==========

Bài giảng

Cơ sở lập trình 2

==========

Mục lục

CHƯƠNG 1.

LÀM QUEN VỚI VISUAL STUDIO 2010

1. Giới thiệu Visual Studio.NET 2010

1.1. Tình hình trước khi Visual Studio.NET ra đời

Với sự phát triển liên tục và đa dạng của thế giới công nghệ thông tin ngày nay, các phần mềm, các hệ điều hành, các môi trường phát triển và các ứng dụng liên tục ra đời. Tuy nhiên, đôi khi việc phát triển không đồng nhất và nhất là do không tương thích về mặt lợi ích của các công ty phần mềm lớn đã làm ảnh hưởng đến công việc của những kỹ sư xây dựng phần mềm.

Trong giới phát triển ứng dụng trên Internet ta có thể sử dụng các ngôn ngữ Java, PHP, ASP… Khi Java mới được Sun Corporation giới thiệu nó đã có một sức mạnh đáng kể và hướng tới việc chạy trên nhiều hệ điều hành khác nhau, độc lập với các bộ xử lý. Đặc biệt Java rất thích hợp cho việc viết các ứng dụng trên Internet. Tuy nhiên, Java lại có hạn chế về mặt tốc độ và trên thực tế vẫn chưa thịnh hành.

Để làm giảm khả năng ảnh hưởng của Java, bên hãng Microsoft cũng cung cấp ngôn ngữ ASP - chuyên dùng để viết các ứng dụng trên Web. Trong các trang ASP vừa chứa thẻ HTML vừa chứa các đoạn script (VBScript, JavaScript). Trong quá trình xử lý một trang ASP, nếu là thẻ HTML thì sẽ được gửi thẳng tới trình duyệt, còn nếu là các đoạn script thì sẽ được chuyển thành các dòng HTML rồi gửi đi. Khi nhà lập trình muốn đóng gói và sử dụng lại một số chức năng nào đó, thì họ dịch các đoạn chương trình thành ActiveX và đưa nó vào Web Server. Tuy nhiên, vì lý do bảo mật nên các Admin của các trang Web thường rất dè dặt khi cài ActiveX lạ trên máy của họ, ngoài ra việc tháo gỡ các phiên bản của ActiveX này cũng là công việc rất khó khăn.

Còn trong giới phát triển ứng dụng trên Windows ta có thể viết ứng dụng bằng Visual C++, Delphi, Visual Basic… đây là một số công cụ phổ biến và mạnh. Trong đó Visual C++ là một ngôn ngữ rất mạnh nhưng cũng rất khó sử dụng. Visual Basic thì đơn giản dễ học, dễ dùng nhất nên rất thông dụng nhưng hạn chế là Visual Basic không phải ngôn ngữ hướng đối tượng và không hỗ trợ khả năng phát triển thuật toán.

Tóm lại trong giới lập trình theo Microsoft thì việc lập trình trên desktop cho đến lập trình hệ phân tán hay trên web là những mảng độc lập.

1.2. Sự ra đời của Visual Studio.NET

Đầu năm 1998, sau khi hoàn tất phiên bản Version 4 của Internet Information Server -IIS, đội ngũ lập trình của Microsoft nhận thấy họ còn có rất nhiều sáng kiến để có thể kiện toàn

IIS, và họ bắt đầu xây dựng một kiến trúc mới trên nền tảng ý tưởng đó và đặt tên là Next Generation Windows Services - NGWS. Tham vọng của họ là cung cấp một môi trường có thể dùng chung cho tất cả ngôn ngữ lập trình trong bộ Visual Studio cũng như cho các ngôn ngữ lập trình của các công ty khác.

Kết quả là năm 2001 Visual Studio.Net 2001 ra đời đánh dấu cho một môi trường lập trình trên nền .NET Framework 1.0 tiên tiến mới.

Năm 2003, sau 2 năm .NET Framework nâng cấp thêm một bậc với phiên bản 1.1 với đặc điểm ngoài các chương trình Windows truyền thống – là các tệp tin .exe giờ đây Windows còn tồn tại những chương trình khác – những chương trình chạy trên nền .NET. Muốn chạy chương trình .NET ta chỉ cần cài .NET Framework là đủ. Một điểm lý thú và cũng là điều mong đợi của tất cả lập trình viên, từ phiên bản Windows 2003 .NET Framework được cài đặt như một phần mặc định của Windows. Song song đó, môi trường phát triển Visual Studio .NET 2001 được nâng cấp thành Visual Studio .NET 2003 cho phép viết và chạy các ứng dụng trên nền .NET Framework 1.1

Cuối năm 2005, Visual Studio 2005 với nền .NET Framework 2.0 mạnh mẽ và vượt trội hơn so với nền .NET Framwork 1.1 trước đó. Ngay sau đó Microsoft công bố phiên bản Windows Vista, và toàn bộ Windows là .NET, tất cả các hàm API lõi trong những phiên bản Windows trước đây đều đã được thay thế bằng các hàm hay thư viện .NET. Microsoft đã viết lại hoàn toàn lõi API, không còn một lớp API nào nữa.

1.3. Tổng quan về Visual Studio.NET

Visual Studio.NET gồm 2 phần: Framework và Integrated Development Environment– IDE, cho phép lập trình viên khi xây dựng các ứng dụng có thể lựa chọn sử dụng nhiều ngôn ngữ lập trình khác nhau như Visual C++.NET, Visual C#.NET, Visual J#.NET, Visual Basic.NET… trong cùng một môi trường phát triển IDE thống nhất trên kiến trúc .NET Framework.

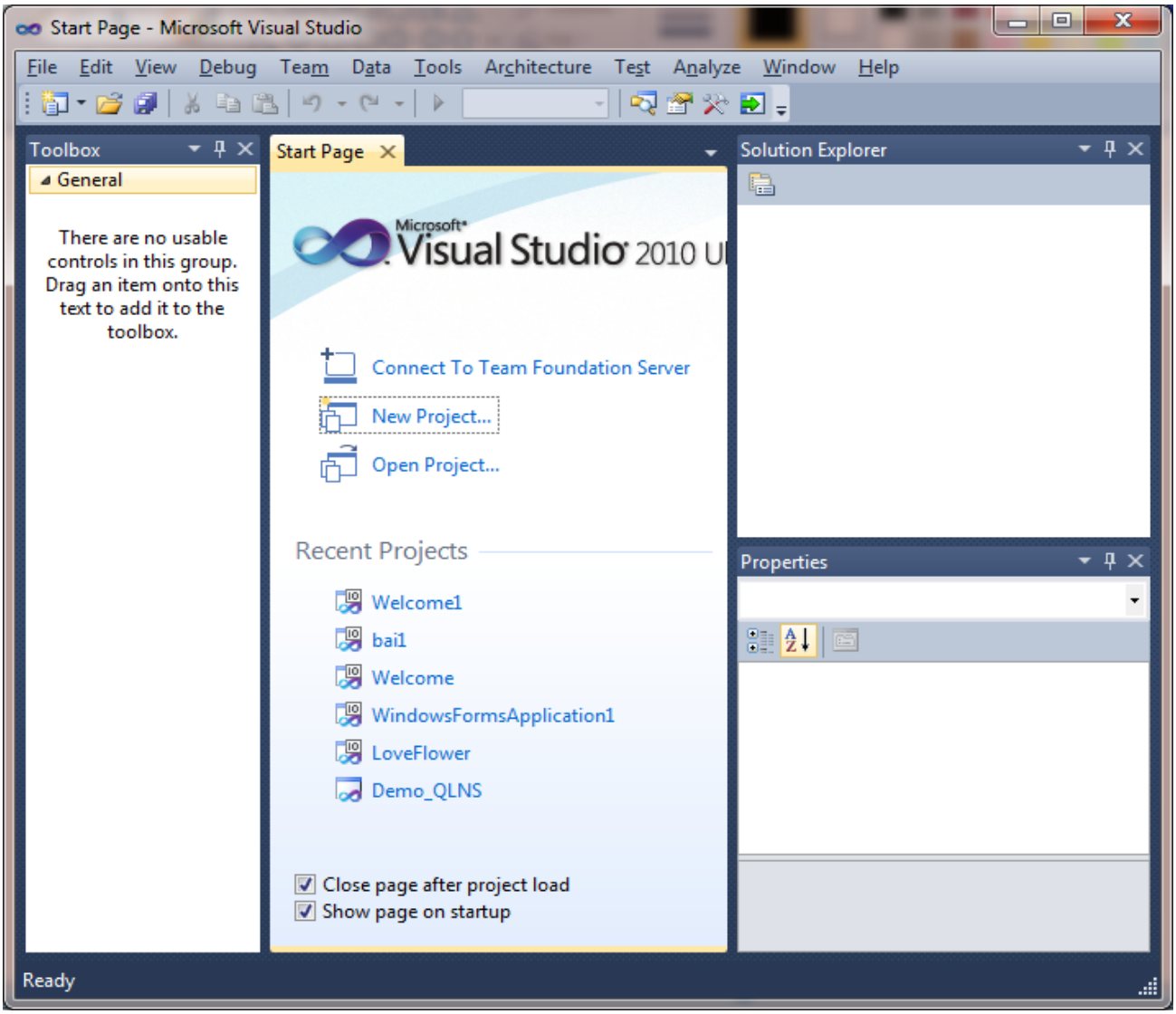
Framework là thành phần quan trọng nhất, là cốt lõi và tinh hoa của môi trường .NET, Framework giúp chúng ta biên dịch và thực thi các ứng dụng .NET (cấu trúc của Framework chúng ta sẽ tìm hiểu ở các chương sau của giáo trình).

IDE cung cấp một môi trường phát triển trực quan, giúp các lập trình viên có thể dễ dàng và nhanh chóng xây dựng giao diện cũng như viết mã lệnh cho các ứng dụng dựa trên nền tảng .NET. Nếu không có IDE chúng ta cũng có thể dùng một trình soạn thảo văn bản bất kỳ, ví dụ như Notepad để viết mã lệnh và sử dụng command line để biên dịch và thực thi ứng dụng. Tuy nhiên việc này mất rất nhiều thời gian, tốt nhất là chúng ta nên dùng IDE để phát triển các ứng dụng, và đó cũng là cách dễ sử dụng nhất.

Ngoài ra trong Visual Studio.NET thì lập trình Winform và Webform là tương tự, ví dụ cả Visual C#.NET lẫn Visual Basic.NET đều hỗ trợ khả năng lập trình trên Win và Web…

2. Khởi động Visual C# 2010 và giao diện

❶ Vào Start/Programs/Microsoft Visual Studio 2010/Microsoft Visual Studio 2010, xuất hiện cửa sổ *Start Page*.



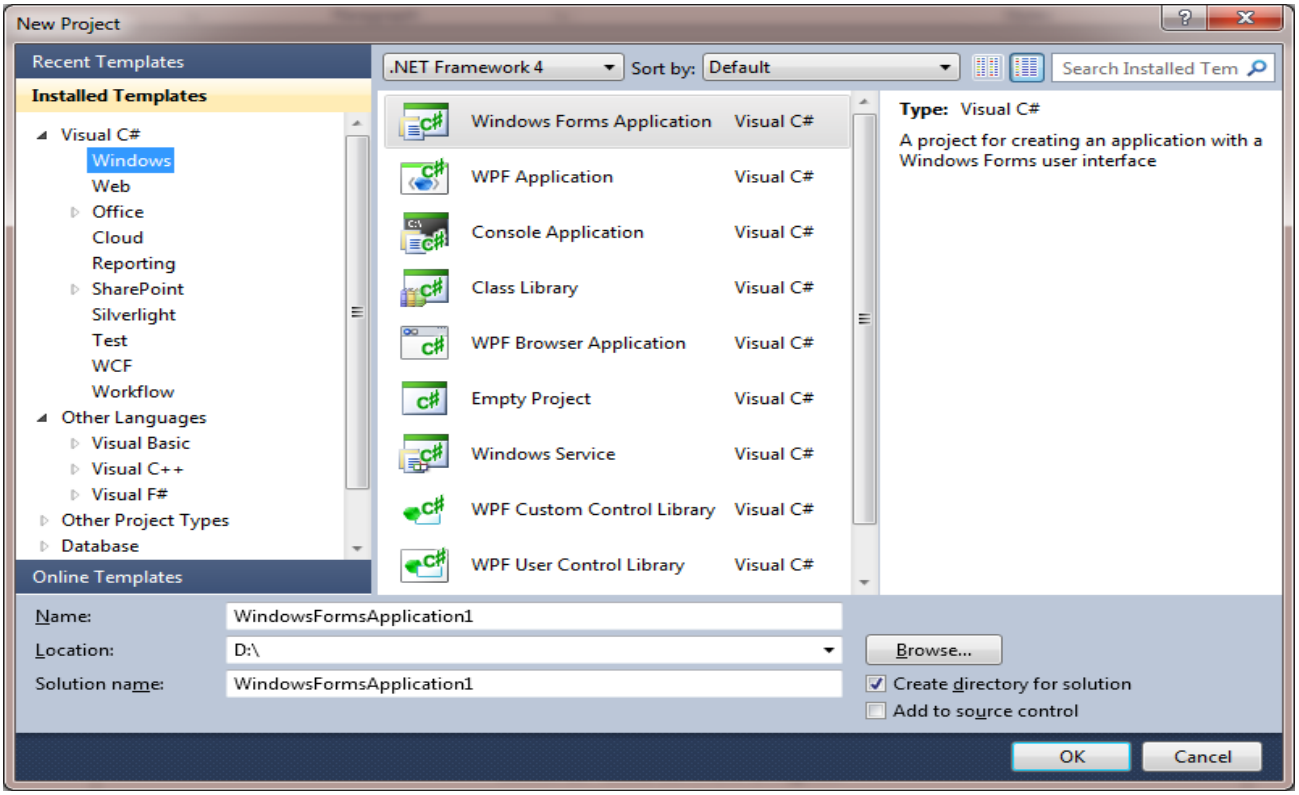
Hình 1. Cửa sổ Start Page

+ New Project: Tạo đồ án mới.

+ Open Project: Mở các đồ án có sẵn.

+ Recent Projects: Danh sách các đồ án gần đây nhất.

❷ Kích chọn mục New Project hoặc vào File/New/Project hoặc bấm phím tắt Ctrl+Shift+N xuất hiện cửa sổ *New Project*



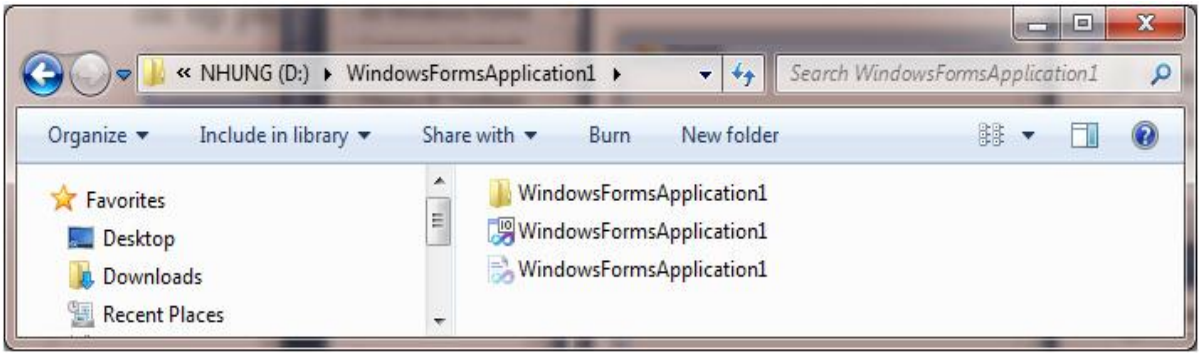
Hình 2. Cửa sổ New Project

+ Chọn ngôn ngữ *Visual C#* và ứng dụng *Windows*.

+ Đặt tên cho đồ án tại mục *Name*.

+ Chọn đường dẫn lưu đồ án tại mục *Location.*

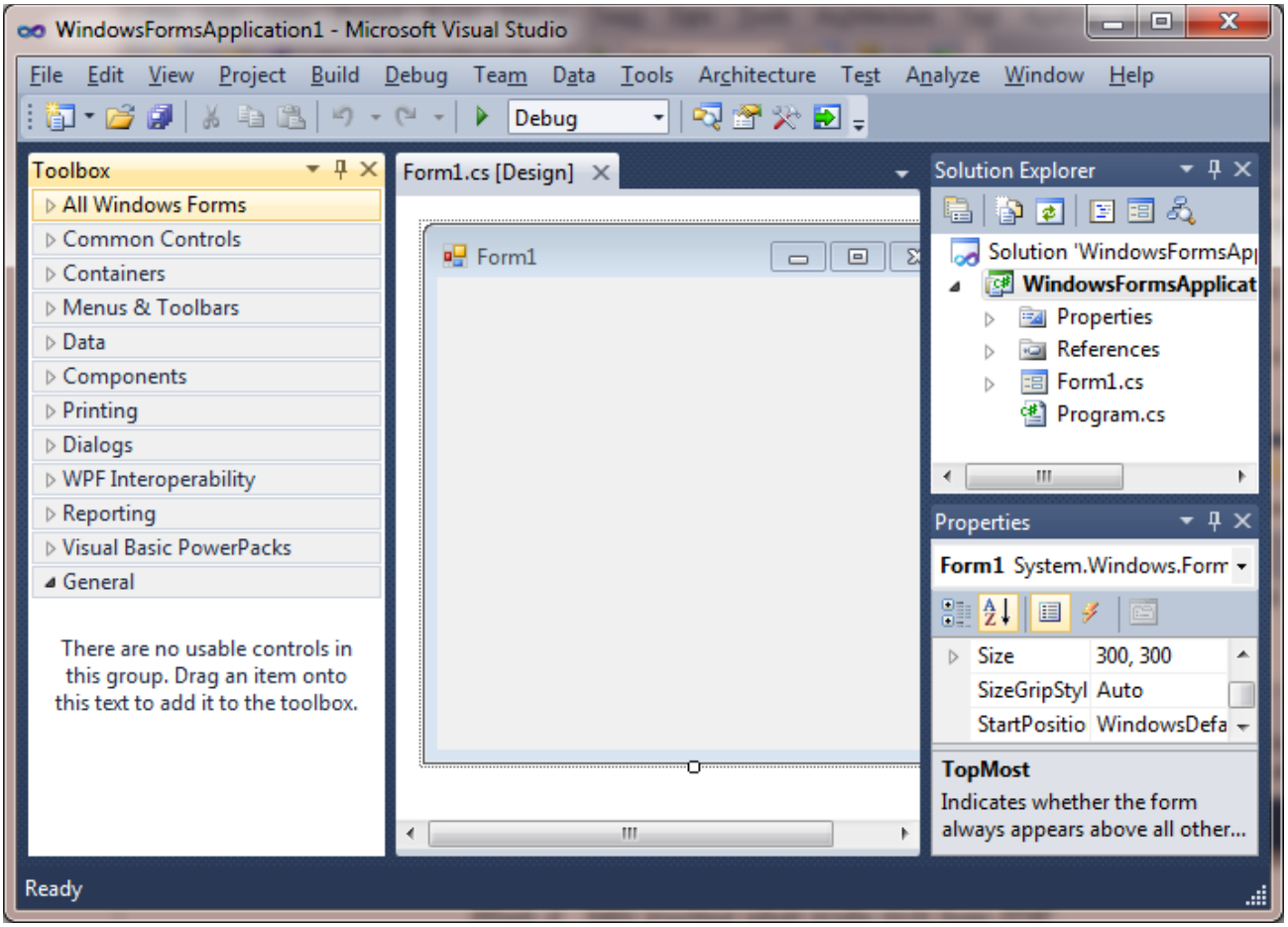
+ Mục *Create directory for solution* cho phép tạo một thư mục tại Location chứa tất cả các tệp phát sinh của đồ án (nếu không các tệp của đồ án sẽ được lưu tại Location).



Hình 3. Thư mục chứa đồ án

+ Chọn *OK* để tạo một đồ án mới.

Kết quả xuất hiện cửa sổ môi trường phát triển tích hợp IDE, với giao diện và các thành phần cơ bản như sau:



Hình 4. Môi trường phát triển tích hợp IDE

① **Title Bar**: Thanh tiêu đề chứa tên đồ án.

② **Menu Bar**: Thanh Menu chứa đầy đủ các công cụ cần để phát triển, thực thi và cài đặt ứng dụng…

+ *File*: cho phép mở, thêm mới và lưu trữ đồ án…
+ *Edit*: gồm các thao tác hỗ trợ việc soạn thảo mã lệnh như: copy, cắt, dán...
+ *View*: cho phép hiển thị các công cụ hỗ trợ người dùng trong quá trình xây dựng đồ án như:
    - Cửa sổ viết mã lệnh - Code
    - Form thiết kế - Designer

- Hộp công cụ - Toolbox
- Thanh công cụ - Toolbars
- Cửa sổ thuộc tính - Properties Window…

+ *Project*: cho phép bổ sung các đối tượng khác nhau vào đồ án như: các form, các component, các modul, các lớp…

+ *Built*: cho phép biên dịch đồ án.

+ *Debug*: cho phép chạy và gỡ rối chương trình.

+ *Data*: cho phép thêm mới và hiển thị cơ sở dữ liệu của đồ án.

+ *Tools*: cung cấp các công cụ cho phép kết nối tới các thiết bị ngoại vi như Pocket PC, Smartphone… hoặc kết nối tới các hệ quản trị cơ sở dữ liệu cũng như kết nối tới máy chủ server…

③ **Toolbar**: thanh công cụ gồm một tập hợp các nút lệnh, mỗi nút lệnh chứa một biểu tượng icons và có chức năng tương đương với chức năng của một mục lựa chọn trong thanh menu. Thanh công cụ rất hữu ích và trực quan, giúp người dùng dễ dàng và nhanh chóng thực hiện một chức năng mong muốn chỉ thông qua một cái kích chuột.

Visual C# 2010 có tới 39 thanh công cụ khác nhau như: Standard, Formatting, Debug, Build... Ví dụ hình ảnh thanh công cụ Standard:

Hình 5. Thanh công cụ Standard

Để gọi các thanh công cụ ta vào *View/Toolbars* khi đó sẽ xuất hiện danh sách tất cả các thanh công cụ. Muốn ẩn/hiện thanh công cụ nào ta kích chọn tại dòng chứa tên thanh công cụ đó.

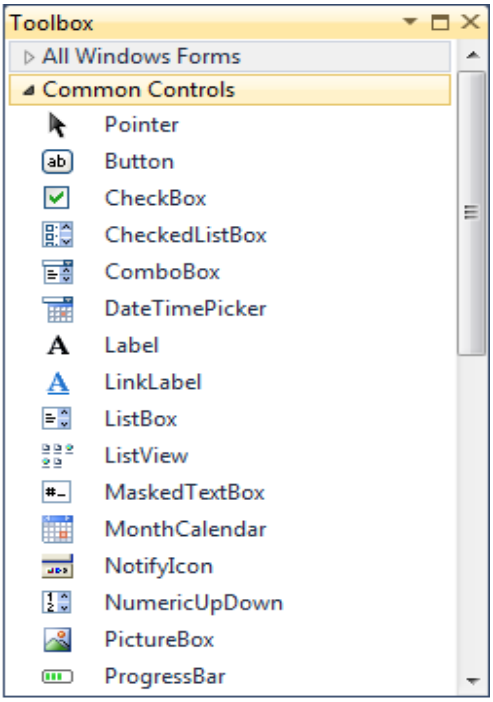
④ **Toolbox**: là hộp công cụ chứa các điều khiển – controls được đặt lên Form khi thiết kế giao diện người dùng.

Để hiển thị hộp công cụ ta thực hiện một trong các cách sau:

+ Vào *View/Toolbox*

+ Bấm tổ hợp phím *Ctrl+W+X*

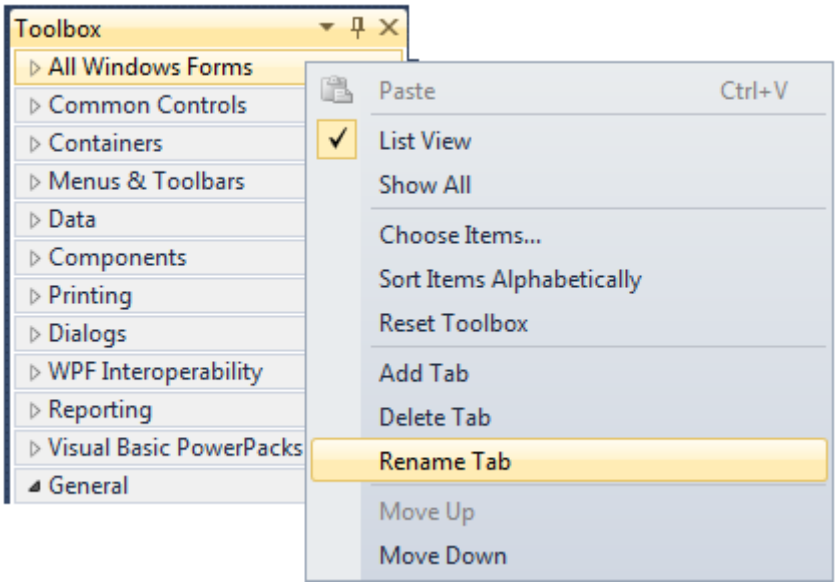
+ Kích chuột tại biểu tượng Toolbox  trên thanh công cụ Standard.



Hình 6. Hộp công cụ Toolbox

Mặc định hộp công cụ được chia thành 11 tab khác nhau như: *All Windows Forms*, *Common Controls*...

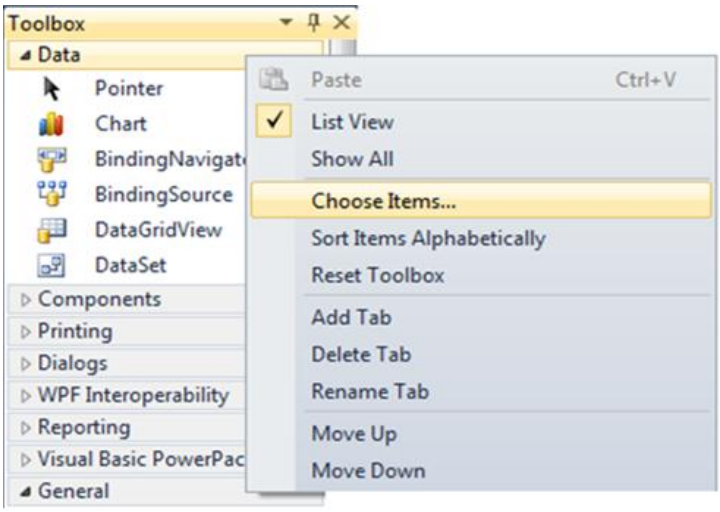
Ta có thể thêm mới, loại bỏ, đổi tên... các tab bằng cách kích chuột phải tại vị trí bất kỳ trên tab, xuất hiện một menu ngữ cảnh cho phép lựa chọn các thao tác cần thực hiện.



Hình 7. Các chức năng làm việc với từng tab trong Toolbox

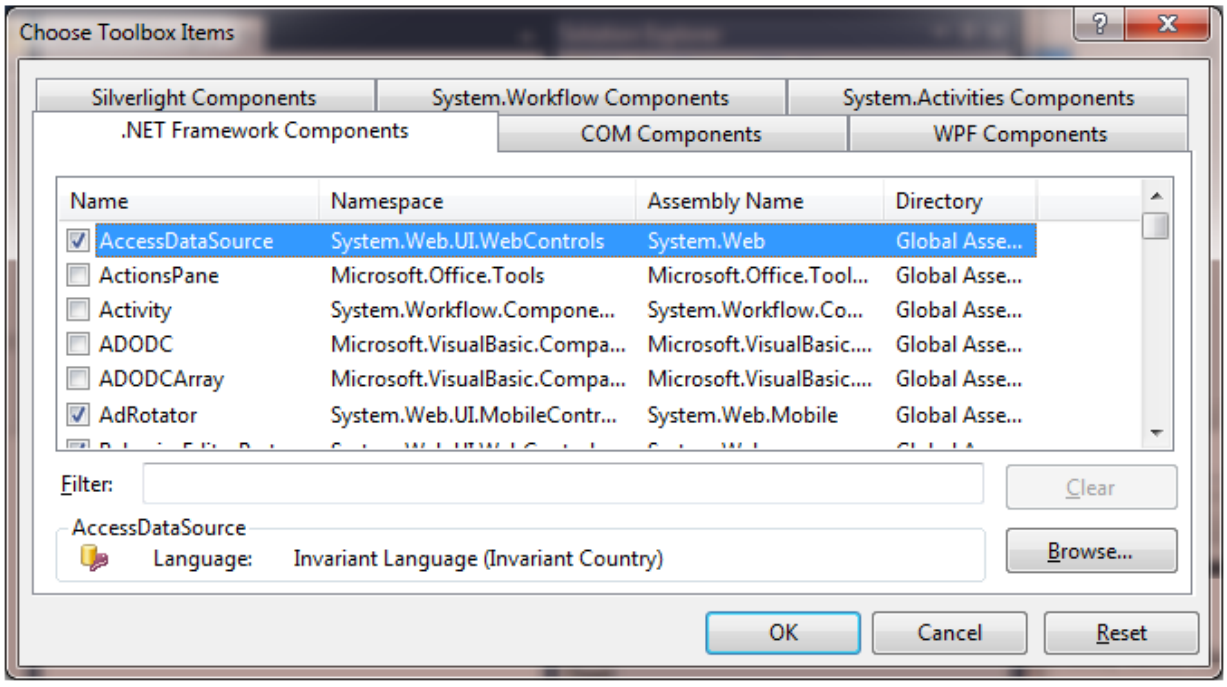
Trong mỗi tab của hộp Toolbox chứa danh sách các loại điều khiển khác nhau, các điều khiển này có thể thêm mới, loại bỏ, thay đổi vị trí… Kích chuột phải tại một điều khiển bất kỳ trên tab, xuất hiện một menu ngữ cảnh cho phép lựa chọn các thao tác cần thực hiện.

