[CHƢƠNG 1: TỔNG QUAN VỀ LẬP TRÌNH TRỰC QUAN VÀ .NET FRAMEWORK 6](#_bookmark0)

* 1. [Giới thiệu về Lập trình trực quan 6](#_bookmark1)
  2. [Tổng quan về .NET Framework 8](#_bookmark2)
     1. [Giới thiệu về .NET Framewwork 8](#_bookmark3)
     2. [Các thành phần chính của .NET Framewwork 8](#_bookmark4)
     3. [Các giai đoạn biên dịch và thi hành chƣơng trình 10](#_bookmark5)
     4. [Kiến trúc của .NET Application 11](#_bookmark6)
     5. [Các phiên bản của .NET Framework 12](#_bookmark7)
  3. [Làm quen với môi trƣờng lập trình Visual Studio 14](#_bookmark8)
     1. [Khởi động Visual studio 2010 15](#_bookmark9)
     2. [Giao diện môi trƣờng lập trình Visual C# trên WinForm 16](#_bookmark10)
  4. [Xây dựng và thực hiện chƣơng trình đầu tiên 21](#_bookmark11)
     1. [Đề bài 21](#_bookmark12)
     2. [Mở đồ án 22](#_bookmark13)
     3. [Thiết kế giao diện 23](#_bookmark14)
     4. [Viết mã lệnh 24](#_bookmark15)
     5. [Chạy chƣơng trình 26](#_bookmark16)
     6. [Dừng chƣơng trình 26](#_bookmark17)
     7. [Mở đồ án đã có 26](#_bookmark18)

[CHƢƠNG 2: DỮ LIỆU, CẤU TRÚC ĐIỀU KHIỂN VÀ GỠ RỐI 28](#_bookmark19)

* 1. [Đặc điểm cơ bản của C# 28](#_bookmark20)
  2. [Biến, hằng và các kiểu dữ liệu 29](#_bookmark21)

[3.2.1. Biến 29](#_bookmark22)

[2.2.2 Hằng 30](#_bookmark23)

* + 1. [Các kiểu dữ liệu 30](#_bookmark24)
    2. [Chuyển đổi giữa các kiểu dữ liệu 37](#_bookmark25)
  1. [Hộp thoại thông báo - Message Box 37](#_bookmark26)
     1. [Khái niệm 37](#_bookmark27)
     2. [Hộp thông báo MessageBox 38](#_bookmark28)
     3. [Hàm thông báo MessageBox 39](#_bookmark29)
  2. [Các câu lệnh điều khiển 40](#_bookmark30)
     1. [Câu lệnh lựa chọn if 40](#_bookmark31)
     2. [Câu lệnh lựa chọn switch … case 40](#_bookmark32)
     3. [Câu lệnh lặp for 41](#_bookmark33)
     4. [Câu lệnh lặp while 42](#_bookmark34)
     5. [Câu lệnh lặp do…while 42](#_bookmark35)
     6. [Câu lệnh lặp foreach 43](#_bookmark36)
     7. [Câu lệnh try...catch 43](#_bookmark37)
     8. [Câu lệnh break 43](#_bookmark38)
     9. [Câu lệnh Continue 43](#_bookmark39)
  3. [Phƣơng thức trong C# 43](#_bookmark40)
     1. [Định nghĩa phƣơng thức 43](#_bookmark41)
     2. [Gọi phƣơng thức 44](#_bookmark42)
  4. [Gỡ rối chƣơng trình 44](#_bookmark43)
     1. [Một số giải pháp gỡ rối chƣơng trình 44](#_bookmark44)
     2. [Dò lỗi từng dòng lệnh 45](#_bookmark45)

[CHƢƠNG 3: TÌM HIỂU CÁC ĐIỀU KHIỂN CƠ BẢN 49](#_bookmark46)

* 1. [Thuộc tính, phƣơng thức, sự kiện và các mối quan hệ giữa chúng 49](#_bookmark47)
  2. [Thuộc tính, phƣơng thức, sự kiện của một số điều khiển cơ bản 50](#_bookmark48)
     1. [Form 50](#_bookmark49)
     2. [Hộp văn bản – TextBox 51](#_bookmark50)
     3. [Nút lệnh – Button 53](#_bookmark51)
     4. [Nhãn – Label 54](#_bookmark52)
     5. [Dòng mách nƣớc – ToolTip 55](#_bookmark53)
     6. [Bài tập 55](#_bookmark54)
  3. [Một số điều khiển cơ bản khác 57](#_bookmark55)
     1. [Nhóm – GroupBox 57](#_bookmark56)
     2. [Hộp đánh dấu – CheckBox 58](#_bookmark57)
     3. [Nút tùy chọn – RadioButton 59](#_bookmark58)
     4. [Hộp danh sách – ListBox 60](#_bookmark59)
     5. [Hộp lựa chọn – ComboBox 65](#_bookmark60)
     6. [Điều khiển CheckedListBox 68](#_bookmark61)
     7. [Điều khiển NumericUpDown 73](#_bookmark62)
     8. [Thanh cuộn HscrollBar và VscrollBar 75](#_bookmark63)
     9. [Điều khiển Timer 76](#_bookmark64)
     10. [Điều khiển RichTextBox 78](#_bookmark65)

[CHƢƠNG 4: CÁC HỘP HỘI THOẠI THÔNG DỤNG 79](#_bookmark66)

* 1. [Hộp hội thoại Open File 79](#_bookmark67)
  2. [Hộp thoại SaveFile và luồng FileStream 81](#_bookmark68)
     1. [Hộp thoại SaveFile 81](#_bookmark69)
     2. [Luồng FileStream 81](#_bookmark70)
  3. [Hộp thoại Color 84](#_bookmark71)
  4. [Hộp thoại Font 86](#_bookmark72)

[CHƢƠNG 5: MENU VÀ CÁC ĐỒ ÁN NHIỀU BIỂU MẪU 89](#_bookmark73)

* 1. [Điều khiển MenuStrip 89](#_bookmark74)
  2. [Popup menu – ContextMenuStrip 91](#_bookmark75)
  3. [Đồ án nhiều biểu mẫu 93](#_bookmark76)
     1. [Bổ sung biểu mẫu 93](#_bookmark77)
     2. [Biểu mẫu khởi động 94](#_bookmark78)
     3. [Mở biểu mẫu 94](#_bookmark79)
     4. [Đóng biểu mẫu 95](#_bookmark80)
     5. [Xóa biểu mẫu 95](#_bookmark81)

[CHƢƠNG 6: LẬP TRÌNH CƠ SỞ DỮ LIỆU 99](#_bookmark82)

* 1. [Giới thiệu về bài toán 99](#_bookmark83)
     1. [Lập trình cơ sở dữ liệu và bài toán quản lý 99](#_bookmark84)
     2. [Cách tổ chức các tài nguyên trong một dự án của bài toán quản lý 100](#_bookmark85)
  2. [Cách tạo cơ sở dữ liệu (Database) trong môi trƣờng visual studio 2010 102](#_bookmark86)

[7.2.1. Tạo mới một DataBase 102](#_bookmark87)

* + 1. [Tạo các bảng CSDL 104](#_bookmark88)
    2. [Tạo quan hệ Relationship cho CSDL 107](#_bookmark89)
  1. [ADO.NET (ActiveX Data Objects for .NET Framework) 109](#_bookmark90)
     1. [Giới thiệu ADO.NET 109](#_bookmark91)
     2. [Kiến trúc của ADO.NET 111](#_bookmark92)
     3. [Các bƣớc làm việc với CSDL sử dụng ADO.NET 112](#_bookmark93)
     4. [Các đối tƣợng trong ADO.NET và một số phƣơng thức 112](#_bookmark94)
  2. [Làm việc với CSDL SQL qua các đối tƣợng: SqlConnection, SqlDataAdaper, SqlCommand. 122](#_bookmark95)
     1. [Đối tƣợng SqlConnection 122](#_bookmark96)
     2. [Đối tƣợng SqlDataAdapter và SqlCommand 123](#_bookmark97)
     3. [Các thao tác dữ liệu 124](#_bookmark98)
  3. [Làm việc với Dataset và DataTale 136](#_bookmark99)
     1. [Cấu trúc của Dataset và DataTable 136](#_bookmark100)
     2. [Các phƣơng thức 136](#_bookmark101)
     3. [Tự tạo DataTable 137](#_bookmark102)
  4. [Xuất dữ liệu ra Excel 138](#_bookmark103)
     1. [Làm việc với đối tƣợng Excel 138](#_bookmark104)
     2. [Ví dụ 141](#_bookmark105)
  5. [In báo cáo với CrystalReport 146](#_bookmark106)
     1. [Cài đặt Crystal report cho visual studio 2010 147](#_bookmark107)
     2. [Làm việc với CrystalReport thông qua DataSet 147](#_bookmark108)

[CHƢƠNG 7: PHÂN PHỐI VÀ CÀI ĐẶT ỨNG DỤNG 159](#_bookmark109)

* 1. [Đặt vấn đề 159](#_bookmark110)
  2. [Các cách đóng gói và triển khai ứng dụng 160](#_bookmark111)
  3. [Tạo dự án phân phối 161](#_bookmark112)
  4. [Tùy biến các lựa chọn đóng gói 164](#_bookmark113)
     1. [Cấu hình các thiết lập 165](#_bookmark114)
     2. [Tạo short cut cho ứng dụng cài đặt 166](#_bookmark115)
     3. [Thiết lập tên công ty và phiên bản chƣơng trình 167](#_bookmark116)

[7.4.1. Đặt các thuộc tính cho gói ứng dụng 168](#_bookmark117)

* 1. [Biên dịch và đóng gói dự án – kiểm tra việc cài đặt 169](#_bookmark118)
     1. [Chạy chƣơng trình cài đặt Setup 170](#_bookmark119)
     2. [Chạy chƣơng trình sau khi cài đặt 172](#_bookmark120)
  2. [Tìm hiểu các file setup và gỡ chƣơng trình 172](#_bookmark121)
     1. [Kiểm tra file cài đặt 172](#_bookmark122)
     2. [Tháo gỡ chƣơng trình 173](#_bookmark123)

# CHƯƠNG 1: TỔNG QUAN VỀ LẬP TRÌNH TRỰC QUAN VÀ

.NET FRAMEWORK

Trong chƣơng đầu tiên này, chúng ta sẽ làm quen với các đặc điểm của *Lập trình trực quan.* Sau đó chúng ta sẽ tìm hiểu tổng quan về công nghệ .NET Framework. Và cuối cùng ta sẽ thử viết và thực hiện một chƣơng trình đơn giản trên .NET Framework (sử dụng công cụ Visual Studio và ngôn ngữ C#).

## Giới thiệu về Lập trình trực quan

Lịch sử phát triển của Tin học luôn gắn liền với việc tìm kiếm các phƣơng pháp lập trình để giúp cho ngƣời sử dụng triển khai các ứng dụng một cách dễ dàng, nhanh chóng và hiệu quả.

Nhƣ chúng ta đã biết, mỗi loại máy tính (sử dụng loại CPU – Central Processing Unit xác định) chỉ có thể hiểu và thực hiện trực tiếp đƣợc các lệnh cũng nhƣ chƣơng trình theo một loại ngôn ngữ dành riêng đƣợc gọi là ngôn ngữ máy. Tuy nhiên, nếu triển khai các ứng dụng trong thực tế mà phải viết chƣơng trình trực tiếp bằng ngôn ngữ máy thì sẽ rất phức tạp, đòi hỏi thời gian và công sức rất lớn, nhiều khi không thể thực hiện đƣợc. Vì vậy, ngƣời ta tìm cách xây dựng một ngôn ngữ lập trình riêng gần với các ngôn ngữ tự nhiên, thuận lợi cho việc triển khai các ứng dụng. Khi thực hiện các chƣơng trình bằng ngôn ngữ này phải qua một bƣớc dịch chƣơng trình đó sang ngôn ngữ máy để có thể thực hiện. Từ trƣớc đến nay có rất nhiều ngôn ngữ lập trình đƣợc ra đời và phục vụ đắc lực cho việc triển khai các ứng dụng trên máy tính.

Trong giai đoạn đoạn đầu, các ngôn ngữ lập trình tuy dễ sử dụng hơn ngôn ngữ máy nhƣng rất khó với các lập trình viên vì chƣa đủ mạnh để dễ dàng triển khai các thuật toán. Chƣơng trình chƣa có tính cấu trúc chặt chẽ về mặt dữ liệu và tổ chức chƣơng trình. Vì vậy, việc triển khai các ứng dụng trong thực tế bằng các ngôn ngữ lập trình này là rất khó khăn.

Giai đoạn 2 là thời kỳ các ngôn ngữ lập trình có cấu trúc. Các ngôn ngữ lập trình này có đặc điểm là có tính cấu trúc chặt chẽ về mặt dữ liệu và tổ chức chƣơng trình. Một loạt các ngôn ngữ lập trình có cấu trúc ra đời và đƣợc sử dụng rộng rãi nhƣ: PASCAL, C, BASIC…

Giai đoạn 3 là thời kỳ của lập trình hƣớng đối tƣợng và phƣơng pháp lập trình có bƣớc biến đổi mạnh. Trong các ngôn ngữ lập trình có cấu trúc thì một ứng dụng bao gồm hai thành phần riêng là dữ liệu và chƣơng trình. Tuy chúng có quan hệ mật thiết nhƣng là hai đối tƣợng riêng biệt. Trong phƣơng pháp lập trình hƣớng đối tƣợng thì mỗi một đối tƣợng lập

trình sẽ bao hàm cả dữ liệu và phƣơng thức hành động trên dữ liệu đó. Vì vậy, việc lập trình sẽ đơn giản và mang tính kế thừa cao, tiết kiệm đƣợc thời gian lập trình.

Tuy nhiên, với các phƣơng pháp lập trình trên đều đòi hỏi lập trình viên phải nhớ rất nhiều câu lệnh với mỗi lệnh có một cú pháp và tác dụng riêng, khi viết chƣơng trình phải tự lắp ghép các lệnh để có một chƣơng trình giải quyết từng bài toán riêng biệt.

Trong xu hƣớng phát triển mạnh mẽ hiện nay của tin học, số ngƣời sử dụng máy tính tăng lên rất nhanh và máy tính đƣợc sử dụng hầu hết trong các lĩnh vực của đời sống nên đòi hỏi các ngôn ngữ lập trình cũng phải đơn giản, dễ sử dụng và mang tính đại chúng cao. Chính vì vậy phƣơng pháp lập trình trực quan ra đời. Đặc điểm của các ngôn ngữ lập trình trực quan là dễ sử dụng, triển khai các ứng dụng một cách nhanh chóng.

Đặc điểm nổi bật của phƣơng pháp lập trình trực quan là:

* + - Cho phép xây dựng chƣơng trình theo hƣớng sự kiện (Event – Driven Programming) nghĩa là một chƣơng trình ứng dụng đƣợc viết theo kiểu này đáp ứng dựa theo tình huống xảy ra lúc thực hiện chƣơng trình. Tình huống này bao gồm ngƣời sử dụng ấn 1 phím tƣơng ứng, chọn lựa một nút lệnh hoặc gọi một lệnh từ một ứng dụng khác chạy song song cùng lúc.
    - Ngƣời lập trình trực tiếp tạo ra các khung giao diện (interface), ứng dụng thông qua các thao tác trên màn hình dựa vào các đối tƣợng (ojbect) nhƣ hộp hội thoại hoặc nút điều khiển (control button), những đối tƣợng này mang các thuộc tính (properties) riêng biệt nhƣ: màu sắc, Font chữ…mà ta chỉ cần chọn lựa trên một danh sách cho sẵn.
    - Khi dùng các ngôn ngữ lập trình trực quan ta rất ít khi phải tự viết các lệnh, tổ chức chƣơng trình…một cách rắc rối, ta chỉ cần khai báo việc gì cần làm khi một tình huống xuất hiện.
    - Máy tính sẽ dựa vào phần thiết kế và khai báo của lập trình viên để tự động tạo lập chƣơng trình.

Nhƣ vậy với kỹ thuật lập trình trực quan, lập trình viên nhƣ một nhà thiết kế, tổ chức để tạo ra các biểu mẫu, đề nghị các công việc cần thực hiện và máy tính sẽ dựa vào đó để xây dựng chƣơng trình. Hiện nay có các ngôn ngữ lập trình, hệ quản trị cơ sở dữ liệu theo hƣớng trực quan thƣờng dùng nhƣ: Visual C#, Visual Basic, Visual Foxpro, Visual C, Delphi…

Với các đặc điểm trên của lập trình trực quan nhƣ trên. Ở môn học này chúng ta sẽ lựa chọn ngôn ngữ lập trình là Visual C# (trên nền .NET Framework) và môi trƣờng lập trình của chúng ta là bộ Visual Studio (phiên bản 2005, 2008, 2010 …).

## Tổng quan về .NET Framework

* + 1. ***Giới thiệu về .NET Framewwork***

.NET Framwork là môi trƣờng mà đoạn mã của bạn sẽ hoạt động. Đây có nghĩa là

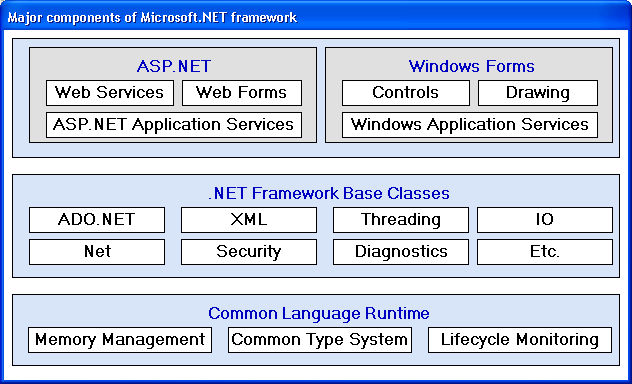
.NET sẽ quản lý việc thi hành chƣơng trình (bao gồm việc khởi động chƣơng trình, cấp phép hoạt động, cấp phát bộ nhớ, cho thu hồi bộ nhớ khi không dùng đến… Tuy nhiên, ngoài việc tiến hành những công tác kể trên, .NET còn chuẩn bị sẵn một thƣ viện lớp gọi là ***.NET base class*** (lớp cơ bản .NET) cho phép thực hiện vô số các tác vụ trên Windows.

Nói tóm lại, .NET giữ 2 vai trò: quản lý việc thi hành chƣơng trình của bạn và cung cấp các dịch vụ mà chƣơng trình của bạn cần đến.

* + 1. ***Các thành phần chính của .NET Framewwork***

.NET gồm có 2 thành phần: Framework và Intergrated Development Environment (IDE). Frame work cung cấp tất cả những gì cần thiết căn bản để chƣơng trình của bạn có thể thi hành đƣợc, còn IDE cung cấp một môi trƣờng giúp ta phát triển dễ dàng, nhanh chóng hơn. Nếu không có IDE ta vẫn có thể dùng notepad và command line để triển khai nhƣng chậm hơn. Trong .NET thì với bất kỳ một ngôn ngữ lập trình nào bạn dùng nhƣ C#, VB.NET… đều dùng cùng một IDE.

Hình vẽ sau đây, cho thấy các thành phần chính của .NET Framework



### Hình 1.1: Các thành phần chính của .NET Framework

#### Common Language Runtime (CLR)

CLR đƣợc xem nhƣ là linh hồn của kiến trúc .NET, nó là bộ phận lo quản lý việc thi hành mã của bạn: nạp chƣơng trình, cho chạy đoạn mã theo các mạch trình (thread) nhất định cũng nhƣ quản lý các mạch trình đó và cung cấp tất cả các dịch vụ hỗ trợ cho đoạn mã của bạn có thể thực thi. Tóm lại, CLR tạo ra môi trƣờng cho đoạn mã của bạn có thể thực thi.

Tất cả các đoạn mã mà đƣợc chạy trên .NET đều đƣợc gọi là đoạn mã đƣợc quản lý.

Các chƣơng trình .NET không đƣợc biên dịch thành tập tin khả thi mà đƣợc biên dịch thành một ngôn ngữ trung gian gọi là Microsoft Intermediate Language (MSIL) hay gọi tắt là Intermediate Language (IL).

Khi bạn biên dịch đoạn mã đƣợc quản lý thì trình biên dịch sẽ cho ra IL, rồi CLR sẽ dịch IL thành mã máy cụ thể.

#### .NET Framework base classes (các lớp cơ sở trong .NET Framework)

Tầng giữa của .NET Framework gồm những dịch vụ tổng quát thiết yếu của hệ thống.

Các dịch vụ này có thể dùng với tất cả các ngôn ngƣ lập trình. Chúng bao gồm:

+ ADO.NET: để truy cập dữ liệu.

+ I/O: Để thực hiện vào ra dữ liệu.

+ XML: Định dạng dữ liệu thông qua các thẻ định dạng.

+ …

ADO.NET là công nghệ truy cập cơ sở dữ liệu mới nhất của Microsoft. Mặc dầu ADO.NET đƣợc viết tắt từ cụm từ Active Data Objects for .NET Framework, nhƣng có lẽ nó đƣợc đặt sai tên vì ADO.NET không phải là một công nghệ Active/COM (Component Object Model). ADO.NET là một tập hợp các lớp hƣớng đối tƣợng, cung cấp một tập hợp phong phú các cấu kiện dữ liệu cho phép tạo các ứng dụng truy cập dữ liệu với hiệu năng cao, mức độ bảo mật tốt và có thể dễ dàng tăng quy mô theo thời gian trong một môi trƣờng Client-server hay môi trƣờng phân tán trên Internet hay Intranet. Trên mô hình ADO.NET, các ứng dụng sẽ kết nối với nguồn dữ liệu khi đọc hoặc nhật tu dữ liệu, sau đó thì cho đóng đƣờng dây kết nối lại. Điểm này rất quan trọng vì trong các ứng dụng Client – server hoặc phấn tán, việc để mở liên tục các đƣờng dây kết nối là một tiêu hao nguồn lực vô lối.

XML là một ngôn ngữ tổng quát định dạng dữ liệu thông qua các thẻ tự định nghĩa, nó là một chuẩn mới mà ngày càng đƣợc định dạng rộng rãi.

Ngoài ra còn có các lớp hỗ trợ lập trình đa luồng, xử lý vào ra dữ liệu (I/O), hỗ trợ bảo

mật.

#### ASP.NET và Windows Forms

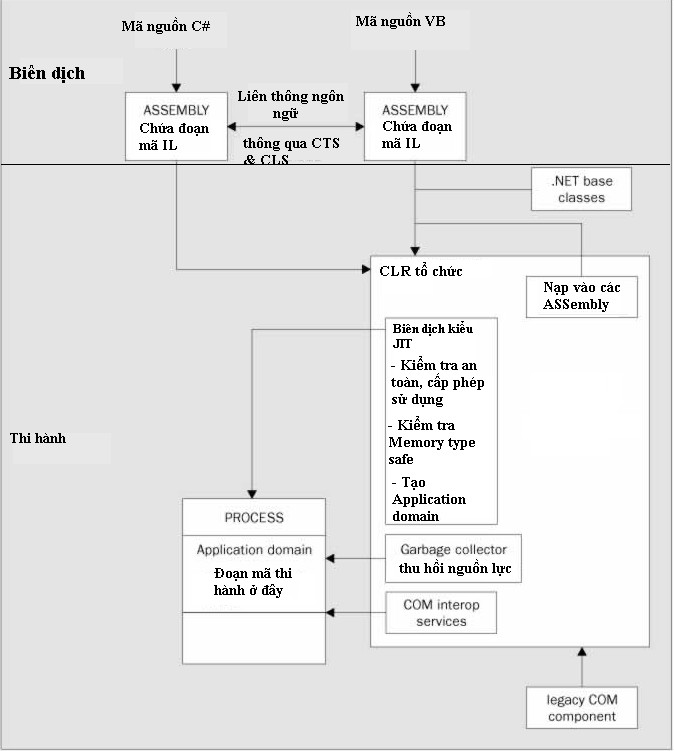
Tầng trên cùng nhất liên quan đến ngƣời sử dụng và giao diện chƣơng trình bao gồm

ASP.NET và Windows Forms.

Windows Forms (còn gọi tắt là Winform) là một cách mới và hay hơn để làm giao diện trong Win32. Winforms có nhiều điểm khác với VB6.

ASP.NET bao gồm Web forms và web services, các ứng dụng ASP.NET.

* + 1. ***Các giai đoạn biên dịch và thi hành chương trình***

Hình vẽ sau sẽ cho thấy quá trình này, các ô hình chữ nhật tƣợng trƣng cho các cấu kiện (Component) chính tham gia vào việc biên dịch và thi hành chƣơng trình, trong khi những mũi tên cho biết những công tác đƣợc thực hiện.

### Hình 1.2: Các giai đoạn biên dịch và thi hành chương trình

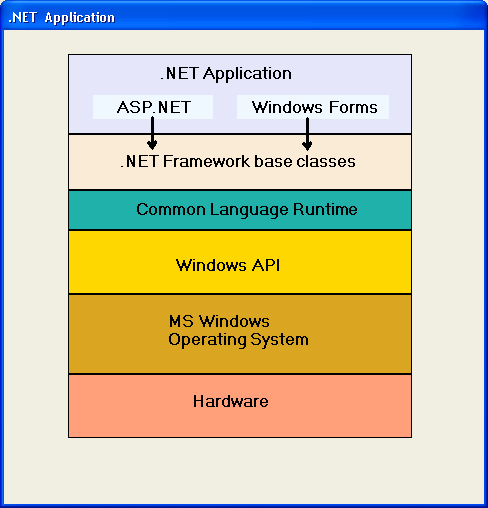
Phần trên đỉnh của hình cho thấy tiến trình biên dịch riêng rẽ mỗi dự án (prọject) thành một Assembly. Các assembly có khả năng tƣơng tác với nhau nhờ vào chức năng liên thông ngôn ngữ của .NET thông qua CTS (Common Type System) - đặc tả dữ liệu thông dụng, và CLS (Common Language Specification) – đặc tả ngôn ngữ thông dụng.

Phần dƣới cho thấy tiến trình biên dịch JIT (Just in time) từ IL trên các assembly thành đoạn mã hiện hành.

Just – in – time (JIT) – biên dịch vừa đúng lúc: Đây là tiến trình thực hiện giai đoạn biên dịch từ IL sang mã máy nguyên sinh. Trình biên dịch JIT chuẩn sẽ chạy theo yêu cầu. JIT compiler khá thông minh để có thể biết đƣợc đoạn mã nào đã đƣợc biên dịch, nên việc biên dịch chỉ xảy ra khi cần thiết. Do đó khi các ứng dụng .NET chạy thì chúng chạy ngày càng nhanh.

* + 1. ***Kiến trúc của .NET Application***

Kiến trúc chung của các ứng dụng đƣợc phát triển trong môi trƣờng .NET nhƣ sau:



### Hình 1.3: Kiến trúc của .NET Application

Các tầng dƣới cùng là phần cứng, hệ điều hành và Windows API (đây là phần mềm hệ thống cung cấp tất cả các chức năng và các tài nguyên mà các lập trình viên có thể rút ra từ đó để tạo nên các tính năng giao tiếp ngƣời – máy, nhƣ trình đơn kéo xuống, tên tệp, lệnh bàn phím).

Tiếp theo là trình diễn dịch ngôn ngữ thực thi chung (CLR). Nhờ nó mà một ứng dụng có thể thực thi đƣợc cho dù nó đƣợc viết trên nhiều ngôn ngữ khác nhau (phần này đã trình bày chi tiết ở phần trên).

Các lớp cơ sở trong .NET Framework cung cấp các hàm chức năng có sẵn, đi từ hiển thị những cửa sổ và biểu mẫu (form), triệu gọi các dịch vụ cơ bản của windows, đọc/viết các tập tin, thâm nhập vào mạng, Internet cũng nhƣ truy xuất các nguồn dữ liệu (trên căn cứ dữ liệu chẳng hạn).

Trên cùng là giao diện ứng dụng, có thể gồm 2 loại đó là web forms và windows forms.

* Windows forms giống nhƣ forms của VB6. Nó hỗ trợ Unicode hoàn toàn, rất tiện cho chứ Việt và thật sự hƣớng đối tƣợng.
* Web forms có những server control làm việc giống nhƣ các điều khiển trong Windows forms, nhất là có thể dùng mã lệnh để xử lý sự kiện y hệt nhƣ của Windows forms.
  + 1. ***Các phiên bản của .NET Framework***
* .NET framework 1.0 - 2002
* .NET framework 1.1 - 2003
* .NET framework 2.0 - 2005
* .NET framework 3.5 - 2008
* .NET framework 4.0 – 2010

#### Ver 1.0 – phát hành năm 2002

Ngày 12/2/2002 đánh dấu bƣớc quan trọng đầu tiên trong “cuộc đời” của .NET Framework, khi phiên bản 1.0 cùng với Visual Studio.NET 2002 đƣợc chính thức ra mắt. Chính .NET Framework 1.0 là điểm nhấn đáng chú ý nhất và làm cho Visual Studio. NET 2002 khác biệt hẳn với Visual Studio 6.0 đã phát hành năm 1998. Lần đầu tiên, Microsoft giới thiệu về “lập trình hợp nhất”, với việc lấy .NET Framework làm nền tảng.

#### Ver 1.1 - phát hành năm 2003

Một năm sau ngày .NET Framework 1.0 ra đời, ngày 24/4/2003, Microsoft đã có ngay bản cập nhật 1.1 ra mắt cùng với Visual Studio.NET 2003. Không có nhiều nâng cấp đáng chú ý trong lần ra mắt này, đáng kể nhất là sự ra đời của .NET Compact Framework, phiên bản thu gọn của .NET Framework cho các thiết bị di động. Điều đáng tiếc là mặc dù có nền tảng rất tốt, cùng với sự hỗ trợ mạnh mẽ từ Microsoft, cho đến nay, .NET Compact Framework vẫn chƣa phát triển nhƣ “lẽ ra nó phải thế”. Hiện nay số thiết bị di

động chạy Windows Mobile/Windows Phone khá khiêm tốn so với các hệ điều hành (HĐH) còn lại. .NET Framework 1.1 cũng mở ra một “truyền thống” là kể từ đây, các HĐH Windows đều đƣợc cài đặt sẵn phiên bản .NET Framework mới nhất. Windows Server 2003 tiên phong với phiên bản 1.1, sau đó là Windows Vista với .NET 3.0, và gần đây nhất là Windows 7/Server 2008 với .NET 3.5 SP1.

#### Ver 2.0 phát hành năm 2005

Microsoft mất đến hơn 2 năm để phát triển .NET Framework 2.0 và Visual Studio 2005, và thời gian bỏ ra là thật sự đáng giá. Tháng 11/2005, hai sản phẩm này ra mắt với hàng loạt tính năng mới, trong đó đáng kể nhất là việc hỗ trợ hoàn toàn cho tính toán 64-bit,

.NET Micro Framework, bổ sung và nâng cấp nhiều control của ASP.NET và đặc biệt là hỗ trợ Generics. .NET 2.0 hoàn toàn khác biệt so với các phiên bản trƣớc. Generic cho phép chúng ta định kiểu an toàn (type safety). Chúng cho phép ta tạo ra một cấu trúc dữ liệu mà không cần phải xác định đó là kiểu dữ liệu gì. Tuy nhiên khi cấu trúc dữ liệu này đƣợc sử dụng, trình biên dịch phải đảm bảo rằng kiểu dữ liệu đƣợc sử dụng với nó là kiểu an toàn. Generic cũng tƣơng đƣơng vơi Template trong C tuy nhiên việc sử dụng Generic trong

.NET dễ dàng hơn nhiều so với Template. Phiên bản 1.0 và 1.1 của .NET Framework không hỗ trợ generics. Thay vào đó, lập trình viên sử dụng lớp Object với các tham số và thành viên sẽ phải chuyển đổi tới các lớp khác dựa trên lớp Object. Generics mang đến hai tính năng cải tiến đáng kể đối với việc sử dụng lớp Object: Giảm bớt lỗi vận hành (Reduced run-time errors), Hiệu suất đƣợc cải thiện (Improved performance).

#### Ver 3.0 & Ver 3.5 (phát hành năm 2008)

Nếu nhƣ 3 phiên bản trƣớc đó, .NET Framwork đều gắn liền với một phiên bản Visual Studio nào đó, thì.NET Framework 3.0 đã “phá” truyền thống này khi ra mắt cùng với hệ điều hành Windows Vista vào cuối năm 2006. Ba “điểm nhấn” trong lần nâng cấp này là thành phần đƣợc kỳ vọng thay thế Winform - Windows Presentation Foundation – WPF, Windows Communitcation Foundation – WCF, Windows Workflow Foundation - WF, và Windows Card Space. .NET Framework 3.0 không phải là một phiên bản mới hoàn toàn, thực tế là một bản nâng cấp của .NET 2.0, hay đúng hơn là một bản nâng cấp cho thƣ viện của .NET 2.0. Chính vì không có Visual Studio “đi kèm”, mà .NET 3.0 đành phải “ký gửi” vào Visual Studio 2005 với một bộ công cụ mở rộng. Ngƣời dùng phải đợi đến tháng 11 năm 2007 mới đƣợc sử dụng một phiên bản Visual Studio hỗ trợ đầy đủ và toàn diện cho

.NET 3.0, và hơn thế nữa. Vâng, chúng ta đang nói đến VS 2008 và .NET Frame work 3.5. Cũng nhƣ phiên bản 3.0, .NET 3.5, là một mở rộng trên nền .NET 2.0. LINQ [LINQ:

Language Integrated Query - đây là thƣ viện mở rộng cho các ngôn ngữ lập trình C# và Visual Basic .NET (có thể mở rộng cho các ngôn ngữ khác) cung cấp khả năng truy vấn trực tiếp dữ liệu đối tƣợng, CSDL v à XML] là phần nổi bật và đáng chú ý nhất trong .NET 3.5.

#### Ver 4.0 – phát hành năm 2010

Ngày 12/4 vừa qua, Microsoft lại nâng cấp .NET Framework và Visual Studio. Đây là phiên bản đầu tiên sau .NET 2.0 kể từ 2005, có một CLR hoàn toàn mới: CLR 4.0. Cũng cần nhắc lại là cả .NET 3.0 và 3.5 đều sử dụng CLR 2.0, và không có CLR 3.0. Việc Microsoft chuyển thẳng lên 4.0 không chỉ để “đồng bộ” phiên bản, mà còn nhằm khẳng định đây là một bƣớc tiến lớn. .NET Framework 4 giới thiệu một model an ninh đƣợc cải thiện. Các tính năng mới và cải tiến trong .NET Framework 4 là:

+ Application Compatibility and Deployment (Khả năng tƣơng thích ứng dụng và triển

khai)

+ Core New Features and Improvements (Các tính năng mới và cải tiến của phần nhân)

+ Managed Extensibility Framework (Quản lý mở rộng Framework)

+ Parallel Computing (Điện toán song song)

+ Networking

+ Web

+ Client

+ Data

+ Windows Communication Foundation (WCF)

+ Windows Workflow Foundation (WF)

## Làm quen với môi trường lập trình Visual Studio

Để có thể phát triển các ứng dụng .NET một cách nhanh chóng ta có thể cài đặt bộ công cụ Visual Studio với một trong các phiên bản sau:

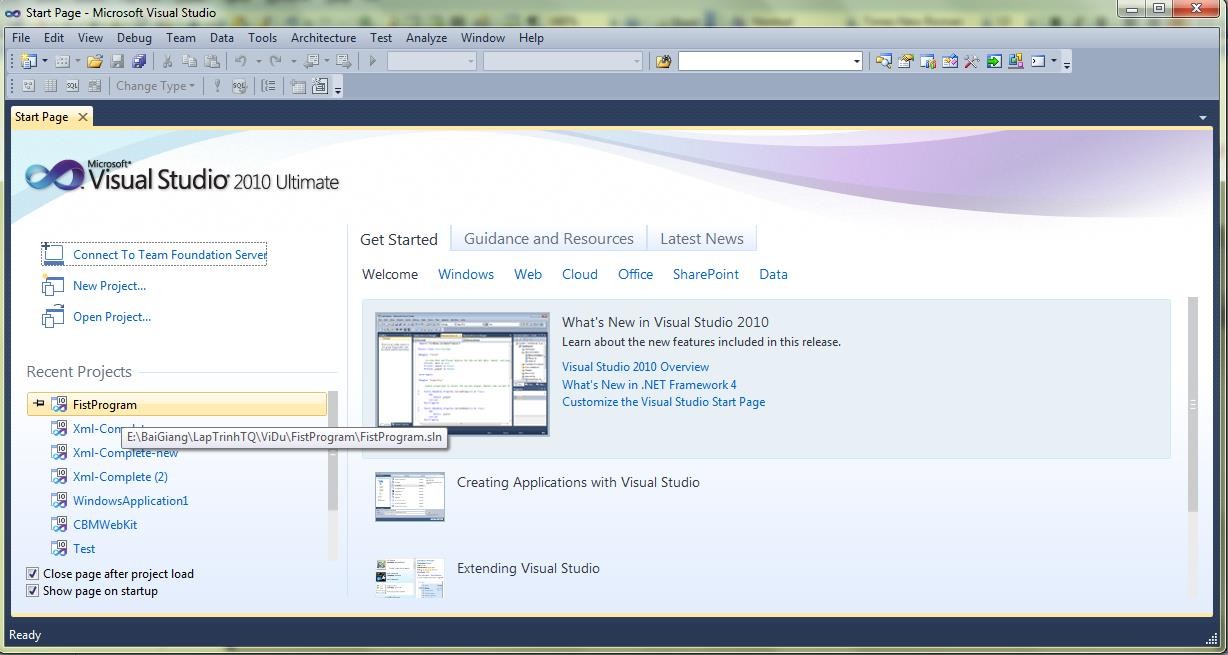
+ Visual studio 2005 – hỗ trợ đến .NET Framework 2.0

+ Visual studio 2008 – hỗ trợ đến .NET Framework 3.0 và 3.5

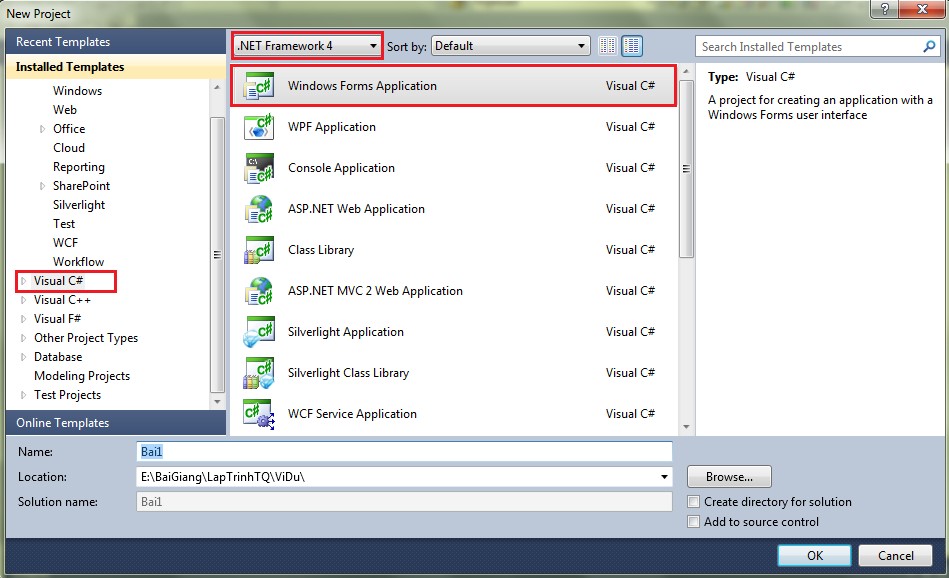
+ Visual studio 2010 – hỗ trợ đến .NET Framework 4.0

Ở tài liệu này tôi sẽ hƣớng dẫn các bạn làm việc trên Visual Studio 2010

* + 1. ***Khởi động Visual studio 2010***

Vào **Start\All Programs\Microsoft Visual Studio 2010\Microsoft Visual Studio 2010**, xuất hiện cửa sổ **Start Page** cửa sổ có hiển thị danh sách các đồ án đƣợc mở gần đây nhất nhƣ sau:

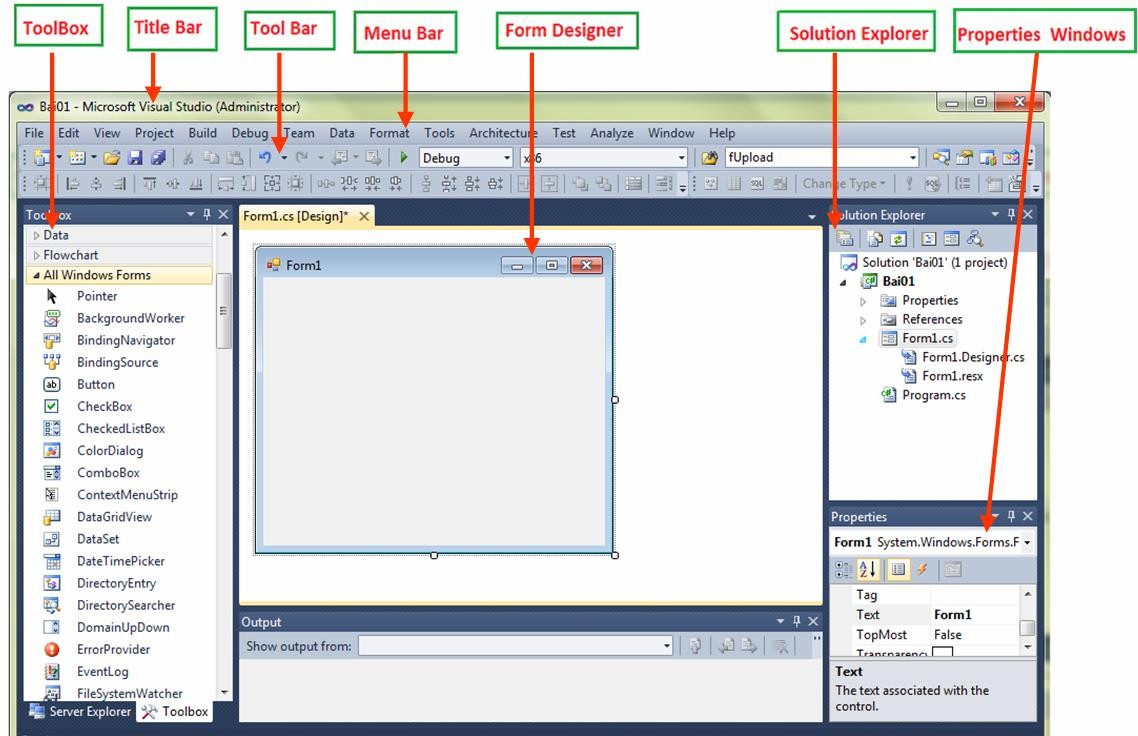
Để mở một đồ án mới ta vào **File/New/Project…** hoặc nhấn Ctrl + Shift + N, xuất hiện cửa sổ New Project nhƣ sau:



Ta chọn ngôn ngữ lập trình là Visual C#, chọn loại ứng dụng là Windows Forms Application, chọn phiên bản mới nhất là .NET Framework 4.0. Tại ô Name ta viết tên của đồ án, tại ô Location ta chọn thƣ mục chứa đồ án bằng cách nhấn nút Browse...

