

Chương 4: Giao diện người dùng (User Interface)

Gv: Đặng Hữu Nghị

Nội dung

4.1. Giới thiệu

4.1.1. Ứng dụng Windows Forms

4.1.2. Thanh công cụ (Toolbox)

4.2. Biểu mẫu (Form)

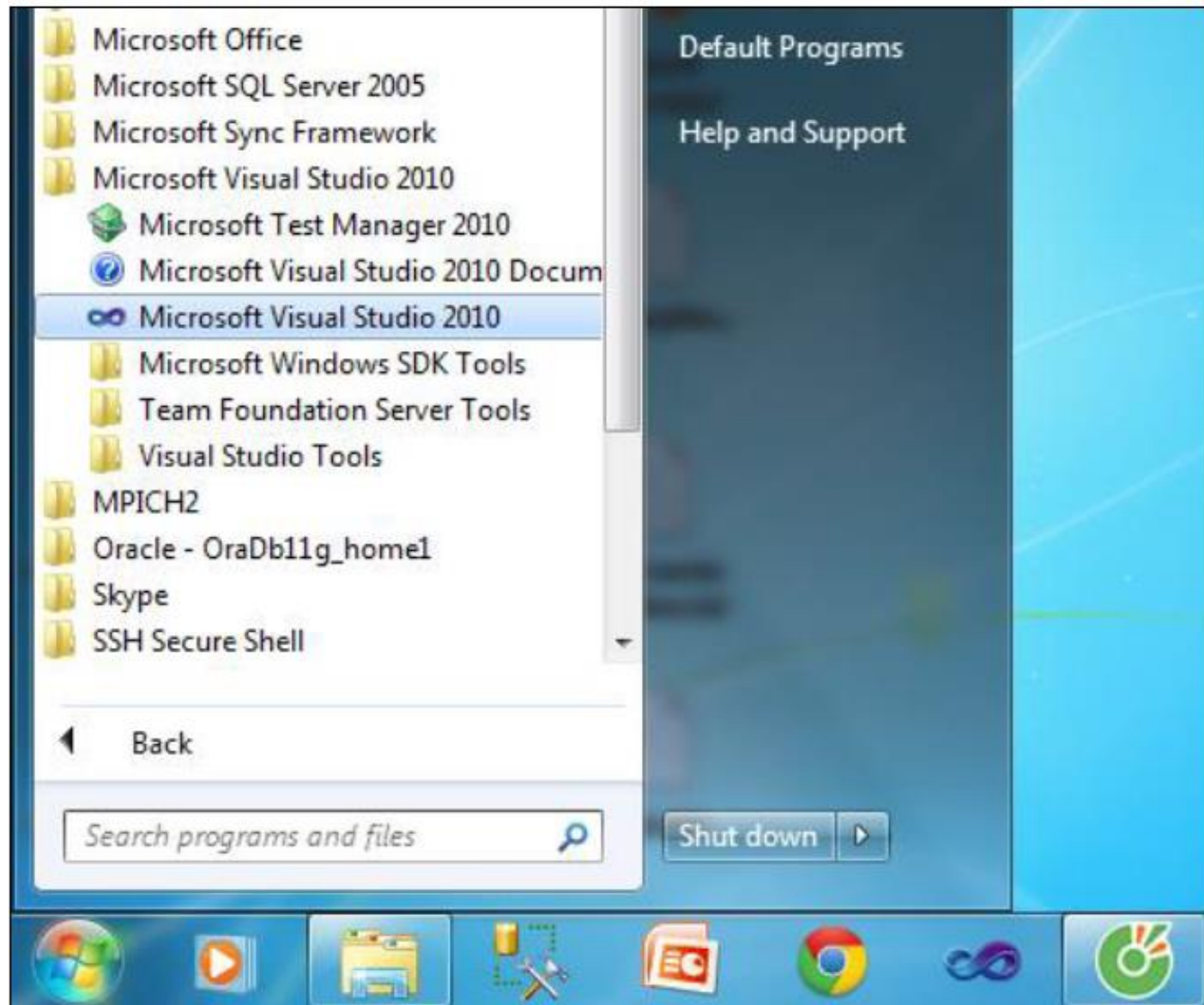
4.3. Các điều khiển thông thường

4.4. Các điều khiển đặc biệt

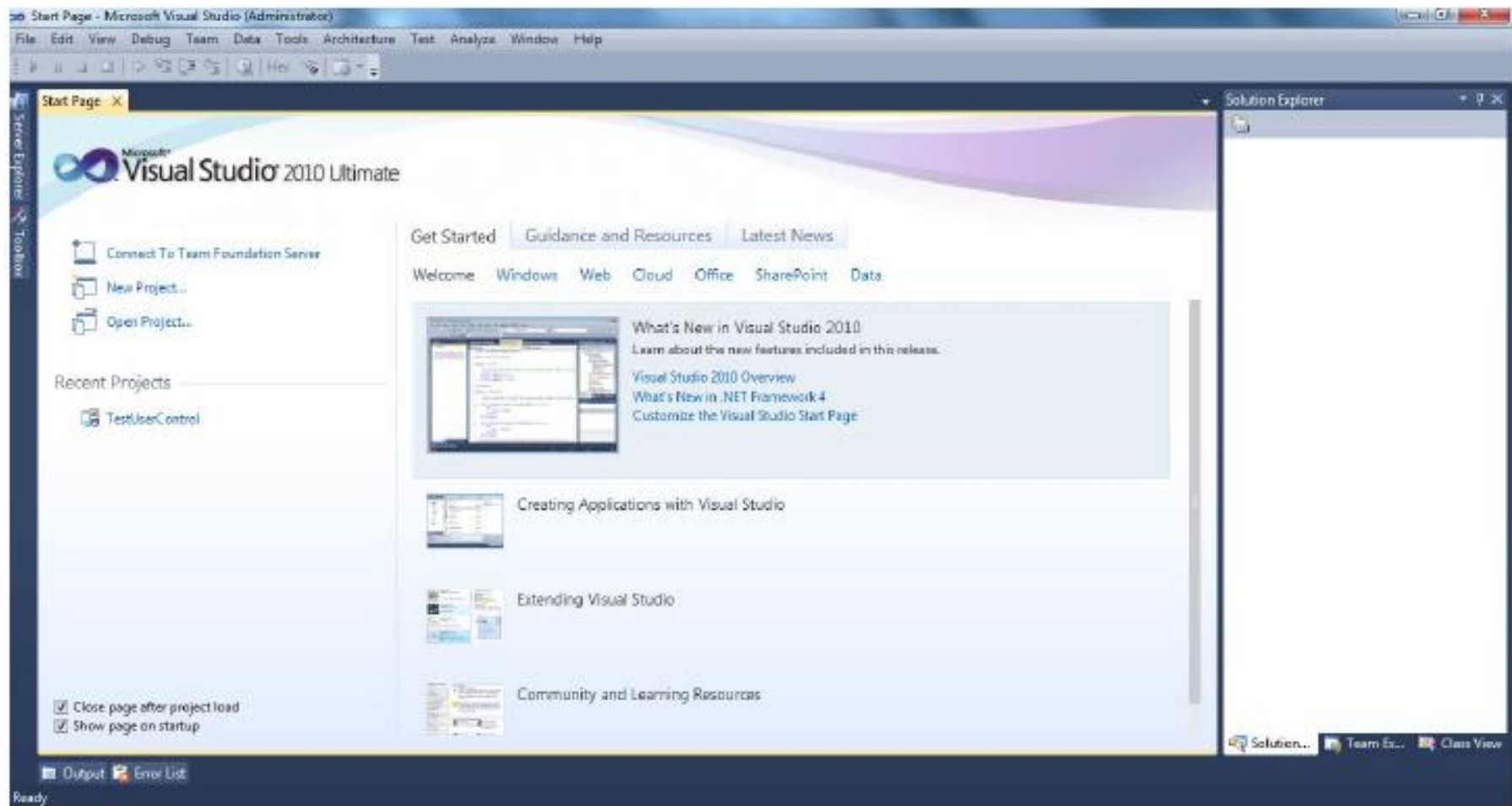
4.1.1. Ứng dụng Windows Forms

- Windows Forms cho phép tạo các form một cách dễ dàng, đồng thời hỗ trợ nhiều điều khiển (Controls) hoặc thành phần (Components)
- Có thể xây dựng, tùy chỉnh các giao diện form một cách nhanh chóng
- Để tạo một dự án Windows Forms với C#, cần thực hiện một số bước sau :

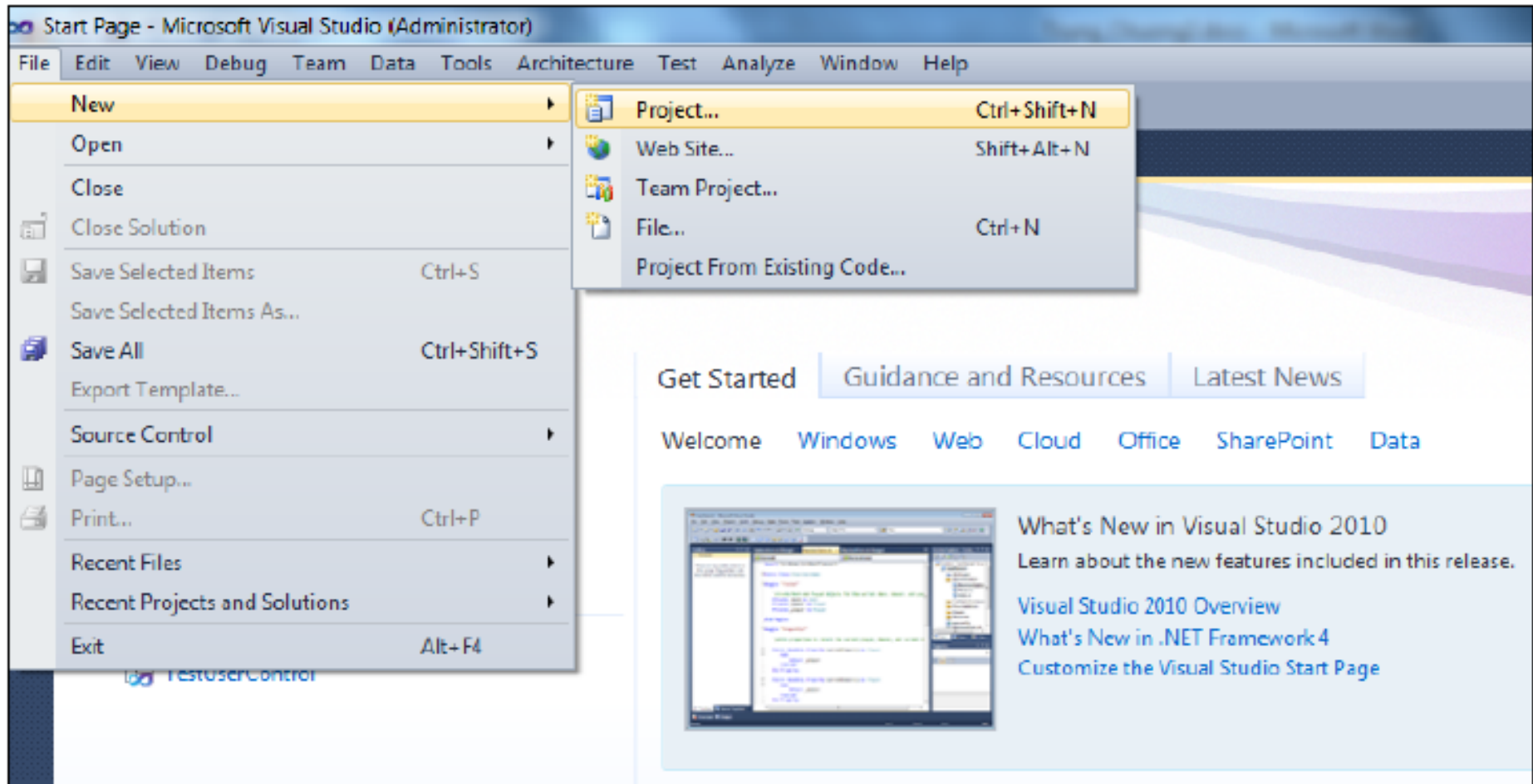
Bước 1: Khởi động ứng dụng Visual Studio 2010 lên bằng cách nhấp vào biểu tượng như hình



Bước 2: Sau khi đã chọn biểu tượng Visual Studio 2010 giao diện như hình sẽ được hiển thị:

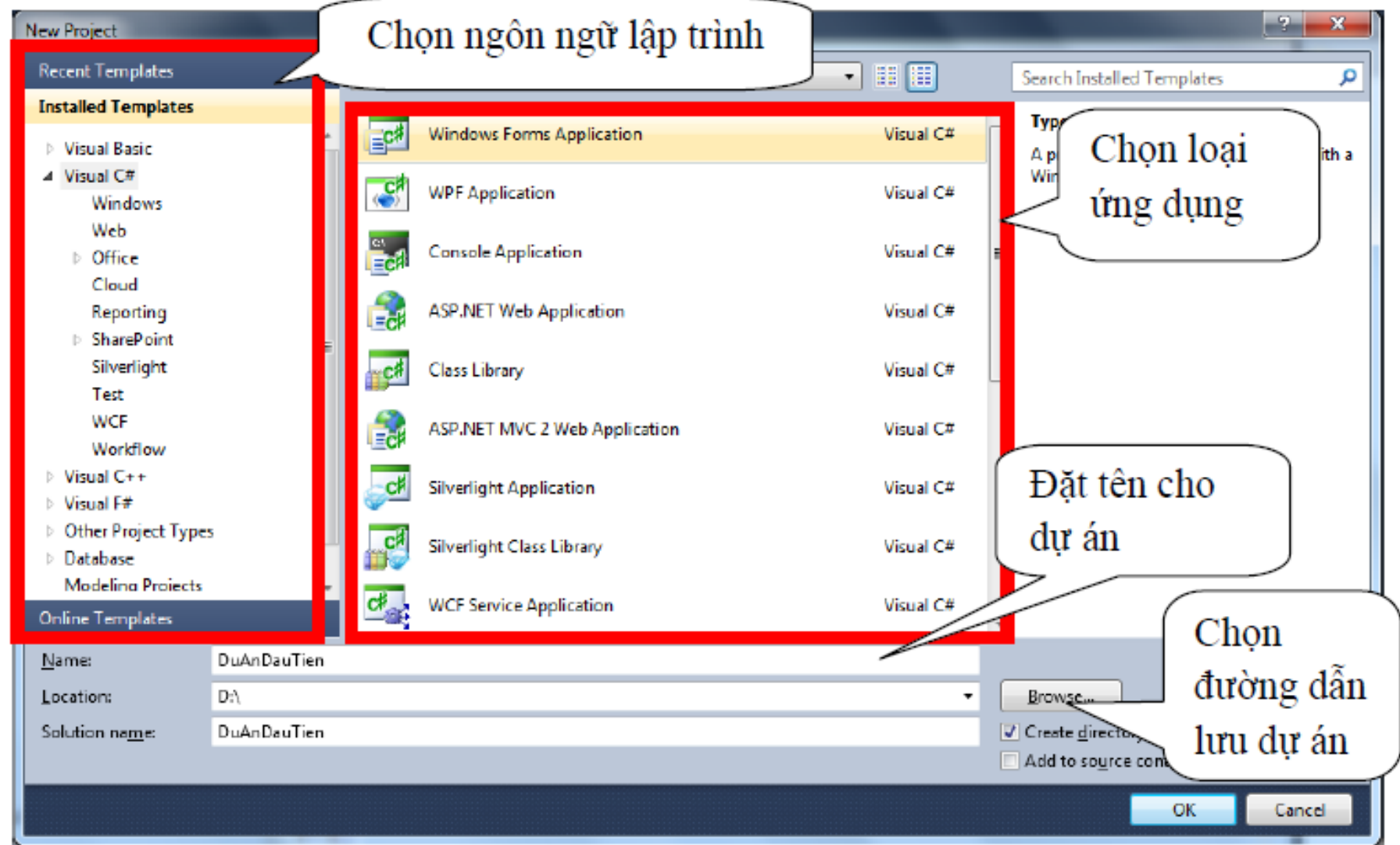


Bước 3: Chọn Menu File > New > Project

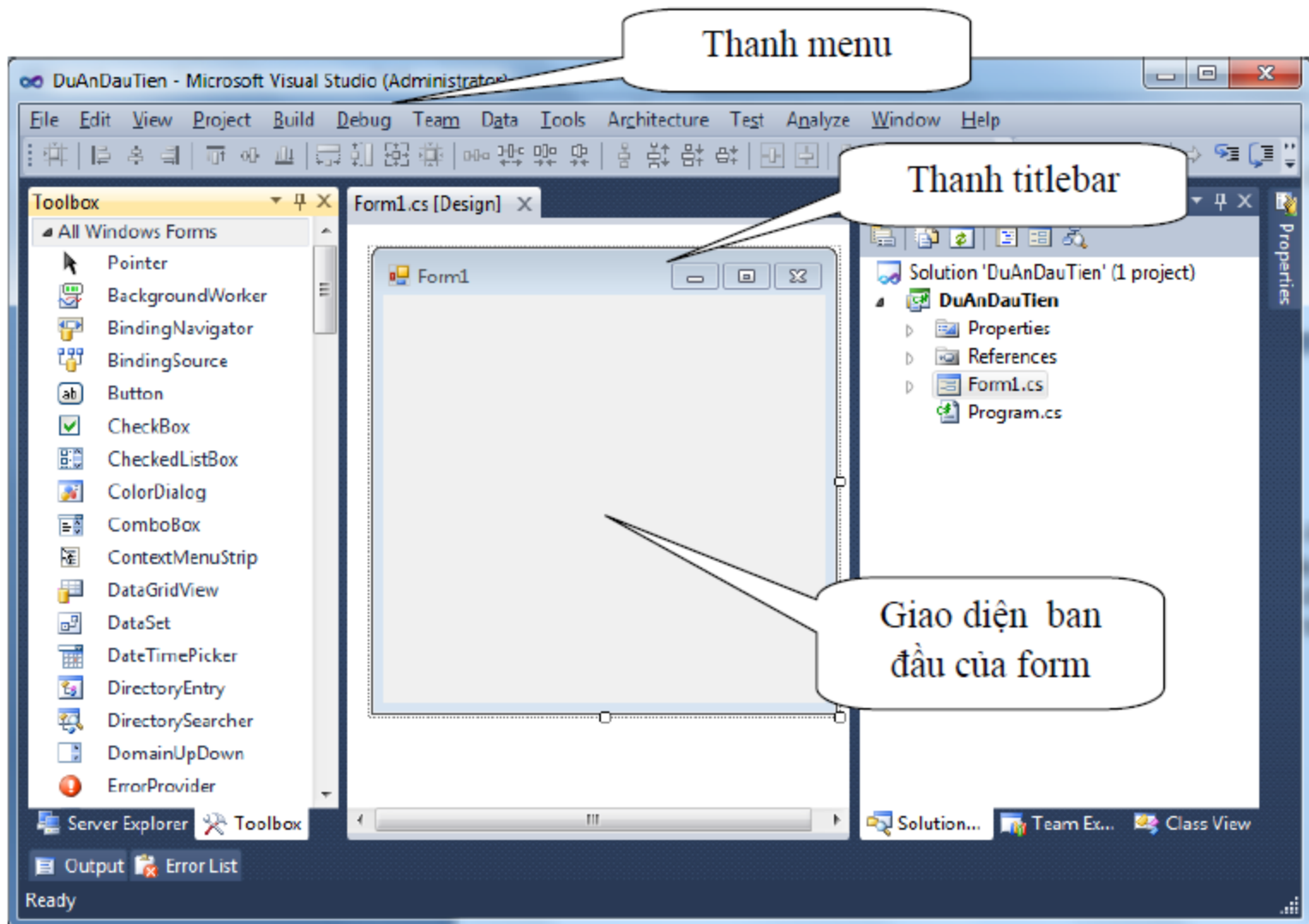


4.1.1. Ứng dụng Windows Forms

- Bước 4:
 - ✓ Sau khi thực hiện xong bước 3, cửa sổ New Project sẽ xuất hiện, tại cửa sổ này lập trình viên có thể chọn ngôn ngữ lập trình sử dụng và chọn loại ứng dụng muốn viết.
 - ✓ Cụ thể ở đây chọn ngôn ngữ Visual C#, và loại ứng dụng là Windows Forms.
 - ✓ Sau khi đã chọn xong loại ứng dụng đã viết, lập trình viên cần đặt tên cho dự án (nếu không muốn sử dụng tên mặc định của Visual Studio) và đường dẫn sẽ lưu dự án như hình:

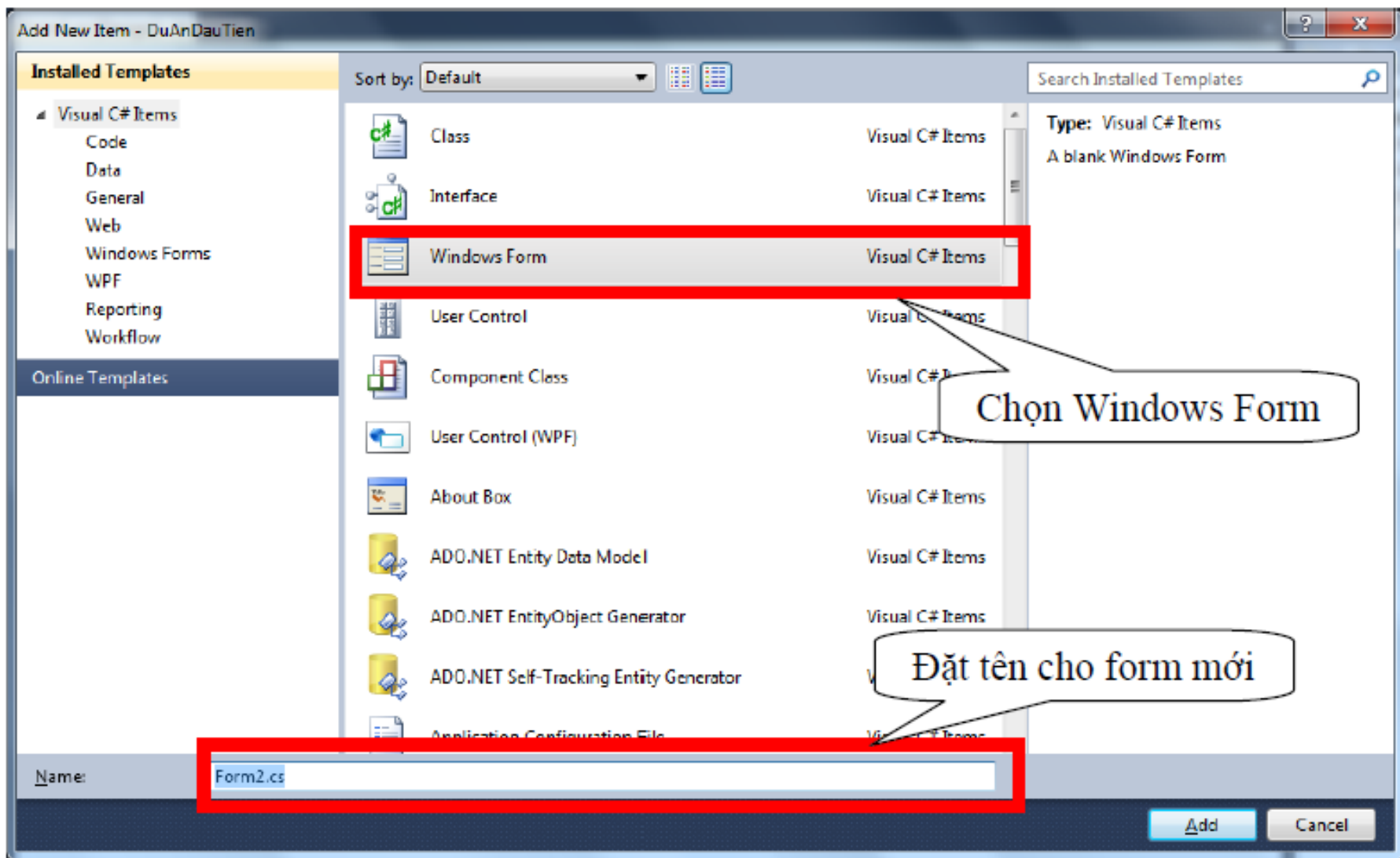


Sau khi đã tạo xong dự án, một form mới có tên Form1 mặc định sẽ được thêm vào dự án vừa tạo như hình



4.1.1. Ứng dụng Windows Forms

- Thêm form mới vào dự án:
 - ✓ Phần lớn các dự án phần mềm đều chứa rất nhiều form
 - ✓ Để thêm một form mới vào dự án tại thời điểm thiết kế, trên menu Project, chọn Add Windows Form
 - ✓ Một hộp thoại Add New Item được mở ra. Sau đó chọn mục Windows Form và đặt tên cho form mới như hình



4.1.1. Ứng dụng Windows Forms

- Khái báo đối tượng thuộc lớp Form như sau:

```
Form form1;
```

```
form1 = new Form();
```

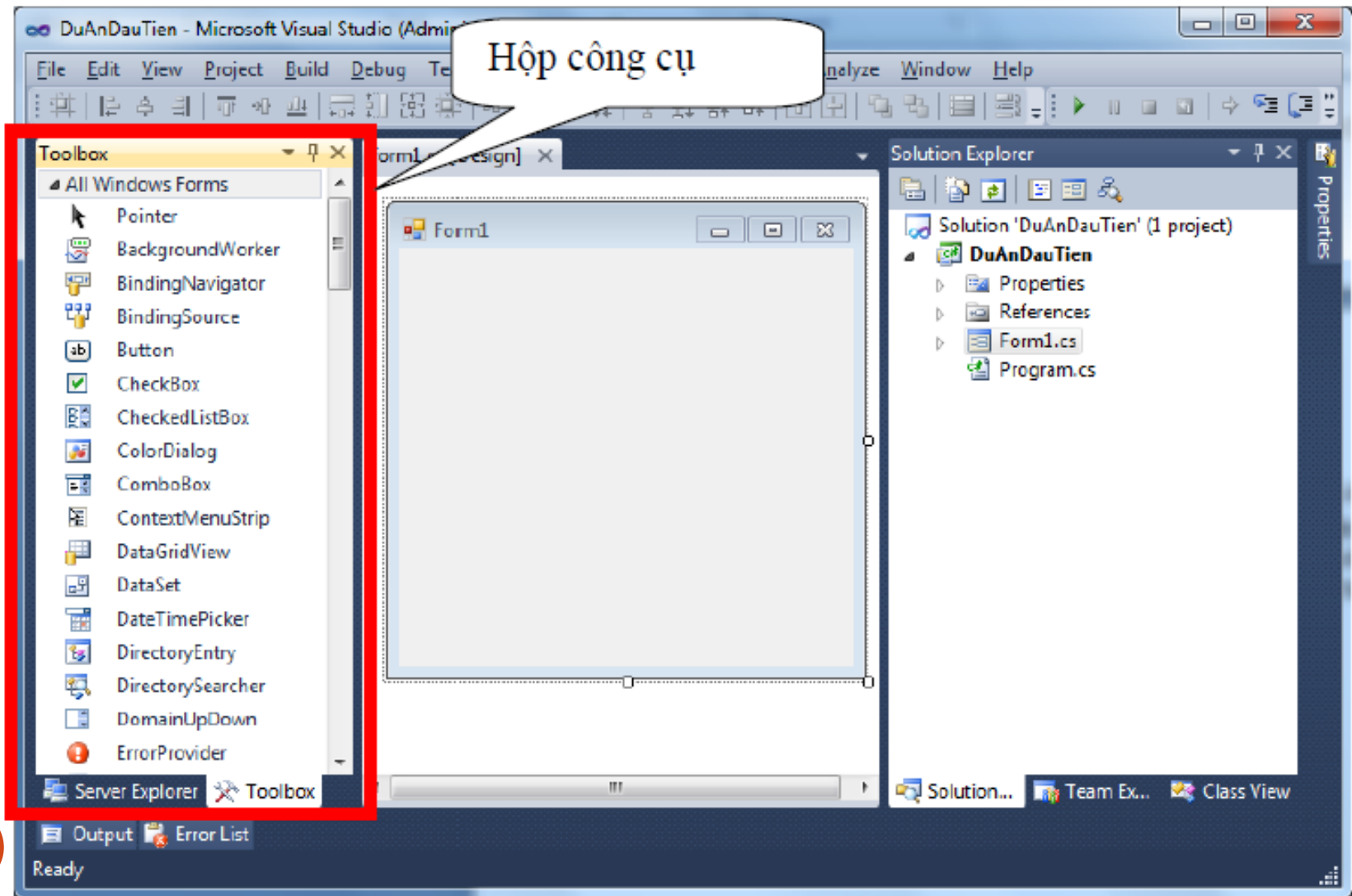
- Hiển thị Form khi tại thời điểm chương trình thực thi cần sử dụng đến phương thức Show() hoặc ShowDialog():

```
form1.Show();
```

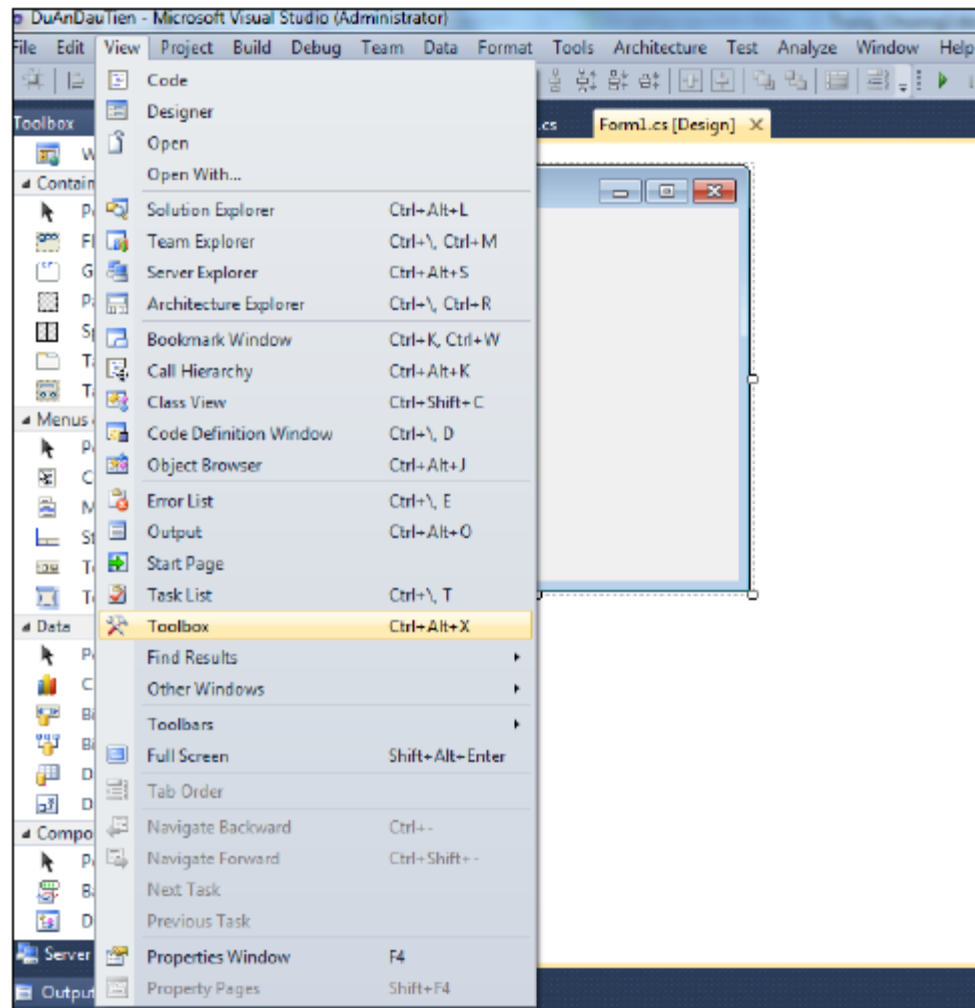
4.1.2. Thanh công cụ (Toolbox)

- Cung cấp danh sách các Component/ Control được liệt kê theo nhóm, cho phép lập trình viên sử dụng thao tác kéo thả vào form để thiết kế giao diện chương trình
- Ngoài những điều khiển hiển thị mặc định trong thanh công cụ, lập trình viên cũng có thể thêm các thành phần mới vào bằng các chọn *Project/AddReference*.