Ví dụ để thêm mới một điều khiển vào trong tab Data, ta kích chuột phải tại vị trí bất kỳ trên tab Data, chọn *Choose Items...*



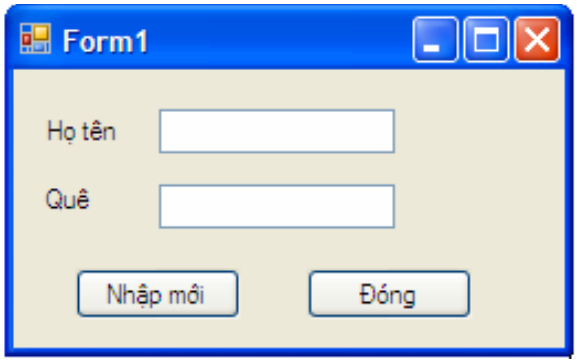
Hình 8. Các chức năng làm việc với từng điều khiển trong tab

Kết quả sẽ xuất hiện cửa sổ *Choose Toolbox Items*, kích chọn các điều khiển mong muốn rồi bấm OK để kết thúc.



Hình 9. Cửa sổ Choose Toolbox Items

⑤ **Form Designer**: cửa sổ thiết kế dùng để thiết kế giao diện cho chương trình, mỗi dự án có thể có một hoặc nhiều Form.



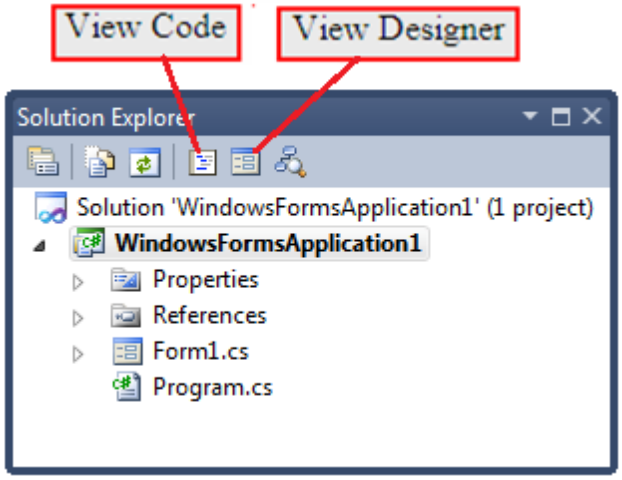
Hình 10. Cửa sổ Form Designer

⑥ **Solution Explorer**: cửa sổ giải pháp - đây là phần cửa sổ giúp ta quản lý tất cả các tài nguyên và tập tin dự án.

Solution Explorer được tổ chức thành một cấu trúc cây bao gồm những mục khác nhau, như: danh sách các Form của đồ án, danh sách các lớp Class, danh sách các tài nguyên cũng như danh sách cơ sở dữ liệu…

Để hiển thị cửa sổ Solution Explorer ta thực hiện một trong các cách sau:

+ Vào *View/Solution Explorer*
+ Bấm tổ hợp phím *Ctrl+W+S*
+ Kích chuột tại biểu tượng Solution Explorer  trên thanh công cụ Standard



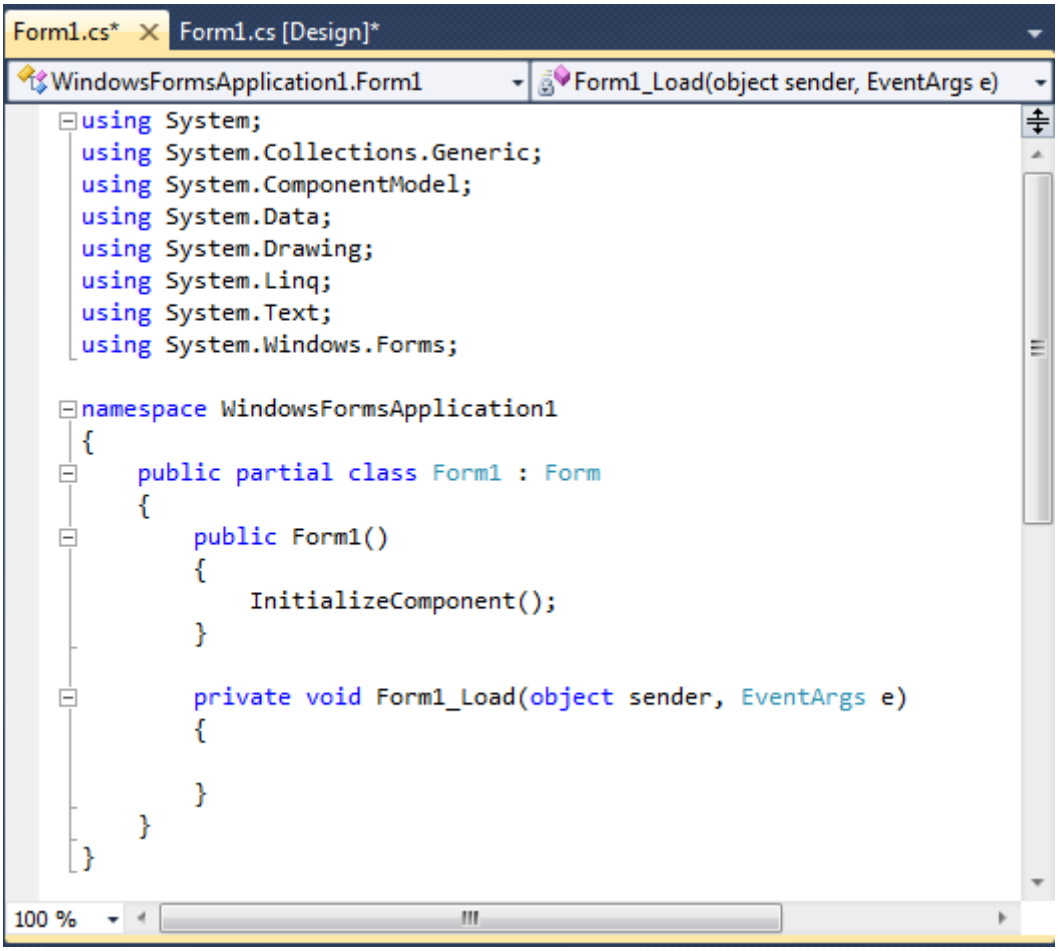
Hình 11. Cửa sổ Solution Explorer

Trong cửa sổ Solution Explorer có hai thành phần hay dùng là *View Code* và *View Designer*.

View Code: có tác dụng hiển thị cửa sổ soạn thảo mã lệnh cho Form đang được chọn. Ngoài ra, để hiển thị cửa sổ soạn thảo mã lệnh ta còn có một số cách khác như sau:

+ Vào *View/Code.*
+ Bấm phím tắt *F7.*
+ Kích đúp chuột tại cửa sổ thiết kế của form.

Giao diện cửa sổ soạn thảo như sau:



Hình 12. Cửa sổ soạn thảo

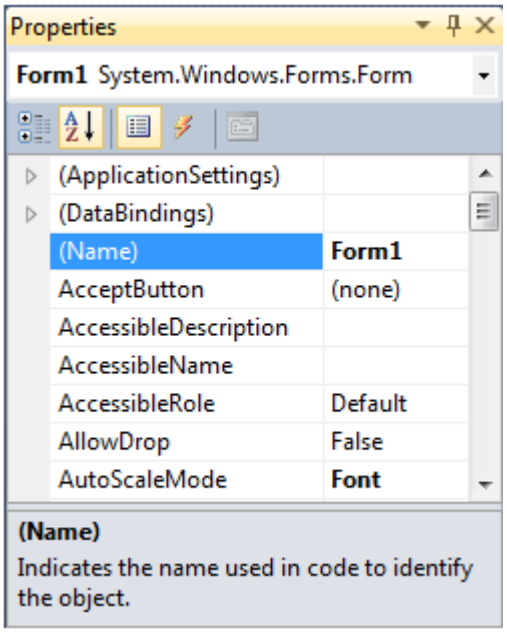
View Designer: có tác dụng hiển thị cửa sổ thiết kế giao diện của Form đang được chọn. Ngoài ra, để hiển thị cửa sổ thiết kế giao diện ta còn có một số cách khác như sau:

+ Vào *View/Designer*

+ Bấm phím tắt *Shift+F7*.

⑦ **Properties Window**: cửa sổ này liệt kê tất cả các thuộc tính, sự kiện của các điều khiển trong form.

Muốn hiển thị thuộc tính của đối tượng nào ta kích chuột chọn đối tượng đó trong cửa sổ thiết kế giao diện, hoặc chọn tên đối tượng trong danh sách thả xuống ở phần đầu của cửa sổ Properties.



Hình 13. Cửa sổ Properties

Mỗi thuộc tính có một giá trị mặc định, ta có thể thay đổi giá trị của các thuộc tính trực tiếp tại cửa sổ Properties trong lúc thiết kế, hoặc thay đổi bằng mã lệnh trong lúc thi hành chương trình.

Để hiển thị cửa sổ Properties ta thực hiện theo một trong các cách sau:

+ Vào *View\Properties Window*.

+ Kích chọn biểu tượng Properties Window trên thanh công cụ Standard.

+ Bấm phím tắt *Ctrl+W+P*

CHƯƠNG 2.

VIẾT CHƯƠNG TRÌNH ĐẦU TIÊN

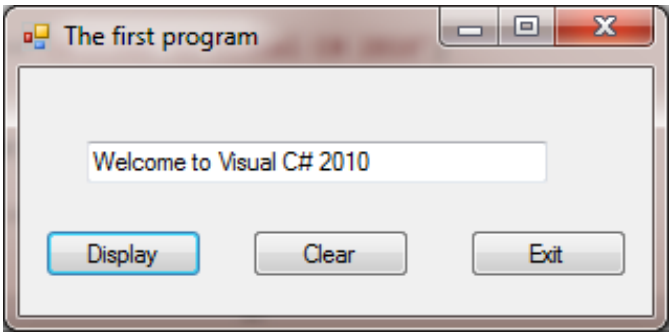
Một chương trình ứng dụng trong C# được thực hiện theo hai bước sau:

+ **Thiết kế giao diện.**
+ **Viết mã lệnh cho chương trình.**

1. Đề bài

Viết chương trình gồm 1 hộp văn bản Textbox và 3 nút lệnh Button: *Display, Clear, Exit* với các yêu cầu sau:

+ Kích chuột vào nút Display thì trong hộp văn bản xuất hiện dòng chữ: "Welcome to Visual C# 2010".
+ Kích chuột vào nút Clear thì nội dung trong hộp văn bản mất đi.
+ Kích chuột vào nút Exit để thoát khỏi chương trình quay lại cửa sổ soạn thảo.



Hình 14. Giao diện chương trình đầu tiên

3. Mở đồ án mới

Mở Microsoft Visual Studio 2010, chọn File/New/Project để khởi động một đồ án mới.

Chọn ngôn ngữ Visual C# và ứng dụng Windows, đặt tên cho đồ án tại mục Name là *Welcome* rồi chọn OK.

4. Thiết kế giao diện

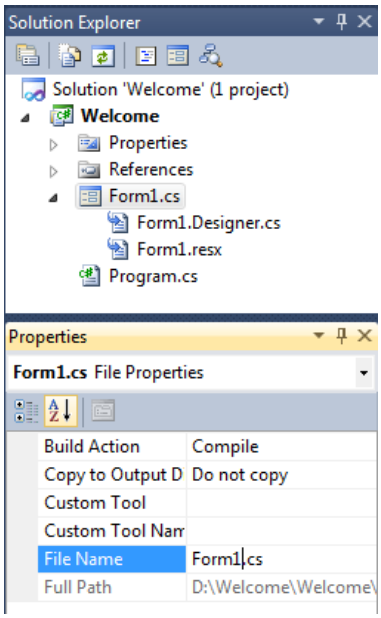
4.1. Đặt tên và tiêu đề cho form

Kích chuột vào vị trí bất kỳ trên Form, trong cửa sổ Properties sửa các thuộc tính:

+ Name: *frmWelcome*

+ Text: *The first program*

Trong cửa sổ *Solution Explorer* kích chuột tại *Form1.cs*, trong cửa sổ *Properties* sửa thuộc tính File Name là *frmWelcome.cs*



Hình 15. Đổi tên form trong cửa sổ Solution Explorer

4.2. Thêm điều khiển hộp văn bản Textbox

Kích chuột vào biểu tượng TextBox trên hộp công cụ Toolbox, giữ và kéo chuột để đặt Textbox vào Form. Ngoài ra ta có thể kích đúp chuột tại biểu tượng TextBox, điều khiển này sẽ được tự động đặt vào Form.

Khi đã có điều khiển TextBox trong Form ta có thể thay đổi vị trí và kích thước của Textbox cho phù hợp.

Khi mới xuất hiện trên Form hộp Textbox có tên mặc định là TextBox1, ta thay đổi giá trị này bằng cách kích chuột chọn điều khiển TextBox1, tại cửa sổ Properties chọn thuộc tính Name sửa thành *txtWelcome*.

4.3. Thêm điều khiển nút lệnh Button

Kích chuột vào biểu tượng Button, giữ và kéo chuột đưa điều khiển nút lệnh lên Form, nút lệnh này có tên mặc định là Button1 và nội dung cũng là Button1.

Thực hiện tương tự đưa thêm 2 nút lệnh Button2 và Button3 vào form.

Để thay đổi hai thuộc tính Name và Text của các nút lệnh ta thực hiện như sau:

+ Kích chuột vào nút lệnh 1, trong cửa sổ Properties sửa thuộc tính Name là *btnDisplay*, thuộc tính Text là *Display*

+ Kích chuột vào nút lệnh 2, sửa thuộc tính Name là *btnClear*, thuộc tính Text là *Clear*

+ Kích chuột vào nút lệnh 3, sửa thuộc tính Name là *btnExit*, thuộc tính Text là *Exit*

Chú ý: Mọi điều khiển đều có thuộc tính Name, để dễ dàng quản lý, gỡ rối chương trình ta nên đặt tên điều khiển tương ứng với chức năng của nó và có tiếp đầu ngữ chỉ loại điều khiển ở đầu.

Ví dụ: Textbox có tiếp đầu ngữ - txt, Button - btn, Form - frm… Các tiếp đầu ngữ được viết chữ thường, tên của điều khiển được viết hoa chữ cái đầu tiên, ví dụ: **txtWelcome**.

5. Viết code

5.1. Viết code cho nút lệnh btnDisplay

Ta mở cửa sổ soạn thảo Code Editor bằng cách kích đúp chuột vào nút Display. Trong cửa sổ Code, C# định nghĩa sẵn cho chúng ta một không gian tên - namespace đại diện cho form đang xét và 2 dòng mở đầu và kết thúc cho sự kiện Click của nút Display.

Gõ vào giữa thủ tục *btnDisplay_Click* dòng lệnh gán giá trị ‘Welcome to Visual C# 2010’ cho thuộc tính Text của điều khiển txtWelcome như sau:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Welcome
{
   public partial class frmWelcome : Form
   {
      public frmWelcome()
      {
         InitializeComponent();
      }
```

```
        private void btnDisplay_Click(object sender, EventArgs e)
        {
            txtWelcome.Text = "Welcome to Visual C# 2010";
        }
    }
}
```

Chú ý:

Hầu như tất cả các thủ tục xử lý một sự kiện nào đó của các điều khiển trên form đều có 2 đối số là sender và e. Trong đó:

+ Đối số *sender* có kiểu *object* đại diện cho đối tượng đã phát sinh sự kiện.

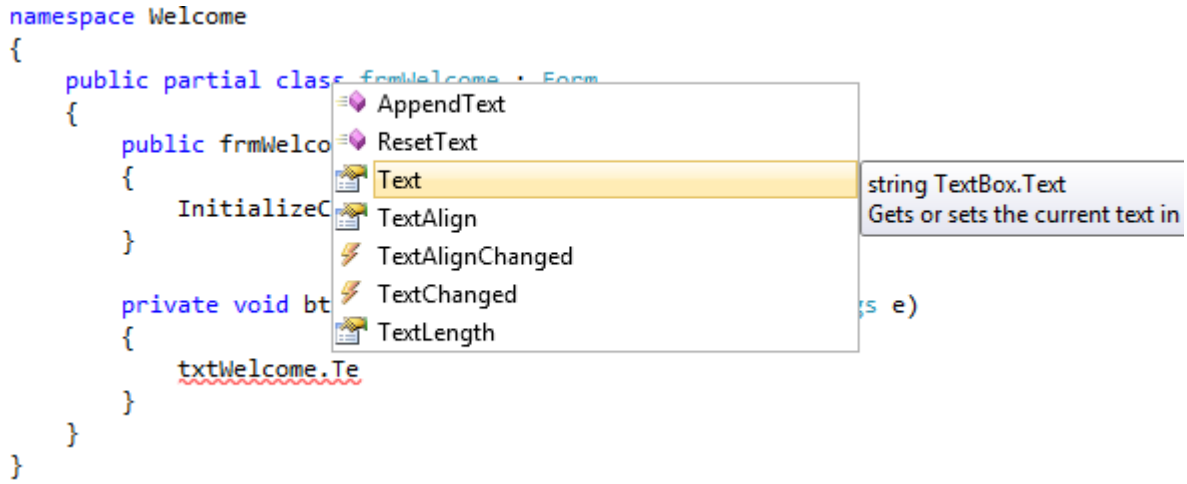
+ Đối số *e* có kiểu *EventArgs* chứa các thông tin về sự kiện như: vị trí chuột, thời gian phát sinh sự kiện…

Ta có cấu trúc chung để gán giá trị cho thuộc tính của một điều khiển khi viết mã lệnh như sau:

<Tên điều khiển>.<Thuộc tính> = <Giá trị>;

Các thuộc tính của các điều khiển trong C# rất phong phú, C# cung cấp tiện ích ***Intellisence*** tự động hiển thị một danh sách các thuộc tính của điều khiển sau khi ta gõ tên điều khiển và dấu chấm '.'

Để lựa chọn một thuộc tính, ta có thể dùng phím mũi tên lên, xuống để lựa chọn hoặc gõ các ký tự đầu của thuộc tính cần sử dụng, sau đó ấn phím Tab hoặc dấu cách để tự động chèn tên thuộc tính vào dòng lệnh.



Hình 16. Tiện ích Intellisence

Trong môi trường soạn thảo, nếu gõ sai cú pháp thì C# sẽ bắt lỗi ngay bằng cách hiển thị một đường gạch chân hình răng cưa dưới câu lệnh sai. Khi sửa xong lỗi thì đường răng cưa sẽ tự động biến mất.

5.2. Viết code cho nút lệnh btnClear

Quay lại cửa sổ thiết kế Design, kích đúp chuột vào nút Clear, gõ mã lệnh cho nút Clear như sau:

```
private void btnClear_Click(object sender, EventArgs e)
    {
      txtWelcome.Text = "";
    }
```

5.3. Viết code cho nút lệnh btnExit

Quay lại cửa sổ thiết kế Design, kích đúp chuột vào nút Exit, gõ mã lệnh như sau:

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

Lệnh Application.Exit() có tác dụng đóng và xoá ra khỏi bộ nhớ tất cả các cửa sổ đang được thực hiện trong chương trình và quay trở về cửa sổ thiết kế.

6. Lưu đồ án

Chọn File/Save All hoặc kích chọn biểu tượng  trên thanh công cụ Standard.

7. Các tệp tin của đồ án

Khi tạo đồ án, Visual Studio.NET sinh ra các tập tin sau:

+ *.sln*: đây là tập tin giải pháp (solution file), mỗi ứng dụng có một tập tin loại này nó bao gồm một hoặc nhiều tập tin dự án.

+ *.csproj*: đây là tập tin dự án (project file) của C#, mỗi tập tin dự án gồm một hoặc nhiều tập tin nguồn, các tập tin nguồn trong cùng một dự án phải được viết cùng một ngôn ngữ.

+ *.cs:* đây là tập tin nguồn (source file) của C# là nơi chứa mã lệnh của chương trình.

+ *AssemblyInfor.cs*: tập tin này cho phép thêm một số thuộc tính vào chương trình như: tên tác giả, ngày tạo chương trình…

8. Chạy chương trình

Để chạy một chương trình C# ta thực hiện theo 1 trong các cách sau:

+ Chọn Debug/Start Debugging.
+ Kích chuột vào biểu tượng Start Debugging  trên thanh công cụ Standard.
+ Bấm phím tắt F5.

9. Dừng chương trình

Khi chạy chương trình, nếu xuất hiện lỗi ta có thể dừng chương trình và quay về cửa sổ soạn thảo để sửa lỗi. Ta có các cách thực hiện sau:

+ Vào Debug/Stop Debugging.
+ Kích chuột vào biểu tượng Stop Debugging  trên thanh công cụ Standard.
+ Bấm phím tắt Ctrl+Alt+Break.

10. Mở đồ án đã có

Để mở một đồ án đã có ta có các cách thực hiện như sau:

+ Mở thư mục chứa đồ án, ví dụ thư mục Welcome trong đường dẫn "D:\", kích đúp vào tập tin Welcome.sln để mở đồ án Welcome.
+ Mở môi trường Microsoft Visual Studio 2010, trong cửa sổ Start Page kích chọn đồ án cần mở trong mục Recent Projects (nếu có).
+ Mở môi trường Microsoft Visual Studio 2010, vào File/Open Project, chọn đường dẫn đến tệp tin *.sln* của đồ án rồi chọn Open.

11. Thoát khỏi Visual C# 2010

+ Chọn File/Exit.

CHƯƠNG 3.

DỮ LIỆU VÀ CẤU TRÚC ĐIỀU KHIỂN

1. Biến, hằng và các kiểu dữ liệu

1.1. Biến

Biến là một đại lượng dùng để chứa dữ liệu tạm thời trong quá trình tính toán. Tất cả các biến được sử dụng trong chương trình đều phải được khai báo ngay từ đầu, biến được chia thành 3 loại bao gồm: Biến đầu vào, biến đầu ra và biến trung gian.

Biến có thể được khai báo tại 2 nơi gồm:

+ Bên trong phần định nghĩa lớp của Form.

+ Bên trong một phương thức.

```
public partial class frmWelcome : Form
    {
            // Nơi khai báo biến;
        private void bntDisplay_Click(object sender, EventArgs e)
        {
            // Nơi khai báo biến;
        }
    }
```

Phạm vi của một biến: phụ thuộc vào vị trí khai báo của biến, nếu biến được khai báo trong phần định nghĩa lớp của Form thì là biến toàn cục có tác dụng trong toàn bộ các đoạn mã lệnh của form, nếu biến được khai báo bên trong một phương thức thì là biến cục bộ chỉ có tác dụng trong phương thức chứa nó.

Biến giống như những chiếc hộp trong bộ nhớ có khả năng lưu giữ giá trị, có nhiều kiểu giá trị khác nhau mà C# có thể xử lý như: kiểu số nguyên, kiểu số thực, kiểu ký tự… Khi khai báo một biến ta phải chỉ ra kiểu giá trị mà nó sẽ lưu trữ.

Cú pháp khai báo biến như sau:

Kiểu dữ liệu *Tênbiến* **[**=*Giá trị*];

Tênbiến: là một chuỗi các ký tự do người lập trình tự đặt bao gồm các chữ cái, chữ số và dấu gạch dưới. Tên biến phải bắt đầu bằng một chữ cái, không được chứa dấu cách, C# phân biệt chữ hoa chữ thường.

1.2. Hằng

Hằng là đại lượng dùng để chứa những dữ liệu có giá trị không đổi trong suốt quá trình tính toán. Sử dụng hằng làm chương trình sáng sủa dễ đọc nhờ tên gọi gợi nhớ thay vì các con số.

Hằng được khai báo theo cú pháp sau:

const *Kiểu dữ liệu* **Tênhằng =** *Giá trị;*

1.3. Các kiểu dữ liệu

1.3.1. Kiểu số

int, long: lưu trữ các số nguyên, int có độ lớn 4 bytes, long có độ lớn 8 bytes.

float, double: biểu diễn các số thực, kiểu float dùng 4 bytes, double dùng 8 bytes.

Ví dụ khai báo biến số:

```
int     a; a=10;
long    b=10L;
float   c =19.34F;
double d=19.34;
```

Các phép toán số học:

+ *Cộng*: +,+=
+ *Trừ*: -, -=
+ *Nhân*: *, *=
+ *Chia:*/, /=. Ta có int/int=int, int/double=double, double/int=double, double/double=double Ví dụ: 8/5=1; 8.0/5=1.6

Chú ý: các phép toán x[+,-,*,/]=y tương đương với phép toán: x=x[+,-,*,/]y

+ *%*: phép chia lấy phần dư, ví dụ 5 % 2=1, 10.8 % 4 = 2.8
+ *Math.Round(x,n)*: hàm làm tròn số thực, ví dụ: Math.Round(11.346,2) = 11.35
+ *Math.Sin(x)*: hàm tính giá trị của sin(x)
+ *Math.Cos(x)*: hàm tính giá trị của cos(x)
+ *Math.Exp(x)*: hàm tính giá trị e^x

+ *Math.Pow(x,y)*: hàm tính giá trị x^y

+ *Math.Abs(x)*: hàm tính giá trị tuyệt đối của x

+ *Math.Sqrt(x):* hàm tính căn bậc hai của x

+ *Math.Floor(x):* hàm trả về số nguyên gần x nhất, ví dụ: Math.Floor(11.756) = 11

+ *Math.Truncate(x):* hàm trả về phần nguyên của x, ví dụ: Math.Truncate(11.756) = 11

  Ta có: Math.Floor(-11.756) = -12 và Math.Truncate(-11.756) = -11

Các phép toán so sánh: kết quả trả về của các phép toán so sánh có dạng đúng hoặc sai.

+ Lớn hơn: >

+ Nhỏ hơn: <

+ Lớn hơn hoặc bằng: >=

+ Nhỏ hơn hoặc bằng: <=

+ Bằng: ==

+ Khác: !=

1.3.2. Kiểu ký tự - char

char là kiểu dữ liệu chứa các ký tự trong bảng mã ASCII và được đặt trong cặp dấu nháy đơn.

Khai báo biến ký tự:

```
char    ch='+';
```

1.3.3. Kiểu chuỗi - string

string là một chuỗi các ký tự được đặt trong cặp dấu nháy kép. Trong VS.NET có hỗ trợ font Unicode nên ta có thể gõ tiếng Việt có dấu.

Ví dụ khai báo biến chuỗi:

```
string  s="Hà Nội mùa thu";
```

Các phép toán trên kiểu dữ liệu chuỗi

+ +: toán tử ghép chuỗi.

  Ví dụ: "Hà Nội " + "mùa thu" cho kết quả "Hà Nội mùa thu"

+ *s.Length:* trả về chiều dài của chuỗi s.

  Ví dụ: "Lâm Anh".Length có kết quả = 7

+ *s.Replace(str1,str2)*: thay thế chuỗi str1 trong chuỗi s bằng chuỗi str2.

  Ví dụ: "Hà Nội".Replace("Nội", "Tây") cho kết quả "Hà Tây"

+ *s.Substring(vt,n)*: trả về một chuỗi con gồm n ký tự trong chuỗi s, bắt đầu từ ký tự ở vị trí vt. (Chuỗi được tính từ vị trí 0).

  Ví dụ: "Hoa hồng ".Substring(1, 2) cho kết quả "oa"

+ *s.Insert(vt,str)*: chèn thêm giá trị của chuỗi str vào chuỗi s tại vị trí vt.

  Ví dụ: "Trời xanh". Insert(4, " màu") cho kết quả "Trời màu xanh"

+ *s.ToLower*: biến đổi chuỗi s về chữ in thường.

  Ví dụ: "Hà Nội".ToLower cho kết quả "hà nội"

+ *s.ToUpper*: biến đổi chuỗi s về chữ in hoa.

  Ví dụ: "Hà Nội".ToUpper có kết quả "HÀ NỘI"

+ *s.Remove(vt,n)*: xóa n ký tự trong chuỗi s, bắt đầu từ ký tự ở vị trí vt.

  Ví dụ: "Hoa hồng".Remove(1, 2) cho kết quả "H hồng"

+ *s.TrimStart*: xóa các ký tự rỗng ở đầu chuỗi s.

  Ví dụ: " Hoa hồng ".TrimStart cho kết quả "Hoa hồng "

+ *s.TrimEnd*: xóa các ký tự rỗng ở cuối chuỗi s.

  Ví dụ: " Hoa hồng ".TrimEnd cho kết quả " Hoa hồng"

+ *s.Trim*: xóa các ký tự rỗng ở đầu và cuối chuỗi s.

  Ví dụ: " Hoa hồng ".Trim cho kết quả "Hoa hồng"

+ *s.Split(ch)*: tách chuỗi s thành các chuỗi con ngăn cách nhau bởi ký tự ch.

  Ví dụ:
```
string  s = "Ha Noi";
        string[]  tu=s.Split(' ');
```
  Kết quả: tu[0]= "Ha", tu[1]= "Noi"

1.3.4. Kiểu logic bool

Kiểu bool là kiểu dữ liệu chỉ nhận một trong hai giá trị true/false.

Các phép toán trên kiểu dữ liệu bool

+ *Phép toán **Và &&***: xét biểu thức A && B chỉ nhận giá trị đúng khi và chỉ khi cả A và B cùng nhận giá trị đúng, còn nhận giá trị sai trong tất cả các trường hợp còn lại

+ *Phép toán **Hoặc ||***: xét biểu thức A || B chỉ nhận giá trị sai khi và chỉ khi cả A và B cùng nhận giá trị sai, còn nhận giá trị đúng trong tất cả các trường hợp còn lại.

+ *Phép toán **Phủ định !***: ta có !A nhận giá trị đúng khi A nhận giá trị sai và ngược lại.

Bảng giá trị chân lý của các phép toán:

A	B	A && B	A || B	A	!A
True	True	True	True	False	True
True	False	False	True	True	False
False	True	False	True		
False	False	False	False		

1.3.5. Kiểu ngày tháng DateTime

Ví dụ khai báo biến ngày tháng:

