhấp chuột vào các chỉ số để tăng giảm chỉ số tab. Muốn thoát khỏi Tab Order mode chọn View → Tab Order một lần nữa. Trong trường hợp có sử dụng các GroupBox thì thứ tự sẽ được ưu tiên dựa trên GroupBox trước.

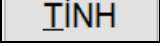
Ví dụ: chỉnh sửa lại các Tab Order cho Form bên dưới:

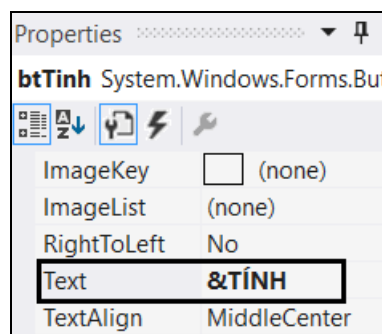


Hình 2.5. Minh họa cách chỉnh sửa Tab Order trên Form

- Phím tắt:

Thuộc tính Text của Button: gõ & để tạo gạch chân trước ký tự, đồng thời khi nhấn tổ hợp phím <Alt+ký tự> sẽ tương tự như khi nhấn trên button. Thiết lập Focus: mục đích yêu cầu người sử dụng phải thực hiện thao tác tại vị trí con trỏ hiện hành.

Ví dụ: tạo phím tắt cho nút  trên Form, cho phép người sử dụng có thể sử dụng phím: Alt+T thay vì nhấn trên nút TÍNH, gõ: &TÍNH tại thuộc tính Text của nút TÍNH.



8. Nguyên tắc khi thiết kế Form và các Control trên Form

8.1. Các thuộc tính của Form

Bảng 3.13. Mô tả ý nghĩa các thuộc tính của Form

Thuộc tính	Ý nghĩa
Maximize Box	Trạng thái hiển thị nút phóng to của Form.
Minimize Box	Trạng thái hiển thị nút thu nhỏ của Form.
Top most	Trạng thái Form luôn hiện ưu tiên trong active Form.
WindowState	Trạng thái hiển thị của Form: normal, minimized, maximized.
StartPosition	Vị trí Form hiển thị khi thực thi chương trình.

Icon	Chọn biểu tượng cho Form.
Size	Quy định kích thước cho Form với đơn vị tính là Pixel.
FormBorderStyle	Kiểu đường viền của Form (None, FixedSingle, Fixed3D, FixedDialog, Sizable, FixedToolWindow, SizableToolWindow).

Bảng 3.14. Mô tả ý nghĩa giá trị của thuộc tính FormBorderStyle

Loại	Giá trị	Diễn giải
None	0	Không có đường biên
FixedSingle	1	Tương tự như Fixed3D
Fixed3D	2	Trông giống như 3 chiều
FixedDialog	3	Như hộp hội thoại
Sizable	4	Mặc nhiên
FixedToolWindow	5	Thanh caption nhỏ và không có nút Close
SizableToolWindow	6	Giống FixedToolWindow nhưng đường viền mỏng

Sau khi đã thiết lập các thuộc tính cho đối tượng Form (nền Form) mới tiến hành tạo và thiết lập thuộc tính cho từng đối tượng khác. Các thuộc tính chung giữa Control và Form, giá trị của thuộc tính trên Control được xác định theo Form.

Bảng 3.15. Mô tả ý nghĩa các thuộc tính của Form

Thuộc tính	Ý nghĩa
AcceptButton	Nút lệnh trên Form sẽ được gọi sự kiện Click khi phím Enter được nhấn.
CancelButton	Nút lệnh trên Form sẽ được gọi sự kiện Click khi phím Escape được nhấn.
IsMdiChild	Trả về True/False cho biết Form có phải là Form con của ứng dụng MDI không.
IsMdiContainer	Giá trị True/False cho biết Form có phải là Form chứa các Form con của ứng dụng MDI hay không.
Text	Văn bản hiển thị trên thanh tiêu đề của Form.
BackgroundImage	Xác định tập tin làm hình nền cho Form.
BackColor	Màu nền của Form.
Font	Chọn Font chữ cho Form.
ForeColor	Chọn màu chữ cho Font.

Size	Thay đổi kích thước của Form.
------	-------------------------------

- Sự kiện thường sử dụng của Form:

Bảng 3.16. Mô tả ý nghĩa các sự kiện thường sử dụng của Form

Sự kiện	Ý nghĩa
Form_Load	Sự kiện này xảy ra khi mỗi lần gọi thể hiện một biểu mẫu. dùng sự kiện này để khởi tạo các biến, điều khiển cho các thể hiện của biểu mẫu.
Form_Closing	Mỗi lần Form đóng lại thì sự kiện này sẽ được phát sinh.
Form_Resize	Sự kiện này xảy ra mỗi khi Form thay đổi kích thước.