4.1.2. Thanh công cụ (Toolbox)



Nếu giao diện chính không hiển thị thanh công cụ, lập trình viên có thể hiển thị bằng cách chọn *View/ Toolbox* như hình



Thanh công cụ bố trí các điều khiển thành những nhóm riêng biệt như hình

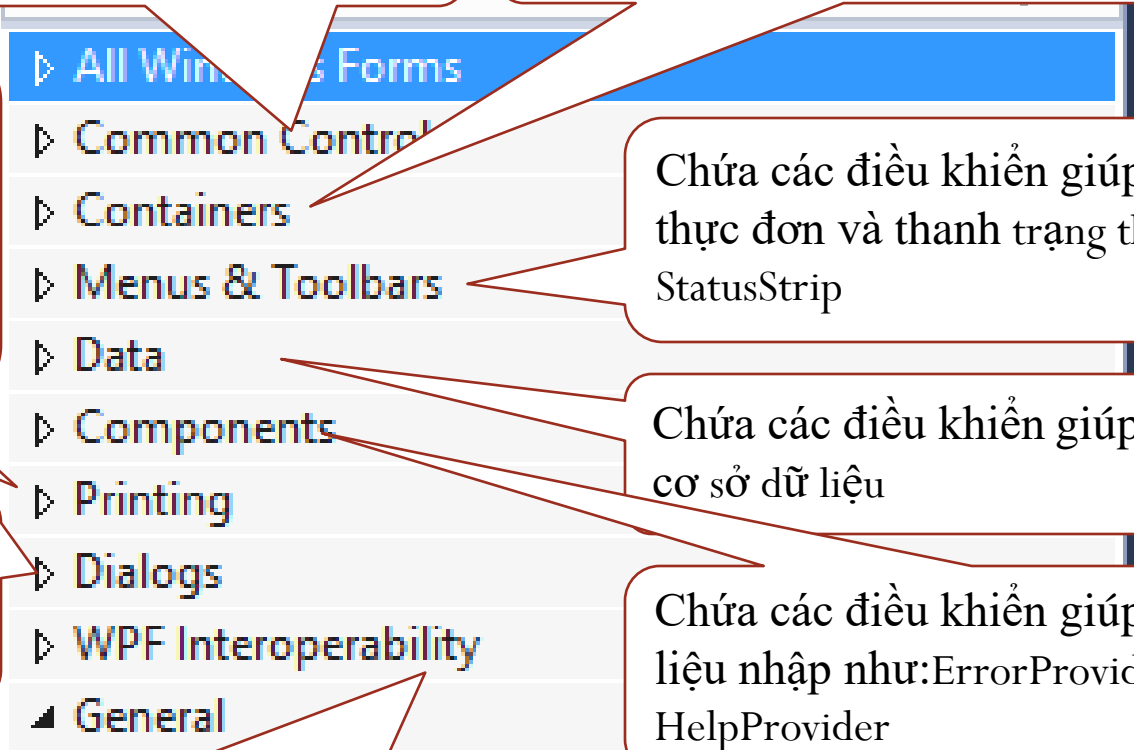
Chứa các điều khiển thông dụng như: TextBox, Label, ..

Chứa các điều khiển giúp trình bày các điều khiển khác trên form

Chứa các điều khiển giúp làm việc với việc in ấn: PrintDialog, PrintReviewDialog..

Chứa các điều khiển làm việc với tập tin..

Chứa điều khiển cho phép đặt điều khiển của WPF trong cửa sổ Windows Form: ElementHost



▶ All Windows Forms
▶ Common Controls
▶ Containers
▶ Menus & Toolbars
▶ Data
▶ Components
▶ Printing
▶ Dialogs
▶ WPF Interoperability
▲ General

Chứa các điều khiển giúp tạo thanh thực đơn và thanh trạng thái: MenuStrip, StatusStrip

Chứa các điều khiển giúp làm việc với cơ sở dữ liệu

Chứa các điều khiển giúp kiểm tra dữ liệu nhập như: ErrorProvider, HelpProvider

4.2. Biểu mẫu (Form)

4.2.1. Thuộc tính Form

4.2.2. Các loại Form

4.2.3. Biến cố của Form

4.2.4. Phương thức

4.2.1. Thuộc tính của Form

- Thiết lập giá trị thuộc tính form trong cửa sổ Properties
- Windows Forms hỗ trợ nhiều thuộc tính cho việc tùy chỉnh form.
- Ta có thể thay đổi giá trị các thuộc tính này trong cửa sổ Properties như hình sau để điều chỉnh kích thước, màu sắc, font chữ, ... của form cho phù hợp

4.2.1. Thuộc tính của Form

The screenshot displays the Microsoft Visual Studio IDE with a project named 'WindowsFormsApplication1'. The main design area shows a 'Form1' window. A red callout bubble with the text 'Cửa sổ Properties' (Properties window) points to the Properties window on the right. The Properties window is highlighted with a red rectangle and shows the 'Form1' object with the following categories and properties:

- Accessibility**
 - AccessibleDescription
 - AccessibleName
 - AccessibleRole: Default
- Appearance**
 - BackColor: Control
- Text**
 - The text associated with the control.

At the bottom left, there is a red circle containing the number 19.

4.2.1. Thuộc tính của Form

Thuộc tính	Mô tả
<i>Name</i>	Đặt tên form
<i>AcceptButton</i>	Giá trị thuộc tính này nhận là tên của một button trên form. Khi đó thay vì nhấp chuột vào button để thực thi thì người dùng có thể ấn phím Enter trên bàn phím
<i>BackColor</i>	Thiết lập màu nền của form
<i>BackgroundImage</i>	Thiết lập hình nền cho form
<i>BackgroundImageLayout</i>	Thiết lập việc hiển thị hình vừa thêm trong thuộc tính <i>BackgroundImage</i> sẽ hiển thị trên form ở dạng: bình thường (None), giữa (Center), ...
<i>CancelButton</i>	Giá trị thuộc tính này nhận là tên của một button trên form. Khi đó thay vì nhấp chuột vào button để thực thi thì người dùng có thể ấn phím Escape trên bàn phím

4.2.1. Thuộc tính của Form

Thuộc tính	Mô tả
<i>ControlBox</i>	Mang giá trị true hoặc false. Nếu thiết lập thuộc tính là false thì sẽ loại bỏ các nút minimize và nút maximize trên form
<i>Cursor</i>	Thiết lập hình dạng con trỏ khi di chuyển con trỏ vào form
<i>Enable</i>	Mang giá trị true hoặc false; Nếu thiết lập thuộc tính là false thì điều khiển trong form sẽ không cho phép người dùng thao tác.
<i>Font</i>	Thiết lập văn bản hiển thị trên điều khiển
<i>ForeColor</i>	Thiết lập màu mặc định cho chuỗi của các điều khiển trên form
<i>FormBorderStyle</i>	Thiết lập đường viền của form và hành vi của form khi chạy chương trình

4.2.1. Thuộc tính của Form

Thuộc tính	Mô tả
<i>HelpButton</i>	Mang giá trị true hoặc false; Nếu thiết lập thuộc tính là true thì trên thanh titlebar sẽ hiện 1 nút có dấu ? (nút này chỉ hiện khi hai thuộc tính MinimizeBox và MaximizeBox được thiết lập là false)
<i>Icon</i>	Biểu tượng hiển thị bên trái trên thanh titlebar của form
<i>KeyReview</i>	Mang giá trị true hoặc false: nếu thiết lập thuộc tính là true cho phép các sự kiện bàn phím của form có hiệu lực
<i>Location</i>	Khi thuộc tính StartPosition được thiết lập là Manual, thì thuộc tính Location có tác dụng thiết lập vị trí hiển thị của form trên màn hình
<i>MaximizeBox</i>	Mang giá trị true hoặc false: nếu thiết lập thuộc tính là false thì nút maximize form trên thanh titlebar sẽ mất đi

4.2.1. Thuộc tính của Form

Thuộc tính	Mô tả
<i>MaximumSize</i>	Thiết lập kích thước lớn nhất của form (chiều rộng x chiều cao)
<i>MinimizeBox</i>	Mang giá trị true hoặc false: nếu thiết lập thuộc tính là false thì nút minimize form trên thanh titlebar sẽ mất đi
<i>MinimumSize</i>	Thiết lập kích thước nhỏ nhất của form (chiều rộng x chiều cao)
<i>Opacity</i>	Thiết lập độ trong suốt cho form
<i>Size</i>	Kích thước form
<i>StartPosition</i>	Vị trí hiển thị của form trên màn hình
<i>Text</i>	Chuỗi văn bản hiển thị trên titlebar của form
<i>TopMost</i>	Mang giá trị true hoặc false: nếu thiết lập thuộc tính là true thì form sẽ luôn hiển thị trên các cửa sổ khác

4.2.1. Thuộc tính của Form

Thuộc tính	Mô tả
<i>Visible</i>	Mang giá trị true hoặc false: nếu thiết lập thuộc tính là true thì form sẽ được hiển thị trên màn hình, nếu là false sẽ không hiển thị trên màn hình
<i>WindowState</i>	Có 3 giá trị: <ul style="list-style-type: none">- Normal: hiển thị form bình thường;- Minimized: khi chạy chương trình form sẽ bị thu nhỏ dưới thanh taskbar;- Maximized: form hiển thị có kích thước đầy màn hình
<i>IsMDIContainer</i>	Mang giá trị True hoặc False. <ul style="list-style-type: none">- Nếu là True: Form ở dạng MDI Form- Nếu là False: Form ở dạng bình thường
<i>MdiParent</i>	mang giá trị là đối tượng MDI Form. Khi thiết lập giá trị cho thuộc tính MdiParent thì form sẽ trở thành Child Form

4.2.1. Thuộc tính của Form

- Thiết lập giá trị thuộc tính form bằng mã lệnh:
 - ✓ Ngoài việc thiết kế form bằng viết thiết lập các giá trị trong cửa sổ properties. Ta cũng có thể thiết lập giá trị thuộc tính của form bằng mã lệnh như sau:
 - ✓ Thay đổi chuỗi văn bản hiển thị trên titlebar bằng cách thiết lập giá trị cho thuộc tính *Text*:

Form1.Text = “Đây là Form1”;

- ✓ Thiết lập kích thước form: Có thể thiết lập giá trị cho thuộc tính *Width* và thuộc tính *Height* để quy định chiều rộng và chiều cao cho form.

Form1.Width = 300;

Form1.Height = 400;

hoặc: Form1.Size = new Size(300, 400);

4.2.1. Thuộc tính của Form

- Thiết lập form hiển thị:
 - ✓ Trong C#, ta có thể thiết lập form hiển thị đầu tiên khi dự án được thực thi bằng cách sửa lại mã lệnh trong hàm main của lớp Program

4.2.1. Thuộc tính của Form

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace DuAnDauTien
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Lớp Program mặc định được tạo ra trong dự án

Các đoạn mã mặc định trong lớp Program

4.2.1. Thuộc tính của Form

- ✓ Lớp Program chứa một hàm bên trong là hàm Main. Khi dự án được thực thi, thì mã lệnh trong hàm Main sẽ thực thi trước. Cụ thể tại dòng:

`Application.Run(new Form1());`

- ✓ Dòng lệnh trên chỉ ra rằng form đầu tiên sẽ hiển thị là Form1. Do đó, ta chỉ cần sửa lại mã lệnh chỉ định một form khác muốn hiển thị bằng cách thay tên form đó cho Form1.

- ✓ Ví dụ hiển thị form có tên Form2 trong dự án

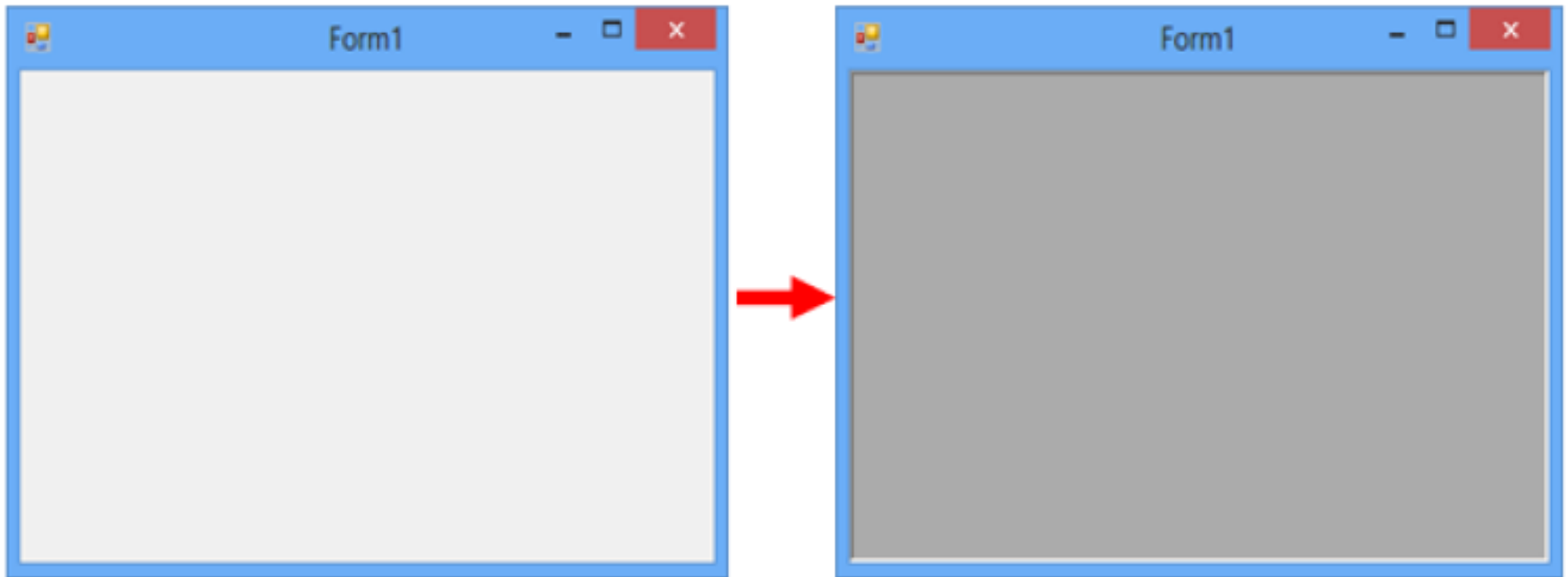
`Application.Run(new Form2());`

4.2.2. Các loại Form

- Trong ứng dụng Windows Forms, có 3 loại form:
 - ✓ form bình thường (Normal Form)
 - ✓ form cha (Mdi Form)
 - ✓ form con (Child Form).
- Mdi Form:
 - ✓ Mdi Form là form có thể chứa các Child Form bên trong. Để làm được công việc đó thì form phải được thiết lập thuộc tính:
IsMdiContainer = True;
 - ✓ Lập trình viên có thể thiết lập thuộc tính *IsMdiContainer* trong cửa sổ Properties trên màn hình thiết kế form, hoặc bằng mã lệnh