```
DateTime      d;
d = DateTime.Now;
```

Các phép toán trên kiểu dữ liệu DateTime

+ *DateTime.Now:* trả về ngày và giờ hiện hành, ví dụ: 09/02/2009 5:20:28PM

+ *Date.Day*: trả về giá trị ngày của Date, ví dụ: d.Day cho kết quả 09

+ *Date.Month*: trả về giá trị tháng của Date, ví dụ:d.Month cho kết quả 02

+ *Date.Year*: trả về giá trị năm của Date, ví dụ: d.Year cho kết quả 2009

+ *Date.AddDays(n)*: trả về một ngày mới cách ngày Date n ngày.

+ *Date.AddMonths(n)*: trả về một ngày mới cách ngày Date n tháng.

+ *Date.AddYears(n)*: trả về một ngày mới cách ngày Date n năm.

1.3.6. Kiểu dữ liệu ngẫu nhiên

C# cung cấp kiểu dữ liệu Random cho phép sinh các số ngẫu nhiên. Ví dụ khai báo biến ngẫu nhiên:

```
Random rnd;
```

Các phép toán trên kiểu dữ liệu ngẫu nhiên

+ *new Random():* khởi tạo bộ số ngẫu nhiên. Ví dụ: rnd=new Random();
+ *rnd.NextDouble()*: trả về một số thực ngẫu nhiên trong khoảng từ 0 đến 1.
+ *rnd.Next()*: trả về một số nguyên có giá trị bất kỳ.
+ *rnd.Next(n,m)*: trả về một số nguyên có giá trị bất kỳ trong khoảng từ n tới m (n>=0).

1.3.7. Kiểu dữ liệu mảng

Mảng là một tập hợp các biến có cùng tên và cùng kiểu dữ liệu. Dùng mảng làm chương trình đơn giản và ngắn gọn hơn. Mảng có cận trên, cận dưới và các thành phần trong mảng là liên tục giữa 2 cận.

Khai báo mảng: mảng được khai báo theo cú pháp sau:

```
Kiểu dữ liệu[] tên mảng ;
```

Để cấp phát bộ nhớ cho mảng ta dùng toán tử new theo sau là tên kiểu dữ liệu và kích thước của mảng được đặt trong cặp dấu ngoặc vuông.

Ví dụ khai báo mảng một chiều nguyên a gồm 10 phần tử :

```
int[] a = new  int[10] ;
```

Khai báo mảng 2 chiều thực b gồm 10 hàng, 5 cột

```
double[,]  b = new  double[10,5];
```

Khởi tạo giá trị cho các phần tử của mảng khi khai báo:

```
double[] a = new double[2] {34.56, -45};     hoặc   double[] a = {34.56, -45};

string[] Tennuoc = {"Anh", "Pháp", "Đức", "Việt Nam"};

int[,] a = {{4, 6, 9}, {5, 7, 9}, {12, 44, 23}};
```

1.4. Hàm chuyển đổi giữa các kiểu dữ liệu

Hàm chuyển đổi	Đổi giá trị sang kiểu
Convert.ToBoolean(Giatri)	Boolean
Convert.ToByte(Giatri)	Byte
Convert.ToDateTime(Giatri)	Date
Convert.ToDouble(Giatri)	Double
Convert.ToInt16(Giatri)	Integer – 2 byte
Convert.ToInt32(Giatri)	Integer – 4 byte
Convert.ToInt64(Giatri)	Integer – 8 byte
Convert.ToSingle(Giatri)	Single
Convert.ToString(Giatri)	String

Chú ý rằng giá trị truyền cho hàm phải hợp lệ, nghĩa là phải thuộc miền giá trị của kiểu kết quả nếu không C# sẽ báo lỗi. Ví dụ:

+ Convert.ToDateTime("25/04/79") trả về giá trị kiểu ngày tháng 25/04/79

+ Convert.ToInt32("25") = 25

+ Convert.ToInt32("25a") hoặc Convert.ToInt32("a25") sẽ báo lỗi.

2. Hộp thoại thông báo – MessageBox

2.1. Khái niệm

Hộp thông báo là hộp thoại cung cấp thông tin để tương tác với người sử dụng, đồng thời cũng là nơi hiển thị các kết quả trung gian trong quá trình tính toán.

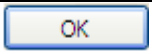
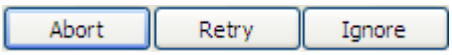
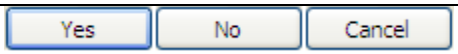
Trong thời gian hiển thị hộp thông báo, C# ngừng mọi hoạt động của biểu mẫu và người dùng chỉ có thể làm việc với hộp thông báo.

2.2. Hộp thông báo MessageBox

MessageBox.Show(*Nội dung thông báo, Tiêu đề, Kiểu chức năng, Kiểu biểu tượng);*

Chú ý: Cú pháp hộp thông báo không nhất thiết phải có đầy đủ bốn thành phần trên, nội dung cần thông báo và tiêu đề của hộp thông báo được đặt trong cặp dấu nháy kép.

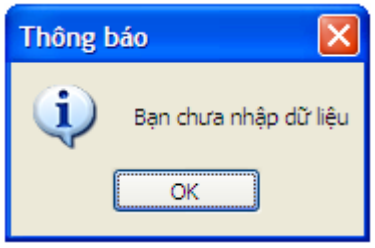
Kiểu chức năng và kiểu biểu tượng có các giá trị như sau:

Hằng tượng trưng	Thể hiện	Ý nghĩa
Các kiểu chức năng: được bắt đầu bởi *MessageBoxButtons*		
.OK		Chỉ hiển thị nút OK.
.OKCancel		Hiển thị các nút OK và Cancel.
.AbortRetryIgnore		Hiển thị các nút Abort, Retry và Ignore.
.YesNoCancel		Hiển thị các nút Yes, No và Cancel.
.YesNo		Hiển thị các nút Yes, No.
.RetryCancel		Hiển thị các nút Retry, Cancel.
Các kiểu biểu tượng: được bắt đầu bởi *MessageBoxIcon*		
.Error hoặc .Hand hoặc .Stop		Dùng cho những thông báo lỗi thất bại khi thi hành một việc nào đó.
.Question		Dùng cho những câu hỏi yêu cầu người dùng chọn lựa.
.Exclamation hoặc .Warning		Dùng cho các thông báo của chương trình.
.Asterisk hoặc .Information		Dùng cho các thông báo cung cấp thêm thông tin cho người dùng.
.None		Không hiển thị biểu tượng.

Ví dụ hiển thị hộp thông báo "Bạn chưa nhập dữ liệu" với một nút lệnh OK và biểu tượng Information ta viết như sau:

```
MessageBox.Show("Bạn chưa nhập dữ liệu", "Thông báo",
                                MessageBoxButtons.OK, MessageBoxIcon.Information)
```

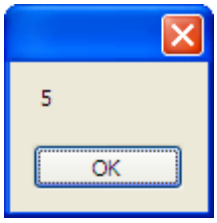
Kết quả ta có:



Chú ý: Trong khi chạy chương trình ta có thể hiển thị giá trị hiện thời của một biến bất kỳ bằng hộp thông báo như sau:

```
int a  = 5;
MessageBox.Show(a.ToString());
```

Kết quả xuất hiện hộp hội thoại sau:



2.3. Hàm thông báo MessageBox

Ngoài chức năng thông báo, hàm MessageBox còn trả về giá trị của các nút chức năng mà người dùng đã chọn. Cú pháp của hàm MessageBox như sau:

MessageBox.Show(*Nội dung thông báo, Tiêu đề, Kiểu chức năng, Kiểu biểu tượng) = Giá trị trả về*

Trong đó giá trị trả về gồm:

+ System.Windows.Forms.DialogResult.OK
+ System.Windows.Forms.DialogResult.Cancel
+ System.Windows.Forms.DialogResult.Abort
+ System.Windows.Forms.DialogResult.Retry
+ System.Windows.Forms.DialogResult.Ignore
+ System.Windows.Forms.DialogResult.Yes
+ System.Windows.Forms.DialogResult.No

Ví dụ, ta có thể viết lại code cho nút *btnThoat* với yêu cầu chỉ thoát khi người dùng trả lời có muốn thoát như sau:

```
private void btnThoat_Click(object sender, EventArgs e)
    {
      if (MessageBox.Show("Bạn có muốn thoát khỏi chương trình không?", "Thông báo",
                                MessageBoxButtons.YesNo, MessageBoxIcon.Question)==
                                                        System.Windows.Forms.DialogResult.Yes)
        Application.Exit();
    }
```

3. Các cấu trúc điều khiển

3.1. Câu lệnh lựa chọn if

Dạng 1:

```
if (Điềukiện)
      {
            Khối lệnh;
      }
```

Hoạt động: Nếu <Điều kiện> nhận giá trị đúng thì <Khối lệnh> được thực hiện.

Dạng 2:

```
if (Điều kiện)
   {
         Khối lệnh 1;
   }
else
   {
         Khối lệnh 2;
   }
```

Hoạt động: Nếu <Điều kiện> nhận giá trị đúng thì <Khối lệnh 1> được thực hiện, <Khối lệnh 2> bị bỏ qua. Ngược lại nếu <Điều kiện> nhận giá trị sai thì <Khối lệnh 2> được thực hiện, <Khối lệnh 1> bị bỏ qua.

3.2. Câu lệnh lựa chọn Case

```
switch (Biểu thức kiểm tra)
{
  case <Biểu thức 1>:
     Khối lệnh 1;
     break;
  case <Biểu thức 2>:
     Khối lệnh 2;
     break;
  ……..
  default:
     Khối lệnh n+1;
     break;
}
```

Hoạt động: máy so sánh giá trị của <Biểu thức kiểm tra> với giá trị của các <Biểu thức i>.

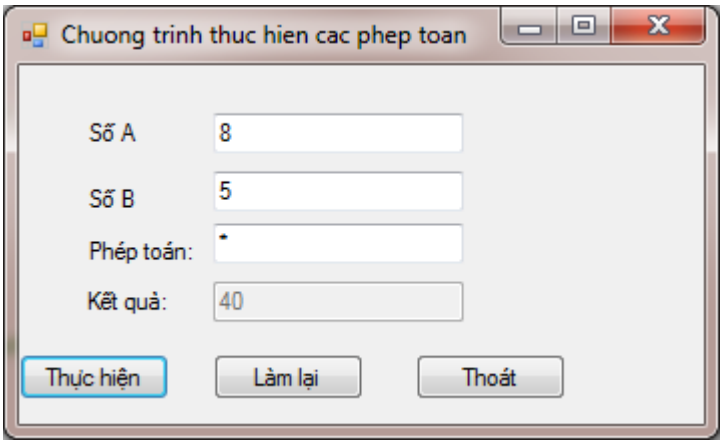
+ Nếu <Biểu thức kiểm tra> có giá trị thỏa mãn <Biểu thức i> thì <Khối lệnh i> được thực hiện, sau đó máy sẽ thoát ngay ra khỏi câu lệnh switch.
+ Nếu <Biểu thức kiểm tra> không thỏa mãn <Biểu thức i> nào thì <Khối lệnh n+1> được thực hiện.

Chú ý: nếu các biểu thức khác nhau cùng thực hiện chung một khối lệnh thì ta có thể viết gộp như sau:

```
switch (Biểu thức kiểm tra)
{
 case <Biểu thức 1>:
 case <Biểu thức 2>:
    Khối lệnh;
    break;
 ……..
 default:
    Khối lệnh n+1;
    break;
 }
```

Bài tập 1.

Xây dựng chương trình thực hiện các phép toán theo giao diện sau (các phép toán bao gồm: +, - , *, /, %).



Hình 17. Giao diện bài tập 1

Yêu cầu: + Nút thực hiện có tác dụng thực hiện phép toán đối với số A và số B, kết quả lưu vào ô kết quả.

+ Kết quả chỉ được tính khi người dùng nhập đủ giá trị cho số A, B và phép toán.

+ Phép toán chia phải kiểm tra trường hợp mẫu =0.

+ Ô kết quả không được phép chỉnh sửa dữ liệu.

Tạo dự án mới và thiết lập các thuộc tính của các điều khiển như sau:

Điều khiển	Thuộc tính	Giá trị
Form1	Name	frmPheptoan
	FormBorderStyle	Fixed3D
	Icon	Chọn file ảnh có đuôi .ico bất kỳ
	Text	Chuong trinh thuc hien cac phep toan
Lable1	Text	Số A
Lable2	Text	Số B
Lable3	Text	Phép toán:
Lable4	Text	Kết quả:
TextBox1	Name	txtSoA
TextBox2	Name	txtSoB
TextBox3	Name	txtPheptoan
TextBox4	Name	txtKetqua
Button1	Name	btnThuchien
	Text	&Thuchien
Button2	Name	btnLamlai
	Text	&Làm lại
Button3	Name	btnThoat
	Text	T&hoát

Mã lệnh:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication3
{
   public partial class frmPheptoan : Form
   {

      public frmPheptoan()
       {
          InitializeComponent();
       }

       private void Form1_Load(object sender, EventArgs e)
```

```
{
    txtKetqua.Enabled = false;
    btnLamlai.Enabled = false;
}

private void btnThuchien_Click(object sender, EventArgs e)
{
    int a, b, kq;
    if (txtSoA.Text == "")
    {
        MessageBox.Show("Bạn phải nhập A", "Thông báo", MessageBoxButtons.OK,
                                                        MessageBoxIcon.Information);
        txtSoA.Focus();
        return;
    }
    if (txtSoB.Text == "")
    {
        MessageBox.Show("Bạn phải nhập B", "Thông báo", MessageBoxButtons.OK,
                                                        MessageBoxIcon.Information);
        txtSoB.Focus();
        return;
    }
    a = Convert.ToInt16(txtSoA.Text);
    b = Convert.ToInt16(txtSoB.Text);
    switch (txtPheptoan.Text)
    {
        case "+":
            kq=a+b;
            txtKetqua.Text = kq.ToString();
            break;
        case "-":
            kq=a-b;
            txtKetqua.Text = kq.ToString();
            break;
        case "*":
            kq = a * b;
            txtKetqua.Text = kq.ToString();
            break;
        case "/":
            if (txtSoB.Text == "0")
            {
                MessageBox.Show("Giá trị B phải khác 0!", "Thông Báo",
                                                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                txtSoB.Text = "";
```

```
                txtSoB.Focus();
                return;
            }
            kq = a / b;
            txtKetqua.Text = kq.ToString();
            break;
        case "%":
            kq = a % b;
            txtKetqua.Text = kq.ToString();
            break;
        default:
            MessageBox.Show("Bạn phải nhập lại phép toán", "Thông Báo",
                                                MessageBoxButtons.OK, MessageBoxIcon.Error);
            txtPheptoan.Text = "";
            txtPheptoan.Focus();
            return;
    }
    btnThuchien.Enabled = false;
    btnLamlai.Enabled = true;
    txtSoA.ReadOnly =true ;
    txtSoB.ReadOnly = true ;
    txtPheptoan.ReadOnly = true;
}

private void btnLamlai_Click(object sender, EventArgs e)
{
    txtSoA.Text = "";
    txtSoA.ReadOnly = false;
    txtSoB.Text ="";
    txtSoB.ReadOnly = false;
    txtPheptoan.Text = "";
    txtPheptoan.ReadOnly = false;
    txtKetqua.Text = "";
    btnThuchien.Enabled = true;
    txtSoA.Focus();
}

private void btnThoat_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Bạn có muốn thoát không?", "Thông báo",
                                    MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
                                                    System.Windows.Forms.DialogResult.Yes)
        Application.Exit();
}
}
}
```

3.3. Cấu trúc lặp for

Cho phép thực hiện lặp đi lặp lại một đoạn chương trình nhiều lần, với số lần lặp xác định.

```
for (biểu thức khởi tạo; biểu thức điều kiện; biểu thức cập nhật)
  {
      Khối lệnh;
      [break;]
  }
```

Hoạt động: Đầu tiên máy thực hiện biểu thức khởi tạo để khởi tạo giá trị của biến điều khiển, sau đó máy kiểm tra giá trị của biểu thức điều kiện, nếu biểu thức này đúng thì <Khối lệnh> được thực hiện và cập nhật giá trị của biến điều khiển thông qua biểu thức cập nhật. Sau đó quay lại kiểm tra giá trị của biểu thức điều kiện, cứ lặp lại như vậy cho đến khi biểu thức điều kiện nhận giá trị sai thì dừng lại.

Chú ý: để thoát ngay ra khỏi vòng lặp for ta có thể dùng lệnh break.

Ví dụ: Dùng vòng lặp for để khởi tạo các giá trị ngẫu nhiên trong khoảng (0, 100) cho mảng một chiều gồm 10 phần tử.

Mở một dự án mới rồi gõ đoạn mã sau vào cửa sổ code.

```
private void Form1_Load(object sender, EventArgs e)
      {
        int[] m = new int[10];
        Random rnd = new Random();
        for(int i = 0; i<10; i++)
        {
          m[i] = rnd.Next(0,100);
          MessageBox.Show(m[i].ToString());
        }
      }
```

Bấm F5 để thực hiện chương trình.

3.4. Cấu trúc lặp while

Cho phép thực hiện lặp đi lặp lại một đoạn chương trình nhiều lần, với số lần lặp không được xác định trước.

```
while (Biểu thức điều kiện)
{
   Khối lệnh;
   [break;]
}
```

Hoạt động: Đầu tiên máy kiểm tra giá trị của <Biểu thức điều kiện>, nếu biểu thức này nhận giá trị đúng thì <Khối lệnh> được thực hiện. Sau đó lại quay lại kiểm tra giá trị của <Biểu thức điều kiện>, cứ lặp lại như vậy cho đến khi <Biểu thức điều kiện> nhận giá trị sai thì dừng lại.

Chú ý: + Vì <Biểu thức điều kiện> được kiểm tra trước, nên <Khối lệnh> có thể không được thực hiện lần nào nếu ngay từ đầu <Biểu thức điều kiện> đã nhận giá trị sai và trước khi thực hiện khối lệnh phải khởi gán giá trị cho <Biểu thức điều kiện>.

+ Trong <Khối lệnh> phải có ít nhất một lệnh làm thay đổi giá trị của <Biểu thức điều kiện> để đến một lúc nào đó <Biểu thức điều kiện> nhận giá trị sai, nhằm dừng vòng lặp lại, nếu không nó sẽ lặp mãi không dừng.

Ví dụ: Nhập số thực a, tìm số tự nhiên n nhỏ nhất sao cho tổng: T = 1+1/2+…+1/n >= a

Mở một dự án mới rồi gõ đoạn mã sau vào cửa sổ code.

```
private void Form1_Load(object sender, EventArgs e)
    {
       double a= 2;
       double T= 0;
       int n = 0;
       while (T<a)
       {
            n = n + 1;
            T = T + 1.0/n;
       }
       MessageBox.Show("Gia tri n thoa man = " + n.ToString ());
    }
```

Bấm F5 thực hiện chương trình, có thể thay đổi giá trị của a để có những kết quả khác nhau.

3.5. Cấu trúc lặp do

Cho phép thực hiện lặp đi lặp lại một đoạn chương trình nhiều lần, với số lần lặp không được xác định trước.

```
do
 {
    Khối lệnh;
    [break;]
 }
while (Biểu thức điều kiện);
```

Hoạt động: Đầu tiên máy thực hiện <Khối lệnh>, sau đó kiểm tra giá trị của <Biểu thức điều kiện>, nếu biểu thức này nhận giá trị đúng thì tiếp tục thực hiện <Khối lệnh>, cứ lặp lại như vậy cho đến khi <Biểu thức điều kiện> nhận giá trị sai thì dừng lại.

Chú ý: + Vì <Biểu thức điều kiện> được kiểm tra sau, nên <Khối lệnh> luôn được thực hiện ít nhất 1 lần.

+ Trong <Khối lệnh> phải có ít nhất một lệnh làm thay đổi giá trị của <Biểu thức điều kiện> nhằm dừng vòng lặp lại.

Ví dụ: Tính tổng T = 1 + 2 + … + 10

Mở một dự án mới rồi gõ đoạn mã sau vào cửa sổ code.

```
private void Form1_Load(object sender, EventArgs e)
    {
      int T = 0;
      int i = 1;
      do
      {
        T = T + i;
        i = i + 1;
      }
      while (i <= 10);
      MessageBox.Show("T = " + T.ToString ());
    }
```

3.6. Câu lệnh try…catch

Được dùng trong các câu lệnh bẫy lỗi của chương trình, cho phép bắt một số lỗi trong quá trình thực thi ứng dụng, ví dụ: biến các ký tự không phải dạng số thành số, thực hiện phép chia cho 0, sử dụng biến null…

Cú pháp bẫy lỗi trong C# được thể hiện như sau:

```
try
{
  // mã cho việc thực thi bình thường
}
catch (System.Exception)
{
  // xử lí lỗi
}
finally
{
  // dọn dẹp
}
```

Cú pháp trên gồm 3 khối:

+ Khối *try* chứa đựng đoạn mã cần phải thực thi trong chương trình, nhưng đoạn mã này có thể gặp phải một vài trạng thái lỗi.

+ Khối *catch* chứa đựng đoạn mã giải quyết những những lỗi xảy ra trong *try*, tham số của *catch* là các lớp bắt lỗi. C# có rất nhiều lớp bắt lỗi, trong đó *System.Exception* là lớp ở mức cao nhất có thể bắt được mọi loại lỗi xảy ra trong *try*.

+ Khối *finally* chứa đựng đoạn mã dọn dẹp tài nguyên hoặc bất kì hành động nào bạn muốn thực hiện sau khối *try* hay *catch*, khối này có thể có hoặc không.

Hoạt động: Đầu tiên chương trình thực thi các câu lệnh trong khối *try*, nếu không xuất hiện lỗi thì các câu lệnh được thực hiện bình thường sau đó sẽ nhảy đến thực hiện các câu lệnh trong khối *finally*, tuy nhiên nếu xuất hiện lỗi trong khối *try* thì chương trình sẽ tự động nhảy ngay tới thực thi các câu lệnh trong khối *catch* mà không đột ngột dừng chương trình và sau đó cũng thực hiện các câu lệnh trong khối *finally*.

Chú ý: C# không cho phép đặt lệnh `return` bên trong khối *finally*.

Ví dụ: Bẫy lỗi đoạn chương trình tính tổng 2 số nguyên a và b trong trường hợp không nhập dữ liệu dạng số.

```
private void bntTong_Click(object sender, EventArgs e)
    {
      try
      {
        txtTong.Text = Convert.ToString(Convert.ToInt32(txtSoA.Text)
                                                    + Convert.ToInt32(txtSoB.Text));
      }
      catch (System.Exception ex)
      {
        MessageBox.Show(ex.ToString());
       //MessageBox.Show("Bạn phải nhập dữ liệu số!");
      }
    }
```

4. Hàm

4.1. Hàm có một giá trị trả về

Cú pháp xây dựng:

```
public|private  Kiểudữliệutrảvề <Tên CTC>([Tham số])
{
    Khai báo các biến cục bộ;
    Kiểudữliệutrảvề    BiếnTG;
```

```
        Tính toán kết quả thông qua BiếnTG;
        return BiếnTG;
}
```

Khi xây dựng hàm có giá trị trả về ta thường khai báo thêm một biến trung gian có kiểu trùng với kiểu dữ liệu trả về và công thức tính toán sẽ được tính thông qua biến trung gian này. Giá trị của biến trung gian sẽ được gán vào cho tên hàm thông qua lệnh return.

Lệnh return có thể được đặt tại vị trí bất kỳ, khi gặp lệnh return chương trình gán giá trị của biến đi kèm sau lệnh return cho tên hàm và thoát ngay ra khỏi hàm.

4.2. Hàm không có giá trị trả về

Cú pháp xây dựng:

```
public|private  void <Tên CTC>([Tham số])
{
  Các dòng lệnh;
}
```

4.3. Cách gọi hàm

Hàm có giá trị trả về: được sử dụng như một thành phần của biểu thức, điều đó có nghĩa nó có thể được dùng trong lệnh gán và trong các biểu thức so sánh.

```
Biến =  <Tên CTC>([Tham số]);
```

Ví dụ, nếu ta có hàm: *private double MyFunction(int a, int b)*

Thì ta gọi hàm như sau: **x = MyFunction(n, m);**

Trong đó x là biến có kiểu double, n và m là hai biến int tương ứng với a và b.

Hàm không có giá trị trả về: được dùng như một câu lệnh độc lập, nó không được dùng trong lệnh gán hoặc trong các biểu thức so sánh.

```
<Tên thủ tục>([Tham số]);
```

Giả sử ta có hàm: *private void MySub(int a, int b)*

Khi đó hàm được gọi như sau: **MySub (a, b);**

Chú ý: Các tham số khi xây dựng hàm được gọi là tham số hình thức, các tham số khi sử dụng hàm gọi là tham số thực sự, hai loại tham số này phải tương ứng nhau về: số lượng, thứ tự và kiểu dữ liệu.

4.4. Ví dụ minh họa

Viết và thực hiện hàm tính k! và hàm bẫy lỗi chia cho 0. Mở một dự án mới rồi gõ đoạn mã sau vào cửa sổ code.

```
namespace WindowsFormsApplication1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
   double a;

   private double Giaithua(int k )
   {
      double gt;
      int i;
      gt = 1;
      for(i=1;i<=k;i++)
         gt = gt * i;
      return gt;
   }

   private void Division()
   {
     int i=0;
     int n;
     try
     {
         n = 4 / i;
         MessageBox.Show(n.ToString());
     }
     catch (System.Exception ex)
     {
        MessageBox.Show(ex.ToString());
     }
   }

  private void Form1_Load(object sender, EventArgs e)
      {
         a = Giaithua(6);
         MessageBox.Show(a.ToString());
         Division();
      }
  }
}
```

5. Gỡ rối chương trình

Ở trong những giai đoạn đầu lập trình các chương trình không tránh khỏi có sai sót và mắc lỗi, tuy nhiên ta có thể giảm khả năng mắc lỗi đến mức tối thiểu.

5.1. Một số giải pháp giảm lỗi

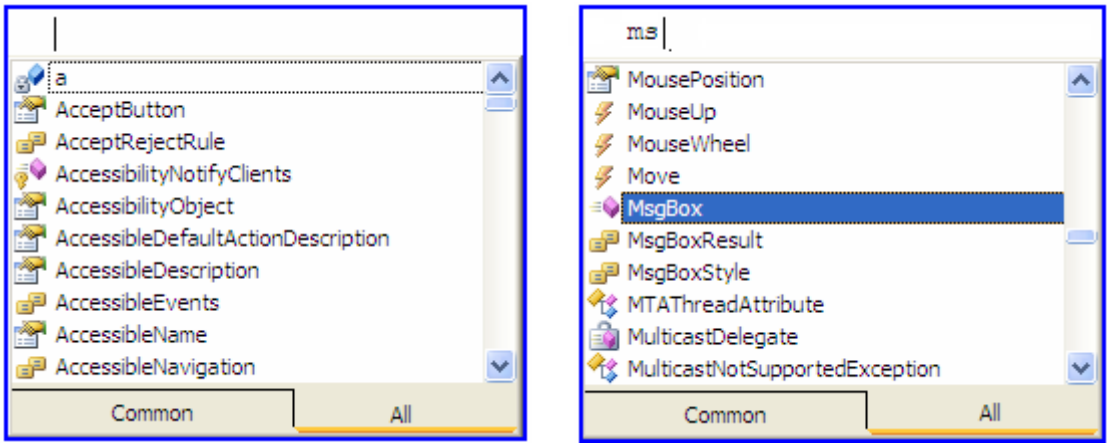
+ Thiết kế cẩn thận, ghi chú các vấn đề quan trọng và cách giải quyết cho từng phần. Ghi chú từng thủ tục và mục đích của nó.

+ Chú thích rõ ràng trong chương trình (trong C# dòng chú thích được bắt đầu bởi 2 dấu gạch chéo //).

  Để đặt chú thích cho các dòng lệnh, ta có thể gõ 2 dấu gạch chéo // tại vị trí cần đặt chú thích, hoặc bôi đen các dòng lệnh rồi kích chọn biểu tượng trên thanh công cụ Standard. Để xóa dấu chú thích ở các dòng lệnh, ta có thể xóa 2 dấu gạch chéo // hoặc bôi đen các dòng lệnh muốn xóa dấu chú thích rồi kích chọn biểu tượng trên thanh công cụ Standard.

+ Dùng cửa sổ danh sách các thuộc tính, các phương thức, các hằng số, các lớp đối tượng… trong C# để tránh việc gõ sai tên thuộc tính, phương thức…

  Để gọi cửa sổ này, trong cửa sổ soạn thảo code bấm tổ hợp phím Ctrl+J, kết quả khi người dùng gõ các ký tự bất kỳ, con trỏ sẽ tự động cuộn tới dòng đầu tiên chứa các ký tự đó cho người dùng chọn.



Hình 18. Cửa sổ danh sách thuộc tính, phương thức, sự kiện.

+ Sử dụng câu lệnh try…catch để bắt các lỗi ngoại lệ của chương trình.

CHƯƠNG 4.

TÌM HIỂU CÁC ĐIỀU KHIỂN CƠ BẢN

1. Tìm hiểu thuộc tính, phương thức và sự kiện

Mỗi một đối tượng trong C# đều có 3 đặc tính là *Thuộc tính - Properties, Phương thức - Methods và Sự kiện – Events.* Trong đó:

+ **Properties:** là tập hợp các thuộc tính để mô tả một đối tượng như: tên, chiều cao, chiều rộng, mầu chữ, mầu nền… Các thuộc tính có thể xác định trong khi thiết kế (Design time) hoặc trong lúc thi hành (Run time).

+ **Methodes:** là những đoạn chương trình chứa trong điều khiển, cho điều khiển biết cách thức để thực hiện một công việc nào đó, chẳng hạn làm ẩn sự xuất hiện của một điều khiển (phương thức Hide).

+ **Evens:** nếu thuộc tính mô tả đối tượng, phương thức chỉ ra cách thức đối tượng hành động thì sự kiện là những phản ứng của đối tượng. Khi tạo một chương trình trong C#, ta lập trình chủ yếu theo sự kiện, lập trình theo cách này có nghĩa là ta phải biết khi nào sự kiện xảy ra và làm gì khi sự kiện đó xảy ra? Điều này có nghĩa là chương trình chỉ thi hành khi người dùng thực hiện một thao tác nào đó trên giao diện.