8.2. Một số lưu ý trong thiết kế Form

Kích thước Form: tỷ lệ 4×3: để màn hình có kích thước cân xứng.

Kích thước Control: chiều cao: nên sử dụng chiều cao mặc định sẵn của Control.

Trường hợp đặc biệt:

- TextBox MultiLines: đảm bảo không bị che 1 phần của dòng.
- Button có image: đảm bảo hiển thị vừa đủ image 16×16 pixel.
- Độ rộng:
 - + Các TextBox, Button, ComboBox trên cùng một Form nên có kích thước thống nhất.
 - + Text trên Button không nên vượt quá 2 từ.
 - + Nếu Text trên Button gồm 2 từ trở xuống: bắt buộc sử dụng độ rộng mặc định.
 - + Đối với những trường có độ rộng cố định hoặc ít khi thay đổi (ví dụ như trường có kiểu dữ liệu là Date thì độ rộng cố định là 10 ký tự), tuân thủ theo quy định: độ rộng Control phải hiển thị hết thông tin trong đó.
- Không nên để độ rộng Control vượt quá độ rộng của trường.
- Lưu ý: Label đặt AutoSize=FALSE, TextBox đặt AutoSize=TRUE
- Quy định khoảng cách giữa các Control
 - + Các Control cách mép Form 1 ô grid (cả 4 phía).
 - + TextBox, ComboBox cách Label dài nhất 1 ô grid.
 - + Các Control cách nhau 1 ô grid cả chiều dọc và ngang.
 - + Riêng trường hợp sử dụng GroupBox có text thì các Control bên trong GroupBox cách mép trên của GroupBox 2 ô grid. GroupBox không có text khoảng cách vẫn là 1 ô Grid.
- Giao diện Control

+ Font & Color: sử dụng thiết lập mặc định. Chỉ thay đổi khi yêu cầu thiết kế chỉ rõ.

+ Canh lề Text trên Control: Chiều ngang (HAlign), chữ canh trái, số canh phải, riêng với Button thì luôn canh giữa.

- Canh lề Control

+ Label: canh trái

+ Textbox, ComboBox: canh đều hai bên

- Tab Order:

+ Nên thiết lập Tab Order trên mọi giao diện (Form, Control, ...) theo nguyên tắc: từ trái sang phải, từ trên xuống dưới.

+ Nên thiết lập Tab Order theo đúng thứ tự cho mọi Control trên Form, kể cả Control không focus vào được như Label, GroupBox, hay Control bị ẩn.

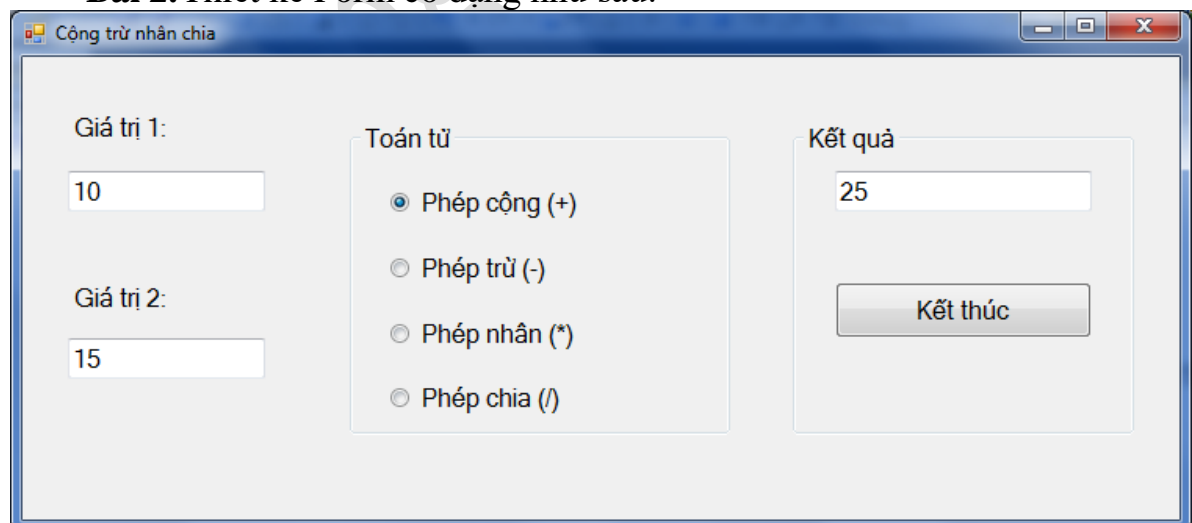
- Anchor & Dock: nên thiết lập Anchor và Dock cho Control trên các Form, Container không cố định kích thước (Sizable).

- Với thông tin yêu cầu người dùng không được bỏ trống mà bắt buộc nhập (AllowNull = FALSE) thì Label cho thông tin đó phải sử dụng ký hiệu (*) ở cuối và thiết lập shortcut key.