4.2.2. Các loại Form

Hình dạng form khi chuyển từ Normal Form sang Mdi Form



4.2.2. Các loại Form

- Child Form:
 - ✓ Child Form là dạng form nằm bên trong vùng làm việc của Mdi Form hay nói cách khác là Child Form được chứa bên trong Mdi Form
 - ✓ Một form muốn trở thành Child Form cần phải khai báo thuộc tính MdiParent có giá trị là đối tượng Mdi Form.
 - ✓ Thuộc tính MdiParent không được biểu diễn trên cửa sổ Properties, do đó ta bắt buộc phải thiết lập giá trị MdiParent bằng mã lệnh

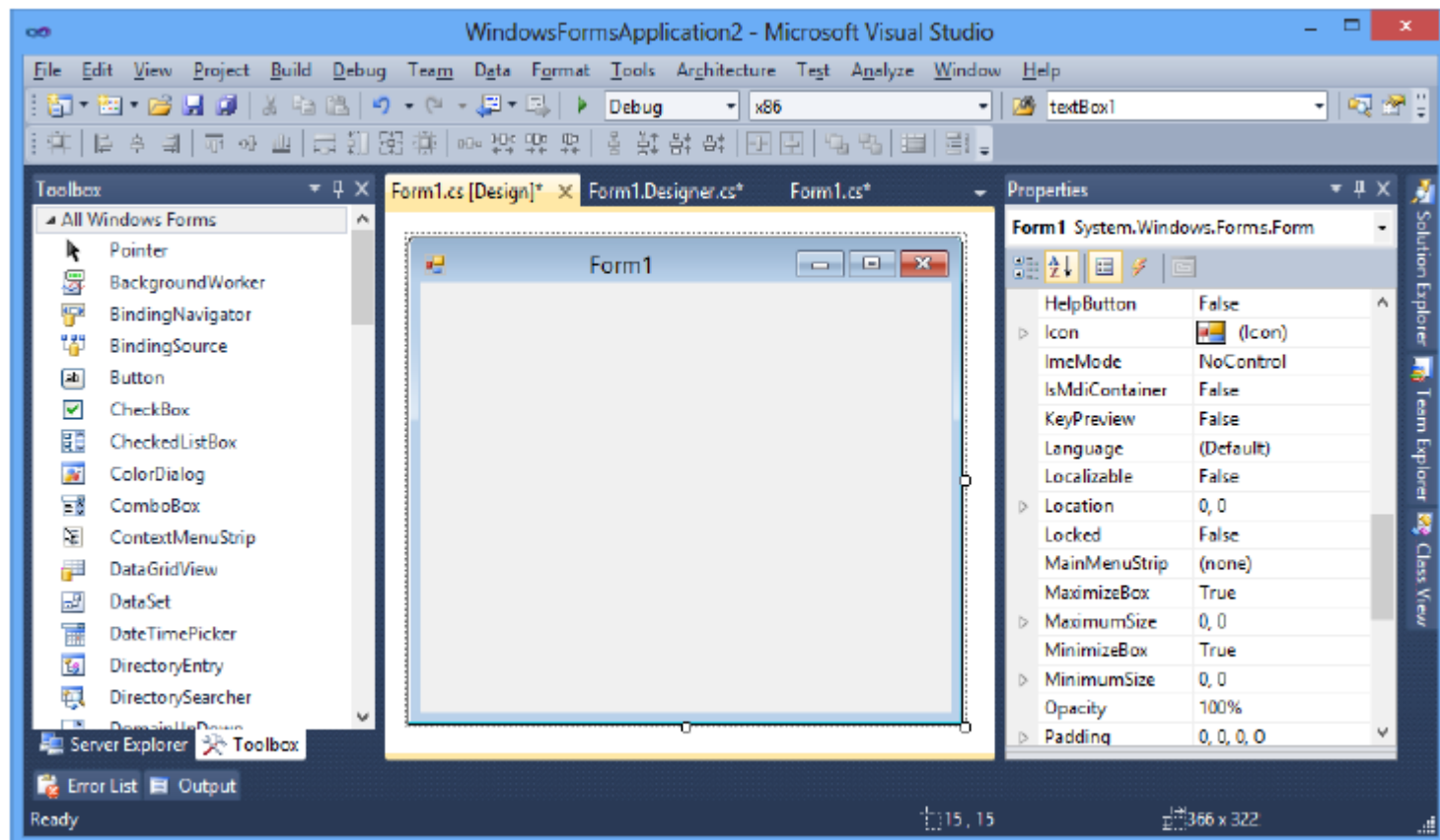
4.2.2. Các loại Form

Ví dụ: Viết chương trình biểu diễn Mdi Form và Child Form như hình



4.2.2. Các loại Form

Bước 1: Tạo một dự án mới bằng C#, loại dự án là Windows Forms Application. Giao diện đầu tiên của chương trình hiển thị Form1



4.2.2. Các loại Form

- Bước 2: Viết mã lệnh cho chương trình Tại sự kiện *Load* của *Form1* viết các dòng lệnh sau:

```
private void Form1_Load(object sender, EventArgs e)  
{  
    this.Text = "Mdi Form";  
    this.IsMdiContainer = true;  
    Form frm = new Form();  
    frm.Text = "Child Form";  
    frm.MdiParent = this;  
    frm.Show();  
}
```

- Bước 3: Nhấn **Ctrl + Shift + B** để biên dịch mã nguồn và nhấn **F5** để thực thi chương trình sẽ được *Mdi Form* và *Child Form*

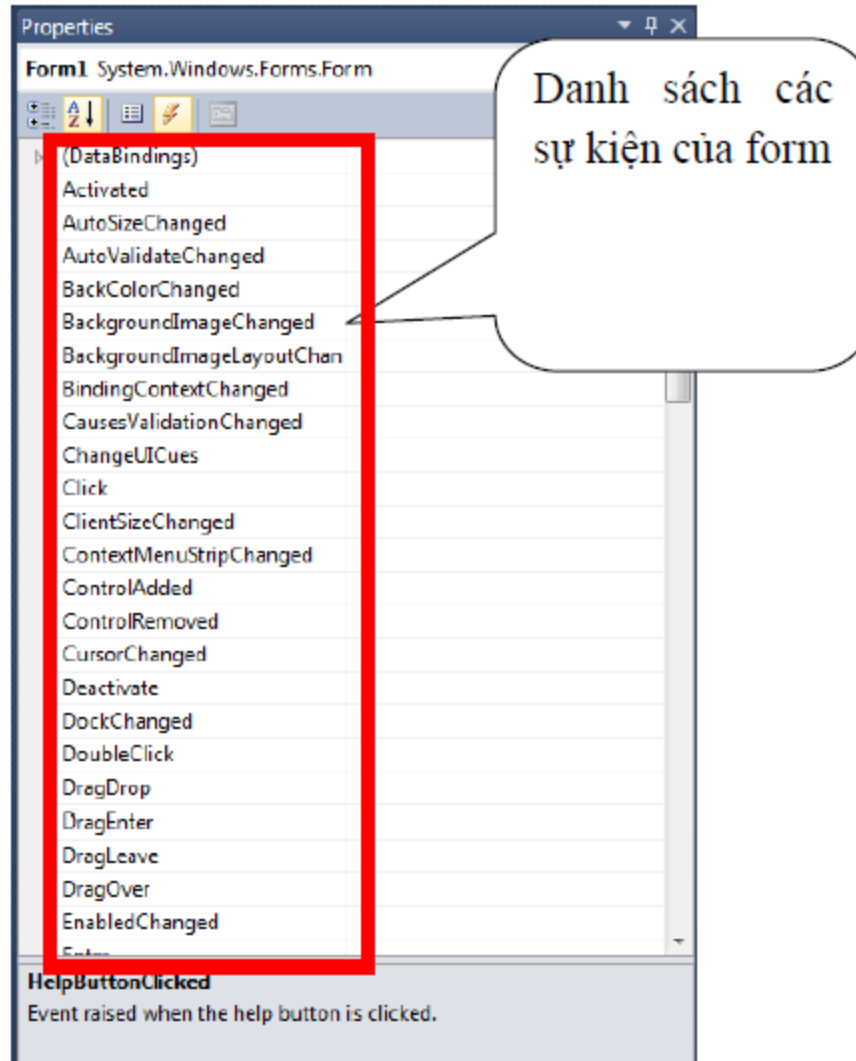
4.2.2. Các loại Form

- Normal Form
 - ✓ Normal Form là form hoạt động độc lập với tất cả các form khác, nghĩa là form sẽ không chứa hoặc được chứa bởi một form nào khác.
 - ✓ Thông thường khi tạo mới dự án Windows Forms Application thì form mặc định được tạo và hiển thị đầu tiên là Normal Form.

4.2.3. Biến cố của Form

- Biến cố của form hay có thể gọi là các hành động hoặc sự kiện liên quan đến form.
- Để sử dụng các sự kiện của form ta thực hiện các thao tác sau:
 - ✓ Bước 1: Nhấp chuột trái vào form
 - ✓ Bước 2: Trên cửa sổ Properties tương ứng, nhấp chuột trái vào biểu tượng để mở hộp thoại các sự kiện như hình

4.2.3. Biến cố của Form



4.2.3. Biến cố của Form

- ✓ Bước 3: Hộp thoại chứa nhiều sự kiện liên quan đến form như: đóng form, mở form, di chuyển chuột vào form, ... ta muốn sử dụng sự kiện nào chỉ cần nhấp đôi chuột trái vào sự kiện đó.
- ✓ Ví dụ: Muốn sử dụng sự kiện nhấp chuột của form chỉ cần nhấp đôi chuột vào sự kiện *Click* thì một phương thức nhấp chuột của form sẽ được tự động phát sinh mã lệnh:

```
private void Form1_Click(object sender, EventArgs e)
{
    //Mã lệnh khi nhấp chuột vào form
}
```

Bảng mô tả các sự kiện của form

Sự kiện	Mô tả
AutoSizeChanged	Xảy ra khi thuộc tính Autotize của Form chuyển từ True sang False hay ngược lại là False sang True
BackColorChanged	Xảy ra khi thuộc tính BackColor của Form thay đổi
Click	Xảy ra khi người dùng Click chuột vào vùng làm việc thuộc Form
ControlAdded	Xảy ra khi một điều khiển được Add vào Form
ControlRemoved	Xảy ra khi một điều khiển bị xóa khỏi Form
CursorChanged	Xảy ra khi thuộc tính Cursor của Form thay đổi
DoubleClick	Xảy ra khi người dùng DoubleClick vào vùng làm việc của Form
FontChanged	Xảy ra khi thuộc tính Font của Form có sự thay đổi
ForeColorChanged	Xảy ra khi thuộc tính ForeColor của Form có sự thay đổi

Bảng mô tả các sự kiện của form

Sự kiện	Mô tả
FormClosed	Xảy ra khi Form đã đóng (Nhấn vào nút X màu đỏ trên titlebar)
FormClosing	Xảy ra khi Form đang đóng (2 sự kiện FormClosed và FormClosing thường dùng trong lập trình CSDL: khi xảy ra sự kiện này thì đóng kết nối CSDL)
KeyDown	Xảy ra khi người dùng nhấn một phím hay một tổ hợp phím
KeyPress	Xảy ra khi người dùng nhấn một phím
KeyUp	Xảy ra khi người dùng nhả một phím
MouseClick	Xảy ra khi người dùng nhấn chuột (một trong 3 lựa chọn: Trái, giữa, phải)
MouseDoubleClick	Xảy ra khi người dùng nhấp đúp chuột vào một vùng làm việc của Form (một trong 3 lựa chọn: Trái, giữa, phải)

Bảng mô tả các sự kiện của form

Sự kiện	Mô tả
MouseDown	Xảy ra khi người dùng nhấn chuột
MouseHover	Xảy ra khi người dùng di chuyển vào các vùng làm việc Form
MouseLeave	Xảy ra khi di chuyển chuột ra khỏi vùng làm việc của Form
MouseMove	Xảy ra khi di chuyển chuột trên một vùng làm việc thuộc Form (nếu Form có chứa một điều khiển nào đó, khi di chuyển chuột trên điều khiển này thì không xảy ra sự kiện MouseMove của Form)
MouseUp	Xảy ra khi người dùng nhả nhấn chuột (có thể là chuột trái, chuột phải, chuột giữa - chuột cuộn)
Move	Xảy ra khi di chuyển Form (có sự thay đổi vị trí của Form)
StyleChanged	Xảy ra khi thuộc tính FormBorderStyle của Form thay đổi
TextChanged	Xảy ra khi thuộc tính Text của Form thay đổi.

4.2.3. Biến cố của Form

- Sự kiện FormClosed: Sự kiện này được gọi khi Form đã đóng

```
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    MessageBox.Show("Sự kiện FormClosed được gọi");
}
```

4.2.3. Biến cố của Form

- Sự kiện FormClosing: Xảy ra khi Form đang đóng

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult kq = MessageBox.Show("Bạn muốn đóng
                                     Form lại không?", "FormClosing",
                                     MessageBoxButtons.YesNo,
                                     MessageBoxIcon.Information);
    if ( kq == DialogResult.Yes)
        e.Cancel = false; // Đóng Form lại
    else
        e.Cancel = true; //Không đóng Form nữa
}
```

4.2.3. Biến cố của Form

- Sự kiện KeyPress: Xảy ra khi ấn phím, nếu không chỉ rõ phím nào được nhấn thì khi nhấn bất cứ phím nào của sự kiện *KeyPress* của form đều xảy ra.