2. Mối quan hệ giữa thuộc tính, phương thức và sự kiện

Mặc dù thuộc tính, phương thức và sự kiện có vai trò khác nhau nhưng chúng thường xuyên liên hệ với nhau. Ví dụ, nếu ta di chuyển một điều khiển bằng phương thức Move thì một số thuộc tính như Top, Height, Left, Width sẽ thay đổi theo theo, khi đó kích cỡ của điều khiển thay đổi tức là sự kiện Resize xảy ra.

Phụ thuộc lẫn nhau còn có nghĩa là ta có thể thực hiện một công việc bằng nhiều cách: xử lý trên thuộc tính hoặc xử lý bằng phương thức.

Ví dụ: ta có 2 cách để làm hộp văn bản textBox1 xuất hiện và biến mất trên màn hình:

+ Thực hiện bằng thuộc tính:

    Xuất hiện: `textBox1.Visible = true;`
    Biến mất: `textBox1.Visible = false;`

+ Thực hiện bằng phương thức:

    Xuất hiện: `textBox1.Show();`

Biến mất: textBox1.Hide();

3. Thuộc tính, phương thức, sự kiện của một số điều khiển cơ bản

3.1. Form

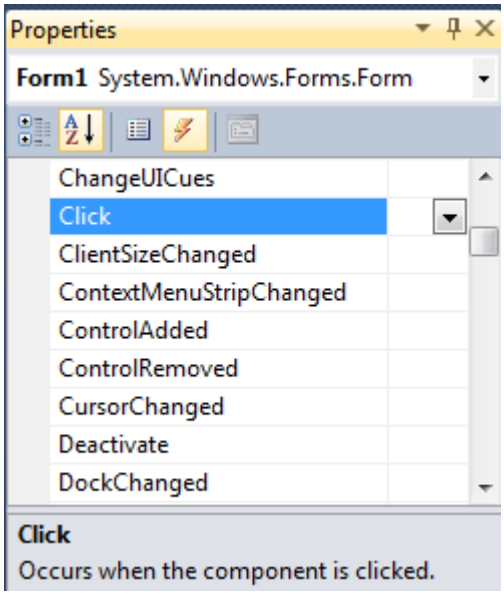
3.1.1. Thuộc tính

Name	Tên form, bắt đầu bởi tiếp đầu ngữ **frm**
BackColor	Thiết lập mầu nền cho Form.
BackgroundImage	Thiết lập ảnh nền cho Form.
BackgroundImageLayout	Thiết lập chế độ hiển thị ảnh nền trên Form. Tile: hiển thị ảnh từ trên xuống, Center: hiển thị ảnh từ giữa ra, Stretch: dãn đều ảnh trên Form.
AcceptButton	Thiết lập nút lệnh Accept. Sự kiện Click của nút lệnh này được gọi bất cứ khi nào người dùng bấm phím Enter.
CancelButton	Thiết lập nút lệnh Cancel. Sự kiện Click của nút lệnh này được gọi bất cứ khi nào người dùng bấm phím Esc.
Cursor	Thiết lập chế độ hiển thị con trỏ trên Form.
Enabled	Nếu nhận giá trị True thì cho phép người dùng tác động lên Form, ngược lại thì nhận giá trị False.
Font	Thiết lập kiểu chữ, cỡ chữ cho các điều khiển trên Form.
ForeColor	Thiết lập mầu chữ cho các điều khiển trên Form.
FormBorderStyle	Thiết lập kiểu đường viền cho Form. Fixed Single: không thể thay đổi kích thước của Form, Sizable: có thể phóng to thu nhỏ và thay đổi kích thước của Form, Sizable ToolWindow: có thể thay đổi kích thước của Form…
Icon	Thiết lập biểu tượng cho Form (các tệp ảnh có đuôi .ico).
MainMenuStrip	Gắn kết Form với Menu.
Opacity	Thiết lập độ trong suốt cho nền của Form, nếu độ trong suốt < 100% thì có thể nhìn xuyên thấu những gì nằm bên dưới Form.
ShowIcon	Nếu nhận giá trị True thì cho phép hiển thị biểu tượng đã được thiết lập ở thuộc tính Icon, ngược lại thì nhận giá trị False.
StartPosition	Thiết lập vị trí xuất hiện của Form trên màn hình. Manual: xuất hiện ở góc trên bên trái màn hình, CenterScreen: giữa màn hình…
Text	Thiết lập dòng tiêu đề của Form.
Window State	Thiết lập trạng thái của Form khi chạy chương trình. Normal: hiển thị Form đúng theo kích cỡ thiết kế, Maximized: phóng to Form

	bằng màn hình, Minimized: thu nhỏ Form trên thanh Taskbar.

3.1.2. Sự kiện

Để hiển thị danh sách các sự kiện của các điều khiển, ta kích chuột tại biểu tượng  trên cửa sổ Properties:



Hình 19. Danh sách sự kiện của điều khiển Form

Muốn gọi sự kiện nào thì ta kích đúp chuột vào tên sự kiện đó, kết quả C# sẽ tự động tạo ra dòng tiêu đề của phương thức chứa sự kiện trong cửa sổ code.

Form có một số sự kiện thông dụng như sau:

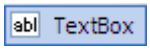
Load	Được kích hoạt khi Form được nạp vào bộ nhớ, nó thường được dùng để khởi tạo các giá trị và trạng thái cho các biến, các điều khiển… trên Form.
Click	Được kích hoạt khi người dùng kích chuột trên Form.
FormClosed	Được kích hoạt khi người dùng kích chuột vào nút Close **x** ở góc trên bên phải để đóng Form.
FormClosing	Cũng được kích hoạt khi người dùng kích chuột vào nút Close **x,** nhưng xảy ra trước sự kiện FormClosed tức là được phát sinh trước khi cửa sổ Form chuẩn bị đóng lại.

Ví dụ: Nếu không muốn người dùng đóng Form bằng cách bấm chọn biểu tượng Close thì trong thủ tục **FormClosing** ta đặt thuộc tính Cancel = True như sau:

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
```

```
{
    e.Cancel = true;
}
```

3.2. Hộp văn bản - TextBox

Hộp văn bản 

là điều khiển rất thông dụng, dùng để nhập dữ liệu đầu vào từ phía người sử dụng và hiển thị các kết quả đã tính toán được.

3.2.1. Thuộc tính

Name	Tên Textbox, bắt đầu bởi tiếp đầu ngữ **txt**
BackColor	Thiết lập mầu nền cho hộp TextBox.
Enabled	Enabled=False: không cho phép người dùng truy cập vào TextBox (Hộp Textbox bị mờ đi), ngược lại thì bằng True.
Font	Thiết lập kiểu chữ và cỡ chữ cho hộp văn bản.
ForeColor	Thiết lập mầu chữ cho hộp văn bản.
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của hộp văn bản trên Form, ngược lại thì nhận giá trị False.
MaxLength	Quy định chiều dài tối đa được chấp nhận của hộp văn bản, giá trị mặc định là 32767 hoặc 0, tức là có thể chứa 32767 ký tự. Mọi xác lập khác 0, ví dụ 5 thì chỉ cho phép người dùng nhập tối đa 5 ký tự vào hộp văn bản.
Multiline	Multiline = False: chỉ cho phép hiển thị văn bản trên một dòng, và khi thiết kế ta chỉ thay đổi được độ dài của hộp văn bản. Multiline = True: cho phép văn bản được hiển thị trên nhiều dòng, và có thể thay đổi cả độ dài lẫn độ rộng của hộp văn bản khi thiết kế.
PasswordChar	Thuộc tính này cho phép người sử dụng bảo mật được thông tin nhập vào Textbox. Ví dụ đặt thuộc tính này bằng ký tự '*' khi đó toàn bộ dữ liệu nhập vào sẽ được hiển thị dưới dạng dấu hoa thị.
ReadOnly	ReadOnly = True: hộp văn bản vẫn được truy cập nhưng người dùng không thể thay đổi được nội dung bên trong.
ScrollBars	Thiết lập thanh cuốn ngang, dọc cho hộp văn bản (có hiệu lực khi Multiline = True). Chú ý: thanh cuốn ngang chỉ có hiệu lực khi WordWrap = False.
TabIndex	Thứ tự truy cập của hộp văn bản khi người dùng bấm phím Tab, thứ tự đầu tiên là 0.
Text	Chứa nội dung của hộp văn bản.
TextAlign	Thiết lập chế độ căn chỉnh: trái, phải hoặc giữa của dữ liệu trong hộp TextBox.
Visible	Visible = True: hiển thị hộp văn bản, Visible = False: ẩn hộp văn bản.

WordWrap	WordWrap = True: dòng văn bản được tự động cuộn xuống dòng khi gặp lề bên phải của hộp TextBox, ngược lại thì nhận giá trị False. Chỉ có hiệu lực khi Multiline = True.

3.2.2. Sự kiện

Hộp văn bản có một số sự kiện cơ bản sau:

TextChanged	Được kích hoạt khi người dùng thực hiện sự thay đổi bất kỳ trong hộp văn bản như: thêm, xoá, sửa, dán văn bản.
Click	Được kích hoạt khi người dùng kích chuột vào hộp văn bản.
DoubleClick	Được kích hoạt khi người dùng kích đúp chuột vào hộp văn bản.
GotFocus	Được kích hoạt khi người dùng chuyển tiêu điểm tới hộp văn bản.
KeyPress	Trả về ký tự (trừ các ký tự đặc biệt như phím Delete, Home, Ctrl, F1…) mà người sử dụng gõ vào hộp văn bản thông qua thuộc tính KeyChar.
KeyDown	Trả về mã Ascii của tất cả các ký tự mà người sử dụng gõ vào hộp văn bản thông qua thuộc tính KeyCode.
LostFocus	Được kích hoạt khi hộp văn bản mất tiêu điểm.
MouseMove	Được kích hoạt khi người dùng di chuyển chuột qua hộp văn bản.
MouseLeave	Được kích hoạt khi người dùng dời chuột ra khỏi hộp văn bản.

Ví dụ 1: Để hiển thị mã Ascii của một ký tự bất kỳ được gõ vào hộp văn bản ***TextBox1*** ta có đoạn chương trình như sau:

```
private void textBox1_KeyDown(object sender, KeyEventArgs e)
 {
     int a;
     a = Convert.ToInt32(e.KeyCode);
     MessageBox.Show(a.ToString());
 }
```

Ví dụ 2: Dùng sự kiện **KeyPress** để kiểm tra việc nhập dữ liệu: chỉ cho phép nhập vào hộp văn bản ***TextBox1*** các số từ 0 tới 9, dấu âm - , dấu chấm thập phân . , phím Del (có mã Ascii=13) và phím Backspace (có mã Ascii = 8) để xóa dữ liệu. Ta có đoạn chương trình như sau:

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
 {
     if (((e.KeyChar >='0') && (e.KeyChar <='9')) ||(e.KeyChar == '-') ||
                             (e.KeyChar == '.') || (Convert.ToInt32(e.KeyChar) == 8) ||
                                                    (Convert.ToInt32(e.KeyChar) == 13))
     {
         e.Handled = false;
     }
     else
```

```
        {
            e.Handled = true;
        }
    }
```

Nếu mỗi ký tự được nhập vào hộp Textbox không thoả mãn điều kiện if thì sẽ bị hủy bỏ bằng cách đặt thuộc tính Handled là true.

3.3. Nút lệnh – Button

Nút lệnh 

cho phép người dùng thực hiện một hành động nào đó.

3.3.1. Thuộc tính

Name	Tên nút lệnh, bắt đầu bởi tiếp đầu ngữ **btn**
BackColor	Thiết lập mầu nền cho nút lệnh.
BackgroundImage	Thiết lập ảnh nền cho nút lệnh.
Enabled	Enabled=False: người dùng không thể tác động lên nút lệnh, ngược lại thì bằng True.
Font	Xác lập kiểu chữ và cỡ chữ cho nút lệnh.
ForeColor	Thiết lập mầu chữ cho nút lệnh.
Image	Thiết lập ảnh hiển thị trên nút lệnh.
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của nút lệnh trên Form, ngược lại thì nhận giá trị False.
TabIndex	Thứ tự truy cập của nút lệnh khi người dùng bấm phím Tab.
Text	Tiêu đề của nút lệnh. Ta có thể quy định phím nóng cho nút lệnh bằng cách đặt dấu "&" trước một ký tự của Text. Ví dụ &Quit sẽ được hiển thị là Quit, khi người sử dụng bấm Alt+Q chương trình sẽ kích hoạt nút lệnh Quit.
Visible	Visible = True: hiển thị nút lệnh, Visible = False: ẩn nút lệnh.

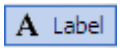
3.3.2. Sự kiện

Nút lệnh có một số sự kiện cơ bản sau:

Click	Được kích hoạt khi người dùng kích chuột vào nút lệnh.
GotFocus	Được kích hoạt khi người dùng chuyển tiêu điểm tới nút lệnh.
LostFocus	Được kích hoạt khi nút lệnh mất tiêu điểm.
MouseDown	Được kích hoạt khi người dùng đặt chuột vào nút lệnh.

MouseUp	Được kích hoạt khi người dùng đưa chuột ra khỏi nút lệnh.
MouseMove	Được kích hoạt khi người dùng di chuyển chuột trên nút lệnh.
MouseLeave	Được kích hoạt khi người dùng dời chuột ra khỏi nút lệnh.

3.4. Nhãn – Lable

Nhãn 

dùng để hiển thị những thông tin có tính chất cố định người sử dụng không có khả năng thay đổi ví dụ như dòng thông báo, hướng dẫn …

Nhãn có một số thuộc tính hay dùng sau:

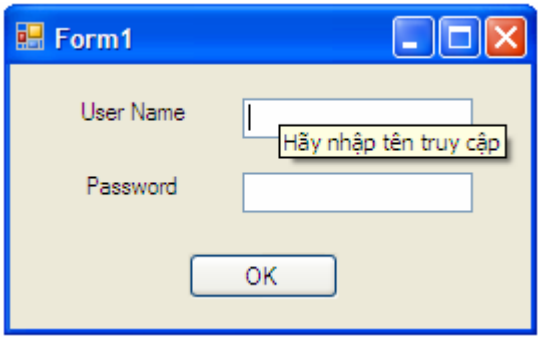
Name	Tên nhãn, bắt đầu bởi tiếp đầu ngữ **lbl**
BackColor	Thiết lập mầu nền cho nhãn, nếu thiết lập BackColor = Transparent (mục lựa chọn đầu tiên trong tab Web) thì nhãn sẽ có nền giống với nền của Form.
BorderStyle	Thiết lập kiểu đường viền cho nhãn.
Font	Thiết lập kiểu chữ và cỡ chữ cho nhãn.
ForeColor	Thiết lập mầu chữ cho nhãn.
Image	Thiết lập ảnh hiển thị trên nhãn.
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của nhãn trên Form, ngược lại thì nhận giá trị False.
TabIndex	Thứ tự truy cập của nhãn khi người dùng bấm phím Tab.
Text	Tiêu đề của nhãn.
TextAlign	Thiết lập chế độ căn chỉnh: trái, phải hoặc giữa của tiêu đề nhãn.
Visible	Hiện hoặc ẩn nhãn.

3.5. Dòng mách nước - ToolTip

Điều khiển mách nước 

cho phép hiển thị các thông tin chú thích khi người dùng đưa chuột qua điều khiển có thiết lập ToolTip.

Ví dụ dòng mách nước “Hãy nhập tên truy cập” như hình dưới đây.



Hình 20. Ví dụ ToolTip

Để tạo dòng mách nước cho một điều khiển ta thực hiện như sau:

+ Kéo điều khiển ToolTip vào Form, điều khiển ToolTip không được hiển thị ở trên Form mà được hiển thị ở thanh ngang cuối Form và được dùng chung cho mọi điều khiển trên form.

+ Kích chuột chọn điều khiển muốn tạo ToolTip, trong cửa sổ Window Properties gõ nội dung dòng ToolTip tại thuộc tính ToolTip on ToolTip1.

3.6. Bài tập

Bài tập 2.

Lập chương trình nhập tháng và năm dương lịch, tính và in ra số ngày của tháng và năm đó.

Hình 21. Giao diện bài tập 2

Yêu cầu: + Chỉ được phép nhập số nguyên vào hai hộp văn bản chứa tháng và năm.

+ Tháng phải có giá trị từ 1 đến 12, năm gồm 4 chữ số và có giá trị >=1900.

+ Kết quả chỉ được tính khi người dùng nhập đủ cả tháng và năm.

Tạo dự án mới và thiết lập các thuộc tính của các điều khiển như sau:

Điều khiển	Thuộc tính	Giá trị
Form1	Name	frmNgaythang
	FormBorderStyle	Fixed3D
	Icon	Chọn file ảnh có đuôi .ico bất kỳ
	Text	Program of date
Lable1	Text	Nhập tháng
Lable2	Text	Nhập năm
Lable3	Name	lblKetqua
	Text	Kết quả:
TextBox1	Name	txtNhapthang
	MaxLength	2
TextBox2	Name	txtNhapnam
	MaxLength	4
Button1	Name	btnKetqua
	Text	&Kết quả
Button2	Name	btnLamlai
	Text	&Làm lại
Button3	Name	btnThoat
	Text	&Thoát

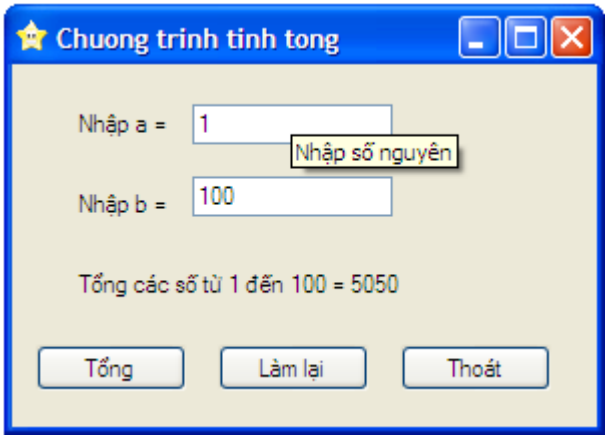
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class frmNgaythang : Form
    {
        public frmNgaythang()
        {
            InitializeComponent();
        }

        private void txtNhapthang_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (((e.KeyChar >='0') && (e.KeyChar <='9'))||(Convert.ToInt32(e.KeyChar)==8))
                e.Handled = false;
            else
                e.Handled = true;
```

```
}

private void txtNhapnam_KeyPress(object sender, KeyPressEventArgs e)
{
    if (((e.KeyChar >='0') && (e.KeyChar <='9'))||(Convert.ToInt32(e.KeyChar)==8))
        e.Handled = false;
    else
        e.Handled = true;
}

private void btnKetqua_Click(object sender, EventArgs e)
{
    int thang, nam, ngay = 0;
    if (txtNhapthang.Text == "")
    {
        MessageBox.Show("Bạn phải nhập tháng", "Thông báo", MessageBoxButtons.OK,
                                                        MessageBoxIcon.Information);
        txtNhapthang.Focus();
        return;
    }
    if (txtNhapnam.Text.Length != 4)
    {
        MessageBox.Show("Bạn phải nhập năm có 4 chữ số", "Thông báo",
                                    MessageBoxButtons.OK, MessageBoxIcon.Information);
        txtNhapnam.Focus();
        txtNhapnam.Text = "";
        return;
    }
    thang = Convert.ToInt32(txtNhapthang.Text);
    nam = Convert.ToInt32(txtNhapnam.Text);
    if (thang > 12 || thang < 0)
    {
        MessageBox.Show("Bạn phải nhập tháng trong [1,12]", "Thông báo",
                                    MessageBoxButtons.OK, MessageBoxIcon.Information);
        txtNhapthang.Focus();
        txtNhapthang.Text = "";
        return;
    }
    if (nam < 1900)
    {
        MessageBox.Show("Bạn phải nhập năm có giá trị >= 1900", "Thông báo",
                                    MessageBoxButtons.OK, MessageBoxIcon.Information);
        txtNhapnam.Focus();
        txtNhapnam.Text = "";
        return;
    }
    switch (thang)
    {
        case 4:
        case 6:
        case 9:
        case 11:
            ngay = 30;
            break;
        case 1:
        case 3:
        case 5:
```

```
                case 7:
                case 8:
                case 10:
                case 12:
                    ngay = 31;
                    break;
                case 2:
                    if (((nam % 4 == 0) && (nam % 100 != 0))||(nam % 400 == 0))
                        ngay = 29;
                    else
                        ngay = 28;
                    break;
            }
            lblKetqua.Text = "Tháng " + thang + " năm " + nam + " có " + ngay + " ngày";
            btnKetqua.Enabled = false;
            btnLamlai.Enabled = true;
            txtNhapthang.ReadOnly = true;
            txtNhapnam.ReadOnly = true;
        }

        private void btnLamlai_Click(object sender, EventArgs e)
        {
            txtNhapthang.Text = "";
            txtNhapthang.ReadOnly = false;
            txtNhapthang.Focus();
            txtNhapnam.Text = "";
            txtNhapnam.ReadOnly = false;
            btnKetqua.Enabled = true;
            lblKetqua.Text = "Kết quả: ";
        }

        private void btnThoat_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Bạn có muốn thoát không?", "Thông báo",
                                    MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
                                                    System.Windows.Forms.DialogResult.Yes)
                Application.Exit();
        }
    }
}
```

Bài tập 3.

Nhập 2 số nguyên a, b và tính tổng các số từ a đến b theo giao diện sau:



Hình 22. Giao diện bài tập 3

Yêu cầu:

+ Tạo dòng ToolTip "Nhập số nguyên" cho 2 hộp văn bản 'Nhập a' và 'Nhập b'.
+ Chỉ cho phép người dùng nhập số vào hai hộp văn bản.
+ Nút Tổng: kiểm tra người dùng phải nhập dữ liệu cho cả hai số a và b, tính tổng các số từ a đến b nếu a < b, hoặc tính tổng các số từ b đến a nếu b < a, rồi hiển thị kết quả vào nhãn ở phía dưới.
+ Nút Làm lại: xóa các dữ liệu cũ ở các điều khiển, sau đó đặt con trỏ vào hộp văn bản Nhập a.
+ Nút Thoát: thoát khỏi chương trình quay về môi trường soạn thảo.

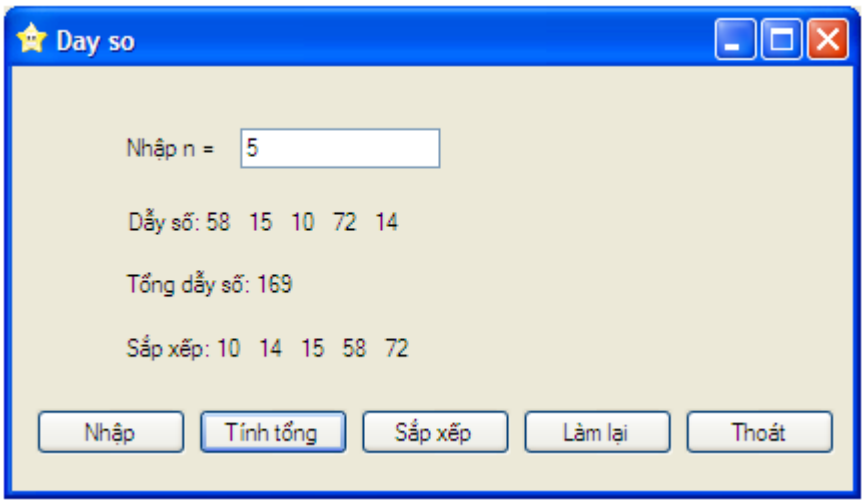
Bài tập 4.

Nhập số nguyên dương n, tạo n số nguyên ngẫu nhiên có giá trị từ 1 tới 100, và thực hiện các yêu cầu sau:

+ Chỉ cho phép người dùng nhập số vào hộp văn bản Nhập n.
+ Nút Nhập: kiểm tra người dùng phải nhập giá trị cho n, sau đó tạo n số ngẫu nhiên và hiển thị các số ngẫu nhiên đó ở nhãn Dãy số.
+ Nút Tính tổng: tính tổng n số ngẫu nhiên và hiển thị kết quả ở nhãn Tổng dãy số.
+ Nút Sắp xếp: sắp xếp n số ngẫu nhiên theo thứ tự tăng dần và hiển thị kết quả ở nhãn Sắp xếp.

+ Nút Làm lại: xóa các dữ liệu cũ ở các điều khiển, sau đó đặt con trỏ vào hộp văn bản Nhập n.

+ Nút Thoát: thoát khỏi chương trình quay về môi trường soạn thảo.

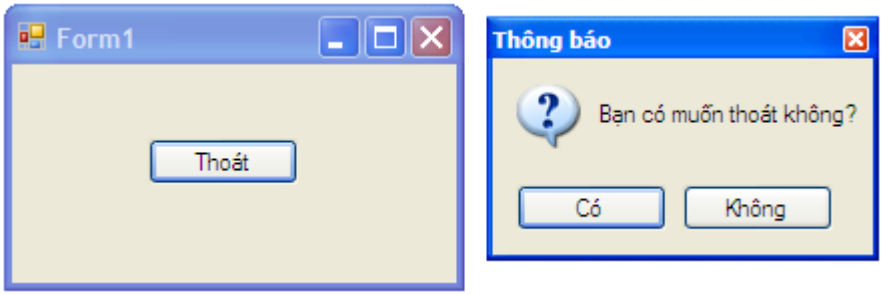


Hình 23. Giao diện bài tập 4

Bài tập 5.

Hộp thoại MessageBox của C# tuy rất đa dạng và phong phú nhưng có lúc ta cũng cần một hộp thoại tương tự theo phong cách riêng của mình, ví dụ hiển thị tiêu đề cho các nút nhấn bằng tiếng Việt.

Ví dụ: viết chương trình tự tạo hộp thoại thông báo thực hiện các yêu cầu theo giao diện sau:



Hình 24. Giao diện bài tập 5

Kích chuột vào nút *Thoát* trên Form1 sẽ xuất hiện hộp thông báo *Bạn có muốn thoát không?* với giao diện tiếng Việt. Chọn nút *Có* để đóng Form1 và thoát khỏi chương trình, chọn nút *Không* để quay lại Form1.

Trong đó hộp thông báo *Bạn có muốn thoát không?* chính là một form được thiết kế tương tự như một hộp thoại thông báo.

Bước 1: Xây dựng Form1

Vào Microsoft Visual Studio 2010 tạo một dự án mới có tên *Hopthoai* và thiết lập các thuộc tính của các điều khiển như sau:

Điều khiển	Thuộc tính	Giá trị
Form1	Name	Form1
	StartPosition	CenterScreen
Button1	Name	btnThoat
	Text	Thoát

Bước 2: Xây dựng form Thông báo

Ta bổ sung thêm một form mới vào dự án theo các bước sau:

+ Chọn menu Project | Add Windows Forms xuất hiện hộp thoại Add New Item.

+ Chọn Windows Form và đặt tên Message.cs tại ô Name rồi chọn nút Add. Kết quả form Message.cs được thêm vào dự án.

+ Thay đổi kích thước của form Message.cs như hình trên và thiết lập các thuộc tính của các điều khiển trên form như sau:

Điều khiển	Thuộc tính	Giá trị
Message.vb	Name	frmMessage
	FormBorderStyle	FixedToolWindow
	ShowIcon	False
	StartPosition	CenterScreen
	Text	Thông báo
Lable1	Name	lblMessage
	Modifiers	Public
PictureBox1	Image	Question.bmp
Button1	Name	btnCo
	DialogResult	Yes
	Text	Có
Button2	Name	btnKhong

	DialogResult	No
	Text	Không

Bước 3: Viết Code

Mở cửa sổ code của Form1 và viết mã lệnh cho nút btnThoat như sau:

```
private void btnThoat_Click(object sender, EventArgs e)
 {
     frmMessage frm = new frmMessage();
     frm.lblMessage.Text = "Bạn có muốn thoát không?";
     if (frm.ShowDialog() == System.Windows.Forms.DialogResult.Yes)
         Application.Exit();
 }
```

Chú ý:

+ Từ một form ta không thể tác động trực tiếp tới một form khác mà phải khai báo một biến đại diện cho form muốn tác động và mọi thao tác đều được thực hiện trên biến này. Ví dụ từ form *Form1* muốn hiển thị hoặc tác động tới các điều khiển trong form *frmMessage* ta khai báo biến *frm* có kiểu là *frmMessage* như trên.

+ Để hiển thị một Form khác ta gọi phương thức ShowDialog theo cú pháp sau:

  TênbiếnForm.ShowDialog();

+ Để truy cập tới thuộc tính của các điều khiển trong một Form khác ta thực hiện theo cú pháp sau:

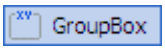
  TênbiếnForm.Tênđiềukhiển.Tênthuộctính = Giátrị;

+ Khi ta thiết lập giá trị DialogResult cho một nút lệnh thì nếu người dùng kích chuột vào nút lệnh đó cửa sổ Form chứa nút lệnh sẽ bị đóng lại và trả về giá trị của DialogResult tại nơi gọi phương thức ShowDialog() để hiển thị Form đó.

+ PictureBox là một điều khiển cho phép chứa các tệp ảnh có đuôi .bmp, .jpg… Trong máy tính không có sẵn tệp ảnh Question.bmp mà ta phải tự tạo bằng ứng dụng Paint.

4. Một số điều khiển cơ bản khác

4.1. Nhóm – GroupBox

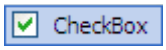
Nhóm 

có thể chứa các điều khiển khác và tạo thành các vùng làm việc độc lập trên một Form.

GroupBox có một số thuộc tính thường dùng sau:

Name	Tên nhóm, bắt đầu bởi tiếp đầu ngữ **grb**
BackColor	Thiết lập mầu nền cho nhóm, nếu BackColor = Transparent thì nhóm sẽ có mầu nền giống với mầu nền của Form.
BackgroundImage	Thiết lập ảnh nền cho nhóm.
BackgroundImageLayout	Thiết lập chế độ hiển thị ảnh nền của nhóm.
Enabled	Nếu Enabled = False nhóm sẽ không hoạt động.
Font	Xác lập kiểu chữ và cỡ chữ của tiêu đề nhóm.
ForeColor	Xác lập mầu chữ của tiêu đề nhóm.
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của nhóm trên Form, ngược lại thì nhận giá trị False.
TabIndex	Thứ tự truy cập của nhóm khi người dùng bấm phím Tab.
Text	Thiết lập tiêu đề của nhóm.
Visible	Visible = True: hiển thị nhóm, Visible = False: ẩn nhóm.

4.2. Hộp đánh dấu – CheckBox

Hộp đánh dấu 

cho phép đồng thời không chọn, chọn một, hoặc chọn nhiều khả năng trong một nhóm các lựa chọn.

4.2.1. Thuộc tính

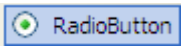
Name	Tên hộp CheckBox, bắt đầu bởi tiếp đầu ngữ **chk**
BackColor	Thiết lập mầu nền cho hộp CheckBox.
BackgroundImage	Thiết lập ảnh nền cho hộp CheckBox.
Checked	Trả về giá trị của hộp CheckBox ứng với trạng thái của nó khi tương tác với người sử dụng: + Checked =True: hộp CheckBox đang được chọn + Checked = False: hộp CheckBox không được chọn.
CheckState	Thiết lập trạng thái cho hộp CheckBox: + CheckState = Checked: hộp CheckBox được chọn + CheckState = Unchecked: hộp CheckBox không được chọn.
Enabled	Nếu Enabled = False hộp CheckBox sẽ không hoạt động.
Font	Xác lập kiểu chữ và cỡ chữ của nội dung hộp CheckBox.
ForeColor	Xác lập mầu chữ của nội dung hộp CheckBox.

Image	Thiết lập ảnh hiển thị trên hộp CheckBox.
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của hộp CheckBox trên Form, ngược lại thì nhận giá trị False.
TabIndex	Thứ tự truy cập khi người dùng bấm phím Tab.
Text	Thiết lập nội dung của hộp CheckBox.
Visible	Visible = True: hiển thị hộp CheckBox Visible = False: ẩn hộp CheckBox.

4.2.2. Sự kiện

Click	Được kích hoạt khi người dùng kích chuột vào hộp CheckBox.
GotFocus	Được kích hoạt khi người dùng chuyển tiêu điểm tới hộp CheckBox.
LostFocus	Được kích hoạt khi hộp CheckBox mất tiêu điểm.
CheckedChanged	Được kích hoạt khi hộp CheckBox thay đổi trạng thái.

4.3. Nút tuỳ chọn – RadioButton

Nút tuỳ chọn 

chỉ cho phép người dùng chọn một khả năng trong một nhóm các lựa chọn.

4.3.1. Thuộc tính

Name	Tên nút tuỳ chọn, bắt đầu bởi tiếp đầu ngữ **rdo**
BackColor	Thiết lập mầu nền cho nút tùy chọn.
BackgroundImage	Thiết lập ảnh nền cho nút tùy chọn.
Checked	Trả về giá trị của nút tùy chọn khi tương tác với người sử dụng. + Checked = True: nút tùy chọn đang được chọn + Checked = False: nút tùy chọn không được chọn.
Enabled	Nếu Enabled = False nút tuỳ chọn sẽ không hoạt động.
Font	Xác lập kiểu chữ và cỡ chữ của nội dung nút tùy chọn.
ForeColor	Xác lập mầu chữ của nội dung nút tùy chọn.
Image	Thiết lập ảnh hiển thị trên nút tùy chọn.
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của nút tùy chọn trên Form, ngược lại thì nhận giá trị False.
TabIndex	Thứ tự truy cập khi người dùng bấm phím Tab.

Text	Thiết lập nội dung của nút tùy chọn.
Visible	True: hiển thị nút tùy chọn, False: ẩn nút tùy chọn.

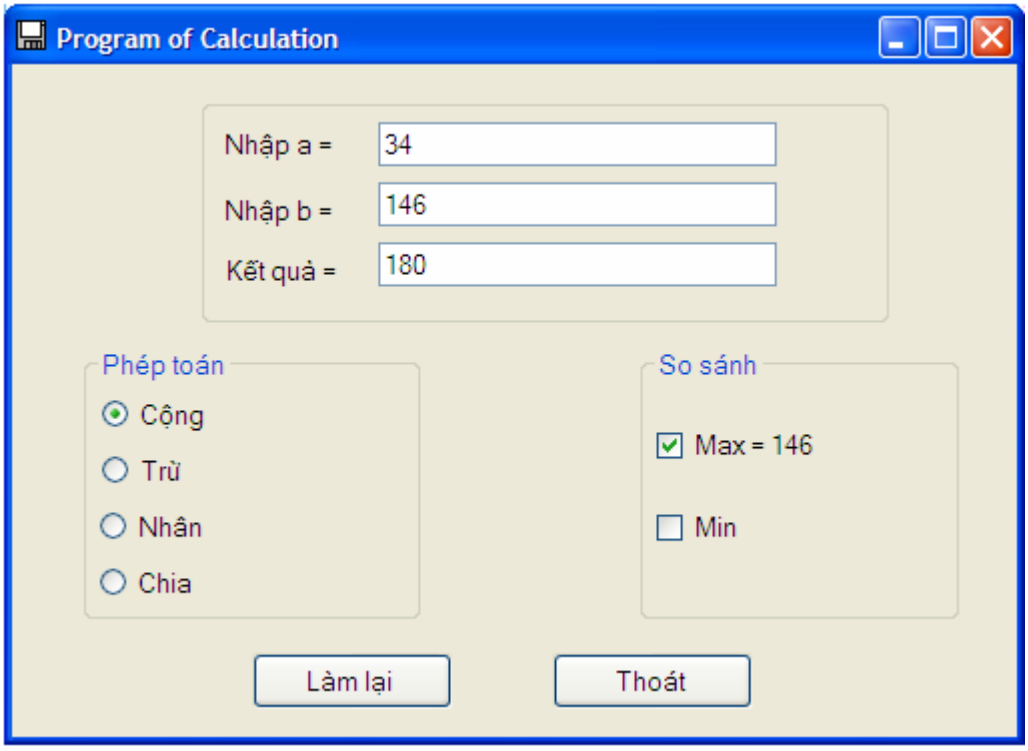
4.3.2. Sự kiện

Click	Được kích hoạt khi người dùng kích chuột vào nút tuỳ chọn.
GotFocus	Được kích hoạt khi người dùng chuyển tiêu điểm tới nút tuỳ chọn.
LostFocus	Được kích hoạt khi nút tuỳ chọn mất tiêu điểm.
CheckedChanged	Được kích hoạt khi nút tùy chọn thay đổi trạng thái.

Bài tập 6.

Lập chương trình nhập 2 số a và b, chọn và thực hiện các phép toán theo yêu cầu sau:

+ Chỉ được nhập số cho a và b, không cho phép nhập dữ liệu vào hộp kết quả.
+ Các phép toán chỉ được thực hiện khi người dùng nhập đủ hai dữ liệu cho a và b. Trong phép chia kiểm tra nếu b = 0 thì thông báo "Mẫu = 0" tại hộp Kết quả.
+ Kích chọn phép toán nào thì thực hiện phép toán đó đối với a, b và lưu kết quả vào hộp Kết quả.
+ Nếu chọn hộp đánh dấu Max thì hiển thị "Max = <Giá trị max>" ngược lại chỉ hiển thị "Max". Thực hiện tương tự cho hộp đánh dấu Min.



Hình 25. Giao diện bài tập 6

Vào Microsoft Visual Studio 2010 tạo một dự án mới đặt tên là **Calculation** và thiết lập các thuộc tính của các điều khiển như sau:

Điều khiển	Thuộc tính	Giá trị
Form1	Name	frmCalculation
	FormBorderStyle	Fixed3D
	Icon	Chọn file ảnh có đuôi .ico bất kỳ
	Text	Program of Calculation
GroupBox1	Text	Rỗng
GroupBox2	Text	Phép toán
GroupBox3	Text	So sánh
Lable1	Text	Nhập a =
Lable2	Text	Nhập b =
Lable3	Text	Kết quả =
TextBox1	Name	txtNhapA
TextBox2	Name	txtNhapB
TextBox3	Name	txtKetqua
RadioButton1	Name	rdoCong
	Text	Cộng

RadioButton2	Name	rdoTru
	Text	Trừ
RadioButton3	Name	rdoNhan
	Text	Nhân
RadioButton4	Name	rdoChia
	Text	Chia
CheckBox1	Name	chkMax
	Text	Max
CheckBox2	Name	chkMin
	Text	Min
Button1	Name	btnLamlai
	Text	Làm lại
Button2	Name	btnThoat
	Text	Thoát

Viết Code: mở cửa sổ soạn thảo Code và viết các đoạn mã lệnh như sau:

```
private void txtNhapa_KeyPress(object sender, KeyPressEventArgs e)
{
    if (((e.KeyChar >='0') && (e.KeyChar <='9'))||(Convert.ToInt32(e.KeyChar)==8))
        e.Handled = false;
    else
        e.Handled = true;
}

private void rdoCong_Click(object sender, EventArgs e)
{
    int a,b;
    if (txtNhapa.Text == "")
    {
        MessageBox.Show("Bạn phải nhập giá trị cho a", "Thông báo",
                                MessageBoxButtons.OK, MessageBoxIcon.Information);
        txtNhapa.Focus();
        rdoCong.Checked = false;
        return;
    }
    if (txtNhapb.Text == "")
    {
        MessageBox.Show("Bạn phải nhập giá trị cho b", "Thông báo",
                                MessageBoxButtons.OK, MessageBoxIcon.Information);
        txtNhapb.Focus();
        rdoCong.Checked = false;
        return;
    }
    a = Convert.ToInt32(txtNhapa.Text);
    b = Convert.ToInt32(txtNhapb.Text);
    txtKetqua.Text = Convert.ToString(a + b);
    txtNhapa.Enabled = false;
```

```
        txtNhapb.Enabled = false;
    }

    private void chkMax_Click(object sender, EventArgs e)
     {
         int a,b;
         if (txtNhapa.Text == "")
         {
             MessageBox.Show("Bạn phải nhập giá trị cho a", "Thông báo",
                                        MessageBoxButtons.OK, MessageBoxIcon.Information);
             txtNhapa.Focus();
             chkMax.Checked = false;
             return;
         }
         if (txtNhapb.Text == "")
         {
             MessageBox.Show("Bạn phải nhập giá trị cho b", "Thông báo",
                                        MessageBoxButtons.OK, MessageBoxIcon.Information);
             txtNhapb.Focus();
             chkMax.Checked = false;
             return;
         }
        a = Convert.ToInt32(txtNhapa.Text);
        b = Convert.ToInt32(txtNhapb.Text);
        if (chkMax.Checked == true)
          {
              if (a>b)
                 chkMax.Text = "Max= " + a.ToString ();
              else
                 chkMax.Text = "Max= " + b.ToString ();
          }
        else
          chkMax.Text = "Max";
        txtNhapa.Enabled = false;
        txtNhapb.Enabled = false;
     }
```

Chú ý: trong đoạn mã lệnh trên tất cả các thủ tục: *rdoCong_Click*, *rdoTru_Click*, *rdoNhan_Click*, *rdoChia_Click*, *chkMax_Click*, *chkMin_Click* đều phải kiểm tra người dùng nhập dữ liệu vào hai hộp văn bản *txtNhapA* và *txtNhapB*. Để không phải viết lại đoạn mã lệnh kiểm tra nhiều lần ta viết một hàm ***KiemtraAB*** như sau:

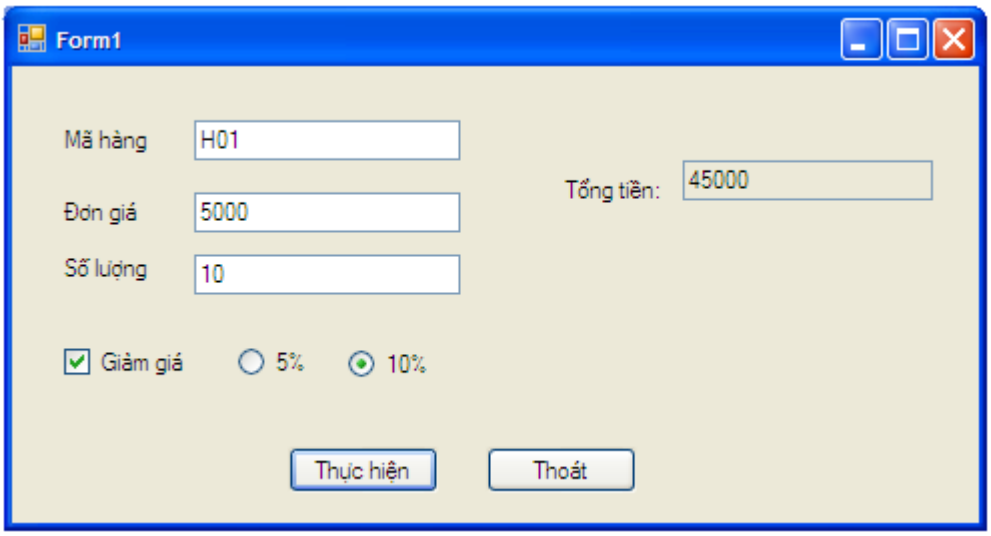
```
    private bool KiemtraAB()
     {
         if (txtNhapa.Text == "")
         {
             MessageBox.Show("Bạn phải nhập giá trị cho a", "Thông
                                 báo",MessageBoxButtons.OK, MessageBoxIcon.Information);
             txtNhapa.Focus();
             return false;
         }
         if (txtNhapb.Text == "")
         {
             MessageBox.Show("Bạn phải nhập giá trị cho b", "Thông báo",
                                        MessageBoxButtons.OK, MessageBoxIcon.Information);
```

```
            txtNhapb.Focus();
            return false;
        }
        return true;
    }

    private void rdoCong_Click(object sender, EventArgs e)
    {
        int a, b;
        if (KiemtraAB() == false)
        {
            rdoCong.Checked = false;
        }
        else
        {
            a = Convert.ToInt32(txtNhapa.Text);
            b = Convert.ToInt32(txtNhapb.Text);
            txtKetqua.Text = Convert.ToString(a + b);
            txtNhapa.Enabled = false;
            txtNhapb.Enabled = false;
        }
    }
```

// Tương tự viết mã lệnh cho các thủ tục khác

Bài tập 7.

Lập chương trình thực hiện bài toán theo yêu cầu và giao diện như sau:

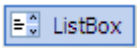


Hình 26. Giao diện bài tập 7

+ Kiểm tra người dùng phải nhập đủ dữ liệu cho Mã hàng, đơn giá và số lượng.
+ Chỉ được phép nhập giá trị số cho Đơn giá và Số lượng.

+ Nếu Giảm giá được chọn thì hiển thị 2 điều khiển giảm giá 5% và 10%, ngược lại không hiển thị 2 điều khiển này.

4.4. Hộp danh sách – ListBox

Hộp ListBox  là một tập hợp các chuỗi ký tự được trình bày dưới dạng liệt kê thành từng dòng trong một khung hình chữ nhật. Ta có thể chọn, bổ sung hoặc xoá một giá trị trong hộp danh sách.

Khi hiển thị dữ liệu, nếu chiều ngang của Listbox nhỏ hơn độ dài các phần tử thì một phần dữ liệu sẽ bị che khuất, còn nếu số phần tử của Listbox vượt quá chiều dài của Listbox thì Listbox tự động cung cấp thanh cuốn dọc để cuộn tới các phần tử phía dưới.

4.4.1. Thuộc tính

Name	Tên hộp ListBox, bắt đầu bởi tiếp đầu ngữ **lst**
BackColor	Thiết lập mầu nền cho hộp danh sách.
DataSource	Thiết lập nguồn dữ liệu cho ListBox
Enabled	Nếu Enabled = False hộp danh sách sẽ không hoạt động.
Font	Xác lập kiểu chữ và cỡ chữ cho hộp danh sách.
ForeColor	Xác lập mầu chữ cho hộp danh sách.
MultiColumn	MultiColumn = True: cho phép hiển thị dữ liệu theo nhiều cột. MultiColumn = False: chỉ cho phép hiển thị dữ liệu theo 1 cột.
ColumnWidth	Thiết lập độ rộng cho mỗi cột trong ListBox.
Items	Khởi tạo giá trị cho các phần tử của hộp danh sách trong thời gian thiết kế. Khi chọn thuộc tính Items trong cửa sổ Properties, C# mở ra một hộp soạn thảo cho phép người lập trình gõ vào giá trị các phần tử. Mỗi phần tử được đặt trên một dòng riêng biệt, để xuống dòng nhấn Enter.
Items.Count	Trả về tổng số phần tử của danh sách trong thời gian thi hành.
Items[n]	Trả về nội dung phần tử thứ n của danh sách trong thời gian thi hành.
SelectedItem hoặc **Text**	Tương tự như thuộc tính Items[n], nhưng chỉ có thể trả về nội dung của phần tử hiện hành đang được chọn.
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của hộp danh sách trên Form, ngược lại thì nhận giá trị False.
SelectedIndex	Trả về số thứ tự của phần tử đang được chọn trong danh sách, phần tử đầu tiên có SelectedIndex = 0, nếu không có phần tử nào được chọn thì SelectedIndex = -1
SelectionMode	Quy định chế độ lựa chọn các phần tử trong hộp danh sách khi thực thi chương trình. SelectionMode có 4 giá trị: None - không cho phép lựa chọn

	các phần tử, One - cho phép chọn một phần tử, MultiSimple - cho phép lựa chọn nhiều phần tử riêng biệt, MultiExtended - cho phép chọn một khối các phần từ liền nhau.
SelectedItems	Trả về tập các phần tử đang được chọn.
Sorted	Nếu Sorted = True thì các phần tử trong danh sách được sắp xếp theo thứ tự ABC.
TabIndex	Thứ tự truy cập khi người dùng bấm phím Tab.
Visible	True: hiển thị hộp danh sách, False: ẩn hộp danh sách.

4.4.2. Sự kiện

Click	Được kích hoạt khi người dùng kích chuột vào hộp danh sách.
DoubleClick	Được kích hoạt khi người dùng kích đúp chuột vào hộp danh sách.
GotFocus	Được kích hoạt khi người dùng chuyển tiêu điểm tới hộp danh sách.
LostFocus	Được kích hoạt khi hộp danh sách mất tiêu điểm.
SelectedIndex_ Changed	Được kích hoạt khi người dùng thay đổi trạng thái lựa chọn các dòng dữ liệu trong hộp văn bản.

4.4.3. Phương thức

Add: dùng để bổ sung một phần tử cho hộp danh sách trong thời gian thi hành và thường được viết trong thủ tục Form_Load. Cú pháp của phương thức này là:

ListName.Items.Add(Item);

Trong đó ListName là tên của hộp danh sách, Item là nội dung của phần tử ta muốn thêm vào hộp danh sách.

Ví dụ, bổ sung phần tử có giá trị "Ha Noi" vào hộp danh sách lstQue ta thực hiện như sau:

```
lstQue.Items.Add("Ha Noi");
```

Remove: dùng để loại bỏ một phần tử của hộp danh sách theo nội dung trong thời gian thi hành. Cú pháp của phương thức này là:

ListName.Items.Remove(Item);

Ví dụ, xóa phần tử có giá trị "Ha Noi" trong hộp danh sách lstQue ta viết như sau:

```
lstQue.Items.Remove("Ha Noi");
```

RemoveAt: dùng để loại bỏ một phần tử của hộp danh sách theo chỉ số trong thời gian thi hành. Cú pháp của phương thức này là:

ListName.Items.RemoveAt(Index);

Ví dụ, xóa phần tử ở vị trí 1 trong hộp danh sách lstQue ta viết như sau:

```
lstQue.Items.RemoveAt(1);
```

Clear: dùng để loại bỏ tất cả các phần tử của hộp danh sách trong thời gian thi hành. Cú pháp của phương thức này là:

ListName.Items.Clear();

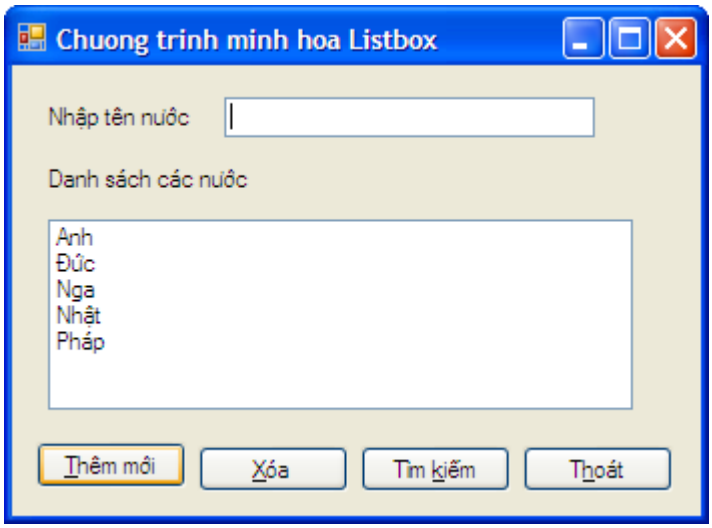
Ví dụ, xóa tất cả các phần tử trong hộp danh sách lstQue ta viết như sau:

```
lstQue.Items.Clear();
```

Bài tập 8.

Viết chương trình minh hoạ các thao tác trên hộp Listbox theo yêu cầu sau:

+ Nhập một tên nước vào hộp văn bản *Nhập tên nước*, chọn nút *Thêm mới* để thêm nước đó vào hộp danh sách, chọn nút *Tìm kiếm* để xem nước đó đã có trong hộp danh sách chưa?
+ Chọn nút *Xóa* để xoá một tên nước bất kỳ được chọn từ hộp danh sách.
+ Chọn nút *Thoát* để thoát khỏi chương trình.



Hình 27. Giao diện bài tập 8

Vào Microsoft Visual Studio 2010 tạo một dự án mới và thiết lập các thuộc tính của các điều khiển như sau:

Điều khiển	Thuộc tính	Giá trị
Form1	Name	frmTennuoc
	FormBorderStyle	Fixed3D
	Icon	Chọn file ảnh có đuôi .ico bất kỳ
	Text	Chuong trinh minh hoa Listbox
Lable1	Text	Nhập tên nước
Lable2	Text	Danh sách các nước
TextBox1	Name	txtNuoc
ListBox1	Name	lstNuoc
Button1	Name	btnThemmoi
	Text	&Thêm mới
Button2	Name	btnXoa
	Text	&Xóa
Button3	Name	btnTimkiem
	Text	Tìm &kiếm
Button4	Name	btnThoat
	Text	T&hoát

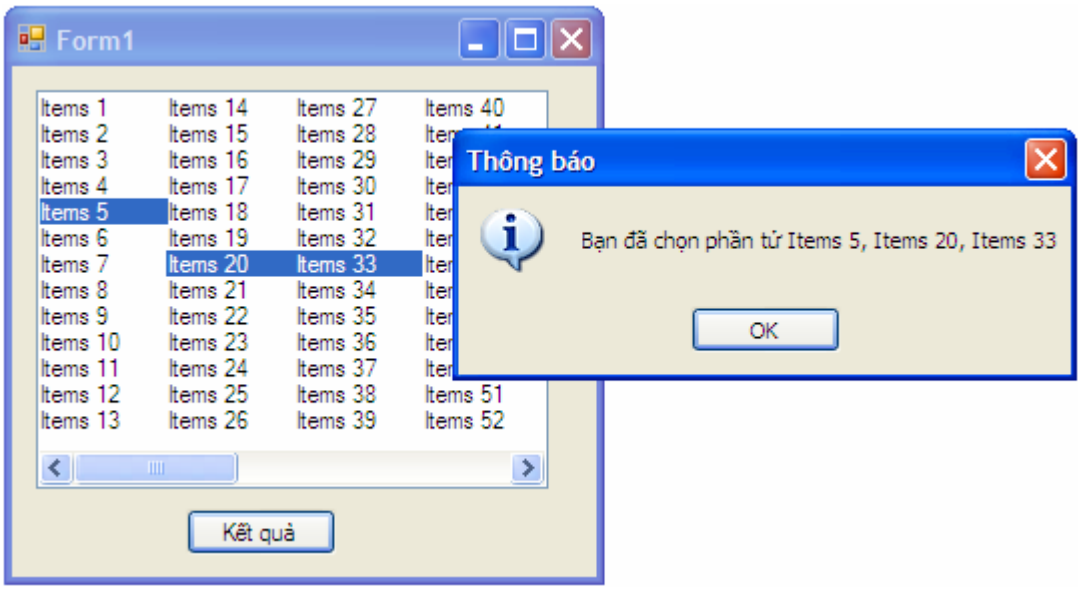
Viết Code: mở cửa sổ soạn thảo Code và viết các đoạn mã lệnh như sau:

```
private void Form1_Load(object sender, EventArgs e)
{
    txtNuoc.Focus();
    btnTimkiem.Enabled = false;
    btnXoa.Enabled = false;
}

private void btnThemmoi_Click(object sender, EventArgs e)
{
    if (txtNuoc.Text == "")
    {
        MessageBox.Show("Bạn phải nhập tên nước", "Thông báo",
                                        MessageBoxButtons.OK, MessageBoxIcon.Information);
        txtNuoc.Focus();
        return;
    }
    lstNuoc.Items.Add(txtNuoc.Text);
    txtNuoc.Text = "";
    txtNuoc.Focus();
    btnXoa.Enabled = true;
    btnTimkiem.Enabled = true;
}

private void btnXoa_Click(object sender, EventArgs e)
{
    if (lstNuoc.Items.Count == 0)
    {
```

```
            MessageBox.Show("Không còn phần tử nào để xoá", "Thông báo",
                                        MessageBoxButtons.OK, MessageBoxIcon.Information);
            btnXoa.Enabled = false;
            btnTimkiem.Enabled = false;
            return;
        }
        if (lstNuoc.SelectedIndex == -1)
        {
            MessageBox.Show("Bạn phải chọn một nước để xoá", "Thông báo",
                                        MessageBoxButtons.OK, MessageBoxIcon.Information);
            return;
        }
        if (MessageBox.Show("Bạn có muốn xoá không?", "Thông báo",
                    MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
            lstNuoc.Items.RemoveAt(lstNuoc.SelectedIndex);
    }
    private void btnTimkiem_Click(object sender, EventArgs e)
    {
        int i, index, d =0;
        if (lstNuoc.Items.Count == 0)
        {
            MessageBox.Show("Không còn phần tử nào để tìm kiếm", "Thông báo",
                                        MessageBoxButtons.OK, MessageBoxIcon.Information);
            btnXoa.Enabled = false;
            btnTimkiem.Enabled = false;
            return;
        }
        if (txtNuoc.Text == "")
        {
            MessageBox.Show("Bạn phải nhập tên nước để tìm kiếm", "Thông báo",
                                        MessageBoxButtons.OK, MessageBoxIcon.Information);
            txtNuoc.Focus();
            return;
        }
        for (i = 0; i<= lstNuoc.Items.Count - 1; i++)
            if (lstNuoc.Items[i].ToString().ToLower() == txtNuoc.Text.ToLower())
            {
                d = 1;
                index = i;
                break;
            }
        if (d == 1)
        {
              MessageBox.Show("Có tìm thấy nước " + txtNuoc.Text, "Thông báo",
                                        MessageBoxButtons.OK, MessageBoxIcon.Information);
              lstNuoc.SelectedIndex = index;
        }
        else
            MessageBox.Show("Không tìm thấy nước " + txtNuoc.Text, "Thông báo",
                                        MessageBoxButtons.OK, MessageBoxIcon.Information);
        txtNuoc.Text = "";
        txtNuoc.Focus();
    }
```

Bài tập 9.

Viết chương trình minh hoạ các thao tác trên hộp Listbox theo giao diện và yêu cầu sau:



Hình 28. Giao diện bài tập 9

Yêu cầu: + Nhập vào hộp danh sách 100 phần tử từ Items 1 đến Items 100.

+ Dữ liệu được hiển thị thành 4 cột trong một trang màn hình.

+ Người dùng có thể lựa chọn đồng thời một hoặc nhiều phần tử.

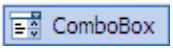
Vào Microsoft Visual Studio 2010 tạo một dự án mới, đặt một hộp danh sách *lstDanhsach* và một nút lệnh *btnKetqua* với tiêu đề là *Kết quả* vào form Form1.

Viết Code: mở cửa sổ soạn thảo Code và viết các đoạn mã lệnh như sau:

```
private void Form1_Load(object sender, EventArgs e)
 {
       // Cho phép hiển thị nhiều cột
    lstDanhsach.MultiColumn = true ;
       // Hiển thị 4 cột trong một trang
    lstDanhsach.ColumnWidth = lstDanhsach.Width / 4;
       // Cho phép chọn đồng thời nhiều phần tử
    lstDanhsach.SelectionMode = SelectionMode.MultiSimple;
          // Add dữ liệu vào hộp danh sách
    for (int i = 1 ; i <= 100; i++)
        lstDanhsach.Items.Add("Items " + i);
 }

 private void btnKetqua_Click(object sender, EventArgs e)
 {
    string str = "";
         // Duyệt qua từng phần tử đã chọn
    foreach (string Item in lstDanhsach.SelectedItems)
```