BÀI TẬP

Bài 1. Sinh viên làm lại các ví dụ trong bài.

Bài 2. Thiết kế Form có dạng như sau:



Bài 3. Thiết kế Form có dạng như sau:

Bài 4. Thiết kế Form như hình sau:

Khi người dùng nhập vào các thông tin trên các Textbox: họ tên, ngày sinh, sở thích và nhấn nút **Xác nhận** sẽ xuất hiện một hộp thoại xác nhận lại những thông tin đã nhập.

Bài 5. Thiết kế Form thực hiện các chức năng sau:

CỬA HÀNG VI TÍNH MINH KHÔI

Các loại hàng hóa

☒ Laptop

☐ USB

☐ Mouse

Các dòng sản phẩm và hình minh họa

HP
 Vaio
 DELL
 Lenovo






Yêu cầu:

- Khi nhấn Laptop: hiển thị các loại laptop trên ListBox. Các hình thể hiện minh họa cho các laptop.
- Khi nhấn USB, Form như sau:

CỬA HÀNG VI TÍNH MINH KHÔI

Các loại hàng hóa

☐ Laptop

☒ USB

☐ Mouse

Các dòng sản phẩm và hình minh họa

Trancer
 KingMax
 Kington
 Adapt






- Khi nhấn Mouse, có 3 loại chuột trên listbox.
- Các hình minh họa, sinh viên tự tìm kiếm.
- Khi nhấn trên ListBox, chỉ hiện duy nhất 1 hình của sản phẩm đang chọn.

CỬA HÀNG VI TÍNH MINH KHÔI

Các loại hàng hóa

☐ Laptop

☒ USB

☐ Mouse

Các dòng sản phẩm và hình minh họa

Trancer
KingMax
Kington
Adapt



Hướng phát triển bài toán:

- Thiết kế đơn giá cho từng sản phẩm. Khi chọn sản phẩm nào, hiện giá của sản phẩm trên ToolTip của hình.
- Thiết kế TextBox, chứa số lượng sản phẩm đặt mua.
- Thiết kế nút **Đặt hàng**, đưa thông tin sản phẩm, đơn giá, số lượng vào listBox.
- Tạo các Label, thể hiện các thông tin: Tổng số lượng mặt hàng, tổng các loại sản phẩm, tổng tiền của từng hóa đơn.
- Tạo Label thể hiện tổng tiền trong ngày (kể từ lúc chương trình thực thi).

Bài 6. Thiết kế mô phỏng gameshow Trúcxinh của đài truyền hình.



Yêu cầu:

- Dòng chữ “Gameshow Trúcxinh”, xuất hiện từng chữ sau 1 s, khi hiện hết dòng thì bắt đầu lại từ đầu.

- Thiết kế 4 Label, 4 PictureBox. Khi Form load, các PictureBox ẩn.
- Khi nhấn vào Label, hiện PictureBox bên dưới. Nếu đã mở 1 Label, kiểm tra hình đã mở trước đó có giống với hình lần 2 không? Nếu giống, hiện 2 hình, nếu không giống đóng 2 hình.



Hướng dẫn thực hiện:

- Khai báo biến flag: dùng để xác định có hình đang mở hay không (True/False)
- Dùng phương thức Load để hiện hình trên PictureBox. PictureBox.Load(đường dẫn)
- Nếu đường dẫn của 2 hình giống nhau thì 2 hình đó giống nhau. (sử dụng thuộc tính ImageLocation)

Hướng phát triển bài toán:

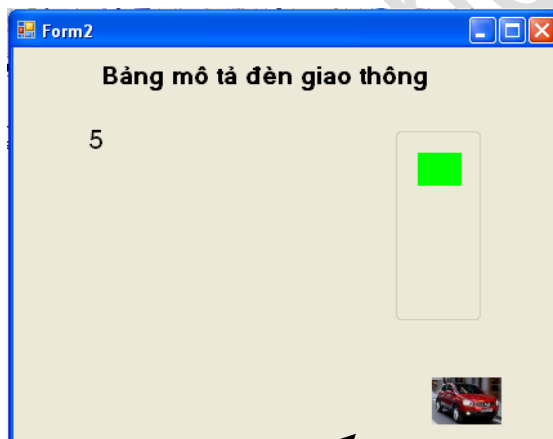
- Tạo TextBox: nhập vào số lượng các Label muốn thể hiện trên Form. Khi nhấn nút Tạo, các Label và PictureBox sẽ hiển thị.
- Thay đổi code trong các Label thành chương trình con. Khi nhấn vào các Label, chỉ cần gọi chương trình con đó. Tránh tình trạng code giống nhau rất nhiều, khó theo dõi.
- Thay đổi ý tưởng chương trình: khi nhấn 2 hình giống nhau thì ẩn luôn 2 hình, khi nào mở hết hình thì đóng Form. Qui định thời gian của chương trình là 20 giây. Hết thời gian đó, nếu không mở hết hình thì thông báo “Hết thời gian. You lost”, ngược lại “You win” dừng thời gian, hiển thị hình nền. Có thể gợi ý đoán ca dao, thơ cho hình nền.