Cuối cùng ta nhấn OK.

* + 1. ***Giao diện môi trường lập trình Visual C# trên WinForm***

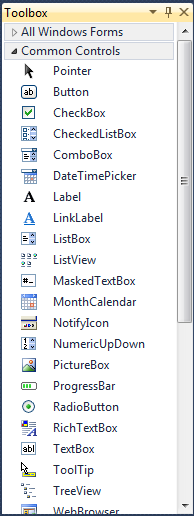
Sau khi khởi động Visual Studio 2010 theo các bƣớc nhƣ trên xuất hiện cửa sổ môi trƣờng phát triển IDE, có giao diện gồm các thành phần nhƣ sau:

* + - **ToolBox:** là hộ công cụ chứa các điều khiển – controls đƣợc đặt lên form khi thiết kế giao diện ngƣời dùng. Để hiển thị hộp công cụ ta thực hiện một trong các cách sau:

+ Vào View/Toolbox

+ Bấm tổ hợp phím Ctrl + Alt + X

+ Bấm vào biểu tƣợng Toolbox  trên thanh công cụ standard



* + - **Title Bar:** Thanh tiêu đề chứa tên đồ án
    - **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 lƣu trữ đồ án...

+ Edit: gồm các thao tác hỗ trợ soạn thảo mã lệnh nhƣ: Copy, cắt, dán, tìm kiếm...

+ View: cho phép hiển thị các công cụ hỗ trợ ngƣời dùng trong quá trình xây dựng đồ án nhƣ: cửa sổ mã lệnh – code, form thiết kế - designer, hộp công cụ - toolbox, thanh công cụ - toolbar, cửa sổ thuộc tính – Properties...

+ Project: cho phép bổ sung các đối tƣợng khác nhau vào đồ án nhƣ: Form, các Component, Các class...

+ Built: cho phép biên dịch đồ án

+ Debug: cho phép chạy và gỡ rối chƣơng trình

+ Data: Cho phép thêm mới và hiển thị cơ sỏ dữ liệu của đồ án.

+ Format: Cho phép căn lề, định dạng kích thƣớc, chế độ hiển thị... của các điều kiển đƣợc đặt trên Form.

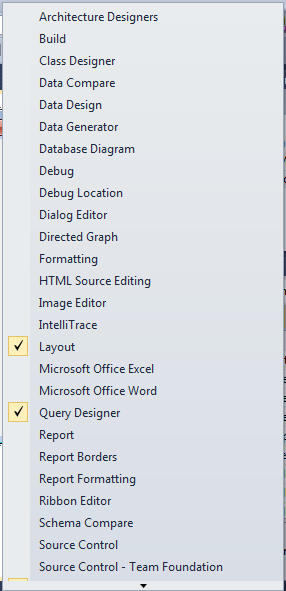
+ Tools: Cung cấp các công cụ cho phép kết nối tới các thiết bị ngoại vi nhƣ Pocket PC, SmartPhone... hoặc kết nối tới các hệ quản trị CSDL cũng nhƣ kết nối tới máy chủ Server...

+ Window: kiểm soát cách bố trí cửa sổ

+ Help: Truy cập hệ trợ giúp trực tuyến MSDN

* + - **Tool Bar:** Thanh công cụ gồm một tập hợp các nút lệnh, mỗi nút lệnh chứa các biểu tƣợng icons và các chức năng tƣơng ứng với chức năng của một mục lựa chọn trong thanh Menu. Thanh công cụ rất hữu ích và trực quan, giúp ngƣời dùng dẽ dàng và nhanh chóng thực hiện một số chức năng mong muốn chỉ thông qua cái Click chuột.

Để gọi các thanh công cụ ra ta vào View/Toolbars hoặc kích chuột phait tại thanh Menu, Khi đó sẽ xuất hiện danh sách tất cả các thanh công cụ. Muốn ẩn/hiện thanh nào ta kích chuột tại dòng chứa tên thanh công cụ đó.



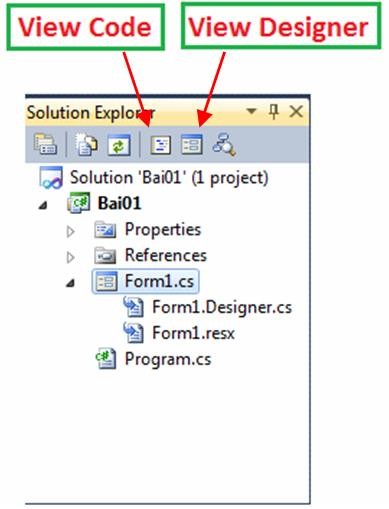
* + - **Form Designer:** Cửa sổ thiết kế dùng để thiết kế giao diện cho chƣơng trình, mỗi dự án có thể có một hoặc nhiều Form.
    - **Solution Explorer:** Cửa sổ giải pháp – đây là phần cửa sổ giúp ta quả lý tất cả các tài nguyên và tệp tin của dự án. Solution Explorer đƣợc tổ chức thành một cấu trúc cây bao gồm các mục khác nhau nhƣ: danh sách các Form, danh sách các hình ảnh, danh sách các class, …

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

+ Vào View\Solution Explorer

+ Bấm tổ hợp phím Ctrl+Alt+L hoặc Ctrl+R

+ Kích chuột vào biểu tƣợng Solution Explorer  trên thanh công cụ Standard.



Trong cửa sổ Solution Explorer có 2 thành phần quan trọng chúng ta rất hay dùng là

*View Code* và *View Designer.*

**View Code:** Có tác dụng hiển thị cửa sổ soạn thảo mã lệnh cho Form đang đƣợc chọn. Ngoài ra, để hiển thị cửa sổ soạn thảo mã lệnh ta có thể làm một trong các cách sau:

+ Vào View/Code

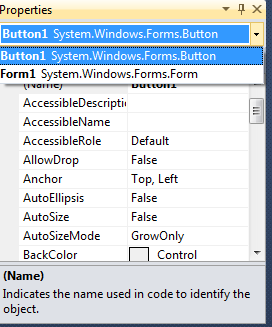
+ Bấm phím F7

+ Kích đúp chuột tại cửa sổ thiết kế của Form.

**View Designer:** có tác dụng hiển thị cửa sổ thiết kế giao diện của Form đang đƣợc chọn, để hiển thị cửa sổ thiết kế giao diện ta còn có một số cách khác nhƣ:

+ Vào View\Designer

+ Bấm phím Shift + F7

* + - **Properties Window:** Cửa sổ này liệt kê tất cả các thuộc tính của forn và các điều khiển trong đồ án. Muốn hiển thị thuộc tính của đối tƣợng nào ta kích chuột chọn đối tƣợng ấy trong cửa sổ thiết kế gao diện, hoặc chọn tên đối tƣợng trong danh sách thả xuống ở phần đầu của cửa sổ Properties.

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

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

+ Vào View\Properties Window

+ Kích chuột vào biểu tƣợng Properties Window  trên thanh công cụ Standard, hoặc biểu tƣợng Properties  trong cửa sổ giải pháp Solution Explorer

+ Bấm phím F4

+ Bấm tổ hợp phím Alt + Enter

## Xây dựng và thực hiện chương trình đầu tiên

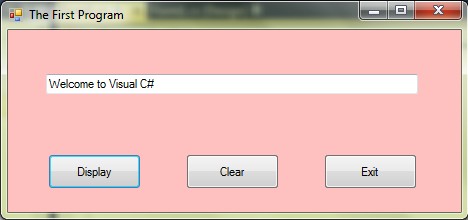
* + 1. ***Đề bài***

Viết chƣơng trình gồm một 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#*”

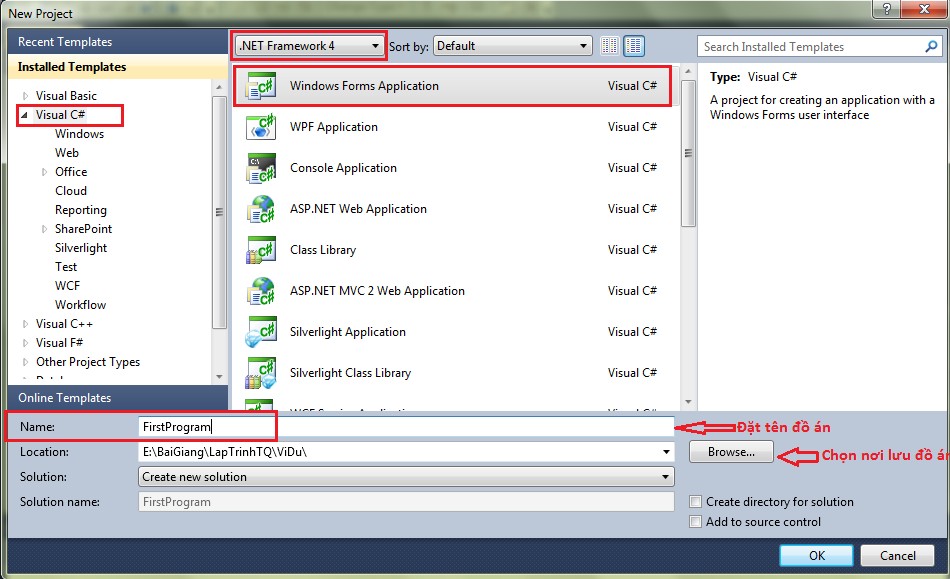
+ Kích chuột vào nút *Clear* chữ ở hộp văn bản biến mất.

+ Kích chuột vào nút *Exit* thì thoát khỏi chƣơng trình quay lại cửa sổ soạn thảo.



* + 1. ***Mở đồ án***

Mở **Visual Studio 2010,** chọn **File/New/Project…** để khởi động một dự án mới. Xuất hiện hộp thoại **New Project** nhƣ sau:



Chọn ngôn ngữ là **Visual C#,** ứng dụng là **Windows Forms Application,** phiên bản

**.NET Framework 4.0**. Đặt tên đồ án tại mục **Name** là **FistProgram**, chọn nơi lƣu đồ án bằng nút **Browse,** Sau đó nhấn **OK.**

* + 1. ***Thiết kế giao diện***

### Đặt tên và tiêu đề cho form

Kích chuột vào vị trí bất kỳ trong Form, trong cửa sổ Properties sửa thuộc tính Name thành *frmWelcome*, sửa thuộc tính Text là *The First Program.*

### Thêm điều khiển hộp văn bản TextBox

Kích chuột vào biểu tƣợng  trên hộp công cụ Toolbox, bấm phím trái cùng với giữ và kéo chuột để đặt hộp văn bản trên Form.

Khi mới xuất hiện hộp văn bản trên Form thì hộp văn bản tên mặc định là TextBox1, Kích chọn hộp văn bản trong cửa sổ Properties sửa thuộc tính *Name* sửa thành txtWelcome.

### Thêm điều khiển nút lệnh Button

Kích chuột vào biểu tƣợng  trên hộp công cụ ToolBox, bấm phím trái cùng với giữ và kéo chuột để đặt nút lệnh trên Form. Nút lệnh này có tên mặc định là Button1 và nội dung cũng là Button1.

Thực hiện tƣơng tự ta đƣa thêm 2 nút lệnh Button2 và Button3. Sau đó ta tiến hành thay đổi 2 thuộc tính Name và Text của các nút lệnh nhƣ sau:

|  |  |  |
| --- | --- | --- |
| **Nút lệnh** | **Name** | **Text** |
| Button1 | btnDisplay | Display |
| Button2 | btnClear | Clear |
| Button3 | btnExit | Exit |

**Chú ý:** Mọi điều khiển đều có thuộc tính Name, để dễ quản lý, gỡ rối và viết chƣơng trình ta nên đặt tên điều khiển tƣơng 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 viết thƣờng, tên điều khiển viết hoa chữ cái đầu tiên của mỗi từ, ví dụ: **txtWelcome, btnDisplay…**

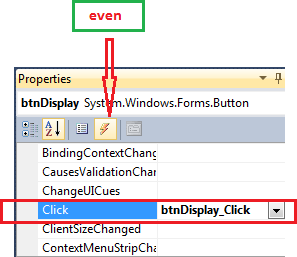
* + 1. ***Viết mã lệnh***

### Viết mã lệnh cho nút Display

Ta mở cửa sổ soạn thảo code bằng một trong hai cách sau:

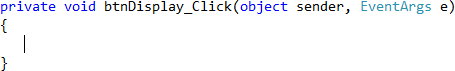
+ Cách 1: Kích đúp chuột vào nút Display

+ Cách 2: Kích chọn đối tƣợng (nút nhấn) btnDisplay. Trong cửa sổ **Properties** chọn biểu tƣợng even, sau đó kích đúp vào sự kiện Click nhƣ hình vẽ sau:

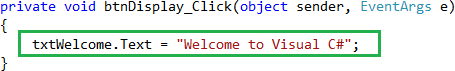


Trong cửa sổ code xuất hiện một hàm để xử lý sự kiện khi ngƣời dùng nhấn vào nút

**Display** nhƣ sau:



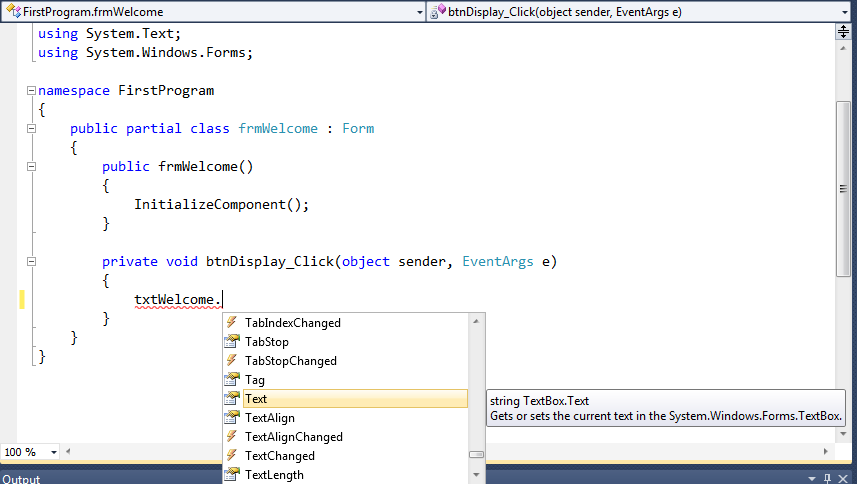
Tiếp theo, ta gõ vào giữa hàm btnDisplay\_Click … dòng lệnh gán giá trị *“Welcome to Visual C#”.* Ta cho thuộc tính Text của điều khiển txtWelcome nhƣ sau:



***Căn dặn:***

1. Hầu nhƣ tất cả các hàm xử lý một sự kiện nào đó của các điều khiển trên Form đều có 2 đối số. Đối số *Sender* có kiểu *Object* đại diện cho đối tƣợng đã phát sinh sự kiện, đối số *e* có kiểu *EventArgs* chứa các thông tin về sự kiện nhƣ: vị trí chuột, thời gian phát sinh sự kiện, thời điểm phát sinh sự kiện, … Từ khó *Handles* nhằm xác định rõ thủ tục đang xử lý dùng cho sự kiện nào của đối tƣợng.
2. Ta có cấu trúc chung để gán giá trị cho thuộc tính của một điều khiển khi viết mã lệnh nhƣ sau:

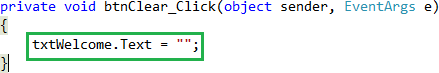
#### <Tên điều khiển>.<Thuộc tính>=<Giá trị>

Các thuộc tính của các điều khiển trong **Visual C#** rất phong phú, trong Visual studio cung cấp tiện ích ***Intellisence*** tự động hiển thị danh sách các thuộc tính của điều khiển sau khi gõ tên điều khiển và dâu chấm „.‟, ví dụ nhƣ sau:

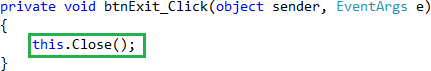
Để lựa chọn thuộc tính ta dùng phím mũi tên lên xuống, hoặc gõ chữ cái đầu của thuộc tính. Sau đó ta dùng phím Enter để chèn tự động tên thuộc tính vào dòng lệnh.

1. Trong môi trƣờng soạn thảo, nếu bạn gõ sai cú pháp thì môi trƣờng lập trình sẽ bắt lỗi ngay bằng cách hiển thị đƣờng gạch chân hình răng cƣa ngay dƣới chỗ sai, khi sửa xong lập tức đƣờng gạch hình răng cƣa bị mất.

### Viết mã lệnh nút Clear



* + - **Viết mã lệnh nút Exit**



Lệnh *this.Close()* có tác dụng đóng Form hiện hành lại và xóa tất cả các đối tƣợng của Form ra khỏi bộ nhớ.

* + 1. ***Chạy chương trình***

Để chạy chƣơng, ta thực hiện một trong các cách sau:

+ C1: Chọn Debug/Start Degugging

+ C2: Kích chuột trái vào biểu tƣợng Start Debugging  trên thanh công cụ Standard

+ C3: Nhấn phím F5

* + 1. ***Dừng chương trình***

Khi chạy chƣơng trình có thể xảy ra lỗi, khi xảy ra lỗi thì bạn cần dừng chƣơng trình để quay lại môi trƣờng soạn thảo code để sửa bằng một trong các cách sau:

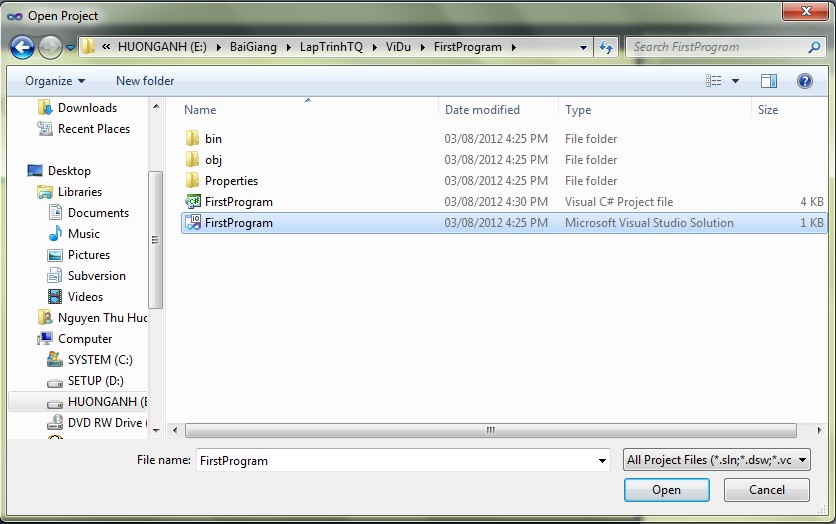
+ C1: Kích chuột vào biểu tƣợng Stop  trên thanh công cụ.

+ C2: Vào Debug\Stop Debugging

+ C3: Nhấn tổ hợp phím Shift + F5

* + 1. ***Mở đồ án đã có***

Vào File/Open/Project…. Xuất hiện hộp thoại Open Project, ta chọn đƣờng dẫn đến thƣ mục chứa dự án. Ta chọn file có tên là tên dự án có kiểu là **Microsoft Visual Studio Solution** hoặc **Visual C# Project File**, sau đó nhấn OK.



# CHƯƠNG 2: DỮ LIỆU, CẤU TRÚC ĐIỀU KHIỂN VÀ GỠ RỐI

Trong chƣơng này, chúng ta sẽ tìm hiểu các đặc điểm cơ bản trong cú pháp của C#.

Các nội dung đƣợc đề cập đến ở chƣơng này bao gồm:

+ Đặc điểm của C# (trong phần này ta sẽ so sánh với C++)

+ Biến, hằng và các kiểu dữ liệu trong C#

+ Hộp thoại thông báo – Message Box và cách dùng nó.

+ Các câu lệnh điều khiển

+ Cách định nghĩa một phƣơng thức trong C#

+ Các giải pháp gỡ rối chƣơng trình

## Đặc điểm cơ bản của C#

Để có thể hiểu về C# một cách sơ lƣợc và nhanh chóng ta có thể so sánh C# và C++ nhƣ sau:

### Giống nhau:

* + - * Đều phân biệt chữ hoa, chữ thƣờng.
      * Mỗi lệnh đƣợc kết thúc bởi dấu chấm phẩy (;)
      * Hỗ trợ 2 kiểu chú thích giống nhau.
      * Đều hỗ trợ lập trình hƣớng đối tƣợng.
      * Các lệnh điều khiển: if, for, while, switch, … có cú pháp giống nhau.

### Khác nhau:

|  |  |  |
| --- | --- | --- |
| **Đặc điểm** | **C++** | **C#** |
| Hƣớng đối tƣợng | - Lào ngôn ngữ lai giữa lập trình hƣớng đối tƣợng và lập trình hƣớng chức năng (nghĩa là nó cho phép tạo cả các chƣơng trình hƣớng chức năng nhƣ trong C) | - Không cho phép tạo chƣơng trình hƣớng chức năng (tất cả phải là hƣớng đối tƣợng) |
| Tệp tin thƣ viện | * Sử dụng từ khóa *“include”* * Khái niệm này gọi là *“header”* | * Sử dụng từ khóa (using) * Khái niệm này gọi là   *“namespace”* |

Trên đây là một số so sánh rất khái quát giữa 2 ngôn ngữ C++ và C#. Tuy nhiên đi vào từng chi tiết thì nó sẽ có sự khác biệt. Phần dƣới đây ta sẽ tìm hiểu chi tiết về dữ liệu, cấu trúc điều khiển trong C#.

## Biến, hằng và các kiểu dữ liệu

***3.2.1. Biến***

Là một giá trị dùng để chứa dữ liệu tạm thời trong quá trình tính toán. Ta có thể khai báo biến theo cú pháp sau:

### [Phạm vi truy xuất] <Kiểu biến> <tên biến> [=<giá trị>]

**+ Tên biến:** Có thể dài 256 ký tự bao gồm các chữ cái, chữ số và dấu gạch dƣới. Tên biến phải bắt đầu bằng một chữ cái. C# phân biệt chữ hoa chữ thƣờng.

### Phạm vi truy xuất gồm có:

***+ public:*** Những biến khai báo public có thể đƣợc dùng ở bất kỳ chỗ nào trong lớp đó và cả các lớp khác trong dự án.

***+ private:*** Những biến khai báo private thì chỉ đƣợc truy cập ở trong lớp đó.

***+ protected:*** Những biến khai báo protected chỉ đƣợc truy cập bên trong lớp chứa nó và những lớp dẫn xuất từ lớp đó.

***2.2.2 Hằng***

Dùng để chứa những dữ liệu có giá trị không đổi trong suốt quá trình tính toán. Sử dụng hằng số làm chƣơng trình sáng sủa, dễ đọc nhờ tên gọi gợi nhớ thay vì các con số. Hằng đƣợc khai báo theo cú pháp sau:

#### <const> <kiểu dữ liệu> <tên hằng> = <giá trị>;

Ví dụ: const double Pi=3.14 ;

Phạm vi hoạt động: Hằng đƣợc khai báo với từ khóa private trong phƣơng thức chỉ hoạt động trong phƣơng thức, khai báo cho biểu mẫu chỉ hoạt động trong biểu mẫu đó. Hằng đƣợc khai báo với từ khóa public hoạt động trên toàn ứng dụng.

* + 1. ***Các kiểu dữ liệu***

### Kiểu số

- Kiểu số nguyên gồm có:

|  |  |  |
| --- | --- | --- |
| **Kiểu dữ liệu** | **Số byte** | **Mô tả** |
| byte | 1 | Số nguyên dƣơng không dấu từ 0 – 255 |
| sbyte | 1 | Số nguyên có dấu (từ -128 đến 127) |
| short | 2 | Số nguyên có dấu giá trị từ -32768 đến 32767 |
| ushort | 2 | Số nguyên không dấu từ 0 – 65535 |
| Int | 4 | Số nguyên có dấu từ -2.147.483.647 đến 2.147.483.647 |
| uint | 4 | Số nguyên không dấu 0 – 4.294.967.295 |
| long | 8 | Kiểu số nguyên có dấu có giá trị trong khoảng  -9.223.370.036.854.775.808 đến 9.223.372.036.854.775.807 |
| ulong | 8 | Số nguyên không dấu từ 0 đến 0xffffffffffffffff |

- Kiểu số thực gồm có:

|  |  |  |
| --- | --- | --- |
| **Kiểu dữ liệu** | **Số byte** | **Mô tả** |
| float | 4 | Kiểu dấu chấm động có giá trị xấp xỉ từ 3,4E-38 đến 3,4E+38 với 7 chữ số có nghĩa |
| double | 8 | Kiểu dấu chấm động có độ chính xác gấp đôi, giá trị xấp xỉ từ 1,7E-308 đến 1,7E+308 với 15, 16 chữ số có nghĩa |
| decimal | 8 | Có độ chính xác đến 28 con số và giá trị thập phân. |

### Các phép toán số học

+ Cộng (+), trừ (-), nhân (\*), chia (/). Các phép toán x[+,-,\*,/]=y tƣơng đƣơng với phép toán x=x[+,-,\*,/]y.

+ Phép chia lấy phần dƣ (%). Ví dụ: 5%3=2

+ Math.Round(x,n): hàm làm tròn số thực x đến n chữ số sau dấu phẩy thập phân, 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 của ex

+ Math.Pow(x,y): hàm tính giá trị xy

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

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

+ Sử dụng lớp Random để sinh số ngẫu nhiên:

* B1: Khởi tạo đối tƣợng Random. Random rd=new Random();
* B2: Dùng các hàm với ý nghĩa nhƣ sau:
  + rd.Next(): Sinh ngẫu nhiên một số kiểu int
  + rd.Next(int MinValue,int MaxValue): Sinh ngẫu nhiên một số kiểu int có giá trị từ MinValue đến MaxValue.
  + rd.Next(int MaxValue): Sinh ngẫu nhiên một số kiểu int có giá trị từ lớn nhất là MaxValue
  + rd.NextDouble(): sinh ngẫu nhiên một số kiểu double có giá trị từ 0 đến 1.

### Các phép toán so sánh

+ Lớn hơn (>), lớn hơn hoặc bằng (>=)

+ Nhỏ hơn (<), nhỏ hơn hoặc bằng (<=)

+ Bằng(= =), khác (!=).

### Kiểu chuỗi

Đối tƣợng kiểu string thƣờng chứa một chuỗi ký tự. Để khai báo chuỗi ta dùng cú pháp nhƣ sau: string myString;

Sau đó ta có thể gán giá trị của chuỗi: myString=”Chào bạn”;

Hoặc ta có thể vừa khai báo vừa khởi tạo nhƣ sau: string myString=”Chào bạn”;

Trong C# có hỗ trợ font Unicode nên ta có thể gõ tiếng việt có dấu.

### Các phép toán trên kiểu chuỗi

+ dấu + : 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: “Đạt Anh”.Length có kết quả =7.

+ s.Replace(str1,str2): Thay thế chuỗi str1 trong chuỗi s bằng chuỗi str2. Ví dụ: "Nguyễn Thu Hƣờng".Replace("Thu", "Anh") cho kết quả chuỗi “Nguyễn Anh Hƣờng”.

+ 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. 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 thƣờng. 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”. (Chuỗi đƣợc tính từ vị trí 0)

+ *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(„c‟): Trả về mảng chuỗi đƣợc phân định bởi ký tự c.

+ *Convert.ToInt16(ch)*: trả về mã ASCII của ký tự ch. Ví dụ: Convert.ToInt16(„A‟)=65

+ *Convert.ToChar(n):* trả về ký tự có mã ASCII = n. Ví dụ: Convert.ToChar(65)=”A”

### Kiểu đúng sai

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 trƣờng hợp còn lại.

*Phép toán ho*ặ*c – ||*: 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 trƣờng hợp còn lại.

*Phép toán ph*ủ đị*nh – !*: xét biểu thức !A sẽ nhận giá trị đúng khi A nhận giá trị sai và ngƣợc 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 |

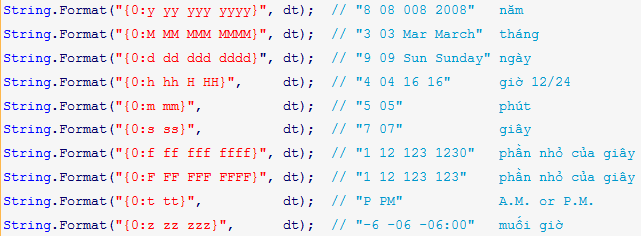
### Kiểu ngày tháng

Kiểu ngày tháng trong C# đƣợc dùng qua lớp DateTime. Để tạo thời gian ta làm nhƣ sau: DateTime dt=new DateTime(int year,int month, int day, int hour, int munite, int second)

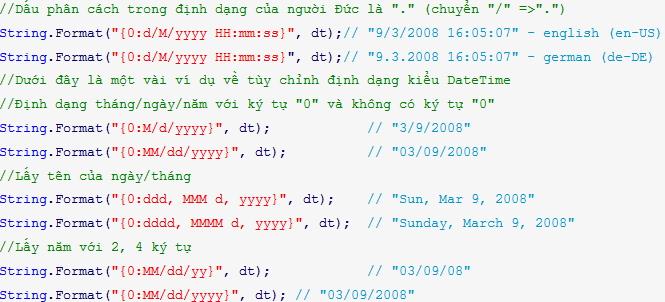
Ví dụ: Khởi tạo thời gian DateTime ngày 23/08/2012 8:30:20 nhƣ sau

DateTime dt=new DateTime(2012,8, 23, 8, 30, 20)

### Các định dạng kiểu ngày tháng

Có các định dạng tùy chỉnh sau đây: y (năm), M (tháng), d (ngày), h (giờ 12), H (giờ 24), m (phút), s (giây), f (phần nhỏ của giây), F (phần nhỏ của giây, không bao gồm số 0 sau cùng), t (PM hoặc AM) và z (múi giờ).

Chèn vào dấu phân cách “/” hoặc “:”, các ký tự này sẽ đƣợc chèn vào kết quả đầu ra để xác định thời gian:



### Các phép toán trên kiểu ngày tháng

*DateTime.Now:* trả về ngày và giờ hiện hành, ví dụ: 15/08/2009 5:20:28PM

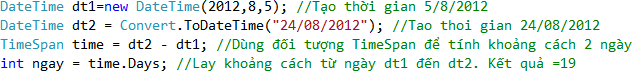
year.

*DateTime.DaysInMonth(int year,int month)*: trả về số ngày trong tháng month của năm

*dt.Month*: trả về tháng của DateTime dt *dt.Year*: trả về năm của DateTime dt *dt.Day*: trả về ngày của DateTime dt

*dt.AddDays(n)*: trả về một ngày mới cách ngày DateTime dt n ngày. *Date.AddMonths(n)*: trả về một ngày mới cách ngày DateTime dt n tháng. *Date.AddYears(n)*: trả về một ngày mới cách ngày Date n năm.

Để tính khoảng cách giữa 2 ngày ta làm nhƣ ví dụ sau:



**Chú ý:** định dạng hiển thị của ngày tháng phụ thuộc vào quy định của Windows, giá trị mặc định là hiển thị theo định dạng kiểu Mỹ (tháng trƣớc - ngày sau). Ta có thể thay đổi dạng hiển thị ngày trƣớc – tháng sau bằng cách thay đổi trong hộp hội thoại Date tại mục Regional Setting của cửa sổ Control Panel trong Windows.

### Kiểu mảng

Mảng là một tập hợp các biến có cùng tên và cùng kiểu dữ liệu. Dùng mảng làm chƣơng trình đơn giản và ngắn gọn hơn.

Ta có thể truy xuất các phần tử của mảng thông qua chỉ số, chỉ số của mảng bắt đầu từ

0.

Có mảng một chiều và mảng nhiều chiều.

### Mảng một chiều

Cú pháp: *type[ ] array-name;*

Ta có thể khai báo mảng với kích thƣớc cố định ngay từ đầu. Ví dụ:

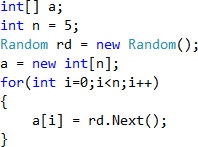
int[5] a; // Khai báo mảng a có 5 phần tử nguyên.

Ta cũng có thể khai báo mảng, sau đó cấp phát bộ nhớ cho mảng với từ khóa new. Điều này giúp ta quản lý bộ nhớ tốt hơn. Kích thƣớc của mảng sẽ đƣợc xác định lúc thi hành chƣơng trình.

*type[ ] array-name;*

*array-name =new type[n]; //khai báo mảng có n phần tử.*

Ví dụ: đoạn chƣơng trình sau sẽ tạo ra 5 phần tử nguyên cho mảng a



Ta cũng có thể gán các phần tử của mảng khi khai báo nhƣ sau:

string[] myArray = {"first element", "second element", "third element"};

#### Ta có thể làm việc với mảng:

Array.Reverse(a): Đảo ngƣợc mảng a. Array.Sort(a): Sắp xếp mảng a theo chiều tăng dần. a.Length: Trả về chiều dài của mảng a.

### Mảng nhiều chiều

Cú pháp: *type[,] array-name;*

Ta có thể khai báo mảng với kích thƣớc cố định ngay từ đầu. Ví dụ:

int[2,3] a; // Khai báo mảng nguyên a có 2 hàng 3 cột.

Cũng nhƣ mảng một chiều, ta cũng có thể khai báo mảng, sau đó cấp phát bộ nhớ cho mảng với từ khóa new.

*type[,] array-name;*

*array-name =new type[n,m]; //khai báo mảng có n hàng, m cột*

* + 1. ***Chuyển đổi giữa các kiểu dữ liệu***

|  |  |
| --- | --- |
| **Hàm chuy**ể**n đổi** | **Đổi giá tr**ị **sang ki**ể**u** |
| 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.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(“12/12/1988”) trả về giá trị kiểu ngày tháng 12/12/1988

+ Convert.ToInt32(“25”) = 25.

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

## Hộp thoại thông báo - Message Box

* + 1. ***Khái niệm***

Hộp thông báo là hộp thoại cung cấp thông tin để tƣơng tác với ngƣời sử dụng, đồng thời cũng là nơi hiển thị các kết quả trung gian trong quá trình tính toán. Trong thời gian hiển thị thông báo. C# ngừng mọi hoạt động của biểu mẫu và ngƣời dùng chỉ có thể làm việc với hộp thông báo.

kép.

* + 1. ***Hộp thông báo MessageBox***

Cú pháp hộp thông báo:

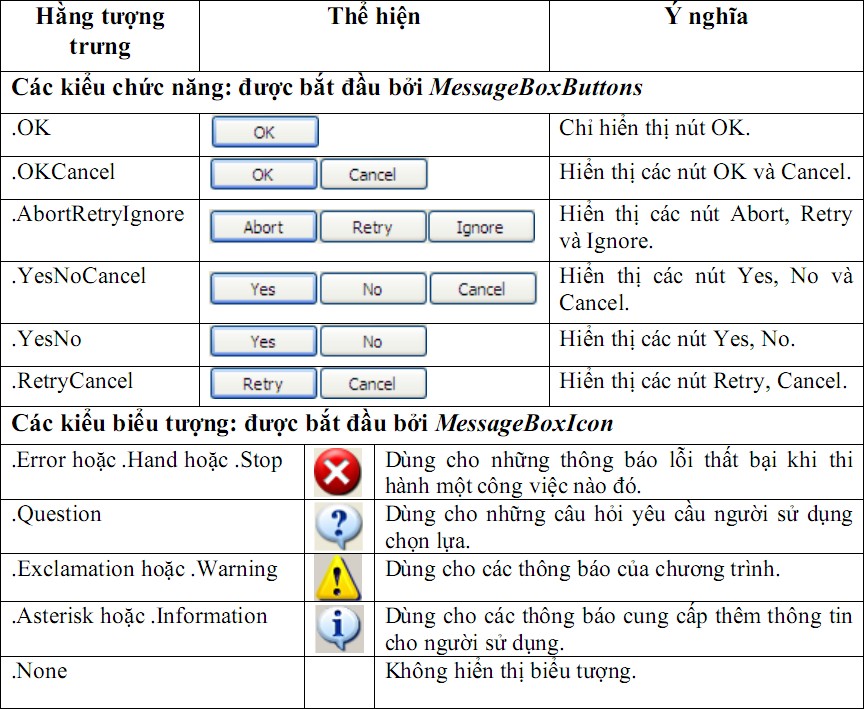
***MessageBox.Show****(Nội dung thông báo[, Tiêu đề, Kiểu chức năng, Kiểu biểu tượng])*

Trong đó:

+ Cú pháp hộp thông báo không nhất thiết phải có đầy đủ 4 thành phần trên.

+ Nội dung cần thông báo và tiêu đề của hộp thông báo đƣợc đặt trong cặp dấu nháy

+ Kiểu chức năng và kiểu biểu tƣợng có các giá trị nhƣ sau:



Ví dụ hiển thị hộp thông báo “Bạn chƣa nhập dữ liệu” với một nút lệnh OK và biểu tƣợng Information ta viết nhƣ sau:

MessageBox.Show("Bạn chưa nhập dữ liệu", "Thông báo", MessageBoxButtons.OK,

MessageBoxIcon.Information);

Kết quả ta có:



* + 1. ***Hàm thông báo MessageBox***

Ngoài chức năng thông báo, hàm MessageBox còn trả về giá trị của các nút chức năng mà ngƣời dùng đã chọn.

Cú pháp của hàm MessageBox nhƣ 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 bao gồm:

+ System.Windows.Forms.DialogResult.OK

+ System.Windows.Forms.DialogResult.Cancel

+ System.Windows.Forms.DialogResult.Abort

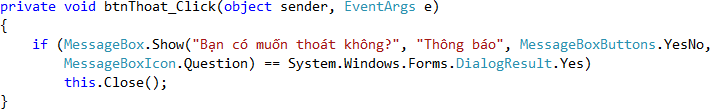
+ System.Windows.Forms.DialogResult.Retry

+ System.Windows.Forms.DialogResult.Ignore

+ System.Windows.Forms.DialogResult.Yes

+ System.Windows.Forms.DialogResult.No

**Ví dụ:** Ta có thể viết code cho nút btnExit với yêu cầu chỉ thoát khi ngƣời dùng trả lời có muốn thoát nhƣ sau:



## Các câu lệnh điều khiển

* + 1. ***Câu lệnh lựa chọn if***

Lệnh **if** có 2 dạng sau:

### Dạng 1:

if (<Biểu thức điều kiện>)

<khối lệnh>

***Hoạt động:*** Khi gặp câu lệnh này, máy sẽ tính giá trị của biểu thức. Nếu biểu thức đúng thì máy sẽ thực hiện khối lệnh và sau đó thực hiện các lệnh tiếp theo. Nếu biểu thức sai thì máy sẽ bỏ qua khối lệnh và chuyển đến các lệnh tiếp theo.