```
            str = str + Item + ", ";
              // Xóa dấu phẩy và dấu cách thừa ở cuối chuỗi str
        str = str.Remove(str.Length - 2, 2);
        MessageBox.Show("Bạn đã chọn phần tử " + str, "Thông báo", MessageBoxButtons.OK,
                                                                   MessageBoxIcon.Information);
    }
```

4.5. Hộp lựa chọn – ComboBox

Hộp ComboBox 

cho phép lưu trữ và lựa chọn một mục dữ liệu trong một hộp danh sách thả xuống.

4.5.1. Thuộc tính

Name	Tên hộp ComboBox, bắt đầu bởi tiếp đầu ngữ **cbo**
BackColor	Thiết lập mầu nền cho hộp Combo.
DataSource	Thiết lập nguồn dữ liệu cho Combo.
DropDownStyle	***DropDown*** gồm một hộp văn bản cho phép người sử dụng có thể nhập dữ liệu, kế bên có một mũi tên ↓, nhấn vào đó sẽ xổ ra một danh sách các mục dữ liệu cho phép người dùng chọn lựa. ***Simple*** luôn hiển thị sẵn danh sách các mục dữ liệu bên dưới hộp văn bản và cho phép người sử dụng có thể nhập dữ liệu vào hộp văn bản. ***DropDownList*** tương tự như DropDown nhưng người sử dụng chỉ có thể chọn các phần tử từ danh sách, khi gõ một ký tự vào hộp văn bản thì danh sách sẽ cuộn đến các phần tử được bắt đầu bởi ký tự đó.
Enabled	Nếu Enabled = False hộp Combo sẽ không hoạt động.
Font	Xác lập kiểu chữ và cỡ chữ cho hộp Combo.
ForeColor	Xác lập mầu chữ cho hộp Combo.
Items	Khởi tạo giá trị các phần tử của hộp Combo trong thời gian thiết kế.
Items.Count	Trả về tổng số phần tử của hộp Combo trong thời gian thi hành.
Items[n]	Trả về nội dung phần tử thứ n của hộp Combo trong thời gian thi hành
SelectedItem hoặc **Text**	Trả về nội dung của phần tử hiện hành đang được chọn.
SelectedIndex	Trả về số thứ tự của phần tử đang được chọn, phần tử đầu tiên có SelectedIndex=0, nếu không có phần tử nào được chọn thì SelectedIndex= -1
Sorted	True: các phần tử trong danh sách được sắp xếp theo thứ tự ABC.
TabIndex	Thứ tự truy cập khi người dùng bấm phím Tab.
Visible	True: hiển thị hộp Combo, False: ẩn hộp Combo.

4.5.2. Sự kiện

Click	Được kích hoạt khi người dùng kích chuột vào hộp Combo.
DoubleClick	Được kích hoạt khi người dùng kích đúp chuột vào hộp Combo.
GotFocus	Được kích hoạt khi người dùng chuyển tiêu điểm tới hộp Combo.
LostFocus	Được kích hoạt khi hộp Combo mất tiêu điểm.
SelectedIndex_ Changed	Được kích hoạt khi người dùng thay đổi trạng thái lựa chọn các dòng dữ liệu trong hộp văn bản.
TextChanged	Được kích hoạt khi người dùng nhập, sửa, xóa dữ liệu tại vùng văn bản của hộp Combo hoặc khi thay đổi thuộc tính Text của hộp Combo từ mã lệnh.
DropDown	Chỉ xảy ra đối với hộp Combo DropDown và DropDownList, sự kiện này được gọi ngay sau khi người dùng nhấp mũi tên để thả hộp danh sách xuống (phím tắt Alt+↓). Vì thế sự kiện này chủ yếu được sử dụng để nhập dữ liệu cho các phần tử của hộp Combo.

4.5.3. Phương thức

Add: dùng để bổ sung một phần tử cho hộp Combo trong thời gian thi hành và thường được viết trong thủ tục Form_Load. Cú pháp:

ComboName.Items.Add(Item);

Với ComboName là tên hộp Combo, Item là nội dung phần tử muốn thêm vào hộp Combo.

Ví dụ, bổ sung phần tử có giá trị "Ha Noi" vào hộp Combo cboQue:

```
cboQue.Items.Add("Ha Noi");
```

Remove: dùng để loại bỏ một phần tử của danh sách theo nội dung trong thời gian thi hành. Cú pháp:

ComboName.Items.Remove(Item);

Ví dụ, xóa phần tử có giá trị "Ha Noi" trong hộp Combo cboQue:

```
cboQue.Items.Remove("Ha Noi");
```

RemoveAt: dùng để loại bỏ một phần tử của hộp Combo theo chỉ số trong thời gian thi hành. Cú pháp:

ComboName.Items.RemoveAt(Index);

Ví dụ, xóa phần tử ở vị trí 1 trong hộp Combo cboQue:

```
cboQue.Items.RemoveAt(1);
```

Clear: dùng để loại bỏ tất cả các phần tử của hộp Combo trong thời gian thi hành. Cú pháp:

ComboName.Items.Clear();

Ví dụ, xóa tất cả các phần tử trong hộp Combo cboQue:

```
cboQue.Items.Clear();
```

Ví dụ: Giả sử có hộp combo **cboQue**, để nhập dữ liệu cho nó ta có 2 cách sau:

Cách 1: Cập nhập bằng thủ tục Form_Load:

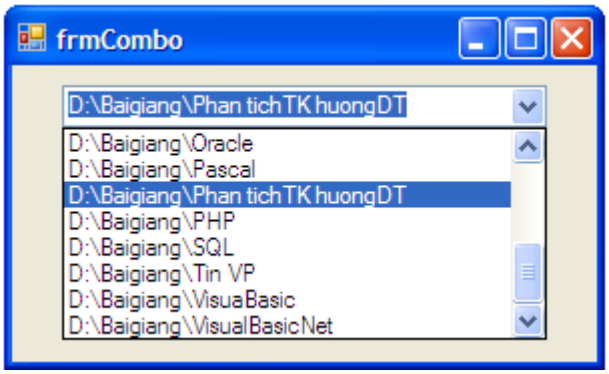
```
private void Form1_Load(object sender, EventArgs e)
 {
     cboQue.Items.Add("Hà Nội");
     cboQue.Items.Add("Nam Định");
 }
```

Cách 2: Cập nhập bằng sự kiện DropDown:

```
private void cboQue_DropDown(object sender, EventArgs e)
 {
     cboQue.Items.Clear();
     cboQue.Items.Add("Hà Nội");
     cboQue.Items.Add("Nam Định");
 }
```

Bài tập 10.

Lấy danh sách các thư mục có trong thư mục "D:\Baigiang" lưu vào hộp Combo thông qua thuộc tính DataSource.



Hình 29. Giao diện bài tập 10

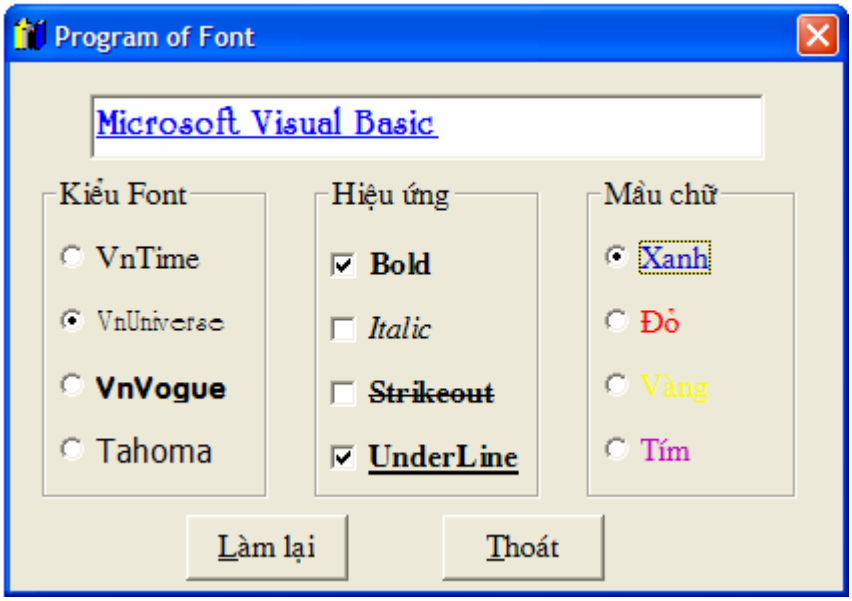
Vào Microsoft Visual Studio 2010 tạo một dự án mới, đặt một hộp combo *cboThumuc* vào form *frmCombo*.

Viết Code: mở cửa sổ soạn thảo Code và viết các đoạn mã lệnh như sau:

```
private void frmCombo_Load(object sender, EventArgs e)
 {
     string[] Folder;
     Folder = System.IO.Directory.GetDirectories("D:\\Baigiang");
     cboThumuc.DataSource = Folder;
 }
```

Bài tập 11.

Lập chương trình thực hiện các công việc thay đổi Font chữ theo giao diện dưới đây:



Hình 30. Giao diện bài tập 11

Gợi ý: Cú pháp thay đổi kiểu Font chữ:

<TênĐiềukhiển>.Font = new Font("TênFont", Cỡchữ) ;

Cú pháp thay đổi hiệu ứng font chữ:

<TênĐiềukhiển>.Font = new Font(<TênĐiềukhiển>.Font, FontStyle.HiệuỨng) ;

Trong đó Hiệu ứng có thể nhận các giá trị: Bold, Regular, Italic, UnderLine, Strikeout. Để kết hợp các hiệu ứng ta dùng toán tử |, để loại trừ các hiệu ứng ta dùng toán tử ^, để lấy hiệu ứng của điều khiển ta dùng cú pháp <Tênđiềukhiển>.Font.Style

Cú pháp thay đổi mầu chữ:

<TênĐiềukhiển>.ForeColor = Color.Màu ;

Bài tập 12.

Lập chương trình ghép tên nước và tên thành phố theo giao diện và yêu cầu dưới đây:

Hình 31. Giao diện bài tập 12

+ Viết thủ tục **EmptyOption()** bỏ chọn tất cả các RadioButton tên thành phố.

+ Khi kích chọn vào một nước, giả sử France thì xuất hiện dòng thông báo: "Hãy chọn thành phố cho France" và gọi thủ tục EmptyOption

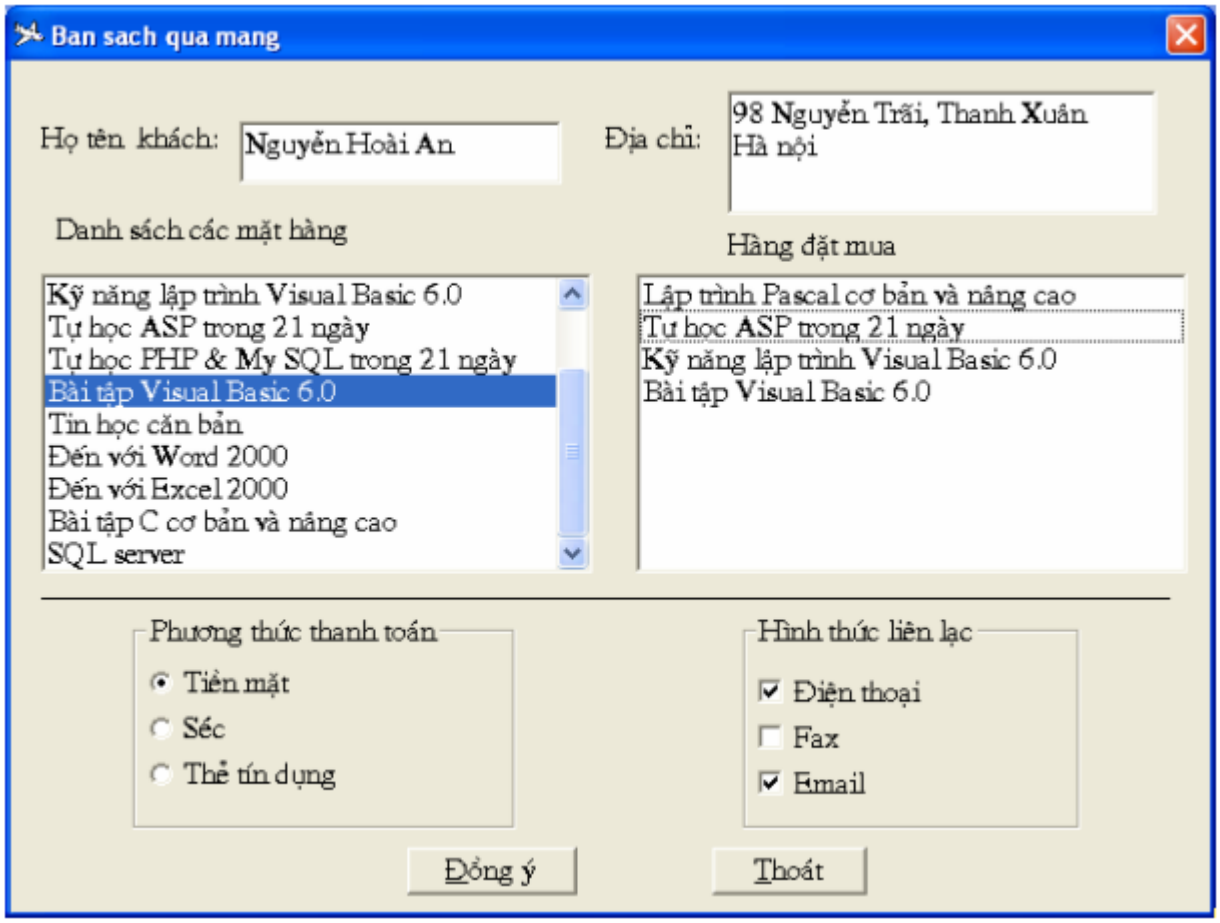
+ Khi kích chọn một thành phố, nếu đúng là thành phố của tên nước đã chọn thì xuất hiện dòng thông báo, ví dụ: "Chúc mừng bạn, thủ đô của France là Paris", ngược lại thông báo, ví dụ: "Bạn sai rồi, thủ đô của France không phải là London"

Bài tập 13.

Lập chương trình thực hiện các công việc theo giao diện và yêu cầu dưới đây:

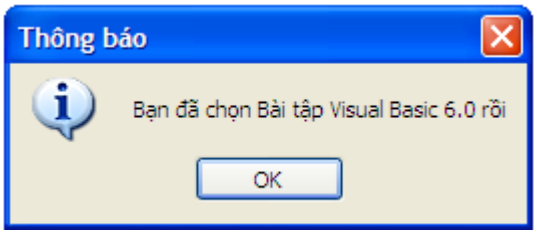
+ Chương trình có một Form bán hàng trực tuyến, danh sách các mặt hàng được hiển thị sẵn trong hộp Listbox hoặc CheckedListBox "Danh sách các mặt hàng"

+ Để mua hàng người dùng kích đúp vào mặt hàng cần mua trong "Danh sách các mặt hàng", mặt hàng được chọn sẽ được hiển thị vào trong "Hàng đặt mua".

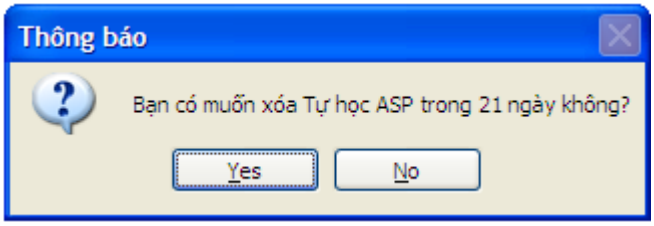


Hình 32. Giao diện bài tập 13

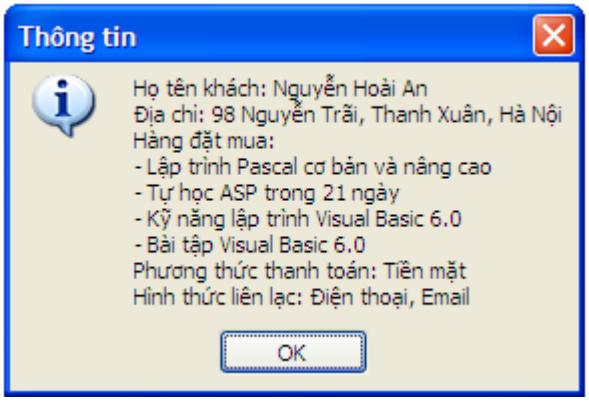
Chú ý: Khi mua hàng phải kiểm tra nếu mặt hàng này đã được mua thì dùng hộp thoại thông báo đã chọn mặt hàng đó và không được mua mặt hàng đó nữa.



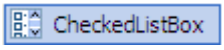
Người dùng có thể xoá các mặt hàng trong danh sách các mặt hàng đã chọn bằng cách kích đúp vào mặt hàng cần xoá, trước khi xoá phải hỏi lại người dùng có muốn xoá mặt hàng đó hay không?



+ Khi kích chuột vào nút "Đồng ý" kiểm tra người dùng phải nhập đầy đủ thông tin và hiện thông báo gồm các thông tin: Tên khách, Địa chỉ, Danh sách các mặt hàng đã mua, Phương thức thanh toán và Hình thức liên lạc.



4.6. Điều khiển CheckedListBox

Điều khiển CheckedListBox 

cho phép lưu trữ và hiển thị các mục dữ liệu theo dòng và có một hộp CheckBox ở đầu dòng.

Điều khiển CheckedListBox có tiếp đầu ngữ là clb và có các thuộc tính, sự kiện, phương thức tương tự như điều khiển ListBox.

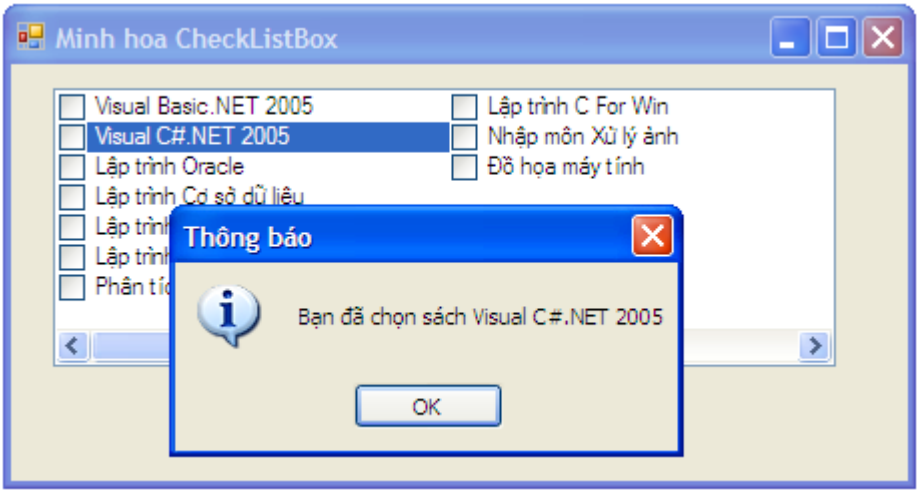
Ngoài ra nó có thêm một số thuộc tính và sự kiện khác như sau:

CheckedItems	Thuộc tính này trả về tập các phần tử được Check.
ItemCheck	Được kích hoạt khi người dùng kích đúp chuột vào hộp Combo.

Bài tập 14.

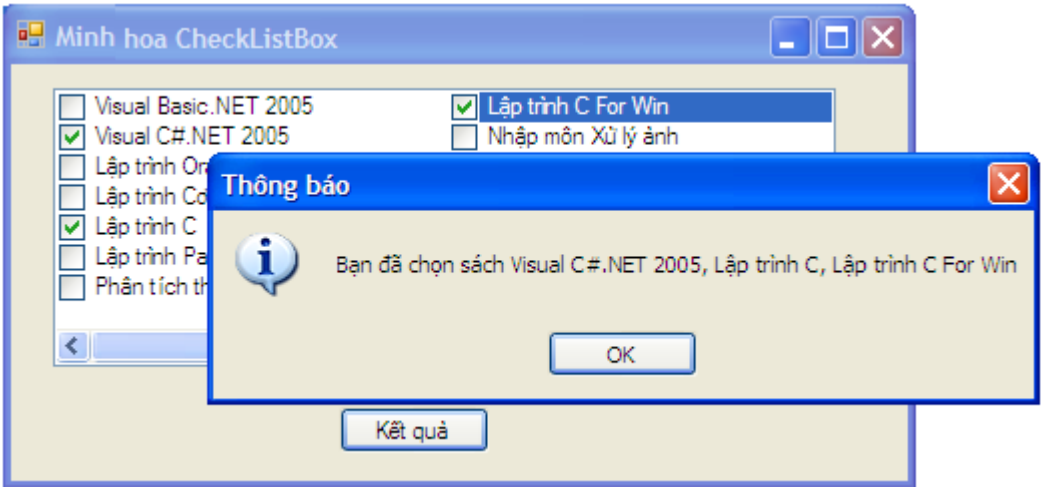
Tạo một form mới đặt tên là frmCheckListBox, đặt lên form hộp điều khiển CheckedListBox *clbSach* và nút lệnh *btnKetqua*.

Viết chương trình hiển thị 10 loại sách được chia thành 2 cột vào trong hộp clbSach. Khi người dùng đánh dấu vào hộp CheckBox của từng phần tử, sẽ xuất hiện hộp thông báo tên quyển sách tương ứng với phần tử đó, như hình minh họa sau:



Hình 33. Giao diện bài tập 14

Khi người dùng kích chọn nút *Kết quả* sẽ xuất hiện hộp thoại thông báo tên tất cả các quyển sách người dùng đã chọn.

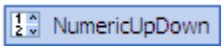


Viết Code: mở cửa sổ soạn thảo Code và viết các đoạn mã lệnh như sau:

```
private void Form1_Load(object sender, EventArgs e)
 {
     clbSach.Items.Add("Visual Basic.NET 2005");
     clbSach.Items.Add("Visual C#.NET 2005");
     clbSach.Items.Add("Lập trình Oracle");
     clbSach.Items.Add("Lập trình Cơ sở dữ liệu");
     clbSach.Items.Add("Lập trình C");
     clbSach.Items.Add("Lập trình Pascal");
     clbSach.Items.Add("Phân tích thiết kế hướng đối tượng");
     clbSach.Items.Add("Lập trình C For Win");
     clbSach.Items.Add("Nhập môn Xử lý ảnh");
     clbSach.MultiColumn = true;
     clbSach.ColumnWidth = clbSach.Width / 2;
 }
```

```
private void btnKetqua_Click(object sender, EventArgs e)
{
    string str="";
       // Duyệt qua từng phần tử đã chọn
    foreach (string item in clbSach.CheckedItems)
        str = str + item + ", ";
      // Xóa dấu phẩy và dấu cách thừa ở cuối chuỗi str
    str = str.Remove(str.Length - 2, 2);
    MessageBox.Show("Bạn đã chọn sách " + str, "Thông báo", MessageBoxButtons.OK,
                                                        MessageBoxIcon.Information);
}

private void clbSach_ItemCheck(object sender, ItemCheckEventArgs e)
{
    // Dựa vào thuộc tính NewValue của đối e để biết trạng thái của hộp Checkbox
    if (e.NewValue == CheckState.Checked)
        MessageBox.Show("Bạn đã chọn sách " + clbSach.Text, "Thông báo",
                                    MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

4.7. Điều khiển NumericUpDown

Điều khiển NumericUpDown 

cho phép người dùng lựa chọn một giá trị số trong một khoảng giá trị với một bước nhảy xác định.

4.7.1. Thuộc tính

Name	Tên điều khiển NumericUpDown, bắt đầu bởi tiếp đầu ngữ **nud**
Increment	Giá trị của bước nhảy.
Maximum	Cận trên của khoảng giá trị.
Minimum	Cận dưới của khoảng giá trị.
Value	Giá trị hiện tại của điều khiển NumericUpDown.

4.7.2. Sự kiện

ValueChanged	Được kích hoạt khi người dùng thay đổi giá trị của điều khiển.

Bài tập 15.

Dùng điều khiển NumericUpDown để giải phương trình bậc hai đủ:

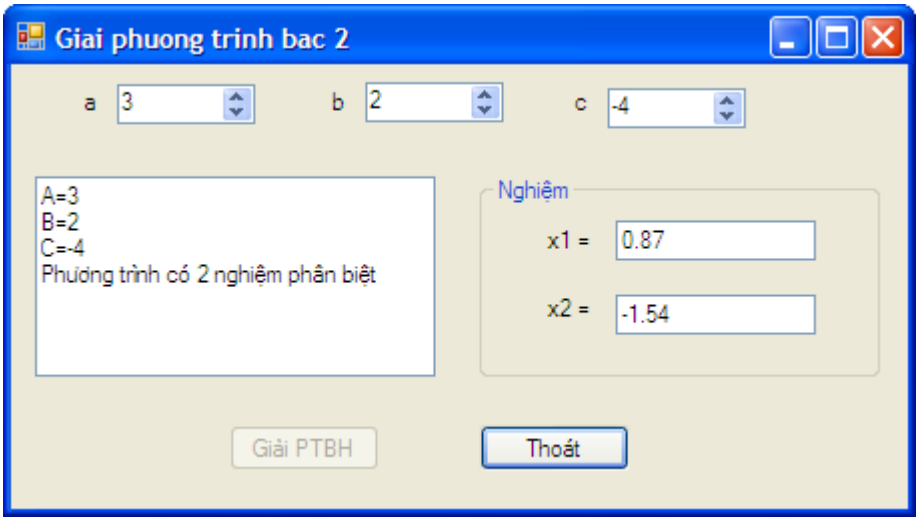
$$ax^2 + bx + c = 0 \text{ (a, b, c có giá trị trong đoạn [-100, 100])}$$

Mở một đồ án mới và thiết kế giao diện với các điều khiển như sau:

+ Ba điều khiển NumericUpDown có tên *nudNhapA, nudNhapB, nudNhapC*. Các thuộc tính: *Maximum = 100, Minimum = -100, Increment = 1*.

+ Ba hộp Textbox có tên *txtKetqua, txtX1, txtX2*. Các thuộc tính: *ReadOnly = True, BackColor =White.* Hộp kết quả có thuộc tính *Multiline = True.*

+ Hai nút lệnh *btnGiaiPTBH, btnThoat* và các nhãn Label…



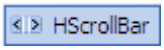
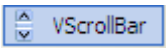
Hình 34. Giao diện bài tập 15

Viết Code: mở cửa sổ soạn thảo Code và viết các đoạn mã lệnh như sau:

```
private void btnGiaiPTBH_Click(object sender, EventArgs e)
 {
     PTBH(Convert.ToInt32(nudNhapA.Value), Convert.ToInt32(nudNhapB.Value),
                                                 Convert.ToInt32(nudNhapC.Value));
 }

 private void PTBH(int a, int b, int c)
 {
     double delta ,x1,x2;
     if ((a == 0) || (b == 0) || (c == 0))
     {
         txtKetqua.Text = "Không giải vì không phải phương trình bậc hai đủ";
         btnGiaiPTBH.Enabled = false ;
         return;
     }
     delta = b * b - 4 * a * c;
     if (delta < 0)
     {
         txtKetqua.Text = txtKetqua.Text + "Phương trình vô nghiệm";
         btnGiaiPTBH.Enabled = false;
         return;
     }
     if (delta == 0)
     {
         x1 = Math.Round ((-b / (2.0 * a)), 2);
         txtKetqua.Text = txtKetqua.Text + "Phương trình có nghiệm kép";
         txtX1.Text = Convert.ToString(x1);
         txtX2.Text = Convert.ToString(x1);
```

```
            btnGiaiPTBH.Enabled = false;
            return;
        }
        x1 = Math.Round((-b + Math.Sqrt(delta)) / (2 * a), 2);
        x2 = Math.Round((-b - Math.Sqrt(delta)) / (2 * a), 2);
        txtKetqua.Text = txtKetqua.Text + "Phương trình có 2 nghiệm phân biệt";
        txtX1.Text = Convert.ToString(x1);
        txtX2.Text = Convert.ToString(x2);
        btnGiaiPTBH.Enabled = false;
    }

    private void EmptyText()
    {
        txtX1.Text = "";
        txtX2.Text = "";
    }

    private void nudNhapA_ValueChanged(object sender, EventArgs e)
    {
        txtKetqua.Text = "A=" + nudNhapA.Value + System.Environment.NewLine;
        txtKetqua.Text = txtKetqua.Text + "B=" + nudNhapB.Value +
                                                    System.Environment.NewLine;
        txtKetqua.Text = txtKetqua.Text + "C=" + nudNhapC.Value +
                                                    System.Environment.NewLine;
        btnGiaiPTBH.Enabled = true;
        EmptyText();
    }

    // Thực hiện tương tự cho nudNhapB_ValueChanged và nudNhapC_ValueChanged
```

4.8. Thanh cuộn HScrollBar và VScrollBar

Thanh cuộn ngang HScrollBar  và thanh cuộn dọc VScrollBar  cho phép người dùng lựa chọn một giá trị số trong một khoảng giá trị xác định với một bước nhảy cho trước.

4.8.1. Thuộc tính

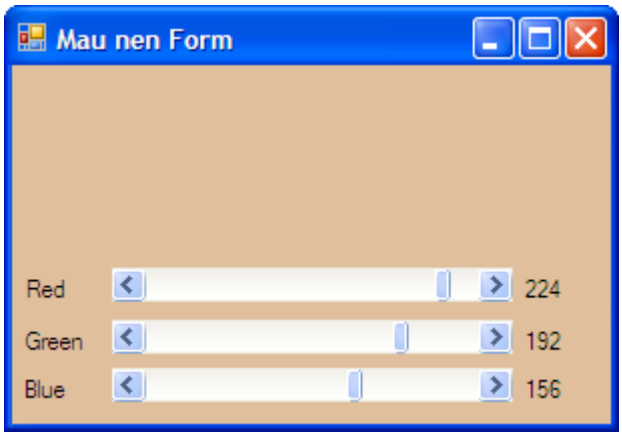
Name	Tên thanh cuộn, bắt đầu bởi tiếp đầu ngữ **hsb** và **vsb.**
Minimum	Số nguyên xác định giá trị nhỏ nhất cho thanh cuộn.
Maximum	Số nguyên xác định giá trị lớn nhất cho thanh cuộn.
Value	Cho biết giá trị hiện thời của thanh cuộn.
LargeChange	Chỉ ra mức độ thay đổi của thuộc tính Value khi người dùng nhấn chuột trên thanh cuộn.
SmallChange	Chỉ ra mức độ thay đổi của thuộc tính Value khi người dùng nhấn chuột vào các mũi tên trên thanh cuộn (giá trị mặc định = 1).

4.8.2. Sự kiện

ValueChanged	Được kích hoạt khi người dùng thay đổi giá trị của thanh cuộn.
Scroll	Xảy ra khi người dùng kéo rê chuột hoặc kích chuột vào các mũi tên trên thanh cuộn.