Bài 7. Thiết kế Form có dạng như sau:

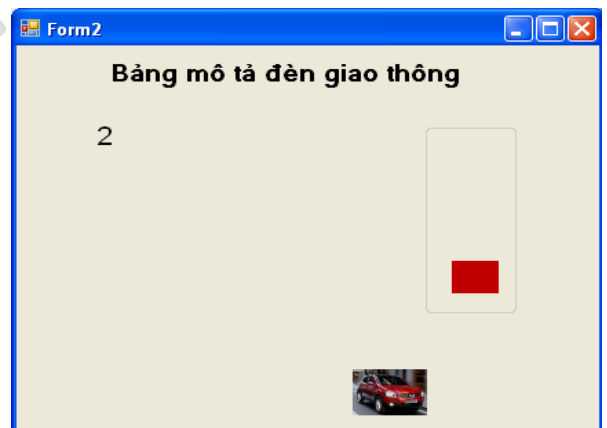


Yêu cầu:

- Thiết kế màn hình gồm các điều khiển sau: 3 Panel (thể hiện đèn xanh, vàng, đỏ), 1 PictureBox thể hiện hình xe hơi, 1 Label thể hiện thời gian. Khi Form Load, chỉ có 1 đèn xanh hiển thị, Label thể hiện giờ.
- Đèn xanh được bật trong 15 giây, đèn vàng 3 giây, đèn đỏ 15 giây. Các đèn luân phiên thay đổi theo thời gian.
- Xe hơi thực thi từ trái qua phải, nếu gặp đèn đỏ phải dừng lại, nếu hết biên phải thì thực thi lại từ đầu.



Đèn xanh, được phép thực thi

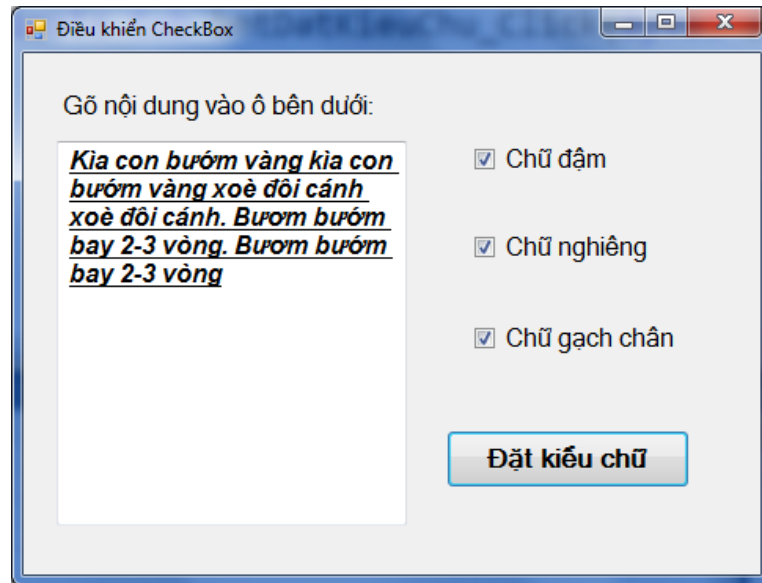


Đèn đỏ, dừng

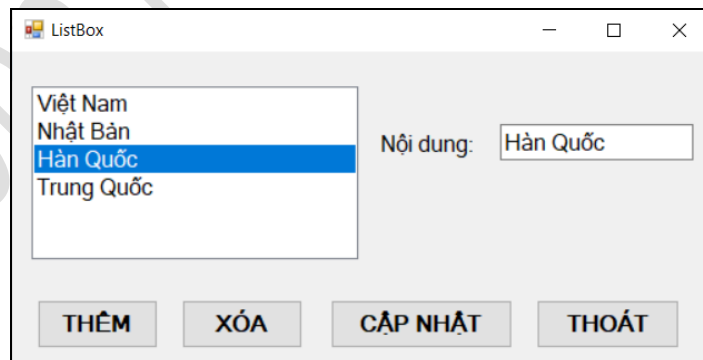
Hướng phát triển bài toán:

Nhập vào số vòng xe thực thi. Hãy cho biết số lần xe phải dừng lại khi gặp đèn đỏ, tổng thời gian đi hết quãng đường theo yêu cầu là bao nhiêu?

Bài 8. Đặt kiểu chữ cho hộp văn bản (đậm, nghiêng, gạch chân) khi người dùng chọn vào các lựa chọn tương ứng sau đó nhấn nút **Đặt kiểu chữ** thì nội dung văn bản trong ô TextBox sẽ được định dạng tương ứng với sự lựa chọn của người dùng.