```
private void Form2_KeyPress(object sender, KeyPressEventArgs e)  
{  
    if (e.KeyChar == 'a') //nhấn phím a trên bàn phím  
        MessageBox.Show("Sự kiện KeyPress được gọi");  
}
```

4.2.3. Biến cố của Form

- Sự kiện KeyDown

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    // Nhấn Ctrl+F gọi sự kiện KeyDown
    if (e.KeyCode == Keys.F && e.Modifiers == Keys.Control)
        MessageBox.Show("Sự kiện KeyDown được gọi");
}
```

4.2.3. Biến cố của Form

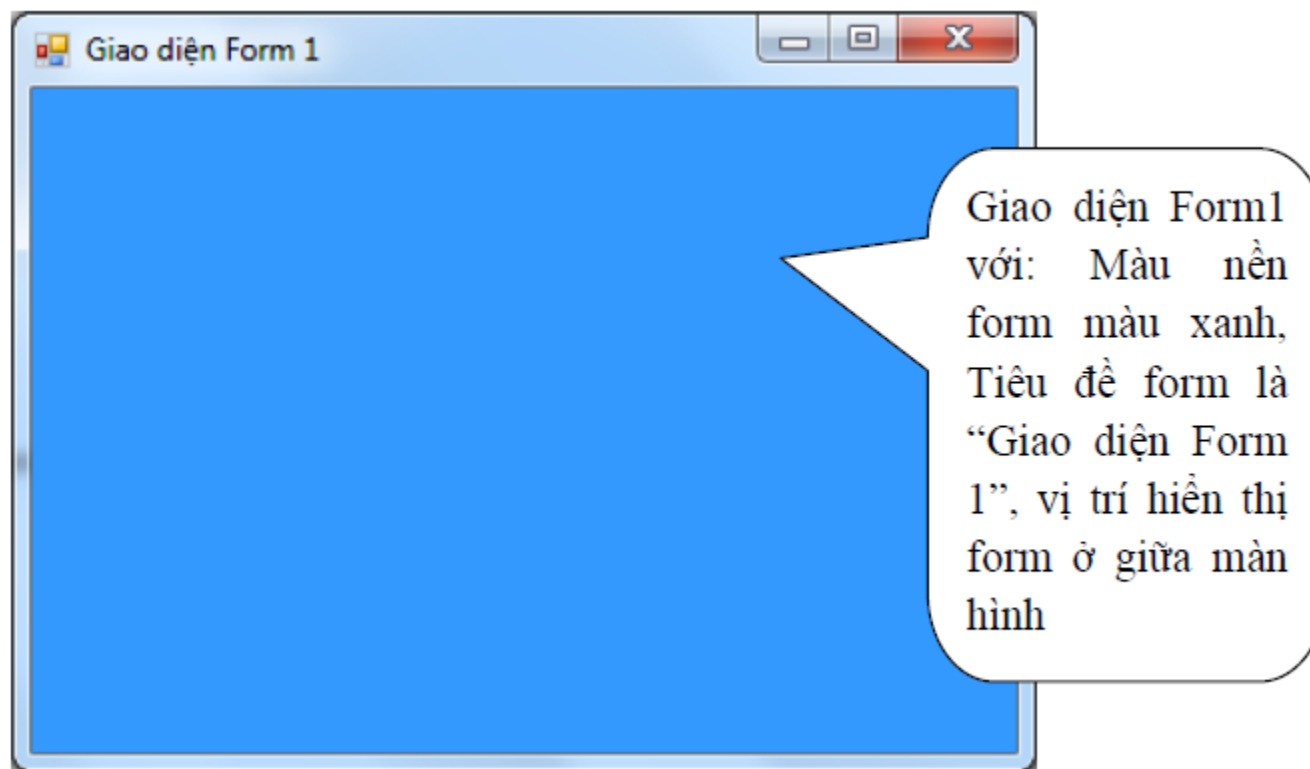
- **Sự kiện MouseClick:** Xảy ra khi nhấn một trong 3 nút chuột trái, giữa hoặc chuột phải

```
private void frmForm_MouseClick(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButton.Left) //nhấn chuột trái
        MessageBox.Show("Bạn đã nhấn chuột trái");
    else
        if (e.Button==MouseButton.Middle)//nhấn chuột giữa
            MessageBox.Show(" Bạn đã nhấn chuột giữa");
        else
            if(e.Button==MouseButton.Right)//nhấn chuột phải
                MessageBox.Show(" Bạn đã nhấn chuột phải");
```

4.2.4. Phương thức

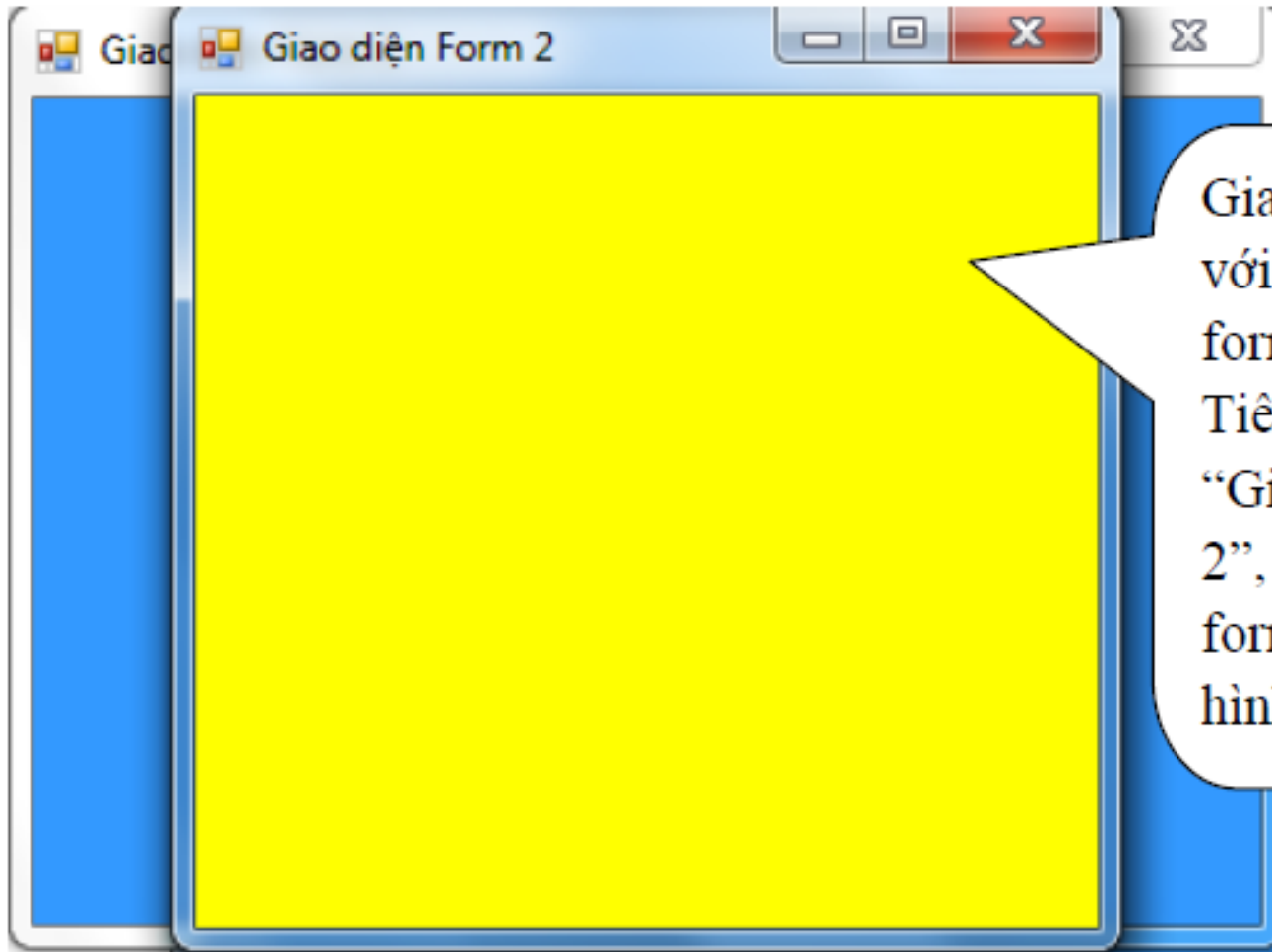
- Một số phương thức thường sử dụng của form: *Show()*, *ShowDialog()*, *Hide()*, *Close()*.
- Phương thức *Show()*: Phương thức *Show()* sử dụng để hiển thị form trên màn hình.
- Ví dụ: Thiết kế chương trình như hình

4.2.4. Phương thức



- Yêu cầu khi nhấp chuột vào Form1 như hình 1 thì sẽ hiển thị Form2 ở giữa màn hình như hình 2

4.2.4. Phương thức



Giao diện Form2
với: Màu nền
form màu vàng,
Tiêu đề form là
“Giao diện Form
2”, vị trí hiển thị
form ở giữa màn
hình

4.2.4. Phương thức

- Bước 1: Tạo Dự án Windows Forms mới với form mặc định ban đầu tên Form1.
- Bước 2: Thiết lập các thuộc tính cho Form1 như sau:
 - ✓ Thiết lập tiêu đề, màu nền và vị trí hiển thị cho Form1: Trên cửa sổ Properties thiết lập thuộc tính *Text*, thuộc tính *BackColor* và thuộc tính *StartPosition*

4.2.4. Phương thức

The image shows a screenshot of the 'Properties' window for a Windows Form. The 'StartPosition' property is set to 'CenterScreen', the 'Text' property is 'Giao diện Form 1', and the 'BackColor' property is set to 'Highlight'. Three red rectangular boxes highlight these three properties. Three callout boxes with arrows point to these properties, providing Vietnamese descriptions: 'Vị trí hiển thị của Form1' (Position of Form1) for StartPosition, 'Thiết lập màu nền Form1' (Set background color of Form1) for BackColor, and 'Tiêu đề hiển thị trên titlebar của Form1' (Title displayed on the titlebar of Form1) for Text.

SizeGripStyle	Auto
StartPosition	CenterScreen
Tab	
Text	Giao diện Form 1
TopMost	False
TransparencyKey	<input type="checkbox"/>
BackColor	Highlight

Vị trí hiển thị của Form1

Tiêu đề hiển thị trên titlebar của Form1

Thiết lập màu nền Form1

4.2.4. Phương thức

- Bước 3: Tạo sự kiện *Click* của Form1 và thêm các mã lệnh sau cho sự kiện Click:

```
private void Form1_Click(object sender, EventArgs e)
{
    //Tạo đối tượng lớp Form
    Form Form2 = new Form();
    //Thiết lập tiêu đề trên titlebar của form
    Form2.Text = "Giao diện Form 2";
    //Thiết lập vị trí hiển thị form
    Form2.StartPosition = FormStartPosition.CenterScreen;
    //Thiết lập màu nền cho form
    Form2.BackColor = Color.CadetBlue;
    //Phương thức Show() giúp hiển thị Form2
    Form2.Show();
}
```

4.2.4. Phương thức

- Phương thức *ShowDialog()*: Phương thức *ShowDialog()* cũng có tác dụng hiển thị form như phương thức *Show()*.
- Nhưng khác biệt cơ bản là phương thức *Show()* giúp hiển thị form mới lên nhưng người dùng vẫn có thể quay lại thao tác trên các form đang hiển thị trước đó; Còn phương thức *ShowDialog()* sẽ hiển thị form mới lên và người dùng sẽ chỉ có thể thao tác trên form vừa hiển thị mà không thao tác được trên các form trước đó

4.2.4. Phương thức

- Phương thức *Hide()*: Phương thức giúp ẩn một form để form đó không hiển thị trên màn hình. Việc ẩn form của phương thức *Hide* (thực chất là thiết lập thuộc tính *Visible = false*.)

```
private void Form1_Click(object sender, EventArgs e)
{
    //this là con trỏ đại diện cho Form hiện hành,
    // Form hiện hành ở đây là Form1
    this.Hide();
    Form Form2 = new Form();
    Form2.Text = "Giao diện Form 2";
    Form2.StartPosition = FormStartPosition.CenterScreen;
    Form2.BackColor = Color.CadetBlue;
    Form2.ShowDialog();
}
```

4.2.4. Phương thức

- Phương thức *Close()*: Sử dụng để đóng form. Ví dụ: Trong sự kiện *Click* của Form1 như bước 3, thực hiện yêu cầu là khi đóng Form2 thì Form1 cũng đóng.

```
private void Form1_Click(object sender, EventArgs e)
{
    Form Form2 = new Form(); //Tạo đối tượng lớp Form
    Form2.Text = "Giao diện Form 2";
    //Thiết lập tiêu đề trên titlebar của form
    //Thiết lập vị trí hiển thị form
    Form2.StartPosition = FormStartPosition.CenterScreen;
    Form2.BackColor = Color.CadetBlue;
    //Thiết lập màu nền cho form
    //Kiểm tra giá trị trả về của phương thức ShowDialog()
    //nếu giá trị trả về là DialogResult.Cancel thì Form2
    //đã đóng, tiến hành đóng Form1 bằng phương thức Close()
    if (Form2.ShowDialog() == DialogResult.Cancel)
        this.Close();
}
```

4.3. Các điều khiển thông thường

4.3.1. Điều khiển Label

4.3.2. Điều khiển Button

4.3.3. Điều khiển TextBox

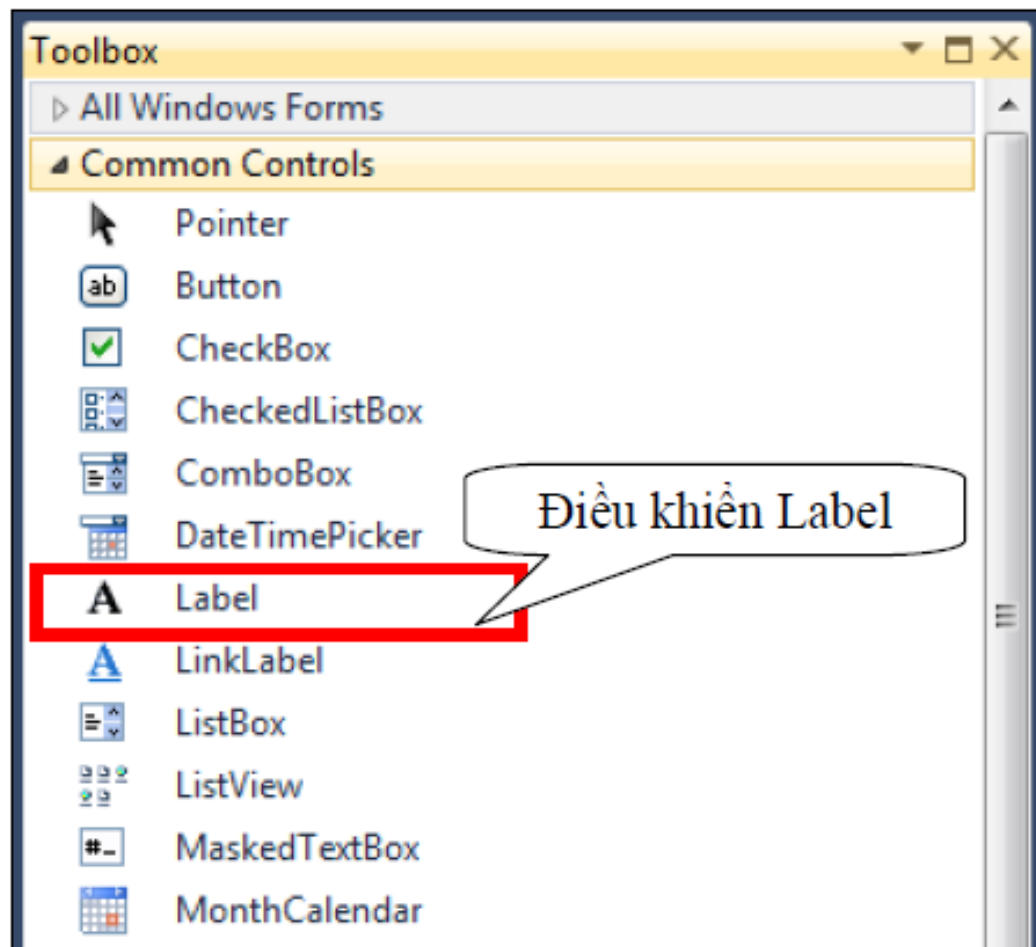
4.3.4. Điều khiển ComboBox và ListBox

4.3.5. Điều khiển CheckBox và RadioButton

4.3.1. 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*

4.3.1. Điều khiển Label

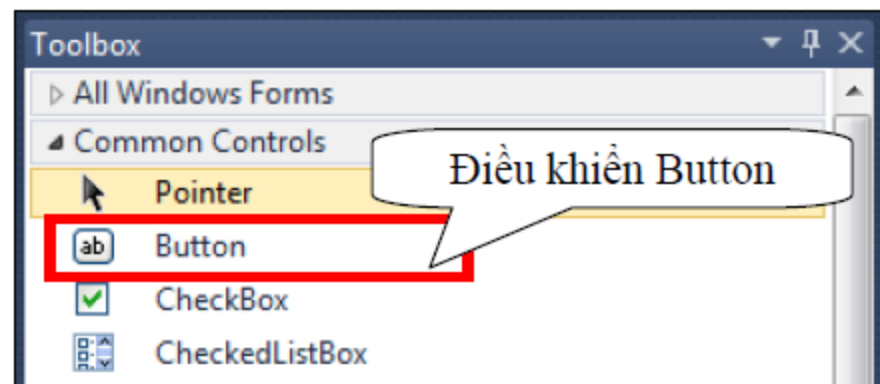


Một số thuộc tính thường dùng của *Label*

Thuộc tính	Mô tả
<i>BorderStyle</i>	Thiết lập đường viền
<i>Font</i>	Kích thước và kiểu chữ hiển thị trên Label
<i>Text</i>	Nội dung hiển thị trên Label
<i>BackColor</i>	Màu nền của Label
<i>Name</i>	Tên của Label
<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>	Đôi hình trỏ chuột khi đưa vào Label
<i>ForeColor</i>	Thiết lập màu chữ hiển thị trên Label
<i>TextAlign</i>	Căn lề chữ hiển thị trên Label
<i>Visible</i>	Ẩn hoặc hiện Label

4.3.2. 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*



Một số thuộc tính thường dùng của *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>	Ẩn/ hiện 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*

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> .

4.3.2. Điều khiển Button

- Ví dụ:
 - ✓ Thiết kế form xử lý nút lệnh như hình. 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.



4.3.2. Điều khiển Button

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



4.3.2. Điều khiển Button

- 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----”

4.3.2. Điều khiển Button

✓ button1:

- Thuộc tính *Name*: btnHienThi
- Thuộc tính *Text*: “Hiển thị”

✓ button2:

- Thuộc tính *Name*: btnThoat
- Thuộc tính *Text*: “Thoát”

4.3.2. Điều khiển Button

- Bước 3: Viết mã lệnh cho các nút lệnh:
 - ✓ Sự kiện *Click* của nút lệnh btnHienThi:

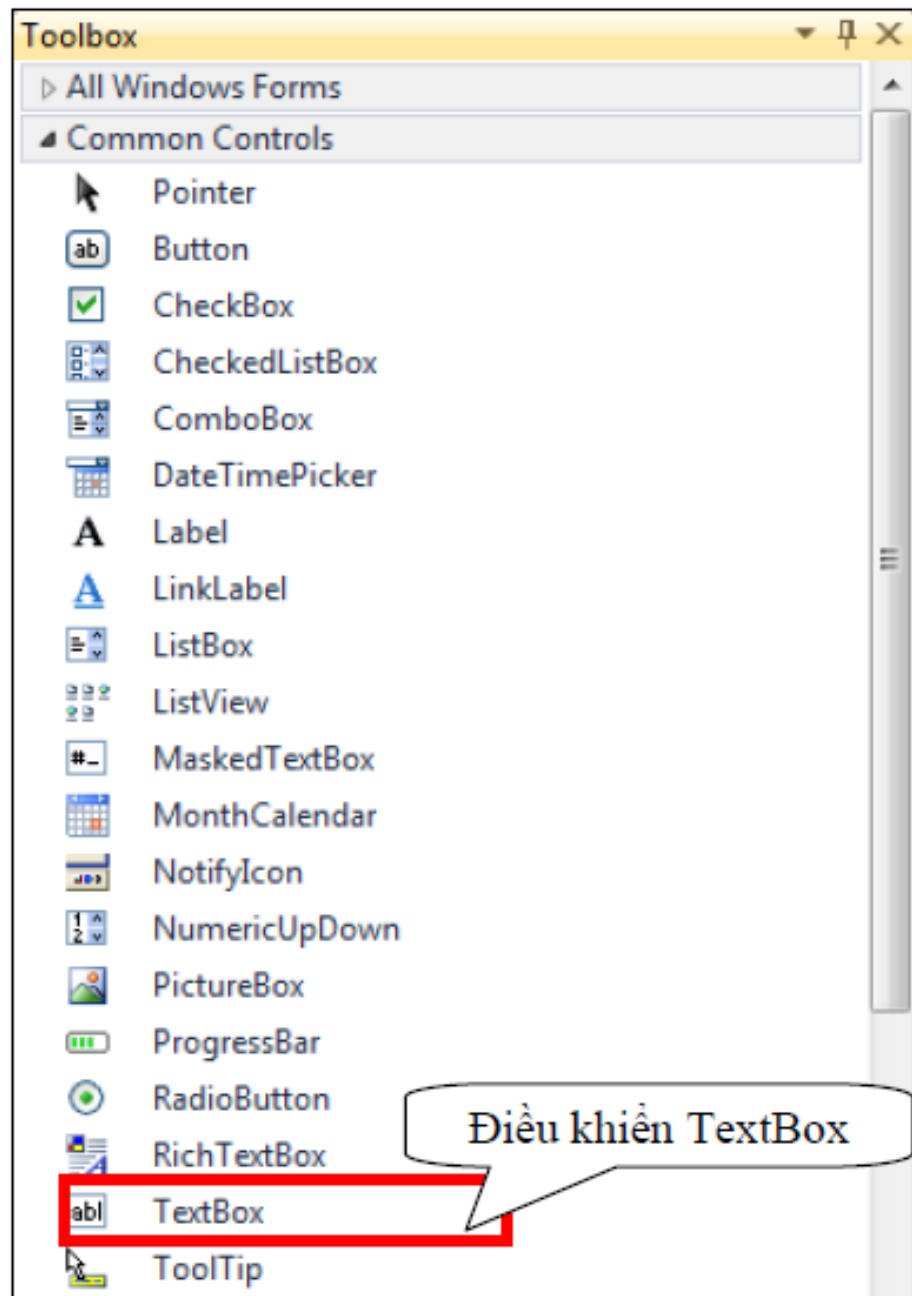
```
private void btnHienThi_Click(object sender, EventArgs e)
{
    LblNoiDung.Text = "Xin chao"
}
```

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

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

4.3.3. 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*



Một số phương thức thường dùng 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*

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/ hiện <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

Một số thuộc tính thường dùng của *TextBox*

Thuộc tính	Mô tả
<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*

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 Text bị thay đổi

4.3.3. Điều khiển TextBox

Ví dụ: Tính $n!$

- Bước 1: Thiết kế form như sau

Bài tập tính giai thừa

Tính giai thừa

Nhập N

N!

Tính Thoát

4.3.3. Điều khiển TextBox

- Bước 2: Thiết lập các thuộc tính của điều khiển trong cửa sổ *Properties*
 - ✓ *Form1*
 - Thuộc tính *Text* = “Bài tập tính giai thừa”
 - ✓ *label1*:
 - Thuộc tính *Text* = “Tính giai thừa”
 - Thuộc tính *Font*: Times New Roman, 20.25pt, style=Bold
 - Thuộc tính *ForeColor* = Red

4.3.3. Điều khiển TextBox

✓ label2:

- Thuộc tính *Text* = “Nhập N”

✓ label2:

- Thuộc tính *Text* = “N!”

✓ textbox1:

- Thuộc tính *Name* = txtN

✓ Textbox2:

- Thuộc tính *Name* = txtGt

4.3.3. Điều khiển TextBox

✓ button1:

- Thuộc tính *Name* = btnTinh
- Thuộc tính *Text* = “Tính”

✓ button2:

- Thuộc tính *Name* = btnThoat
- Thuộc tính *Text* = “Thoát”

4.3.3. Điều khiển TextBox

- Bước 3: Viết mã lệnh cho các nút lệnh:

✓ Sự kiện **Click** của nút lệnh btnTinh

```
private void btnTinh_Click(object sender, EventArgs e)
{
    int n, i;
    Int32 gt = 1;
    n = int.Parse(txtN.Text);
    for (i = 1; i <= n; i++)
        gt = gt * i;
    txtGt.Text = gt.ToString();
}
```

4.3.3. Điều khiển TextBox

✓ Sự kiện **Click** của nút lệnh btnThoat

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

Bài tập

- **Bài 1:** Viết chương trình nhập vào 2 số a, b và c cho biết số lớn nhất và nhỏ nhất trong 3 số a, b và c với giao diện như sau

Chương trình tìm số lớn nhất, nhỏ nhất của 2 số

Số nhập vào

A 4 B 9 C 6

Tìm

Kết quả

Số lớn nhất 9 Số nhỏ nhất 4

Bài tập

- **Bài 2:** Viết chương trình nhập vào giá trị nguyên dương N, tính tổng $S = 1 + 2 + 3 + \dots + N$ với giao diện như sau

The screenshot shows a Windows application window titled "TinhTong". Inside the window, there is a label "Nhập vào số tự nhiên N" followed by a text input field containing the number "4". To the right of the input field is a button labeled "Tính tổng". Below this, there is a section titled "Kết quả" (Result) which contains two lines of output: "S = 1+2+3+4" and "S = 10".

4.3.4. Điều khiển 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*



Một số thuộc tính thường dùng của *ListBox*

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.

Một số thuộc tính thường dùng của *ListBox*

Thuộc tính	Mô tả
<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

Một số thuộc tính thường dùng của *ListBox*

Thuộc tính	Mô tả
<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>

- **Một số thao tác với ListBox:**

- ✓ **Add():** Thêm một mục chọn vào cuối danh sách **ListBox**.

- listBox1.Items.Add("Hello");

- ✓ **Insert():** Chèn thêm mục vào vào vị trí **l**

- listBox1.Items.Add("A");
 - listBox1.Items.Add("C");
 - listBox1.Items.Insert(1, "B");

- ✓ **Count:** Trả về số mục chọn hiện đang có

- int n = listBox1.Items.Count;

- **Một số thao tác với ListBox:**

- ✓ **Items[i]:** Trả về mục chọn ở vị trí i.

- `MessageBox.Show(listBox1.Items[1].ToString());`

- ✓ **Remove:** Xóa đối tượng Item ra khỏi danh sách:
`Items.Remove(<Chuoi>);`

- `listBox1.Items.Remove("B");`

- ✓ **RemoveAt:** Xóa một item tại vị trí index ra khỏi danh sách: `Items.RemoveAt(<index>);`

- `listBox1.Items.RemoveAt(1);`

- ✓ **Clear:** Xóa tất cả những Item có trong danh sách

- `listBox1.Items.Clear();`

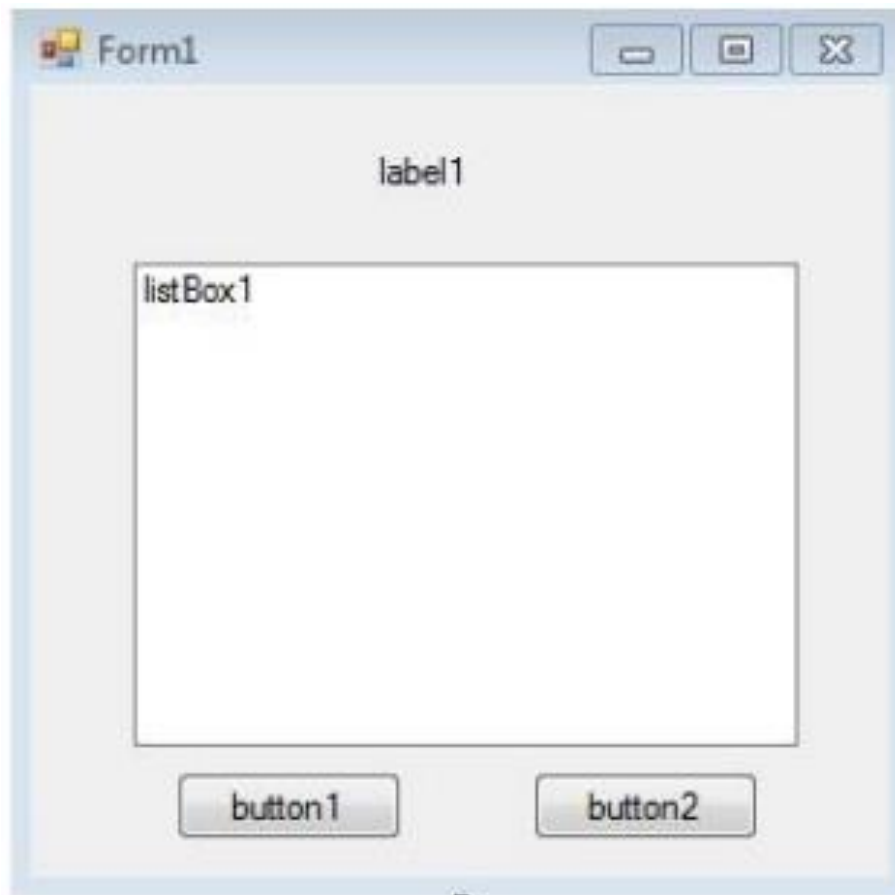
4.3.4. Điều khiển ListBox

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



4.3.4. Điều khiển ListBox

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



4.3.4. Điều khiển ListBox

- 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

4.3.4. Điều khiển ListBox



4.3.4. Điều khiển ListBox

- 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);
}
```

4.3.4. Điều khiển ListBox

- Sự kiện *Click* của nút lệnh btnThoat

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

4.3.5. Điều khiển 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



Một số thuộc tính thường dùng 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>

Một số thuộc tính thường dùng của *ComboBox*

Thuộc tính	Mô tả
<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, <i>ArrayList</i> , ..

Một số thuộc tính thường dùng của *ComboBox*

Thuộc tính	Mô tả
<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>

4.3.5. Điều khiển ComboBox

- Ví dụ: Xây dựng chương trình với giao diện như hình dưới gồm *ComboBox* chứa danh sách 4 phép tính (+, -, *, /).

The screenshot shows a Windows application window titled "Form1". Inside the window, there are four labels and four corresponding input controls arranged vertically:

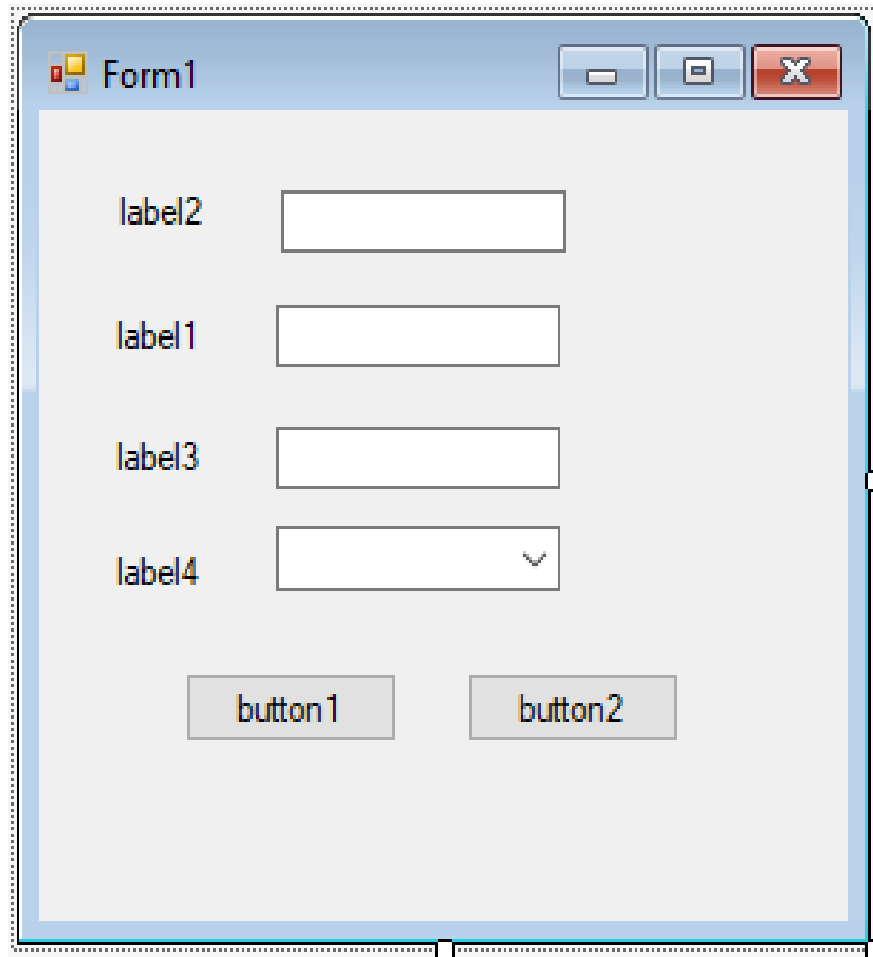
- Label: "Số thứ nhất" (First number), followed by a text box.
- Label: "Số thứ hai" (Second number), followed by a text box.
- Label: "Kết quả" (Result), followed by a text box.
- Label: "Phép tính" (Operation), followed by a *ComboBox* with a dropdown arrow.

At the bottom of the form, there are two buttons:

- A button labeled "Tính" (Calculate) with a dashed border, indicating it is currently selected or being edited.
- A button labeled "Thoát" (Exit).

4.3.5. Điều khiển ComboBox

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



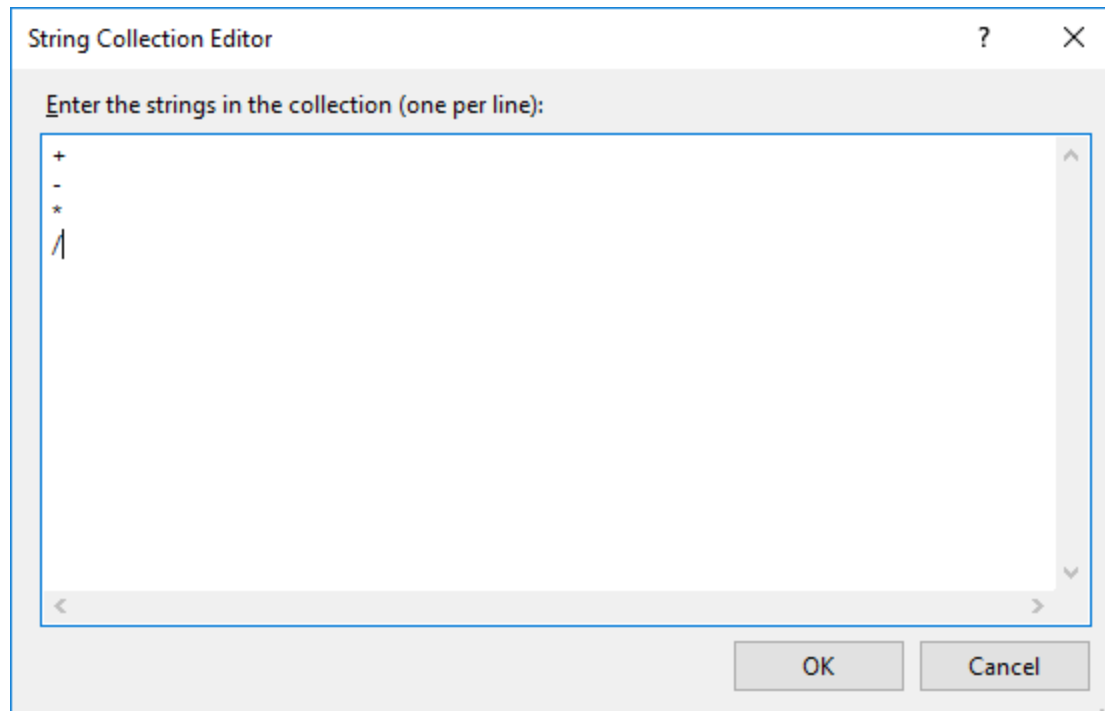
4.3.5. Điều khiển ComboBox

- Bước 2: Thiết lập giá trị thuộc tính cho điều khiển
 - ✓ label1:
 - Thuộc tính Text = “Số thứ nhất”
 - ✓ label2:
 - Thuộc tính Text = “Số thứ hai”
 - ✓ label3:
 - Thuộc tính Text = “Kết quả”
 - ✓ label4:
 - Thuộc tính Text = “Phép tính”
 - ✓ button1:
 - Thuộc tính Name = btnTinh
 - Thuộc tính Text = “Tính”
 - ✓ button2:
 - Thuộc tính Name = btnThoat
 - Thuộc tính Text = “Thoát”

4.3.5. Điều khiển ComboBox

✓ comboBox1:

- Thuộc tính Name = cmbPheptinh
- comboBox1 chứa danh sách 4 phép tính gồm: +, -, *, / bằng cách thiết lập thuộc tính *Items* của comboBox1 trong cửa sổ *Properties* như hình



4.3.5. Điều khiển ComboBox

- Bước 3: Viết mã lệnh cho các nút lệnh:
 - ✓ Sự kiện *Click* của nút lệnh btnTinh

```
private void btnTinh_Click(object sender, EventArgs e)
{
    float so1 = float.Parse(txtSo1.Text);
    float so2 = float.Parse(txtSo2.Text);
    float kq=0;
    if (cmbPheptinh.Text == "+") kq = so1 + so2;
    if (cmbPheptinh.Text == "-") kq = so1 - so2;
    if (cmbPheptinh.Text == "*") kq = so1 * so2;
    if (cmbPheptinh.Text == "/") kq = so1 / so2;
    txtKq.Text = kq.ToString();
}
```

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

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>Count</i>	Trả về số mục chọn hiện đang có
<i>Item(i)</i>	Trả về mục chọn ở vị trí thứ 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 khỏi <i>ComboBox</i> hoặc <i>ListBox</i>

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

Phương thức	Mô tả
<i>Contains(<Mục cần tìm>)</i>	Trả về True nếu có mục chọn <Mục cần tìm> trong danh sách, trả về False 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

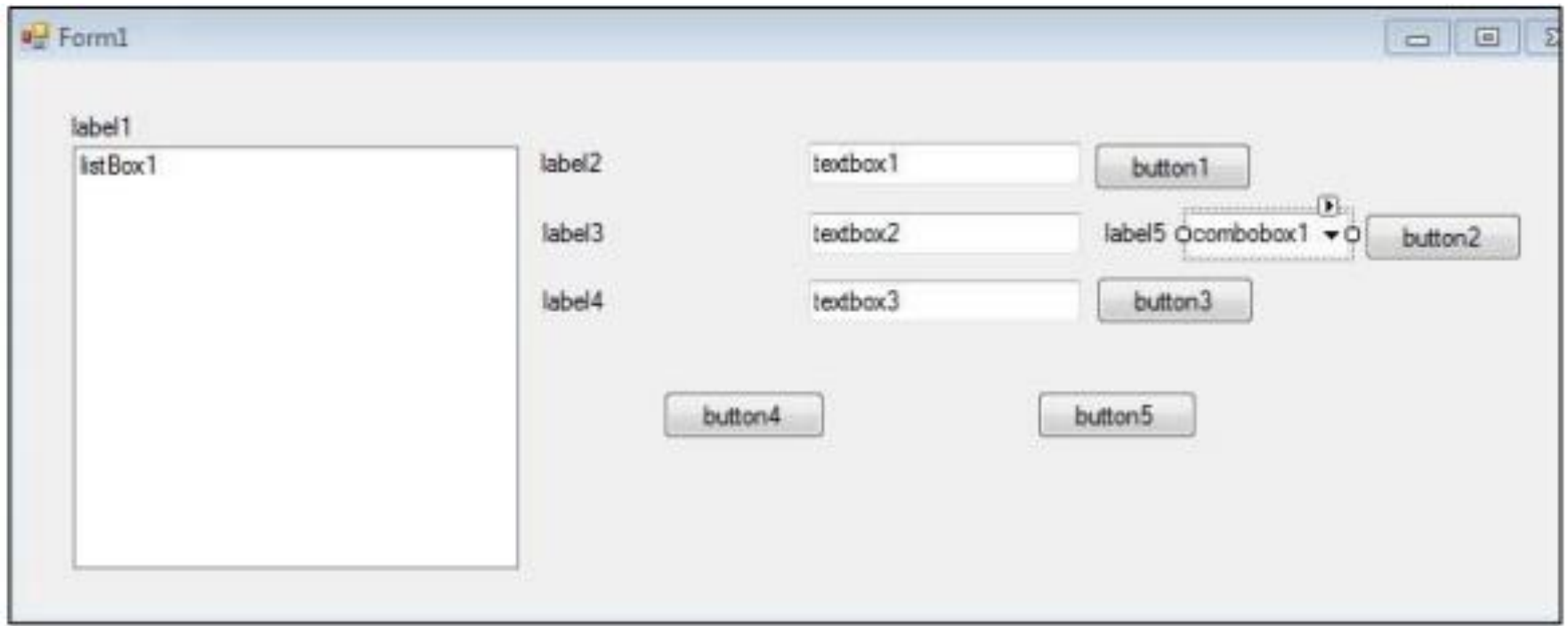
- Ví dụ: Thiết kế form quản lý danh sách sản phẩm như hình. Với CboViTri là điều khiển *ComboBox* chứa danh sách chỉ số của danh sách sản phẩm trong *ListBox* Li_DSSanPham.

The image shows a Windows form titled "Form1" with a light blue header. The form contains the following elements:

- A list box on the left labeled "Danh sách sản phẩm" and "listBox1". A callout bubble points to it with the text "Li_DSSanPham".
- Two input fields for "Thêm sản phẩm:" with a "Thêm" button next to each. A callout bubble points to the first "Thêm" button with the text "btnThem".
- A "Thêm sản phẩm:" input field with a "Thêm VT" button next to it. A callout bubble points to this button with the text "btnThemVT".
- A "Nhập sản phẩm cần tìm:" input field with a "Tìm kiếm" button next to it. A callout bubble points to this button with the text "btnTimKiem".
- A "Xóa sản phẩm" button with a callout bubble pointing to it with the text "btnXoa".
- A "Xóa danh sách sản phẩm" button with a callout bubble pointing to it with the text "btnXoaDS".
- A "CboViTri" label with a callout bubble pointing to the "Thêm VT" button.

- 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 CboVitri 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.

- Hướng dẫn:
 - ✓ Bước 1: Thiết kế giao diện ban đầu như hình



✓ 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

- ✓ **Sự kiện *Form_Load* cho Form1 như sau:**

- ```
Private void Form1_Load(object sender, EventArgs e)
{
 ThietLapViTriComboBox();
}
```

- **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 *Click* nút lệnh btnThem

```
Private void 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

```
Private void 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

```
Private void 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

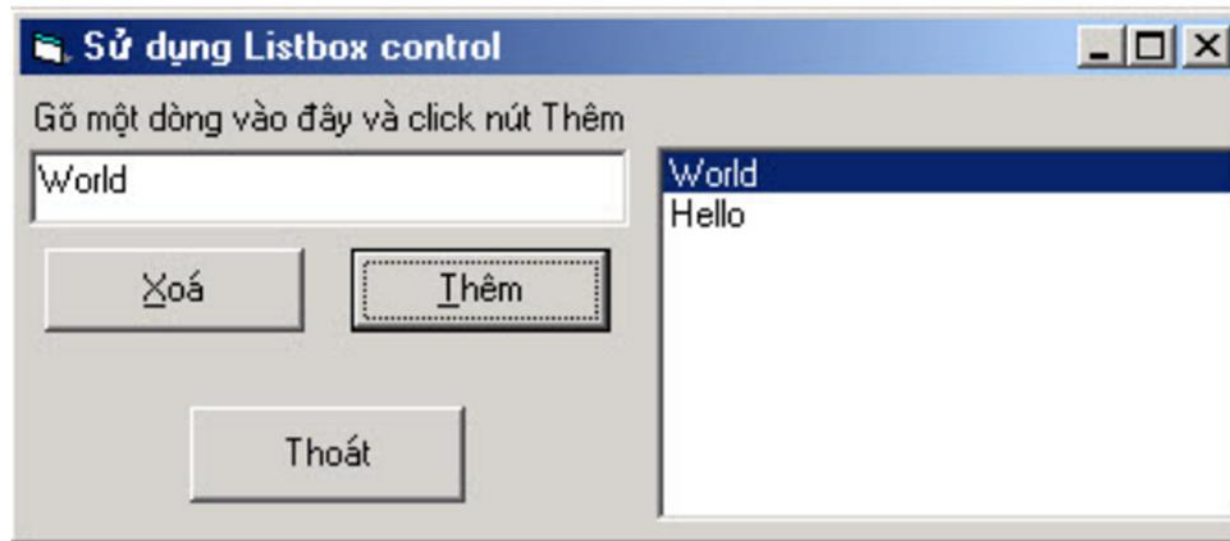
```
Private void btnXoa_Click(object sender, EventArgs e)
{
    if (DSSanPham.SelectedIndex < 0)
        MessageBox.Show("Chọn sản phẩm muốn xóa ");
    else
    {
        DSSanPham.Items.Remove(DSSanPham.SelectedItem);
        MessageBox.Show("Xóa sản phẩm thành công");
    }
}
```

✓ Sự kiện *Click* nút lệnh btnTimKiem

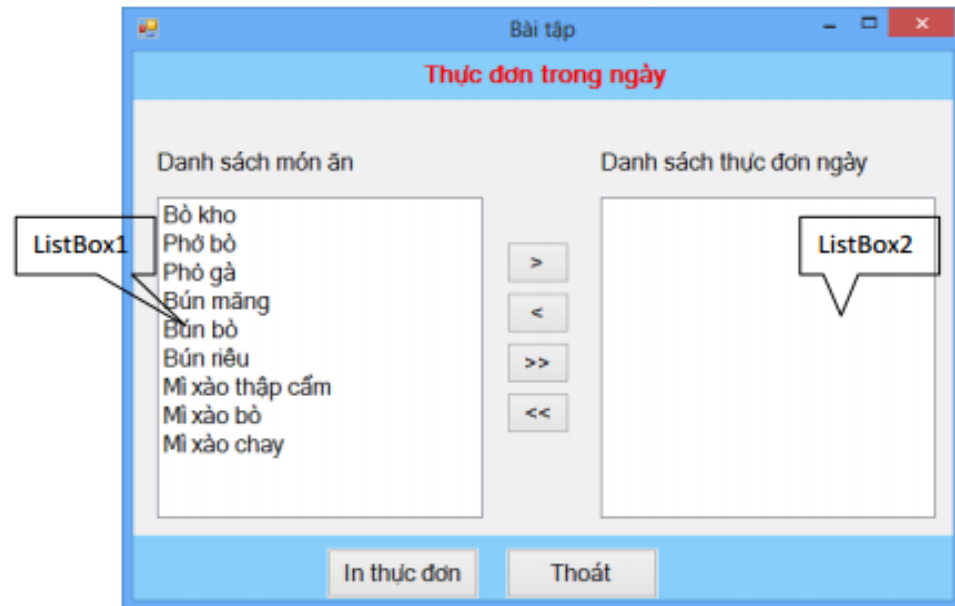
```
Private void btnTimKiem_Click(object sender, EventArgs e)
{
    if(txtTimSP.Text != "")
    {
        if(DSSanPham.Items.Contains(txtTimSP.Text) == true)
            MessageBox.Show("Tìm thấy sản phẩm");
        else
            MessageBox.Show("Không Tìm thấy sản phẩm ");
    }
    else
       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

Bài tập

- Bài tập 1: Thêm các mục vào danh sách (ListBox) khi người dùng click vào nút "Thêm" và xoá tất cả các mục khi người dùng click vào nút "Xoá"



- Bài 2: 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



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

4.3.6. Điều khiển 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*



Một số thuộc tính thường dùng 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>	<p>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>.</p> <p><i>Checked</i>: Điều khiển đang được chọn</p> <p><i>UnChecked</i>: Điều khiển không được chọn</p> <p><i>Indeterminate</i>: Điều khiển ở trạng thái không hoạt động</p>

Một số thuộc tính thường dùng của *CheckBox*

Thuộc tính	Mô tả
<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*.

4.3.6. Điều khiển CheckBox

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

The screenshot shows a Windows application window titled "Form1". Inside the window, the text "Thông tin sinh viên" is centered at the top. Below this, there are three input fields: "Nhập họ tên:" followed by a text box, "Lớp:" followed by a text box, and "Danh sách môn học tự chọn:" followed by a list box. The list box contains four items, each preceded by an unchecked checkbox: "Lập trình C#", "Mạng máy tính", "Xử lý ảnh", and "Lập trình Web". At the bottom of the form, there are two buttons: "Lưu" (Save) and "Thoát" (Exit).

4.3.6. Điều khiển CheckBox

- Bước 1: Thiết kế giao diện form như hình



The screenshot shows a Windows form titled "Form2" with a standard Windows XP-style title bar. The form contains the following controls:

- label1**: A label positioned at the top center of the form.
- label2**, **label3**, and **label4**: Three labels arranged vertically on the left side of the form.
- textBox1**: A single-line text box located to the right of label2.
- textBox2**: A multi-line text box located to the right of label3 and label4. It has a vertical scrollbar on its right side.
- checkBox1**, **checkBox2**, **checkBox3**, and **checkBox4**: Four unchecked checkboxes arranged vertically in the lower-middle section of the form.
- button1** and **button2**: Two buttons located at the bottom of the form, with button1 on the left and button2 on the right.

Thiết lập thuộc tính điều khiển

- 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: “Lưu”
 - Thuộc tính Name: btnLuu
- button2:
 - Thuộc tính Text: “Thoát”
 - Thuộc tính Name: btnThoat

4.3.6. Điều khiển CheckBox

✓ Bước 3: Viết mã lệnh cho Sự kiện *Click* của 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);
}
```

4.3.7. Điều khiển 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.

4.3.7. Điều khiển RadioButton

- Điều khiển *RadioButton* được đặt trong nhóm *Common Controls* của cửa sổ *Toolbox*



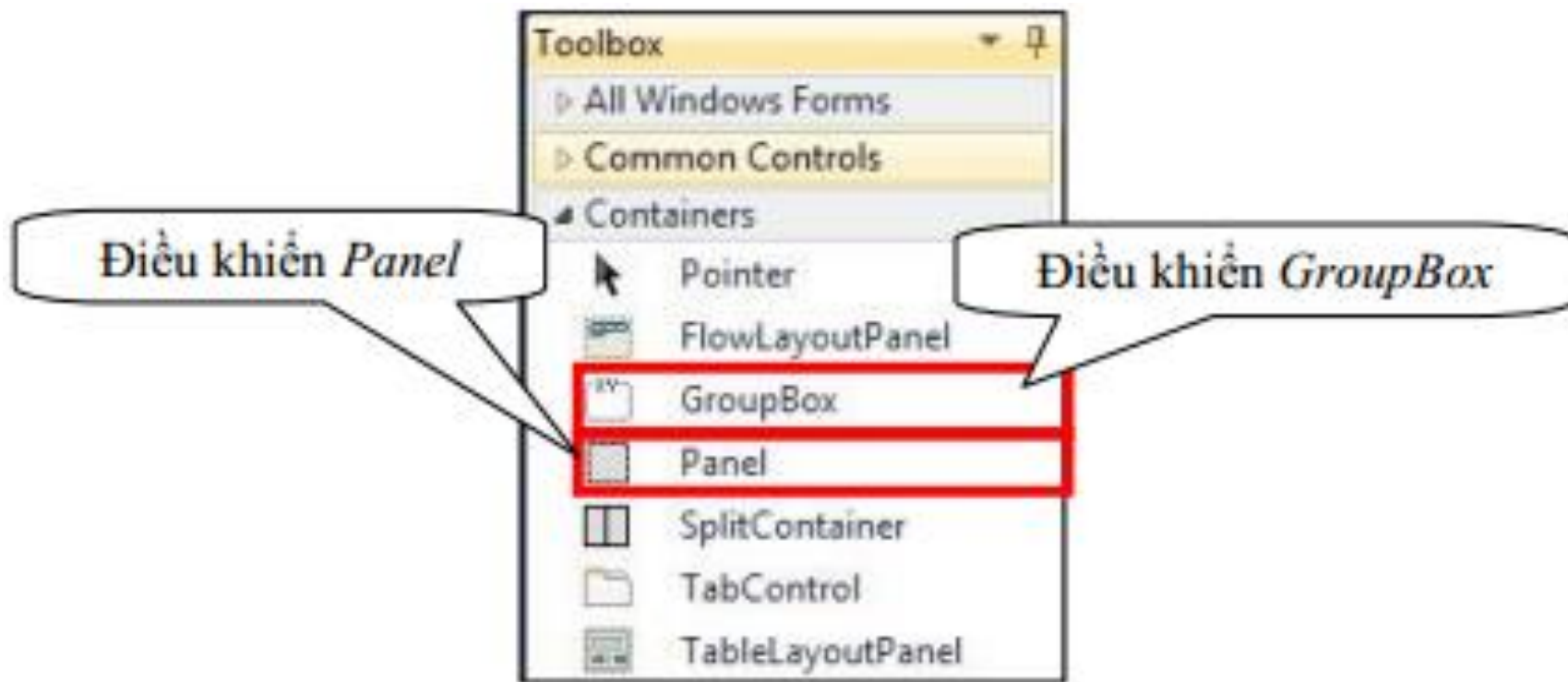
Một số thuộc tính thường dùng của *RadioButton*

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>

4.3.7. Điều khiển RadioButton

- Ta 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*. Điều này cho phép người dùng chỉ có thể chọn duy nhất một *RadioButton* trong nhóm

4.3.7. Điều khiển RadioButton



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.

Sự kiện thường sử dụng của RadioButton

- Ví dụ: Xây dựng chương trình với giao diện như hình

The screenshot shows a Windows application window titled "Form1". Inside the window, there is a user interface for a simple calculator. On the left side, there are three text boxes: the first is labeled "Số thứ nhất" (First number), the second is labeled "Số thứ hai" (Second number), and the third is labeled "Kết quả" (Result). To the right of these text boxes are three empty text input fields. On the right side of the form, there is a group box titled "Phép tính" (Operation). Inside this group box, there are four radio buttons: "Cộng" (Add), "Trừ" (Subtract), "Nhân" (Multiply), and "Chia" (Divide). The "Cộng" radio button is currently selected. Below the group box, there are two buttons: "Tính" (Calculate) and "Thoát" (Exit).

Sự kiện thường sử dụng của RadioButton

- Thiết lập giá trị thuộc tính cho điều khiển
 - ✓ label1:
 - Thuộc tính Text = “Số thứ nhất”
 - ✓ label2:
 - Thuộc tính Text = “Số thứ hai”
 - ✓ label3:
 - Thuộc tính Text = “Kết quả”
 - ✓ button1:
 - Thuộc tính Name = btnTinh
 - Thuộc tính Text = “Tính”
 - ✓ button2:
 - Thuộc tính Name = btnThoat
 - Thuộc tính Text = “Thoát”

Sự kiện thường sử dụng của RadioButton

- ✓ groupBox3:
 - Thuộc tính *Text*: “Phép tính”
- ✓ radioButton1:
 - Thuộc tính *Name*: radCong
 - Thuộc tính *Checked*: True
- ✓ radioButton2:
 - Thuộc tính *Name*: radTru
- ✓ radioButton3:
 - Thuộc tính *Name*: radNhan
- ✓ radioButton4:
 - Thuộc tính *Name*: radChia
 -

Sự kiện thường sử dụng của RadioButton

- Viết mã lệnh sự kiện *Click* nút lệnh btnTinh

```
private void btnTinh_Click(object sender, EventArgs e)
{
    float so1 = float.Parse(txtSo1.Text);
    float so2 = float.Parse(txtSo2.Text);
    float kq=0;
    if (radCong.Checked == true) kq = so1 + so2;
    if (radTru.Checked == true) kq = so1 - so2;
    if (radNhan.Checked == true) kq = so1 * so2;
    if (radChia.Checked == true) kq = so1 / so2;
    txtKq.Text = kq.ToString();
}
```

Bài tập

Bài 1: Viết chương trình thực hiện việc thiết lập màu của Form là Xanh, Đỏ và Vàng tương ứng khi người dùng chọn nút radio Xanh, Đỏ, Vàng



Bài tập

- **Bài 2:** Viết chương trình tạo form đăng nhập có giao diện như hình

The image shows a Windows-style application window titled "Bài tập" (Exercise). Inside the window, there is a form titled "Đăng nhập" (Login) in red text. The form has a light gray background and a blue border. It contains the following elements:

- A label "Tên đăng nhập:" (Username) followed by a text input field containing the text "admin". A callout box labeled "txtTen" points to this field.
- A label "Mật khẩu:" (Password) followed by a password input field containing six asterisks "*****". A callout box labeled "txtMatKhai" points to this field.
- A checkbox labeled "Hiển thị mật khẩu" (Show password) with a callout box labeled "chkHienThi" pointing to it.
- At the bottom of the form, there are two buttons: "Đăng nhập" (Login) and "Thoát" (Exit).

- Người dùng nhập tên tài khoản và mật khẩu trên TextBox txtTen và txtMatkhau.
 - ✓ Nếu CheckBox chkHienThi không được chọn thì ở TextBox txtMatkhau sẽ hiện dấu * với mỗi ký tự người dùng gõ.
 - ✓ Nếu CheckBox chkHienThi được chọn thì TextBox txtMatkhau 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

Bài tập

- Hướng dẫn đặt hoặc bỏ kí tự thay thế

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if(checkBox1.Checked == true)
        textBox1.PasswordChar = '\0';
    else
        textBox1.PasswordChar = '*';
}
```

Bài tập

Bài 3: Chương trình sử dụng hộp OptionBox và CheckBox để chọn màu sắc (đỏ, xanh da trời, đen) và kiểu chữ (nghiêng, đậm, gạch dưới) cho đoạn văn bản “Trường đại học Mở - Địa chất – Khoa Công nghệ Thông tin”



- Hướng dẫn

- ✓ Thay đổi màu chữ

- ```
textBox1.ForeColor = Color.Red;
```

- ✓ Thay đổi Font chữ, cỡ chữ

- ```
/*Khởi tạo font mới*/
```

- ```
FontFamily f = new FontFamily("Times New Roman");
```

- ```
textBox1.Font = new Font(f, 20);
```

- ✓ Font chữ đậm

- ```
textBox1.Font = new Font(textBox1.Font, textBox1.Font.Style
^ FontStyle.Bold)
```

- ✓ Font chữ đậm và nghiêng

- ```
textBox1.Font = new Font(textBox1.Font, textBox1.Font.Style  
^ FontStyle.Italic^ FontStyle.Bold);
```

Bài tập

Bài 4: Xây dựng chương trình nhập vào một số nguyên dương n , sau đó lựa chọn tính một trong các tổng sau đây:

$$S_1 = \sum_{i=1}^n (2.i-1) = 1 + 3 + 5 + \dots + (2n - 1)$$

$$S_2 = \sum_{i=1}^n i^2 = 1 + 4 + 9 + \dots + n^2$$

$$S_3 = \sum_{i=1}^n \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$$

$$S_4 = \sum_{i=1}^n \frac{1}{i!} = 1 + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{n!}$$

Bài tập

- ✓ Thiết kế giao diện chương trình lựa chọn tính tổng như sau:

Bài tập

Bài 5: Viết chương trình nhập họ tên, phái và ngày tháng năm sinh của một người. Chương trình sẽ thông báo tuổi và nhân xưng của người đó theo tuổi.

The image shows a Windows application window titled "Bài tập 1". It contains a form with the following fields and controls:

- Họ tên:** A text box containing "Nguyễn thị Hoa".
- Ngày tháng năm sinh:** A text box containing "15/2/1970".
- Phái:** A group box containing two radio buttons: "Nam" (unselected) and "Nữ" (selected).
- Buttons:** Two buttons labeled "Nhập" and "Kết thúc".

Below the form, a message box titled "Ngày sinh" is displayed with the text: "Nguyễn thị Hoa, chị 30 tuổi, sinh vào ngày thứ bảy".

Bài tập

- Thang phân loại như sau:

Nam:

Tuổi	Nhân xưng
1 -10	Cháu
11 -17	Em
18-34	Anh
35-70	Ông
71 ->	Cụ

Nữ:

Tuổi	Nhân xưng
1 -10	Cháu
11 -23	Em
24-30	Chị
31 -65	Bà
66->	Cụ

Hướng dẫn: Phải nhập được chọn bằng các Option Button

```
DateTime da = new DateTime();  
da = Convert.ToDateTime(textBox1.Text);  
DateTime ds = new DateTime();
```

```
int i = ds.Year - da.Year;
```

Bài tập

Bài 6: Thiết kế form và thực hiện các chức năng

The screenshot shows a Windows application window titled "Bài tập ListBox". It features a text input field labeled "Nhập Số" with a "Cập nhật" button to its right. Below the input field is a list box labeled "Lớp A:" containing the numbers 1 through 14. The number 3 is currently selected in the list box. To the right of the list box is a panel titled "Các chức năng" (Functions) which contains several buttons: "Tổng cửa danh sách", "Xóa phần tử đầu và cuối", "Xóa phần tử đang chọn", "Tăng mỗi phần tử lên 2", "Thay bằng bình phương", "Chọn số chẵn", and "Chọn số lẻ". At the bottom of the window is a large button labeled "KẾT THÚC".

Yêu cầu

Khi người sử dụng nhập một số vào textbox rồi Enter hoặc nhấp vào nút cập nhật thì số này được thêm vào listbox, đồng thời nội dung trong textbox bị xóa và focus được chuyển về textbox.

Người dùng nhấn vào nút nào thì thực hiện chức năng tương ứng của nút đó.