Bài tập 16.

Dùng thanh cuộn HScrollBar để thay đổi mầu nền cho Form.



Hình 35. Giao diện bài tập 16

Vào Microsoft Visual Studio 2010 tạo một dự án mới, đặt tên form là *frmMaunen* và đặt vào form các điều khiển sau:

+ Ba điều khiển HScrollBar có tên *hsbRed, hsbGreen, hsbBlue*. Các thuộc tính: *Maximum = 255, Minimum = 0, LargeChange = 10, SmallChange = 1.*

+ Ba nhãn Label đặt ở bên phải các thanh cuộn có tên *lblRed, lblGreen, lblBlue* để lưu giá trị hiện thời của các thanh cuộn tương ứng.

Viết Code: mở cửa sổ soạn thảo Code và viết các đoạn mã lệnh như sau:

```
private void hsbRed_ValueChanged(object sender, EventArgs e)
{
    this.BackColor = Color.FromArgb(hsbRed.Value, hsbGreen.Value, hsbBlue.Value);
    lblRed.Text = hsbRed.Value.ToString ();
}

private void hsbGreen_ValueChanged(object sender, EventArgs e)
{
    this.BackColor = Color.FromArgb(hsbRed.Value, hsbGreen.Value, hsbBlue.Value);
    lblGreen.Text = hsbGreen.Value.ToString();
}

private void hsbBlue_ValueChanged(object sender, EventArgs e)
{
```

```
        this.BackColor = Color.FromArgb(hsbRed.Value, hsbGreen.Value, hsbBlue.Value);
        lblBlue.Text = hsbBlue.Value.ToString();
    }
```

Chú ý: *Color.FromArgb(Red, Green, Blue)* cho phép tạo một màu bằng cách kết hợp các giá trị của 3 mầu cơ bản Red, Green, Blue. Ba mầu này nhận các giá trị từ 0 đến 255.

4.9. Điều khiển Timer

Điều khiển định thời gian Timer 

cho phép thực thi lại một hành động sau một khoảng thời gian xác định.

Khi ta đưa điều khiển Timer vào Form nó không xuất hiện trên Form mà xuất hiện như một biểu tượng ở trên một khay đặt ở cuối cửa sổ thiết kế. Khi chạy chương trình điều khiển Timer cũng không xuất hiện.

4.9.1. Thuộc tính

Name	Tên điều khiển Timer, bắt đầu bởi tiếp đầu ngữ **tmr**
Interval	= n là chu kỳ thực hiện sự kiện Tick của điều khiển Timer. n là số nguyên, được tính bằng mili giây và có giá trị >0
Enabled	Enabled = True: cho phép điều khiển Timer hoạt động Enabled = False: không cho phép điều khiển Timer hoạt động.

4.9.2. Sự kiện

Tick	Sự kiện này được kích hoạt sau mỗi chu kỳ Interval.

4.9.3. Phương thức

Start: kích hoạt điều khiển Timer, phương thức này tương tự thuộc tính Enabled = true. Cú pháp:

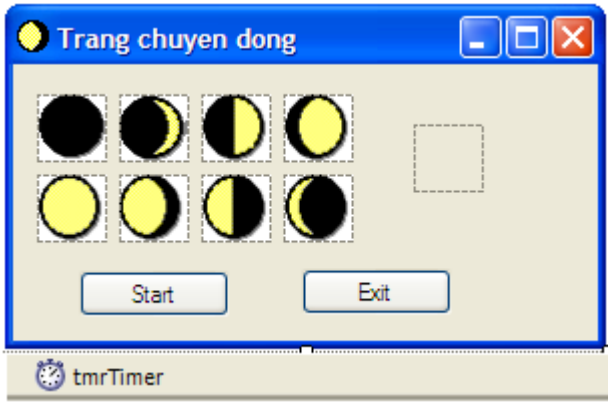
TimerName.Start();

Stop: dừng điều khiển Timer, phương thức này tương đương với thuộc tính Enabled = false. Cú pháp:

TimerName.Stop();

Bài tập 17.

Viết chương trình mô tả sự chuyển động của mặt trăng gồm 8 hình ảnh đặt trong thư mục D:\Icons\Moon như sau:



Hình 36. Giao diện bài tập 17

Vào Microsoft Visual Studio 2010 tạo một dự án mới và thiết lập các thuộc tính của các điều khiển như sau:

Điều khiển	Name	Text	Image	Visible
Form1	frmMoon	Trang chuyen dong		
PictureBox1	pic1		Moon01.bmp	False
PictureBox2	pic2		Moon02.bmp	False
PictureBox3	pic3		Moon03.bmp	False
PictureBox4	pic4		Moon04.bmp	False
PictureBox5	pic5		Moon05.bmp	False
PictureBox6	pic6		Moon06.bmp	False
PictureBox7	pic7		Moon07.bmp	False
PictureBox8	pic8		Moon08.bmp	False
PictureBox9	pic			True
Button1	btnStart	Start		True
Button2	btnExit	Exit		True
Timer	tmrTimer	Enabled = True và Interval = 250		

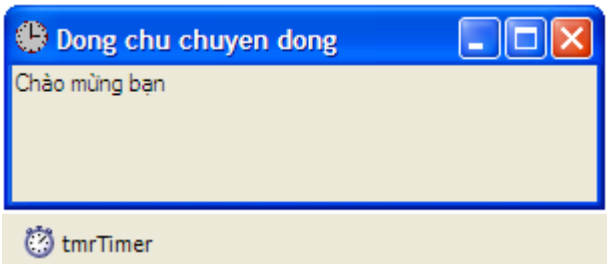
Viết Code: mở cửa sổ soạn thảo Code và viết các đoạn mã lệnh như sau:

```
public partial class frmMoon : Form
{
 int Coquay, Curmoon;

 public frmMoon()
 {
     InitializeComponent();
 }
 private void frmMoon_Load(object sender, EventArgs e)
```

```
        {
            Coquay = 0 ;        // Không cho phép quay
            Curmoon = 1 ;       // Chỉ số ảnh hiện hành
        }
        private void btnStart_Click(object sender, EventArgs e)
        {
            if (Coquay ==0)
            {
                Coquay = 1;
                btnStart.Text = "Stop";
            }
            else
            {
                Coquay = 0;
                btnStart.Text = "Start";
            }
        }
        private void Hienthi (int index)
        {
            switch (index)
            {
                case 1:
                    pic.Image = pic1.Image;
                    break;
                case 2:
                    pic.Image = pic2.Image;
                    break;
                case 3:
                    pic.Image = pic3.Image;
                    break;
                case 4:
                    pic.Image = pic4.Image;
                    break;
                case 5:
                    pic.Image = pic5.Image;
                    break;
                case 6:
                    pic.Image = pic6.Image;
                    break;
                case 7:
                    pic.Image = pic7.Image;
                    break;
                case 8:
                    pic.Image = pic8.Image;
                    break;
            }
        }
        private void tmrTimer_Tick(object sender, EventArgs e)
        {
            if (Coquay == 1 )
                Hienthi(Curmoon);
            Curmoon = Curmoon + 1;
            if (Curmoon == 9)
                Curmoon = 1;
        }
    }
```

Bài tập 18.

Viết chương trình tạo dòng chữ "Chào mừng bạn" chuyển động từ trái qua phải màn hình, khi gặp mép màn hình thì chuyển động ngược lại từ phải qua trái. Cứ lặp lại như vậy cho đến khi kết thúc chương trình.



Hình 37. Giao diện bài tập 18

Vào Microsoft Visual Studio 2010 tạo một dự án mới và thiết lập các thuộc tính của các điều khiển như sau:

Điều khiển	Name	Text
Form1	frmMove	Dong chu chuyen dong
Label1	lblMove	Chào mừng bạn
Timer	tmrTimer	Enabled = True và Interval = 200

Viết Code: mở cửa sổ soạn thảo Code và viết các đoạn mã lệnh như sau:

```
public partial class frmMove : Form
 {
    bool Kt;         //Kt=true chuyen dong sang phai, Kt=false chuyen dong sang trai

    public frmMove()
    {
     InitializeComponent();
    }

    private void frmMove_Load(object sender, EventArgs e)
    {
     Kt = true;
    }

    private void tmrTimer_Tick(object sender, EventArgs e)
    {
     if (Kt)
     {
         if (lblMove.Left + lblMove.Width < this.Width)
             lblMove.Left = lblMove.Left + 13;
         else
             Kt = false;
     }
```

```
            if (!Kt)
            {
                if (lblMove.Left > 0 )
                    lblMove.Left = lblMove.Left - 13;
                else
                    Kt = true;
            }
        }
    }
```

4.10. Điều khiển RichTextBox

Điều khiển RichTextBox 

cho phép tạo và hiển thị các tệp văn bản có phần mở rộng '.rtf'. RichTextBox khác với TextBox ở chỗ nó cho phép hiển thị màu sắc, loại Font, cỡ chữ… khác nhau cho từng đoạn khác nhau trong văn bản.

Để tạo một tệp có phần mở rộng .rtf ta có thể sử dụng trình soạn thảo Word và lưu tệp – Save As theo định dạng *Rich TextFormat (*.rtf)*

Điều khiển RichTextBox có tiếp đầu ngữ là **rtb** và có 2 phương thức cơ bản sau:

LoadFile: nạp nội dung một tệp .rtf vào RichTextBox.

Ví dụ nạp nội dung tệp *Bai1.rtf* trong ổ D vào hộp RichTextBox *rtbBai1*:

```
rtbBai1.LoadFile("D:\\Bai1.rtf", RichTextBoxStreamType.RichText);
```

SaveFile: lưu nội dung trong hộp RichTextBox vào một tệp có phần mở rộng .rtf.

Ví dụ lưu nội dung của hộp RichTextBox *rtbBai1* vào tệp *Bai2.rtf* trong ổ D: *(nếu tệp Bai2.rtf chưa tồn tại thì sẽ được tạo mới)*

```
rtbBai1.SaveFile("D:\\Bai2.rtf", RichTextBoxStreamType.RichText);
```

CHƯƠNG 5.

CÁC HỘP THOẠI THÔNG DỤNG

Chúng ta thấy trong hầu hết các ứng dụng của Windows đều có các hộp thoại Open để mở tập tin, Save để lưu các tập tin hoặc các hộp thoại Color để chọn màu, Font để chọn phông chữ... các hộp thoại này gọi là các hộp thoại thông dụng - Common Dialog.

Để sử dụng các hộp thoại thông dụng ta dùng các lớp đối tượng .NET trong thư viện **System.IO** bao gồm 5 lớp tương ứng với 5 hộp hội thoại cơ bản như sau:

Hộp thoại	Lớp đối tượng
Mở tập tin – Open File	OpenFileDialog()
Lưu tập tin – Save File	SaveFileDialog()
Chọn màu – Color	ColorDialog()
Chọn phông – Font	FontDialog()
In ấn – Print	PrintDialog()

1. Hộp hội thoại Open File

Các thuộc tính và phương thức quan trọng của hộp hội thoại OpenFile:

Thuộc tính	Chức năng
FileName	Cung cấp tên và đường dẫn của tập tin đã chọn.
Filter	Xác định danh sách các bộ lọc tập tin mà hộp hội thoại sẽ hiển thị.Ví dụ: “Text|*.txt|Icons|*.ico|All files|*.*” (không được chứa dấu cách)
FilterIndex	Chỉ ra bộ lọc tập tin mặc định, giả sử có 3 bộ lọc (*.com), (*.exe) và (*.ico) nếu FilterIndex = 2 thì hộp thoại sẽ hiển thị sẵn bộ lọc (*.exe)
InitialDirectory	Xác định thư mục mặc định cho hộp hội thoại khi vừa được gọi.
Multiselect	Multiselect = True: cho phép người dùng chọn đồng thời nhiều file.
FileNames	Cung cấp tên và đường dẫn của các tập tin đã chọn.
Title	Xác định tiêu đề của hộp hội thoại.
OpenFile	Mở nội dung File đã được chọn (ReadOnly).

Bài tập 19.

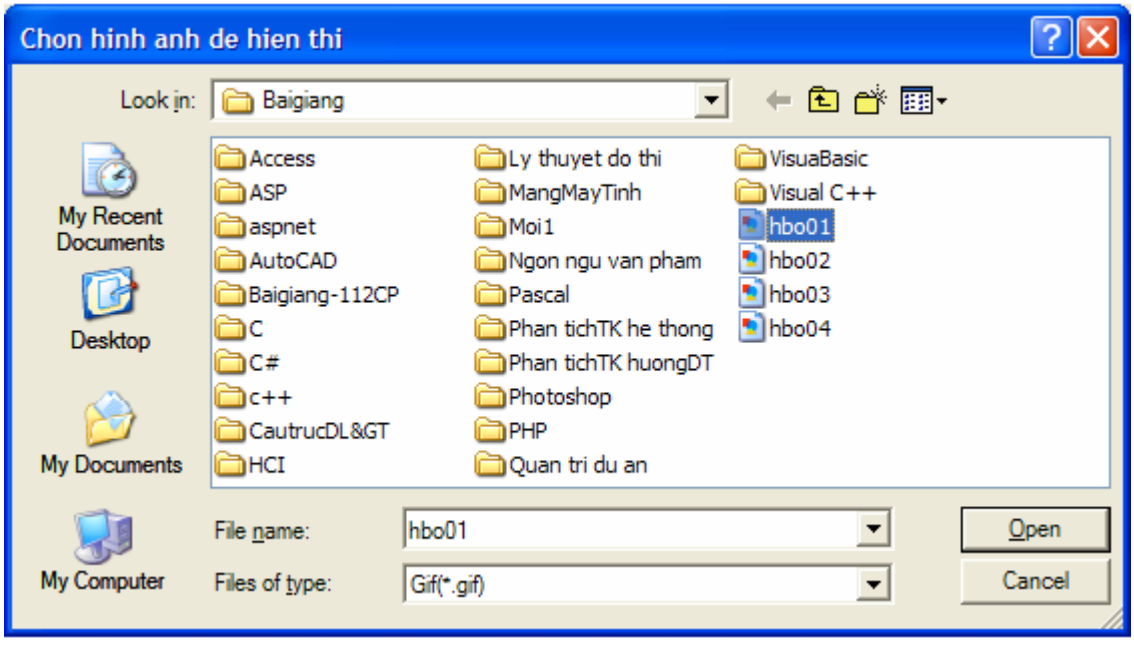
Viết chương trình cho phép tìm kiếm các hình ảnh để hiển thị lên form.

Hình 38. Giao diện bài tập 19

Vào Microsoft Visual Studio 2010 tạo dự án mới đặt tên là **OpenDialog** lưu vào thư mục *D:\Baigiang\C#HVNH\Baitap* và thiết lập thuộc tính của các điều khiển như sau:

Điều khiển	Name	Text
Form1	frmPicture	Hien thi hinh anh
PictureBox	picAnh	
Button1	btnOpen	Open

Chạy chương trình, kích chọn nút lệnh Open sẽ xuất hiện hộp thoại sau:



Chọn một ảnh bất kỳ muốn hiển thị sau đó kích chọn **Open**, ảnh sẽ được hiển thị lên điều khiển PictureBox trên form.

Trong trường hợp người dùng chọn **Cancel** sẽ xuất hiện thông báo: "You clicked Cancel!"

Viết Code: mở cửa sổ soạn thảo Code và viết các đoạn mã lệnh như sau:

```
private void btnOpen_Click(object sender, EventArgs e)
 {
     OpenFileDialog dlgOpen = new OpenFileDialog();
     dlgOpen.Filter = "Bitmap(*.bmp)|*.bmp|Gif(*.gif) |*.gif|All files(*.*)|*.*";
     dlgOpen.InitialDirectory = "D:\\Baigiang";
     dlgOpen.FilterIndex = 2;
     dlgOpen.Title = "Chon hinh anh de hien thi";
     if (dlgOpen.ShowDialog() == DialogResult.OK)
         picAnh.Image = Image.FromFile(dlgOpen.FileName);
     else
         MessageBox.Show("You clicked Cancel" , "Open Dialog", MessageBoxButtons.OK,
                                                              MessageBoxIcon.Information);
 }
```

2. Hộp thoại SaveFile và luồng FileStream

2.1. Hộp thoại SaveFile

Các thuộc tính và phương thức của hộp hội thoại SaveFile:

Thuộc tính	Chức năng
FileName	Cung cấp tên và đường dẫn của tập tin đã chọn.
Filter	Xác định danh sách các bộ lọc tập tin mà hộp hội thoại sẽ hiển thị, ví dụ: "Text|*.txt|Icons|*.ico|All files|*.*"
FilterIndex	Chỉ ra bộ lọc tập tin mặc định, giả sử có 3 bộ lọc (*.com), (*.exe) và (*.ico) nếu FilterIndex = 2 thì hộp thoại sẽ hiển thị sẵn bộ lọc (*.exe)
AddExtension	= True tự động thêm phần mở rộng hiện hành vào tên tệp mà người dùng chọn nếu người dùng không chỉ rõ phần mở rộng của tên tệp.
DefaultExt	Cung cấp phần mở rộng mặc định cho tên tệp, nếu người dùng không chỉ rõ phần mở rộng của tên tệp, ví dụ: ".doc"
InitialDirectory	Xác định thư mục mặc định cho hộp hội thoại khi vừa được gọi.
Title	Xác định tiêu đề của hộp hội thoại.

2.2. Luồng FileStream

Với một file dữ liệu có kiểu bất kỳ đều được coi là một luồng dữ liệu, ta có thể mở một luồng dữ liệu để đọc thông tin từ file hoặc ghi thông tin vào file sau đó đóng luồng lại.

Luồng ghi dữ liệu - StreamWriter

+ Mở luồng để ghi file:

```
StreamWriter  Tenluong = new StreamWriter(Tenfile);
```

+ Ghi từng dòng dữ liệu vào file:

```
Tenluong.WriteLine("Noidung");
```

+ Ghi tất cả dữ liệu vào file:

```
Tenluong.Write("Noidung");
```

Luồng đọc dữ liệu - StreamReader

+ Mở luồng để đọc file:

```
StreamReader Tenluong = new StreamReader(Tenfile);
```

+ Đọc từng dòng dữ liệu của file: ta dùng vòng lặp với số lần lặp không xác định để đọc từng dòng dữ liệu, nếu đọc thành công thì trả về chuỗi chứa dữ liệu đọc được, nếu đến cuối file thì trả về Nothing.

```
Noidung = Tenluong.ReadLine();
```

+ Đọc tất cả dữ liệu của file lưu vào một biến:

```
Noidung = Tenluong.ReadToEnd();
```

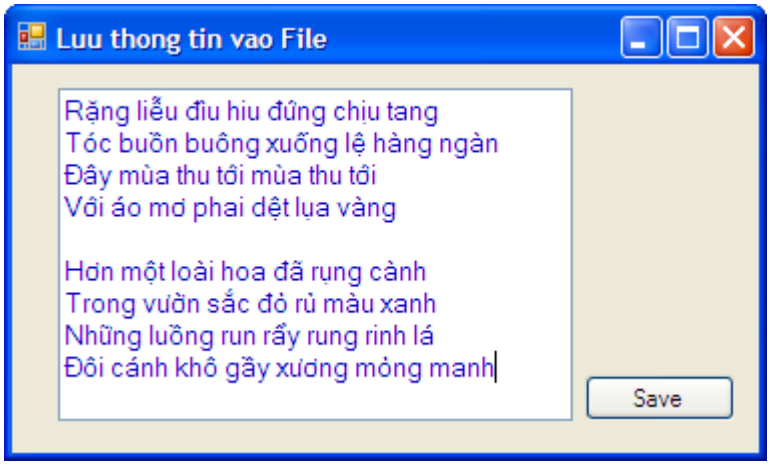
+ Đóng luồng:

```
Tenluong.Close();
```

Chú ý: StreamWriter và StreamReader nằm trong lớp System.IO

Bài tập 20.

Viết chương trình cho phép lưu nội dung của hộp Textbox vào một File trên máy tính, với giao diện và yêu cầu như sau:



Hình 39. Giao diện bài tập 20

Kích chọn nút *Save* xuất hiện hộp hội thoại *Chon file de luu* với đường dẫn mặc định là *D:\Baigiang*. Gõ tên tệp cần lưu dữ liệu vào ô *FilName*, ví dụ: *Day_mua_thu_toi.doc* và chọn *Save*. Kết quả trong thư mục *D:\Baigiang* xuất hiện tệp *Day_mua_thu_toi.doc* chứa nội dung của hộp Textbox.

Vào Microsoft Visual Studio 2010 tạo dự án mới đặt tên là **SaveDialog** lưu vào thư mục *D:\Baigiang\C#HVNH\Baitap* và thiết lập thuộc tính của các điều khiển như sau:

Điều khiển	Name	Text
Form1	frmSave	Luu thong tin vao File
TextBox (MultiLine = True)	txtSave	
Button1	btnSave	Save

Viết Code: mở cửa sổ soạn thảo Code khai báo không gian lớp IO `using System.IO;` và viết đoạn mã lệnh như sau:

```
private void btnSave_Click(object sender, EventArgs e)
 {
     SaveFileDialog dlgSave = new SaveFileDialog();
     dlgSave.Filter = "Text file(*.txt)|*.txt  |Word Document(*.doc) |*.doc|
                                                          All files(*.*)|*.*";
     dlgSave.InitialDirectory = "D:\\Baigiang";
     dlgSave.FilterIndex = 2;
     dlgSave.Title = "Chon file de luu";
     dlgSave.AddExtension = true;
     dlgSave.DefaultExt = ".doc";
     if (dlgSave.ShowDialog() == DialogResult.OK)
     {
         StreamWriter File;
         File = new StreamWriter(dlgSave.FileName);
         try
         {
                File.Write(txtSave.Text);
         }
         catch(System.Exception)
         {
                MessageBox.Show("Lỗi ghi file");
         }
             File.Close();
     }
 }
```

3. Hộp thoại Color

Hộp thoại Color có tác dụng hiển thị bảng màu hiện hành cho phép người sử dụng chọn một màu bất kỳ để tô màu chữ, màu nền…

Các thuộc tính của hộp thoại Color:

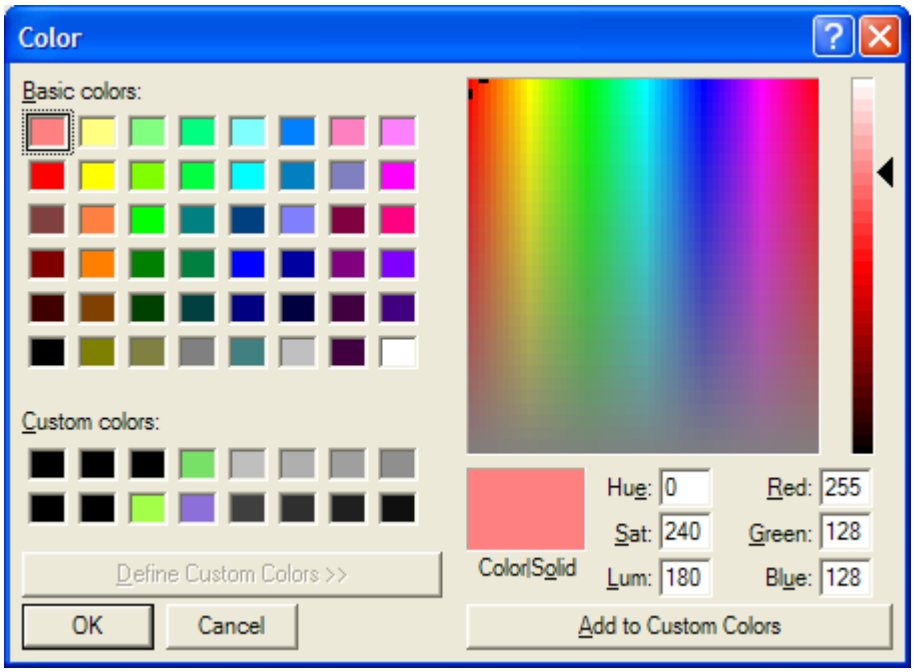
Thuộc tính	Chức năng
Color	Trả về màu được chọn trong hộp thoại Color.
FullOpen	Hiển thị toàn bộ hộp thoại Color.
SolidColorOnly	Không hiển thị phần Define Custom Colors.

Bài tập 21.

Viết chương trình cho phép lựa chọn màu nền cho form theo giao diện và yêu cầu sau:

Hình 40. Giao diện bài tập 21

Kích chọn nút lệnh *Màu nền*, xuất hiện hộp thoại Color:



Chọn một màu bất kỳ rồi bấm OK, khi đó màu được chọn sẽ được gán làm màu nền cho form.

Vào Microsoft Visual Studio 2010 tạo dự án mới có tên là **ColorDialog** lưu vào thư mục *D:\Baigiang\C#HVNH\Baitap* và thiết lập thuộc tính của các điều khiển như sau:

Điều khiển	Name	Text
Form1	frmColor	Hop thoai mau
Button1	btnColor	Màu nền

Viết Code: mở cửa sổ soạn thảo Code và viết các đoạn mã lệnh như sau:

```
private void btnColor_Click(object sender, EventArgs e)
{
    ColorDialog dlgColor = new ColorDialog();
    dlgColor.FullOpen = true;
    if (dlgColor.ShowDialog() == DialogResult.OK)
        this.BackColor = dlgColor.Color;
}
```

4. Hộp thoại Font

Có chức năng hiển thị hộp hội thoại Font cho phép người sử dụng chọn font chữ, kiểu chữ, cỡ chữ…

Các thuộc tính quan trọng của hộp thoại Font:

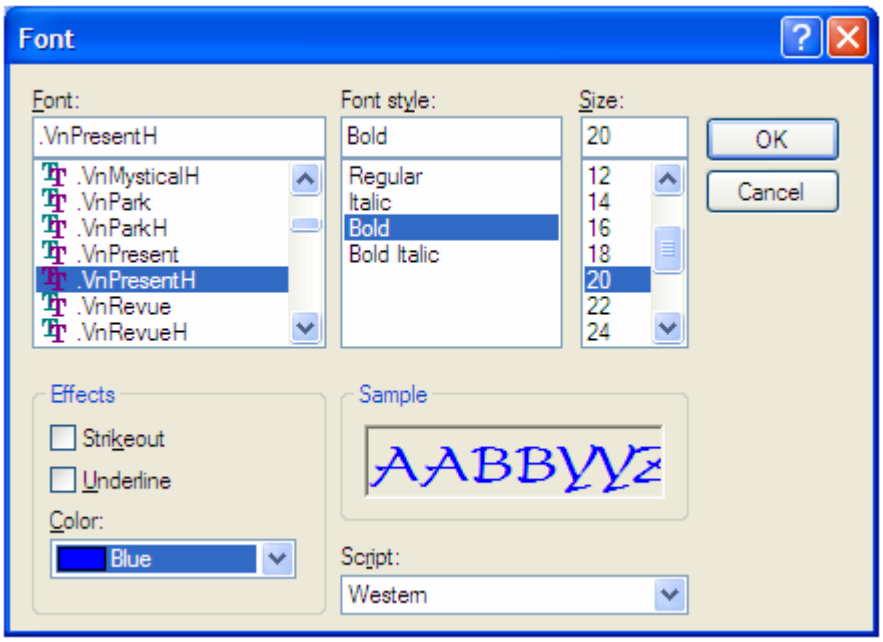
Thuộc tính	Chức năng
Font	Trả về kiểu Font chữ được chọn trong hộp thoại Font.
ShowColor	= True: cho phép hiển thị hộp thoại Color.
Color	Trả về màu được chọn trong hộp thoại Font.

Bài tập 22.

Viết chương trình cho phép lựa chọn các kiểu font chữ khác nhau cho hộp Textbox theo giao diện và yêu cầu như sau:

Hình 41. Giao diện bài tập 22

Kích chọn nút lệnh *Chọn Font*, xuất hiện hộp thoại Font:



Chọn giá trị cho các thuộc tính rồi bấm OK, kết quả sẽ được hiển thị trong hộp Textbox.

Vào Microsoft Visual Studio 2010 tạo dự án mới đặt tên là **FontDialog** lưu vào thư mục *D:\Baigiang\C#HVNH\Baitap* và thiết lập thuộc tính của các điều khiển như sau:

Điều khiển	Name	Text
Form1	frmFont	Hop thoai chon Font
TextBox1	txtFont	Hà Nội
Button1	btnFont	Chọn Font

Viết Code: mở cửa sổ soạn thảo Code và viết các đoạn mã lệnh như sau:

```
private void btnSave_Click(object sender, EventArgs e)
 {
     FontDialog  dlgFont = new FontDialog();
     dlgFont.ShowColor = true;
     if (dlgFont.ShowDialog() == DialogResult.OK)
     {
         txtFont.Font = dlgFont.Font;
         txtFont.ForeColor = dlgFont.Color;
     }
 }
```

Bài tập 23.

Viết chương trình cho phép đọc nội dung một tệp *.rtf* trong máy lưu vào hộp RichTextBox theo giao diện và yêu cầu sau:

Hình 42. Giao diện bài tập 23

Kích chọn nút *Open* xuất hiện hộp thoại *OpenFile*, chọn tệp *Main dans la main.rtf* ở thư mục *D:\Baigiang\C#HVNH\Baitap\Music\Nhac_Loi* lưu vào hộp RichTextBox.

Kích chọn nút *Font* xuất hiện hộp thoại *Font*, cho phép chọn mầu chữ và kiểu chữ cho hộp RichTextbox.