### Dạng 2:

if (<Biểu thức điều kiện>)

<Khối lệnh 1>

else

<Khối lệnh 2>

***Hoạt động:*** Khi gặp câu lệnh này, máy sẽ tính giá trị của biểu thức. Nếu biểu thức đúng máy sẽ thực hiện khối lệnh 1 sau đó nhảy tới các lệnh sau khối lệnh 2. Nếu biểu thức sai thì máy không thực hiện khối lệnh 1 mà chỉ thực hiện khối lệnh 2 và sau đó thực hiện các lệnh viết sau nó.

**Chú ý:** Lệnh if có thể lồng nhau.

* + 1. ***Câu lệnh lựa chọn switch … case***

switch (Biểu thức kiểm tra)

{

case n1:

Các câu lệnh break;

case n2:

Các câu lệnh break;

case nk:

Các câu lệnh break;

[default:

Các câu lệnh]

}

**Hoạt động:** Sự hoạt động của toán tử switch phụ thuộc vào giá trị của biểu thức kiểm tra viết trong dấu ngoặc tròn.

* Khi giá trị này bằng ni máy sẽ nhảy tới câu lệnh có nhãn case ni.
* Khi giá trị của biểu thức khác tất cả các ni thì cách làm việc của máy lại phụ thuộc vào sự có mặt hay vắng mặt của default
  + Khi có default, máy nhảy tới câu lệnh có nhãn default.
  + Khi không có default, máy ra khỏi toán tử switch.
    1. ***Câu lệnh lặp for***

Lệnh for cho phép thực hiện một khối lệnh một số lần hữu hạn. Lệnh for thƣờng dùng để giải các bài toán có tính chu trình, ví dụ nhƣ các bài toán về dãy số, về ma trận.

### Cú pháp của for

Lệnh for có dạng sau:

for (<Biểu thức khởi tạo>;<Biểu thức kiểm tra> ; <Biểu thức tăng,giảm>)

<Khối lệnh>

### Hoạt động của for

Lệnh for làm việc theo các bƣớc sau:

* Bƣớc 1: Tính giá trị biểu thức khởi tạo.
* Bƣớc 2: Tính giá trị của biểu thức kiểm tra. Nếu đúng tới bƣớc 3, nếu sai thoát khỏi lệnh for.
* Bƣớc 3: Thực hiện khối lệnh.
* Bƣớc 4: Tính giá trị biểu thức tăng,giảm sau đó qua lại bƣớc 2.

### Chú ý:

* Nếu biểu thức kiểm tra vắng mặt thì nó đƣợc xem là đúng. Để tránh lặp vô hạn thì trong phần thân của lệnh for phải có lệnh nhảy ra khỏi for bằng các lệnh break, return hoặc goto.
  + 1. ***Câu lệnh lặp while***

### Cú pháp của while

Cú pháp của while có dạng:

while (<Biểu thức kiểm tra>)

<Khối lệnh>

Trong cú pháp trên:

* while: từ khóa của lệnh while.
* Biểu thức kiểm tra bắt buộc phải đặt trong cập ngoặc tròn.
* Khối lệnh, còn gọi là phần thân của while, là khối lệnh cần thực hiện nhiều lần.

### Hoạt động của while

Lệnh while làm việc theo các bƣớc sau:

* Bƣớc 1: Tính giá trị biểu thức kiểm tra. Nếu biểu thức đúng sang bƣớc 2, ngƣợc lại thoát khỏi lệnh while.
* Bƣớc 2: Thực hiện khối lệnh sau đó quay trở lại bƣớc 1.

**Nhận xét:** Thân của lệnh while có thể đƣợc thực hiện một lần hoặc nhiều lần và cũng có thể không đƣợc thực hiện lần nào nếu ngay từ đầu biểu thức kiểm tra đã sai.

* + 1. ***Câu lệnh lặp do…while***

Lệnh **do … while** có dạng sau:

do

<Khối lệnh>

while (Biểu thức kiểm tra);

### Hoạt động của do … while

Lệnh **do … while** thực hiện nhƣ sau:

* Bƣớc 1: Thực hiện khối lệnh.
* Bƣớc 2: Tính giá trị biểu thức kiểm tra. Nếu đúng quay trở lại bƣớc 1, ngƣợc lại thoát khỏi lệnh **do … while**.
  + 1. ***Câu lệnh lặp foreach***

Cho phép bạn duyệt qua tất cả các phần tử của mảng hoặc các tập hợp khác, và tuần tự xem xét từng phần tử một.Cú pháp nhƣ sau:

**foreach** (<kiểu dl> <tên biến truy cập> in <mảng hoặc tập hợp>)

<khối lệnh>

* + 1. ***Câu lệnh try...catch***

Đƣợc dùng trong các câu lệnh bẫy lỗi ngoại lệ của chƣơng trình. Cú pháp nhƣ sau:

try

{

}

catch

{

}

//Lệnh thực thi trong trường hợp bình thường

//Lệnh xử lý lỗi

lặp.

* + 1. ***Câu lệnh break***

Ta dùng câu lệnh break khi muốn ngƣng ngang chừng việc thi hành và thoát khỏi vòng

* + 1. ***Câu lệnh Continue***

Câu lệnh continue đƣợc dùng trong vòng lặp khi bạn muốn khởi động lại một vòng lặp

nhƣng lại không muốn thi hành phần lệnh còn lại trong vòng lặp, ở một điểm nào đó trong thân vòng lặp.

## Phương thức trong C#

* + 1. ***Định nghĩa phương thức***

Cú pháp xây dựng một phƣơng thức nhƣ sau:

[*modifiers*] *return\_type MethodName*([*parameters*])

{

// Thân phương thức

}

**Modifiers:** Là phạm vi truy xuất có thể là private, public… thành phần này có thể vắng mặt, nếu vắng mặt thì Modifiers đƣợc coi mặc định là private.

Có 2 loại phƣơng thức nhƣ trong C++, java đó là:

+ Phƣơng thức có giá trị trả về (Có kiểu khác void). Loại này trong thân phƣơng thức phải có lệnh return.

+ Phƣơng thức không có giá trị trả về, phƣơng thức void. Loại này không có lệnh return trong thân hàm.

* + 1. ***Gọi phương thức***

Phƣơng thức có giá trị trả về đƣợc sử dụng nhƣ một thành phần của biểu thức, nên ta có thể dùng theo cú pháp sau:

*<biến>= MethodName*([*parameters*])

Còn phƣơng thức kiểu void (không có giá trị trả về) đƣợc dùng nhƣ một lệnh độc lập, nó không đƣợc dùng trong lệnh gán hay biểu thức

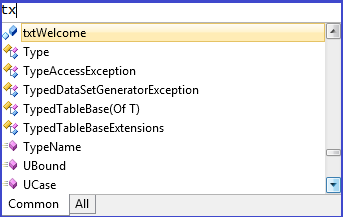
*MethodName([parameters])*

## Gỡ rối chương trình

Khi lập trình, ngƣời lập trình không tránh khỏi những sai sót và mắc lỗi, tuy nhiên ta có thể giảm khả năng mắc lỗi đến mức tối thiểu.

* + 1. ***Một số giải pháp gỡ rối chương trình***
    - Thiết kế cẩn thận, ghi chú các vấn đề quan trọng và cách giải quyết cho từng phần. Ghi chú từng phƣơng thức và mục đích của nó.
    - Chú thích rõ ràng trong chƣơng trình.
* Để đặt chú thích cho các dòng lệnh, ta có thể gõ dấu nháy đơn // tại vị trí cần đặt chú thích, hoặc bôi đen các dòng lệnh rồi kích vào biểu tƣợng  trên thanh công cụ standard.
* Để xóa dấu chú thích ở các dòng lệnh, ta có thể xóa dấu // hoặc bôi đen các dòng lệnh muốn xóa chú thích rồi kích chọn biểu tƣợng  trên thanh công cụ Standard.
  + - Dùng cửa sổ danh sách các thuộc tính, phƣơng thức, các hằng số, các lớp đối tƣợng... trong C# để tránh gõ sai tên thuộc tính, phƣơng thức, ...

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



* + - Sử dụng câu lệnh try...catch để bắt các lỗi ngoại lệ của chƣơng trình.
    1. ***Dò lỗi từng dòng lệnh***

Một trong những cách dò lỗi hiệu quả là thực hiện chƣơng trình bằng cách chạy từng dòng lệnh. Khi đó, ta có thể kiểm tra sự thay đổi giá trị của các biến, các thuộc tính… qua từng bƣớc thực hiện chƣơng trình. Để tìm hiểu cách dò lỗi từng dòng lệnh ta thực hiện theo các bƣớc của ví dụ sau:

**Đề bài:** viết một phƣơng thức tính giai thừa, sau đó tính và in ra kết quả của 6!

*Bước 1:* mở một dự án mới rồi gõ đoạn mã sau vào cửa sổ code.

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 MinhHoaChuong3

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

long GiaiThua(int n)

{

int i; long gt; gt = 1;

for (i = 1; i <= n; i++) gt += i;

return gt;

}

private void Form1\_Load(object sender, EventArgs e)

{

long a;

a = GiaiThua(6); MessageBox.Show(a.ToString());

}

}

}

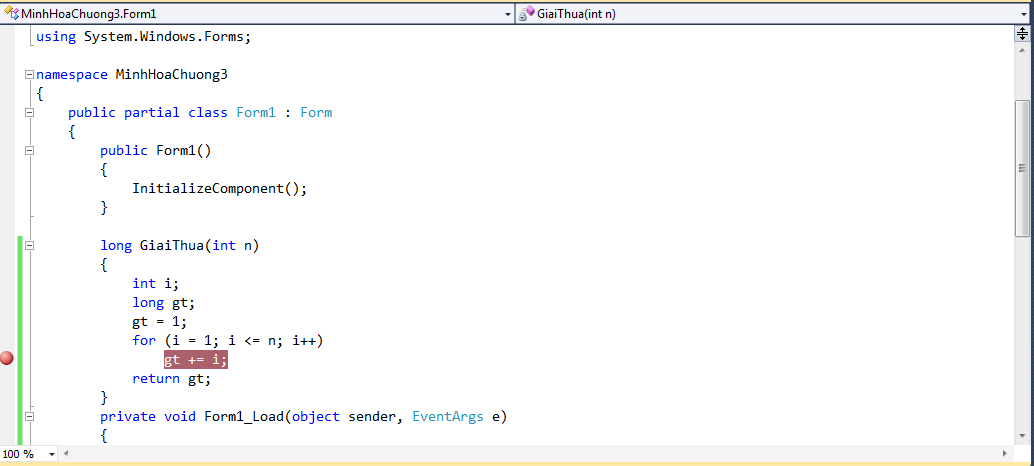
*Bước 2:* bấm F5 để thực hiện chƣơng trình, ta thấy kết quả của chƣơng trình là 6!=22. Đây là một kết quả sai. Nhiệm vụ của chúng ta là xác định chƣơng trình sai ở đâu?

*Bước 3*: khoanh vùng và đặt điểm dừng Breakpoint tại dòng mã lệnh ta nghi ngờ là sai để theo dõi kết quả thực hiện của nó. Trong cửa sổ soạn thảo code, đặt con trỏ tại dòng lệnh muốn đặt Breakpoint (ví dụ dòng gt +=i) và thực hiện theo 1 trong 3 cách sau:

+ Vào Debug\Toggle Breakpoint

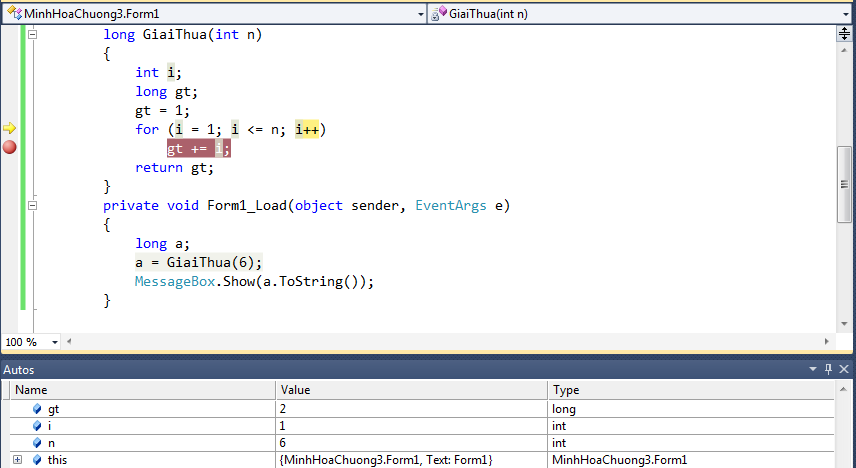
+ Bấm phím F9

+ Di chuột ra bên lề trái của cửa sổ soạn thảo code, kích trái chuột tại dòng lệnh muốn đặt Breakpoint

Kết quả C# sẽ hiển thị một vòng tròn màu đỏ và tô nền mầu đỏ tại dòng lệnh đã đặt Breakpoint. Khi thực hiện đến dòng lệnh này, chƣơng trình sẽ dừng lại để ta tự chạy chƣơng trình kiểm tra từng bƣớc.

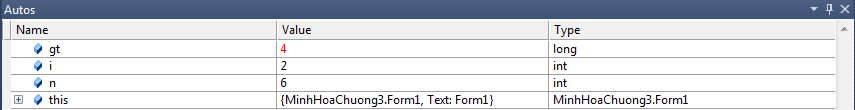
*Bước 4:* bấm F5 để thực hiện chƣơng trình, C# dừng chƣơng trình tại dòng lệnh đặt Breakpoint và màu nền của dòng lệnh này đã chuyển từ màu đỏ sang màu vàng cho biết tiếp theo chƣơng trình sẽ thực hiện lệnh này.

*Bước 5:* nhấn nút Step Into  trên thanh công cụ Standard để thực hiện dòng lệnh có nền màu vàng, rồi vào Debug\Windows\Autos để mở cửa sổ Autos xem giá trị của các biến tại thời điểm hiện thời.

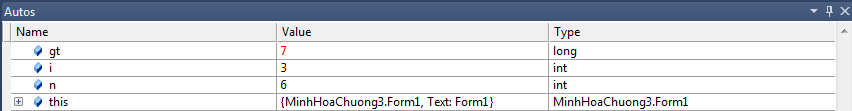


Trong cửa sổ Autos giá trị của gt =2.0, i=1 và n=6. Ta thấy giá trị của gt sai vì với i=1 thì gt cũng phải có giá trị là 1.0

*B*ƣớ*c 6*: tiếp tục nhấn 2 lần nút Step Into, cửa sổ Autos có giá trị nhƣ sau



*B*ƣớ*c 7*: tiếp tục nhấn 2 lần nút Step Into, ta có:



Đến đây ta thấy giá trị gt mới đƣợc tính bằng cách cộng thêm giá trị của i hiện thời vào giá trị gt cũ, chứ không phải là nhân theo cách tính giai thừa. Nhƣ vậy trong công thức tính gt ta đã gõ nhầm dấu \* thành dấu +.

*B*ƣớ*c 8*: nhấn nút Stop Debugging  để thoát khỏi chế độ ngắt trở về môi trƣờng phát triển của Visual Studio và sửa lại công thức tính Gt thành gt \*= i.

*B*ƣớ*c 9*: ta thấy khi trở về môi trƣờng phát triển của Visual Studio thì điểm dừng Breakpoint vẫn tồn tại. Để loại bỏ điểm dừng Breakpoint, trong cửa sổ soạn thảo code, đặt con trỏ tại dòng lệnh muốn bỏ Breakpoint và thực hiện theo 1 trong 3 cách sau:

+ Vào Debug\Toggle Breakpoint

+ Bấm phím F9

+ Di chuột ra bên lề trái của cửa sổ soạn thảo code, kích trái chuột tại dòng lệnh đang đặt Breakpoint.

Ngoài ra, nếu chƣơng trình có nhiều điểm Breakpoint và để loại bỏ hết các điểm này cùng một lúc thì ta thực hiện theo 1 trong 2 cách sau:

+ Vào Debug\Delete All Breakpoints

+ Bấm tổ hợp phím Ctrl + Shift + F9

*B*ƣớ*c 10*: đến đây hãy bấm F5 để chạy lại chƣơng trình, ta thấy kết quả là 6! = 720

# CHƯƠNG 3: TÌM HIỂU CÁC ĐIỀU KHIỂN CƠ BẢN

Trong chƣơng này chúng ta sẽ tìm hiểu về cách sử dụng (các thuộc tính, phƣơng thức, sự kiện) của các điều khiển cơ bản trên form.

## Thuộc tính, phương thức, sự kiện và các mối quan hệ giữa chúng

Mỗi một đối tƣợng trong C# đều có 3 đặc tính là *Thu*ộ*c tính - Properties, Ph*ƣơ*ng th*ứ*c - Methods và S*ự *ki*ệ*n – Events.* Trong đó:

+ **Thuộc tính - Properties:** là tập hợp các đặc tính để mô tả một đối tƣợng nhƣ: 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).

+ **Phương thức - Methodes:** là những đoạn chƣơng 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 (phƣơng thức Hide).

+ **Sự kiện - Events:** nếu nhƣ thuộc tính mô tả đối tƣợng, phƣơng thức chỉ ra cách thức đối tƣợng hành động thì sự kiện là những phản ứng của đối tƣợng. Khi tạo một chƣơng 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 chƣơng trình chỉ thi hành khi ngƣời dùng thực hiện một thao tác nào đó trên giao diện.

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

Mặc dù thuộc tính, phƣơng thức và sự kiện có vai trò khác nhau nhƣng chúng thƣờng xuyên liên hệ với nhau. Ví dụ, nếu ta di chuyển một điều khiển bằng phƣơng thức Move thì một số thuộc tính nhƣ Top, Height, Left, Width sẽ thay đổi 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 phƣơng thức.

**Ví d**ụ: ta có 2 cách để làm biểu mẫu frmWelcome xuất hiện và biến mất trên màn hình Thực hiện bằng thuộc tính:

*Xu*ấ*t hi*ệ*n*: frmWelcome.Visible = True *Bi*ế*n m*ấ*t*: frmWelcome.Visible = False Thực hiện bằng phƣơng thức:

*Xu*ấ*t hi*ệ*n*: frmWelcome.Show()

*Bi*ế*n m*ấ*t*: frmWelcome.Hide()

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

* + 1. ***Form***

### 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. |
| **Cursor** | Thiết lập chế độ hiển thị con trỏ trên Form. |
| **Enabled** | Nếu nhận giá trị True thì cho phép ngƣời dùng tác động lên Form, ngƣợc 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 đƣờng viền cho Form. Fixed Single: không thể thay đổi kích thƣớc của Form, Sizable: có thể phóng to thu nhỏ và thay đổi kích thƣớc của Form, Sizable ToolWindow: có thể thay đổi  kích thƣớc của Form… |
| **Icon** | Thiết lập biểu tƣợng 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 Form sẽ dần trở nên trong suốt có thể nhìn xuyên qua thấy những gì nằm bên dƣới Form. |
| **ShowIcon** | Nếu nhận giá trị True thì cho phép hiển thị biểu tƣợng đã đƣợc thiết lập ở thuộc tính Icon, ngƣợc 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: xuất hiện ở giữa màn hình… |
| **Text** | Thiết lập dòng tiêu đề của Form. |
| **Window State** | Thiết lập trạng thái của Form khi chạy chƣơng 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 của hệ điều hành. |

1. **Sự kiện**

Form có một số sự kiện thông dụng nhƣ sau:

|  |  |
| --- | --- |
| **Load** | Sự kiện Load đƣợc kích hoạt khi Form đƣợc nạp vào bộ nhớ, nó thƣờng đƣợc dùng để khởi tạo các giá trị và trạng thái cho các biến, các điều khiển… trên Form. |
| **Click** | Đƣợc kích hoạt khi ngƣời dùng kích chuột trên Form. |
| **FormClosed** | Đƣợc kích hoạt khi ngƣời dùng kích chuột vào nút Close **x** ở góc trên bên phải để đóng Form. |
| **FormClosing** | Cũng đƣợc kích hoạt khi ngƣời dùng kích chuột vào nút Close **x,** nhƣng xảy ra trƣớc sự kiện FormClosed tức là đƣợc phát sinh trƣớc khi cửa sổ Form chuẩn bị đóng lại. |

Chú ý: Nếu không muốn ngƣời dùng đóng Form bằng cách bấm chọn biểu tƣợng Close thì trong sự kiện **FormClosing** ta đặt thuộc tính Cancel = True nhƣ sau:

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

e.Cancel = true;

}

* + 1. ***Hộp văn bản – TextBox***

Hộp văn bản  là điều khiển rất thông dụng, dùng để nhập dữ liệu đầu vào từ phía ngƣời sử dụng và hiển thị các kết quả đã tính toán đƣợc.

### 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 ngƣời dùng truy cập vào TextBox (Hộp Textbox bị mờ đi), ngƣợc 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, ngƣợc lại thì nhận giá trị False. |
| **MaxLength** | Quy định chiều dài tối đa đƣợc 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 ngƣời 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 đƣợc độ dài của hộp văn bản. Multiline = True: cho phép văn bản đƣợc 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 ngƣời sử dụng bảo mật đƣợc thông tin nhập vào Textbox. Ví dụ đặt thuộc tính này bằng ký tự „\*‟ khi đó toàn bộ dữ liệu nhập vào sẽ đƣợc hiển thị dƣới dạng dấu hoa thị. Chú ý: thuộc tính này chỉ đƣợc hỗ trợ khi thuộc tính Multiline = False. |
| **ReadOnly** | ReadOnly = True: hộp văn bản vẫn đƣợc truy cập nhƣng ngƣời dùng không thể thay đổi đƣợc nội dung bên trong. |
| **ScrollBars** | Thiết lập thanh cuốn ngang và dọc cho hộp văn bản, có hiệu lực khi thuộc tính Multiline = True. Chú ý: thanh cuốn ngang chỉ có hiệu lực  khi thuộc tính WordWrap = False. |
| **TabIndex** | Thứ tự truy cập của hộp văn bản khi ngƣời 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** | Chỉ có hiệu lực khi thuộc tính Multiline = True. WordWrap = True: dòng văn bản đƣợc tự động cuộn xuống dòng khi gặp lề bên phải của hộp TextBox, ngƣợc lại thì nhận giá trị False. |

1. **Sự kiện**

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

|  |  |
| --- | --- |
| **TextChanged** | Đƣợc kích hoạt khi ngƣời dùng thực hiện sự thay đổi bất kỳ trong hộp văn bản nhƣ: thêm, xoá, sửa, dán văn bản. |
| **Click** | Đƣợc kích hoạt khi ngƣời dùng kích chuột vào hộp văn bản. |
| **DoubleClick** | Đƣợc kích hoạt khi ngƣời dùng kích đúp chuột vào hộp văn bản. |
| **Enter** | Đƣợc kích hoạt khi ngƣời dùng chuyển tiêu điểm tới hộp văn bản. |
| **KeyPress** | Trả về ký tự (trừ các ký tự đặc biệt nhƣ phím Delete, Home, Ctrl, F1…) mà ngƣời sử dụng gõ vào hộp văn bản thông qua thuộc tính KeyChar. |
| **KeyDown** | Trả về mã Ascii của tất cả các ký tự mà ngƣời sử dụng gõ vào hộp văn bản thông qua thuộc tính KeyValue. |
| **Leave** | Đƣợc kích hoạt khi hộp văn bản mất tiêu điểm. |
| **MouseMove** | Đƣợc kích hoạt khi ngƣời dùng di chuyển chuột qua hộp văn bản. |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **MouseLeave** | Đƣợc kích hoạt khi ngƣời dùng dời chuột ra khỏi hộp văn bản. |  |

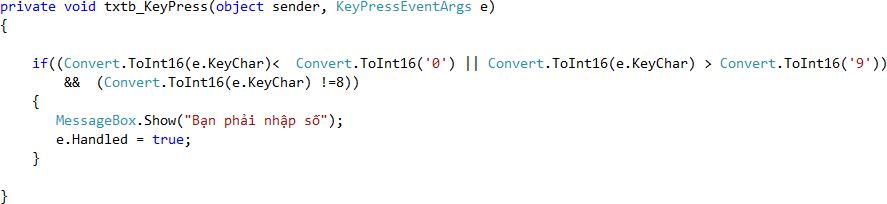
**Ví dụ 1**: Để hiển thị mã Ascii của một ký tự bất kỳ đƣợc gõ vào hộp văn bản ***txta*** ta có đoạn chƣơng trình nhƣ sau cho sự kiện KeyDown:

private void txta\_KeyDown(object sender, KeyEventArgs e)

{

MessageBox.Show(e.KeyValue.ToString());

}

**Ví dụ 2**: Dùng thủ tục **KeyPress** để kiểm tra việc nhập dữ liệu: chỉ cho phép nhập vào hộp văn bản ***txtb*** các số từ 0 tới 9 và phím Backspace (có mã Ascii = 8) để xóa dữ liệu, ta có đoạn chƣơng trình nhƣ sau:

Nếu mỗi ký tự đƣợc nhập vào hộp Textbox nằm ngoài khoảng từ ký tự "0" đến ký tự "9" và mã Ascii của ký tự đó khác 8 thì thông báo phỉ nhập số và đổi ký tự đó thành ký tự rỗng. Vì vậy hộp văn bản chỉ nhận các số từ 0 đến 9 và phím xóa Backspace.

* + 1. ***Nút lệnh – Button***

Nút lệnh  cho phép ngƣời dùng thực hiện một hành động nào đó.

### 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 λền cho nút lệnh. |
| **Enabled** | Enabled=False: ngƣời dùng không thể tác động lên nút lệnh, ngƣợc 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, ngƣợc lại thì nhận giá trị False. |
| **TabIndex** | Thứ tự truy cập của nút lệnh khi ngƣời dùng bấm phím Tab. |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Text** | Tiêu đề của nút lệnh. Ta có thể quy định phím nóng cho nút lệnh bằng cách đặt dấu “&” trƣớc một ký tự của Text, ví dụ &Quit sẽ đƣợc hiển thị là Quit, khi ngƣời sử dụng bấm Alt+Q chƣơng trình sẽ kích hoạt nút lệnh Quit. |  |
| **Visible** | Visible = True: hiển thị nút lệnh, Visible = False: ẩn nút lệnh. |

1. **Sự kiện**

Nút lệnh có một số sự kiện cơ bản sau:

|  |  |
| --- | --- |
| **Click** | Đƣợc kích hoạt khi ngƣời dùng kích chuột vào nút lệnh. |
| **Enter** | Đƣợc kích hoạt khi ngƣời dùng chuyển tiêu điểm tới nút lệnh. |
| **Leave** | Đƣợc kích hoạt khi nút lệnh mất tiêu điểm. |
| **MouseDown** | Đƣợc kích hoạt khi ngƣời dùng đặt chuột vào nút lệnh. |
| **MouseUp** | Đƣợc kích hoạt khi ngƣời dùng đƣa chuột ra khỏi nút lệnh. |
| **MouseMove** | Đƣợc kích hoạt khi ngƣời dùng di chuyển chuột trên nút lệnh. |
| **MouseLeave** | Đƣợc kích hoạt khi ngƣời dùng dời chuột ra khỏi nút lệnh. |

* + 1. ***Nhãn – Label***

Nhãn  dùng để hiển thị những thông tin có tính chất cố định ngƣời sử dụng không có khả năng thay đổi ví dụ nhƣ dòng thông báo, hƣớng 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 đƣờng 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, ngƣợc lại thì nhận giá trị False. |
| **TabIndex** | Thứ tự truy cập của nhãn khi ngƣời 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. |

* + 1. ***Dòng mách nước – ToolTip***

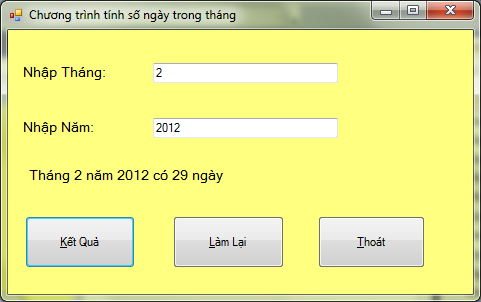
Dòng mách nƣớc  cho phép hiển thị các thông tin chú thích khi ngƣời dùng đƣa chuột qua một điều khiển dùng ToolTip. Ví dụ dòng mách nƣớc “Hãy nhập tên truy cập” nhƣ hình dƣới đây.



Để tạo dòng mách nƣớc, ta phải kéo một điều khiển ToolTip vào Form (điều khiển ToolTip không đƣợc hiển thị ở trên Form mà đƣợc hiển thị ở thanh ngang cuối Form). Sau đó kích chuột chọn một điều khiển bất kỳ muốn tạo ToolTip, trong cửa sổ Window Properties gõ nội dung dòng ToolTip tại thuộc tính ToolTip on ToolTip1.

* + 1. ***Bài tập***

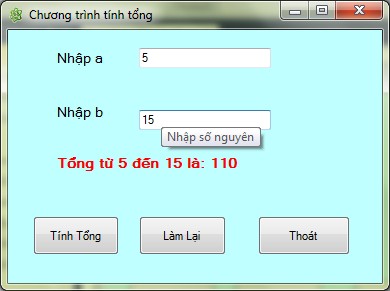
**Bài tập 1:** *(Giáo viên hướng dẫn trên lớp)*

Lập chƣơng trình nhập một tháng và một năm dƣơng lịch, tính và đƣa ra số ngày của tháng và năm đó?

**Yêu cầu:** Chỉ đƣợc 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. Kết quả chỉ đƣợc tính khi ngƣời dùng nhập đủ cả tháng và năm.

### Bài tập 2:

Lập trình giải bài toán nhập 2 số nguyên a, b và tính tổng các số từ a đến b.



### Yêu cầu:

+ Tạo giao diện theo Form trên.

+ Tạo dòng ToolTip “Nhập số nguyên” cho 2 hộp văn bản „Nhập a‟ và „Nhập b‟.

+ Chỉ cho phép ngƣời dùng nhập số vào hai hộp văn bản.

+ Nút Tổng, kiểm tra ngƣời 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 dƣới.

+ 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 chƣơng trình quay về môi trƣờng soạn thảo.

### Bài tập 3:

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



### Yêu cầu:

+ Tạo giao diện theo Form trên.

+ Chỉ cho phép ngƣời dùng nhập số vào hộp văn bản Nhập n.

+ Nút Nhập, kiểm tra ngƣời dùng phải nhập giá trị cho n, sau đó tạo n số ngẫu nhiên và hiển thị các số ngẫu nhiên đó ở nhãn Dẫy số.

+ Nút Tính tổng, tính tổng n số ngẫu nhiên và hiển thị kết quả ở nhãn Tổng dẫy số.

+ Nút Sắp xếp, sắp xếp n số ngẫu nhiên theo thứ tự giảm 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 chƣơng trình quay về môi trƣờng soạn thảo.

## Một số điều khiển cơ bản khác

* + 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 thƣờng 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, ngƣợc lại thì nhận giá trị False. |
| **TabIndex** | Thứ tự truy cập của nhóm khi ngƣời dùng bấm phím Tab. |
| **Text** | Thiết lập tiêu đề của nhóm. |
| **Visible** | Visible = True: hiển thị nhóm, Visible = False: ẩn nhóm. |

* + 1. ***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.

### 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 tƣơng tác với ngƣời sử dụng. Checked =True: hộp CheckBox đang đƣợc chọn, Checked = False: hộp CheckBox không đƣợc chọn. |
| **CheckState** | Thiết lập trạng thái cho hộp CheckBox, CheckState = Checked: hộp CheckBox đƣợc chọn, CheckState = Unchecked: hộp CheckBox không đƣợc 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 |
| **TabIndex** | Thứ tự truy cập khi ngƣời dùng bấm phím Tab. |
| **Text** | Thiết lập nội dung của hộp CheckBox. |
| **Visible** | Visible = True: hiển thị hộp CheckBox, Visible = False: ẩn hộp |

1. **Sự kiện**

|  |  |
| --- | --- |
| **Click** | Đƣợc kích hoạt khi ngƣời dùng kích chuột vào hộp CheckBox. |
| **Enter** | Đƣợc kích hoạt khi ngƣời dùng chuyển tiêu điểm tới hộp CheckBox. |
| **Leave** | Đƣợc kích hoạt khi hộp CheckBox mất tiêu điểm. |
| **CheckedChanged** | Đƣợc kích hoạt khi hộp CheckBox thay đổi trạng thái. |

* + 1. ***Nút tùy chọn – RadioButton***

Nút tuỳ chọn  chỉ cho phép ngƣời dùng chọn một khả năng trong một nhóm các lựa chọn.

### 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 tƣơng tác với ngƣời sử dụng. Checked = True: nút tùy chọn đang đƣợc chọn, Checked = False: nút tùy chọn không đƣợc 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, ngƣợc lại thì nhận giá trị False. |
| **TabIndex** | Thứ tự truy cập khi ngƣời 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. |

1. **Sự kiện**

|  |  |
| --- | --- |
| **Click** | Đƣợc kích hoạt khi ngƣời dùng kích chuột vào nút tuỳ chọn. |
| **Enter** | Đƣợc kích hoạt khi ngƣời dùng chuyển tiêu điểm tới nút tuỳ chọn. |
| **Leave** | Đƣợc kích hoạt khi nút tuỳ chọn mất tiêu điểm. |
| **CheckedChanged** | Đƣợc kích hoạt khi nút tùy chọn thay đổi trạng thái. |

**Bài tập 4:** *(Giáo viên hướng dẫn trên lớp)*

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



+ Chỉ đƣợc nhập số cho a và b, không cho phép nhập dữ liệu vào hộp kết quả.

+ Các phép toán chỉ đƣợc thực hiện khi ngƣời 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 lƣu kết quả vào hộp Kết quả.

+ Nếu chọn hộp đánh dấu Max thì hiển thị “Max = <Giá trị max>” ngƣợc lại chỉ hiển thị “Max”. Thực hiện tƣơng tự cho hộp đánh dấu Min.

* + 1. ***Hộp danh sách – ListBox***

Hộp ListBox  là một tập hợp các chuỗi ký tự đƣợc trình bày dƣới 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 vƣợt quá chiều dài của Listbox thì Listbox tự động cung cấp thanh cuốn dọc để cuộn tới các phần tử phía dƣới.

### 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 ngƣời lập trình gõ vào giá trị các phần tử. Mỗi phần tử đƣợc 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** | Tƣơng tự nhƣ thuộc tính Items(n), nhƣng chỉ có thể trả về nội dung của phần tử hiện hành đang đƣợc chọn. |
| **Locked** | Locked = True: khóa không cho phép dịch chuyển vị trí của hộp danh sách trên Form, ngƣợc lại thì nhận giá trị False. |
| **SelectedIndex** | Trả về số thứ tự của phần tử đang đƣợc chọn trong danh sách, phần tử đầu tiên có SelectedIndex = 0, nếu không có phần tử nào đƣợc chọn thì SelectedIndex = -1 |
| **SelectionMode** | Quy định chế độ lựa chọn các phần tử trong hộp danh sách khi thực thi chƣơng 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 đƣợc chọn. |
| **Sorted** | Nếu Sorted = True thì các phần tử trong danh sách đƣợc sắp xếp theo |
| **TabIndex** | Thứ tự truy cập khi ngƣời dùng bấm phím Tab. |
| **Visible** | True: hiển thị hộp danh sách, False: ẩn hộp danh sách. |

1. **Sự kiện**

|  |  |
| --- | --- |
| **Click** | Đƣợc kích hoạt khi ngƣời dùng kích chuột vào hộp danh sách. |
| **DoubleClick** | Đƣợc kích hoạt khi ngƣời dùng kích đúp chuột vào hộp danh sách. |
| **Enter** | Đƣợc kích hoạt khi ngƣời dùng chuyển tiêu điểm tới hộp danh sách. |
| **Leave** | Đƣợc kích hoạt khi hộp danh sách mất tiêu điểm. |
| **SelectedIndex\_ Changed** | Đƣợc kích hoạt khi ngƣời dùng thay đổi trạng thái lựa chọn các dòng dữ liệu trong hộp văn bản. |

1. **Phương thức**

**Add**: dùng để bổ sung một phần tử cho hộp danh sách trong thời gian thi hành và thƣờng đƣợc viết trong thủ tục Form\_Load. Cú pháp của phƣơng 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 nhƣ 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 phƣơng thức này là:

#### ListName.Items.Remove(Item)

**Ví dụ**, xóa phần tử có giá trị “Ha Noi” trong hộp danh sách lstQue ta viết nhƣ 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 phƣơng thức này là:

#### ListName.Items.RemoveAt(Index)

Ví dụ, xóa phần tử ở vị trí 1 trong hộp danh sách lstQue ta viết nhƣ sau: lstQue.Items.RemoveAt(1)

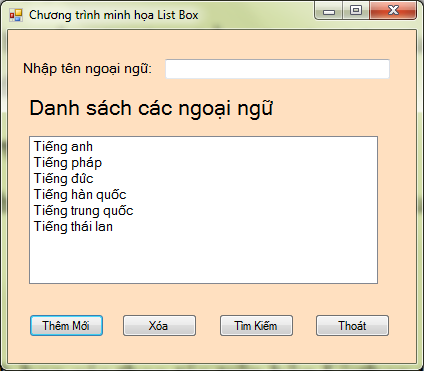
**Clear:** dùng để loại bỏ tất cả các phần tử của hộp danh sách trong thời gian thi hành. Cú pháp của phƣơng thức này là:

#### ListName.Items.Clear()

Ví dụ, xóa tất cả các phần tử trong hộp danh sách lstQue ta viết nhƣ sau: lstQue.Items.Clear()

**Bài tập 5:** *(Giáo viên hướng dẫn trên lớp)*

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



Nhập một tên ngoại ngữ vào hộp văn bản *Nh*ậ*p tên ngoại ngữ*, chọn nút *Thêm m*ớ*i* để thêm ngoại ngữ đó vào hộp danh sách, chọn nút *Tìm ki*ế*m* để xem ngoại ngữ đó đã có trong hộp danh sách chƣa?

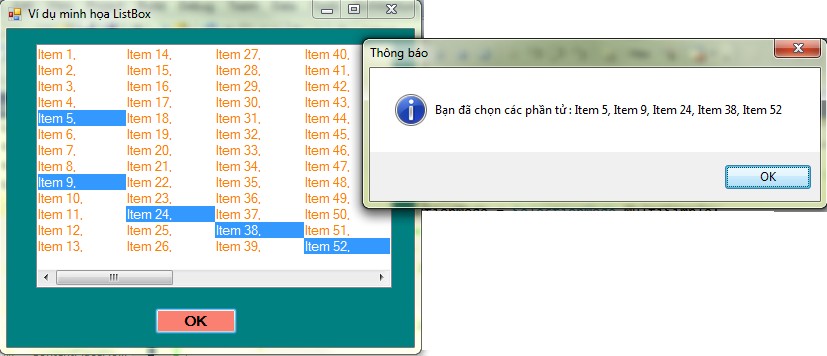
Chọn nút *Xóa* để xoá một tên ngoại ngữ bất kỳ đƣợc chọn từ hộp danh sách. Chọn nút

*Thoát* để thoát khỏi chƣơng trình.

### Bài tập 6:

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

Nhập vào hộp danh sách 100 phần tử từ Items 1 đến Items 100, dữ liệu đƣợc hiển thị thành 4 cột trong một trang màn hình và ngƣời dùng có thể lựa chọn một hoặc nhiều phần tử cùng một lúc.



### Hướng dẫn:

Vào Microsoft Visual Studio 2010 tạo một dự án mới, đặt một hộp danh sách *lstDanhSach* và một nút lệnh *btnOK* với tiêu đề là *OK* vào form Form1.

#### Viết mã lệnh:

Khai báo một biến toàn cục dùng để chứa chuỗi thông báo nhƣ sau:

string thongbao = "";

Trong sự kiện Load của Form1 ta viết lệnh nhƣ sau:

private void Form1\_Load(object sender, EventArgs e)

{

int i;

//Cho phép hiển thị trên nhiều cột lstDanhSach.MultiColumn = true;

//Hiển thị trên 4 cột

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 (i = 1; i <= 100; i++)

lstDanhSach.Items.Add("Item " + i.ToString());

}

Trong sự kiện Click của button *btnOK* ta viết mã lệnh nhƣ sau:

private void btnOK\_Click(object sender, EventArgs e)

{

foreach (string item in lstDanhSach.SelectedItems)

{

thongbao = thongbao + item + ", ";

}

//Xóa dấu phẩy và dấu cách thừa ở cuối chuỗi thongbao = thongbao.Remove(thongbao.Length - 2, 2);

MessageBox.Show("Bạn đã chọn các phần tử : " + thongbao, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

* + 1. ***Hộp lựa chọn – ComboBox***

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

### 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 ngƣời 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 ngƣời dùng chọn lựa.  ***Simple*** luôn hiển thị sẵn danh sách các mục dữ liệu bên dƣới hộp văn bản và cho phép ngƣời sử dụng có thể nhập dữ liệu vào hộp văn bản.  ***DropDownList*** tƣơng tự nhƣ DropDown nhƣng ngƣời 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 đƣợc 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 đƣợc chọn. |

|  |  |  |
| --- | --- | --- |
|  | **SelectedIndex** | Trả về số thứ tự của phần tử đang đƣợc chọn trong hộp Combo, phần tử đầu tiên có SelectedIndex = 0, nếu không có phần tử nào đƣợc chọn thì SelectedIndex = -1 |
| **Locked** | True: không cho phép dịch chuyển vị trí của hộp Combo trên Form |
| **Sorted** | True: các phần tử trong danh sách đƣợc sắp xếp theo thứ tự ABC. |
| **TabIndex** | Thứ tự truy cập khi ngƣời dùng bấm phím Tab. |
| **Visible** | True: hiển thị hộp Combo, False: ẩn hộp Combo. |

1. **Sự kiện**

|  |  |
| --- | --- |
| **Click** | Đƣợc kích hoạt khi ngƣời dùng kích chuột vào hộp Combo. |
| **DoubleClick** | Đƣợc kích hoạt khi ngƣời dùng kích đúp chuột vào hộp Combo. |
| **Enter** | Đƣợc kích hoạt khi ngƣời dùng chuyển tiêu điểm tới hộp Combo. |
| **Leave** | Đƣợc kích hoạt khi hộp Combo mất tiêu điểm. |
| **SelectedIndex\_ Changed** | Đƣợc kích hoạt khi ngƣời dùng thay đổi trạng thái lựa chọn các dòng dữ liệu trong hộp văn bản. |
| **TextChanged** | Đƣợc kích hoạt khi ngƣời dùng nhập, sửa, xóa dữ liệu tại vùng văn bản của hộp Combo hoặc khi ta 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 đƣợc gọi ngay sau khi ngƣời dùng nhấp mũi tên để thả hộp danh sách xuống (phím tắt Alt+ ). Vì thế sự kiện này chủ yếu đƣợc sử dụng để nhập dữ liệu cho các phần tử của hộp Combo. |

1. **Phương thức**

**Add**: dùng để bổ sung một phần tử cho hộp Combo trong thời gian thi hành và thƣờng đƣợc viết trong thủ tục Form\_Load. Cú pháp của phƣơng thức này là:

#### ComboName.Items.Add(Item)

Trong đó ComboName là tên của hộp Combo, Item là nội dung của phần tử ta muốn thêm vào hộp Combo. Ví dụ, bổ sung phần tử có giá trị “Ha Noi” vào hộp Combo cboQue ta thực hiện nhƣ sau:

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 của phƣơng thức này là:

#### ComboName.Items.Remove(Item)

Ví dụ, xóa phần tử có giá trị “Ha Noi” trong hộp Combo cboQue ta viết nhƣ sau:

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 của phƣơng thức này là:

#### ComboName.Items.RemoveAt(Index)

Ví dụ, xóa phần tử ở vị trí 1 trong hộp Combo cboQue ta viết nhƣ sau: cboQue.Items.RemoveAt(1)

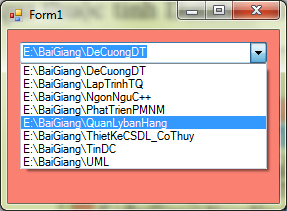
**Clear:** dùng để loại bỏ tất cả các phần tử của hộp Combo trong thời gian thi hành.

Cú pháp của phƣơng thức này là:

#### ComboName.Items.Clear()

Ví dụ, xóa tất cả các phần tử trong hộp Combo cboQue ta viết nhƣ sau: cboQue.Items.Clear()

### Bài tập 7

Lấy danh sách các thƣ mục có trong thƣ mục “E:\Baigiang” lƣu vào hộp Combo thông qua thuộc tính DataSource.

**Hướng dẫn**

Vào Microsoft Visual Studio 2010 tạo một dự án mới, đặt một hộp combo

*cboThumuc* vào Form1.

***Viết code:*** Mở sự kiện Load của Form1 ra và viết vào sự kiện đó mã lệnh sau:

private void Form1\_Load(object sender, EventArgs e)

{

String[] forder;

forder =System.IO.Directory.GetDirectories("E:\\BaiGiang"); cboThuMuc.DataSource =forder ;

}

* + 1. ***Điều khiển CheckedListBox***

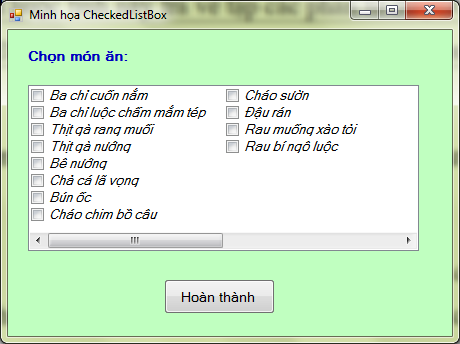
Điều khiển CheckedListBox  cho phép lƣu trữ và hiển thị các mục dữ liệu theo dòng và có một hộp CheckBox ở đầu dòng.

Điều khiển CheckedListBox có tiếp đầu ngữ là clb và có các thuộc tính, sự kiện, phƣơng thức tƣơng tự nhƣ điều khiển ListBox. Ngoài ra nó có thêm một số thuộc tính và sự kiện khác nhƣ sau:

|  |  |
| --- | --- |
| **CheckedItems** | Thuộc tính này trả về tập các phần tử đƣợc Check. |
| **ItemCheck** | Đƣợc kích hoạt khi ngƣời dùng kích đúp chuột vào một lựa chọn. |

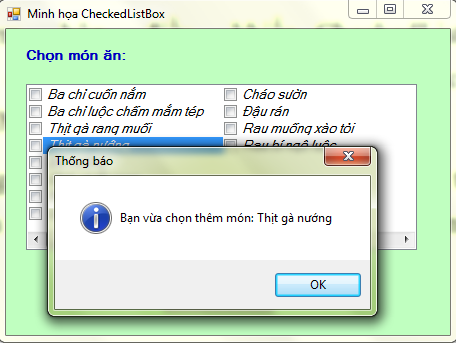
### Bài tập 8:

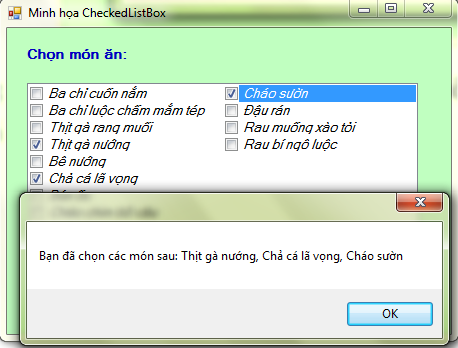
Tạo một chƣơng trình cho phép chọn các món ăn nhƣ sau:



Đặt lên form hộp điều khiển CheckedListBox *clbMonAn* và nút lệnh *btnHoanThanh*. Viết chƣơng trình hiển thị danh sách các mó n ăn đƣợc chia thành 2 cột vào trong hộp *clbMonAn*.

Khi ngƣời dùng đánh dấu vào hộp CheckBox của từng phần tử, sẽ xuất hiện hộp thông báo tên món ăn tƣơng ứng với phần tử đó, nhƣ hình minh họa sau:



Khi ngƣời dùng kích chọn nút *Hoàn thành* sẽ xuất hiện hộp thoại thông báo tên tất cả các món ăn ngƣời dùng đã chọn.

### Hướng dẫn

Mở sự kiện Load của Form1 ra và viết mã lệnh nhƣ sau:

private void Form1\_Load(object sender, EventArgs e)

{

//Cho phép hiển thị trên nhiều cột clbMonAn.MultiColumn = true;

//Cho phép hiển thị trên 2 cột clbMonAn.ColumnWidth = clbMonAn.Width / 2;

//Thêm các phần tử vào CheckedListBox clbMonAn.Items.Add("Ba chỉ cuốn nắm"); clbMonAn.Items.Add("Ba chỉ luộc chấm mắm tép"); clbMonAn.Items.Add("Thịt gà rang muối"); clbMonAn.Items.Add("Thịt gà nướng"); clbMonAn.Items.Add("Bê nướng"); clbMonAn.Items.Add("Chả cá lã vọng"); clbMonAn.Items.Add("Bún ốc"); clbMonAn.Items.Add("Cháo chim bồ câu"); clbMonAn.Items.Add("Cháo sườn"); clbMonAn.Items.Add("Đậu rán"); clbMonAn.Items.Add("Rau muống xào tỏi"); clbMonAn.Items.Add("Rau bí ngô luộc");

}

Mở sự kiện ItemCheck của CheckedListBox *clbMonAn* và viết lệnh:

private void clbMonAn\_ItemCheck(object sender, ItemCheckEventArgs e)

{

if (e.NewValue == CheckState.Checked)

MessageBox.Show("Bạn vừa chọn thêm món: "+clbMonAn.SelectedItem, "Thông báo",MessageBoxButtons.OK,MessageBoxIcon.Information);

}

Mở sự kiện Click của nút lệnh btnHoanThanh ra và viết mã lệnh nhƣ sau:

private void btnHoanThanh\_Click(object sender, EventArgs e)

{

string str = "";

foreach (string item in clbMonAn.CheckedItems) str = str + item + ", ";

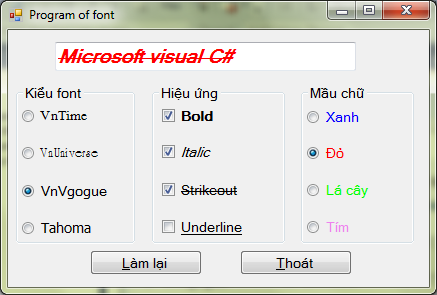
//Xóa dấu phẩy và dấu cách thừa cuối chuỗi str = str.Remove(str.Length - 2, 2);

MessageBox.Show("Bạn đã chọn các món sau: "+ str);

}

### Bài tập 9:

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



Gợi ý: Để thay đổi kiểu Font chữ cho điều khiển ta thực hiện theo cú pháp sau:

*<TênĐiềukhiển>.Font = New Font("TênFont", Cỡchữ)*

Để thay đổi hiệu ứng của font chữ ta thực hiện theo cú pháp sau:

*<TênĐiềukhiển>.Font = New Font(<TênĐiềukhiển>.Font, FontStyle.HiệuỨng)*

Trong đó Hiệu ứng có thể nhận các giá trị: Bold, Regular, Italic, UnderLine, Strikeout. Nếu tạo font với nhiều hiệu ứng thì ta dùng toán tử OR để kết hợp chúng với nhau.

Để thay đổi mầu chữ ta thực hiện theo cú pháp sau:

*<TênĐiềukhiển>.ForeColor = Color.Màu*

### Bài tập 10:

Lập chƣơng trình ghép tên nƣớc và tên thành phố theo giao diện và yêu cầu dƣới đây:



Viết phƣơng thức **EmptyOption()** bỏ chọn tất cả các RadioButton tên thành phố.

Khi kích chọn vào một nƣớc, giả sử France thì xuất hiện dòng thông báo: “Hãy chọn thành phố cho France” và gọi phƣơng thức **EmptyOption.**

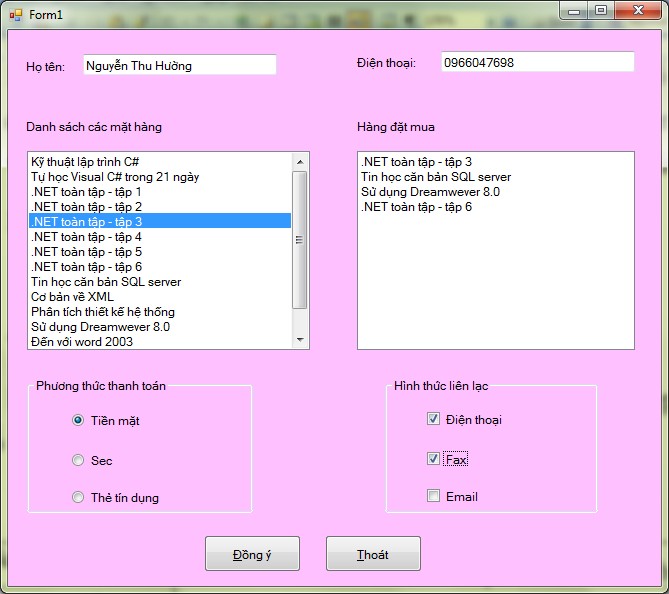
Khi kích chọn một thành phố, nếu đúng là thành phố của tên nƣớc đã chọn thì xuất hiện dòng thông báo, ví dụ: “Chúc mừng bạn, thủ đô của France là Paris”, ngƣợc lại thông báo, ví dụ: “Bạn sai rồi, thủ đô của France không phải là London”

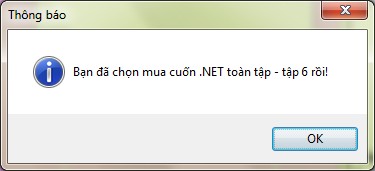
### Bài tập 11:

Lập chƣơng trình thực hiện các công việc theo giao diện và yêu cầu dƣới đây:

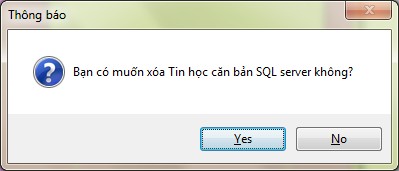
Chƣơng trình có một Form bán hàng trực tuyến, danh sách các mặt hàng đƣợc hiển thị sẵn trong hộp Listbox hoặc CheckedListBox “Danh sách các mặt hàng”.

Để mua hàng ngƣời dùng kích đúp vào mặt hàng cần mua trong “Danh sách các mặt hàng”, mặt hàng đƣợc chọn sẽ đƣợc hiển thị vào trong “Hàng đặt mua”.



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

Ngƣời dùng có thể xoá mặt hàng trong số các mặt hàng đã chọn bằng cách kích đúp vào mặt hàng cần xoá, trƣớc khi xoá phải hỏi lại ngƣời dùng có muốn xoá hay không?



Khi kích chuột vào nút “Đồng ý” kiểm tra ngƣời dùng phải nhập đầy đủ thông tin và hiện thông báo gồm các thông tin: Tên khách, Điện thoại, Danh sách các mặt hàng đã mua, Phƣơng thức thanh toán và Hình thức liên lạc.

**Chú ý:** Khi thực hiện chƣơng trình, để các dòng thông báo hay các kết quả đƣợc hiển thị trên nhiều dòng khác nhau thì khi viết mã lệnh ta có thể dùng hằng ký tự điều khiển *\n*

* + 1. ***Điều khiển NumericUpDown***

Điều khiển NumericUpDown  cho phép ngƣời dùng lựa chọn một giá trị số trong một khoảng giá trị với một bƣớc nhảy xác định.

* + - 1. **Thuộc tính**

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

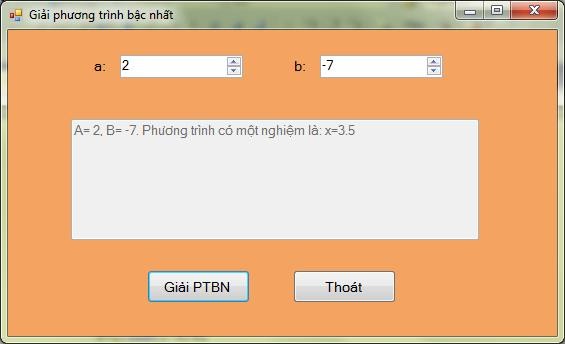
* + - 1. **Sự kiện**

|  |  |
| --- | --- |
| **ValueChanged** | Đƣợc kích hoạt khi ngƣời dùng thay đổi giá trị của điều khiển. |

### Bài tập 12:

Dùng điều khiển NumericUpDown để giải phƣơng trình bậc nhất:

ax2 + b = 0 (a, b có giá trị nguyên trong đoạn [-100, 100])



### Hướng dẫn:

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

* Hai điều khiển NumericUpDown có tên *nudA, nudB*. Các thuộc tính: *Maximum = 100, Minimum = -100, Increment = 1.*
* Một hộp Textbox có tên *txtKetQua*. Các thuộc tính: *Enable = False.* Hộp kết quả có thuộc tính *Multiline = True.*

Hai nút lệnh *btnGiai, btnThoat* và các nhãn Label…

### Viết mã lệnh:

Mở sự kiện Click của nút lệnh btnGiai viết mã lệnh nhƣ sau:

private void btnGiai\_Click(object sender, EventArgs e)

{

int a, b; float x;

string str="A= " +nudA.Value.ToString() +", B= "+nudB.Value.ToString();

a = Convert.ToInt16(nudA.Value); b = Convert.ToInt16(nudB.Value); if (a == 0)

{

}

else

{

}

if (b == 0)

str = str+ ". Phương trình vô số nghiệm"; else

str = str+ ". Phương trình vô nghiệm";

x=(float)-b/a;

str = str+ ". Phương trình có một nghiệm là: x="+x.ToString();

txtKetQua.Text=str;

}

* + 1. ***Thanh cuộn HscrollBar và VscrollBar***

Thanh cuộn ngang HScrollBar  và thanh cuộn dọc VScrollBar  cho phép ngƣời dùng lựa chọn một giá trị số trong một khoảng giá trị xác định.

* + - 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 ngƣời dùng nhấn chuột trên thanh cuộn. |
| **SmallChange** | Chỉ ra mức độ thay đổi của thuộc tính Value khi ngƣời dùng nhấn chuột vào các mũi tên trên thanh cuộn (giá trị mặc định = 1). <=LargeChange |

* + - 1. **Sự kiện**

|  |  |
| --- | --- |
| **ValueChanged** | Đƣợc kích hoạt khi ngƣời dùng thay đổi giá trị của thanh cuộn. |
| **Scroll** | Xảy ra khi ngƣời dùng kéo rê chuột hoặc kích chuột vào các mũi tên trên thanh cuộn. |

### Bài tập 13:

Dùng thanh cuộn HScrollBar để thay đổi mầu nền cho TextBox nhƣ giao diện sau:



* + 1. ***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 đƣa điều khiển Timer vào Form nó không xuất hiện trên Form mà xuất hiện nhƣ một biểu tƣợng ở trên một khay đặt ở cuối cửa sổ thiết kế. Khi chạy chƣơng trình điều khiển Timer cũng không xuất hiện.

* + - 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 đƣợc tính bằng mili giây và có giá trị >0 (1s=1000ms) |
| **Enabled** | Enabled = True: cho phép điều khiển Timer hoạt động, Enabled = False: không cho phép điều khiển Timer hoạt động. |

* + - 1. **Sự kiện**

Sự kiện này đƣợc kích hoạt sau mỗi chu kỳ Interval.

**Tick**

* + - 1. **Phương thức**

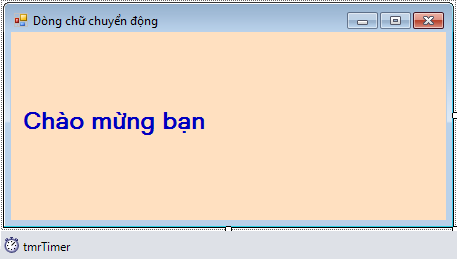
**Start**: kích hoạt điều khiển Timer, phƣơng thức này tƣơng đƣơng với thuộc tính Enabled = True. Cú pháp của phƣơng thức này là:

#### TimerName.Start()

**Stop**: dừng điều khiển Timer, phƣơng thức này tƣơng đƣơng với thuộc tính Enabled

= False. Cú pháp của phƣơng thức này là: ***TimerName****.****Stop(****)*

### Bài tập 14:

Viết chƣơng trình tạo dòng chữ “Chào mừng bạn” chuyển động từ trái qua phải màn hình, khi gặp mép màn hình thì chuyển động ngƣợc lại từ phải qua trái. Cứ lặp lại nhƣ vậy cho đến khi kết thúc chƣơng trình.

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

|  |  |  |
| --- | --- | --- |
| Đ**i**ề**u khi**ể**n** | **Name** | **Text** |
| Form1 | frmVDTimer | Dòng chữ chuyển động |
| Label1 | lblMove | Chào mừng bạn |
| Timer | tmrTimer | Enabled = True và Interval =  200 |

### Viết Code:

Đặt biến toàn cục:

bool hp; // hp=true chuyển động sang phải, hp=false chuyển động sang trái.

Mở sự kiện Load của Form frmVDTimer viết mã lệnh sau:

private void frmVDTimer\_Load(object sender, EventArgs e)

{

hp = true; tmrTimer.Start();

}

Mở sự kiện Tick của điều khiển Timer tmrTimer viết mã lệnh sau:

private void tmrTimer\_Tick(object sender, EventArgs e)

{

if (hp)

{

if ((lblMove.Left + lblMove.Width) < this.Width) lblMove.Left = lblMove.Left + 10;

else

hp = false;

}

else

{

if (lblMove.Left > 0)

lblMove.Left = lblMove.Left - 10; else

hp = true;

}

}

### Bài tập 15:

Viết chƣơng trình đồng hồ đếm ngƣợc theo giao diện sau:



### Yêu cầu:

+ Nút chạy: sẽ kích hoạt đồng hồ bắt đầu đếm.

+ Nút tạm dừng: sẽ tạm dừng đếm lại. Nếu muốn đếm tiếp kích vào nút chạy.

+ Nút kết thúc: Sẽ làm cho đồng hồ về 00:00.

* + 1. ***Điều khiển RichTextBox***

Điều khiển RichTextBox  cho phép hiển thị đƣợc văn bản có định dạng nhƣ một tệp Word, tức là có thể hiển thị màu sắc, loại Font, cỡ chữ… khác nhau trong từng đoạn văn bản.

Điều khiển RichTextBox có thể tạo và lƣu trữ các tệp văn bản có phần mở rộng .rtf. Để tạo một tệp có phần mở rộng .rtf ta có thể sử dụng trình soạn thảo Word và lƣu tệp – Save As theo định dạng *Rich TextFormat (\*.rtf)*

Tên điều khiển RichTextBox có tiếp đầu ngữ **rtb** và nó có 2 phƣơng 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* ta thực hiện nhƣ sau:

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

**SaveFile**: lƣu nội dung trong hộp RichTextBox vào một tệp có phần mở rộng .rtf. Ví dụ lƣu nội dung của hộp RichTextBox *rtbBai1* vào tệp *Bai2.rtf* trong ổ D ta thực hiện nhƣ sau: *(n*ế*u t*ệ*p Bai2.rtf ch*ƣ*a t*ồ*n t*ạ*i thì s*ẽ đƣợ*c t*ạ*o m*ớ*i)*

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

# CHƯƠNG 4: CÁC HỘP HỘI 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 lƣu 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 tƣợng .NET trong thƣ viện **System.IO** bao gồm 5 lớp tƣơng ứng với 5 hộp hội thoại cơ bản nhƣ sau:

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

## Hộp hội thoại Open File

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

|  |  |
| --- | --- |
| **Thuộc tính** | **Chức năng** |
| Name | Tên điều khiển OpenDialog. Tiếp đầu ngữ của tên điều khiển là odlg |
| FileName | Cung cấp tên và đƣờng 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 đƣợc 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 thƣ mục mặc định cho hộp hội thoại khi vừa đƣợc gọi. |
| Multiselect | Multiselect = True: cho phép ngƣời dùng chọn đồng thời nhiều file. |
| FileNames | Cung cấp tên và đƣờng dẫn của các tập tin đã chọn. |
| Title | Xác định tiêu đề của hộp hội thoại. |
| OpenFile | Mở nội dung File đã đƣợc chọn (ReadOnly). |

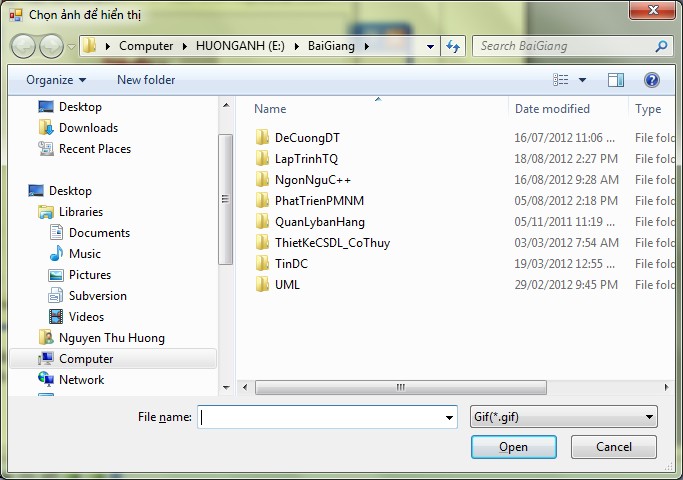
### Bài tập 16:

Viết chƣơng trình cho phép dùng hộp thoại OpenDialog chọn hình ảnh để hiển thị.

Vào Microsoft Visual Studio 2010 tạo dự án mới có tên là **OpenDialog** và thiết lập thuộc tính của các điều khiển nhƣ sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Đ**i**ề**u khi**ể**n** | **Name** | **Text** |  |
| Form1 | frmPicture | Hiển thị hình ảnh |
| PictureBox | picAnh |  |
| Button1 | btnOpen | Open |

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



Mở đƣờng dẫn chứa ảnh cần hiển thị, kích chọn ảnh sau đó kích chọn **Open**, ảnh sẽ đƣợc hiển thị lên điều khiển PictureBox trên form nhƣ sau:

Trong trƣờng hợp ngƣời dùng chọn **Cancel** sẽ xuất hiện thông báo: “You clicked Cancel!”.

### Hướng dẫn:

Mở sự kiện Click của nút Open rồi viết lệnh 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 = "E:\\BaiGiang"; dlgOpen.FilterIndex = 2;

dlgOpen.Title = "Chọn ảnh để hiển thị";

if (dlgOpen.ShowDialog ()== DialogResult.OK) picAnh.Image = Image.FromFile(dlgOpen.FileName);

else

MessageBox.Show("You clicked Cancel" , "Open Dialog",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

## Hộp thoại SaveFile và luồng FileStream

* + 1. ***Hộp thoại SaveFile***

Các thuộc tính và phƣơng thức quan trọng của hộp hội thoại SaveFile:

|  |  |
| --- | --- |
| **Thu**ộ**c tính** | **Ch**ứ**c n**ă**ng** |
| FileName | Cung cấp tên và đƣờng 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) |
| InitialDirectory | Xác định thƣ mục mặc định cho hộp hội thoại khi vừa đƣợc gọi. |
| AddExtension | = True tự động thêm phần mở rộng hiện hành vào tên tệp ngƣời dùng chọn nếu ngƣời dùng không chỉ rõ phần mở rộng của tên tệp. |
| DefaultExt | Cung cấp phần mở rộng mặc định cho tên tệp nếu ngƣời dùng không chỉ rõ phần mở rộng của tên tệp, ví dụ: “.doc” |
| Title | Xác định tiêu đề của hộp hội thoại. |

* + 1. ***Luồng FileStream***

Với một file dữ liệu có kiểu bất kỳ đều đƣợc coi nhƣ 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.

Để làm việc với luồng dữ liệu ta sẽ làm việc với namespace System.IO bằng cách dùng từ khóa using:

using System.IO;

#### 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”)*

Đóng luồng:

*Tenluong.Close()*

#### 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 đƣợc, nếu đến cuối file thì trả về Nothing.

*Noidung = Tenluong.ReadLine()*

Đọc tất cả dữ liệu của file lƣu vào một biến:

*Noidung = Tenluong.ReadToEnd()*

Đóng luồng:

*Tenluong.Close()*

**Bài tập 17:**

Viết chƣơng trình cho phép lƣu nội dung của một hộp TextBox vào một file trên máy

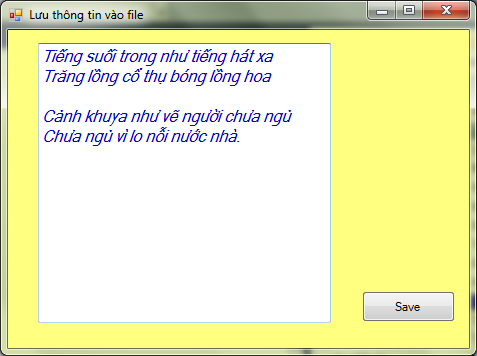
tính.

Vào Microsoft Visual Studio 2010 tạo dự án mới có tên là **SaveDialog** và thiết lập

thuộc tính của các điều khiển nhƣ sau:

|  |  |  |
| --- | --- | --- |
| Đ**i**ề**u khi**ể**n** | **Name** | **Text** |
| Form1 | frmSave | Lƣu thông tin vào File |
| TextBox (MultiLine = True) | txtSave |  |
| Button1 | btnSave | Save |

Khi chạy chƣơng trình gõ nội dung cần lƣu vào TextBox



Kích chọn nút *Save* xuất hiện hộp hội thoại *Chọn file để lưu* với đƣờng dẫn mặc định là *E:\Baigiang*. Gõ tên tệp cần lƣu dữ liệu vào ô *FilName*, ví dụ: *Canh\_Khuya.doc* và chọn *Save*.

Kết quả trong thƣ mục *E:\Baigiang* xuất hiện tệp *Canh\_Khuya.doc* chứa nội dung của hộp Textbox *txtSave*.

**Hướng dẫn:**

Khai báo thêm namespace System.IO nhƣ sau:

using System.IO;

Mở sự kiện Click của nút btnSave và viết lệnh nhƣ 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 = "E:\\Baigiang" ; dlgSave.FilterIndex = 2;

dlgSave.Title = "Chọn File để lưu"; dlgSave.AddExtension = true ; dlgSave.DefaultExt = ".doc";

if (dlgSave.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

StreamWriter file = new StreamWriter(dlgSave.FileName); try

{

file.Write(txtSave.Text); MessageBox.Show("Thành công");

}

catch

{

MessageBox.Show("Lỗi ghi file");

}

file.Close();

}

else

MessageBox.Show("you clicked Cancel");

}

## Hộp thoại Color

Có chức năng hiển thị bảng màu hiện hành cho phép ngƣời sử dụng chọn một màu bất kỳ để tô màu chữ, màu nền.

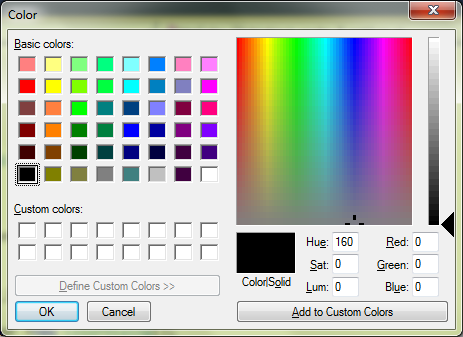
Các thuộc tính quan trọng của hộp thoại Color:

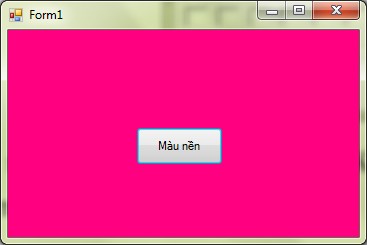
|  |  |
| --- | --- |
| **Thuộc tính** | **Chức năng** |
| Color | Trả về màu đƣợc chọn trong hộp thoại Color. |
| FullOpen | Hiển thị toàn bộ hộp thoại Color. |
| SolidColorOnly | Không hiển thị phần Define Custom Colors. |

### Bài tập 18:

Viết chƣơng trình cho phép chọn màu nền cho form.

Chạy chƣơng trình, kích chọn nút lệnh *Màu n*ề*n* sẽ xuất hiện hộp thoại sau:



Chọn một màu bất kỳ rồi bấm OK, khi đó màu đƣợc chọn sẽ đƣợc gán làm màu nền cho form, ví dụ:

### Hướng dẫn:

Vào Microsoft Visual Studio 2010 tạo dự án mới có tên là **ColorDialog** và kéo vào Form một điều khiển button đặt tên là btnColor.

Mở cửa sổ code của sự kiện Click của nút btnColor viết lệnh sau:

private void btnColor\_Click(object sender, EventArgs e)

{

ColorDialog dlgColor =new ColorDialog(); dlgColor.FullOpen = true;

if (dlgColor.ShowDialog() == System.Windows.Forms.DialogResult.OK) this.BackColor = dlgColor.Color;

else

MessageBox.Show("You clicked Cancel" , "Color Dialog", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

## Hộp thoại Font

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

Các thuộc tính quan trọng của hộp thoại Font:

|  |  |
| --- | --- |
| **Thu**ộ**c tính** | **Ch**ứ**c n**ă**ng** |
| Font | Trả về kiểu Font chữ đƣợc chọn trong hộp thoại Font. |
| ShowColor | = True: cho phép hiển thị hộp thoại Color. |
| Color | Trả về màu đƣợc chọn trong hộp thoại Font. |

### Bài tập 19:

Viết chƣơng trình cho phép chọn kiểu font chữ khác nhau cho TextBox.

Tạo dự án mới với các điều khiển có thuộc tính Name, Text nhƣ sau: *Form1(frmFont, ”Hộp thoại chọn Font”), TextBox1(txtFont, “Hà N*ộ*i’), Button1(btnFont, “Ch*ọ*n Font”)*

Mở sự kiện Click của nút lệnh btnFont và viết lệnh nhƣ sau:

private void btnFont\_Click(object sender, EventArgs e)

{

FontDialog dlgFont = new FontDialog(); dlgFont.ShowColor = true;

if (dlgFont.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

txtFont.Font = dlgFont.Font; txtFont.ForeColor = dlgFont.Color;

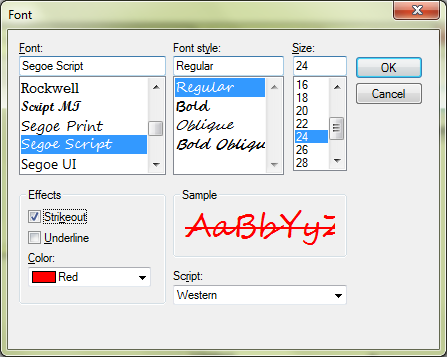
}

else

MessageBox.Show("You clicked Cancel" , "Font Dialog", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

Chạy chƣơng trình, kích chọn nút lệnh *Ch*ọ*n Font* sẽ xuất hiện hộp thoại Font:



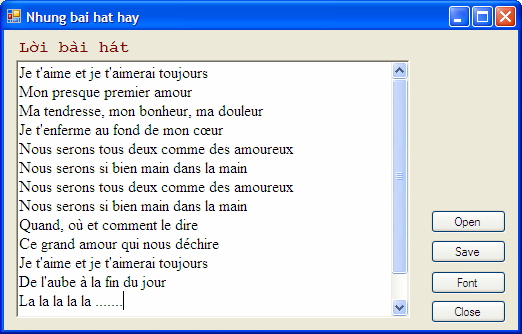
Chọn giá trị cho các thuộc tính rồi bấm OK ta đƣợc kết quả nhƣ sau:



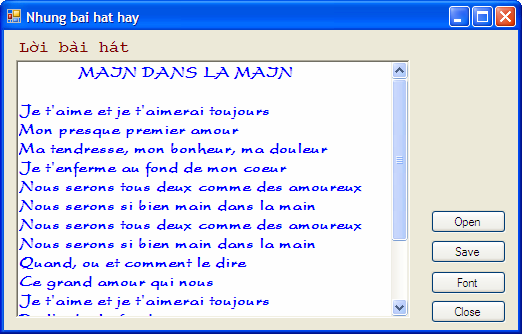
### Bài tập 20:

Viết chƣơng trình cho phép đọc nội dung một tệp *.rtf* trong máy lƣu vào hộp RichTextBox và có thể sửa nội dung, thay đổi Font chữ, màu chữ… rồi lƣu lại.

Kích chọn nút *Open* xuất hiện hộp thoại *OpenFile*, chọn tệp *Main dans la main.rtf* ở thƣ mục *E:\Baigiang\ Nhac\_Loi* lƣu vào hộp RichTextBox



Kích chọn nút *Font* xuất hiện hộp thoại *Font*, chọn mầu chữ và kiểu chữ nhƣ sau:

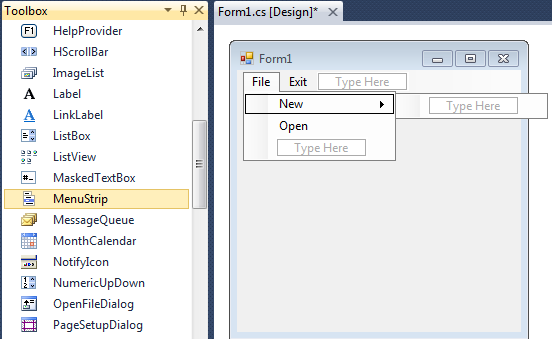


Kích chọn nút *Save* xuất hiện hộp thoại *SaveFile*, lƣu nội dung hộp RichTextBox vào tệp *Main dans la main.rtf*

# CHƯƠNG 5: MENU VÀ CÁC ĐỒ ÁN NHIỀU BIỂU MẪU

Trong chƣơng này, chúng ta sẽ học cách sử dụng điều khiển MenuStrip để tạo các ứng dụng có menu (thực đơn). Tiếp đó ta sẽ học cách tạo ra menu dạng Popup. Cuối cùng ta sẽ làm quen với các dạng đồ án (project) có nhiều biểu mẫu (form).

## Điều khiển MenuStrip

Điều khiển MenuStrip giúp ngƣời lập trình có thể thiết kế thanh menu trên Form. Khi đƣa 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.

### 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 tƣợng 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 & trƣớc một chữ cái trong thuộc tính Text thì khi chạy chƣơng trình ngƣời 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 đƣợc xác lập là một dấu trừ (-) chƣơng trình sẽ hiển thị một đƣờng thẳng ngăn cách giữa các khoản mục menu. |  |
| Visible | = Flase: các mục menu sẽ không đƣợc hiển thị. |
| ToolTipText | Tạo dòng mách nƣớc cho các mục menu. |

* + 1. **Sự kiện**

|  |  |
| --- | --- |
| Click | Đƣợc kích hoạt khi ngƣời dùng kích chuột để chọn một mục menu. |

**Chú ý:**

*Chèn m*ụ*c 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 đƣợc chèn vào trên mục menu đang đƣợc chọn.

*Xóa m*ụ*c menu*: kích chuột phải tại mục menu muốn xóa, chọn *Delete*

### Bài tập 21:

Tạo một menu với giao diện nhƣ sau:



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

### Hướng dẫn:

Vào Microsoft Visual Studio 2005 tạo dự án mới có tên là **Menu** và thiết lập thuộc tính của các điều khiển nhƣ sau:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Đ**i**ề**u khi**ể**n** | **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 | Lƣu tệp |
| mnu1 | - |  |  |  |
| mnuExit | E&xit |  |  |  |
| … |  |  |  |  |

**Vi**ế**t Code:** Lần lƣợt mở sự kiện click của các menu con và viết mã lệnh. Ví dụ với sự kiện click của mnuNew nhƣ sau:

private void mnuNew\_Click(object sender, EventArgs e)

{

MessageBox.Show("Bạn đã chọn mục menu New",

"Thông báo",MessageBoxButtons.OK,MessageBoxIcon.Information);

}

Các mục menu khác làm tƣơng tự

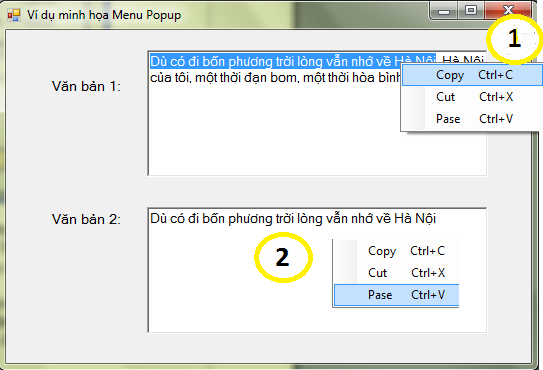
## Popup menu – ContextMenuStrip

Điều khiển ContextMenuStrip  cho phép tạo các menu dạng Popup đƣợc gắn với một điều khiển nào đó trên Form, menu này chỉ xuất hiện khi ngƣời dùng kích chuột phải tại điều khiển gắn menu đó.

Điều khiển ContextMenuStrip có tiếp đầu ngữ là **cmnu** và có các thuộc tính và phƣơng thức tƣơng tự nhƣ điều khiển MenuStrip.

### Bài tập 22:

Viết chƣơng trình tạo menu Popup có giao diện và các chức năng nhƣ sau:



Bôi đen các dòng văn bản trong hộp *V*ă*n b*ả*n 1* và kích chuột phải tại vị trí bất kỳ trong hộp *V*ă*n b*ả*n 1*, xuất hiện menu gồm 3 mục lựa chọn: ***Copy, Cut, Paste***. Kích chọn mục *Copy* để lƣu các dòng văn bản đƣợc bôi đen vào bộ nhớ đệm ClipBoard. Kích chuột phải tại vị trí bất kỳ trong hộp *V*ă*n b*ả*n 2* chọn *Paste* để gán nội dung dòng văn bản trong ClipBoard vào hộp *V*ă*n b*ả*n 2*.

### Hướng dẫn

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 nhƣ sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Đ**i**ề**u khi**ể**n** | **Name** | **Text** | **Context MenuStrip** | **Shortcut Keys** |
| ContextMenu Strip | cmnuEdit |  |  |  |
| cmnuCopy | Copy |  | Ctrl + C |
| cmnuCut | Cut |  | Ctrl + X |
| cmnuPaste | Paste |  | Ctrl + V |
| Label1 |  | Văn bản 1 |  |  |
| Label2 |  | Văn bản 2 |  |  |
| RichTextBox1 | rtbVanban1 |  | cmnuEdit |  |
| RichTextBox2 | rtbVanban2 |  | cmnuEdit |  |

Mở sự kiện Click của 2 mnu cmnuCopy và cmnuPase viết mã lệnh nhƣ sau:

private void cmnuCopy\_Click(object sender, EventArgs e)

{

Clipboard.SetText(rtbVanBan1.SelectedText);

}

private void cmnuPase\_Click(object sender, EventArgs e)

{

rtbVanBan2.Text = Clipboard.GetText();

}

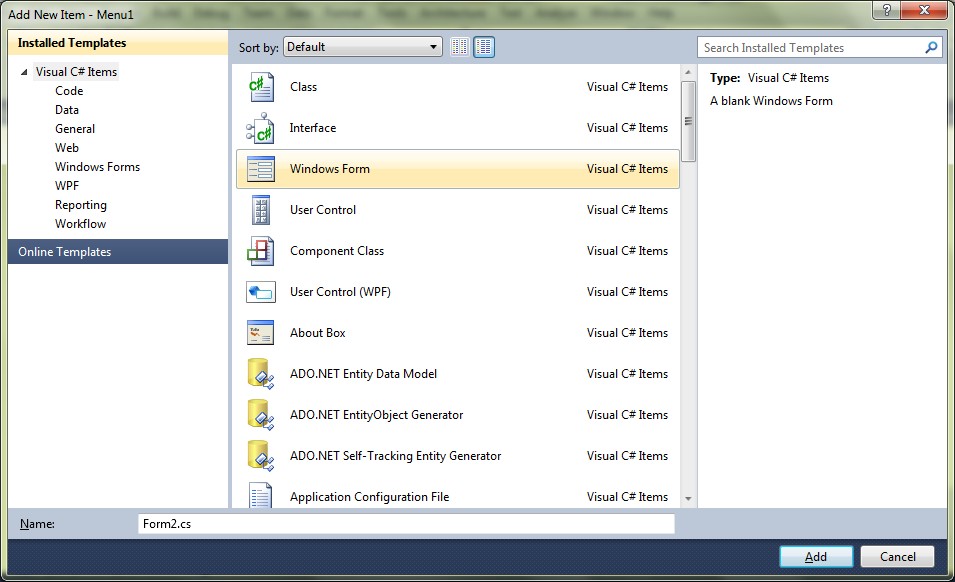
Các mục menu khác viết tƣơng tự.

## Đồ án nhiều biểu mẫu

Khi ứng dụng trở lên phức tạp, chƣơng 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.

* + 1. ***Bổ sung biểu mẫu***

Để bổ sung thêm các biểu mẫu - Form cho ứng dụng đang thiết kế, chọn menu *Project/ Add New Item…* hoặc kích chuột vào biểu tƣợng  trên thanh công cụ hoặc bấm phím tắt *Ctrl+Shift+A* xuất hiện cửa sổ Add New Item:



C# cung cấp vài biểu mẫu đƣợc xây dựng sẵn để có thể bổ sung vào dự án, ví dụ biểu mẫu cơ bản „Windows Form‟ nhƣ ta đã biết, hoặc „Uer control‟- dùng để thiết kế các điều khiển riêng cho ngƣời dùng…

Chọn loại biểu mẫu cần bổ sung rồi bấm Add, ví dụ chọn Windows Form khi đó một biểu mẫu mới sẽ đƣợc chèn vào ứng dụng.

Ngoài ra C# còn cho phép bổ sung các form đã đƣợc 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 ngƣời, mỗi ngƣời làm một phần sau đó ghép lại với nhau tạo thành ứng dụng hoàn chỉnh.

### Bài tập 23:

Vào Microsoft Visual Studio 2010 tạo dự án mới có tên là ***BaiTapToan.*** Đổi tên cho Form1 thành ***frmMain*** và tiêu đề là ***Bài tập toán lớp 2*.**

Bổ sung thêm 2 Windows Form mới có tên là ***frmPhepCong1*** và ***frmSoSanh1*** với tiêu đề lần lƣợt là ***Bài tập về phép cộng 1*** và ***Bài tập so sánh 1***.

Lúc này dự án có 03 Form là: frmMain, frmPhepCong1 và frmSoSanh1.

* + 1. ***Biểu mẫu khởi động***

Khi chạy một ứng dụng, biểu mẫu đƣợc thực hiện đầu tiên gọi là biểu mẫu khởi động. Biểu mẫu Startup này mặc định là biểu mẫu đầu tiên đƣợc xây dựng trong ứng dụng khi thiết kế giao diện, ví dụ biểu mẫu frmMain trong ứng dụng *BaiTapToan*.

Ta có thể thay đổi mặc định này để có thể chọn một biểu mẫu bất kỳ làm biểu mẫu khởi động khi chạy chƣơng trình. Ví dụ muốn form *frmPhepCong1* đƣợc thực hiện đầu tiên, ta thực hiện nhƣ sau:

+ Mở file *Program.cs*

*+* Thay đổi tên form bắt đầu chạy trong dòng lệnh sau:

Application.Run(new frmMain());

Thành nhƣ sau:

Application.Run(new frmPhepCong());

* + 1. ***Mở biểu mẫu***

Để mở một biểu mẫu ta có phải khởi tạo đối tƣơng form cho Form cần mở nhƣ sau:

+ Ví dụ cần mở form frmPhepCong ta khởi tạo nhƣ sau: frmPhepCong phepCong=new frmPhepCong(); Sau đó ta thể thực hiện theo 2 cách sau:

**Cách 1:** cú pháp **phepCong.Show()**

Tải biểu mẫu vào bộ nhớ sau đó đƣa biểu mẫu lên phía trên các biểu mẫu khác (nếu có), phƣơng thức này cho phép ngƣời sử dụng có thể tƣơng tác đƣợc với các biểu mẫu nằm phía dƣới.

**Cách 2:** cú pháp **phepCong.ShowDialog()**

Tƣơng tự nhƣ cách 1 nhƣng ngƣời sử dụng chỉ có thể tƣơng tác đƣợc với biểu mẫu hiện hành mà không thể tƣơng tác với các biểu mẫu nằm phía dƣới nó.

* + 1. ***Đóng biểu mẫu***

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

Cách 1: đóng form đang làm việc **this.Close();**

Cách 2: đóng một form bất kỳ, ví dụ form frmPhepCong đƣợc khởi tạo nhƣ trên:

### phepCong.Close();

* + 1. ***Xóa biểu mẫu***

Kích chuột phải tại Form cần xóa trong cửa sổ *Solution Explorer*, chọn *Delete* kết quả Form sẽ bị xóa ra khỏi dự án.

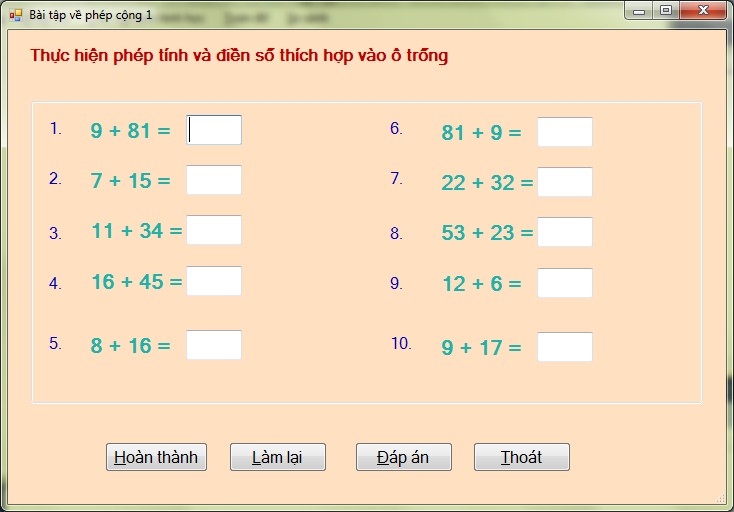
### Bài tập 24:

Tiếp theo bài tập 23, tạo MenuStrip cho Form *frmMain* theo giao diện sau:



### Bài tập 25:

Tiếp bài tập 24, tạo form frmPhepCong1 với giao diện nhƣ sau:



### Yêu cầu:

+ Khi ngƣời dùng chọn mục *Phép cộng 1* t r ê n m e n u Form *frmPhepConf1* xuất hiện nội dung bài tập cần điền số thích hợp vào trong hộp Textbox.

+ Chỉ cho phép gõ số nguyên vào các ô TextBox.

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

+ Khi chọn nút **Hoàn thành** chƣơng trình kiểm tra kết quả, câu nào đúng thì đổi mầu nền ở Textbox tƣơng ứng với câu trả lời sang mầu xanh, câu nào sai thì đổi Textbox có nền mầu đỏ. Và hiển thị điểm đạt đƣợc cho ngƣời dùng (mỗi câu trả lời đúng đƣợc 1 điểm).

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



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

+ Bấm nút ***Thoát*** để đóng form frmPhepCong1 trở về form chính frmMain.

### Bài tập 26:

Tiếp bài tập 25, tạo form frmSoSanh1 với giao diện nhƣ sau:



Khi chọn mục *So sánh/So sánh 1* thì Form *frmSoSanh1* xuất hiện nội dung các câu cần điền dấu >, < hoặc =.

Các nút *Hoàn thành, Làm l*ạ*i, Đáp án* và *Thoát* có chức năng tƣơng tự nhƣ trong form *frmPhepCong1*.

# CHƯƠNG 6: LẬP TRÌNH CƠ SỞ DỮ LIỆU

Lập trình với Cơ sở dữ liệu là một phần quan trọng trong hầu hết các ngôn ngữ lập trình. Trong chƣơng này chúng ta sẽ học cách sử dụng công nghệ ADO .NET (của .NET Framework) và ngôn ngữ lập trình C# để làm việc với Cơ sở dữ liệu SQL Server. Nội dung chính của chƣơng này bao gồm:

+ Cách tạo một DataBase SQL trong Visual Studio.

+ Các thành phần của ADO.NET

+ Cách làm việc với SQL Server thông qua ADO.NET

Trong các ví dụ minh họa của chƣơng này ta sẽ tạo ra một project Quản lý bán hàng.

## Giới thiệu về bài toán

* + 1. ***Lập trình cơ sở dữ liệu và bài toán quản lý***

Hiện nay, một trong những mảng phần mềm đƣợc ứng dụng rộng rãi nhất đó là phần mềm quản lý, ví dụ nhƣ: quản lý bán hàng, quản lý điểm, quản lý đăng ký học, quản lý công văn giấy tờ. Bài toán quản lý đã dần trở thành bài toán phổ biến, quen thuộc và nó có tầm quan trọng ảnh hƣởng lớn đến sự phát triển kinh tế của xã hội.

Bất kể một bài toán quản lý nào cũng cần có một cơ sở dữ liệu thích hợp để lƣu trữ quản lý thông tin của nó. Và một chƣơng trình quản lý đòi hỏi phải có những yêu cầu về kết nối và các thao tác trên CSDL.

Lập trình CSDL là một trong các thế mạnh của .NET cho phép dễ dàng tạo ra những phần mềm quản lý chuyên nghiệp, có thể kết nối tới CSDL của hầu hết các hệ quản trị dữ liệu nhƣ: MS Access, SQL server, MySQL server, Oracle, … Hơn thế nữa nó còn có thể kết nối đến các dạng CSDL khác nhƣ Excel, … để thêm mới, sửa, xóa, tìm kiếm và kết xuất dữ liệu với độ chính xác cao, chức năng phong phú và giao diện thân thiện.

Trong chƣơng trình này ta sẽ học về cách thao tác và làm việc với hệ quản trị CSDL SQL server cùng các tiện ích khác nhƣ xuất dữ liệu ra Excel.

Để học phần này máy tính của bạn học cần cài đặt các thành phần sau:

+ Bộ Visual Studio (bản 2005 hoặc bản 2008 hoặc bản 2010 – Cài bản đầy đủ). Khi cài đầy đủ thì chúng đã có sẵn cả SQL server rồi.

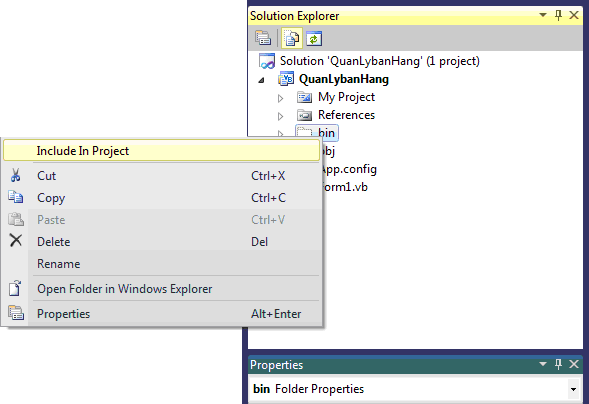
* + 1. ***Cách tổ chức các tài nguyên trong một dự án của bài toán quản lý***

Để dễ dàng quản lý tài nguyên của ứng dụng ta nên phân chia ứng dụng thành các thƣ mục nhỏ, mỗi một thƣ mục chứa các file cùng thực hiện một nhóm chức năng. Ví dụ thƣ mục ***Images*** dùng để chứa các file ảnh.

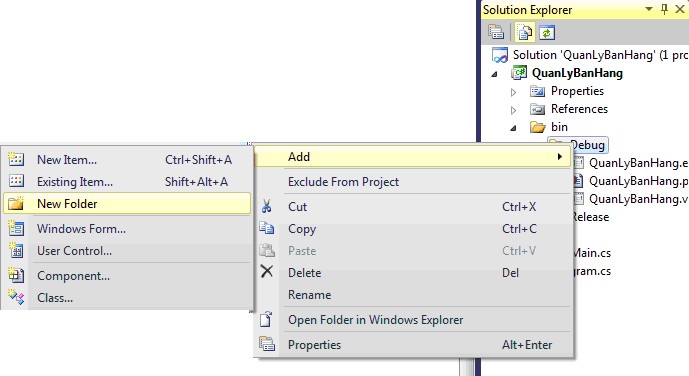
Để tổ chức tài nguyên cho một chƣơng trình quản lý, thông thƣờng ta làm nhƣ sau:

**+** Khởi động Microsoft Visual Studio 2010 tạo dự án mới (giả sử dự án mới có tên là ***QuanLyBanHang*** lƣu vào thƣ mục E:\). Trong dự án ta tạo một số thƣ mục để lƣu các nhóm đối tƣợng khác nhau.

+ Sau khi tạo xong dự án mới, trên cửa sổ Solution Explorer kích vào biểu tƣợng show all file  để thƣ mục **bin** và **ogj** hiện lên. Lúc này thƣ mục **bin** bị mờ đi ta kích chuột phải vào nó rồi chọn **Include In Project** nhƣ hình sau:



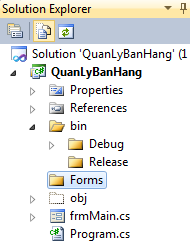
+ Bây giờ ta tạo thêm trong /**bin/Debug/** thƣ mục ***DataBase*** để chứa cơ sở dữ liệu cho ứng dụng. Bằng cách kích chuột phải vào **bin** chọn **Add** và chọn **new folder**, sau đó đổi tên thƣ mục là **DataBase**



+ Tƣơng tự ta tạo thƣ mục để chứa ảnh là **Images** trong **bin/Debug.**

+ Trong cửa sổ Solution Explorer kích chuột phải tại tên dự án chọn ***Add,*** chọn ***new folder,*** đổi tên ***Folder*** thành ***Forms.*** Thƣ mục này có tác dụng chứa các form trong chƣơng trình.

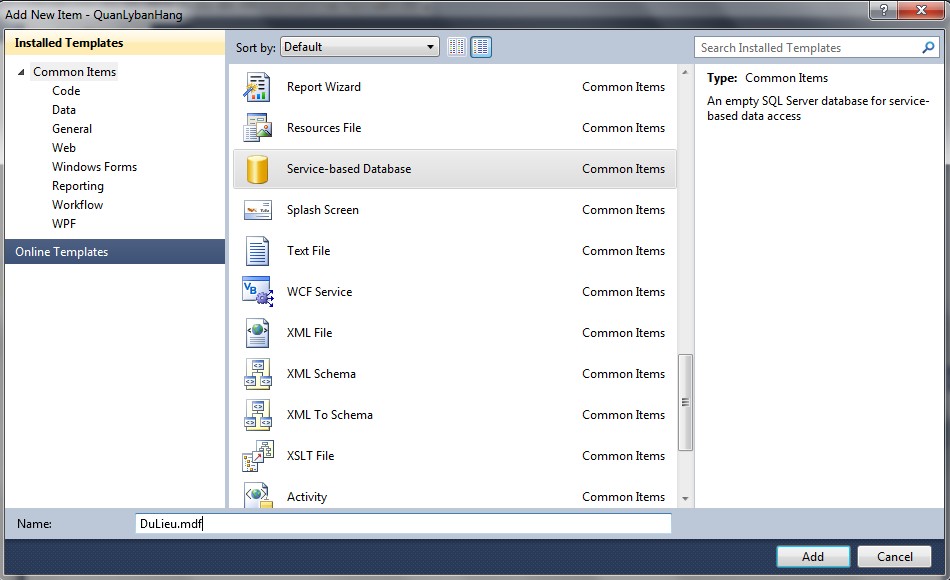
Cuối cùng cây thƣ mục chứa tài nguyên của ứng dụng nhƣ sau:



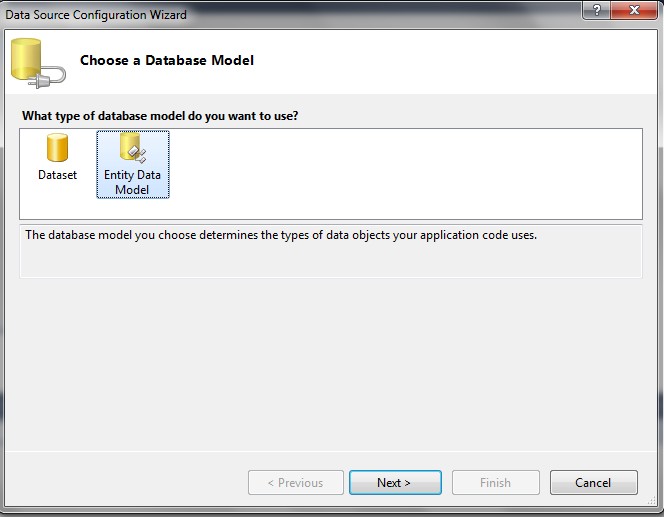
## Cách tạo cơ sở dữ liệu (Database) trong môi trường visual studio 2010

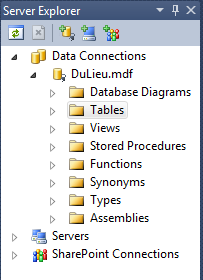
Trong bộ Visual Studio 2010 có hỗ trợ SQL server 2008, nên ta dùng luôn hỗ trợ này để tạo CSDL nhƣ hƣớng dẫn dƣới đây:

***7.2.1. Tạo mới một DataBase***

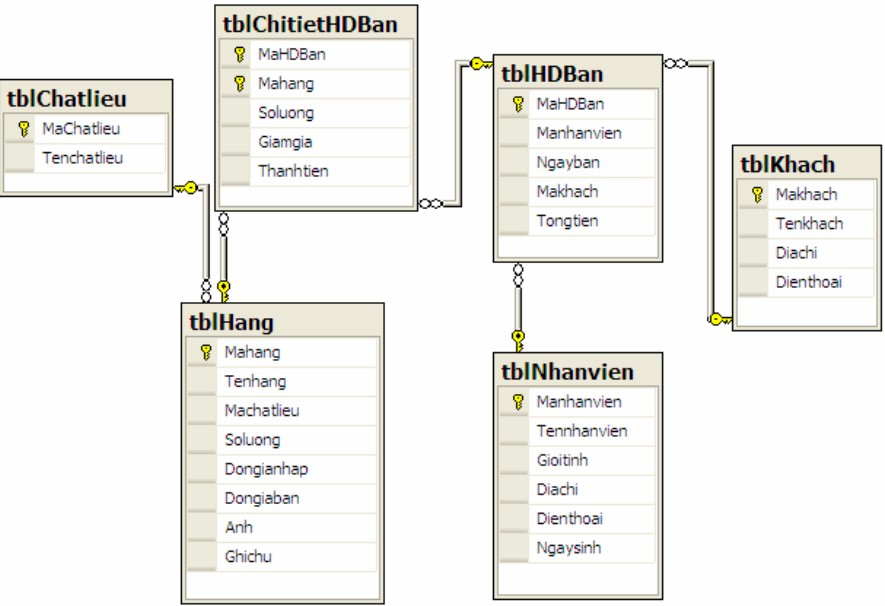
Ta sẽ lƣu CSDL vào thƣ mục ***DataBase*** đã tạo trong thƣ mục **bin/Degug/** ở trên. Trong cửa sổ Solution Explorer kích chuột phải tại thƣ mục DataBase, chọn ***Add/New Item…*** xuất hiện cửa sổ ***Add new item***

Chọn ***Service-based DataBase,*** ở hộp **Name** đặt tên CSDL là ***DuLieu.mdf*** sau đó nhấn nút ***Add***. Kết quả xuất hiện cửa sổ Data Source Configuration Wizrad:



Chọn ***Cancel*** để tạo một CSDL rỗng**.** Kết quả trong thƣ mục **DataBase** xuất hiện vào file **DuLieu.mdf.** Kích đúp chuột vào đối tƣợng này xuất hiện cửa sổ **Server Explorer** cho phép tạo CSDL cho dự án nhƣ sau:

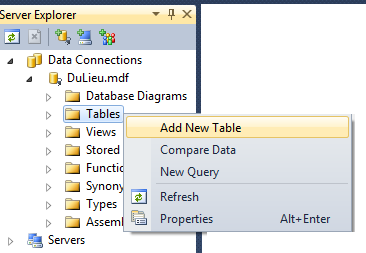
Bây giờ ta bắt đầu tạo ra một CSDL quan hệ nhƣ sau:



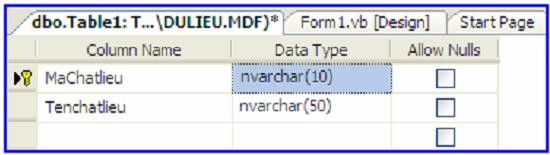
* + 1. ***Tạo các bảng CSDL***

Để tạo một bảng, ví dụ nhƣ bảng **tblChatlieu** ta làm nhƣ sau:

* Kích chuột phải vào **Table** và chọn **Add New Table**



* Tạo bảng chất liệu gồm các trƣờng nhƣ sau:



* Để tạo khóa chính cho trƣờng *MaChatLieu*, ta kích cuột phải tại dòng *MaChatLieu*

chọn *Set Primary Key.*

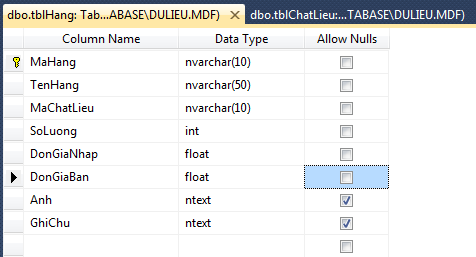
* Để lƣu bảng nhấn *Ctrl + S* xuất hiện cửa sổ *Choose Name,* đặt tên cho bảng là

*tblChatlieu* rồi nhấn *OK.*

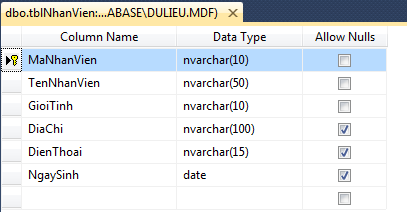


**Bài tập:** Sinh viên tự tạo tƣơng tự với các bảng khác với tên bảng và các trƣờng dữ liệu nhƣ sau:

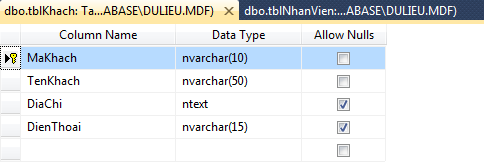
*Bảng tblHang*



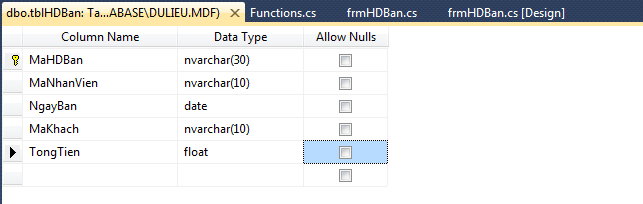
*Bảng tblNhanvien*



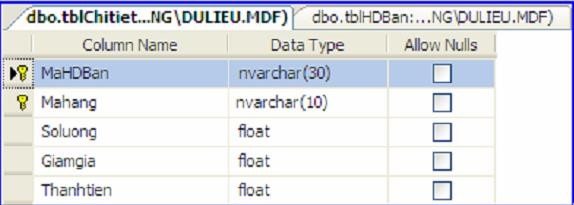
*Bảng tblKhach*



*Bảng tblHDBan*

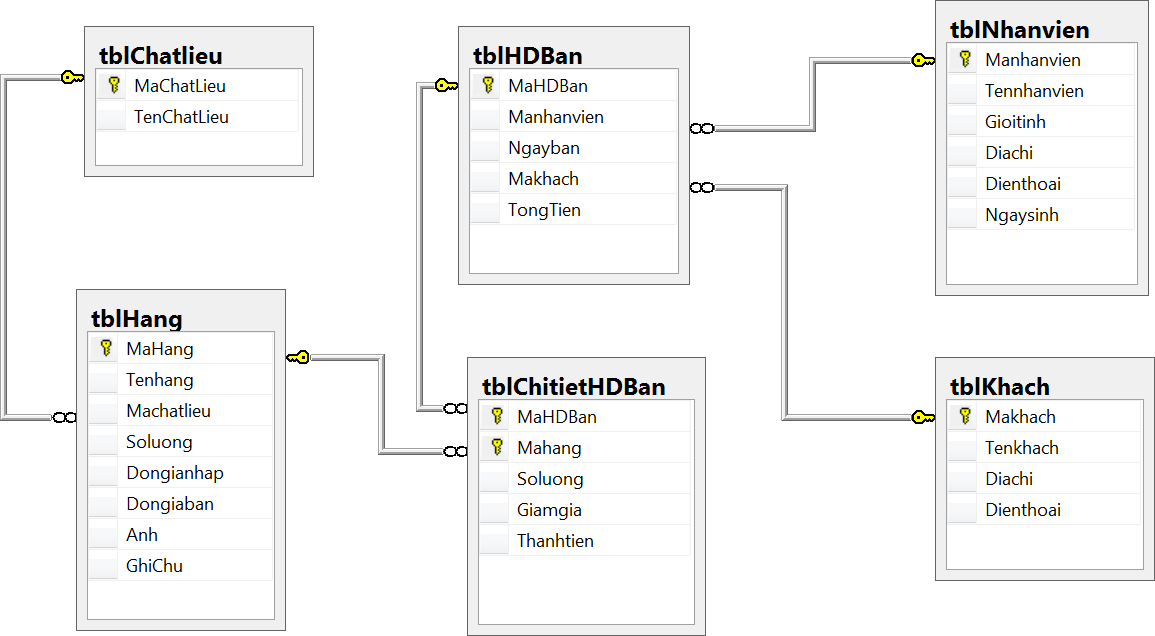


*Bảng tblChitietHDBan*



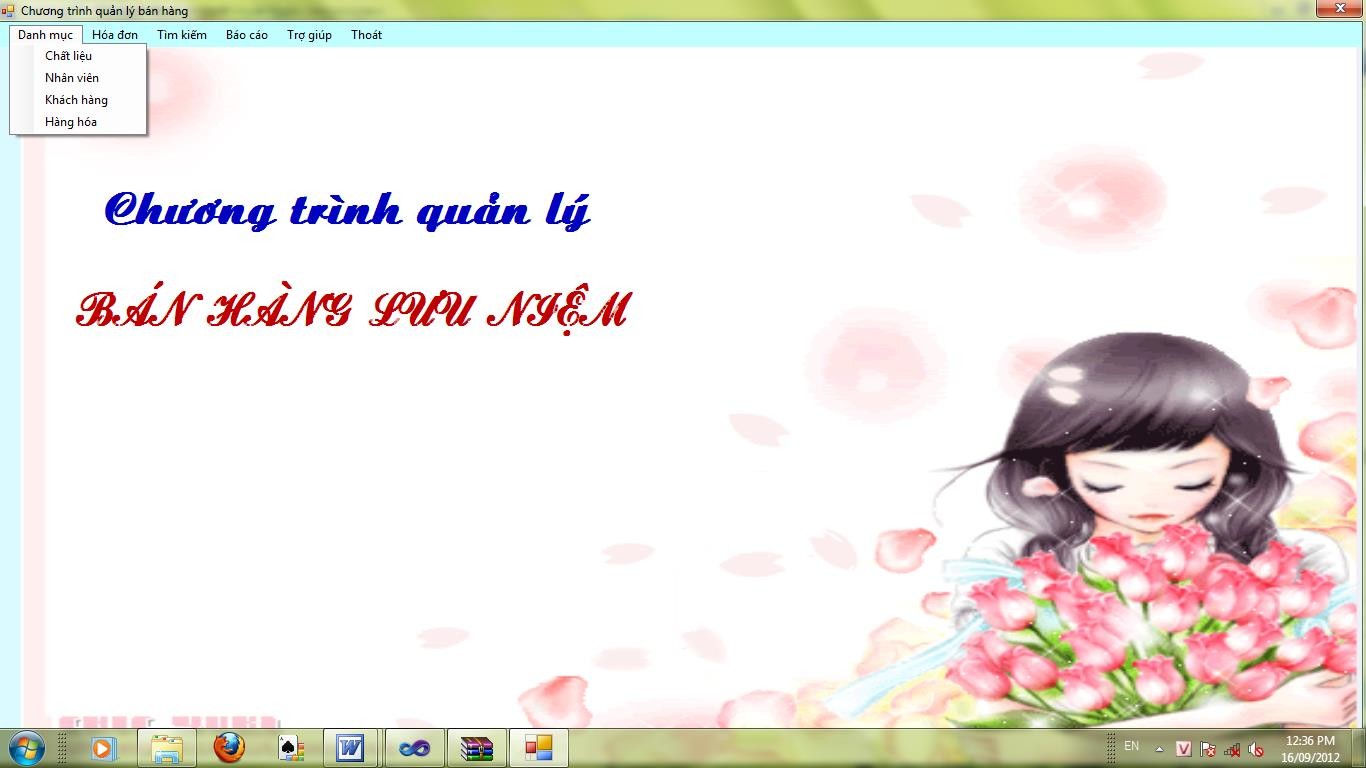
* + 1. ***Tạo quan hệ Relationship cho CSDL***

Kích chuột phải tại *DataBase Diagrams* chọ *Add New Diagram* và tạo quan hệ cho CSDL nhƣ hình dƣới đây:



### Bài tập 27:

Trong project QuanLyBanHang tạo form frmMain có giao diện nhƣ sau:



sau:

Trong form trên có các điều khiển chính:

+ Tạo menu cho chƣơng trình gồm các mục menu với thuộc tính Name và Text nhƣ

Danh mục *mnuDanhMuc*

Chất liệu *mnuChatLieu*

Nhân viên *mnuNhanVien* Khách hàng *mnuKhachHang* Hàng hóa *mnuHangHoa*

Hóa đơn *mnuHoaDon*

Hóa đơn bán *mnuHDBan*

Tìm kiếm *mnuTimKiem* Hóa đơn *mnuTKHoaDon* Hàng *mnuTKHang*

Khách hàng *mnuTKKhach*

Báo cáo *mnuBaoCao*

Hàng tồn *mnuBCHangTon*

Doanh thu *mnuBCDoanhThu*

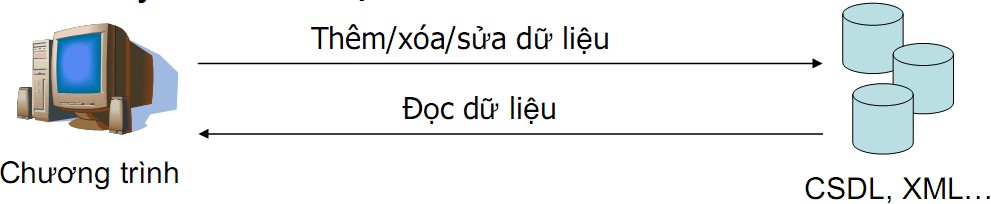
Trợ giúp *mnuTroGiup*

Thoát *mnuThoat*

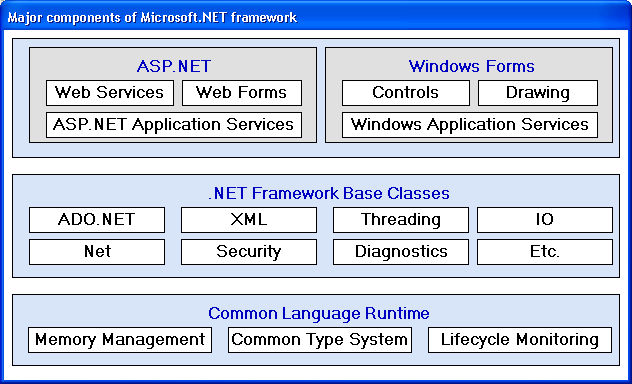
+ Thêm 2 nhãn Label với Text là: „Chƣơng trình quản lý‟ và „Bán hàng lƣu niệm‟

## ADO.NET (ActiveX Data Objects for .NET Framework)

* + 1. ***Giới thiệu ADO.NET***

ADO.NET là một công nghệ truy cập dữ liệu, nó gồm các lớp nằm trong bộ thƣ viện lớp cơ sở của .NET Framework để cho phép các ứng dụng Windows (nhƣ c#, VB.net) hay các ứng dụng Web (nhƣ ASP.Net) thao tác dễ dàng với các nguồn dữ liệu.

### Vị trí của ADO.NET trong công nghệ .NET Framework

ADO.NET nằm trong tầng các lớp cơ sở của .NET Framework (.NET Framework Base Classes). Nó nằm giữa tầng Common Language Runtime và tầng ứng dụng.

Từ kiến trúc này ta thấy rằng ADO.NET là một phần nội tại của .NET Framework, do vậy nó có thể đƣợc sử dụng trong tất cả các ngôn ngữ hộ trợ .NET nhƣ C#, VB.Net… mà không có sự khác biệt nào (Tức là các chức năng cũng nhƣ cách sử dụng hoàn toàn giống nhau).

### Đặc điểm chính của ADO.NET

Khả năng làm việc với dữ liệu không kết nối. Dữ liệu đƣợc lƣu trữ trong bộ nhớ nhƣ một CSDL thu nhỏ (DataSet). Nhằm tăng tốc độ sử lý tính toán và hạn chế sử dụng tài nguyên. Trên mô hình ADO.NET, các ứng dụng sẽ kết nối với nguồn dữ liệu khi đọc hoặc cập nhật dữ liệu, sau đó thì cho đóng đƣờng dây kết nối lại. Điểm này rất quan trọng vì trong các ứng dụng Client – server hoặc phấn tán, việc để mở liên tục các đƣờng dây kết nối là một tiêu hao nguồn lực vô lối.

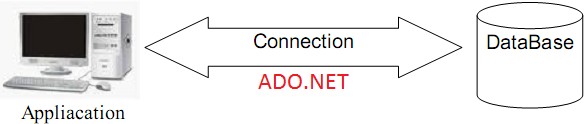
Khả năng xử lý dữ liệu chuẩn XML (có thể trao đổi với bất kỳ hệ thống nào). XML là một ngôn ngữ tổng quát định dạng dữ liệu thông qua các thẻ tự định nghĩa, nó là một chuẩn mới mà ngày càng đƣợc định dạng rộng rãi.

### So sánh ADO.NET với các công nghệ ADO

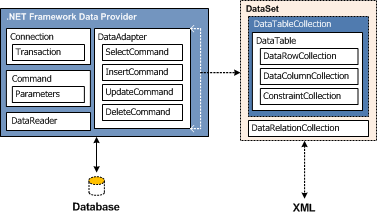
|  |  |
| --- | --- |
| **ADO** | **ADO.NET** |
| Sử dụng RecordSet giống nhƣ một bảng dữ liệu | Sử dụng DataSet có thể bao hàm một hay nhiều bảng thực hiện bởi đối tƣợng DataTable. |
| Duyệt mẩu tin tuần tự trên Recordset | Cho phép duyệt không tuần tự nhiều mẩu tin trên bảng dữ liệu.Cho phép định hƣớng một số mẩu tin trên bảng này với một số mẩu tin trên bảng khác. |
| Sử dụng OLEDB Provide để kết nối cơ sở dữ liệu | Sử dụng đối tƣợng Connection, DataSet, Command, DataAdapter |
| Sử dụng kết nối dữ liệu trong một thời gian dài | Ngắt kết nối cơ sở dữ liệu |

* + 1. ***Kiến trúc của ADO.NET***

Ứng dụng thì đƣợc kết nối với CSDL (DataBase) thông qua các công nghệ kết nối nhƣ ADO, DAO, ADO.NET, … Ở đây ta quan đến ADO.NET



Kiến trúc của ADO.NET đƣợc thể hiện trong hình vẽ sau:



Kiến trúc ADO.NET có thể chia ra làm 2 phần chính là:

### .NET Framework Data provider

Thành phần này giữ nhiệm vụ làm việc trực tiếp với DataBase. Nó bao gồm các thành phần sau:

+ ***Connection:*** Đối tƣợng cho phép bạn kết nối tới CSDL nhƣ: SQL server, Oracle, ODBC, OLEDB, MySQL.

+ **Command:** Đối tƣợng cho phép bạn truy cập cơ sở dữ liệu và thực thi câu lệnh SQL hay các store procedure.

+ **DataReader:** Bộ đọc dùng để đọc nhanh dữ liệu theo một chiều

+ **DataAdapter:** Cầu nối giữa đối tƣợng DataSet, DataTable với CSDL.

### DataSet

Đây là một loại dữ liệu tách rời hẳn với DataBase. Nó là bản sao thu nhỏ của CSDL bao gồm các DataTable và các mối quan hệ (DataRelation)

* + 1. ***Các bước làm việc với CSDL sử dụng ADO.NET***

Bƣớc 1: Tạo kết nối (thiết lập chuỗi kết nối, khai báo và tạo Connection) Bƣớc 2: Mở kết nối dữ liệu

Bƣớc 3: Tạo lệnh SQL Bƣớc 4: Thực thi lệnh SQL

Bƣớc 5: Đóng kết nối và hủy các đối tƣợng (nếu cần)

* + 1. ***Các đối tượng trong ADO.NET và một số phương thức***

Từ kiến trúc của ADO.NET, ta có thể thấy rằng ADO.NET có các đối tƣợng chính là: Connection, Command, DataReader, DataAdapter, DataSet. Sau đây ta sẽ đi tìm hiểu từng đối tƣợng một.

Các namespaces trong ADO.NET để làm việc với cơ sở dữ liệu là:

+ System.Data

+ System.Data.Oledb: Sử dụng với Access

+ System.Data.SqlCient: Sử dụng với SQLServer

+ System.data.OcracleClient (Sử dụng với Ocracle)

+ MicroSoft.data.Odbc (Thông qua ODBC của HĐH)

### Connection

Vai trò của Connection trong ADO.NET là tạo kết nối giữa ứng dụng với cơ sở dữ liệu.

Khi thực hiện kết nối cơ sở dữ liệu ta phải khai báo các thông tin cho Connection trong chuỗi kết nối. Tùy thuộc vào từng loại CSDL khác nhau mà chuỗi kết nối cũng khác nhau.

Tùy từng loại CSDL khác nhau mà ta dùng các Connection khác nhau nhƣ: SqlConnection, OledbConnection, OracleConnection, OdbcConnection.

Khi thực hiện kết nối cần khai báo các thông tin cho Connection thông qua các thuộc tính trong chuỗi kết nối. Tùy từng loại CSDL khác nhau mà chuỗi kết nối cũng khác nhau nhƣ:

Nếu kết nối với CSDL Access

Provider: Khai báo Data Provider của Access (Provider=Microsoft.Jet.OLEDB.4.0)

Data Source: Đường dẫn đến tên tệp tin CSDL (.mdb) User ID: Tên người dùng

Password : Mật khẩu

Nếu kết nối với CSDL SQL server

Data Source/Server: Tên Server Initial Catalog/DataBase: Tên CSDL User ID/UID: Tên người dùng Password/ PWD: Mật khẩu

Integrated Security: Cơ chế chứng thực đăng nhập

+ true: tài khoản Windows;

+ false: Tài khoản SqlServer (ví dụ: sa)

### Khai báo và khởi tạo Connection

<Loại Connection> <Biến Connection>;

<Biến Connection>=new <Loại Connection>(chuỗi kết nối);

Vừa khai báo vừa khởi tạo:

<Loại Connection> <Biến Connection>=new <Loại Connection>(chuỗi kết nối)

### Các phương thức:

+ Close: Đóng kết nối

+ Open: Thực hiện kết nối

+ Dispose: Hủy đối tƣợng

### Các thuộc tính của Connection:

+ State: Trạng thái của Connection:

+ Open: Đã kết nối

+ Closed: Đã đóng kết nối

+ Executing: Kết nối đang thực hiện một lệnh

+ Fetching: Kết nối đang truy xuất dữ liệu

+ Connecting: Đang thực hiện kết nối

### Command

Sau khi tạo kết nối CSDL, mọi thao tác với nguồn dữ liệu (nhƣ thêm, sửa, xóa dữ liệu) có thể đƣợc thông qua Command.

### Cú pháp tạo Command:

<Loại Command> <Biến Command>;

<Biến Command>.Connection=<Biến Connection >;

<Biến Command>.CommandText=<Lệnh SQL>;

Hoặc

<Loại Command> <Biến Command> = New <Loại Command>(<Lệnh SQL>)

<Biến Command>.Connection=<Biến Connection >

Mỗi loại CSDL có những loại Command khác nhau, các loại Command thƣờng dùng nhƣ: SqlCommand, OledbCommand, OracleCommand, OdbcCommand.

### Thuộc tính hay dùng của Command

+ CommandText: Lệnh SQL hay tên Procedure

+ Connection: Đối tƣợng Connection

### Các phương thức:

*+ ExecuteReader*: Trả về đối tƣợng DataReader để đọc dữ liệu mỗi lần một dòng với phƣơng thức Read. Cú pháp:

<Loại DataReader> <Tên DataReader>;

<Tên DataReader> = <tên Command>.ExecuteReader;

VD:

SqlDataReader rd ;

rd = cmd.ExecuteReader;

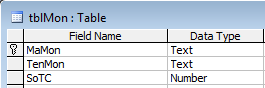
Tƣơng tự nhƣ các đối tƣợng Connection hay Command thì mỗi loại CSDL sẽ tƣơng ứng với các loại DataReader khác nhau. Nội dung này sẽ đƣợc nói đến trong phần DataReader ngay phía dƣới.

*+ ExcuteNoneQuery*: Dùng thực thi các phát biểu T-Sql nhƣ: Insert, Update, Delete, Create Table,…

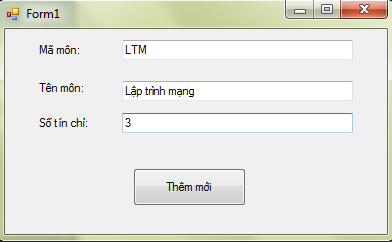
*+ ExcuteScalar*: Trả về phát biểu SQL chỉ có một hàng, một cột (1 giá trị đơn)

+ + Dispose: Hủy đối tƣợng

**Ví dụ :** Cho bảng CSDL Access tblMon có cấu trúc nhƣ sau :



Viết chƣơng trình thêm mới một môn học có giao diện nhƣ sau :



Khi ta điền đầy đủ thông tin vào form rồi nhấn nút Thêm mới thì dữ liệu đƣợc thêm mới vào CSDL.

### Hướng dẫn

Trong ví dụ này file DuLieu.mdb nằm trong thƣ mực /bin.Debug/DataBase của dự án. Khai báo thêm 2 namespace sau :

using System.Data;

using System.Data.OleDb; //Làm việc với CSDL Access

Viết mã lệnh cho sự kiện click của nút thêm mới làm việc theo các bƣớc của ADO.NET nhƣ sau:

private void btnThemMoi\_Click(object sender, EventArgs e)

{

//Bước 1: Tạo kết nối: Thiết lập chuỗi kết nối, khai báo và tạo connection

string strConnect="Provider=Microsoft.Jet.OLEDB.4.0;"+ "Data Source= DataBase\\DuLieu.mdb;"+

"Persist Security Info=True";

OleDbConnection conection = new OleDbConnection(strConnect);

//Bước 2: Mở kết nối

if (conection.State != ConnectionState.Open) conection.Open();

//Bước 3: Tạo lệnh SQL

string sql = "insert into tblMon(MaMon,TenMon,SoTC)"+

" values('"+txtMaMon.Text+"','"+txtTenMon.Text+"','"+txtSoTC.Text+"')";

//Bước 4: Thực thi lệnh SQL sử dụng đối tượng OLEDBCommand OleDbCommand command = new OleDbCommand(); command.Connection = conection; //Chỉ định đối tượng kết nối

command.CommandText = sql; //Truyền lệnh sqlcho thuộc tính CommandText command.ExecuteNonQuery(); //Thực thi lệnh sql

//Bước 5: Đóng kết nối

if (conection.State != ConnectionState.Closed) conection.Close();

conection.Dispose();//Hủy đối tượng

command.Dispose();//Hủy đối tượng Command

}

### DataReader

Là đối tƣợng truy cập dữ liệu trực tiếp, sử dụng con trỏ phía Server và duy trì kết nối với Server trong suốt quá trình đọc dữ liệu.

Dùng để đón nhận kết quả trả về từ phƣơng thức ExecuteReader của đối tƣợng Command. Nó tƣơng tự nhƣ một Recordset của ADO, tuy nhiên dữ liệu là Readonly và chỉ đọc theo chiều tiến.

Tùy từng loại CSDL khác nhau mà ta dùng các DataReader khác nhau nhƣ: SqlDataReader, OledbDataReader, OracleDataReader, OdbcDataReader.

//Khai báo một DataReader, không có new

<Loại DataReader> <Tên DataReader>;

//Lấy kết quả từ Commmand

<Tên DataReader> = <tên Command>.ExecuteReader();

### Các thuộc tính

+ FieldCount: Số cột trên dòng hiện hành của DataReader

+ IsClosed : Cho biết dataReader đã đóng

### Các phương thức

+ GetFieldType: Trả về kiểu dữ liệu của tham số truyền vào

+ GetName: Trả về tên của cột truyền vào (cột là số TT cột)

+ GetValue: Trả về trị của cột truyền vào (cột là số TT cột)

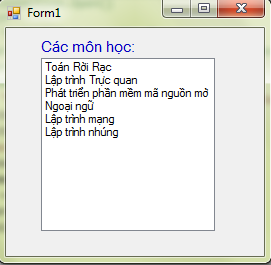
+ Read: Di chuyển đến dòng kế tiếp và trả về true nếu còn dòng để di chuyển, ngƣợc lại trả về false.

+ Close: đóng DataReader

+ Dispose: Hủy đối tƣợng

Trong khi dataReader đang mở các thao tác dữ liệu trên nguồn dữ liệu đều không thể cho đến khi dataReader đóng lại bằng lệnh Close.

**Ví dụ:** Vẫn lấy CSDL Access ở trên. Viết đoạn chƣơng trình sử dụng DataReader đọc dữ liệu trong bảng tblMon đổ ra ListBox nhƣ hình minh họa sau:

Tạo giao diện form gồm các điều khiển: Label có Text là: "Các môn học"; điều khiển ListBox có Name là: lstMonHoc.

sau:

### Hướng dẫn:

Khai báo thêm 2 namespace sau :

using System.Data;

using System.Data.OleDb; //Làm việc với CSDL Access

Viết mã lệnh cho sự kiện Load của Form làm việc theo các bƣớc của ADO.NET nhƣ

private void frmDataReader\_Load(object sender, EventArgs e)

{

//Bước 1: Tạo kết nối: Thiết lập chuỗi kết nối, khai báo và tạo connection

string strConnect = "Provider=Microsoft.Jet.OLEDB.4.0;" + "Data Source=DataBase\\DuLieu.mdb;" +

"Persist Security Info=True";

OleDbConnection conection = new OleDbConnection(strConnect);

//Bước 2: Mở kết nối

if (conection.State != ConnectionState.Open) conection.Open();

//Bước 3: Tạo lệnh SQL

string sql = "Select \* from tblMon";

//Bước 4: đọc kết quả với DataReader từ đối tượng Command OleDbCommand command = new OleDbCommand(); command.Connection = conection; //Chỉ định đối tượng kết nối

command.CommandText = sql; //Truyền lệnh sql cho thuộc tính CommandText

OleDbDataReader dataReader; dataReader = command.ExecuteReader();

//Duyệt các bản ghi trong DataReader điền vào listBox while (dataReader.Read() == true)

lstMonHoc.Items.Add(dataReader.GetValue(1));

//Bước 5: Đóng kết nối

if (conection.State != ConnectionState.Closed) conection.Close();

conection.Dispose();//Hủy đối tượng connection dataReader.Close();//đóng dataReader dataReader.Dispose()//Hủy đối tượng dataReader

}

### DataSet

DataSet là mô hình CSDL quan hệ thu nhỏ tách rời hẳn với phía server đáp ứng các nhu cầu với ứng dụng.

DataSet chứa các bảng (DataTable), các quan hệ (DataRelation) và các ràng buộc (Constraint)

### Phương thức hay dùng:

+ Thêm một bảng vào Dataset. Một bảng mới tự động đƣợc tạo ra với tên mặc định Table1, Table2 . . .

Tables.Add()

+ Một bảng mới tạo ra theo đúng <tên bảng>. Tên bảng có phân biệt chữ in, thƣờng

Tables.Add(<Tên bảng>)

+ Xóa bảng ra khỏi Dataset

Tables.Remove(<Tên bảng>)

+ Kiểm tra bảng có thuộc Dataset không

Tables.Contains(<Tên bảng>)

+ Lấy chỉ số của bảng

Tables.IndexOf[<tên bảng>]

+ Lấy số bảng trong Dataset

Tables.Count

+ Lấy ra một bảng trong Dataset

Tables[<Chỉ số>]

+ Để xóa bỏ mọi dữ liệu trên dataSet

Clear()

+ Để tạo một bản sao của Dataset

Clone()

+ Xóa bỏ Dataset. Giải phóng mọi tài nguyên trên vùng nhớ Dataset đang sủ dụng.

Dispone()

### DataAdapter

Để lấy dữ liệu từ nguồn dữ liệu về cho ứng dụng, chúng ta sử dụng đối tƣợng DataAdapter. Đối tƣợng này cho phép ta lấy cấu trúc và dữ liệu của các bảng.

Mỗi loại CSDL ta sẽ phải dùng tƣơng ứng với một loại DataAdapter khác nhau: SqlDataAdapter, OdbcDataAdapter, OledbDataAdapter, OracleDataAdapter

### Cú pháp tạo DataAdapter:

<Loại DataAdapter> <Biến DataAdapter> = New <Loại DataAdapter>(<Lệnh sql>,<Biến Connection>)

**Chú ý:** DataAdapter chỉ thao tác với nguồn dữ qua đối tƣợng Connection đang kết nối.

### Các thuộc tính của DataAdapter

*+ DeleteCommand:* Xóa dữ liệu, tƣơng ứng với lệnh DELETE

*+ InsertCommand:* Chèn dữ liệu mới, tƣơng ứng với lệnh INSERT

*+ UpdateCommand:* Cập nhật dữ liệu, tƣơng ứng với lệnh UPDATE

*+ SelectCommand:* Truy vấn dữ liệu, tƣơng ứng với lệnh SELECT trong SQL

### Phương thức

Đối tƣợng DataAdapter có 2 phƣơng thức cơ bản:

+ *Fill:* Thực thi câu lệnh SelectCommand và đổ dữ liệu vào đối tƣợng DataTable. Cú pháp:

Fill(<DataSet>,<Tên dataTable>)

+ *Update:* Thực thi các câu lệnh InsertCommand, UpdateCommand, DeleteCommand,

… Nói chung là các câu lệnh làm thay đổi dữ liệu. Cú pháp:

Update(<mảng dòng>): Cập nhật các dòng (Các đối tƣợng DataRow) vào nguồn dữ

liệu.

Update(<Dataset>): Cập nhật các thay đổi trên tất cả các bảng của Dataset vào nguồn

dữ liệu.

Update(<DataTable>): Cập nhật tất cả các thay đổi trên DataTable vào nguồn dữ liệu.

Update(<Dataset>,<Tên bảng>) Cập nhật các thay đổi trên bảng trong Dataset vào nguồn dữ liệu.

### Điều khiển DataGridView

DataGridView là một cấu trúc bảng gồm các cột, các dòng. Nó dùng để hiển thị dữ liệu dƣới dạng bảng.

### Thuộc tính:

+ Name : Tên của điều khiển, với tiếp đầu ngữ là dgv

+ DataSource: Truyền vào một DataTable chứa dữ liệu

+ Thuộc tính ***Columns:*** liên quan đến việc thiết lập cho các cột của Grid, thiết lập cột nào thì ta truyền chỉ số của cột đó vào bắt đầu từ 0. Ví dụ: Colums(0) – Thiết lập cho cột thứ 1,... Có các thuộc tính thiết lập cho Columns nhƣ sau:

* *HeaderText:* Thiết lập chuỗi tiêu đề cho cột
* *DataPropertyName:* Thiết lập tên trƣờng dữ liệu hiển thị cho cột.
* *Width:* Thiết lập độ rộng cho cột

+ ReadOnly: Thuộc tính thiết lập là true nếu grid chỉ để đọc ngƣợc lại là false.

+ BackGroundColor: Thuộc tính quy định màu nền cho grid (dùng lớp Color để chọn màu cho Grid)

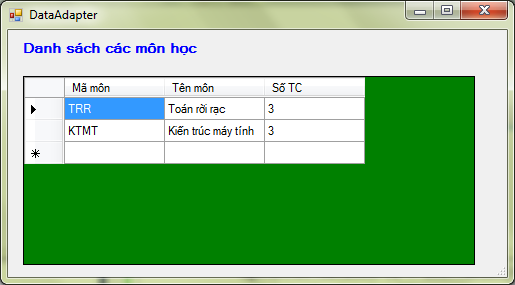
+ ForeColor: Quy định màu chữ cho Grid

+ Font: Quy định về Font chữ cho Grid (nhƣ kiểu chữ, màu chữ, …)

### Các sự kiện

+ Click: Đƣợc kích hoạt khi ngƣời dùng click vào lƣới

**Ví dụ**: Vẫn dùng CSDL Access ở trên. Viết đoạn chƣơng trình sử dụng DataAdapter đọc CSDL và liệt kê danh sách các môn học có trong CSDL nhƣ hình sau :



### Hướng dẫn :

Tạo giao diện form nhƣ trên gồm các điều khiển: Label có Text là: "Danh sách các môn học"; điều khiển DataGridView có Name là: dgvMonHoc.

Khai báo thêm 2 namespace sau :

using System.Data;

using System.Data.OleDb; //Làm việc với CSDL Access

Viết mã lệnh cho sự kiện Load của Form làm việc theo các bƣớc của ADO.NET nhƣ

sau:

private void frmDataAdapter\_Load(object sender, EventArgs e)

{

//Bước 1: Tạo kết nối: Thiết lập chuỗi kết nối, khai báo và tạo connection

string strConnect = "Provider=Microsoft.Jet.OLEDB.4.0;" + "Data Source=DataBase\\DuLieu.mdb;" +

"Persist Security Info=True";

OleDbConnection conection = new OleDbConnection(strConnect);

//Bước 2: Mở kết nối

if (conection.State != ConnectionState.Open) conection.Open();

//Bước 3: Tạo lệnh SQL

string sql = "Select \* from tblMon";

//Bước 4: Sử dụng đối tượng DataAdapter thực thi lệnh Select đổ ra DataTable OleDbDataAdapter dataAdapter = new OleDbDataAdapter(sql, conection); DataSet dts = new DataSet();

dataAdapter.Fill(dts, "dtMon");//Đổ dữ liệu vào dataTable dtMon trong dataset dts

//Bước 5: Đóng kết nối

if (conection.State != ConnectionState.Closed) conection.Close();

conection.Dispose();//Hủy đối tượng connection

//Lấy dữ liệu ra datagridview dgvMonHoc.DataSource = dts.Tables[0];

//Trình bày dataGridView dgvMonHoc.Columns[0].HeaderText = "Mã môn"; dgvMonHoc.Columns[1].HeaderText = "Tên môn";

dgvMonHoc.Columns[2].HeaderText = "Số TC"; dgvMonHoc.BackgroundColor = Color.Green;

//Hủy đối tượng DataSet dts.Dispose();

}

## Làm việc với CSDL SQL qua các đối tượng: SqlConnection, SqlDataAdaper, SqlCommand.

Mỗi loại CSDL khác nhau thì ta dùng các đối tƣợng Connection, DataAdapter và Command khác nhau. Ở phần này chúng ta chỉ đề cập đến một loại CSDL đó chính là SQL server. Khi dùng Sql server thì các đối tƣợng tƣơng ứng của nó sẽ có là: SqlConnection, SqlDataAdapter, SqlCommand.

* + 1. ***Đối tượng SqlConnection***

Đối tƣợng này có chức năng kết nối với cơ sở dữ liệu Sql server nhờ các thuộc tính và phƣơng thức mà chúng có.

Để kết nối CSDL SQL server, chúng ta làm theo các bƣớc nhƣ sau:

**Bước 1:** Khai báo sử dụng không gian tên:

using System.Data;

using System.Data.SqlClient;

**Bước 2:** Khai báo đối tƣợng SqlConnection

SqlConnection <tên đối tượng SqlConnection>;

**Ví dụ:**

SqlConnection Conn;

**Bước 3:** Khai báo chuỗi kết nối

Ví dụ ta dùng CSDL SQL server đƣợc hỗ trợ sẵn trong VS 2010 thì ta sẽ tìm đƣợc chuỗi kết nối bằng cách:

+ Kích vào tên CSDL ở Server explorer.

+ Chuỗi kết nối chính là thuộc tính ConnectinString của CSDL. Ở CSDL ***DuLieu.mdf***

có chuỗi kết nối nhƣ sau:

Data source =

.\SQLEXPRESS;AttachDbFilename=E:\QuanLybanHang\bin\Debug\DataBase\DuLieu.mdf;Integrated Security=True;User Instance=True;

Ta thấy với đƣờng dẫn trên là đƣờng dẫn tuyệt đối và liên kết nên sẽ gặp rắc rối nếu ta chuyển ứng dụng sang máy khác. Nên ta sử dụng đƣờng dẫn tƣơng đối bằng cách dùng: Với

*System.IO.Directory.GetCurrentDirectory().ToString()* để lấy ra đƣờng dẫn của ứng dụng nơi chứa file chạy của chƣơng trình. Sau đó dẫn đến ***\DataBase\DuLieu.mdf*** nhƣ sau:

string strConnect = "Data Source=.\\SQLEXPRESS;AttachDbFilename="+ System.IO.Directory.GetCurrentDirectory().ToString()+"\\DataBase\\" +

"DuLieu.mdf;Integrated Security=True";

**Bước 4:** Khởi tạo đối tƣợng SqlConnection

Conn = New SqlConnection(strConnect);

**Bước 5:** Mở kết nối. Trƣớc khi mở cần kiểm tra xem kết nối đã đƣợc mở chƣa?

if (Conn.State != ConnectionState.Open) Conn.Open();

**Bước 6:** Thiết lập câu lệnh SQL và thực thi nó thông qua các đối tƣợng SqlDataAdapter hoặc SqlCommand

**Bước 7:** Đóng kết nối và hủy các đối tƣợng (nếu có)

if (Conn.State != ConnectionState.Closed) Conn.Close();

Conn.Dispose();

//Hủy các đối tượng khác nếu có

* + 1. ***Đối tượng SqlDataAdapter và SqlCommand***

### Dùng SqlDataAdapter

Ta dùng SqlDataAdapter nhƣ một thành phần điều phối dữ liệu, tức nó không lƣu trữ dữ liệu mà chỉ phát đi các lệnh SQL và nhận dữ liệu trả về của CSDL sau đó chuyển cho đối tƣợng khác (thƣờng là DataTable)

Thông thƣờng ta dùng SqlDataAdapter để lấy dữ liệu khi thực hiện câu lệnh Select đổ ra một DataTable.

Các bƣớc thực hiện nhƣ sau:

+ Khai báo chuỗi lệnh SQL

+ Khai báo và khởi tạo một SqlDataAdapter với 2 đối truyền vào là câu lệnh Sql và đối tƣợng SqlConnection

SqlDataAdapter DA = new SqlDataAdapter(strsql, Conn); //strsql là chuỗi câu lệnh sql, Conn là tên đối tượng SqlConnection

+ Khai báo và khởi tạo một DataTable

DataTable DT = new DataTable();

+ Đổ dữ liệu vào DataTable thông qua phƣơng thức Fill của SqlDataAdapter

DA.Fill(DT)

Ngoài thực thi câu lệnh Select thi đối tƣợng này cũng thực thi các câu lệnh Insert, update, delete,.. tuy nhiên ta hay dùng SqlCommand để làm điều đó hơn.

### Dùng SqlCommand

Cũng nhƣ SqlDataAdapter thì sau khi kết nối CSDL xong thì mọi thao tác trên CSDL SQL server có thể đƣợc thực hiện thông qua SqlCommand. Các bƣớc tiến hành nhƣ sau:

+ Khai báo và khởi tạo đối tƣợng SqlCommand

SqlCommand sqlcommand = new SqlCommand();

+ Truyền đối tƣợng SqlConnection cho thuộc tính Connection.

sqlcommand.Connection = Conn;

+ Truyền chuỗi lệnh Sql vào cho thuộc tính CommandText

sqlcommand.CommandText = strsql; //strsql là một chuỗi sql

+ Thực thi lệnh sql bằng phƣơng thức ***ExecuteNonQuery()***

sqlcommand.ExecuteNonQuery();

* + 1. ***Các thao tác dữ liệu***

Ở đây ta sẽ học các thao tác dữ liệu hay dùng là: truy vấn, thêm mới, sửa, xóa dữ liệu.

Ta sẽ học phần này thông qua việc phân tích một ví dụ cụ thể.

Trong Project **QuanLyBanHang**, trong thƣ mục Forms ta tạo một form mới đặt tên là frmChatLieu:

+ Form này cho phép chúng ta cập nhật các danh mục chất liệu.

+ Bảng CSDL liên quan là: tblChatLieu. Giao diện của form nhƣ sau:



### Các điều khiển dùng trong form là :

|  |  |  |
| --- | --- | --- |
| **Điều khiển** | **Name** | **Text** |
| Form | frmChatLieu | Cập nhật danh mục chất liệu |
| TexBox | txtMaCL |  |
| txtTenCL |  |
| Button | btnThemMoi | Thêm mới |
| btnSua | Sửa |
| btnXoa | Xóa |
| btnBoQua | Bỏ qua |
| btnThoat | Thoát |
| DataGridView | dgvChatLieu |  |

**Nhận xét:** Ta thấy mỗi lần muốn thực thi một câu lệnh SQL dạng select, update, delete hay insert ta đều phải thực hiện trình tự các công việc giống nhau chỉ khác nhau về câu lệnh SQL. Vì vậy để thuận tiện, ta nên xây dựng lớp xử lý dữ liệu đảm nhận việc kết nối CSDL và việc xử lý, cập nhật dữ liệu.

Lớp xử lý dữ liệu gồm các phƣơng thức sau đây:

+ Phƣơng thức ***KetNoiCSDL()*** nhằm thực hiện kết nối tới CSDL.

+ Phƣơng thức ***DongKetNoiCSDL()*** nhằm đóng lại kết nối CSDL.

+ Phƣơng thức ***Docbang(string strsql)*** nhằm thực hiện câu lệnh Select trả về một DataTable.

+ Phƣơng thức ***CapNhatDuLieu(string strsql)*** nhằm thực hiện các câu lệnh Insert, Update, Delete để cập nhật dữ liệu cho CSDL.

Trong cửa sổ Solution Explorer tạo một thƣ mục mới đặt tên là Class, kích chuột phải tại Class chọn Add/New Item… chọn Class trong hoopk thoại Add New Item và đặt tên cho Class là *ProcessDataBase.cs* tại ô Name.

Khai báo thêm 2 namespace sau :

using System.Data;

using System.Data.SqlClient;

Sau đó xây dựng lớp xử lý cơ sở dữ liệu nhƣ sau:

public class ProcessDataBase

{

string strConnect = "Data Source=.\\SQLEXPRESS;AttachDbFilename="+ System.IO.Directory.GetCurrentDirectory().ToString()+"\\DataBase\\" +

"DuLieu.mdf;Integrated Security=True"; SqlConnection sqlConnect = null;

//Hàm mở kết nối CSDL private void KetNoiCSDL()

{

sqlConnect = new SqlConnection(strConnect); if (sqlConnect.State != ConnectionState.Open)

sqlConnect.Open();

}

//Hàm đóng kết nối CSDL private void DongKetNoiCSDL()

{

if (sqlConnect.State != ConnectionState.Closed) sqlConnect.Close();

sqlConnect.Dispose();

}

//Hàm thực thi câu lệnh dạng Select trả về một DataTable public DataTable DocBang(string sql)

{

DataTable dtBang=new DataTable (); KetNoiCSDL();

SqlDataAdapter sqldataAdapte = new SqlDataAdapter(sql, sqlConnect); sqldataAdapte.Fill(dtBang);

DongKetNoiCSDL(); return dtBang;

}

//Hàm thực lệnh insert hoặc update hoặc delete public void CapNhatDuLieu(string sql)

{

KetNoiCSDL();

SqlCommand sqlcommand = new SqlCommand(); sqlcommand.Connection = sqlConnect; sqlcommand.CommandText = sql; sqlcommand.ExecuteNonQuery(); DongKetNoiCSDL();

}

}

Sau đó trong mỗi lớp sử dụng lớp *ProcessDataBase* ở trên ta phải khai báo khởi tạo đối tƣợng này nhƣ sau:

*ProcessDataBase dtBase=new ProcessDataBase();*

#### Viết sự kiện Load của form

Khi bắt đầu load form nên ta sẽ phải hiển thị hết các danh mục chất liệu trong bảng tblChatlieu lên DataGridView. Để làm điều này ta viết chƣơng trình cho sự kiện Load của form Chất liệu theo các bƣớc sau:

+ Bƣớc 1: Gọi phƣơng thức **DocBang** trong lớp DataBase ở trên với câu lệnh SQL truyền vào là: Select \* from tblChatlieu. Hàm này sẽ trả về một DataTable chứa danh sách các Chất liệu.

+ Bƣớc 2: Gán DataTable lấy đƣợc ở bƣớc 1 gán vào thuộc tính DataSource của DataGridView.

+ Bƣớc 3: Giải phóng bộ nhớ cho DataTable

Đồng thời khi load Form lên thì ngƣời dùng chỉ kích đƣợc vào nút Thêm hoặc Thoát; các nút còn lại không kích đƣợc. Để làm điều này ta sử dụng thuộc tính Enabled của các nút

Đoạn chƣơng trình cụ thể nhƣ sau:

private void frmChatLieu\_Load(object sender, EventArgs e)

{

//Gọi pt DocBang lấy dữ liệu của bảng tblChatLieu đổ vào DataTable DataTable dtChatLieu = dtBase.DocBang("select \* from tblChatLieu"); dgvChatLieu.DataSource = dtChatLieu;

//Định dạng dataGrid dgvChatLieu.Columns[0].HeaderText = "Mã chất liệu"; dgvChatLieu.Columns[1].HeaderText = "Tên chất liệu"; dgvChatLieu.Columns[0].Width = 150;

dgvChatLieu.Columns[1].Width = 250; dgvChatLieu.BackgroundColor = Color.LightBlue; dtChatLieu.Dispose();//Giải phóng bộ nhớ cho DataTable

btnXoa.Enabled = false; btnSua.Enabled = false; btnLuu.Enabled = false; btnBoQua.Enabled = false; btnThemMoi.Enabled = true;

}

#### Viết sự kiện click cho DataGridView

Khi ngƣời dùng kích chuột chọn một dòng dữ liệu trên DataGridView các dữ liệu tƣơng ứng sẽ đƣợc hiển thị lên trên 2 TextBox mã chất liệu và tên chất liệu. Đồng thời nút Sửa, xóa, Bỏ qua sáng lên để ngƣời dùng có thể chọn sửa, xóa hoặc bỏ qua thao tác này. Nút thêm và lƣu không kích đƣợc.

Khi sửa dữ liệu ta không sửa mã cho nên ô nhập mã không thể kích hoạt đƣợc (dùng thuộc tính Enabled)

private void dgvChatLieu\_Click(object sender, EventArgs e)

{

txtMaCL.Text = dgvChatLieu.CurrentRow.Cells[0].Value.ToString(); txtTenCL.Text = dgvChatLieu.CurrentRow.Cells[1].Value.ToString();

txtMaCL.Enabled= false; btnXoa.Enabled = true; btnSua.Enabled = true; btnLuu.Enabled = false; btnBoQua.Enabled = true; btnThemMoi.Enabled = false;

}

#### Viết sự kiện click cho nút Sửa

Khi ngƣời dùng kích chuột vào một dòng bản ghi bất kỳ trên lƣới để hiển thị dữ liệu của bản ghi đó lên Form thì ngƣời dùng có thể chỉnh sửa các thông tin đó. Phƣơng thức *btnSua\_Click* có tác dụng lƣu các thông tin ngƣời dùng đã sửa vào CSDL.

private void btnSua\_Click(object sender, EventArgs e)

{

if (txtTenCL.Text == "")

{

MessageBox.Show("Bạn phải nhập tên chất liệu", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

txtTenCL.Focus();

}

else

{

dtBase.CapNhatDuLieu("update tblChatLieu set TenChatLieu=N'"

+txtTenCL.Text+ "' where MaChatLieu='"+txtMaCL.Text+"'"); ResetValue();//Xóa dữ liệu ở các ô nhập TextBox

//Sau khi update cần lấy lại dữ liệu để hiển thị lên lưới dgvChatLieu.DataSource = dtBase.DocBang("select \* from

tblChatLieu");

btnXoa.Enabled = false; btnSua.Enabled = false; btnLuu.Enabled = false; btnBoQua.Enabled = false; btnThemMoi.Enabled = true;

}

}

ResetValue() là một phƣơng thức nằm trong lớp frmChatLieu dùng để xóa rỗng dữ liệu trong các ô nhập dữ liệu:

void ResetValue()

{

txtMaCL.Text = ""; txtTenCL.Text = "";

}

#### Viết sự kiện click cho nút Xóa

Ta có thể chọn 1 chất liệu trên DataGridView sau đó chọn nút xóa để xóa dữ liệu đi. Ta sẽ viết mã lệnh cho sự kiện Click của nút btnXoa nhƣ sau:

+ Ta sẽ hỏi xem ngƣời dùng có thực sự muốn xóa không? Nếu ngƣời dùng chọn “yes” thì là có. Nếu ngƣời dùng chọn không thì sẽ không làm gì cả.

+ Sau khi xóa xong thì hiển thị lại dữ liệu lên lƣới và gọi phƣơng thức ResetValue() xóa dữ liệu ở 2 ô TextBox.

Mã lệnh đƣợc viết nhƣ sau:

private void btnXoa\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Bạn có muốn xóa chất liệu có mã là:"+ txtMaCL.Text+" không?","Thông báo", MessageBoxButtons.YesNo,MessageBoxIcon.Question) == System.Windows.Forms.DialogResult.Yes)

{

dtBase.CapNhatDuLieu("delete tblChatLieu where MaChatLieu='" + txtMaCL.Text + "'");

dgvChatLieu.DataSource = dtBase.DocBang("Select \* from

tblChatLieu");

ResetValue(); btnXoa.Enabled = false; btnSua.Enabled = false; btnLuu.Enabled = false; btnBoQua.Enabled = false; btnThemMoi.Enabled = true;

}

}

#### Viết sự kiện click cho nút Thêm

Nút này sẽ xóa rỗng các ô nhập liệu để ngƣời dùng có thể nhập mới một chất liệu. Sau đó cho phép ngƣời dùng có thể nhập dữ liệu vào ô TextBox nhập mã chất liệu. Làm cho nút Lƣu, bỏ qua có thể kích đƣợc; nút xóa, sửa, Thêm không kích đƣợc nữa.

Mã lệnh cụ thể nhƣ sau:

private void btnThemMoi\_Click(object sender, EventArgs e)

{

ResetValue(); txtMaCL.Enabled = true; txtMaCL.Focus();

btnXoa.Enabled = false; btnSua.Enabled = false; btnLuu.Enabled = true; btnBoQua.Enabled = true; btnThemMoi.Enabled = true;

}

#### Viết sự kiện click cho nút Lưu

Ngƣời dùng kích nút Thêm rồi điền đầy đủ thông tin vào các ô nhập dữ liệu, sau đó ấn nút Lƣu thì dữ liệu vừa nhập sẽ đƣợc bổ sung vào bảng tblChatLieu trong CSDL.

Để làm việc này ta viết mã lệnh thực thi cho sự kiện Click của nút btnLuu nhƣ sau:

+ Bƣớc 1: Kiểm tra xem ngƣời dùng có nhập đủ hết các thông tin cho 2 ô TextBox nhập mã chất liệu và tên chất liệu không. Nếu chƣa nhập thì yêu cầu ngƣời dùng nhập và đặt con trỏ về ô cần nhập cho ngƣời dùng nhập.

+ Bƣớc 2: Kiểm tra xem dữ liệu ngƣời dùng nhập có mã trùng với mã đã có trong CSDL không, nếu trùng thì đặt con trỏ vào ô nhập mã chất liệu để nhập lại.

DataTable dtChatLieu = dtBase.DocBang("Select \* from tblChatLieu where"+ " MaChatLieu='"+txtMaCL.Text+"'");

if (dtChatLieu.Rows.Count > 0)

{

MessageBox.Show("Mã chất liệu này đã tồn tại, bạn hãy nhập mã khác!"); txtMaCL.Focus();

}

+ Bƣớc 3: Sau khi mọi dữ liệu đã đƣợc nhập thì gọi thủ tục **ThucThiLenh** trong lớp DataBase để thêm mới chất liệu vào. Với tham số truyền vào thủ tục là lệnh SQL nhƣ sau:

insert into tblChatlieu(MaChatLieu,TenChatLieu) values(N'" + txtMaCL.Text + "',N'" & txtTenCL.Text & "')

+ Bƣớc 4: Đọc lại dữ liệu từ bảng tblChatLieu đặt lại lên lƣới.

+ Bƣớc 5: Gọi phƣơng thức ResetValue() xóa rỗng các ô nhập dữ liệu. Làm cho nút Lƣu, xóa, sửa, bỏ qua không kích đƣợc. Nút Thêm kích đƣợc.

Đoạn chƣơng trình cụ thể là:

private void btnLuu\_Click(object sender, EventArgs e)

{

if (txtMaCL.Text == "")

{

MessageBox.Show("Bạn phải nhập mã chất liệu"); txtMaCL.Focus();

}

else

{

if (txtTenCL.Text == "")

{

MessageBox.Show("Bạn phải nhập tên chất liệu"); txtTenCL.Focus();

}

else

{

DataTable dtChatLieu = dtBase.DocBang("Select \* from tblChatLieu where"+" MaChatLieu='"+(txtMaCL.Text).Trim()+"'");

if (dtChatLieu.Rows.Count > 0)

{

MessageBox.Show("Mã chất liệu này đã có, hãy nhập mã khác!"); txtMaCL.Focus();

}

else

{

dtBase.CapNhatDuLieu("insert into tblChatLieu values(N'" + txtMaCL.Text + "',N'" + txtTenCL.Text + "')");

MessageBox.Show("Bạn đã thêm mới thành công"); dgvChatLieu.DataSource = dtBase.DocBang("select \* from

tblChatLieu");

ResetValue(); btnXoa.Enabled = false;

btnSua.Enabled = false; btnLuu.Enabled = false; btnBoQua.Enabled = false; btnThemMoi.Enabled = true;

}

}

}

}

#### Viết sự kiện click cho nút Bỏ qua

Nút bỏ qua sẽ xóa rỗng tất cả các ô nhập dữ liệu, trạng thái các nút nhấn trở về nhƣ khi form mới đƣợc Load lên:

private void btnBoQua\_Click(object sender, EventArgs e)

{

ResetValue(); btnXoa.Enabled = false; btnSua.Enabled = false; btnLuu.Enabled = false; btnBoQua.Enabled = false; btnThemMoi.Enabled = true;

}

#### Viết sự kiện click cho nút Thoát

Trƣớc khi thoát hỏi ngƣời dùng có muốn thoát không? Nếu ngƣời dùng muốn thoát kiểm tra xem nếu ngƣời dùng đang thêm mới thì hỏi ngƣời dùng có lƣu bản ghi đang thêm không xong mới thoát.

Mã lệnh nhƣ sau:

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)

{

if (btnLuu.Enabled == true)

{

if (MessageBox.Show("Bạn có muốn Lưu lại chất liệu đang thêm mới không?", "Thông báo", MessageBoxButtons.YesNo, MessageBoxIcon.Question)

== System.Windows.Forms.DialogResult.Yes) btnLuu\_Click(sender, e);

else

this.Close();

}

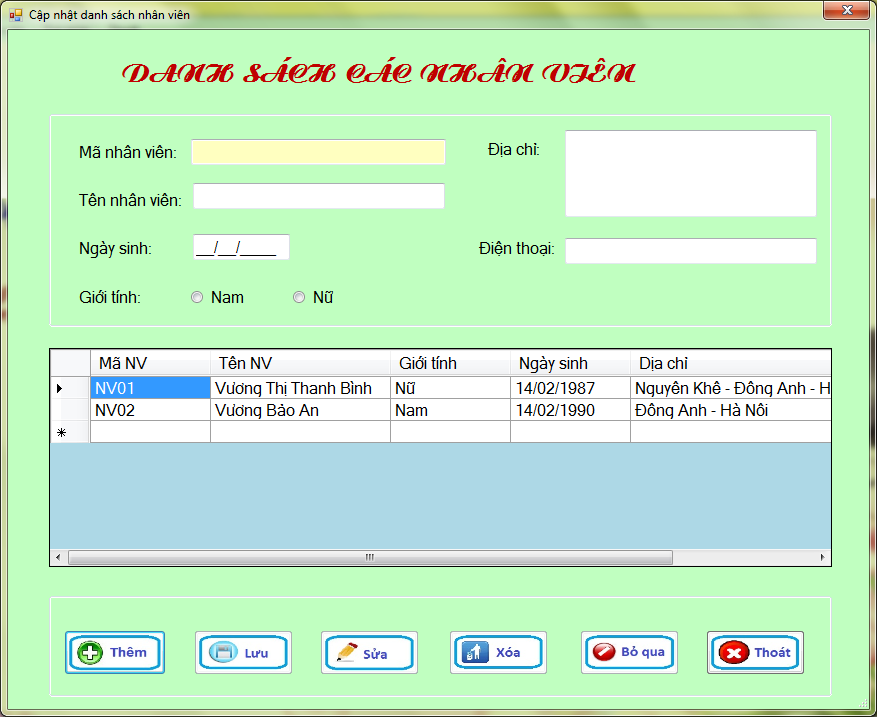
else

this.Close();

}

}

Tiếp tục bài tập 27, tạo form cập nhật danh mục Nhân viên có giao diện nhƣ sau:

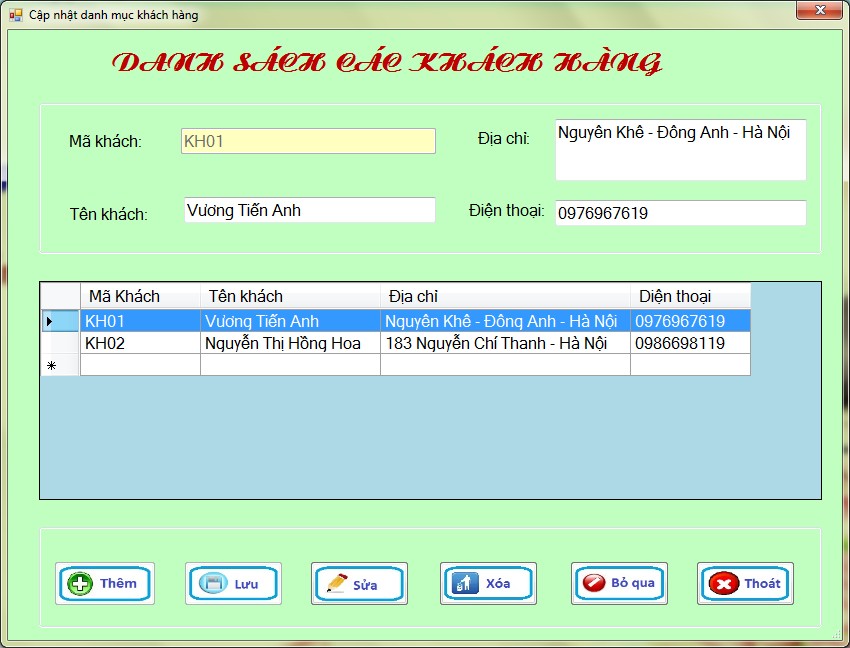


### Chú ý:

+ Điều khiển dùng để nhập ngày sinh của nhân viên là điều khiển MaskedTextBox. Để quy định nhập ngày tháng năm cho điều khiển này ta quy định thuộc tính ***Mask*** là *Short date*

+ Trƣớc khi lƣu vào CSDL cần kiểm tra xem ngày sinh ngƣời dùng nhập và có phải là kiểu ngày tháng không?

Tiếp tục bài tập 28, tạo form cập nhật danh mục Khách hàng có giao diện nhƣ sau:



### Bài tập 30:

Tiếp tục bài tập 29, tạo form cập nhật danh mục Hàng có giao diện nhƣ sau:



### Chú ý:

+ Khi kích vào nút Ảnh thì sẽ mở ra hộp thoại OpenFileDialog để ngƣời dùng chọn ảnh hiển thị cho sản phẩm.

+ Lƣu ảnh vào CSDL dƣới dạng đƣờng dẫn của ảnh.

+ Để lấy dữ liệu đổ ra một ComboBox, ta gán một DataTable chứa dữ liệu cho thuộc tính DataSource của ComboBox.

## Làm việc với Dataset và DataTale

* + 1. ***Cấu trúc của Dataset và DataTable***

Một DataSet tƣơng tự nhƣ một tập tin database vật lý hoàn chỉnh nhƣng đƣợc lƣu trong bộ nhớ. DataSet bao gồm các DataTable, DataTable bao gồm các DataColumn, DataRow, các constraint đƣợc minh họa nhƣ hình dƣới:

* + 1. ***Các phương thức***

Ở phần trên ta đã đề cập đến các phƣơng thức cơ bản của DataSet và DataTable. Ở phần này ta chỉ đề cập thêm một số phƣơng thức làm việc với các DataColum và DataRow của DataTable để phục vụ cho việc tự tạo một DataTable.

+ Dùng phƣơng thức Add(“Tên cột”, kiểu dl) để thêm cột mới vào DataTable

table.Columns.Add("Name", Type.GetType(“Tên\_kiểu”))

+ Dùng phƣơng thức newRow() để tạo một dòng dữ liệu mới vào DataTable

DataRow newR;

newR = tabes.NewRow();

+ Dùng phƣơng thức Add(newRow) để chèn dòng vừa tạo vào DataTable

table.Rows.Add(newR)

**hoặc**

table.Rows.Add(“DL\_cột 1”, “DL\_cột 2”,…, “DL\_cột n”)

* + 1. ***Tự tạo DataTable***

Ở các ví dụ ở trên hầu hết ta nạp dữ liệu cho DataTable là dữ liệu đƣợc lấy từ CSDL. Vì DataTable là một loại dữ liệu độc lập với server nên ta có thể tự tạo một DataTable thông qua các DataColum và DataRow của DataTable.

Kiểu dữ liệu DataColumn chứa đầy đủ các property cần thiết để bạn tạo ra một mô hình dữ liệu hoàn chỉnh cho DataTable.

**Ví dụ:** ta cần tạo một bảng DataTable có cấu trúc gồm 2 cột là: STT (có kiểu dữ liệu là số nguyên và tự động tăng) và cột thứ 2 là HoTen (có kiểu dữ liệu là String).

//Khai báo và khởi tạo một DataTable DataTable tables = new DataTable();

//Khai báo và khởi tạo một cột với tên là STT kiểu là số nguyên DataColumn col1 = new DataColumn("STT", Type.GetType("System.Int32"));

col1.AllowDBNull = false; // Cột không được phép Null col1.AutoIncrement = true ; //Giá trị tăng tự động col1.AutoIncrementSeed = 1; //Bắt đầu đánh số từ 1 col1.Unique = true ;//Giá trị là duy nhất tables.Columns.Add(col1) ; //Thêm cột STT vào bảng tables

//Chèn thêm cột HoTen

tables.Columns.Add("HoTen", Type.GetType("System.String"));

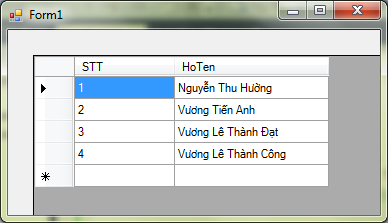
Sau khi tạo xong cấu trúc của bảng, Ta sẽ chèn dữ liệu vào bảng thông qua kiểu dữ liệu DataRow nhƣ sau:

//Khai báo một DataRow DataRow newR;

//Chèn dòng vào bảng newR = tables.NewRow();

//Chèn dữ liệu vào bảng. Ta không phải thêm vào cột STT vì nó tự động tăng bắt đầu từ 1 newR["HoTen"]= "Nguyễn Thu Hường"; //Thêm dữ liệu cho cột Hoten tables.Rows.Add(newR); //Chèn hàng vào bảng

//Chèn các hàng vào bảng tables.Rows.Add(null, "Vương Tiến Anh"); tables.Rows.Add(null, "Vương Lê Thành Đạt"); tables.Rows.Add(null, "Vương Lê Thành Công");

Thế là ta đã có một tables. Dùng DataGridView để hiển thị Table này lên ta sẽ có kết quả nhƣ sau :

Bảng dữ liệu mà ta tạo hoàn toàn không liên quan đến một dữ liệu nguồn nào cả.

## Xuất dữ liệu ra Excel

* + 1. ***Làm việc với đối tượng Excel***

Trong quá trình xây dựng ứng dụng, đôi khi ta muốn xuất dữ liệu ra một file excel, ví dụ nhƣ in danh sách nhân viên, danh sách khách hàng, hóa đơn, bảng điểm, bảng lƣơng, ...

Để làm điều này ta thực hiện các bƣớc nhƣ sau:

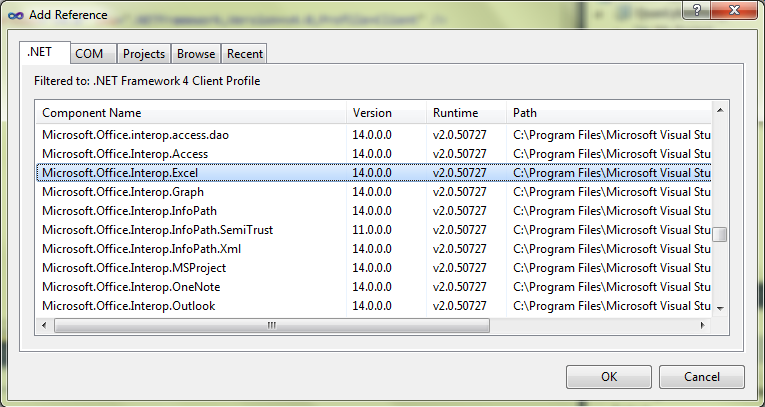
#### Bước 1. Tham chiếu đến file thư viện

Để xuất đƣợc dữ liệu ra file excel, ta phải tham chiếu (reference) đến đối tƣợng

**Microsoft.Office.Interop.Excel.dll** bằng thao tác:

+ Vào menu **Project/Add Reference…** xuất hiện hộp hội thoại Add Reference

+ Trong tab .NET chọn **Microsoft.Office.Interop.Excell** rồi nhấn OK nhƣ hình vẽ:



***Bước 2. Khai báo sử dụng đối tượng Microsoft.Office.Interop.Excel***

Phần khai báo này đặt ở đầu file .cs và đƣợc khai báo theo cú pháp sau:

using Excel = Microsoft.Office.Interop.Excel;

Với **Excel** là tên của đối tƣợng, ta có thể đặt một tên khác ở đây.

***Bước 3. Khai báo và khởi tạo các thành phần của đối tượng Microsoft.Office.Interop.Excel***

Đối tƣợng **Microsoft.Office.Interop.Excel** bao gồm các thành phần chính sau:

* Application (chƣơng trình excel)
* Workbook (file .xls, có nhiều workbook trong app)
* Worksheet (các trang bảng tính trong file .xls, có nhiều worksheet trong workbook)
* Range (Là một khối ô làm việc của Excel – nó cũng có thể chỉ là một ô) Ta khai báo và khởi tạo các đối tƣợng này theo cú pháp sau:

//Khai báo và khởi tạo các đối tượng

Excel.Application exApp = new Excel.Application();

Excel.Workbook exBook = exApp.Workbooks.Add(Excel.XlWBATemplate.xlWBATWorksheet); Excel.Worksheet exSheet = (Excel.Worksheet)exBook.Worksheets[1];

Excel.Range tenTruong = (Excel.Range)exSheet.Cells[1, 1]; //Đưa con trỏ vào ô A1

Với **Excel** là tên của đối tƣợng đã khai báo ở bƣớc 2.

#### Bước 4: Định dạng cho file Excel

Ta có một vài các thao tác định dạng cơ bản nhƣ sau:

#### Các định dạng liên quan đến font chữ

Ở đây ta làm việc với các thuộc tính của đối tƣợng Range. Ví dụ ta định dạng từ ô A1 đến B3 nhƣ sau:

tenTruong.Font.Size = 10; //Đặt cỡ chữ là 10

tenTruong.Font.Name = "Times new roman"; //Chọn font Times new roman

tenTruong.Font.Bold = true; //Định dạng kiểu font chữ là in đậm

tenTruong.Font.Color = Color.Blue; //Màu xanh da trời hoặc

exSheet.get\_Range(“A1”).Font.Color=Color.Blue;

hoặc dùng cú pháp

exSheet.get\_Range("A7:G7").Font.Bold = true;

#### Các định dạng khác

+ Chỉnh độ rộng của ô

tenTruong.ColumnWidth = 7

+ Trộn các ô lại với nhau

exSheet.get\_Range ("C8:E8”). Merge(true);

+ Căn lề cho các ô

exRange.Range["C2:E2"].HorizontalAlignment = Microsoft.Office.Interop.Excel.XlHAlign.xlHAlignCenter

Ở trên là ta căn giữa, ta có thể căn kiểu khác nhƣ:

* + Microsoft.Office.Interop.Excel.XlHAlign.*xlHAlignLeft*: Căn trái
  + Microsoft.Office.Interop.Excel.XlHAlign.*xlHAlignRight*: Căn phải

+ Viết nội dung cho các ô

exSheet.get\_Range("A7").Value = "STT"; hoặc

tenTruong.Value=”Nội dung”

+ Đặt tên cho sheet đang làm việc

exSheet.Name=”Name”

#### Bước 5: Kích hoạt cho file excel hoạt động

exBook.Activate();

#### Bước 6: Lưu file excel lại

Để lƣu file ta dùng lệnh:

exBook.SaveAs(đường dẫn file excel);

Ta dùng kèm theo hộp thoại ***SaveFileDialog*** để cho phép ngƣời dùng chọn nơi để lƣu file Excel.

Khi ấy đƣờng dẫn đƣợc lấy bằng thuộc tính FileName của hộp thoại lƣu file

exBook.SaveAs (dlgSaveFile.FileName.ToString());

#### Bước 7: Thoát khỏi ứng dụng

exApp.Quit()

* + 1. ***Ví dụ***

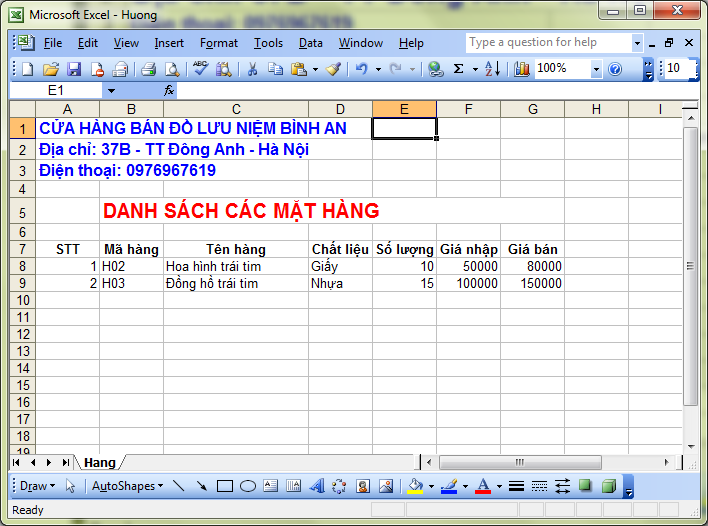
Tạo form tìm kiếm hàng với giao diện nhƣ sau:



Ngƣời dùng điền thông tin các tiêu chí cần tìm, việc tìm kiếm là tìm kiếm gần đúng (với từ khóa like), điều kiện tìm kiếm là điều kiện and.

Khi ngƣời dùng nhấn nút *Tìm kiếm* thì kết quả tìm kiếm hiện lên trên lƣới.

Khi ngƣời dùng chọn nút *In ra excel* thì kết quả tìm kiếm đƣợc in ra file Excel nhƣ sau:



**Hướng dẫn**

Tạo form frmTKHang với các điều khiển cơ bản sau:

|  |  |  |
| --- | --- | --- |
| **Loại** | **Tên điều khiển** | **Text** |
| TextBox | txtMaHang |  |
| txtTenHang |  |
| txtDonGiaBan1 |  |
| txtDonGiaBan2 |  |
| ComboBox | cboChatLieu |  |
| Button | btnTim | Tìm kiếm |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | btnIn | In ra Excel |  |
| DataGridView | dgvHang |  |
| SaveFileDialog | dlgSave |  |

Mở sự kiện Click của ComboBox cblChatLieu viết vào đoạn mã lệnh lấy ra danh sách chất liệu nhƣ sau:

private void cboChatLieu\_Click(object sender, EventArgs e)

{

cboChatLieu.DataSource = dtBase.DocBang("Select \* from tblChatLieu"); cboChatLieu.ValueMember = "MaChatLieu"; //Giá trị của phần tử cboChatLieu.DisplayMember = "TenChatLieu";//Chuỗi hiển thị của phần tử

}

sau:

Mở sự kiện Click nút Tìm kiếm viết vào đoạn mã thực hiện chức năng tìm kiếm nhƣ

private void btnTim\_Click(object sender, EventArgs e)

{

//Trường hợp không chọn tiêu chí tìm sẽ lấy ra tất cả các mặt hàng

sql = "select MaHang,TenHang,TenChatLieu,SoLuong,DonGiaNhap,DonGiaBan,"+ "tblHang.MaChatLieu from tblHang inner join tblChatLieu on "+ "tblHang.MaChatLieu=tblChatLieu.MaChatLieu where MaHang is not null";

//Khi chọn tiêu chí nào sẽ ghép với tiêu chí đó bằng từ and

//Tìm kiếm gần đúng với từ khóa like if (txtMaHang.Text != "")

sql = sql + " and MaHang like N'%"+txtMaHang.Text.Trim()+"%'"; if(txtTenHang.Text!="")

sql = sql + " and TenHang like N'%" + txtTenHang.Text.Trim() + "%'"; if(txtDonGiaBan1.Text!="")

sql = sql + " and DonGiaBan >= "+txtDonGiaBan1.Text; if (txtDonGiaBan2.Text != "")

sql = sql + " and DonGiaBan <= " + txtDonGiaBan2.Text; if(cboChatLieu.Text!="")

sql = sql + " and tblHang.MaChatLieu='" +

cboChatLieu.SelectedValue.ToString() + "'";

//Trình bày gridView

dtHang = dtBase.DocBang(sql); dgvHang.DataSource = null; dgvHang.DataSource = dtHang;

dgvHang.Columns[0].HeaderText = "Mã hàng"; dgvHang.Columns[1].HeaderText = "Tên hàng"; dgvHang.Columns[2].HeaderText = "Chất liệu"; dgvHang.Columns[3].HeaderText = "Giá nhập"; dgvHang.Columns[4].HeaderText = "Giá bán"; dgvHang.Columns[5].HeaderText = "Số lượng"; dgvHang.Columns[0].DataPropertyName = "MaHang"; dgvHang.Columns[1].DataPropertyName = "TenHang"; dgvHang.Columns[2].DataPropertyName = "TenChatLieu"; dgvHang.Columns[3].DataPropertyName = "DonGiaNhap"; dgvHang.Columns[4].DataPropertyName = "DonGiaBan"; dgvHang.Columns[5].DataPropertyName = "SoLuong";

sau:

}

Mở sự kiện Click nút *In ra Excel* viết vào đoạn mã thực hiện chức năng in ra Excel nhƣ

private void btnIn\_Click(object sender, EventArgs e)

{

if (dtHang.Rows.Count > 0) //TH có dữ liệu để ghi

{

//Khai báo và khởi tạo các đối tượng Excel.Application exApp = new Excel.Application(); Excel.Workbook exBook =

exApp.Workbooks.Add(Excel.XlWBATemplate.xlWBATWorksheet); Excel.Worksheet exSheet = (Excel.Worksheet)exBook.Worksheets[1];

//Định dạng chung

Excel.Range tenCuaHang = (Excel.Range)exSheet.Cells[1, 1]; tenCuaHang.Font.Size = 12;

tenCuaHang.Font.Bold = true; tenCuaHang.Font.Color = Color.Blue;

tenCuaHang.Value = "CỬA HÀNG BÁN ĐỒ LƯU NIỆM BÌNH AN";

Excel.Range dcCuaHang = (Excel.Range)exSheet.Cells[2, 1]; dcCuaHang.Font.Size = 12;

dcCuaHang.Font.Bold = true; dcCuaHang.Font.Color = Color.Blue;

dcCuaHang.Value = "Địa chỉ: 37B - TT Đông Anh - Hà Nội";

Excel.Range dtCuaHang = (Excel.Range)exSheet.Cells[3, 1]; dtCuaHang.Font.Size = 12;

dtCuaHang.Font.Bold = true; dtCuaHang.Font.Color = Color.Blue; dtCuaHang.Value = "Điện thoại: 0976967619";

Excel.Range header = (Excel.Range)exSheet.Cells[5, 2]; exSheet.get\_Range("B5:G5").Merge(true); header.Font.Size = 13;

header.Font.Bold = true; header.Font.Color = Color.Red; header.Value = "DANH SÁCH CÁC MẶT HÀNG";

//Định dạng tiêu đề bảng

exSheet.get\_Range("A7:G7").Font.Bold = true; exSheet.get\_Range("A7:G7").HorizontalAlignment =

Microsoft.Office.Interop.Excel.XlHAlign.xlHAlignCenter; exSheet.get\_Range("A7").Value = "STT"; exSheet.get\_Range("B7").Value = "Mã hàng"; exSheet.get\_Range("C7").Value = "Tên hàng"; exSheet.get\_Range("C7").ColumnWidth = 20; exSheet.get\_Range("D7").Value = "Chất liệu"; exSheet.get\_Range("E7").Value = "Số lượng"; exSheet.get\_Range("F7").Value = "Giá nhập"; exSheet.get\_Range("G7").Value = "Giá bán";

//In dữ liệu

for (int i = 0; i < dtHang.Rows.Count; i++)

{

exSheet.get\_Range("A"+(i + 8).ToString()+":G"+(i + 8).ToString())

.Font.Bold = false;

exSheet.get\_Range("A" + (i + 8).ToString()).Value = (i + 1)

.ToString();

exSheet.get\_Range("B" + (i + 8).ToString()).Value = dtHang.Rows[i]["MaHang"].ToString();

exSheet.get\_Range("C" + (i + 8).ToString()).Value = dtHang.Rows[i]["TenHang"].ToString();

exSheet.get\_Range("D" + (i + 8).ToString()).Value = dtHang.Rows[i]["TenChatLieu"].ToString(); exSheet.get\_Range("E" + (i + 8).ToString()).Value =

dtHang.Rows[i]["SoLuong"].ToString(); exSheet.get\_Range("F" + (i + 8).ToString()).Value =

dtHang.Rows[i]["DonGiaNhap"].ToString(); exSheet.get\_Range("G" + (i + 8).ToString()).Value =

dtHang.Rows[i]["DonGiaBan"].ToString();

}

exSheet.Name = "Hang";

exBook.Activate(); //Kích hoạt file Excel

//Thiết lập các thuộc tính của SaveFileDialog

dlgSave.Filter = "Excel Document(\*.xls)|\*.xls |Word Document(\*.doc)

|\*.doc|All files(\*.\*)|\*.\*"; dlgSave.FilterIndex = 1; dlgSave.AddExtension = true; dlgSave.DefaultExt = ".xls";

if (dlgSave.ShowDialog() == System.Windows.Forms.DialogResult.OK) exBook.SaveAs(dlgSave.FileName.ToString());//Lưu file Excel

exApp.Quit();//Thoát khỏi ứng dụng

}

else

MessageBox.Show("Không có danh sách hàng để in");

}

### Bài tập 31:

Xây dựng form *Hóa đơn bán* với các chức năng nhƣ giao diện sau:



### Yêu cầu:

+ Khi chọn Mã nhân viên, mã khách, mã hàng thì các thông tin liên quan của nhân viên, khách hàng và hàng tƣơng ứng hiện ra.

+ Khi nhấn nút Thêm mới thì mã hóa đơn tự động sinh ra.

+ Khi viết số lƣợng, giảm giá thành tiền đƣợc tự động cập nhật.

+ Khi lƣu một mặt hàng vào hóa đơn (mua hàng), số lƣợng đƣợc cập nhật tại danh mục hàng. Trƣớc khi mua cần kiểm tra số lƣợng có đủ không.

## In báo cáo với CrystalReport

Để có thể in dữ liệu, ta có thể dùng in ra excel nhƣ trên. Tuy nhiên ta có thể dùng CrystalReport. CrystalReport là một công cụ tạo báo cáo đơn giản nhƣng cũng khá hiệu quả.

 Crystal Report là công cụ để tạo các báo cáo cho các ứng dụng Windows application hoặc Web application. Có thể tích hợp file .rpt vào ứng dụng Windows application hoặc Web application để hiển thị report

 Crystal Report có thể hiển thị thông tin dƣới dạng bảng, đồ hoạ, biểu đồ, … có khả năng tính toán nhƣ tính tổng, trung bình, …

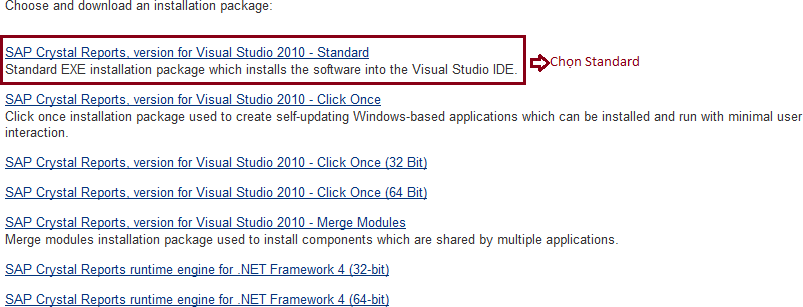
 Crystal Reports .NET đã tích hợp sẵn Report Designer trong bộ Visual Studio

.NET để có thể thiết kế report (file .rpt).

 Việc chạy các application có tích hợp Crystal Report đòi hỏi phải có Crystal Report Engine đƣợc cài đặt trên máy.

* + 1. ***Cài đặt Crystal report cho visual studio 2010***

Download bộ cài tại địa chỉ: [http://www.businessobjects.com/jump/...s2\_default.asp](http://www.businessobjects.com/jump/xi/crvs2010/us2_default.asp)



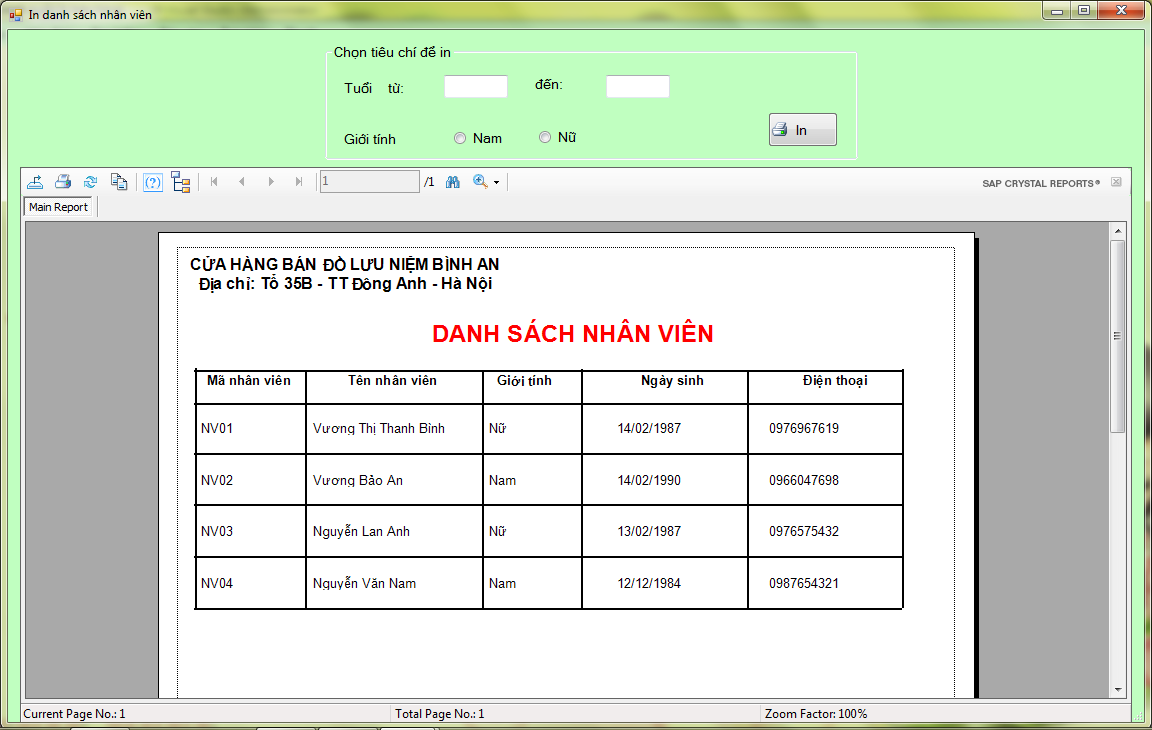
Sau khi download về, tiến hành cài đặt.

* + 1. ***Làm việc với CrystalReport thông qua DataSet***

Có nhiều cách để tạo report, tuy nhiên trong bài giảng này tôi hƣớng dẫn các bạn cách tạo report từ dataset.

Trƣớc khi tạo Crystal Report ta cần xác định xem hình dạng và dữ liệu cần in ra báo cáo gồm những cái gì. Ví dụ ta cần tạo một báo cáo để in danh sách nhân viên nhƣ hình vẽ sau:

Đầu tiên ta tạo giao diện cho form gồm các điều khiển sau:



Ta tiếp tục tạo thêm form mới đê in danh sách nhân viên trong project QuanLyBanHang và có giao diện nhƣ trên. Form gồm có các điều khiển chính nhƣ sau:

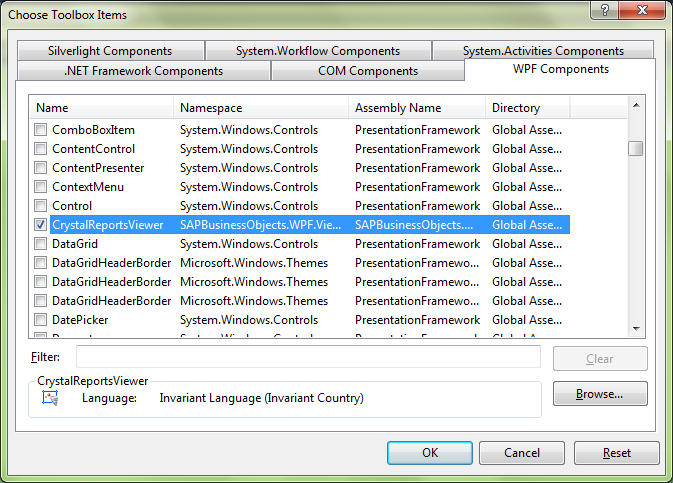
+ 2 TextBox: để nhập số tuổi có tên là txtTuoi1, txtTuoi2.

+ 2 RadioButton: để chọn giới tính là rdoNam và rdoNu.

+ 1 Button: btnIn

+ 1 CrystalReportViewer: dùng để hiển thị văn bản khi in có tên là rptvNhanVien. Muốn thêm điều khiển *CrystalReportViewer* ta kích chuột phải vào hộp ToolBox chọn

*Choose Item..* sau đó chọn đến thành phần CrystalreportViewer nhƣ sau:



**Chú ý:** Khi thêm điều khiển làm việc với Crystal Report vào chạy chƣơng trình sẽ báo lỗi về Namespace CrystalReportViewer… Ta sẽ sửa lại nhƣ sau: Kích chuột phải vào tên dự án chọn properties đổi Target Framework từ .NET Framework 4 Client Profile thành .NET Framework 4.

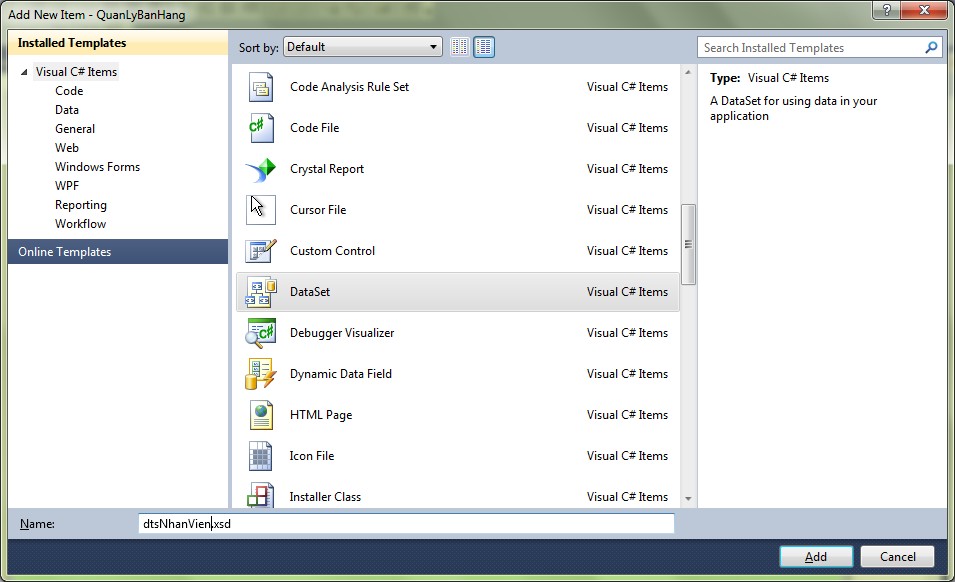
Để làm việc với Crystal Report ta sẽ khai báo sử dụng namespace sau:

using CrystalDecisions.CrystalReports.Engine;

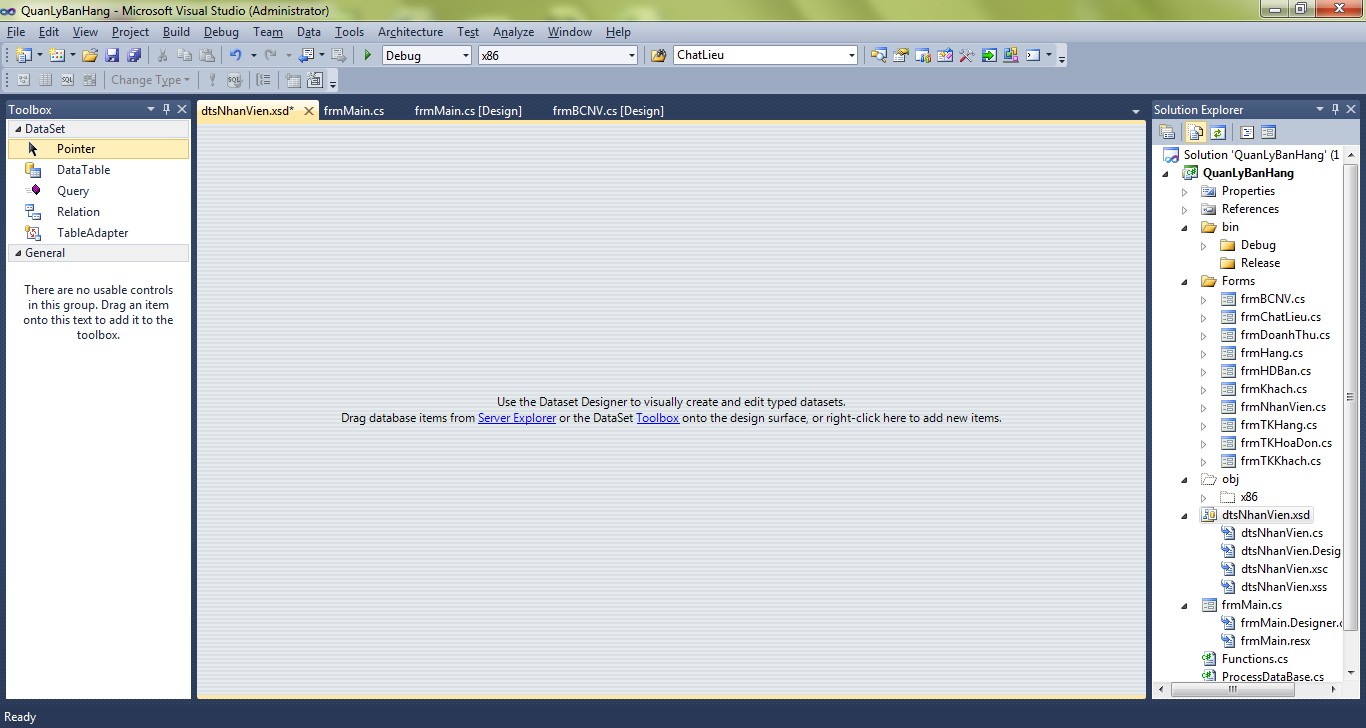
Để tạo một báo cáo từ DataSet ta thực hiện theo các bƣớc sau:

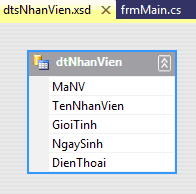
***Bước 1:*** Tạo DataSet

Kích chuột phải vào tên dự án chọn Add/New Item… sau đó chọn Dataset nhƣ hình sau và nhấn Add để thêm vào một dataset.



Sau khi chọn Add, xuất hiện giao diện nhƣ sau:



Bƣớc tiếp theo là tạo các datatable cho dataset, bằng cách kéo điều khiển DataTable trên thanh ToolBox vào. Ví dụ tạo một DataTable có cấu trúc nhƣ sau:

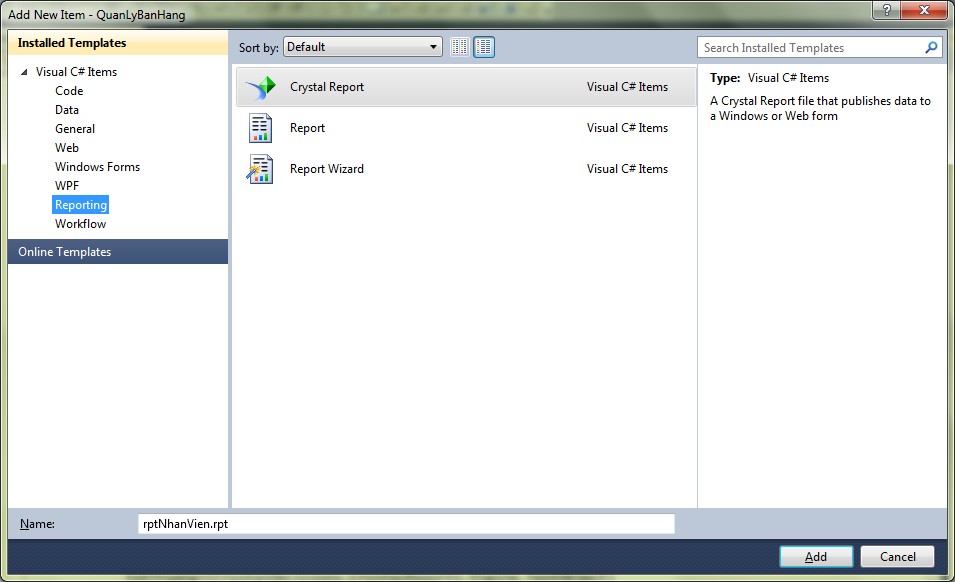
+ Để đổi tên của datatable, kích chuột phải vào tiêu đề của dataTable chọn Rename.

+ Để thêm một trƣờng (cột) dữ liệu mới, kích chuột phải chọn Add/Column.

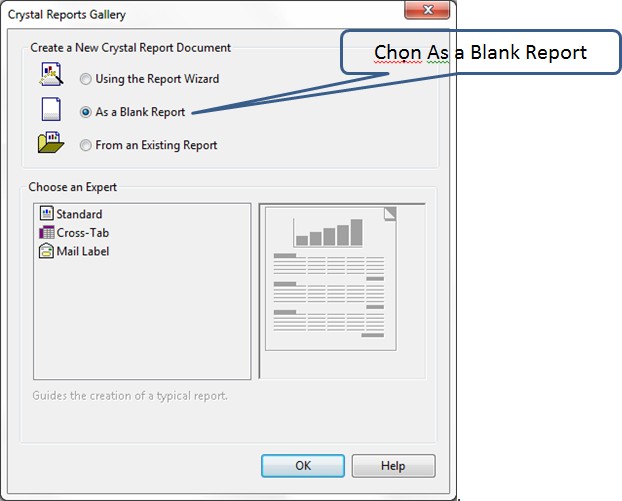
***Bước 2:*** Tạo report.

+ Tạo thƣ mục chứa các report của ứng dụng, đƣờng dẫn thƣ mục là:

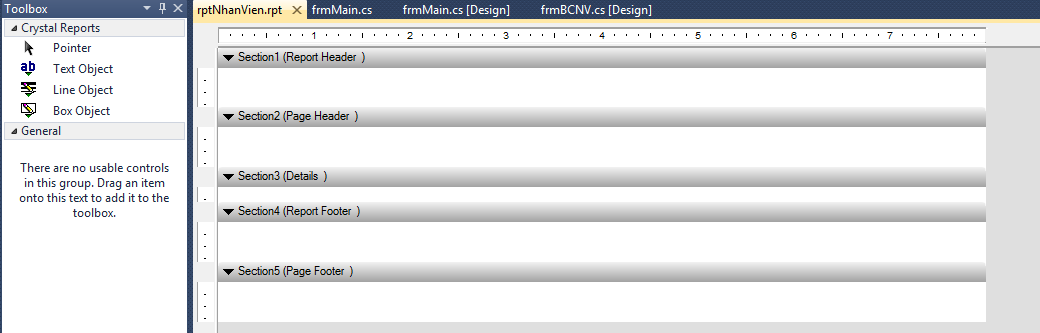
/bin/debug/Report/

+ Kích chuột phải vào thƣ mục chứa report, chọn ***Add/ New Item***… Sau đó trong nhánh ***Reporting*** chọn ***Crystal report***, đặt tên cho report là ***rptNhanVien.rpt*** rồi chọn ***Add*** nhƣ hình sau:

Xuất hiện cửa sổ sau, ta chọn nhƣ hình vẽ



Nhấn OK sẽ xuất hiện trên file *rptNhanVien.rpt* cùng với giao diện để thiết kế Crystal Report nhƣ sau:



### Cấu trúc của một Cystal report:

Ở hình vẽ trên ta thấy một crystal report sẽ có 5 section (từ section 1 đến section 5) với các ý nghĩa của từng section nhƣ sau:

 Section 1 (Report Header): Đƣợc hiển thị ở phần đầu báo cáo (đầu trang 1. không hiển thị ở các trang 2, 3, 4, …)

 Section 2 (Page Header): Đƣợc hiển thị ở đầu tất cả các trang báo cáo.

 Section 3 (Detail): Hiển thị phần thông tin dữ liệu, từng hàng dữ liệu sẽ đƣợc lặp lại.

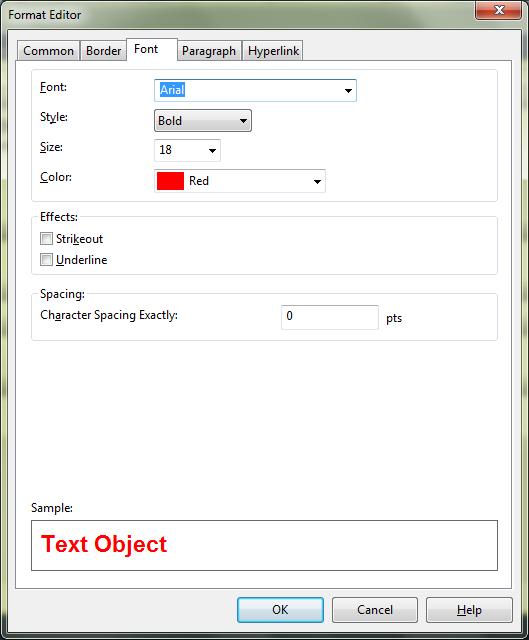
 Section 4 (Report Footer): Đƣợc hiển thị ở phần cuối báo cáo (ở cuối trang cuối cùng, không hiển thị ở các trang khác)

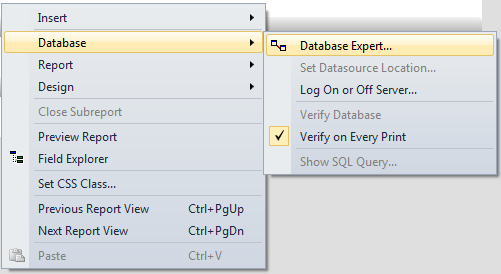
 Section 5 (Page Footer): Đƣợc hiển thị ở cuối tất cả các trang báo cáo.

*Bước 3:* Bắt đầu thiết kế Cystal Report.

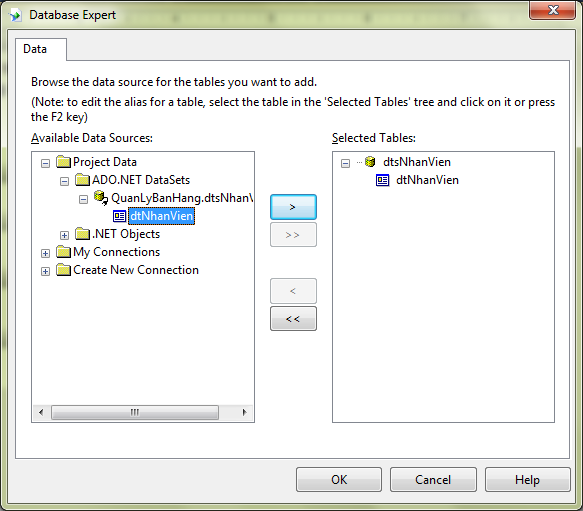
+ Sử dụng TextOject để tạo dòng chữ CỬA HÀNG BÁN ĐỒ LƢU NIỆM BÌNH AN, dòng chữ *Địa chỉ: Tổ 35B – TT Đông Anh – Hà Nội* và dòng chữ DANH SÁCH NHÂN VIÊN trên phần Section 1 (Report Header) nhƣ hình sau:

**Text Object** là đối tƣợng dùng để hiển thị một chuỗi chữ giống nhƣ Label. Để định dạng chữ cho Text Object ta kích chuột phải vào đối tƣợng Text Object cần định dạng chọn Format Object xuất hiện cửa sổ sau để ta định dạng:

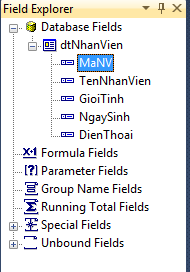


+ Bƣớc tiếp theo, ta bắt đầu chọn dữ liệu cho Crystal Report từ DataSet mà ta đã tạo ở bƣớc 1. Kích chuột phải vào màn hình thiết kế Report chọn Database/Database Expert…

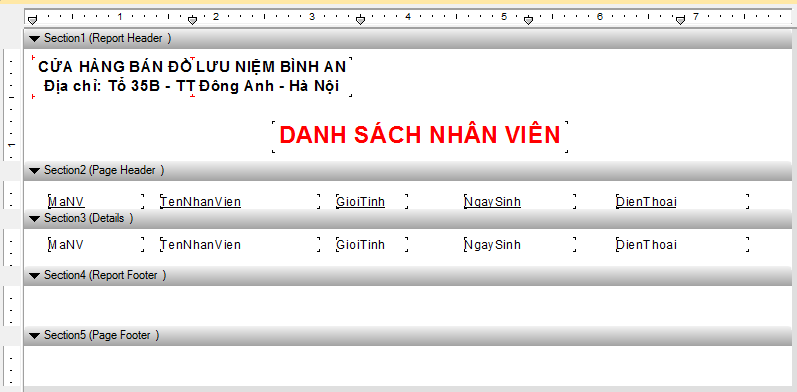
Xuất hiện cửa sổ chọn nguồn dữ liệu, ta chọn dataset vừa tạo ở bƣớc 1, nhƣ hình sau:



Nhấn OK, trên cửa sổ Field Explorer xuất hiện các trƣờng trong DataTable dtNhanVien đã chọn ở trên. Để mở ra cửa sổ Field Explorer ta kích chuột phải vào giao diện thiết kế report chọn Field Explorer.

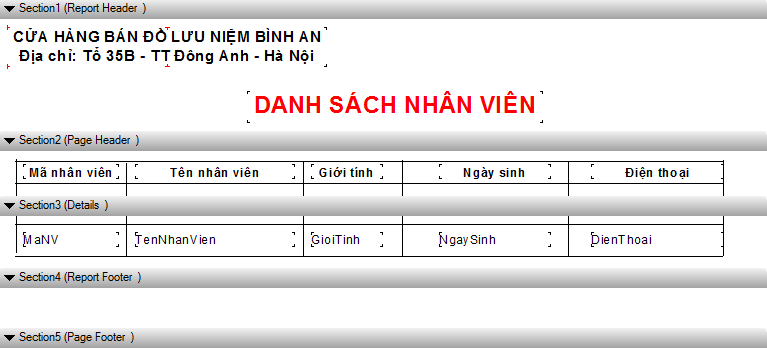


+ Lần lƣợt kéo các trƣờng dữ liệu cần hiển thị vào phần *Detail* của report.



Khi đó trên phần Page Header cũng xuất hiện ra các tiêu đề cột tƣơng ứng, để sửa lại các tiêu đề này ta kích chuột phải vào từng tiêu đề chọn Edit Text Object.

Để kẻ các đƣờng kẻ ngăn cách giữa các cột ta dùng đối tƣợng Line Object.



*Bước 4:* Lấy dữ liệu đổ vào DataTable dữ liệu nguồn của report, gắn Crystal report đã thiết kế ở bƣớc 2 vào CrystalReportViewer.

Với ví dụ trên, ta sẽ viết lệnh cho sự kiện click nút btnIn nhƣ sau:

private void btnIn\_Click(object sender, EventArgs e)

{

int NamSinh = DateTime.Today.Year; //Lấy ra năm hiện tại

//Xác định câu lệnh truy vấn

string sql = "select \* from tblNhanVien where MaNhanVien is not null "; if (txtTuoi1.Text != "")

{

int tuoi1 = Convert.ToInt16(txtTuoi1.Text);

sql = sql + " and Year(NgaySinh)<=" + (NamSinh-tuoi1).ToString() ;

}

if (txtTuoi2.Text != "")

{

int tuoi2 = Convert.ToInt16(txtTuoi2.Text);

sql = sql + " and Year(NgaySinh)>=" + (NamSinh - tuoi2).ToString();

}

if (rdoNam.Checked == true)

sql = sql + " and GioiTinh='Nam'"; if(rdoNu.Checked==true)

sql = sql + " and GioiTinh='Nữ'";

//Lấy dữ liệu từ câu lệnh SQL DataTable dtNV = dtBase.DocBang(sql);

//Khởi tạo đối tượng DataSet dtsNhanVien dtsNhanVien datasetNV = new dtsNhanVien();

//Đổ dữ liệu vào DataTable dtNhanVien đã tạo

//Dữ liệu được lấy từ DataTable chứa dữ liệu của lệnh sql trên DataRow dtrNew; //Khai báo đối tượng DataRow

for (int i = 0; i < dtNV.Rows.Count; i++)

{//Lần lượt thêm các dòng dữ liệu vào dtNhanVien của Dataset datasetNV dtrNew = datasetNV.Tables["dtNhanVien"].NewRow();

dtrNew["MaNV"] = dtNV.Rows[i]["MaNhanVien"].ToString(); dtrNew["TenNhanVien"] = dtNV.Rows[i]["TenNhanVien"].ToString(); dtrNew["GioiTinh"] = dtNV.Rows[i]["GioiTinh"].ToString(); dtrNew["NgaySinh"] =string.Format("{0:dd/MM/yyyy}",

Convert.ToDateTime( dtNV.Rows[i]["NgaySinh"].ToString())); dtrNew["DienThoai"] = dtNV.Rows[i]["DienThoai"].ToString(); datasetNV.Tables["dtNhanVien"].Rows.Add(dtrNew);

}

//Gọi đối tượng Report đã thiết kế là file rptNhanVien.rpt

string fileBaoCao = System.IO.Directory.GetCurrentDirectory() +

"//Report//rptNhanVien.rpt";

//Khai báo và khởi tạo đối tượng ReportDocument ReportDocument myReportDSNV = new ReportDocument();

//Load file rptNhanVien.rpt thông qua đối tượng ReportDocument myReportDSNV.Load(fileBaoCao);

//Đặt dữ liệu cho báo cáo chính là dataset chứa dữ liệu nhân viên

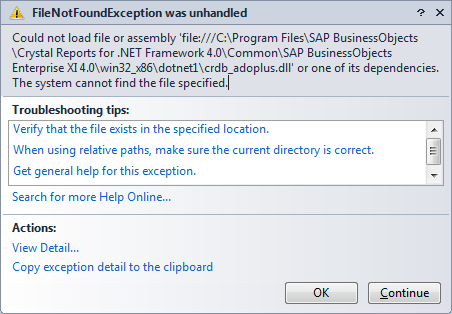
// datasetNV ở trên myReportDSNV.SetDataSource(datasetNV);

//Đặt nguồn cho điều khiển CrystalReportViewer là đối tượng ReportDocument

//ReportDocument đã chứa report đã tạo rptvNhanVien.ReportSource = myReportDSNV;

}

**Chú ý:** Khi chạy chƣơng trình ta sẽ thấy lỗi sau:



Ta xử lý nhƣ sau:

+ Mở file *app.config* thay nội dung của nó từ:

<?xml version="1.0"?>

<configuration>

<startup><supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/></startup></configuration>

Thành:

**<?xml version="1.0"?>**

**<configuration>**

**<startup useLegacyV2RuntimeActivationPolicy="true">**

**<supportedRuntime version="v4.0"/>**

**<requiredRuntime version="v4.0.20506"/>**

**</startup>**

**</configuration>**

# CHƯƠNG 7: PHÂN PHỐI VÀ CÀI ĐẶT ỨNG DỤNG

Khi đã hoàn tất chƣơng trình ta có thể đóng gói ứng dụng bằng cách tạo bộ cài đặt để đem đi cài trên máy khác đƣợc. Ở chƣơng này, chúng ta sẽ học cách tạo thêm dự án đóng gói cài đặt với các thƣ viện cần thiết.

## Đặt vấn đề

Cài đặt theo mô hình COM trƣớc kia có rất nhiều hạn chế nhƣ: các đăng ký DLL bị trùng lặp, không phù hợp phiên bản, xung đột. Một số chƣơng trình cài đặt đƣợc nhƣng không thực thi đƣợc, cài đặt xong không gỡ đƣợc hay gỡ nhƣng vẫn để lại rác…Giờ đây C# đã khắc phục những nhƣợc điểm đó vì nó có thể cài đặt mà không cần đăng ký vào Registry của hệ thống. Nó chủ yếu dựa trên bộ khung .NET Framework thay vì yêu cầu đối tƣợng COM gọi đến hàm API của Windows. Chƣơng trình sẽ đƣợc biên dịch thành các gói (Assembly) và chƣơng trình cài đặt sẽ kết hợp các thành phần lại cùng với các file thƣ viện cần thiết.

Một gói Assembly của ứng dụng .NET gồm 4 thành phần: ngôn ngữ diễn dịch trung gian MSIL (Microsoft Intermediate Language), mã (MSIL code), dữ liệu mô tả (metadata) và file chứa thông tin (manisfest file): các file hỗ trợ (support files) và tài nguyên (resource) dành cho chƣơng trình.

Hình sau minh họa chƣơng trình Luckyseven.exe đƣợc xem là một gói Assembly với 4 thành phần cơ bản:

|  |  |  |
| --- | --- | --- |
|  | **LuckySeven.EXE** |  |
| **MSIL Code** |
| **Metadata**   * Định nghĩa kiểu dữ liệu, phƣơng thức * Các tham chiếu về kiểu dữ liệu, đối tƣợng, phƣơng thức |
| **Manifest**   * Tên và phiên bản của chƣơng trình * Danh sách các file * Các tham chiếu đến gói (Assembly) khác * Các thông tin bảo mật * Các kiểu xuất (Exported) cho môi trƣờng sử dụng * Các thông tin về tài nguyên |
| **Resource**  - Các tài nguyên chƣơng trình cần dùng |

## Các cách đóng gói và triển khai ứng dụng

Khi bắt đầu chuẩn bị cho việc phân phối sản phẩm của mình, hãy xem xét các phƣơng thức cài đặt khác nhau:

* + - Cài dặt ứng dụng lên máy tính và đăng ký nó với Windows Registry.
    - Tạo chƣơng trình cài đặt cho phép chƣơng trình có thể cài từ mạng LAN hay Internet.
    - Đóng gói ứng dụng cho phép cài từ CD.
    - Đóng gói ứng dụng thành các file .CAB (một dạng file tự bung) có thể dùng cho các trình duyệt download về hay sao chép đi cài ở bất kỳ đâu.

Trong Visual studio (VS) có thể nhanh chóng tạo ra dự án đóng gói phân phối bằng trình Setup Winzard. Việc tùy biến dự án là hoàn toàn dễ dàng bằng cách thay đổi các tùy chọn. Nếu muốn đóng gói lên CD thì cần có đầu ghi CD.

Trên mỗi máy chạy chƣơng trình .NET cần có một bộ khung .NET FrameWork. Bạn có thể download bộ cài là file Dotnetfx.exe (20MB) về cài đặt. Sau này trong các phiên bản hệ điều hành sẽ chứa sẵn .NET Framework Runtime. Nếu máy đã có sẵn thì việc cài đặt chỉ đơn giản là sao chép và chạy.

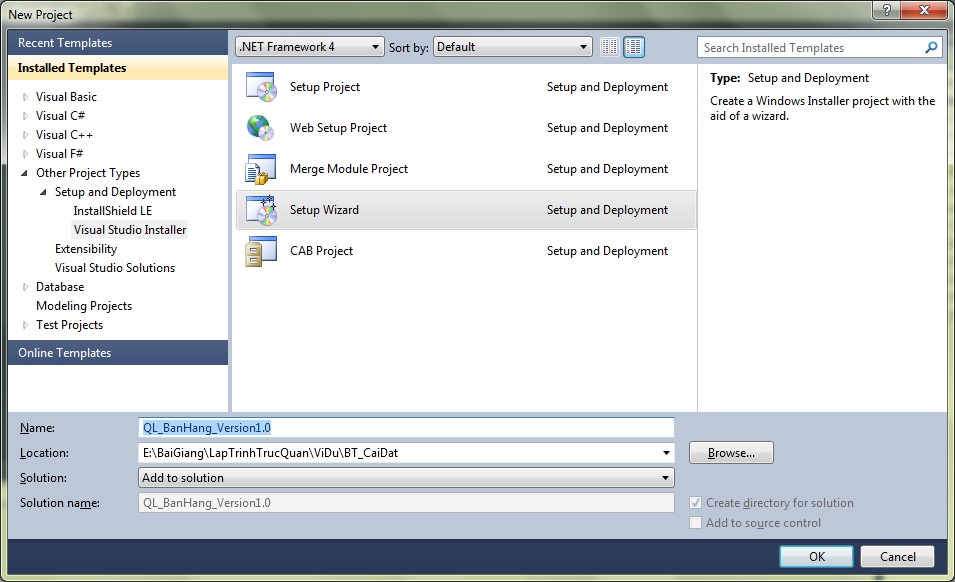
Tuy nhiên hầu nhƣ khi đóng gói VS đã nhúng luôn thƣ viện .NET runtime kèm theo chƣơng trình nên nếu dung lƣợng bộ cài có lớn hơn nhiều so với chƣơng trình thì bạn cũng đừng ngạc nhiên.

## Tạo dự án phân phối

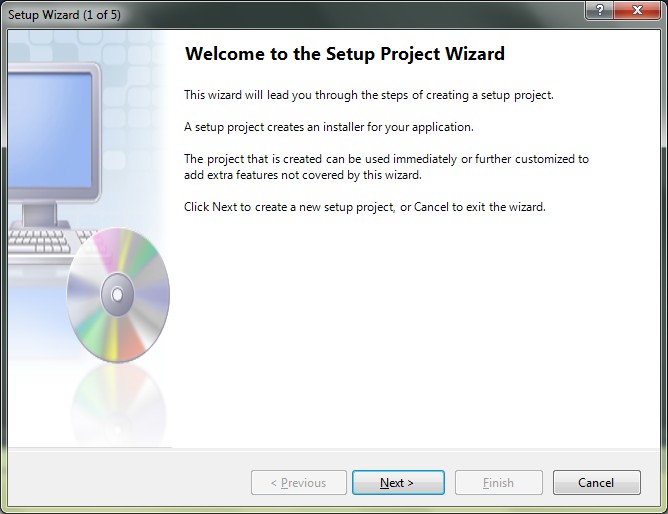
Bây giờ chúng ta sẽ tạo dự án Deployment. Chƣơng trình setup của chúng ta sẽ tạo một short cut icon cho chƣơng trình trên menu Start | Programs của windows, thêm file readme vào thƣ mục C:\Program Files \ microsoft press\QL\_BanHang, đăng ký với Registry để cuối chƣơng tiến hành gỡ bằng chức năng Add/Remove Programs.

Bạn làm theo các bƣớc sau đây:

* Tạo một thƣ mục bất kỳ có tên BT\_CaiDat, sau đó copy toàn bộ dự án QuanLyBanHang đã làm ở chƣơng 6 vào thƣ mục này.
* Mở dự án QuanLyBanHang bằng Visual Studio và chọn File | New | Project từ menu File. Xuất hiện giao diện sau:

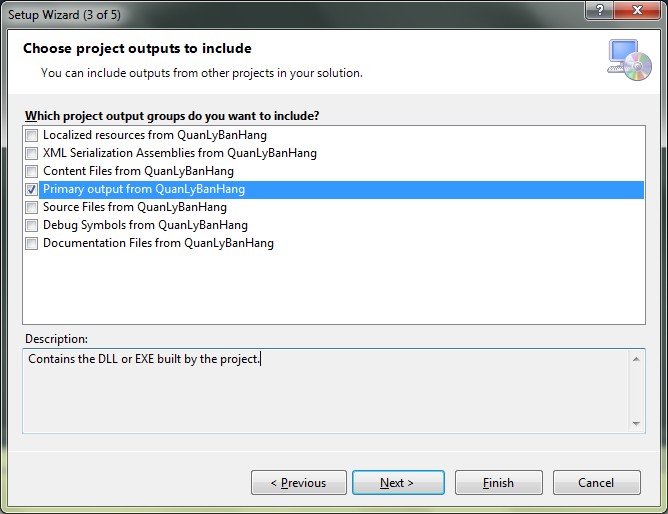


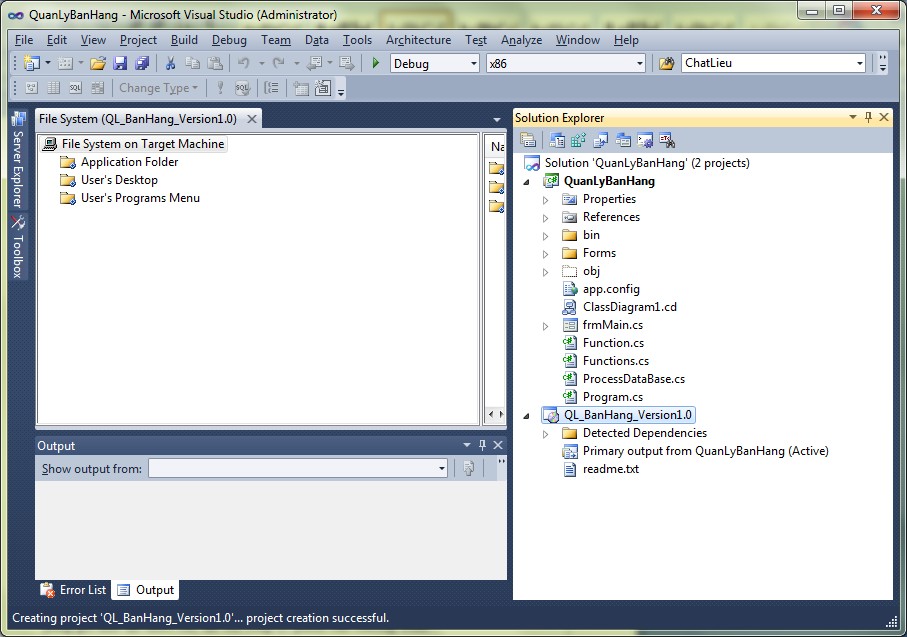
* Chọn Other Project Types/Visual studio Installer chọn Setup and Deployment. Có 5 mẫu phía bên phải khung cho phép lựa chọn. Mẫu *CAB project* cho phép đóng gói chƣơng trình thành nhiều gói .cab. Mẫu *Merge Module Project* cho phép đóng gói chung các dự án khác nhau (tạo file .msm có thể trộn tiếp vào các gói cài đặt khác). Mẫu *Setup Project* tạo bộ đóng gói cài đặt bởi Windows Installer. Mẫu *Web Installer* cài đặt chƣơng trình từ một Webserver…Có lẽ chúng ta sẽ chọn mẫu *Setup Winzard*, mẫu này đóng gói chƣơng trình thông qua các câu hỏi và yêu cầu thực hiện một số bƣớc cấu hình đơn giản.
* Nhấn vào biểu tƣợng Setup Wizard và gõ vào chuỗi QL\_BanHang\_Version1.0 và chọn đƣờng dẫn đến Sulution QuanLyBanHang.
* Chọn *Add To Solution* từ danh sách Solution để kết hợp hai dự án cùng lúc vào giải pháp hiện hành.
* Nhấn OK để làm xuất hiện trình Setup Winzard.



### Sử dụng trình Setup Winzard

1. Nhấn nút Next ở màn hình giới thiệu
2. Hộp thoại *Choose a Project Type* xuất hiện. Hộp thoại này cho phép ta kiểm soát cách đóng gói và phân phối dự án chƣơng trình. Trong bài tập này ta để mặc định là lựa chọn đầu tiên “Creat a setup for a windows application”
3. Nhấn nút Next. Hộp thoại *Choose Project Output To Include* hiển thị cho phép chỉ định file kèm theo trình cài đặt và các thông tin về chƣơng trình.
4. Nhấn chọn Primary Output nhƣ hình:



1. Nhấn Next. Hộp thoại *Choose Files To Include* hiển thị cho phép chọn các files dữ liệu, file hƣớng dẫn…
2. Nhấn nút Add và chọn file readme.txt trong dự án.
3. Nhấn Next để hiển thị hộp thoại *Creat Project*. Tại đây các thông tin về dự án đóng gói đã chọn. Bạn có thể nhấn nút Previous để hiệu chỉnh thông tin trƣớc khi hoàn thiện.
4. Nhấn nút Finish để tạo dự án **Phân phối** cho chƣơng trình QuanLyBanHang. Lúc này dự án mới đóng gói sẽ đƣợc liệt kê trong cửa sổ Solution Explorer. Bộ soạn thảo *File System Editor* cũng xuất hiện. Bạn có thể sử dụng *File System Editor* để thêm vào các file kết xuất, cùng các mục khác để xác định dự án có cần cài thêm những file phục lên máy khách trong quá trình cài đặt hay không. File System Object hiển thị danh sách các folder chuẩn sẽ đƣợc sử dụng hay tạo ra khi cài dặt. Bạn có thể tùy biến những folder này và thêm vào những folder đặc biệt khác tùy theo mục đích của chƣơng trình. Bạn cũng có thể yêu cầu File System Editor tạo short cut cho chƣơng trình. Bạn hãy xem qua các file chúng ta đã đính kèm và các file phụ thuộc vào thƣ viện thực thi .Net Framework trong folder *Detected Dependencies*, nơi chứa file .exe (Primary Output) nhƣ hình:
5. Tùy biến các lựa chọn đóng gói. Phần này chúng ta sẽ tách riêng trong mục 7.4.

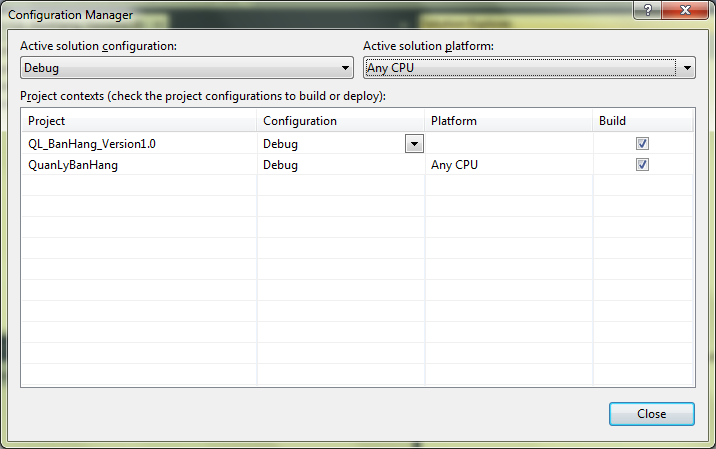
## Tùy biến các lựa chọn đóng gói

Bây giờ dự án của chúng ta đã sẵn sàng đóng gói. Khi có yêu cầu đóng gói, các file sẽ kết xuất ra thƣ mục chúng ta đã chọn và lƣu trong file .msi (Microsoft Installer). Bạn có thể cài đặt từ file này.

Tuy nhiên chúng ta sẽ tạo thêm một số tùy chọn cho chƣơng trình cài đặt trƣớc khi yêu cầu đóng gói nhƣ tạo short cut, tạo tên công ty, phiên bản chƣơng trình…

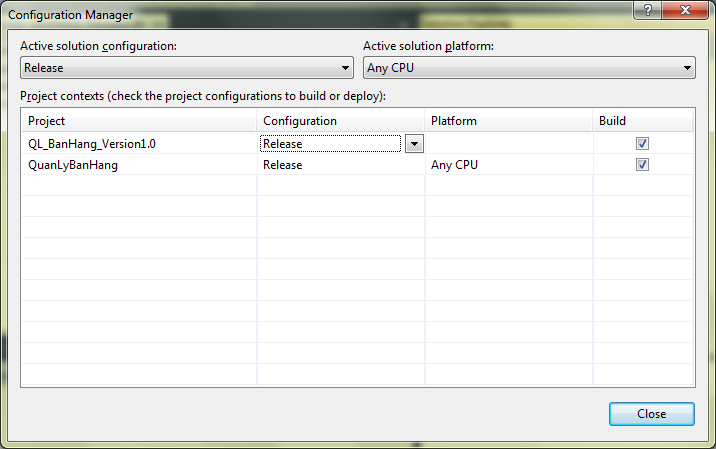
* + 1. ***Cấu hình các thiết lập***

1. Chọn Build | Configuration Manager từ menu làm xuất hiện hộp thoại sau:



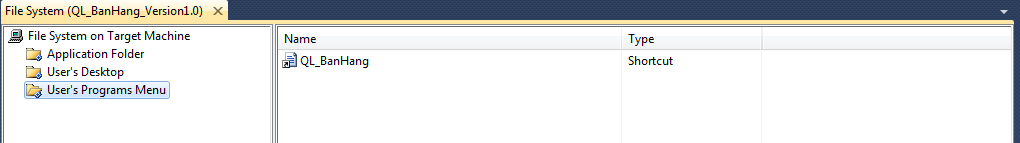
Hộp thoại này cho phép hiển thị chế độ biên dịch hiện hành cho các dự án đang có trong Solution. Lúc này chúng ta nên chọn là Release – biên dịch tối ƣu thay cho Debug.

1. Nhấn chọn danh sách Active Solution Configuration và chọn là Release
2. Nhấn chọn danh sách Configuration cho cả hai dự án QuanLyBanHang và QL\_BanHang\_Version1.0 và chọn Release.
3. Đánh dấu chọn mục Buil cho cả hai dự án.
4. Nhấn nút Close.



Tiếp theo chúng ta sử dụng *File System Editor* để yêu cầu trình cài đặt tạo short cut cho chƣơng trình sau khi cài đặt thành công.

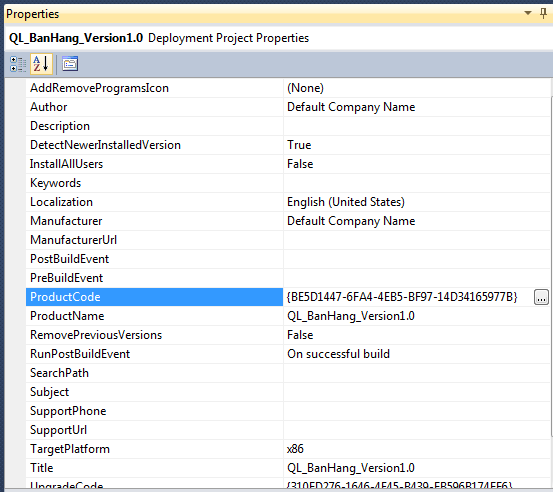
* + 1. ***Tạo short cut cho ứng dụng cài đặt***

1. Chọn *Application Folder* trong khung trái cửa sổ *File System Editor*. Nếu File System Editor chƣa xuất hiện bạn có thể cho nó xuất hiện bằng cách chọn dự án trong Solution Explorer và chọn *View | Editor | File System*.
2. Bên khung phải bạn nhắp chuột phải vào mục Primary Output From QuanLyBanHang và chọn Creat Shortcut To Primary Output From QuanLybanHang từ menu.
3. Đổi tên thành QL\_BanHang và ấn Enter.
4. Kéo Shortcut QL\_BanHang vào trong thƣ mục User‟s Programs Menu Folder bên khung trái nhƣ hình:

Tiếp theo chúng ta sẽ tìm hiểu thêm về cách tùy biến thông tin về tên công ty và phiên bản chƣơng trình.

* + 1. ***Thiết lập tên công ty và phiên bản chương trình***

1. Chọn dự án QL\_BanHang\_Version1.0 từ cửa sổ Solution Explorer.
2. Mở cửa sổ Properies của nó.



Thuộc tính Author thƣờng dùng đặt tên cho tác giả hay nhà sản xuất. Tên này cũng đƣợc dùng làm thƣ mục cho chƣơng trình đặt trong Program Files. Khi chƣơng trình Setup thực thi, thông tin về tác giả sẽ đƣợc đặt trong trƣờng Contact của hộp thoại Support Info, hộp thoại này bạn có thể truy xuất sau đó thông qua mục Add/Remove Programs trong Control Panel.

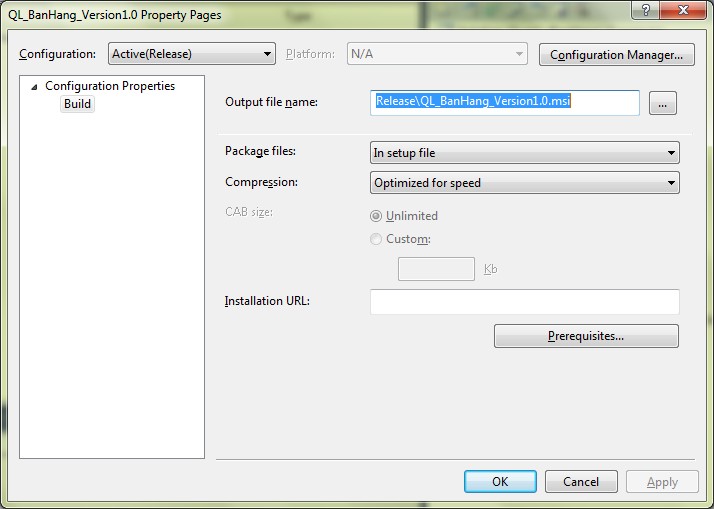
Thuộc tính Tiile chứa tên chƣơng trình setup, thuộc tính Version chứa số hiệu phiên bản cho chƣơng trình.

1. Thay đổi thuộc tính Author thành ThuHuong
2. Đổi Version thành 1.0
3. Ấn Yes khi có hộp thoại xuất hiện
4. Xem lại các thuộc tính khác nữa trƣớc khi đóng gói thực sự.

Bây giờ hãy thử mở hộp thoại Property Pages bằng cách R-Click vào dự án chọn Properties xem các thông tin thiết lập có đúng hay không.

***7.4.1. Đặt các thuộc tính cho gói ứng dụng***

Bạn làm xuất hiện cửa sổ Property Pages nhƣ hình:



Hộp thoại này cho phép quyết định xem lại những gì đã tùy chọn trong trình Setup Winzard và tùy biến số ít các thiết lập bổ sung không có trong winzard.

* Output File name cho phép đặt tên cho file cài đặt.
* Danh sách Package files: có 3 tùy chọn là As Loose Uncompress Files – tạo ra các file không nén trong cùng thƣ mục với file .msi; mục In Setup File lúc này chọn mặc

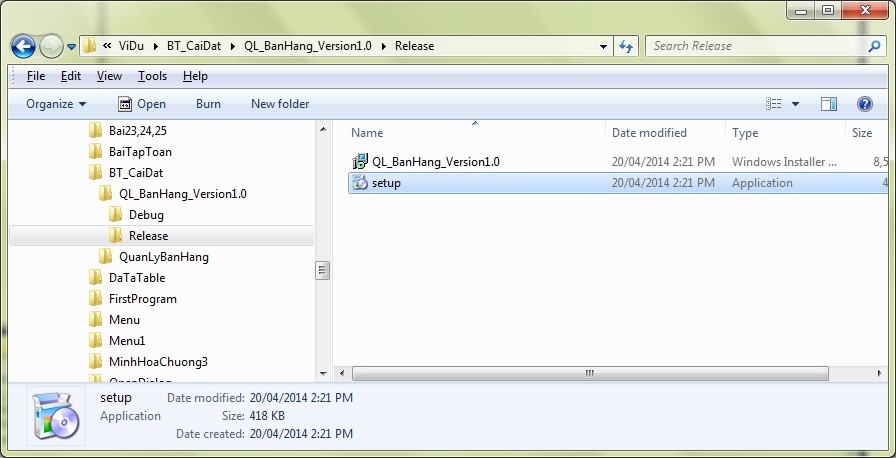
định do lựa chọn của chúng ta trong quá trình dùng Setup Winzard; mục In Cabinet Files – tạo ra một hay nhiều file .CAB là các file chứa ứng dụng của bạn trong đó.

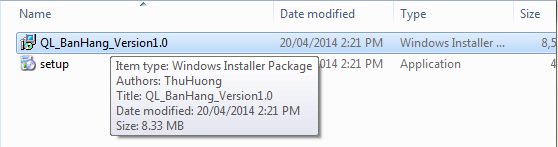
* Chọn mục In Cabinet Files, bạn có thể tùy chỉnh các thông số tƣơng ứng
* Nhấn chuột trở lại và lại chọn In Setup File. Chúng ta sẽ tạo ra một file cài đặt đơn chứa tất cả các file yêu cầu của chƣơng trình.
* Nhấn OK để lƣu lại các thay đổi của bạn trong hộp thoại Property Pages.

## Biên dịch và đóng gói dự án – kiểm tra việc cài đặt

Bây giờ dự án đã sẵn sàng để biên dịch, các bƣớc thực hiện nhƣ sau:

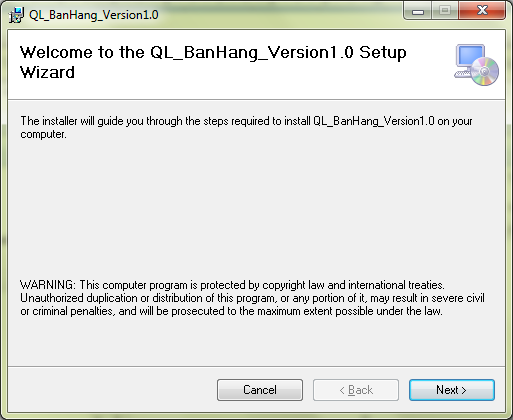
* Biên dịch bằng lệnh Build | Build Solution. Quá trình diễn ra hơi lâu. Bạn để ý thanh theo dõi tiến trình  xem quá trình biên dịch dự án. Nếu biên dịch thành công thì một thông báo hiện ở cuối góc trái màn hình .
* Chạy chƣơng trình Setup để cài đặt ứng dụng. Bạn mở thƣ mục chứa dự án đã chọn và tìm file cài đặt:



Khi bạn chọn file QL\_BanHang\_Version1.0 thì thông tin cấu hình trong phần trƣớc sẽ hiện lên nhƣ thế này:

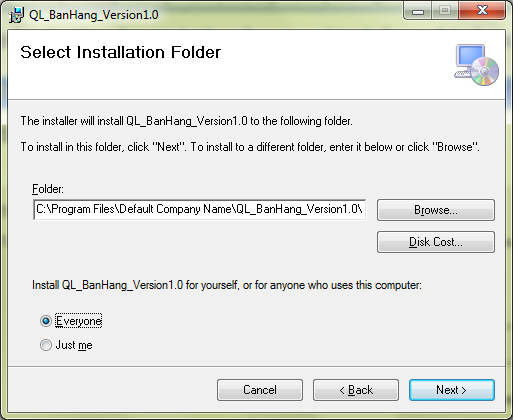
Kiểm tra quá trình cài đặt xem các file có đƣợc cài đủ và đúng vị trí hay không:

* + 1. ***Chạy chương trình cài đặt Setup***

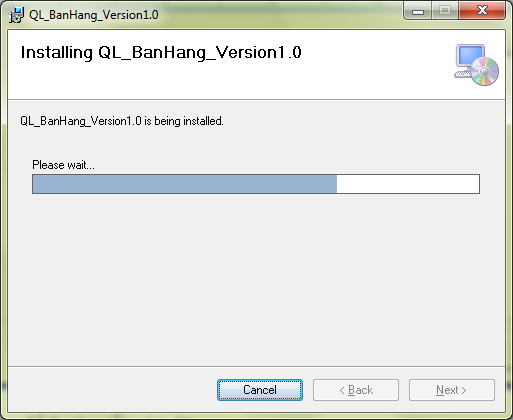
Kích đúp chuột vào file setup.exe để khởi động chƣơng trình cài đặt, sau một lúc màn hình chào mừng hiện ra nhƣ sau:

Nhấn Next để tiếp tục cài đặt. Hộp thoại Select Installation Folder yêu cầu nhập thông tin về thƣ mục cài đặt và các tùy chọn bổ sung.

Để mặc định các tùy chọn trừ tùy chọn ngƣời sử dụng thì chọn Everyone. Nhấn Next



Nhấn nút Next để bắt đầu cài đặt.



Nhấn Close để hoàn tất cài dặt.

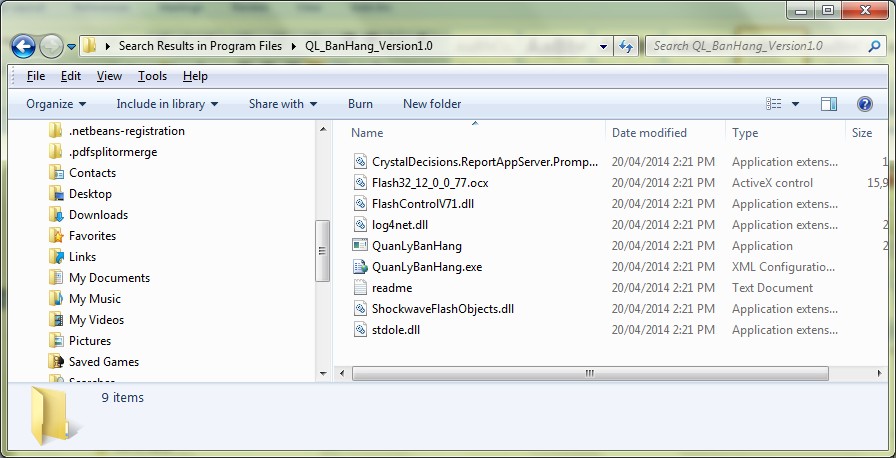
* + 1. ***Chạy chương trình sau khi cài đặt***

Chọn Start | Program và chọn Shortcut QL\_BanHang. Chƣơng trình khởi động thành công. Bạn chạy thử chƣơng trình xem có gặp lỗi gì không.

Nhƣ vậy là chúng ta đã đóng gói thành công một ứng dụng C# bằng Visual studio 2010.

## Tìm hiểu các file setup và gỡ chương trình

* + 1. ***Kiểm tra file cài đặt***

Bạn mở thƣ mục chứa phần cài đặt trong Program Files và xem các file mà chƣơng trình đã cài vào.

* + 1. ***Tháo gỡ chương trình***

Bạn tháo gỡ chƣơng trình nhƣ mọi chƣơng trình khác.