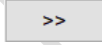



Bài 9. Thiết kế giao diện gồm ListBox có các giá trị: Việt Nam, Trung Quốc, Nhật Bản, Hàn Quốc. Khi người dùng nhập nội dung vào TextBox và nhấp nút Thêm nội dung sẽ được thêm vào ListBox. Khi chọn 1 dòng trên ComboBox nội dung sẽ được hiển thị trên TextBox, nếu người dùng chỉnh sửa nội dung trên Textbox và nhấp nút Cập nhật thì nội dung sẽ được cập nhật vào ListBox, nếu chọn Xóa thì dòng nội dung sẽ được xóa khỏi ListBox.



Bài 10. Thiết kế giao diện như hình bên dưới:

Yêu cầu:

1. Thiết kế giao diện
2. Khi người dùng nhập thông tin vào Textbox Tên món và chọn Nước uống hoặc Món ăn và nhấn nút Thêm thì thông tin món ăn sẽ được thêm vào Listbox tương ứng.
3. Chọn một Món ăn và nhấn nút  thì dòng dữ liệu được chọn bên Món ăn sẽ được xóa khỏi Listbox Món ăn và thêm vào Listbox Nước uống.
4. Chọn một Nước uống và nhấn nút  thì dòng dữ liệu được chọn bên Nước uống sẽ được xóa khỏi Listbox Nước uống và thêm vào Listbox Món ăn.

Bài 11. Thiết kế giao diện như hình bên dưới:

DANH SÁCH CÁC LOẠI XE

Loại xe:

Thông tin chi tiết:

Tên xe:

Màu sắc:

Đơn giá:

Số lượng:

Thành tiền: **105000000**

Yêu cầu:

1. Thiết kế giao diện.
2. Combobox loại xe gồm các loại xe: Vision, Click, SH mode. Khi chọn 1 loại xe sẽ hiển thị thông tin tương ứng như bảng bên dưới.

Loại xe	Màu	Đơn giá
Vision	Xanh	35.000.000
Click	Trắng	42.000.000
SH mode	Đỏ	50.000.000

3. Khi người dùng chọn số lượng và nhấp nút **Tính tiền** thì giá tiền sẽ hiển thị trong Label Thành tiền.

Bài 12. Thiết kế giao diện như hình bên dưới:

Form1

Lớp:

Danh sách lớp:

Họ tên	Lập trình GD	Mạng MT	CSDL SQL
Anh Tuấn	9	8	9
Hoàng Giang	10	7	9
Trâm Anh	8	7	10

Thêm Cập nhật Thoát

Họ tên:

Lập trình GD:

Mạng MT:

CSDL SQL:

Điểm trung bình của SV: Trâm Anh là: 8.33

Yêu cầu:

- Thiết kế giao diện, Combobox Lớp gồm có 2 lớp là: **Cao đẳng 17** và **Cao đẳng 18**
- Khi chọn lớp **Cao đẳng 17** sẽ xuất hiện các thông tin tương ứng như sau

Họ tên	Lập trình GD	Mạng MT	CSDL SQL
Anh Tuấn	9	8	9
Hoàng Giang	10	7	9
Trâm Anh	8	7	10

- Khi chọn lớp **Cao đẳng 18** sẽ xuất hiện các thông tin tương ứng như sau:

Họ tên	Lập trình GD	Mạng MT	CSDL SQL
Hữu Tiến	7	8	9
Thanh Lâm	5	4	6
Yến Như	4	7	7

- Khi chọn 1 dòng vào ListView danh sách lớp thì thông tin chi tiết của sinh viên sẽ được hiện thị chi tiết lên các đối tượng: Họ tên, Lập trình GD, Mạng MT, CSDL SQL. Đồng thời điểm trung bình của sinh viên đang được chọn sẽ hiển thị bên dưới ListView như hình trên.

- Khi người dùng chỉnh sửa nội dung các đối tượng Họ tên, Lập trình GD, Mạng MT, CSDL SQL của dòng đang chọn trên ListView và nhấp nút **Cập nhật** thì thông tin chi tiết của sinh viên sẽ được cập nhật lên ListView

- Khi người dùng nhập các thông tin vào các đối tượng: Họ tên, Lập trình GD, Mạng MT, CSDL SQL và nhấp nút **Thêm** thông tin của sinh viên sẽ được thêm vào ListView

Bài 13. Thiết kế giao diện như sau:

Họ tên	Địa chỉ	Điểm Văn	Điểm Toán
Lê Minh Tấn	Tiền Giang	10	8
Nguyễn Trọng Nhân	Bến Tre	5	8
Nguyễn Thị Lê Thủy	Tiền Giang	3	5
Trần Văn Nhon	Cà Mau	10	9

DS Sv đang được chọn:

- 0 Lê Minh Tấn
- 2 Nguyễn Thị Lê Thủy
- 3 Trần Văn Nhon

Yêu cầu:

- Nhập dữ liệu vào các TextBox, nhấn nút **THÊM** dữ liệu sẽ được thêm vào ListView.

- Khi nhấn vào **1 hoặc nhiều dòng** chọn trên ListView chỉ số dòng và họ tên được hiển thị trên TextBox bên dưới, đồng thời thông tin chi tiết của dòng đầu tiên được chọn sẽ được hiển thị trên các TextBox.

- Sau khi chọn dữ liệu trên ListView, người sử dụng chỉnh sửa thông tin trên các TextBox và click nút **CẬP NHẬT** thông tin mới sẽ được cập nhật trên ListView (có xác nhận thông tin trước khi cập nhật)

- Nhấn nút **XÓA** thông tin các dòng đang được chọn trên ListView sẽ bị xóa (có xác nhận thông tin trước khi xóa).

Bài 14. Thiết kế Form có dạng như sau:

Cafe manage program

Bàn số

Menu

STT	Thuc uong	Don gia
1	cafe	4000
2	cafe-sua	4000
3	sting	7000

>

<

Thuc uong	So luong	Thanh tien
cafe	1	4000
cafe-sua	3	12000
sting	1	7000

Tính 23, 000

Yêu cầu:

- Trong menu, hiển thị tất cả danh mục thức uống trong quán.
- Khi nhấn nút >, thức uống được chọn trong menu sẽ được thêm vào danh sách các thức uống đã chọn. Số lượng tăng thêm 1, thành tiền tăng theo đơn giá của thức uống.
- Khi nhấn nút <, thức uống đang chọn trên danh sách thức uống đã chọn sẽ giảm số lượng -1, thành tiền giảm theo đơn giá. Nếu soluong =0 thì xóa khỏi danh sách.
- Khi nhấn nút Tính, tính tổng thành tiền của các thức uống đã chọn.

Bài 15. Thiết kế Form có dạng như sau:

Mã HP	Tên học phần	Loại HP	Số TC
T	Toán	Bắt buộc	2
VH	Văn học	Tự chọn	3
Sinh	Sinh học	Bắt buộc	4
Hoa	Hóa học	Tự chọn	5

Học phần

Mã học phần:

Tên học phần:

Số tín chỉ:

Loại HP: ☒ Tự chọn ☐ Bắt buộc

Tổng số tín chỉ bắt buộc: **6**

Tổng số tín chỉ tự chọn: **8**

Yêu cầu:

- Khi nhấn nút **Thêm** các thông tin tương ứng trên các textbox và Radio *Loại HP* sẽ được thêm vào Listview thông tin môn học.
- Khi click vào 1 dòng Listview những thông tin tương ứng với dòng được chọn sẽ được hiển thị trên textbox và Radio tương ứng.
- Khi chọn **1 hoặc nhiều dòng** trên Listview và nhấn nút **Xóa** chương trình sẽ xuất hiện 1 thông báo hỏi: *có thật sự muốn xóa không?* Nếu chọn Yes thì mới **Xóa**.
- Sau khi chọn 1 dòng trên Listview các thông tin sẽ hiển thị trên các Textbox và Radio, nếu sửa lại 1 số thông tin **trừ mã học phần** và nhấn nút **Cập nhật** thông tin sẽ được sửa lại tương ứng với mã học phần trên ListView.
- Khi nhấn nút **Tính** sẽ tính tổng các số tín chỉ bắt buộc và tổng số tín chỉ tự chọn.
- Nhấn nút **Thoát** sẽ đóng Form hiện hành.

Bài 16. Thiết kế giao diện như hình bên dưới:

Form1

Mã SV: 018307020 Tên sinh viên: Nguyễn Thiên Trí

Thêm Cập nhật Tìm kiếm

Danh sách sinh viên:

STT	Mã SV	Họ tên sinh viên
1	018307013	Lê Trần Minh Thoại
2	018307020	Nguyễn Thiên Trí
3	018307014	Trần Hữu Tiến

>

<

Danh sách sinh viên nhận học bổng:

STT	Mã SV	Họ tên sinh viên
-----	-------	------------------

Yêu cầu:

1. Thiết kế giao diện.
2. Khi Form thực thi sẽ xuất hiện danh sách sinh viên bên dưới:

STT	Mã SV	Họ tên sinh viên
1	018307013	Lê Trần Minh Thoại
2	018307020	Nguyễn Thiên Trí
3	018307014	Trần Hữu Tiến

3. Khi nhập vào Mã SV và Tên sinh viên và nhấp nút **Thêm** thì thông tin sinh viên sẽ thêm vào Danh sách sinh viên.

4. Khi chọn 1 dòng trên ListView Danh sách sinh viên thì thông tin chi tiết của sinh viên sẽ hiển thị trên các TextBox tương ứng. Nếu người dùng có chỉnh sửa thông tin của sinh viên và nhấp nút **Cập nhật** thì thông tin của sinh viên sẽ được cập nhật vào danh sách sinh viên.

5. Khi chọn 1 dòng trên danh sách sinh viên và nhấp nút > thì thông tin của sinh viên sẽ được thêm vào Danh sách sinh viên nhận học bổng.

6. Khi chọn 1 dòng trên danh sách sinh viên đã chọn và nhấp nút < thì thông tin sinh viên sẽ được xóa khỏi danh sách sinh viên nhận học bổng.

7. Khi người dùng nhập Mã sinh viên hoặc họ tên sinh viên vào Textbox Tìm kiếm và nhấp nút **Tìm kiếm** nếu có dữ liệu trong danh sách sinh viên thì dòng dữ liệu này sẽ được sáng lên (đổi màu chữ hoặc màu nền).

Bài 17. Thiết kế chương trình “Chọn thực đơn món ăn” như sau:

Loại món ăn:

Việt Nam

Chi tiết món ăn:

- ☒ Gỏi cuốn
- ☒ Bánh lan
- ☒ Bún thịt nướng
- ☐ Gà hấp chao

Số lượng:

3

Chọn

Bỏ chọn

Các món ăn đã chọn:

Tên món ăn	Số lượng
Gỏi cuốn	3
Bánh lan	3
Bún thịt nướng	3

Tạo mới

Thoát

Mô tả:

- Combobox **cbLoaiMonAn** chứa danh sách các loại món ăn: Trung Quốc, Việt Nam, Hàn Quốc

- CheckedListBox **listChiTietMonAn** chứa danh sách các món ăn của từng loại món ăn trên

Khi chọn **Trung Quốc** sẽ hiển thị danh sách các món ăn: **Mì vịt tiềm, Mì xào giòn, Bánh canh vịt, Hột vịt lộn**

Khi chọn **Việt Nam** sẽ hiển thị danh sách các món ăn: **Gỏi cuốn, Bánh Flan, Bún thịt nướng, Gà hấp chao.**

- Khi chọn **Hàn Quốc** sẽ hiển thị danh sách các món ăn: **Su si, Canh rong biển, Kim chi, Cơm trộn**

- ListView **dsMonAnChon** là danh sách các món ăn được chọn ở CheckedListBox

- NumericUpDown là số lượng đặt chọn cho món ăn đã chọn có giá trị mặc định từ 1→100.

Yêu cầu:

- Ban đầu cho Enabled **Chi tiết món ăn**. Khi người dùng chọn loại món ăn trong Combobox **LoaiMonAn**, danh sách các món ăn chi tiết sẽ được hiển thị ở dưới CheckedListBox **ChiTietMonAn**.

- Người dùng chọn **1 hoặc nhiều** món ăn ở CheckedListBox **ChiTietMonAn**, gõ vào số lượng (mặc định là 1), sau đó nhấn **Button btChon**, các món ăn này sẽ được thêm vào trong ListView **dsMonAnChon**.

- Khi nhấn button **Chọn**, nếu món ăn đã có trong danh sách thì sẽ cập nhật lại số lượng của món ăn đó với giá trị số lượng mới (không thêm dữ liệu trùng).

- Nếu người dùng chọn **1 hoặc nhiều** món ăn ở **dsMonAnChon**, sau đó nhấn Button **Bỏ Chọn** số lượng sẽ giảm đi **-1**, nếu sau khi **-1** mà số lượng **=0** món ăn này sẽ loại khỏi danh sách **dsMonAnChon**.

- Khi nhấn Button **Tạo Mới**, sẽ xóa toàn bộ món ăn ở **DSMonAnChon** (trước khi oMoi, hỏi người dùng có chắc chắn chưa).

- Khi nhấn **Kết thúc**, kết thúc xử lý Form.

Bài 18. Thiết kế giao diện như hình bên dưới:

Học phần	Tác giả	Đơn giá
Lập trình giao diện	Phương Linh	35000
Mạng máy tính	Minh Khánh	45000
Cơ sở dữ liệu	Thiên Trang	30000

Tổng thành tiền là: 110000

Thể loại sách	Học phần	Tác giả	Đơn giá
Tin học	Lập trình giao diện	Phương Linh	35000
	Mạng máy tính	Minh Khánh	45000
	Cơ sở dữ liệu	Thiên Trang	30000
Thiếu nhi	Tấm Cám	Chuyện cổ tích	25000
	Thánh Gióng	Chuyện cổ tích	40000

Yêu cầu:

1. Thiết kế giao diện.
2. Load dữ liệu như hình lên TreeView.

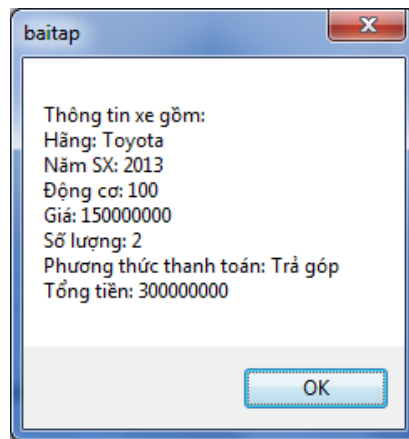
3. Khi chọn một thể loại sách sẽ hiển thị thông tin chi tiết các quyền sách của thể loại như hình bên dưới và tính tổng thành tiền tương ứng.

Bài 19. Thiết kế giao diện như sau:

Mô tả chi tiết của các phương tiện

Tên	Hãng	Năm SX	Động cơ	Giá
SH	Ducati	2012	125	40000000
	Ducati	2013	125	42000000
Spacy	Yamaha	2014	150	52000000
Innova	Toyo	2013	100	150000000
Camry	Huyndai	2012	300	200000000
	Huyndai	2015	400	500000000

Khi người sử dụng chọn tên một xe trên TreeView thì thông tin chi tiết của xe đó sẽ hiển thị trên ListView, nhập vào *số lượng*, chọn *Phương thức thanh toán* và nhấn nút *Đặt hàng* sẽ xuất hiện hộp thoại xác nhận thông tin như sau:



Bài 20. Thiết kế giao diện như hình bên dưới:

Mã SV	Họ tên	Lớp
018301001	Nguyễn Hồng Phấn	CĐ CNTT18
018301002	Trần Thị Thúy Sang	CĐ CNTT18
018301003	Trần Ngọc Yến Như	CĐ CNTT18

Yêu cầu:

1. Thiết kế giao diện.
2. Thiết kế TreeView gồm các thông tin như hình trên.
3. Khi nhấp chuột chọn **SV Khoa CNTT** trên TreeView, ListView Danh sách sinh viên sẽ hiển thị thông tin sinh viên như sau:

Mã SV	Họ tên sinh viên	Lớp
017301001	Nguyễn Hoàng Giang	CĐ CNTT17
017301002	Nguyễn Minh Tuấn	CĐ CNTT17

018307001	Nguyễn Hồng Phấn	CĐ CNTT18
018307002	Trần Thị Thúy Sang	CĐ CNTT18
018307003	Trần Ngọc Yến Như	CĐ CNTT18

4. Khi nhấp chuột chọn **CĐ CNTT17** trên TreeView, ListView Danh sách sinh viên sẽ hiển thị thông tin sinh viên như sau:

Mã SV	Họ tên sinh viên	Lớp
017301001	Nguyễn Hoàng Giang	CĐ CNTT17
017301002	Nguyễn Minh Tuấn	CĐ CNTT17

5. Khi nhấp chuột chọn **CĐ CNTT18** trên TreeView, ListView Danh sách sinh viên sẽ hiển thị thông tin sinh viên như sau:

Mã SV	Họ tên sinh viên	Lớp
018307001	Nguyễn Hồng Phấn	CĐ CNTT18
018307002	Trần Thị Thúy Sang	CĐ CNTT18
018307003	Trần Ngọc Yến Như	CĐ CNTT18

6. Khi chọn 1 mã sinh viên trên TreeView thì thông tin chi tiết của sinh viên sẽ được hiển thị trên các TextBox tương ứng.

7. Khi người dùng chỉnh sửa thông tin trên các TextBox và nhấp nút **Cập nhật** thì thông tin của sinh viên sẽ được cập nhật vào TreeView và ListView.

8. Khi người dùng chọn 1 mã sinh viên trên TreeView và nhấp nút **Hủy** thì sinh viên sẽ được xóa khỏi TreeView, có cảnh báo người dùng trước khi xóa.

Bài 21. Thiết kế giao diện như hình dưới:

Phát sinh ma trận vuông

10

Phát sinh ma trận

Thoát

Tính tổng các phần tử trên đường chéo chính và chéo phụ

Tổng các phần tử trên đường chéo chính:

Tổng các phần tử trên đường chéo phụ:

Tính

Ô NumericUpDown cho phép nhập vào số dòng và cột của ma trận vuông (1→10). Sau đó người dùng nhấn nút **Phát sinh** sẽ phát sinh ma trận vuông tương ứng (mỗi ô là một TextBox). Người dùng nhập vào các giá trị số nguyên vào ma trận. Nhấn nút **Tính** chương trình sẽ tính tổng các số nguyên trên đường chéo chính và đường chéo phụ.

Phát sinh ma trận vuông

10

Phát sinh ma trận

Thoát

Tính tổng các phần tử trên đường chéo chính và chéo phụ

Tổng các phần tử trên đường chéo chính:

Tổng các phần tử trên đường chéo phụ:

Tính