Kích chọn nút *Save* xuất hiện hộp thoại *SaveFile*, lưu nội dung hộp RichTextBox vào tệp *Main dans la main.rtf*

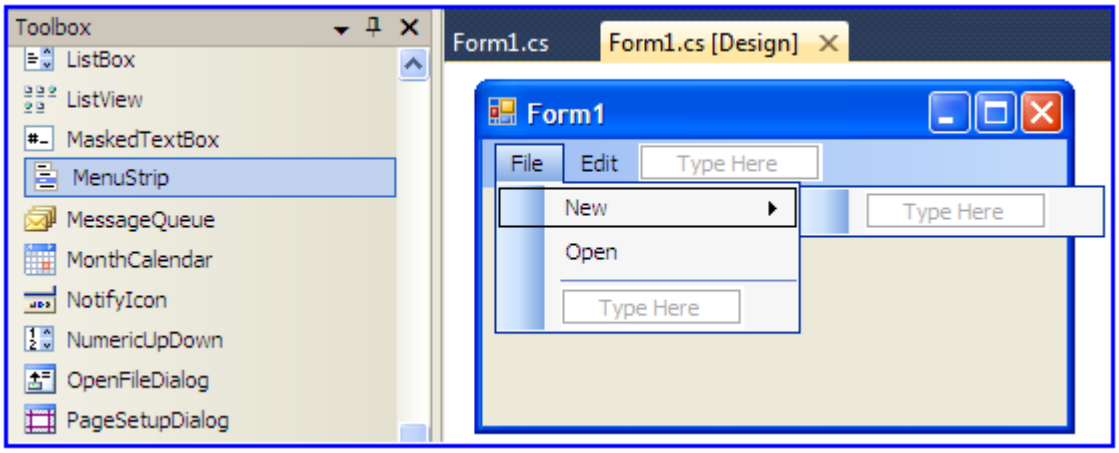
CHƯƠNG 6.

MENU VÀ ĐỒ ÁN NHIỀU BIỂU MẪU

1. Menu - MenuStrip

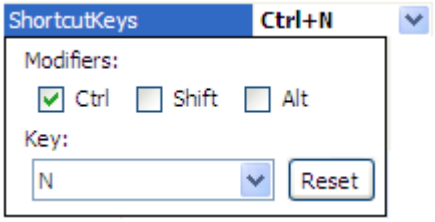
Điều khiển MenuStrip giúp người lập trình có thể thiết kế thanh menu trên Form.

Khi đưa điều khiển MenuStrip vào Form, một thanh menu ngang sẽ xuất hiện trên dòng đầu tiên của Form với các ô chứa dòng chữ *Type Here* cho phép tạo các mục menu mới.



Hình 43. Giao diện điều khiển MenuStrip

1.1. Thuộc tính

Name	Mọi mục menu đều phải có tên, menu có tiếp đầu ngữ là **mnu**
Enabled	= False: mục menu sẽ bị xám, ta không thể chọn mục menu đó.
Image	Thiết lập hình ảnh biểu tượng cho mỗi mục menu.
ShortcutKeys	Cho phép tạo phím tắt để mở các mục menu. 
Text	Tạo tiêu đề của các mục menu. Nếu đặt ký tự & trước một chữ cái trong thuộc tính Text thì khi chạy chương trình người dùng có thể bấm tổ hợp phím Alt + Chữ cái đó để kích hoạt menu. Ví dụ : &File sẽ cho phép bấm Alt+F để kích hoạt menu File.

	Nếu Text được xác lập là một dấu trừ (-) C# sẽ hiển thị một đường thẳng ngăn cách giữa các khoản mục menu.
Visible	= Flase: các mục menu sẽ không được hiển thị.
ToolTipText	Tạo dòng mách nước cho các mục menu.

1.2. Sự kiện

Click	Được kích hoạt khi người dùng kích chuột để chọn một mục menu.

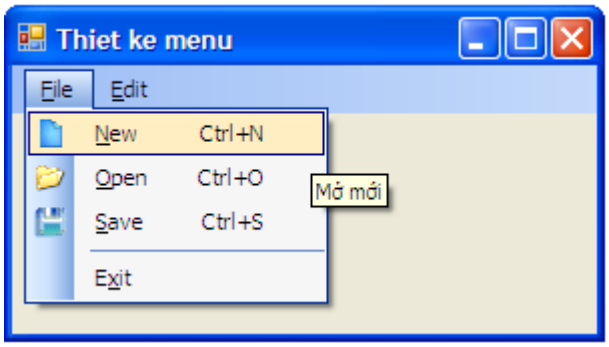
Chú ý:

+ *Chèn mục menu*: để chèn thêm một mục menu vào trong danh sách các mục menu đã có, ta kích chuột phải tại mục menu, chọn ***Insert/MenuItem***. Kết quả một mục menu mới được chèn vào trên mục menu đang được chọn.

+ *Xóa mục menu*: kích chuột phải tại mục menu muốn xóa, chọn ***Delete.***

Bài tập 24.

Tạo một menu với giao diện như sau:



Hình 44. Giao diện bài tập 24

Khi người dùng kích chuột tại mục menu nào thì xuất hiện hộp hội thoại thông báo tên mục menu đó.

Vào Microsoft Visual Studio 2010 tạo dự án mới đặt tên là **Menu** và thiết lập thuộc tính của các điều khiển như sau:

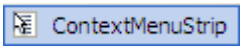
Điều khiển	Name	Text	Image	ShortcutKeys	ToolTipText
Form1	frmMenu	Thiet ke menu			
MenuStrip1					

	mnuFile	&File			
	mnuNew	&New	New.ico	Ctrl + N	Mở mới
	mnuOpen	&Open	Open.ico	Ctrl + O	Mở tệp đã có
	mnuSave	&Save	Save.ico	Ctrl + S	Lưu tệp
	mnu1	-			
	mnuExit	E&xit			
	…				

Viết Code: mở cửa sổ soạn thảo Code và viết các đoạn mã lệnh như sau:

```
private void mnuNew_Click(object sender, EventArgs e)
 {
     MessageBox.Show("Bạn đã chọn mục menu New", "Thông báo", MessageBoxButtons.OK,
                                                    MessageBoxIcon.Information);
 }

private void mnuOpen_Click(object sender, EventArgs e)
 {
     MessageBox.Show("Bạn đã chọn mục menu Open", "Thông báo", MessageBoxButtons.OK,
                                                    MessageBoxIcon.Information);
 }
……// Các mục menu khác làm tương tự
```

2. Popup menu - ContextMenuStrip

Điều khiển ContextMenuStrip 

cho phép tạo menu ngữ cảnh, menu này chỉ xuất hiện khi người dùng kích chuột phải tại điều khiển gắn menu. Ta gắn menu ngữ cảnh cho một điều khiển thông qua thuộc tính Context MenuStrip của điều khiển đó.

Điều khiển ContextMenuStrip có tiếp đầu ngữ là **cmnu** và có các thuộc tính và phương thức tương tự như điều khiển MenuStrip.

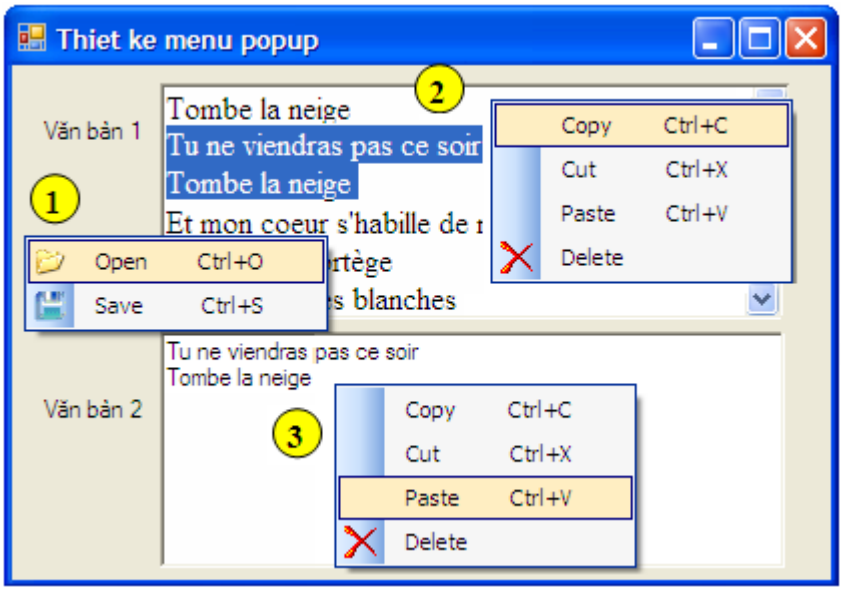
Bài tập 25.

Viết chương trình tạo menu Popup có giao diện và các chức năng như sau:

① Người dùng kích chuột phải tại vị trí bất kỳ trên Form, xuất hiện menu gồm 2 mục lựa chọn: ***Open*** và ***Save***. Kích chọn mục Open sẽ xuất hiện hộp thoại Open cho phép chọn một tệp văn bản bất kỳ có đuôi *.rtf* và hiển thị nội dung của tệp văn bản vào hộp RichTextBox *Văn bản 1*.

② Bôi đen các dòng văn bản trong hộp *Văn bản 1* và kích chuột phải tại vị trí bất kỳ trong hộp *Văn bản 1*, xuất hiện menu gồm 4 mục lựa chọn: ***Copy, Cut, Paste*** và ***Delete***. Kích chọn mục *Copy* để lưu các dòng văn bản được bôi đen vào bộ nhớ đệm ClipBoard.

③ Kích chuột phải tại vị trí bất kỳ trong hộp *Văn bản 2* chọn *Paste* để gán nội dung dòng văn bản trong ClipBoard vào hộp *Văn bản 2*.



Hình 45. Giao diện bài tập 25

Vào Microsoft Visual Studio 2010 tạo dự án mới có tên là **MenuPopup** và thiết lập giá trị cho các thuộc tính của các điều khiển như sau:

Điều khiển	Name	Text	Image	Context MenuStrip	Shortcut Keys
Form1	frmMenuPopup	Thiet ke menu popup		***cmnuFile***	
ContextMenu Strip1	cmnuFile				
	cmnuOpen	Open	Open.ico		Ctrl + O
	cmnuSave	Save	Save.ico		Ctrl + S
ContextMenu Strip2	cmnuEdit				
	cmnuCopy	Copy			Ctrl + C
	cmnuCut	Cut			Ctrl + X
	cmnuPaste	Paste			Ctrl + V
	cmnuDelete	Delete	Delete.ico		
Label1		Văn bản 1			
Label2		Văn bản 2			
RichTextBox1	rtbVanban1			***cmnuEdit***	
RichTextBox2	rtbVanban2			***cmnuEdit***	

Viết Code: mở cửa sổ soạn thảo Code và viết các đoạn mã lệnh cho 2 mục menu *cmnuCopy* và *cmnuPaste*… như sau:

```
private void cmnuCopy_Click(object sender, EventArgs e)
 {
     Clipboard.SetText(rtbVanban1.SelectedText);
 }

 private void cmunPaste_Click(object sender, EventArgs e)
 {
     //rtbVanban2.Text = Clipboard.GetText();  // cách 1
     //rtbVanban2.Text = rtbVanban2.Text.Insert(rtbVanban2.SelectionStart,
                                                Clipboard.GetText()); // Cách 2
     rtbVanban2.SelectedText = Clipboard.GetText();  // cách 3
 }

 private void mnuCut_Click(object sender, EventArgs e)
 {
     Clipboard.SetText(rtbVanban1.SelectedText);
     rtbVanban1.SelectedText = String.Empty;
 }

 // Các mục menu khác sinh viên tự làm
```

3. Đồ án nhiều biểu mẫu

Khi ứng dụng trở lên phức tạp, chương trình không thể chỉ chứa trong một biểu mẫu mà phải sử dụng nhiều biểu mẫu để bổ sung tính linh hoạt và năng lực cho ứng dụng.

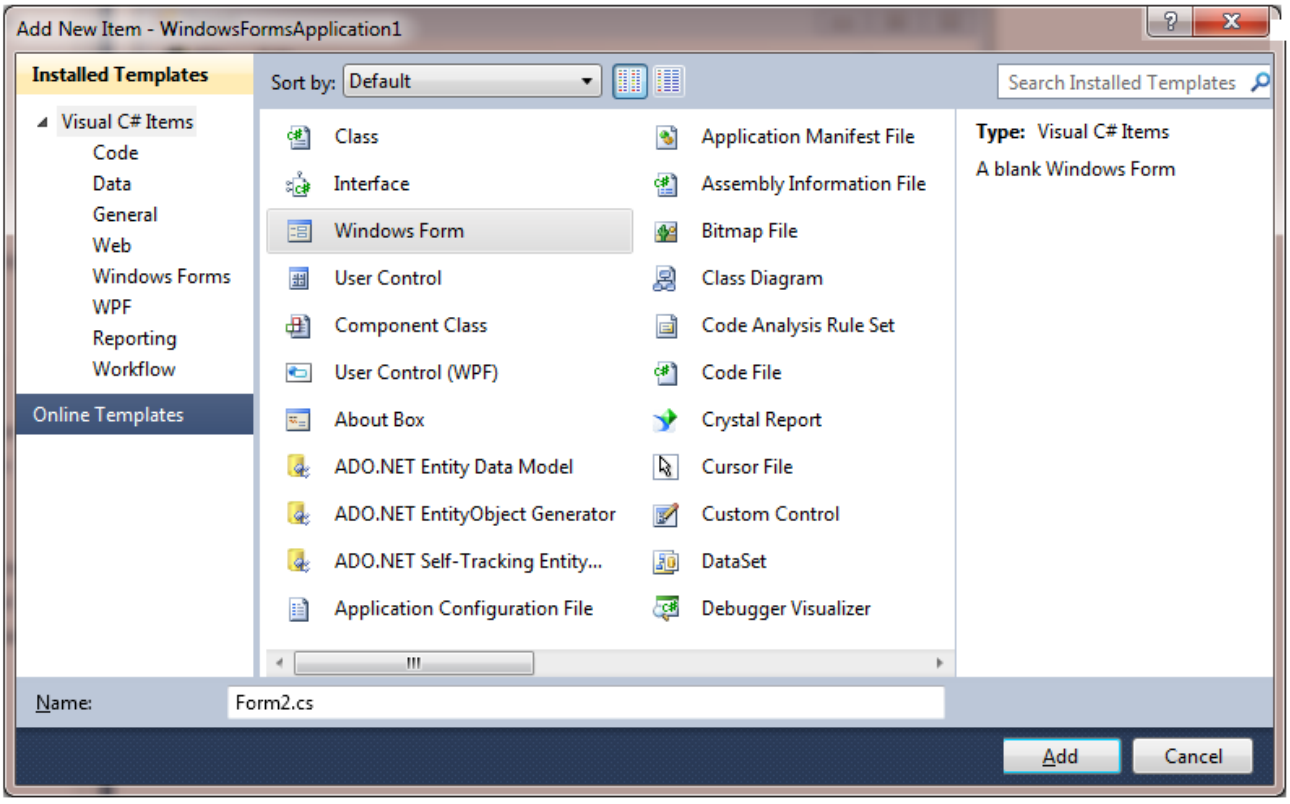
3.1. Bổ sung biểu mẫu

Để bổ sung thêm các biểu mẫu - Form cho ứng dụng đang thiết kế, ta có thể thực hiện một trong các cách sau :

+ Chọn menu *Project/Add Windows Form*…
+ Hoặc chọn menu *Project/Add New Item*…
+ Hoặc bấm phím tắt *Ctrl+Shift+A*

Kết quả xuất hiện cửa sổ Add New Item. C# cung cấp vài biểu mẫu được xây dựng sẵn để có thể bổ sung vào dự án, ví dụ: biểu mẫu cơ bản ‘Windows Form’, hoặc ‘About Box’ - dùng để đặt logo của công ty hoặc ký hiệu bản quyền...

Chọn loại biểu mẫu cần bổ sung ví dụ Windows Form, đặt tên cho form tại ô Name rồi chọn nút Add, khi đó một biểu mẫu mới sẽ được chèn vào ứng dụng.



Hình 46. Cửa sổ Add New Item

Ngoài ra C# còn cho phép bổ sung các form đã được xây dựng trong các dự án khác bằng cách chọn menu *Project/ Add Existing Item…* hoặc bấm phím tắt *Shift+Alt+A* rồi chọn Form cần bổ sung. Việc chèn Form đã có sẵn tạo điều kiện cho việc phân nhỏ ứng dụng cho nhiều người, mỗi người làm một phần sau đó ghép lại với nhau tạo thành ứng dụng hoàn chỉnh.

Bài tập 26.

Vào Microsoft Visual Studio 2010 tạo dự án mới có tên là ***English*** lưu vào thư mục *D:\Baigiang\C#HVNH\Baitap*.

+ Đổi tên cho Form1 thành ***frmMain*** và tiêu đề là ***Bai tap tieng Anh.***

+ Bổ sung thêm 2 Windows Form mới có tên là ***frmDientu1*** và ***frmBiendoi1*** với tiêu đề lần lượt là ***Bai tap Dien tu 1*** và ***Bai tap Bien doi cau***.

Lúc này dự án có 03 Form là: frmMain, frmDientu1 và frmBiendoi1.

3.2. Biểu mẫu khởi động

Khi chạy một ứng dụng, biểu mẫu được thực hiện đầu tiên gọi là biểu mẫu khởi động - Startup.

Biểu mẫu Startup mặc định là biểu mẫu đầu tiên được xây dựng trong ứng dụng khi thiết kế giao diện, ví dụ biểu mẫu frmMain trong ứng dụng English.

Ta có thể thay đổi mặc định này để có thể chọn một biểu mẫu bất kỳ làm biểu mẫu khởi động khi chạy chương trình. Ví dụ muốn form *frmDientu1* được thực hiện đầu tiên, ta thực hiện như sau:

+ Kích chuột phải tại tệp *Program.cs* trong cửa sổ Solution Explorer, chọn Open hoặc View Code.

+ Trong hàm Main của cửa sổ code sửa dòng Application.Run(new frmMain()); thành Application.Run(new frmDientu1());

3.3. Gọi biểu mẫu

Để gọi một biểu mẫu ta phải khai báo một biến có kiểu của biểu mẫu cần gọi và thực hiện gọi thông qua 2 phương thức tương ứng với 2 cách sau :

Cách 1: Dùng phương thức Show()

Ví dụ:
```
frmDientu1 frm = new frmDientu1();
frm.Show();
```

Phương thức này tải biểu mẫu vào bộ nhớ sau đó hiển thị biểu mẫu, đồng thời vẫn cho phép người sử dụng có thể tương tác được với các biểu mẫu nằm phía dưới.

Cách 2: Dùng phương thức ShowDialog()

Ví dụ:
```
frmDientu1 frm = new frmDientu1();
frm.ShowDialog();
```

Tương tự như cách 1 nhưng người sử dụng chỉ có thể tương tác được với biểu mẫu hiện hành mà không thể tương tác với các biểu mẫu nằm phía dưới nó.

3.4. Đóng biểu mẫu

Để đóng một biểu mẫu ta có thể thực hiện theo 2 cách sau:

+ Cách 1: cú pháp **this.Hide();**

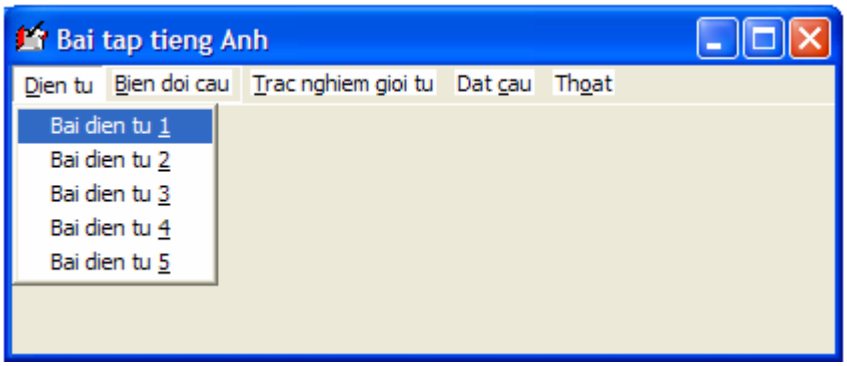
+ Cách 2: cú pháp **this.Close();**

3.5. Xoá biểu mẫu

Trong cửa sổ *Solution Explorer*, kích chuột phải tại Form cần xóa, chọn *Delete*. Kết quả Form được chọn sẽ bị xóa ra khỏi dự án.

Bài tập 27.

Tiếp theo bài tập 26, tạo MenuStrip cho Form *frmMain* theo giao diện sau:

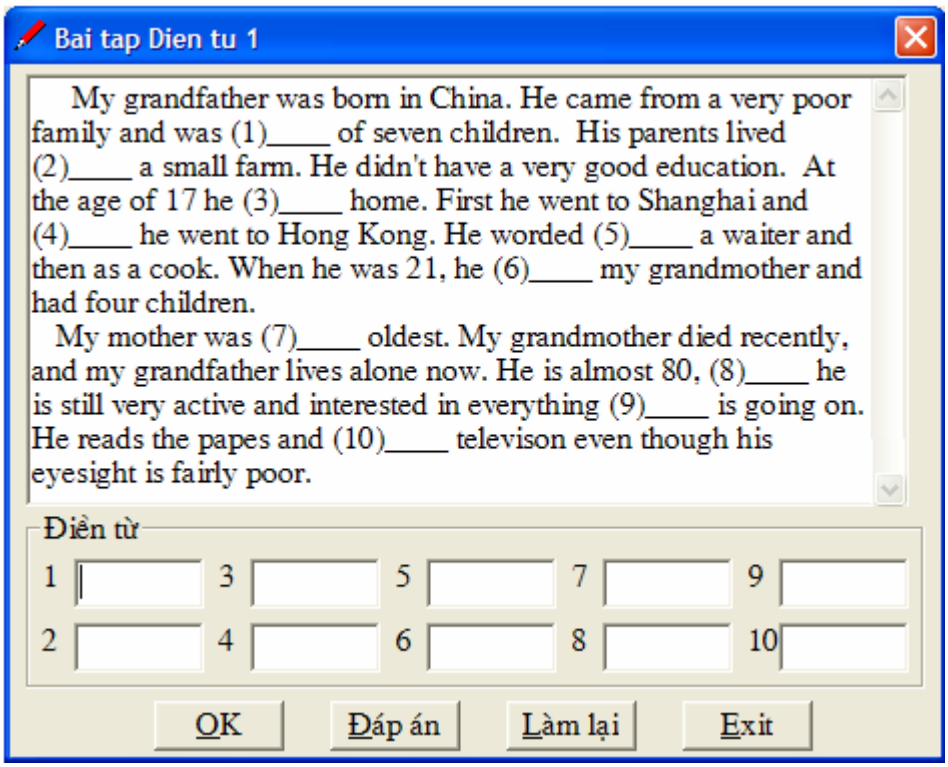


Hình 47. Giao diện bài tập 27

Đặt tên cho mục menu Dien tu là *mnuDientu*, tiêu đề là *&Dien tu*. Đặt tên cho menu con 1 của Dien tu là *mnuDientu1*, tiêu đề là *Bai dien tu &1*...

Bài tập 28.

Tiếp theo bài tập 27, tạo form **frmDientu1** với giao diện và các chức năng như sau:



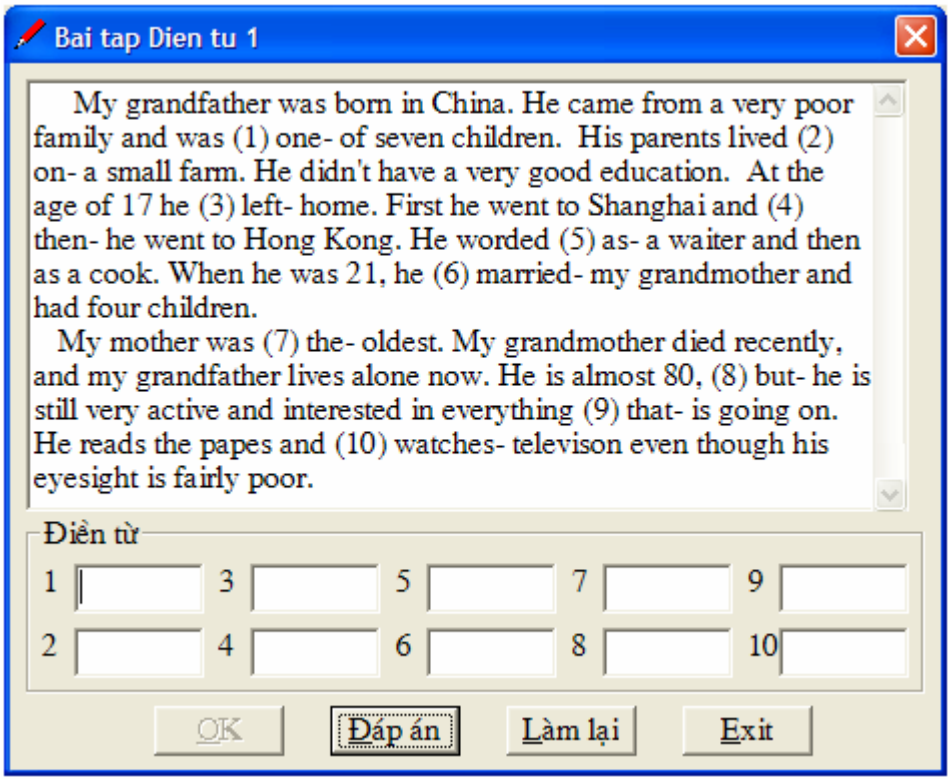
Hình 48. Giao diện bài tập 28

+ Khi gọi Form *frmDientu1* xuất hiện nội dung bài tập cần điền từ trong hộp Textbox.

+ Người sử dụng viết đáp án cho các câu vào các ô Textbox từ 1 đến 10.

+ Khi chọn ***OK*** chương trình kiểm tra kết quả, câu nào đúng thì đổi mầu nền ở Textbox tương ứng với câu trả lời sang mầu xanh, câu nào sai thì đổi Textbox có nền mầu hồng. Hiển thị điểm đạt được cho người dùng (mỗi câu trả lời đúng được 1 điểm).

+ Khi chọn nút ***Dapan*** thì form sẽ hiển thị đáp án như giao diện sau:



+ Bấm nút ***Làm lại,*** xuất hiện nội dung đề bài và xoá rỗng các đáp án cũ để người dùng có thể trả lời lại các câu hỏi.

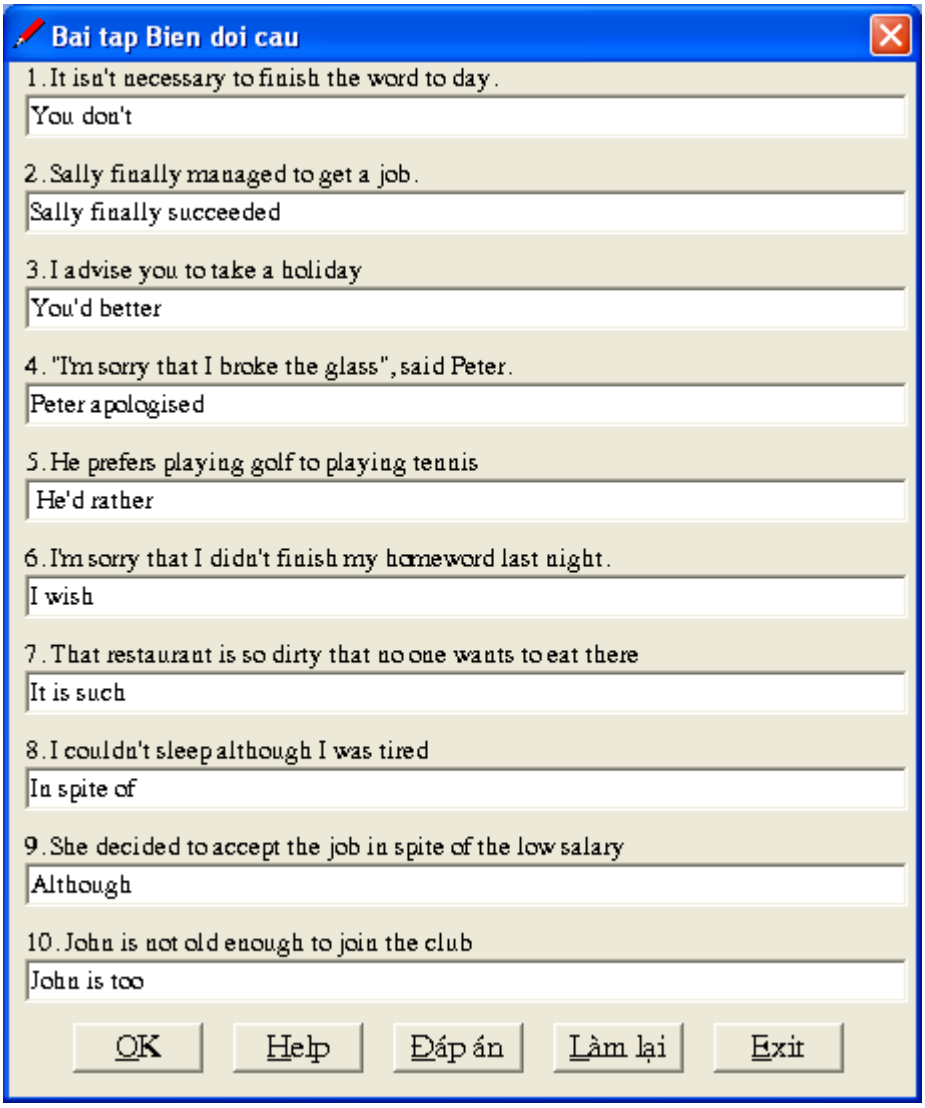
+ Bấm nút ***Exit*** để đóng form frm Dientu1 trở về form chính frmMain.

Bài tập 29.

Tiếp theo bài tập 28, tạo form **frmBiendoi1** có giao diện và các chức năng như sau:

+ Khi gọi Form *frmBiendoi1* xuất hiện nội dung các câu cần viết lại trong các nhãn Label với thứ tự từ 1 đến 10.

+ Người dùng viết câu mới có ý nghĩa tương tự với câu đã cho trong các nhãn Label vào các hộp Textbox tương ứng ở phía dưới bắt đầu bởi các từ hướng dẫn cho trước.



Hình 49. Giao diện bài tập 29

Các nút *OK, Đáp án, Làm lại* và *Exit* có chức năng tương tự như trong form frmDientu1.

Nút *Help* hiện hộp thông báo chứa các gợi ý cho các câu trả lời như sau: