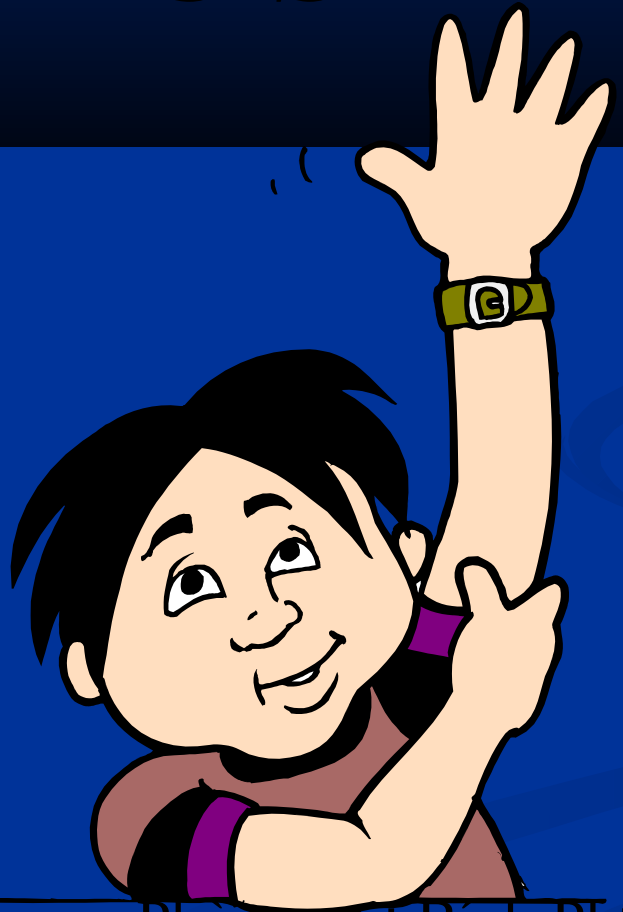
# NGÔN NGỮ LẬP TRÌNH C SHARP



# TÌM HIỂU VỀ C SHARP

***Trước khi tìm hiểu C# chúng ta xem một số những khái niệm sau đây:***

- Thứ nhất, LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (OPP). Lập trình hướng đối tượng là kỹ thuật lập trình hỗ trợ công nghệ đối tượng. OPP được xem là giúp tăng năng xuất, đơn giản hóa độ phức tập khi bảo trỉ cũng như mở rộng phần mềm bằng cách cho phép lập trình viên tập trung vào các đối tượng phần mềm ở bậc cao hơn. Ngoài ra, nhiều người còn cho rằng OPP dex tiếp thu hơn cho những người mới học về lập trình hơn là các phương pháp trước đó.

# TÌM HIỂU VỀ C SHARP

- Một cách giản lược đây là khái niệm và là nỗ lực nhằm giảm nhẹ các thao tác viết mã cho người lập trình, cho phép họ thao tác các ứng dụng mà các yếu tố bên ngoài có thể tương tác với các chương trình đó giống như là tương tác với các đối tượng vật lý.
- Những đối tượng trong một ngôn ngữ OPP là các kết hợp giữa mà và dữ liệu mà chúng được nhìn nhận như là một đơn vị duy nhất. Mỗi đối tượng có một tên riêng biệt và tất cả đều tham chiếu đến đối tượng đó và tiến hành thông qua chính tên nó. Như vậy, mỗi đối tượng có khả năng nhận thông báo, xử lý dữ liệu (bên trong của nó), và gửi ra hay trả lời đến các đối tượng khác hay môi trường.

# TÌM HIỂU VỀ C SHARP

- Thứ hai, NGÔN NGỮ LẬP TRÌNH C. C là ngôn ngữ lập trình tương đối nhỏ vận hành gần với phần cứng và nó gần với ngôn ngữ Assembler hơn hầu hết các ngôn ngữ bậc cao. Hơn thế, C đôi khi được đánh giá như là "có khả năng di động", cho thấy sự khác nhau quan trọng giữa nó và các ngôn ngữ bậc thấp hơn như Assembler, đó là việc mã C có thể dịch và thi hành trong hầu hết các máy tính, hơn hẳn các ngôn ngữ hiện tại trong khi đó Assembler chỉ có thể chạy trong một số máy tính đặc biệt. Vì lý do này mà C được xem là ngôn ngữ bậc trung.

# TÌM HIỂU VỀ C SHARP

- C đã được tạo ra với một mục tiêu là làm cho nó thuận tiện để viết các chương trình lớn với số lỗi ít hơn trong mẫu hình lập trình thủ tục mà lại không đặt gánh nặng lên vai người viết ra trình dịch C, là những người bề bộn với các đặc tả phức tạp của ngôn ngữ.

# TÌM HIỂU VỀ C SHARP

- Ngôn ngữ C# trong ứng dụng . NET có các tính năng vượt trội hơn so với C. Hay nói cách khác C# là cuộc cách mạng của ngôn ngữ lập trình Microsoft C và Microsoft C++ với tính năng đơn giản, hiện đại, hướng đối tượng và có độ bảo mật cao.

# Phần I: Tìm hiểu ngôn ngữ C Sharp (C#)

- Ngôn ngữ C# khá đơn giản, chỉ khoảng 80 từ khóa và hơn mười mấy kiểu dữ liệu được xây dựng sẵn. Tuy nhiên ngôn ngữ C# có ý nghĩa cao khi nó thực thi những khái niệm lập trình hiện đại. C# bao gồm tất cả các hỗ trợ cho cấu trúc, thành phần Component, lập trình hướng đối tượng. Những tính chất đó hiện diện trong ngôn ngữ lập trình hiện đại. Và ngôn ngữ C# hội tụ những điều kiện như vậy. Hơn nữa, nó được xây dựng trên nền tảng của hai ngôn ngữ mạnh nhất là C++ và Java.

# Phần I: Tìm hiểu ngôn ngữ C Sharp (C#)

- Ngôn ngữ C# được phát triển bởi đội ngũ kỹ sư của Microsoft, trong đó người dẫn đầu là Anders Hejlsberg và Scott Wiltamuth. Cả hai người này đều là những người nổi tiếng, trong đó Anders Hejlsberg được biết đến là tác giả của Torbo Pascal, ngôn ngữ lập trình PC phổ biến. Và ông đứng đầu nhóm thiết kế Borland Delphi, một trong những thành công đầu tiên của việc xây dựng môi trường phát triển tích hợp IDE cho lập trình Client/Server.

# Phần I: Tìm hiểu ngôn ngữ C Sharp (C#)

- Phần cốt lõi hay còn gọi là trái tim của bất kỳ ngôn ngữ lập trình hướng đối tượng nào là sự hỗ trợ của nó cho việc định nghĩa và làm việc với những lớp. Những lớp thì định nghĩa những kiểu dữ liệu mới, cho phép người phát triển mở rộng ngôn ngữ để tạo mô hình tốt hơn để giải quyết vấn đề. Ngôn ngữ C# chứa những từ khóa cho việc khai thác những kiểu lớp đối tượng mới và những phương thức hay thuộc tính của lớp, và cho việc thực thi đóng gói, kế thừa và đa hình, ba thuộc tính cơ bản của bất kỳ ngôn ngữ lập trình hướng đối tượng.

# Phần I: Tìm hiểu ngôn ngữ C Sharp (C#)

- Trong ngôn ngữ C#, mọi thứ liên quan đến khai báo lớp đều được tìm thấy trong phần khai báo của nó. Định nghĩa một lớp trong C# không đòi hỏi phải chia ra tập tin header và tập tin nguồn giống như ngôn ngữ C++. Hơn thế nữa, C# hỗ trợ kiểu XML, cho phép chèn các tag XML để phát sinh tự động các document cho lớp.

# Phần I: Tìm hiểu ngôn ngữ C Sharp (C#)

- C# cũng hỗ trợ giao diện Interface, nó được xem như một cam kết với một lớp cho những dịch vụ mà giao diện quy định. Trong ngôn ngữ C#, một lớp chỉ có thể kế thừa từ duy nhất một lớp cha, tức là không cho đa kế thừa như trong ngôn ngữ C++, tuy nhiên một lớp có thế thực thi nhiều giao diện. Khi một lớp thực thi một giao diện thì nó sẽ hứa là nó sẽ cung cấp chức năng thực thi giao diện.

# Phần I: Tìm hiểu ngôn ngữ C Sharp (C#)

- Trong ngôn ngữ C#, những cấu trúc cũng được hỗ trợ, nhưng khái niệm về ngữ nghĩa của nó thay đổi khác với C#. Trong C# một cấu trúc được giới hạn, là kiểu dữ liệu nhỏ gọn và khi tạo thể hiện thì nó yêu cầu ít hơn về hệ điều hành và bộ nhớ so với một lớp. Một cấu trúc thì không thể kế thừa từ một lớp hay được kế thừa, nhưng một cấu trúc có thể thực thi một giao diện.

# Phần I: Tìm hiểu ngôn ngữ C Sharp (C#)

- Ngôn ngữ C# cung cấp những đặc tính hướng thành phần, như là những thuộc tính, những sự kiện. Lập trình hướng thành phần được hỗ trợ bởi CLR cho phép lưu trữ metadata với mã nguồn cho một lớp. Metadata mô tả cho một lớp, bao gồm những thuộc tính và những phương thức của nó, cũng như những sự bảo mật cần thiết và những thuộc tính khác. Mã nguồn chứa đựng những logic cần thiết để thực hiện những chức năng của nó.

# Phần I: Tìm hiểu ngôn ngữ C Sharp (C#)

Do vậy, một lớp được biên dịch như là một khối Self-contained, nên môi trường Hosting biết được cách đọc metadata của một lớp và mã nguồn cần thiết mà không cần những thông tin khác để sử dụng nó.

# Phần I: Tìm hiểu ngôn ngữ C Sharp (C#)

- Một lưu ý cuối cùng về ngôn ngữ C#, là ngôn ngữ này cũng hỗ trợ truy cập bộ nhớ trực tiếp sử dụng con trỏ của C++ và từ khóa cho dấu [] trong toán tử. Các mã nguồn này là không an toàn. Và bộ giải phóng bộ nhớ tự động của CLR sẽ không thực hiện giải phóng những đối tượng được tham chiếu bằng sử dụng con trỏ cho đến khi chúng được giải phóng.

# Tại sao phải sử dụng ngôn ngữ C#

- Nhiều người tin rằng không cần thiết phải có một ngôn ngữ lập trình mới. Java, C++, Visual basic, Perl, và những ngôn ngữ khác được nghĩ rằng đã cung cấp tất cả những chức năng cần thiết.

# Tại sao phải sử dụng ngôn ngữ C#

- Ngôn ngữ C# là ngôn ngữ được dẫn xuất từ C và C++, nhưng nó được tạo từ nền tảng phát triển hơn. Microsoft bắt đầu công việc trong C và C++ và thêm vào những đặc tính mới để làm cho ngôn ngữ này dễ sử dụng hơn. Nhiều trong số những đặc tính này khá giống với những đặc tính có trong ngôn ngữ java. Không dừng lại ở đó, Microsoft đưa ra một số mục đích khi xây dựng ngôn ngữ này. Những mục đích này được tóm tắt như sau:

# Tại sao phải sử dụng ngôn ngữ C#

- C# là ngôn ngữ đơn giản
- C# là ngôn ngữ hiện đại
- C# là ngôn ngữ hướng đối tượng
- C# là ngôn ngữ mạnh mẽ và mềm dẻo
- C# là ngôn ngữ ít có từ khóa
- C# là ngôn ngữ hướng Module
- C# sẽ trở nên phổ biến

# Tại sao phải sử dụng ngôn ngữ C#

- Thứ nhất, C# là ngôn ngữ đơn giản

C# loại bỏ một vài sự phức tạp và rối rắm của những ngôn ngữ như java và C++, bao gồm việc loại bỏ những Macro, những template, đa kế thừa, và lớp cơ sở ảo ( virtual base class). Chúng là những nguyên nhân gây ra sự nhầm lẫn hay dẫn đến những vấn đề cho người phát triển C++. Nếu chúng ta là người học ngôn ngữ này đầu tiên thì chắc chắn là ta sẽ không trải qua những thời gian để học nó! Nhưng khi đó chúng ta sẽ không biết được hiệu quả của nó khi loại bỏ những vấn đề khó khăn trên.

# Tại sao phải sử dụng ngôn ngữ C#

- Ngôn ngữ C# đơn giản vì nó dựa trên nền tảng C và C++. Nếu chúng ta thân thiện với C và C++ hoặc thậm trí là java, chúng ta sẽ thấy C# khá giống về diện mạo, cú pháp, biểu thức, toán tử và những chức năng khác được lấy trực tiếp từ ngôn ngữ C và C++, nhưng nó đã được cải tiến để làm ngôn ngữ đơn giản hơn. Một vài trong sự cải tiến là sự loại bỏ dư thừa, hay là thêm vào những cú pháp thay đổi. Ví dụ như,  C++ có 3 toán tử làm việc với các thành viên là: :, . , và ->

# Tại sao phải sử dụng ngôn ngữ C#

- Để biết khi nào dùng ba toán tử này cũng phức tạp và dễ nhầm lẫn. Trong C# chúng được thay thế bởi một toán tử duy nhất là: . (dot).  . Đối với người mới học thì điều này và những việc cải tiến khác làm bớt nhầm lẫn và đơn giản hơn

# Tại sao phải sử dụng ngôn ngữ C#

- **Thứ hai, C# là ngôn ngữ hiện đại**

Điều gì làm cho một ngôn ngữ hiện đại? những đặc tính như là xử lý ngoại lệ, thu gom bộ nhớ tự động, những kiểu dữ liệu mở rộng, và bảo mật mã nguồn là những đặc tính được mong đợi trong những ngôn ngữ hiện đại. C# chứa tất cả các đặc tính trên. Nếu là người mới học lập trình có lẽ chúng ta sẽ cảm thấy những đặc tính trên là khá phức tạp và khó hiểu. Tuy nhiên, khi bạn học thì bạn thấy nó cực kỳ dễ hiểu.!!

# Tại sao phải sử dụng ngôn ngữ C#

- **Thứ ba, C# là ngôn ngữ hướng đối tượng**

Những đặc tính chính của ngôn ngữ hướng đối tượng là sự đóng gói, kế thừa và tính đa hình. C# hỗ trợ tất cả các đặc tính trên.

# Tại sao phải sử dụng ngôn ngữ C#

- **Thứ tư, C# mạnh mẽ và mềm dẻo**

  Như đã đề cập, C# chúng ta chỉ bị giới hạn bởi chính bản thân hay trí tưởng tượng của chúng ta. Ngôn ngữ này không đặt những ràng buộc lên những việc có thể làm. C# được sử dụng cho nhiều những dự án khác nhau như là tạo ra ứng dụng sử lý văn bản, ứng dụng đồ họa, bản tính hay thậm chí những trình biên dịch cho những ngôn ngữ khác.

# Tại sao phải sử dụng ngôn ngữ C#

- **Thứ năm, C# là ngôn ngữ ít từ khóa**

  C# là ngôn ngữ sử dụng giới hạn những từ khóa. Phần lớn các từ khóa dùng để mô tả các thông tin. Chúng ta có thể sẽ nghĩ rằng một ngôn ngữ với nhiều từ khóa sẽ mạnh hơn. Điều này không phải là sự thật, ít nhất là trong trường hợp ngôn ngữ C#, chúng ta có thể tìm thấy rằng ngôn ngữ này có thể được sử dụng để làm bất cứ nhiệm vụ nào.

# Tại sao phải sử dụng ngôn ngữ C#

- **Thứ sáu, C# là ngôn ngữ hướng Module**

  Mã nguồn C# có thể được viết trong những phần được gọi là những lớp, những lớp này chứa các phương thức thành viên của nó. Những lớp và phương thức có thể được sử dụng lại trong ứng dụng hay những chương trình khác. Bằng cách truyền những mẫu thông tin đến những lớp hay phương thức chúng ta có thế tạo ra những mã nguồn dùng lại có hiệu quả.

# Tại sao phải sử dụng ngôn ngữ C#

- Thứ bảy, C# là ngôn ngữ phổ biến
- Thứ tám, C# và các ngôn ngữ khác

# Tại sao phải sử dụng ngôn ngữ C#

- Thứ tám, C# và các ngôn ngữ khác

- Chúng ta đã từng nghe đến những ngôn ngữ khác như: Visual Basic, C++ và Java. Có lẽ chúng ta cũng tự hỏi sự khác nhau giữa ngôn ngữ C# và những ngôn ngữ đó. Và cũng tự hỏi tại sao lại chọn ngôn ngữ này để học mà không chọn một trong những ngôn ngữ kia. Có rất nhiều lý do và chúng ta hãy xem một số sự so sánh giữa ngôn ngữ C# và các ngôn ngữ khác vừa đề cập giúp chúng ta phần nào trả lời những thắc mắc.

# Tại sao phải sử dụng ngôn ngữ C#

- Thứ tám, C# và các ngôn ngữ khác

- Microsoft nói rằng C# mang đến sức mạnh của ngôn ngữ C++ với sự dễ dàng của ngôn ngữ Visual Basic, nhưng với phiên bản của Visual Basic.NET (Version 7.0) thì ngang nhau. Bởi vì chúng được viết lại từ một nền tảng. Chúng có thể viết nhiều chương trình với ít mã nguồn hơn nếu dùng C#.

# Tại sao phải sử dụng ngôn ngữ C#

- Thứ tám, C# và các ngôn ngữ khác

- Mặc dù C# loại bỏ một vài đặc tính của C++, nhưng bì lại nó tránh được những lỗi mà thường gặp trong C++. Điều này có thể tiết kiệm được hàng giờ hay thậm chí hàng ngày trong việc hoàn tất một chương trình.
- Một điều quan trọng khác với C++ là mã nguồn C# không đòi hỏi phải có tập tin Header. Tất cả mã nguồn được viết trong khai báo một lớp.

# Tại sao phải sử dụng ngôn ngữ C#

- Thứ tám, C# và các ngôn ngữ khác

- Như đã nói trên .NET runtime trong C# thực hiện việc thu gom bộ nhớ tự động. Do điều này nên việc sử dụng con trỏ trong C# ít quan trọng hơn trong C++. Những con trỏ cũng có thể được sử dụng trong C#, khi đó những đoạn mã nguồn này sẽ được đành dấu là không an toàn (unsafe code).

# Tại sao phải sử dụng ngôn ngữ C#

- Thứ tám, C# và các ngôn ngữ khác

- C# cũng từng bỏ ý tưởng đa kế thừa như trong C++. Và sự khác nhau là C# đưa thêm thuộc tính vào trong một lớp giống như trong Visual Basic. Và các thành viên của lớp được gọi duy nhất bằng toán tử "." Khác với C++ có nhiều cách gọi trong các tình huống khác nhau.

# Tại sao phải sử dụng ngôn ngữ C#

- Thứ tám, C# và các ngôn ngữ khác

Một ngôn ngữ khác cũng rất mạnh và phổ biến là Java, giống như C++ và C# được phát triển dựa trên C. Nếu chúng ta quyết định sẽ học Java sau này, chúng ta sẽ tìm được nhiều cái mà học từ C# có thể được áp dụng.

# Tại sao phải sử dụng ngôn ngữ C#

- Thứ tám, C# và các ngôn ngữ khác

- Điểm giống nhau giữa C# và Java là cả hai cùng biên dịch ra mã trung gian: C# biên dịch ra MSIL còn Java biên dịch ra Bytecode. Sau đó chúng được thực hiện bằng cách thông dịch hoặc biên dịch just-in-time trong từng máy ảo tương ứng. Tuy nhiên trong ngôn ngữ C# nhiều hỗ trợ được đưa ra để biên dịch mã ngôn ngữ trung gian sang mã máy. C# chứa nhiều kiểu dữ liệu cơ bản hơn java và cũng cho phép nhiều sự mở rộng với kiểu dữ liệu giá trị. Ví dụ C# hỗ trợ kiểu liệt kê.

# Tại sao phải sử dụng ngôn ngữ C#

- Thứ tám, C# và các ngôn ngữ khác

- Tương tự Java, C# cũng bỏ tính đa kế thừa trong một lớp, tuy nhiên mô hình kế thừa đơn giản này được mở rộng bởi tình đa kế thừa nhiều giao diện.

# Phần II: Khởi đầu học C Sharp

- Các bước chuẩn bị cho một chương trình
  - Xác định mục tiêu của chương trình
  - Xác định phương pháp giải quyết vấn đề
  - Tạo một chương trình để giải quyết vấn đề
  - Thực thi chương trình để xem kết quả

# Phần II: Khởi đầu học C Sharp

- Khởi đầu một chương trình C# đơn giản

  Phần này ta sẽ tạo, biên dịch và chạy chương trình "Hello World" bằng ngôn ngữ C#. Phân tích ngắn gọn chương trình để giới thiệu các đặc trưng chính yếu trong ngôn ngữ C#.

# Phần II: Khởi đầu học C Sharp

**Ví dụ 2-1 Chương trình Hello World**

```
class HelloWorld
{
    static void Main( )
    {
        // sử dụng đối tượng console của hệ thống
        System.Console.WriteLine("Hello World");
    }
}
```

# Phần II: Khởi đầu học C Sharp

=> Sau khi biên dịch và chạy HelloWorld, kết quả là dòng chữ “Hello World” hiển thị trên màn hình.

# Phần II: Khởi đầu học C Sharp

1.1 Lớp, đối tượng và kiểu

- Bản chất của lập trình hướng đối tượng là tạo ra các kiểu mới. Một *kiểu* biểu diễn một vật gì đó. Giống với các ngôn ngữ lập trình hướng đối tượng khác, một kiểu trong C# cũng định nghĩa bằng từ khoá *class* (và được gọi là lớp) còn thể hiện của lớp được gọi là *đối tượng*.

# Phần II: Khởi đầu học C Sharp

Xem Ví dụ 2-1 ta thấy cách khai báo một lớp HelloWorld. Ta thấy ngay là cách khai báo và nội dung của một lớp hoàn toàn giống với ngôn ngữ Java và C++, chỉ có khác là cuối khai báo lớp không cần dấu “;”

# Phần II: Khởi đầu học C Sharp

1.1.1 Phương thức

- Các hành vi của một lớp được gọi là các phương thức thành viên (gọi tắt là phương thức) của lớp đó. Một *phương thức* là một *hàm* (phương thức thành viên còn gọi là hàm thành viên). Các phương thức định nghĩa những gì mà một lớp có thể làm.

# Phần II: Khởi đầu học C Sharp

- Cách khai báo, nội dung và cách sử dụng các phương thức giống hoàn toàn với Java và C++. Trong ví dụ trên có một phương thức đặc biệt là phương thức Main() (như hàm main() trong C++) là phương thức bắt đầu của một ứng dụng C#, có thể trả về kiểu void hay int. Mỗi một chương trình (assembly) có thể có nhiều phương thức Main nhưng khi đó phải chỉ định  phương thức Main() nào sẽ bắt đầu chương trình.

# Phần II: Khởi đầu học C Sharp

1.1.2 Các ghi chú

- C# có ba kiểu ghi chú trong đó có hai kiểu rất quen thuộc của C++ là dùng: "//" và "/* … */". Ngoài ra còn một kiểu ghi chú nữa sẽ trình bày ở các chương kế.

# Phần II: Khởi đầu học C Sharp

**Ví dụ 2-2 Hai hình thức ghi chú trong C#**

```
class HelloWorld
{
    static void Main( ) // Đây là ghi trên một dòng
    {
        /* Bắt đầu ghi chú nhiều dòng
                Vẫn còn trong ghi chú
           Kết thúc ghi chú bằng */
            System.Console.WriteLine("Hello World");
    }
}
```

# Phần II: Khởi đầu học C Sharp

## 1.1.3 Ứng dụng dạng console

- "Hello World" là một ứng dụng console. Các ứng dụng dạng này thường không có giao diện người dùng đồ họa Các nhập xuất đều thông qua các console chuẩn (dạng dòng lệnh như DOS).

# Phần II: Khởi đầu học C Sharp

- Trong ví dụ trên, phương thức Main() viết ra màn hình dòng "Hello World". Do màn hình quản lý một đối tượng Console, đối tượng này có phương thức WriteLine() cho phép đặt một dòng chữ lên màn hình. Để gọi phương thức này ta dùng toán tử ".", như sau: Console.WriteLine(…).

# Phần II: Khởi đầu học C Sharp

1.1.4 Namespaces - Vùng tên

- Console là một trong rất nhiều (cả ngàn) lớp trong bộ thư viện .NET. Mỗi lớp đều có tên và như vậy có hàng ngàn tên mà lập trình viên phải nhớ hoặc phải tra cứu mỗi khi sử dụng. Vấn đề là phải làm sao giảm bớt lượng tên phải nhớ.

# Phần II: Khởi đầu học C Sharp

- Ngoài vấn đề phải nhớ quá nhiều tên ra, còn một nhận xét sau: một số lớp có mối liên hệ nào đó về mặt ngữ nghĩa, ví dụ như lớp Stack, Queue, Hashtable … là các lớp cài đặt cấu trúc dữ liệu túi chứa. Như vậy có thể nhóm những lớp này thành một nhóm và thay vì phải nhớ tên các lớp thì lập trình viên chỉ cần nhớ tên nhóm, sau đó có thể thực hiện việc tra cứu tên lớp trong nhóm nhanh chóng hơn. Nhóm là một *vùng tên* trong C#.

# Phần II: Khởi đầu học C Sharp

- Một vùng tên có thể có nhiều lớp và vùng tên khác. Nếu vùng tên A nằm trong vùng tên B, ta nói vùng tên A là vùng tên con của vùng tên B. Khi đó các lớp trong vùng tên A được ghi như sau:

# Phần II: Khởi đầu học C Sharp

B.A.Tên_lớp_trong_vùng_tên_A

System là vùng tên chứa nhiều lớp hữu ích cho việc giao tiếp với hệ thống hoặc các lớp công dụng chung như lớp Console, Math, Exception….Trong ví dụ HelloWorld trên, đối tượng Console được dùng như sau:

System.Console.WriteLine("Hello World");

# Phần II: Khởi đầu học C Sharp

1.1.5 Toán tử chấm “.”

- Như trong Ví dụ 2-1 toán tử chấm được dùng để truy suất dữ liệu và phương thức một lớp (như Console.WriteLine()), đồng thời cũng dùng để chỉ định tên lóp trong một vùng tên (như System.Console).

- Toán tử dấu chấm cũng được dùng để truy xuất các vùng tên con của một vùng tên
- Vùng_tên.Vùng_tên_con.Vùng_tên_con_con

# Phần II: Khởi đầu học C Sharp

1.1.6 Từ khoá using

- Nếu chương trình sử dụng nhiều lần phương thức Console.WriteLine, từ System sẽ phải viết nhiều lần. Điều này có thể khiến lập trình viên nhàm chán. Ta sẽ khai báo rằng chương trình có sử dụng vùng tên System, sau đó ta dùng các lớp trong vùng tên System mà không cần phải có từ System đi trước.

# Phần II: Khởi đầu học C Sharp

**Ví dụ 2-3 Từ khóa using**

```
// Khai báo chương trình có sử dụng vùng tên System
using System;
class HelloWorld
{
    static void Main( )
    {
        // Console thuộc vùng tên System
        Console.WriteLine("Hello World");
    }
}
```

# Phần II: Khởi đầu học C Sharp

1.1.7 Phân biệt hoa thường

- Ngôn ngữ C# cũng phân biệt chữ hoa thường giống như Java hay C++ (không như VB). Ví dụ như WriteLine khác với writeLine và cả hai cùng khác với WRITELINE. Tên biến, hàm, hằng … đều phân biệt chữ hoa chữ thường.

# Phần II: Khởi đầu học C Sharp

## 1.1.8 Từ khoá static

- Trong Ví dụ 2-1 phương thức Main() được khai báo kiểu trả về là void và dùng từ khoá **static**. Từ khoá static cho biết là ta có thể gọi phương thức Main() mà không cần tạo một đối tượng kiểu HelloWorld.

# Phần II: Khởi đầu học C Sharp

1.2 Phát triển "Hello World"

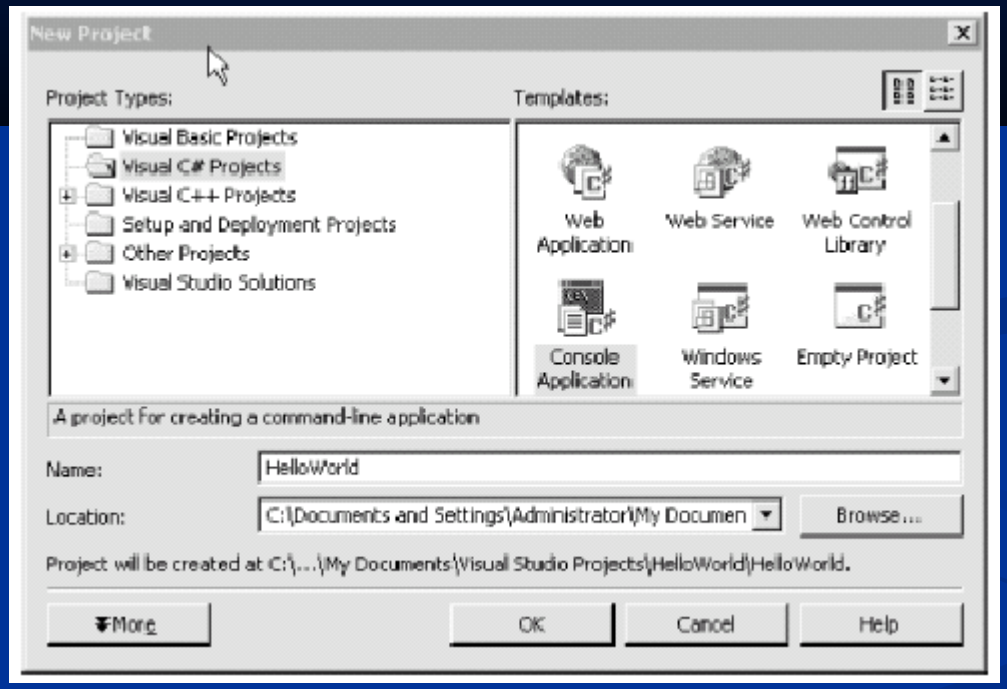
- Có hai cách để viết, biên dịch và chạy chương trình HelloWorld là dùng môi trưởng phát triển tích hợp (IDE) Visual Studio .Net hay viết bằng trình soạn thảo văn bản và biên dịch bằng dòng lệnh. IDE Vs.Net dễ dùng hơn. Do đó, trong đề tài này chỉ trình bày theo hướng làm việc trên IDE Visual Studio .Net.

# Phần II: Khởi đầu học C Sharp

1.2.1 Soạn thảo “Hello World”

- Để tạo chương trình “Hello World” trong IDE, ta chọn Visual Studio .Net từ thanh thực đơn. Tiếp theo trên màn hình của IDE chọn *File* > *New* > *Project* từ thanh thực đơn, theo đó xuất hiện một cửa sổ như sau:

# Phần II: Khởi đầu học C Sharp

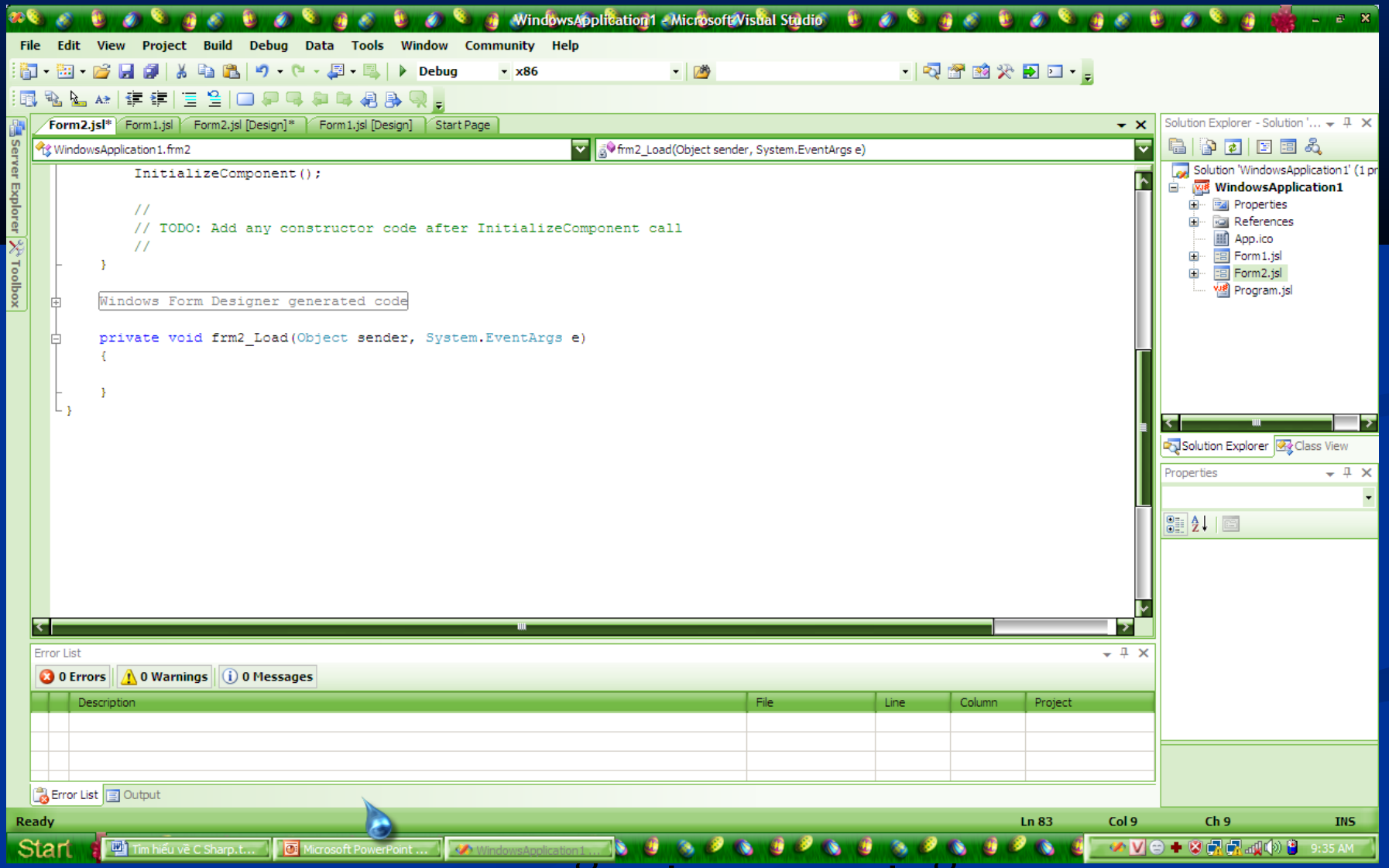


# Phần II: Khởi đầu học C Sharp

**Hình 2-1: Tạo một ứng dụng console trong VS.Net (trên)**

**Để tạo chương trình "Hello World" ta chọn Visual *C# Project* > *Console Application*, điền *HelloWorld* trong ô Name, chọn đường dẫn và nhấn OK. Một cửa sổ soạn thảo xuất hiện.**

# Phần II: Khởi đầu học C Sharp



# Phần II: Khởi đầu học C Sharp

**Hình 2-2 Cửa sổ soạn thảo nội dung mã nguồn**

- Vs.Net tự tạo một số mã, ta cần chỉnh sửa cho phù hợp với chương trình của mình.

# Phần II: Khởi đầu học C Sharp

1.2.2 Biên dịch và chạy "Hello World"

- Sau khi đã đầy đủ mã nguồn ta tiến hành biên dịch chương trình: nhấn "Ctrl–Shift–B" hay chọn *Build* > *Build Solution*. Kiểm tra xem chương trình có lỗi không ở của sổ *Output* cuối màn hình. Khi biên dịch chương trình nó sẽ lưu lại thành tập tin *.cs*.

- Chạy chương trình bằng "Ctrl–F5" hay chọn *Debug* > *Start Without Debugging*.

# Phần II: Khởi đầu học C Sharp

1.2.3 Trình gở rối của Visual Studio .Net

- Trình gỡ rối của VS.Net rất mạnh hữu ích. Ba kỹ năng chính yếu để sử dụng của trình gở rối là:

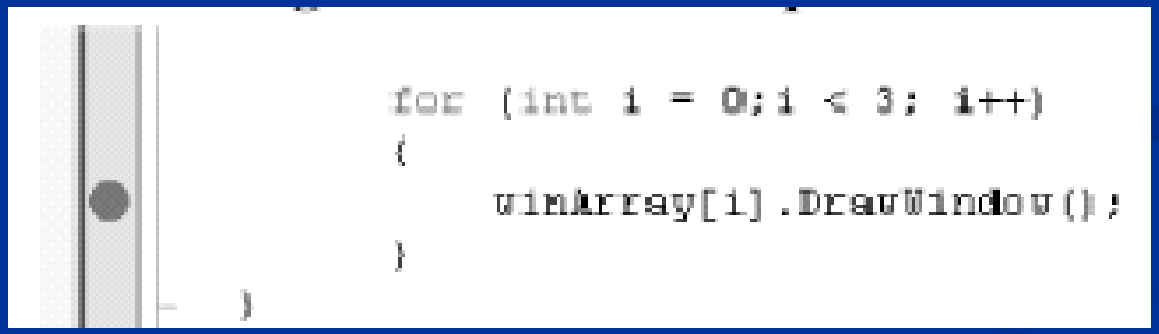
- Cách đặt điểm ngắt (breakpoint) và làm sao chạy cho đến điểm ngắt
- Làm thế nào chạy từng bước và chạy vượt qua một phương thức.
- Làm sao để quan sát và hiệu chỉnh giá trị của biến, dữ liệu thành viên, …

# Phần II: Khởi đầu học C Sharp

Cách đơn giản nhất để đặt điểm ngắt là bấm chuột trái vào phía lề trái, tại đó sẽ hiện lên một chấm đỏ.

# Phần II: Khởi đầu học C Sharp

## Hình 2-3 Minh họa một điểm ngắt



# Phần II: Khởi đầu học C Sharp

- Cách dùng trình gở rối hoàn toàn giống với trình gở rối trong VS 6.0. Nó cho phép ta dừng lại ở một vị trí bất kỳ, cho ta kiểm tra giá trị tức thời bằng cách di chuyển chuột đến vị trị biến. Ngoài ra, khi gở rối ta cũng có thể xem giá trị các biến thông qua cửa sổ *Watch* và *Local*.
- Để chạy trong chế độ gở rối ta chọn *Debug* → *Start* hay nhấn **F5**, muốn chạy từng bước ta bấm **F11** và chạy vượt qua một phương thức ta bấm **F10**

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

Trong chương này sẽ trình bày về hệ thống kiểu trong C#; phân biệt kiểu dựng sẵn (*int*, *long*, *bool*, …) với các kiểu do người dùng định nghĩa. Ngoài ra, chương này cũng sẽ trình bày cách tạo và dùng biến, hằng; giới thiệu kiểu liệt kê, chuỗi, kiểu định danh, biểu thức, và câu lệnh. Phần hai của chương trình bày về các cấu trúc điều kiện và các toán tử logic, quan hệ, toán học, …

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.1 Các kiểu

- C# buộc phải khai báo kiểu của đối tượng được tạo. Khi kiểu được khai báo rõ ràng, trình biên dịch sẽ giúp ngăn ngừa lỗi bằng cách kiểm tra dữ liệu được gán cho đối tượng có hợp lệ không, đồng thời cấp phát **đúng** kích thước bộ nhớ cho đối tượng.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

- C# phân thành hai loại: loai dữ liệu dựng sẵn và loại do người dùng định nghĩa.
- C# cũng chia tập dữ liệu thành hai kiểu: *giá trị* và *tham chiếu*. Biến kiểu giá trị được lưu trong vùng nhớ stack, còn biến kiểu tham chiếu được lưu trong vùng nhớ heap.
- C# cũng hỗ trợ kiểu con trỏ của C++, nhưng ít khi được sử dụng. Thông thường con trỏ chỉ được sử dụng khi làm việc trực tiếp với Win API hay các đối tượng COM.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.1.1 Loại dữ liệu định sẵn

- C# có nhiểu kiểu dữ liệu định sẵn, mỗi kiểu ánh xạ đến một kiểu được hổ trợ bởi CLS (Commom Language Specification), ánh xạ để đảm bảo rằng  đối tượng được tạo trong C# không khác gì đối tượng được tạo trong các ngôn ngữ .NET khác Mỗi kiểu có một kích thước cố định được liệt kê trong bảng sau

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.1.1.1 Chọn một kiểu định sẵn

- Tuỳ vào từng giá trị muốn lưu trữ mà ta chọn kiểu cho phù hợp. Nếu chọn kiểu quá lớn so với các giá trị cần lưu sẽ làm cho chương trình đòi hỏi nhiều bộ nhớ và chạy chậm. Trong khi nếu giá trị cần lưu lớn hơn kiểu thực lưu sẽ làm cho giá trị các biến bị sai và chương trình cho kết quả sai.
- Kiểu *char* biểu diễn một ký tự Unicode. Ví dụ "\u0041" là ký tự "A" trên bảng Unicode. Một số ký tự đặc biệt được biểu diễn bằng dấu "\" trước một ký tự khác.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

- **Bảng 3-2 Các ký tự đặc biệt thông dụng**

| Ký tự | Nghĩa |
|---|---|
| \’ | dầu nháy đơn |
| \” | dấu nháy đôi |
| \\ | dấu chéo ngược “\” |
| \0 | Null |
| \a | Alert |
| \b | lùi về sau |
| \f | Form feed |
| \n | xuống dòng |
| \r | về đầu dòng |
| \t | Tab ngang |
| \v | Tab dọc |

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

2.1.1.2 Chuyển đổi kiểu định sẵn

Một đối tượng có thể chuyển từ kiểu này sang kiểu kia theo hai hình thức: ngầm hoặc tường minh. Hình thức ngầm được chuyển tự động còn hình thức tường minh cần sự can thiệp trực tiếp của người lập trình (giống với C++ và Java).

```
short x = 5;
int y ;
y = x; // chuyển kiểu ngầm định - tự động
x = y; // lỗi, không biên dịch được
x = (short) y; // OK
```

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.2 Biến và hằng

- Biến dùng để lưu trữ dữ liệu. Mỗi biến thuộc về một kiểu dữ liệu nào đó.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.2.1 Khởi tạo trước khi dùng

- Trong C#, trước khi dùng một biến thì biến đó phải được khởi tạo nếu không trình biên dịch sẽ báo lỗi khi biên dịch. Ta có thể khai báo biến trước, sau đó khởi tạo và sử dụng; hay khai báo biến và khởi gán trong lúc khai báo.

- int x; // khai báo biến trước
- x = 5; // sau đó khởi gán giá trị và sử dụng
- int y = x; // khai báo và khởi gán cùng lúc

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.2.2 Hằng

- Hằng là một biến nhưng giá trị không thay đổi theo thời gian. Khi cần thao tác trên một giá trị xác định ta dùng hằng. Khai báo hằng tương tự khai báo biến và có thêm từ khóa const ở trước. Hằng một khi khởi động xong không thể thay đổi được nữa.

```
const int HANG_SO = 100;
```

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.2.3 Kiểu liệt kê

- Enum là một cách thức để đặt tên cho các trị nguyên (các trị kiểu số nguyên, theo nghĩa nào đó tương tự như tập các hằng), làm cho chương trình rõ ràng, dễ hiểu hơn. Enum không có hàm thành viên. Ví dụ tạo một enum tên là Ngay như sau:

```
enum Ngay {Hai, Ba, Tu, Nam, Sau, Bay, ChuNhat};
```

Theo cách khai báo này enum ngày có bảy giá trị nguyên đi từ 0 = Hai, 1 = Ba, 2 = Tư … 7 = ChuNhat.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

**Ví dụ 3-1 Sử dụng enum Ngay**

```
using System;
public class EnumTest
{
   enum Ngay {Hai, Ba, Tu, Nam, Sau, Bay, ChuNhat };
   public static void Main()
   {
      int x = (int) Ngay.Hai;
      int y = (int) Ngay.Bay;
      Console.WriteLine("Thu Hai = {0}", x);
      Console.WriteLine("Thu Bay = {0}", y);
   }
}
```

Kết quả

Thu Hai = 0

Thu Bay = 5

Mặc định enum gán giá trị đầu tiên là 0 các trị sau lớn hơn giá trị trước một đơn vị, và các trị này thuộc kiểu int. Nếu muốn thay đổi trị mặc định này ta phải gán trị mong muốn.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.2.4 Chuỗi

- Chuỗi là kiểu dựng sẵn trong C#, nó là một chuổi các ký tự đơn lẻ. Khi khai báo một biến chuỗi ta dùng từ khoá *string*. Ví dụ khai báo một biến string lưu chuỗi "Hello World"

```
string myString = "Hello World";
```

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.2.5 Định danh

- Định danh là tên mà người lập trình chọn đại diện một kiểu, phương thức, biến, hằng, đối tượng… của họ. Định danh **phải** bắt đầu bằng một ký tự hay dấu "_". Định danh không được trùng với từ khoá C# và phân biệt hoa thường.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

2.3 Biểu thức

- Bất kỳ câu lệnh định lượng giá trị được gọi là một biểu thức (*expression)*. Phép gán sau cũng được gọi là một biểu thức vì nó định lượng giá trị được gán (là 32)

x = 32;

vì vậy phép gán trên có thể được gán một lần nữa như sau

y = x = 32;

Sau lệnh này y có giá trị của biểu thức x = 32 và vì vậy y = 32.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.4 Khoảng trắng

- Trong C#, khoảng trống, dấu tab, dấu xuống dòng đều được xem là khoảng trắng (*whitespace*). Do đó, dấu cách dù lớn hay nhỏ đều như nhau nên ta có:

```
x = 32;
```

cũng như

```
x     =          32;
```

Ngoại trừ khoảng trắng trong chuỗi ký tự thì có ý nghĩa riêng của nó.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.5 Câu lệnh

- Cũng như trong C++ và Java một chỉ thị hoàn chỉnh thì được gọi là một câu lệnh (*statement)*. Chương trình gồm nhiều câu lệnh, mỗi câu lệnh kết thúc bằng dấu ";". Ví dụ:

```
int x; // là một câu lệnh
x = 23; // một câu lệnh khác
```

Ngoài các câu lệnh bình thường như trên, có các câu lệnh khác là: lệnh rẽ nhánh không điều kiện, rẽ nhánh có điều kiện và lệnh lặp.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.5.1 Các lệnh rẽ nhánh không điều kiện

- Có hai loại câu lệnh rẽ nhánh không điều kiện. Một là lệnh gọi phương thức: khi trình biên dịch thấy có lời gọi phương thức nó sẽ tạm dừng phương thức hiện hành và nhảy đến phương thức được gọi cho đến hết phương thức này sẽ trở về phương thức cũ.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.5.2 Lệnh rẽ nhánh có điều kiện

- Các từ khóa if-else, while, do-while, for, switch-case, dùng để điều khiển dòng chảy chương trình. C# giữ lại tất cả các cú pháp của C++, ngoại trừ switch có vài cải tiến.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

2.5.2.1 Lệnh If .. else …

Cú pháp:

if ( biểu thức logic )

    khối lệnh;

hoặc

if ( biểu thức logic )

    khối lệnh 1;

else

    khối lệnh 2;

Ghi chú: Khối lệnh *là một tập các câu lện trong cặp dấu "{…}". Bất kỳ nơi đâu có câu lệnh thì ở đó có thể viết bằng một khối lệnh.*

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

2.5.2.2 Lệnh switch

Cú pháp:

```
switch ( biểu_thức_lựa_chọn )
{  case biểu_thức_hằng :
     khối lệnh;
     lệnh nhảy;
   [ default :
     khối lệnh;
     lệnh nhảy; ]
   }
```

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.5.3 Lệnh lặp

- C# cung cấp các lệnh lặp giống C++ như *for*, *while*, *do-while* và lệnh lặp mới *foreach*. Nó cũng hổ trợ các câu lệnh nhảy như: *goto*, *break*, *continue* và *return*.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

2.5.3.1 Lệnh goto

Lệnh *goto* có thể dùng để tạo lệnh nhảy nhưng nhiều nhà lập trình chuyên nghiệp khuyên không nên dùng câu lệnh này vì nó phá vỡ tính cấu trúc của chương trình. Cách dùng câu lệnh này như sau: (giống như trong C++)

Tạo một nhãn

goto đến nhãn đó.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

2.5.3.2 Vòng lặp while

Cú pháp:

```
while ( biểu_thức_logic )
    khối_lệnh;
```

Khối_lệnh sẽ được thực hiện cho đến khi nào biểu thức còn đúng. Nếu ngay từ đầu biểu thức sai, khối lệnh sẽ không được thực thi.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

2.5.3.3 Vòng lặp do … while

Cú pháp:

```
do
   khối_lệnh
while ( biểu_thức_logic )
```

Khác với while khối lệnh sẽ được thực hiện trước, sau đó biệu thức được kiểm tra. Nếu biểu thức đúng khối lệnh lại được thực hiện.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

2.5.3.4 Vòng lặp for

Cú pháp:

```
for ( [khởi_tạo_biến_đếm]; [biểu_thức];
   [gia_tăng_biến_đếm] )
   khối lệnh;
```

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.5.3.5 Câu lệnh break, continue, và return

Cả ba câu lệnh *break*, *continue*, và *return* rất quen thuộc trong C++ và Java, trong C#, ý nghĩa và cách sử dụng chúng hoàn toàn giống với hai ngôn ngữ này.

# Phần II: Khởi đầu học C Sharp

## Chương 2: Những cơ sở của ngôn ngữ C#

### 2.6 Toán tử

- Các phép toán +, -, *, / là một ví dụ về toán tử. Áp dụng các toán tử này lên các biến kiểu số ta có kết quả như việc thực hiện các phép toán thông thường.

```
int a = 10;
int b = 20;
int c = a + b; // c = 10 + 20 = 30
```

C# cung cấp cấp nhiều loại toán tử khác nhau để thao tác trên các kiểu biến dữ liệu, được liệt kê trong bảng sau theo từng nhóm ngữ nghĩa.

# Phần II: Khởi đầu học C Sharp
## Chương 2: Những cơ sở của ngôn ngữ C#

**Bảng 3-3 Các nhóm toán tử trong C#**

| Nhóm toán tử | Toán tử | Ý nghĩa |
|---|---|---|
| Toán học | + - * / % | cộng , trừ, nhân chia, lấy phần dư |
| Logic | & \| ^ ! ~ && \|\| true false | phép toán logic và thao tác trên bit |
| Ghép chuỗi | + | ghép nối 2 chuỗi |
| Tăng, giảm | ++, -- | tăng / giảm toán hạng lên / xuống 1. Đứng trước hoặc sau toán hạng. |
| Dịch bit | << >> | dịch trái, dịch phải |
| Quan hệ | == != < > <= >= | bằng, khác, nhỏ/lớn hơn, nhỏ/lớn hơn hoặc bằng |
| Gán | = += -= *= /= %= &= \|= ^= <<= >>= | phép gán |
| Chỉ số | [] | cách truy xuất phần tử của mảng |
| Ép kiểu | () | |
| Indirection và Address | * -> [] & | dùng cho con trỏ |

# Phần III: Lập trình Windows Form

# Chương 1.

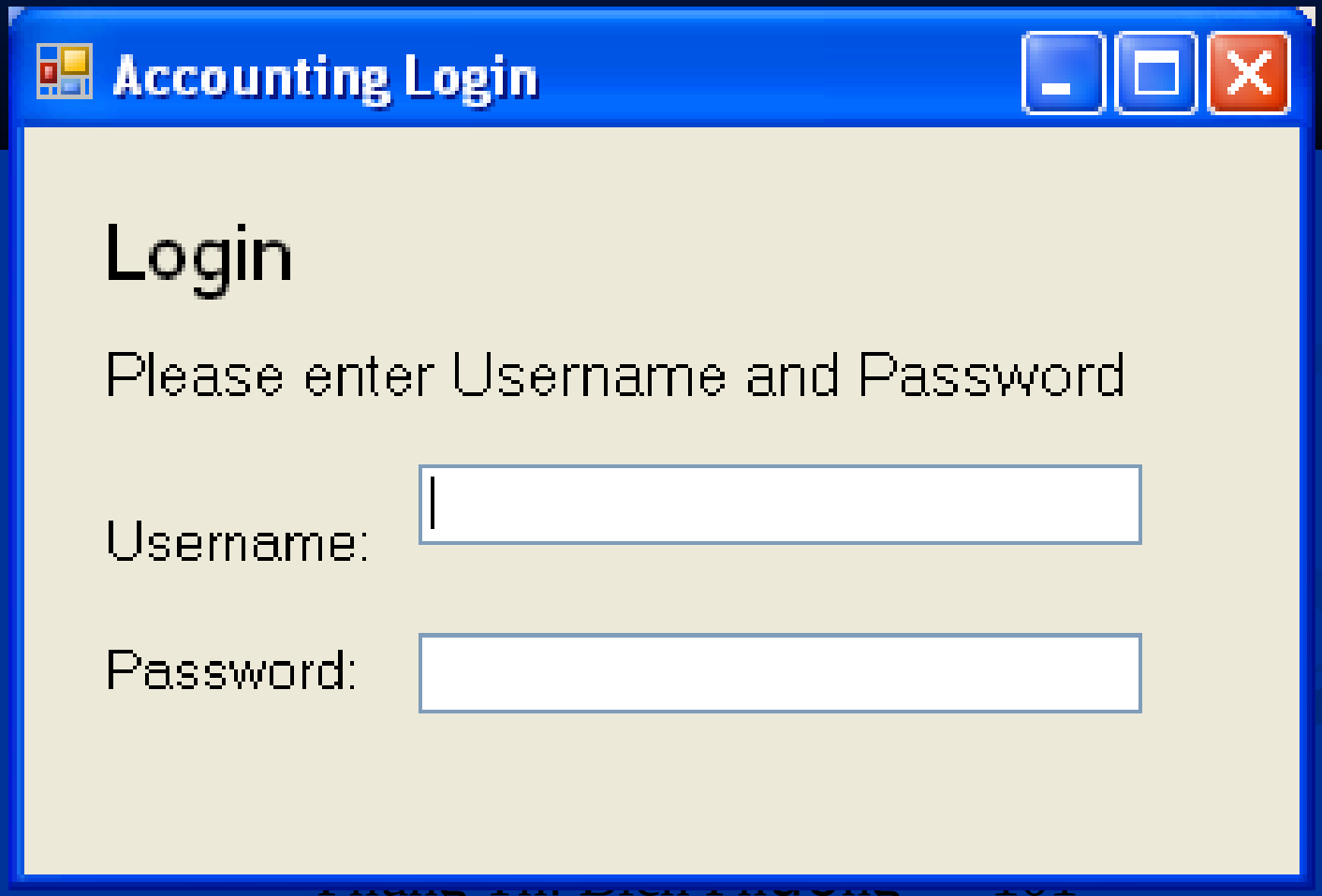
# Giới thiệu Windows Form

# 1.1. Giới thiệu

- Tạo ra các ứng dụng chạy trên máy để bàn có cài đặt .NET Framework 2.0
- Sử dụng không gian tên System. Windows.Forms
- Thiết kế giao diện trực quan sử dụng Visual Studio 2005 IDE.

# 1.1. Giới thiệu

**Ví dụ:**

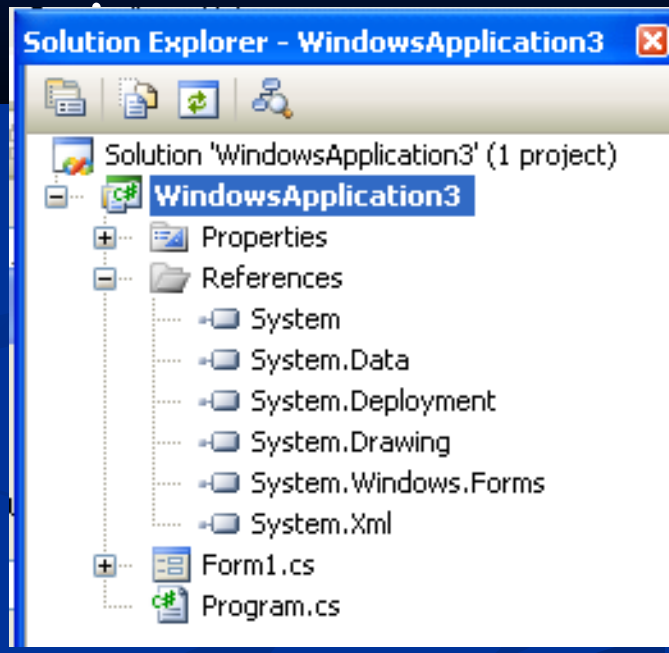


# 1.2. Ứng dụng Windows Forms

- Các chương trình quản lý tài chính, nhân sự, sản xuất, quản lý doanh nghiệp…

# 1.3. Không gian tên

- Khi tạo Project loại Windows Application có 6 không gian tên mặc định:
  - System,
  - System.Data,
  - System.Deployment
  - System.Drawing,
  - System.Windows.Forms
  - System.Xml.

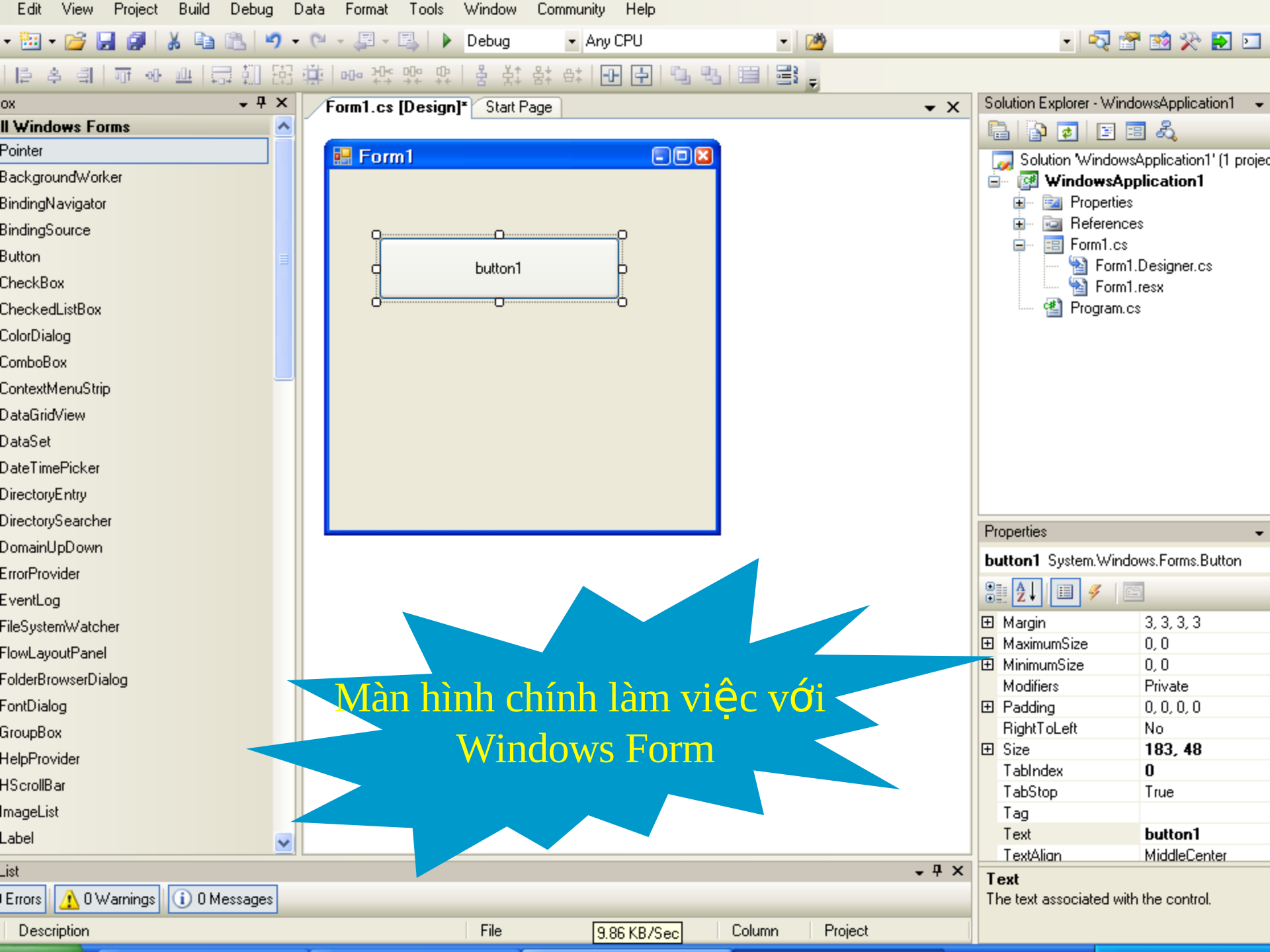


# System.Windows.Forms

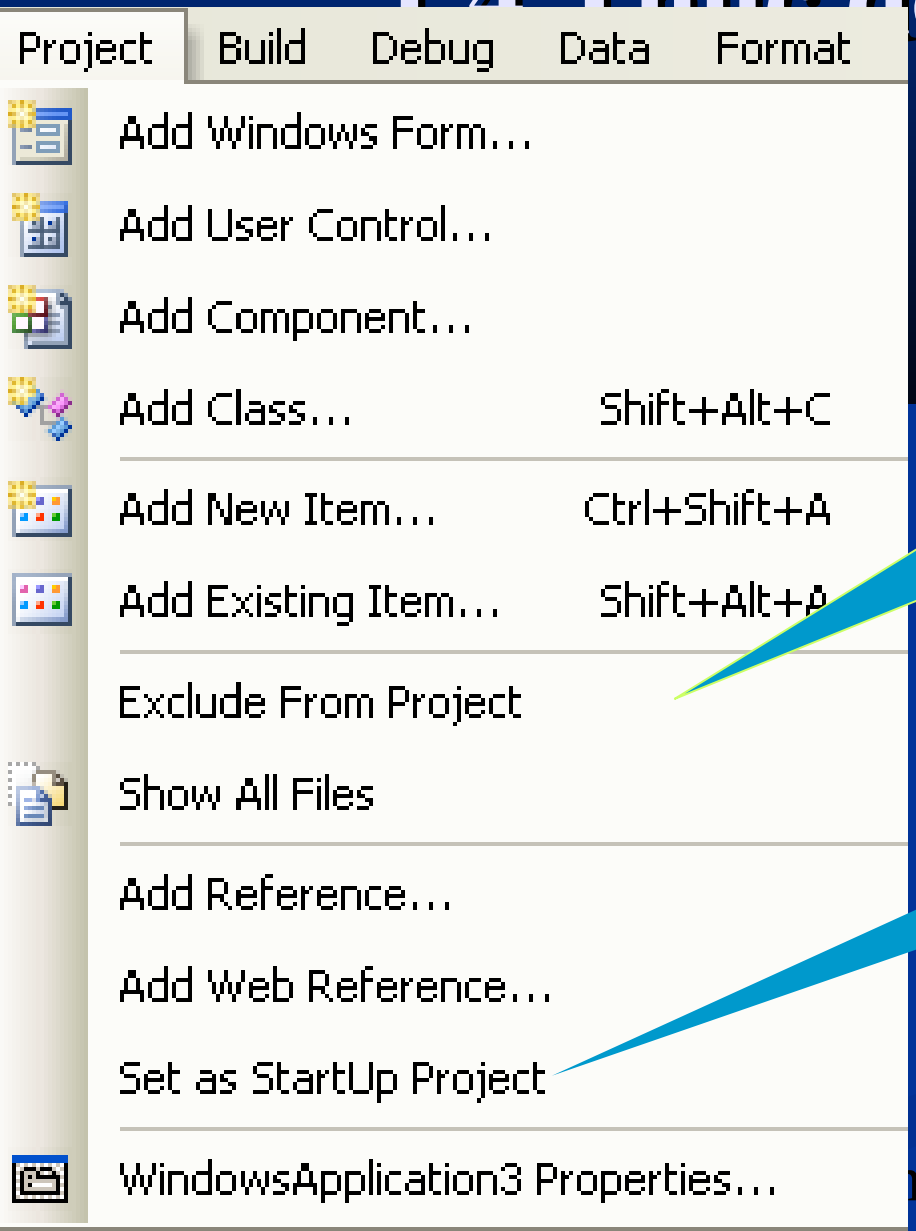
- Là không gian chính cung cấp các lớp dùng để xây dựng các ứng dụng Desktop.
- Các lớp của System.Windows.Forms chia thành các nhóm sau:
  - Control, User Control, Form
  - Menu, Toolbar: ToolStrip, MenuStrip, ContextMenuStrip, StatusStrip
  - Controls: Textbox, Combobox, Label, Listview, WebBrowser, HtmlDocument

# System.Windows.Forms

- Layout: Giúp định dạng và tổ chức các điều khiển trình bày trên Form
- Data và Data Binding: định nghĩa kiến trúc đa dạng cho việc liên kết dữ liệu nguồn hay tệp tin XML vào các điều khiển.
  - VD: DataGridView
- Componets: ToolTop, ErrorProvider, HelpProvider
- Command Dialog Boxes: Làm việc với File, Font, Color, Print. VD: OpenFileDiaglog, SaveFileDiaglog. ColorFileDiaglog…

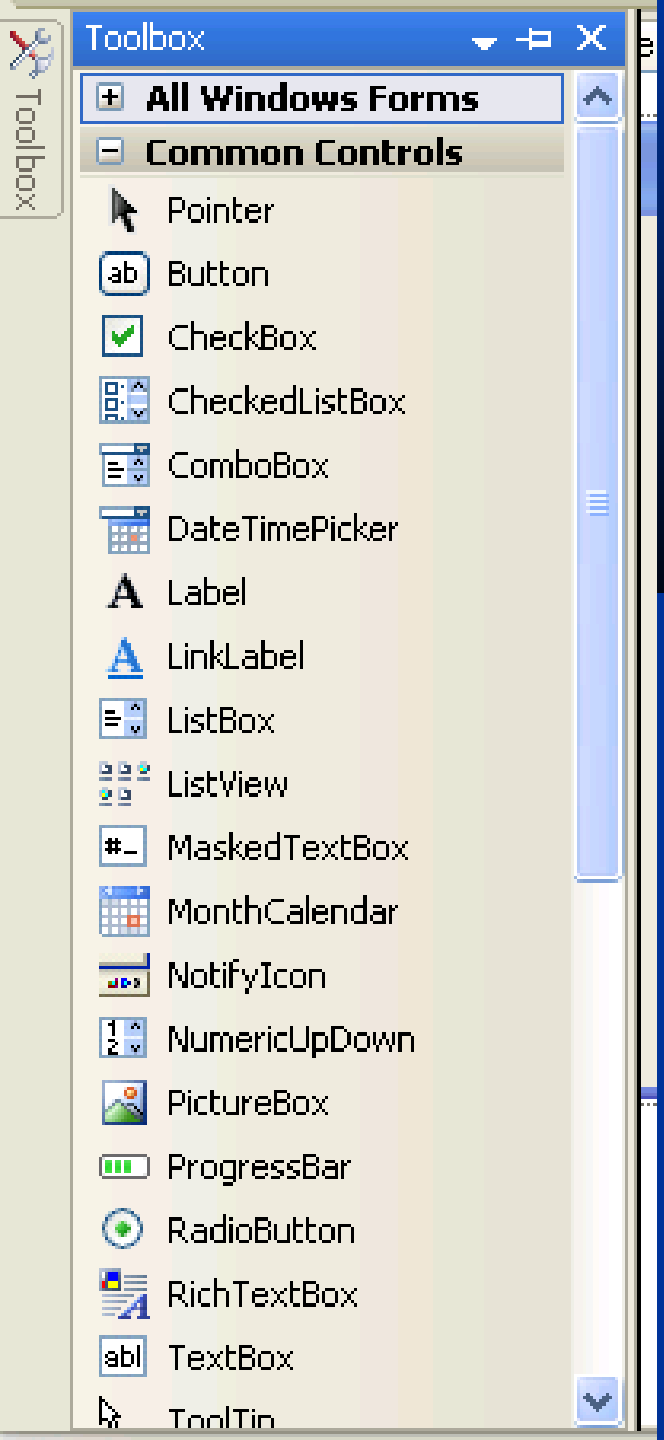

Edit View Project Build Debug Data Format Tools Window Community Help
Debug
Any CPU
Form1.cs [Design]*
Start Page
Form1
button1
Solution Explorer - WindowsApplication1
Solution 'WindowsApplication1' (1 projec
WindowsApplication1
Properties
References
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Pointer
BackgroundWorker
BindingNavigator
BindingSource
Button
CheckBox
CheckedListBox
ColorDialog
ComboBox
ContextMenuStrip
DataGridView
DataSet
DateTimePicker
DirectoryEntry
DirectorySearcher
DomainUpDown
ErrorProvider
EventLog
FileSystemWatcher
FlowLayoutPanel
FolderBrowserDialog
FontDialog
GroupBox
HelpProvider
HScrollBar
ImageList
Label
Properties
button1 System.Windows.Forms.Button
Margin 3, 3, 3, 3
MaximumSize 0, 0
MinimumSize 0, 0
Modifiers Private
Padding 0, 0, 0, 0
RightToLeft No
Size 183, 48
TabIndex 0
TabStop True
Tag
Text button1
TextAlign MiddleCenter
Text
The text associated with the control.
0 Warnings
0 Messages
Description
File
9.86 KB/Sec
Column
Project
Màn hình chính làm việc với
Windows Form

# 1.4 Thực đơn Projector




Loại bỏ đối tượng khỏi Project

Đặt Project khởi động




# 1.5. Hộp công cụ

- Cung cấp danh sách các Component liệt kê theo nhóm, cho phép thiết kế giao tiếp với người dùng.
  - Windows Forms: Windows Control.
- Hiện ToolBox:
  - View \ Toolbox
  - nhấn chọn biểu tượng trên thanh công cụ
  - Ctrl+W và X

# 1.5. Hộp công cụ

Chứa tất cả đối tượng

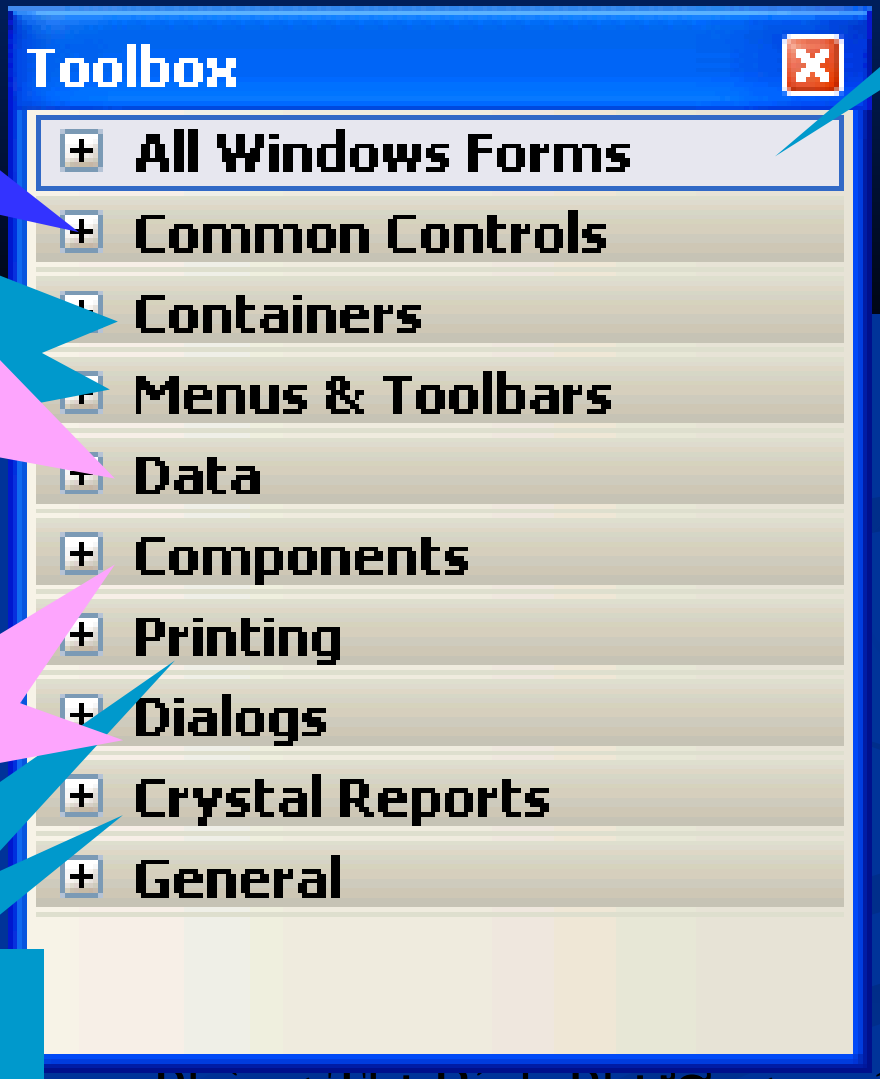
Chứa các điều khiển thông thường: TextBox, Label, Button,

Chứa các điều

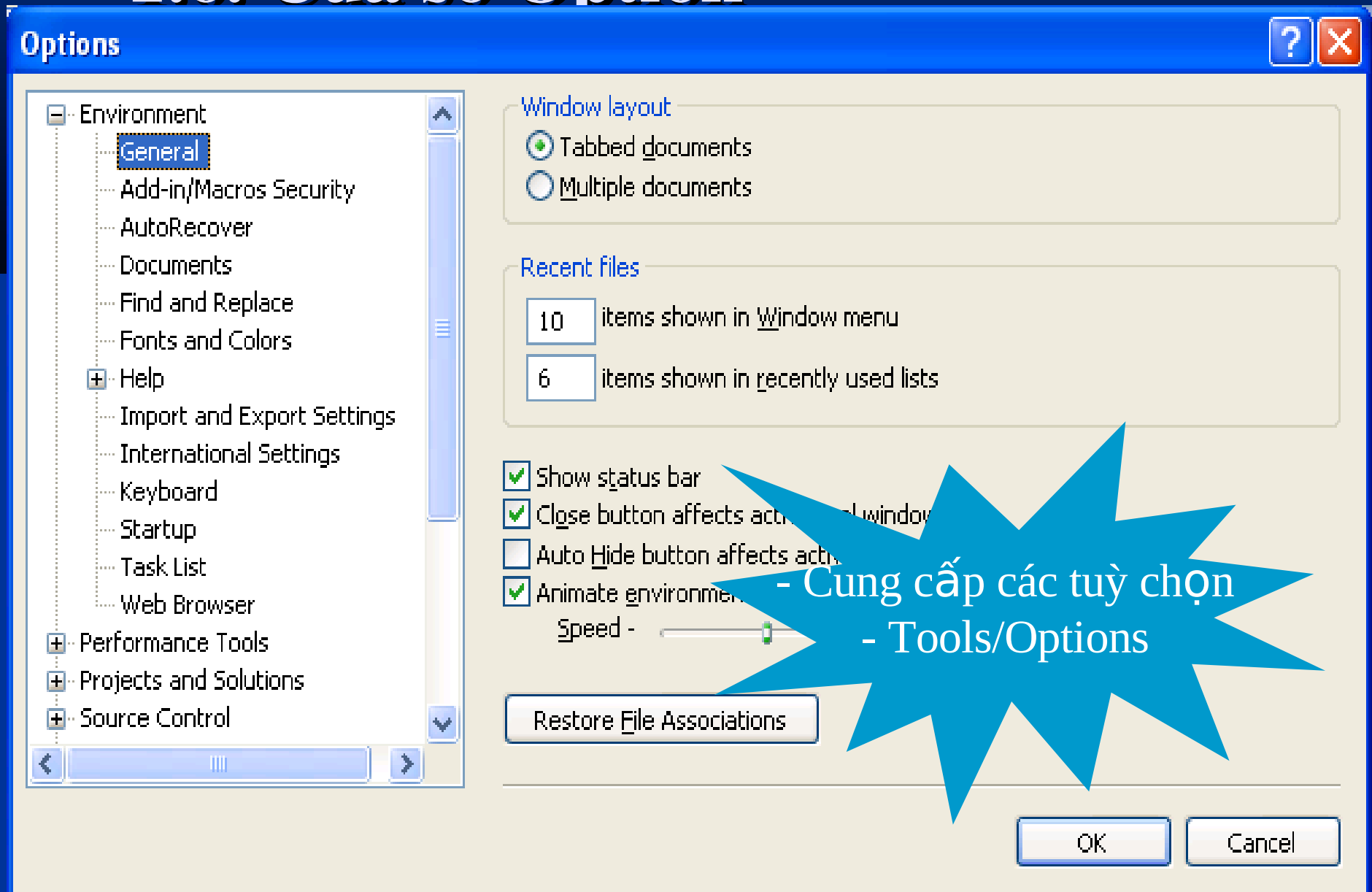
Chứa tất cả đối tượng làm việc với CSDL: DataSet, DataGridView,

**Cung cấp các điều khiển sử dụng để kiểm tra dữ liệu nhập như: ErrorProvider, HelpProvider,…**

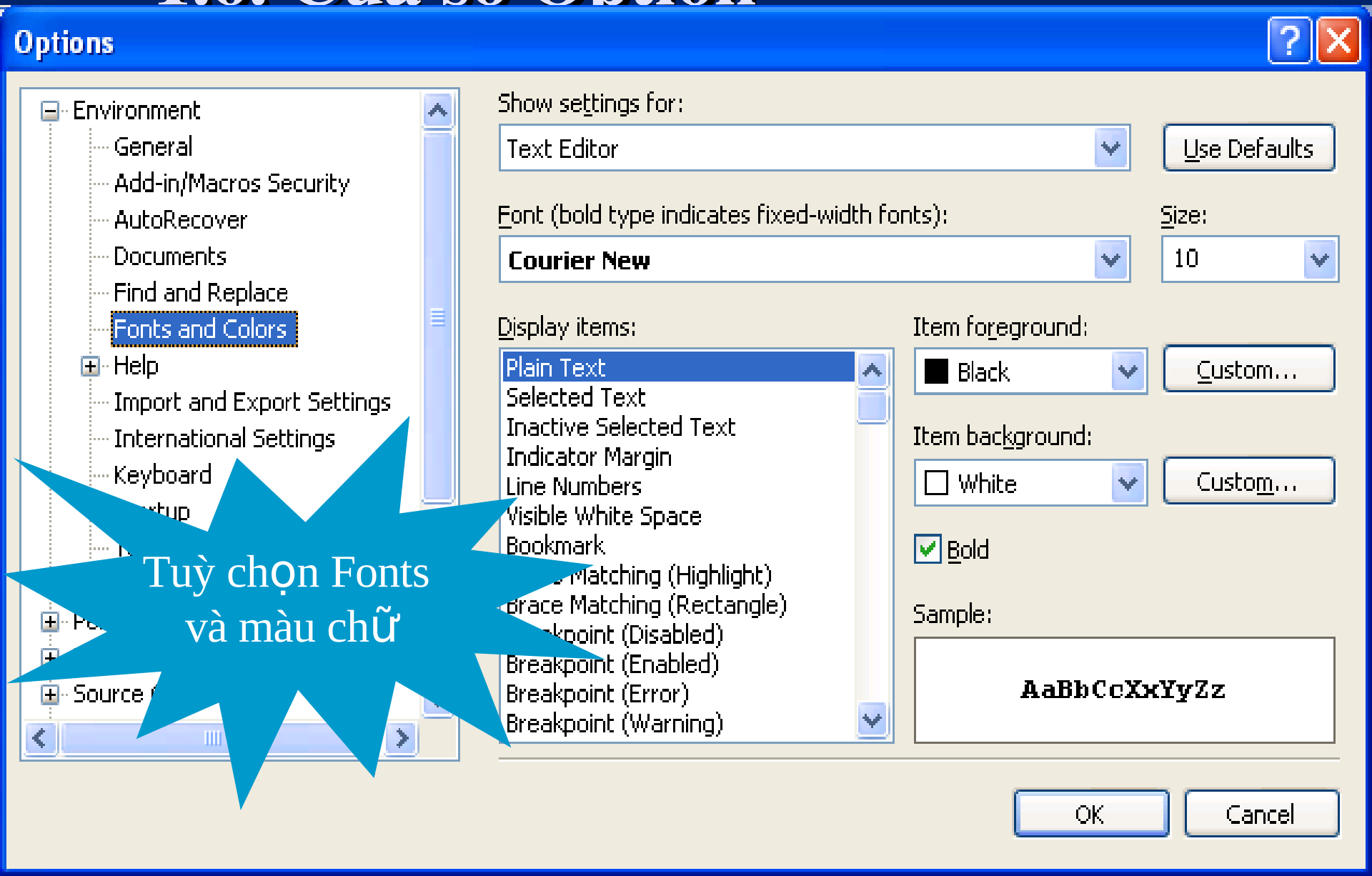
**khiển làm việc với báo cáo**



# 1.6. Cửa sổ Option

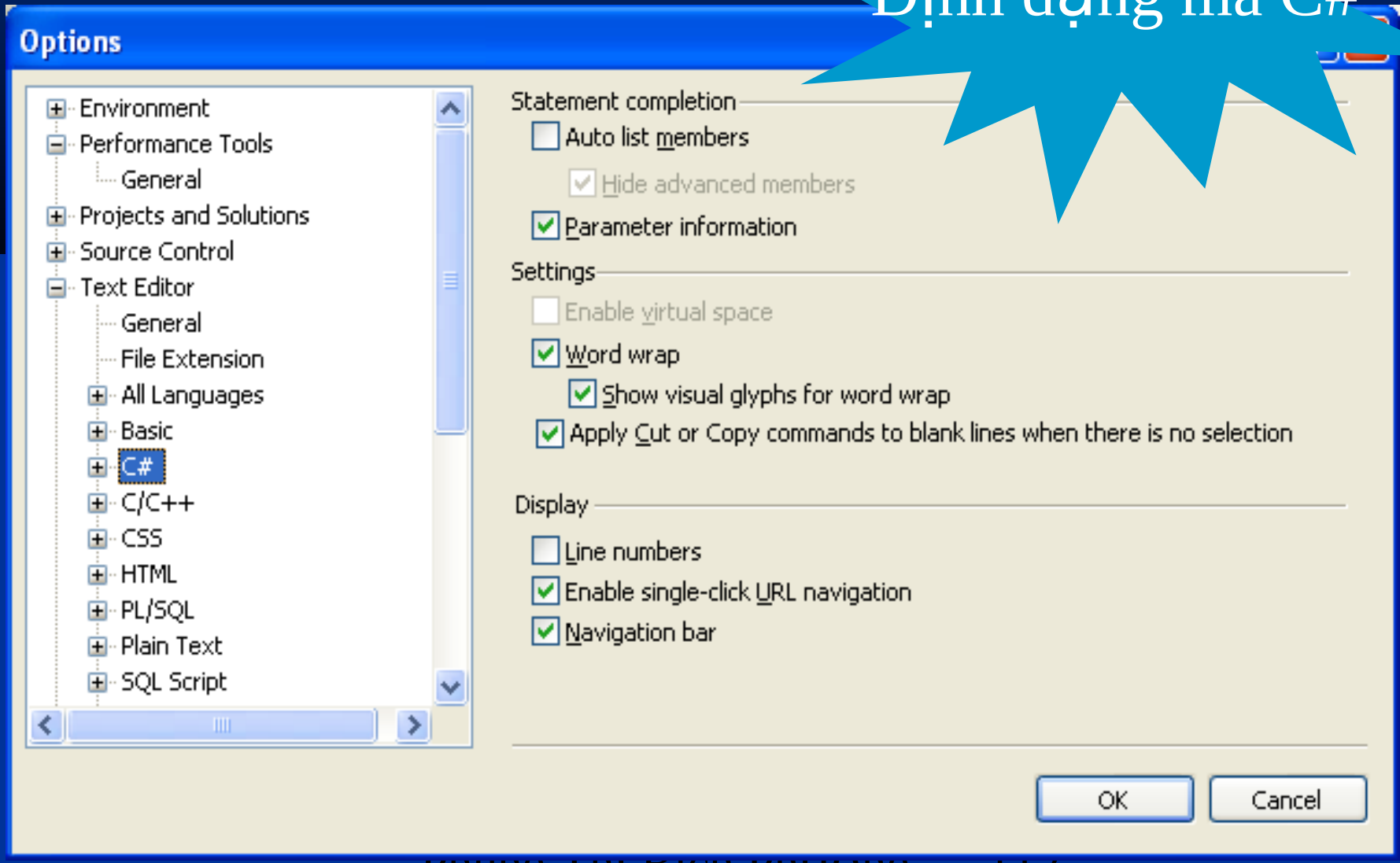


# 1.6. Cửa sổ Option



# 1.6. Cửa sổ Option

Định dạng mã C#



# Chương 2.

# Form và các định dạng Form

# 2.1. Các loại Form

- MDI Form:
  - Form chứa các form khác
  - Thuộc tính isMDIFormContainer=true
  - VD: Form frm=new Form2()

    Frm. isMDIFormContainer=true

    Frm.Show()

  ➔ Tạo Form2 và cho Form2 là MDI Form

# 2.1. Các loại Form

- Child Form:
  - Form nằm trong MDI Form
  - Phải khai báo thuộc tính MDIParent ứng với MDI Form
  - VD: Form Frm=new Form3()

    Frm. isMDIParent=this

    Frm.Show()

  ➔ This là từ khoá chỉ định Form gọi đến Form3 là MDI Form

# 2.1. Các loại Form

- Normal Form:
  - Không phải MDI Form hoặc ChildForm

# Nạp Form

- VD: frm=new Form()
- Frm.Show(): Hiển thị Form
- Frm.ShowDialog(): Form mở ở dạng Modal. Form modal không cho phép người sử dụng dùng Form khác trtừ khi Form này được đóng lại

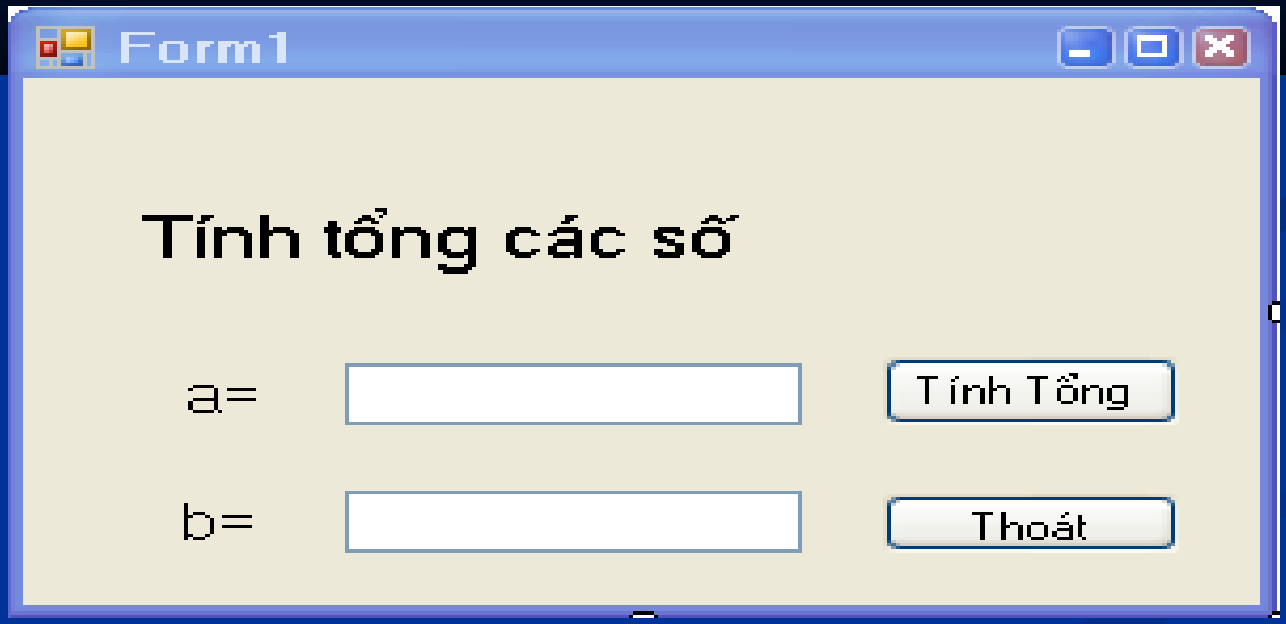
# Tạo Form lúc thi hành

- Sử dụng từ khoá New để tạo Form, sau đó gán các thuộc tính cho Form
- VD:

```
Form Frm=new Form()
Frm.Text="New Form";
Frm.Show();
```

# Form kế thừa

VD: Thiết kế Form1 như sau:



# Form kế thừa

- **Thêm Form2: Project\Add Windows Form**
- **D-Click vào Form2 xuất hiện**

```
public partial class Form2 : Form1
{
    public Form2()
    {
        InitializeComponent();
    }
```

Thay class Form2: Form bởi class Form2: Form1

# Form kế thừa

- Kết quả


Có thể thiết kế lại Form2

# 2.2. Các thuộc tính của Form

- Nhóm thuộc tính nhận dạng
  - Name: Tên duy nhất của đối tượng Form trong Project
  - Text: Chuỗi hiển thị trên thanh tiêu đề
  - ShowIcon=True: Cho hiện Icon góc trên bên trái; =False: Không hiện
  - ShowInTaskBar: =True: Khi chạy hiện biểu tượng trên TaskBar; False: Không hiện
  - Icon: Cho phép chỉ định tệp tin *.ico làm biểu tượng trên thanh tiêu đề của Form

# 2.2. Các thuộc tính của Form

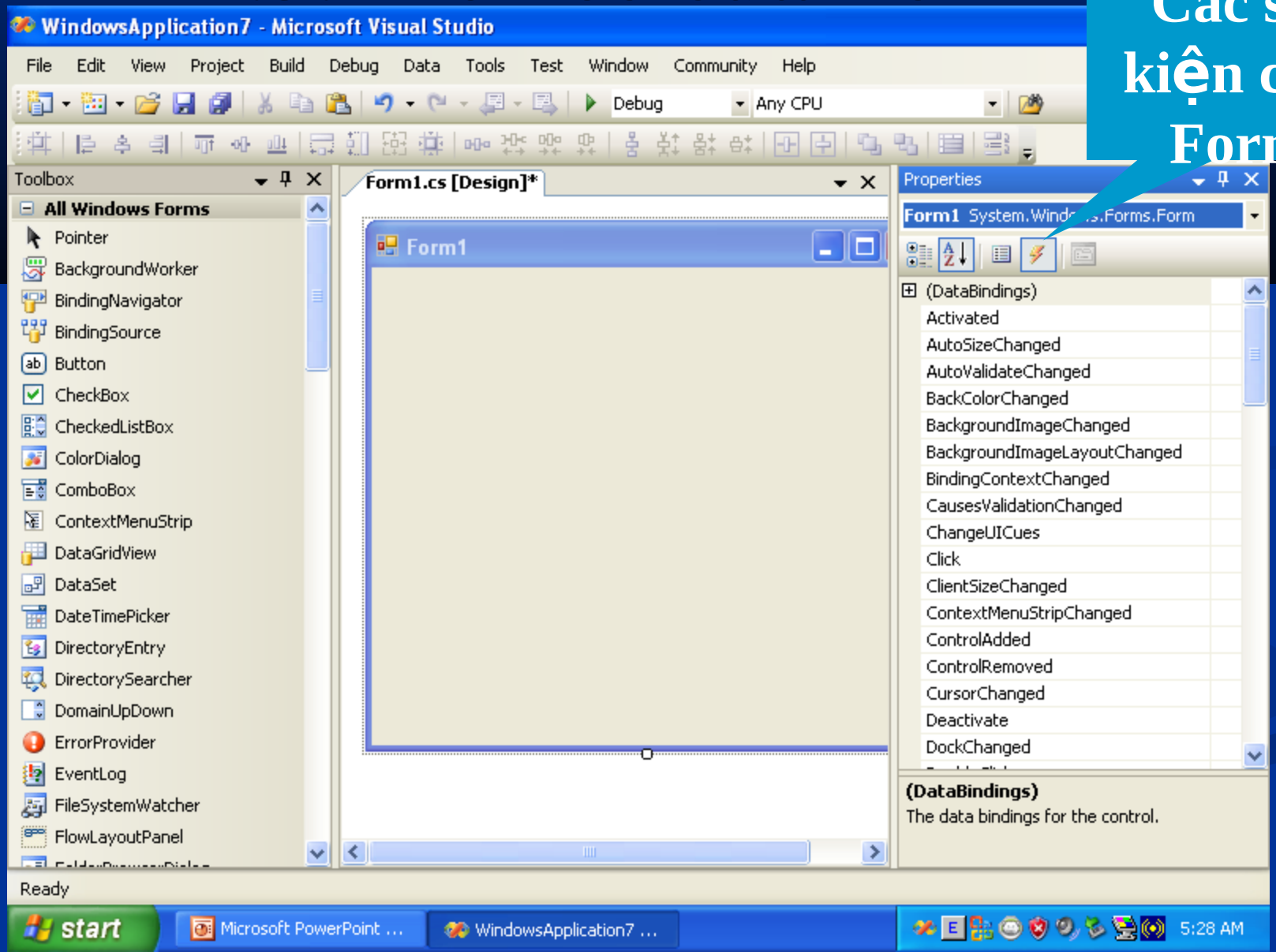
- Nhóm thuộc tính Định dạng
  - BackColor: Màu nền của Form
    - VD: Form1.BackColor=Color.Azủe;
  - ForeColor: Màu của các chuỗi trên các Control của Form
  - StartPossition: Vị trí hiển thị Form
  - WindowStates: =Minimized (thu nhỏ), Maximized (phóng to), Nomal (trạng thái như thiết kế)
  - isMDIContainer: =True (Form được chọn là MDI Form); False: không
  - ControlBox

# 2.3. Biến cố của Form

- FormClosed: Thực hiện khi Form đã đóng
- FormClosing: SỰ kiện khi đang đóng Form
- Click: Sự kiện khi Click vào Form
- Activated: Xảy ra khi Form được kích hoạt bằng mã hay do tác động của người sử dụng
- Disactiave: Xảy ra khi Form khác kích hoạt trên màn hình.
- Load: Xả ra khi nạp Form
- KeyPress: Xảy ra khi 1 phím được nhấn
- Resize: Xảy ra khi thay đổi kích thước Form

# 2.3. Biến cố của Form

Các sự kiện của Form



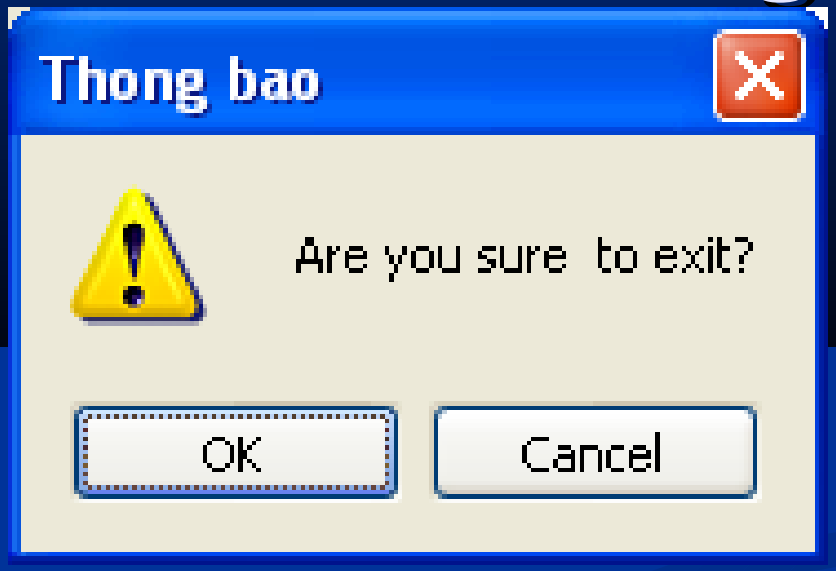
# Ví dụ: Biến cố Load Form

```
private void Form1_Load(object sender, EventArgs
  e)
    {
   MessageBox.Show("Dang Load Form");
   //…
    }
```

# Ví dụ: Biến cố Click form

```
private void Form1_Load(object sender, EventArgs
  e)
    {
   MessageBox.Show("Dang Load Form");
   //…
    }
```

# Ví dụ: Biến cố Closing Form



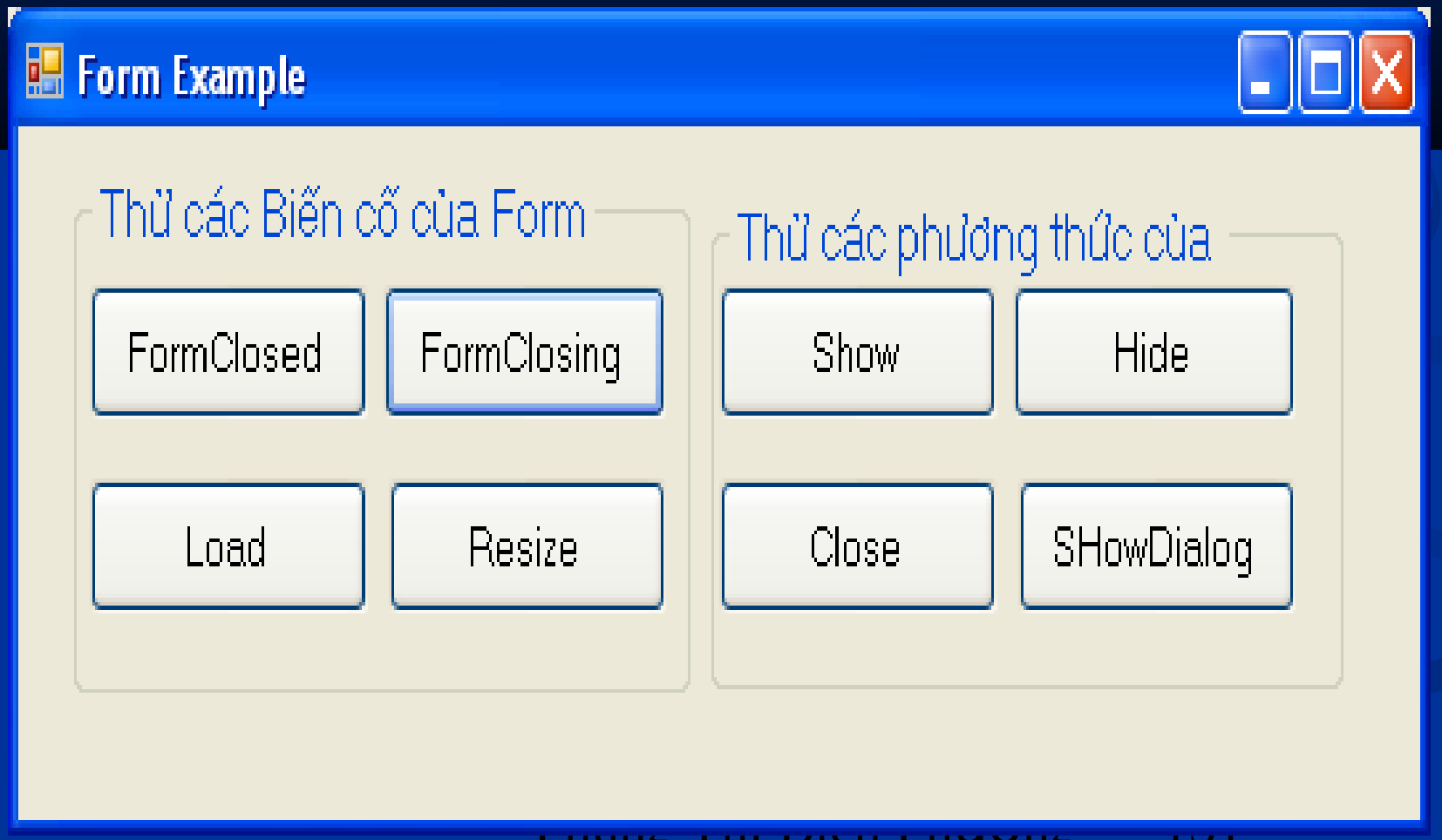

```
 private void Form1_FormClosing(object sender,
FormClosingEventArgs e)
        {
 MessageBox.Show("Are you sure  to exit?", "Thong bao",
MessageBoxButtons.OKCancel, MessageBoxIcon.Warning);

        }
```

# 2.4. Phương thức của Form

- Close(): Dùng để đóng Form
  - Vd: this.Close()
- Hide(): Ẩn form
  - VD: this.hide
- Show(): Nạp form
  - VD: Frm.Show()
- ShowDialog(): Nạp Form dạng Modal
  - VD: frm.ShowDialog

# Thực hành

## Thử các biến cố và phương thức của Form



# Chương 3.
# Điều khiển thông thường

# Thuộc tính chung của các điều khiển

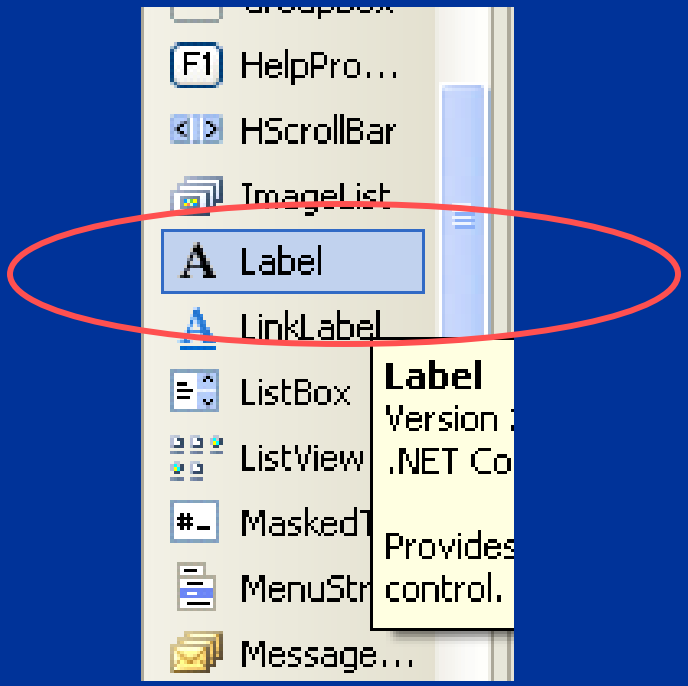
- BackColor: Màu nền của điều khiển
- ForeColor: Màu chữ của chuỗi trình bày trên điều khiển
- Text: Chuỗi trình bày trên điều khiển
- Visible: Thuộc tính che dấu hay hiển thị điều khiển
- Name: Tên của điều khiển
- Looked: Khoá không cho di chuyển trên Form

# Sự kiện chung của các điều khiển

- Click: Xảy ra khi người dùng nhấn chuột phải
- MouseMove: Xảy ra khi người dùng di chuyển chuột qua vùng làm việc cảu điều khiển
- MouseUp: Nhấn chuột xuống vùng làm việc cảu điều khiển rồi thả ra
- MouseDown: Nhấn chuột xuống vùng làm việc cảu điều khiển
- Move: Xảy ra khi di chuyển điều khiển bằng mã hay bởi người sử dụng
- REsize: Xảy ra khi kích thước điều khiển được thay đổi bằng mã hay bởi người sử dụng

# 3.1. Điều khiển Label

- Trình bày thuộc tính dạng tiêu đề, chú giải cho các điều khiển khác (đã quen thuộc)



# 3.1. Điều khiển Label

- BorderStyle: Đường viền của điều khiển
- Font: Kích thước và Font chữ
- TextAlign: Căn chỉnh

# 3.1. Điều khiển Label

- Ví dụ

```
//Khai báo và khởi tạo đối tượng Label
Void CreatControls()
{
    Label lb=new Label();
    Lb.Text="This is Label Object";
    this.Controls.Add(lb);
}
```

# 3.2. Điều khiển TextBox

- Dùng để nhập dữ liệu
- Một số thuộc tính:
  - BorderStyle: Kiểu đường viền của điều khiển
  - CharacterCasing: Định dạng chuỗi nhập vào chuyển sang kiểu chữ hoa (Upper), chữ thường (Lower) hay mặc định (Normal)
  - Enabled: Vô hiệu hoá hay cho phép sử dụng
  - Maxlength: Số ký tự cho phép nhập
  - MultiLine : Giá trị True cho phép nhập nhiều dòng
  - PasswordChar: Giá trị nhập được thay thế bởi ký tự khai báo trong thuộc tính này (Multiline=False)

# 3.2. Điều khiển TextBox

- Một số thuộc tính:
  - ReadOnly: =True chỉ cho phép đọc giá trị
  - ScrollBars: Nếu thuộc tính MultiLine=true thì cho phép hiện thanh trượt hay không (Vertical - Cuộn dọc, Horizontal - Cuộn ngang, both - Cả 2 thanh cuộn, none – Không có thanh cuộn)
  - WordWrap: Tự động xuống dòng nếu chuỗi giá trị dài hơn kích thước của điều khiển

# 3.2. Điều khiển TextBox

- Một số biến cố
  - MouseClick: Xảy ra khi Click vào Textbox
  - MouseDoubleClick: Xảy ra khi Click đúp vào Textbox
  - TextChanged: Xảy ra khi chuỗi trên điều khiển thay đổi

# 3.3. Điều khiển Button

- Cho phép người dùng chuột để nhấn, phím Enter hay phím Spacebar nếu điều khiển này đang được kích hoạt
- Các thuộc tính, biến cố (giống VB6.0)
  - Lưu ý: thuộc tính Caption trong VB ←→ thuộc tính Text trong C#

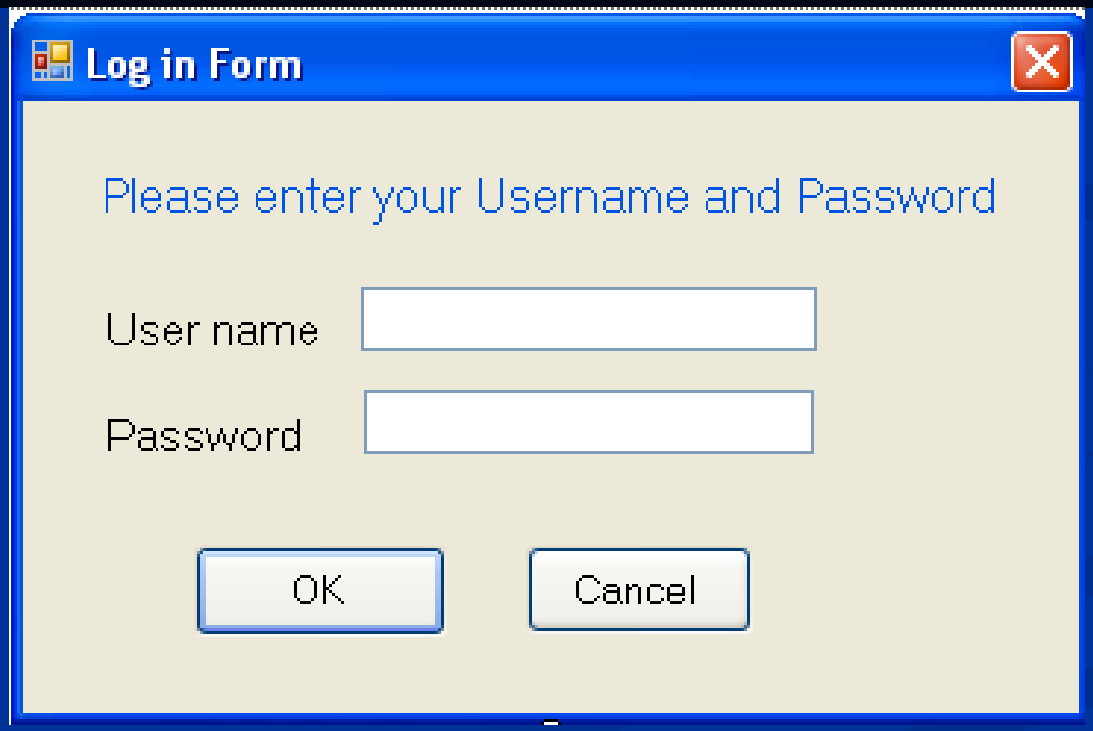
# 3.3. Điều khiển Button

- Khai báo và khởi tạo đối tượng Button sau đó thêm vào Form

```
Button btn=new Button();
btn.Name="btnSave";
btn.Text="&Save";
this.Controls.Add (btn);
```

# Ví dụ 1

- Tạo Form đăng nhập hệ thống như sau:



# Ví dụ 1

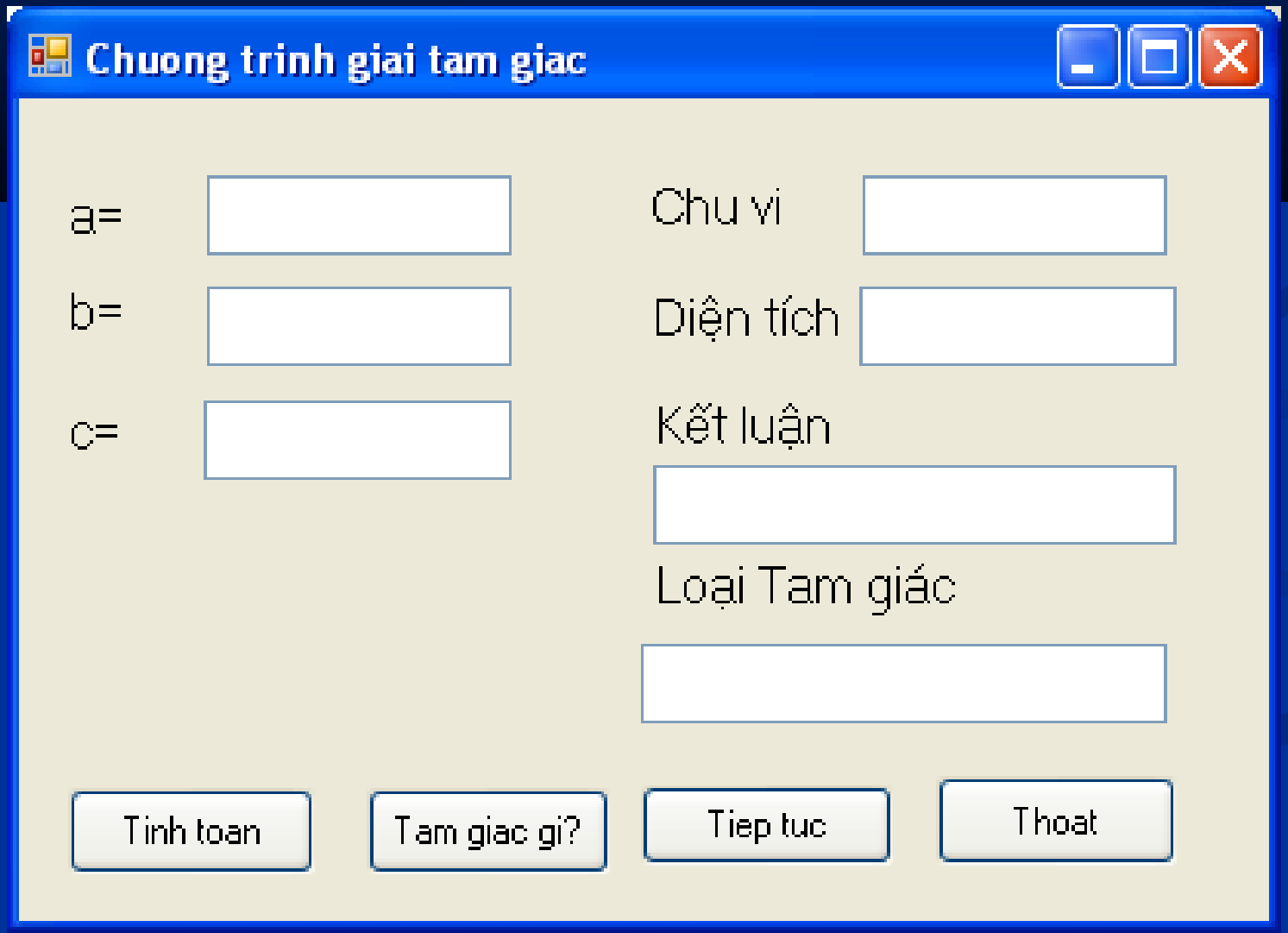
- Yêu cầu:
  - Nếu Username khác rỗng → Nút OK được kích hoạt
  - Không nhập Password mà nhấn OK→ có thông báo yêu cầu nhập Password
  - Nhập sai Uername, Password → Thông báo nhập sai, không cho đăng nhập hệ thống
  - Nhập Username="admin" và Password = "123456" → có thông báo đăng nhập thành công và hiện Form chính của chương trình

# Ví dụ 2

- Viết chương trình nhập 3 số a, b, c vào 3 textbox và kiểm tra 3 số có là 3 cạnh tam giác hay không? Nếu là 3 cạnh tam giác thì tính diện tích, chu vi tam giác đó và kiểm tra xem đó là tam giác gì?

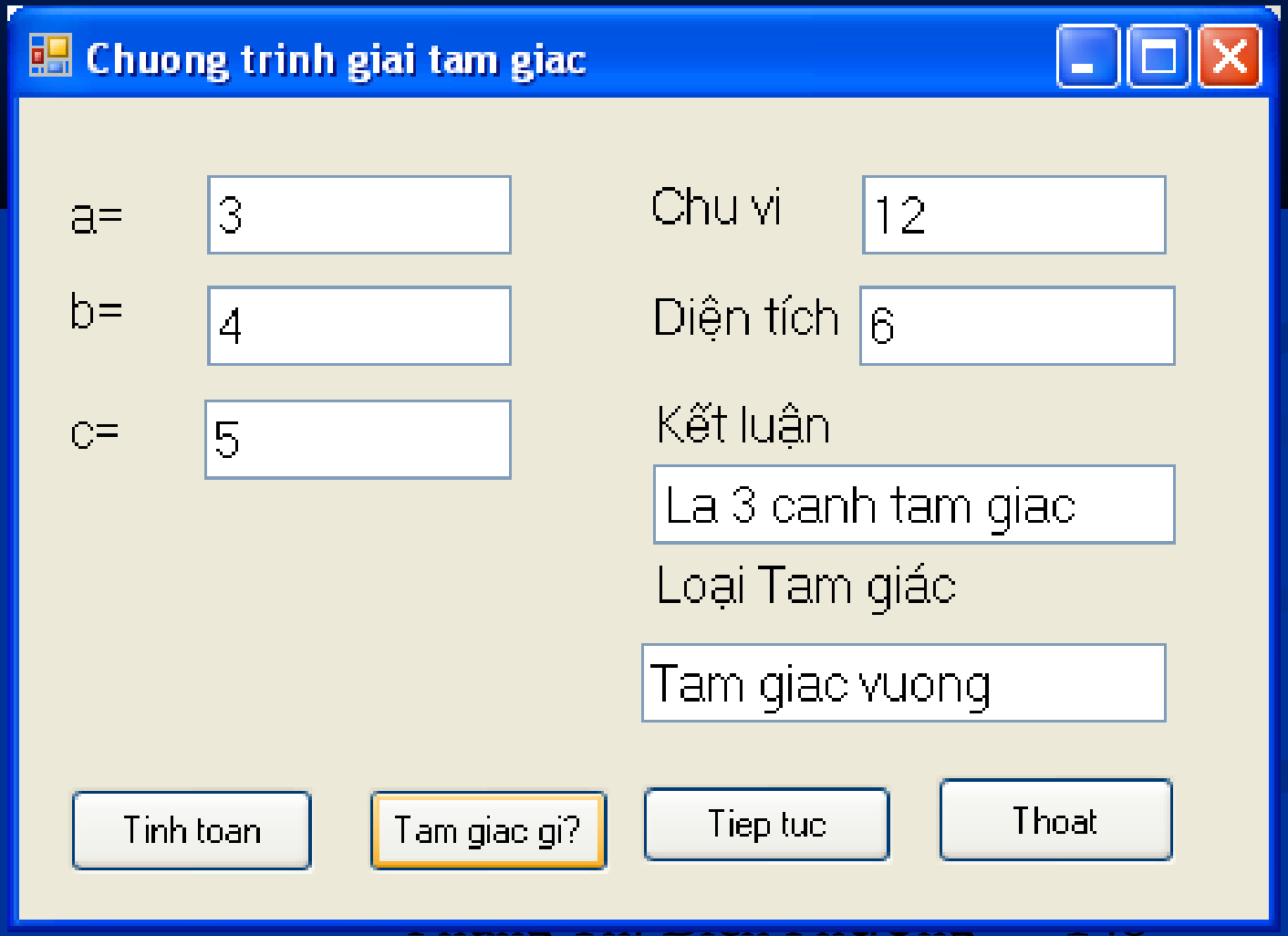
# Ví dụ 2

**Thiết kế Form như sau:**



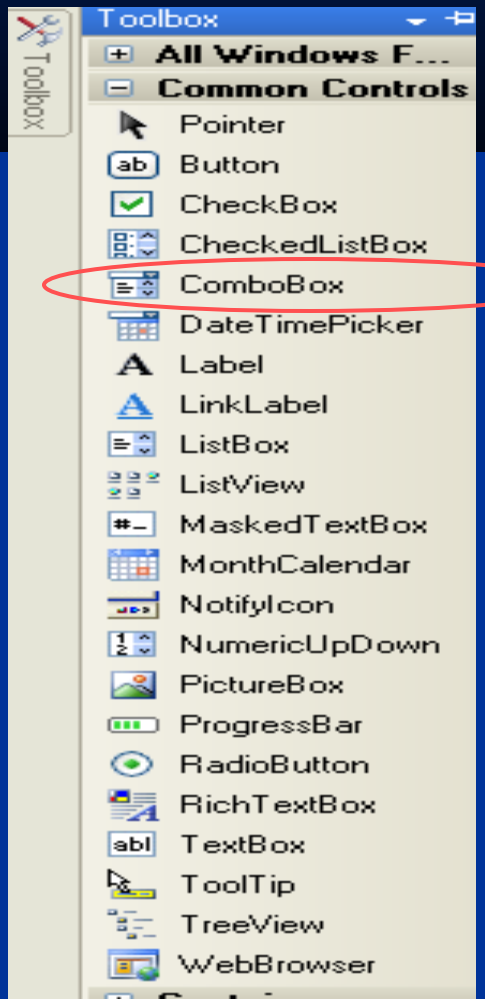
# Ví dụ 2

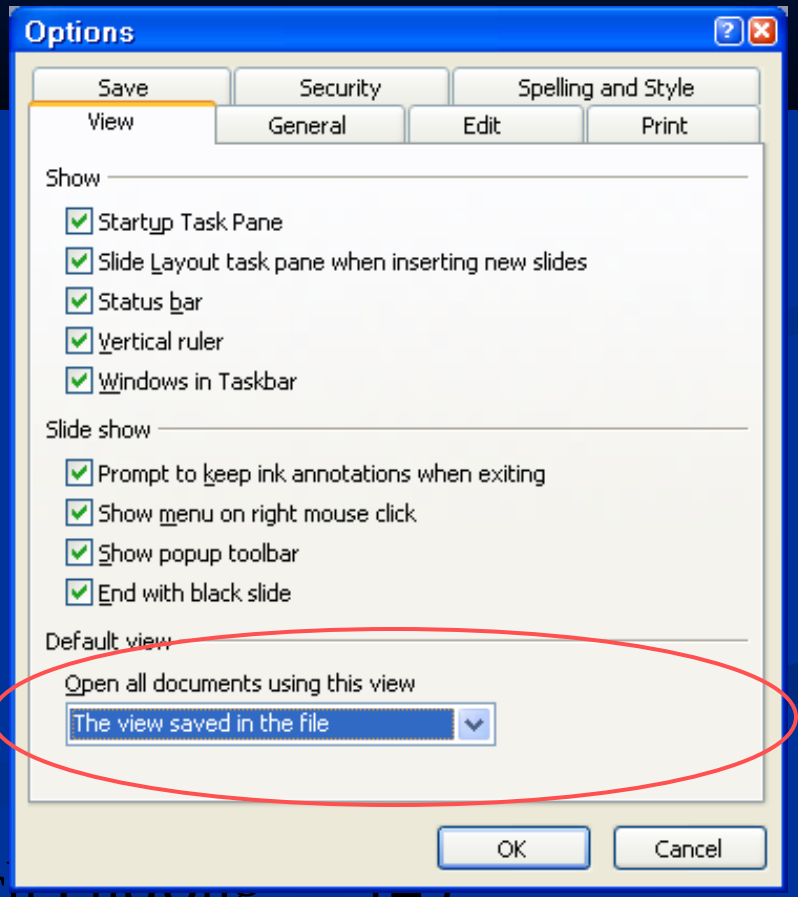
## Kết quả thực hiện chương trình



# 3.3. Nhóm điều khiển ComboBox, ListBox

- ComboBox: Giống VB



# 3.3. Nhóm điều khiển ComboBox, ListBox

- ComboBox – Một số thuộc tính
  - DataSource: Tập dữ liệu điền vào điều khiển
  - Items: Tập các phần tử có trong điều khiển, có thể sử dụng phương thức Add và AddRange để thêm phần tử vào ComboBox

# 3.3. Nhóm điều khiển ComboBox, ListBox

- ComboBox: Ví dụ Thêm các mục vào ComboBox1 bằng phương thức Add

```
private void button1_Click(object sender, EventArgs e)
    {
        for (int i = 1; i < 10; i++ )
            comboBox1.Items.Add("Phan tu " + i.ToString());
    }
```

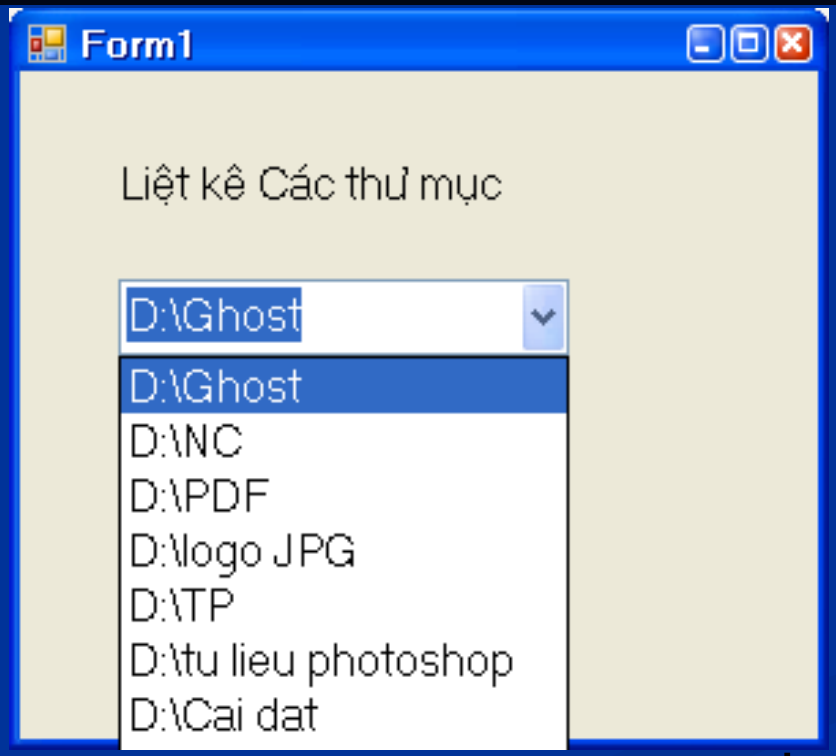
# 3.3. Nhóm điều khiển ComboBox, ListBox

- ComboBox: Ví dụ Thêm các mục vào ComboBox1 bằng phương thức AddRange

```
private void button2_Click(object sender, EventArgs e)
    {
        string[] week = new string[7]
        { “Sun", “Mon", “Tue", "Wed", "Thu", "Fri", "Sat" };
        comboBox2.Items.AddRange(week);
    }
```

# 3.3. Nhóm điều khiển ComboBox, ListBox

- ComboBox: Ví dụ liệt kê các thư mục. Sử dụng phương thức DataSource



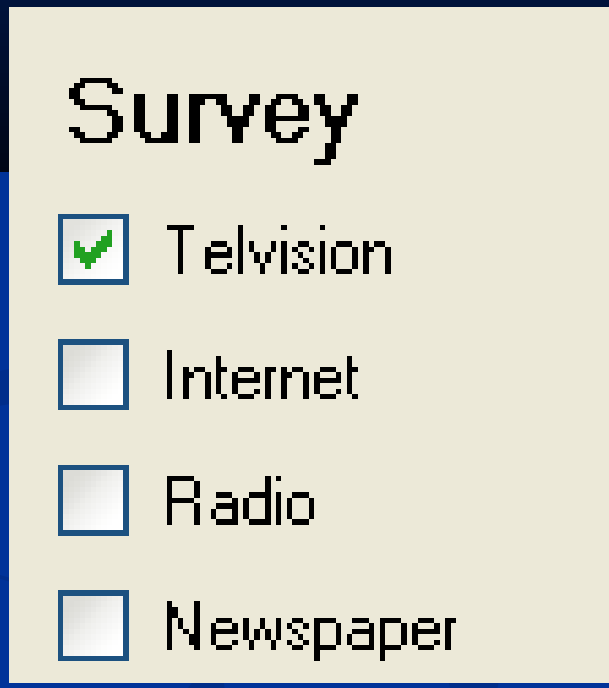
# 3.3. Nhóm điều khiển ComboBox, ListBox

- ListBox: Giống VB
  - Các thuộc tính và phương thức: Tương tự COmboBOx

  ➔ SV tự tìm hiểu

# 3.4. Nhóm điều khiển CheckBox, RadioButton

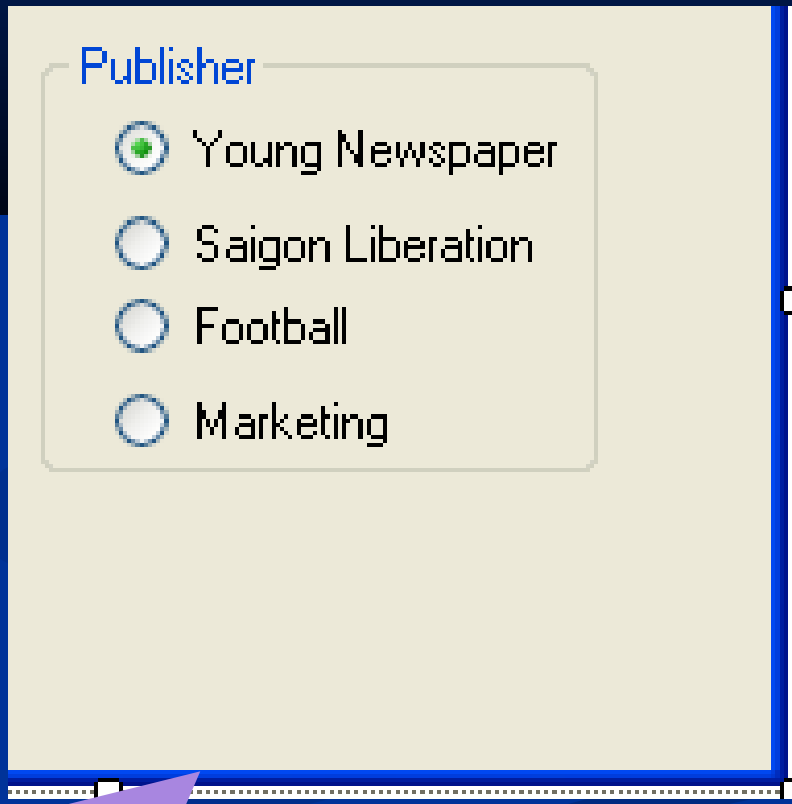
- CheckBox: Giống VB
- Một số thuộc tính đáng chú ý:
  - Checked: Trạng thái chọn (true), không chọn (False)
  - CheckState: Trạng thái của điều khiển CHeckBox đang chọn, có 3 trạng thái: Checked, Unchecked, Indeterminate.




SV tự tìm hiểu

# 3.4. Nhóm điều khiển CheckBox, RadioButton

- RadioButton: Giống VB




SV tự tìm hiểu

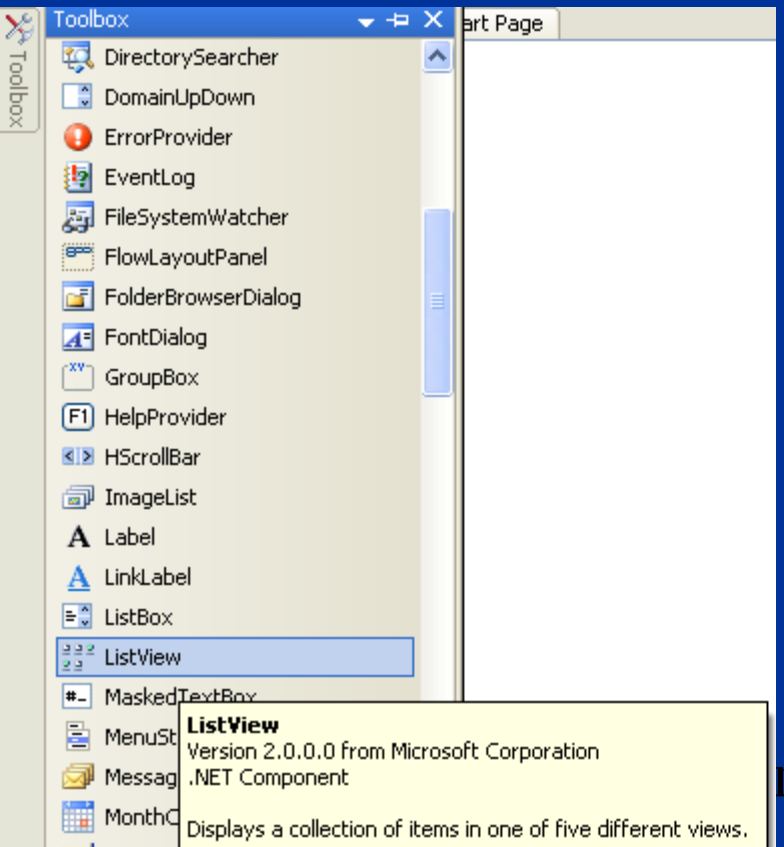
# Chương 4.
# ĐIỀU KHIỂN ĐẶC BIỆT

# 4.1. Điều khiển ImageList

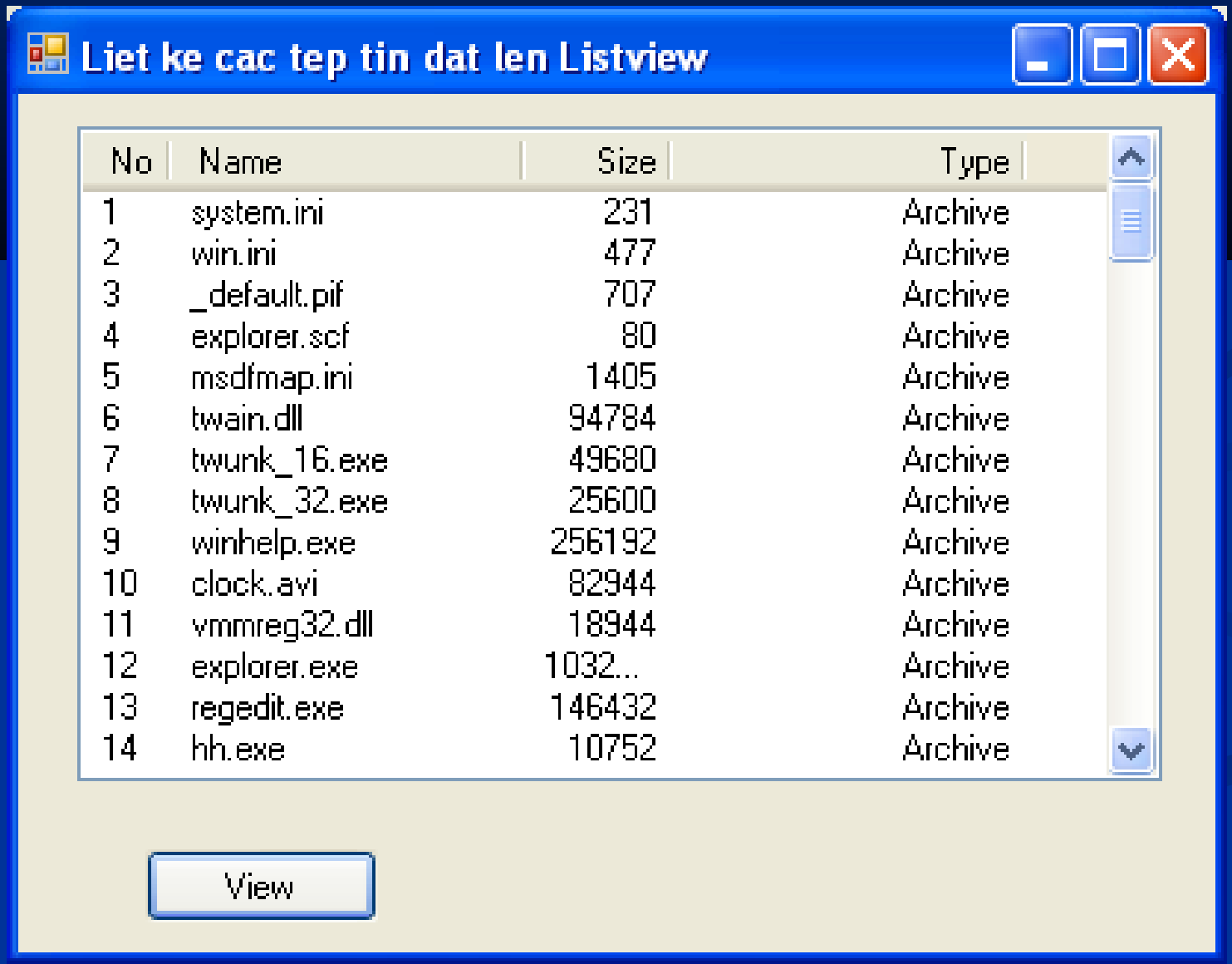
- Chứa mảng các Picture, thường sử dụng với Listview, Treeview
- Giống VB 6.0
- Ví dụ:

# 4.2 Điều khiển ListView

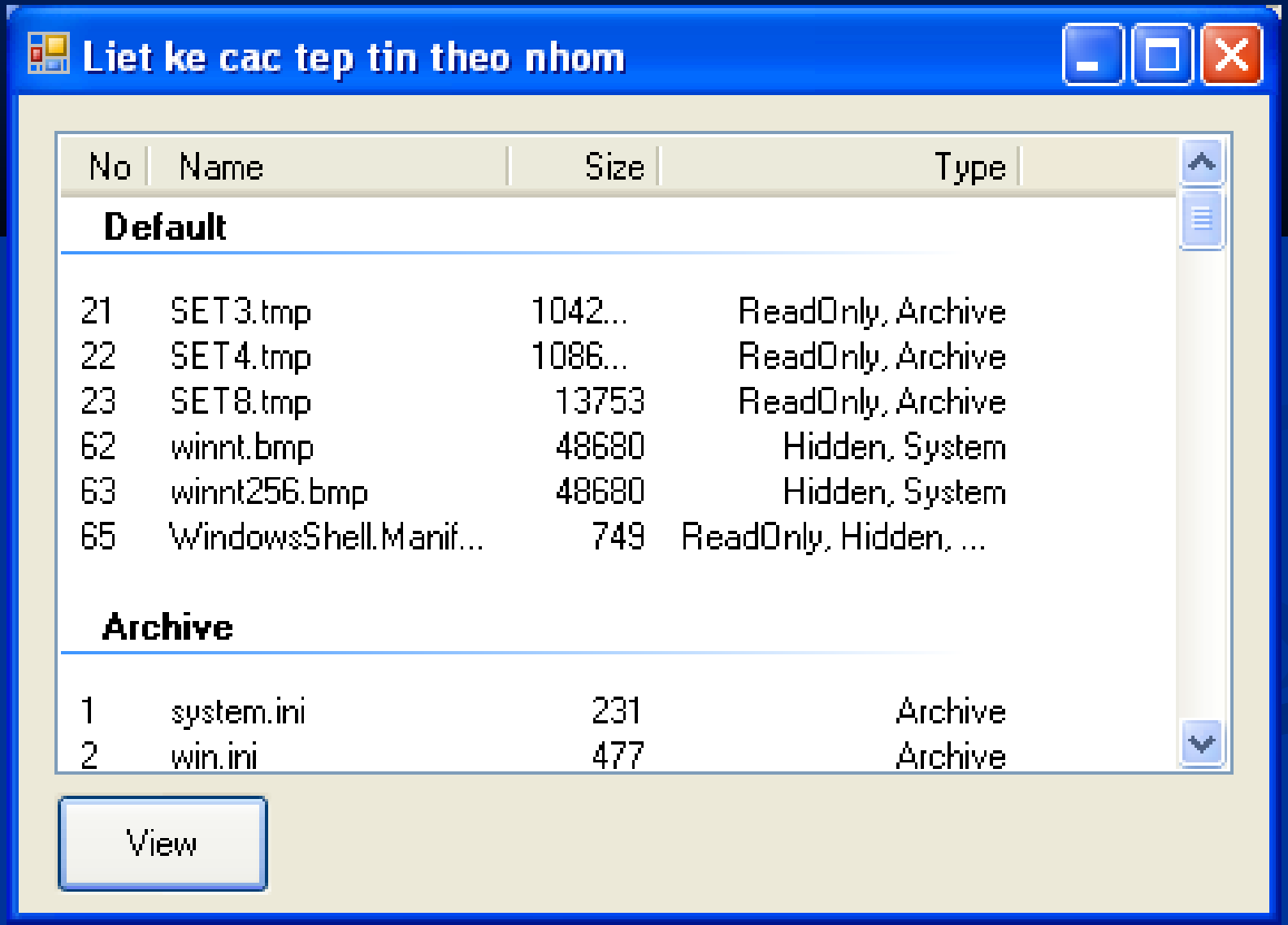
- Trình bày các phần tử dạng danh sách với nhiều hình dạng khác nhau.



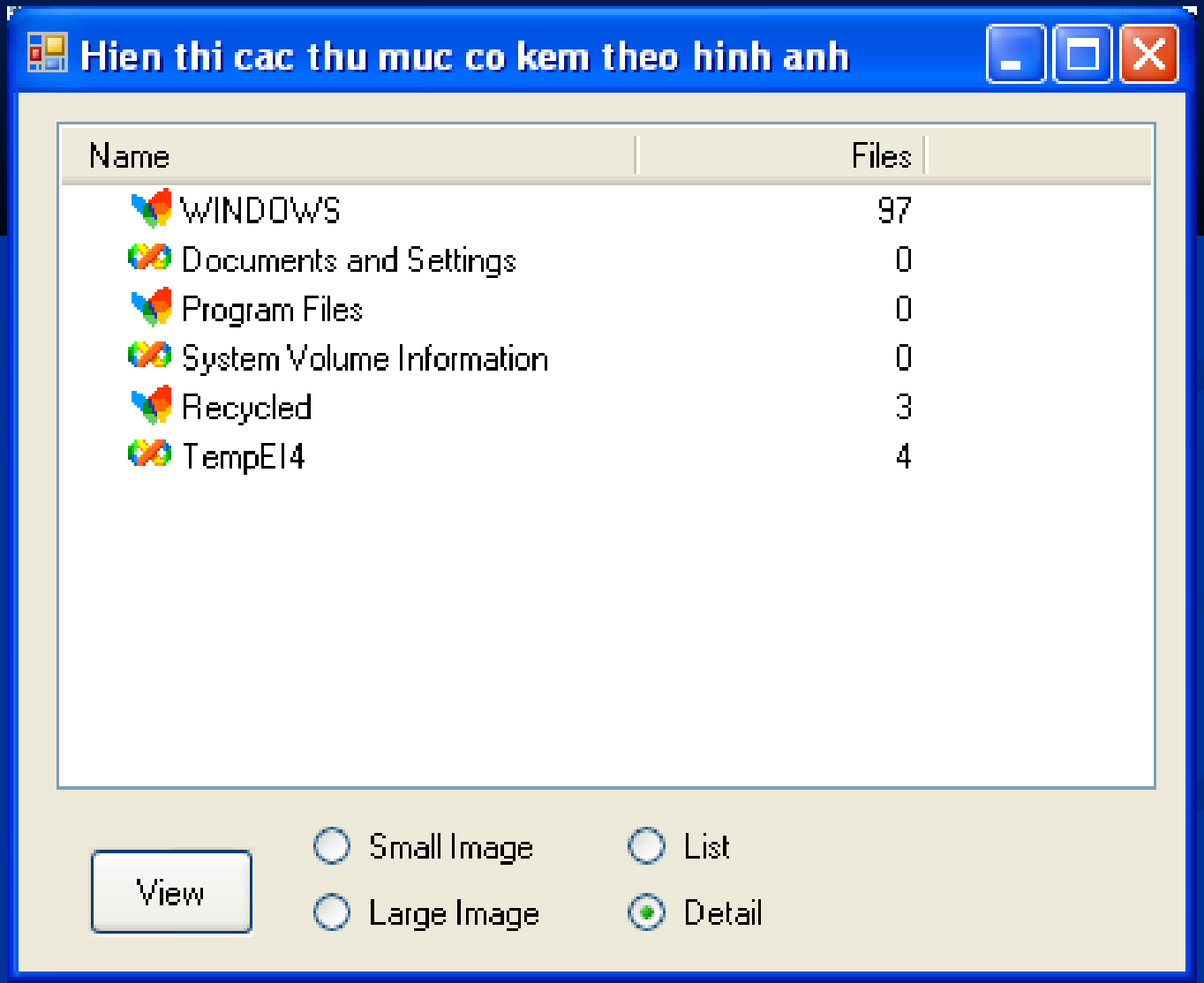
# 4.2. Điều khiển ListView



# 4.2 Điều khiển ListView



# 4.2. Điều khiển ListView

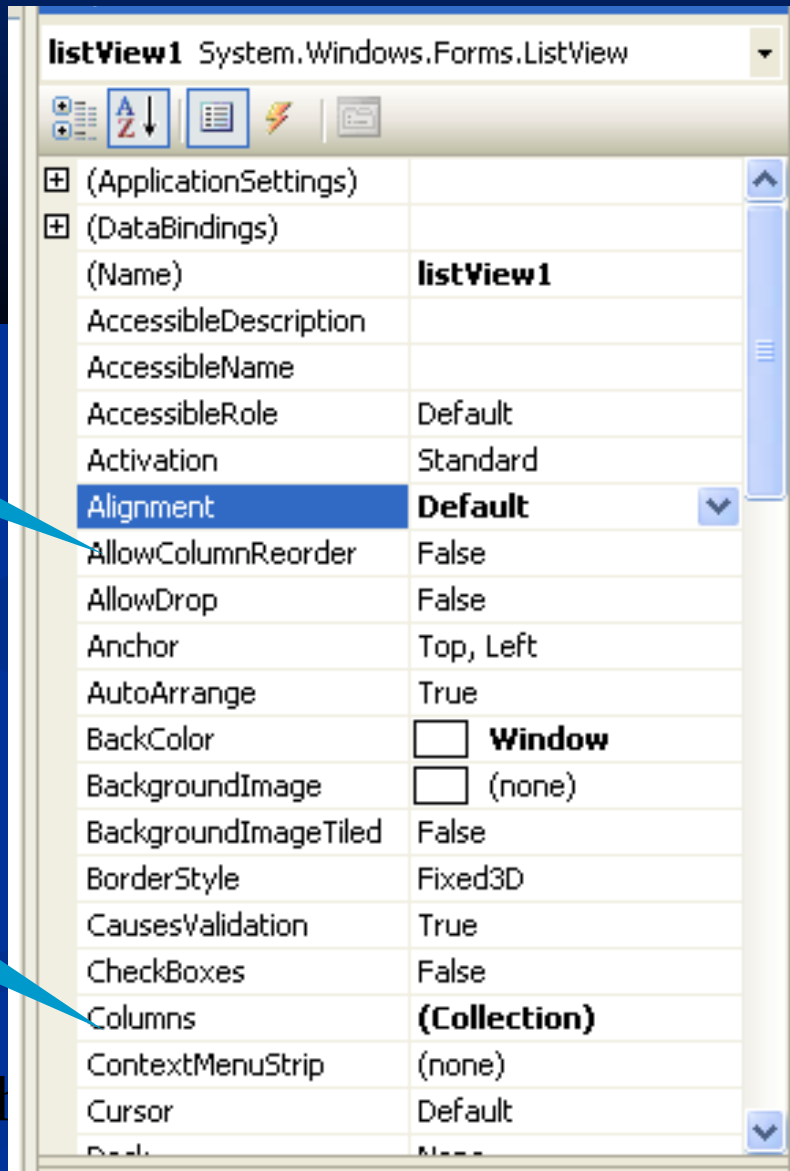


# 4.2. Điều khiển ListView

## Một số thuộc tính cơ bản

Cho phép sắp xếp cột trên điều khiển Listview ở chế độ thi hành

Khai báo số cột (có Header) của điều khiển Listview



# 4.2. Điều khiển ListView

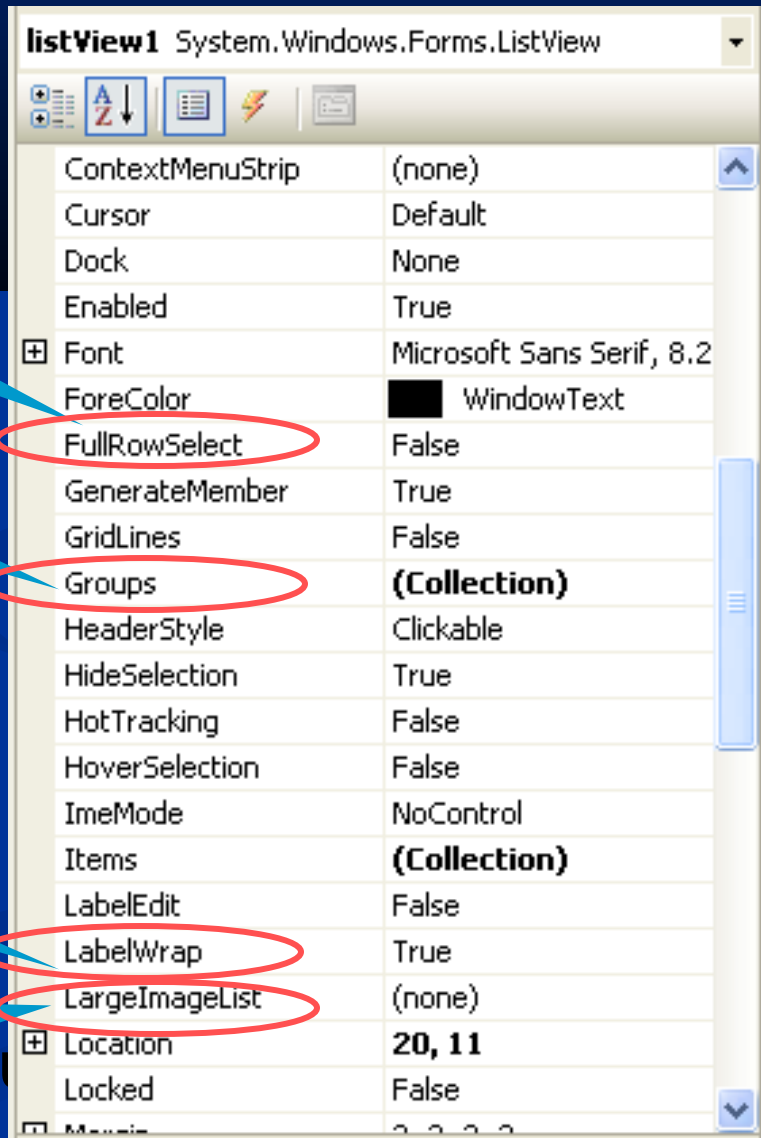
## Một số thuộc tính cơ bản

=True: Cho phép tô màu ứng với hàng của phần tử được chọn

Khai báo nhóm để phân loại các phần tử sau khi trình bày trên điều khiển Listview

=True: Chuỗi sẽ tự động xuống dòng khi chiều dài không đủ để trình bày

Đối tượng ImageList chứa danh sách các Image theo số chỉ mục từ 0 đến n-1 được sử dụng cho trường hợp thuộc tính View là LargeIcon

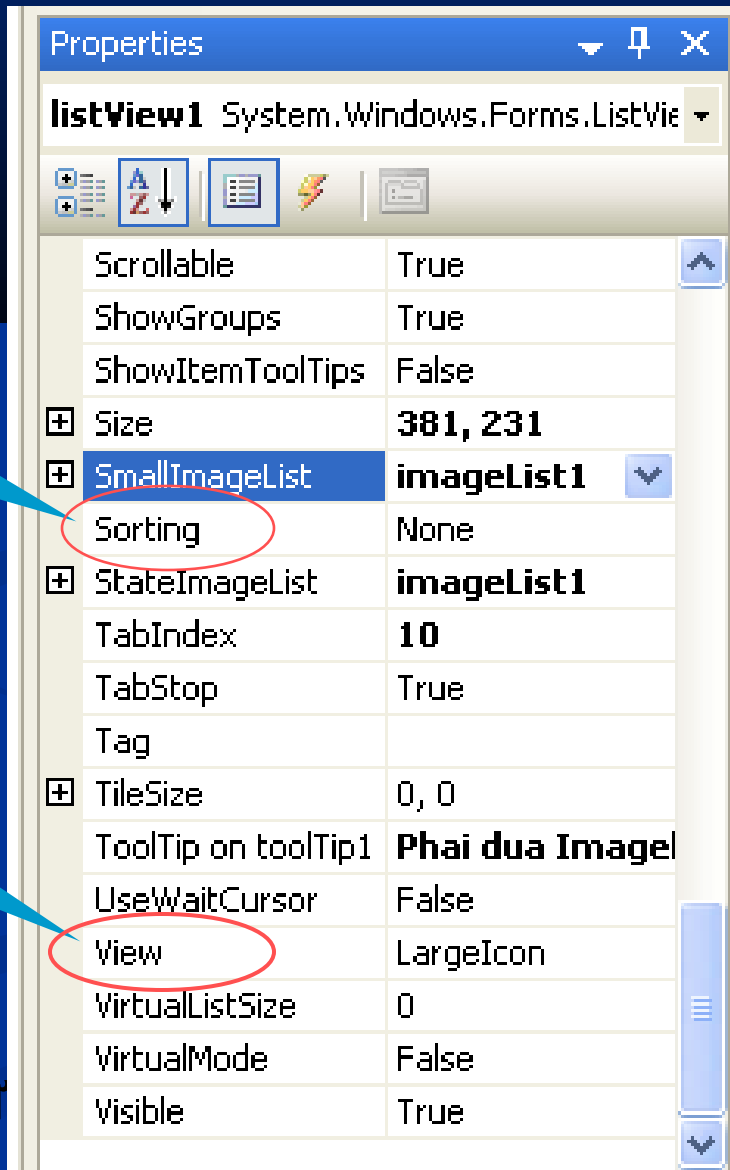


# 4.2. Điều khiển ListView

## Một số thuộc tính cơ bản

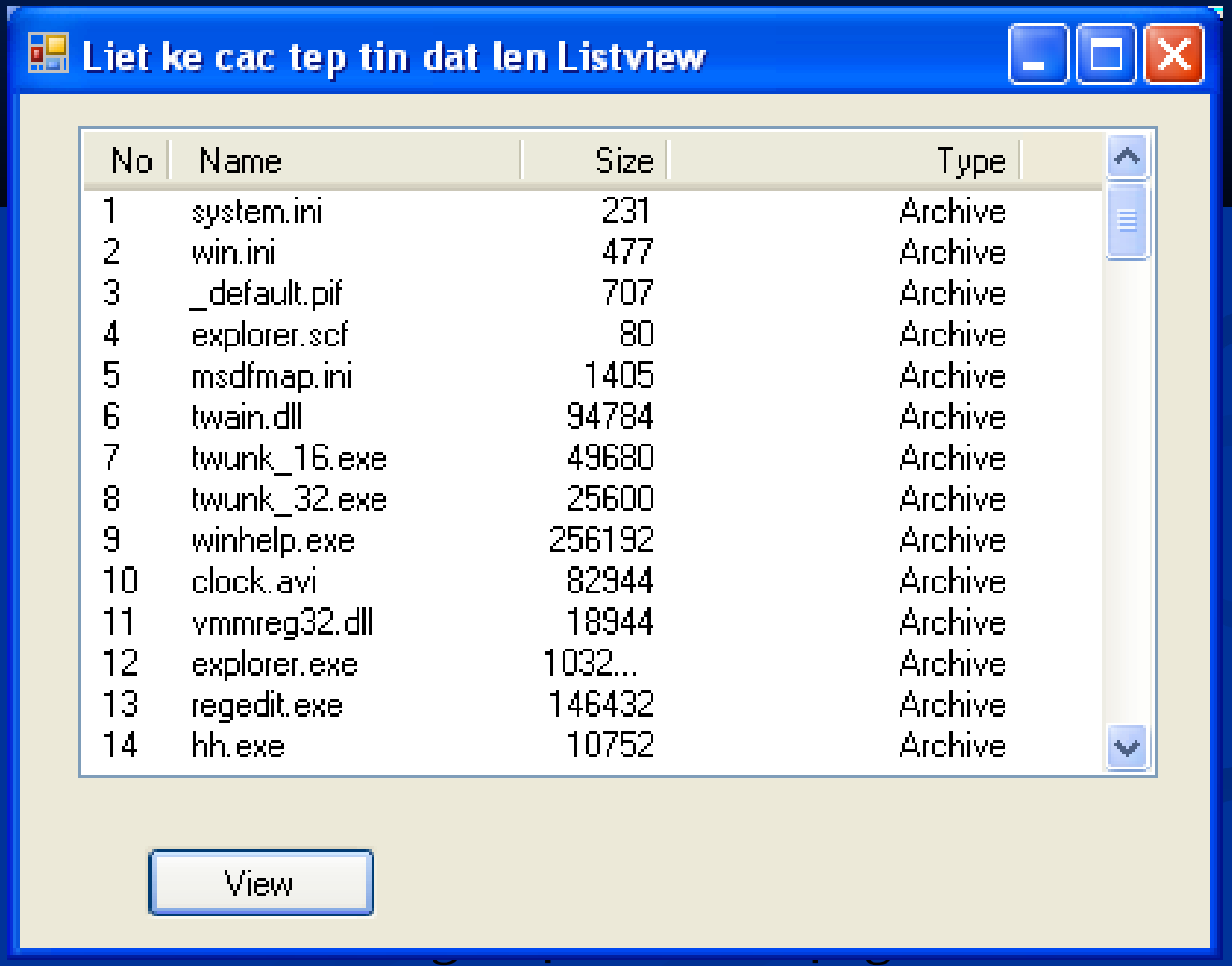
Các phần tử trên List view sẽ được sắp xếp tăng dần (Asccending), giảm dần (Descending) hoặc không sắp (None)

Chế độ trình bày tương ứng trên điều khiển như: List, Details, LargeIcon, SmallIcon, Title.



# 4.2. Điều khiển ListView

## Ví dụ: Liệt kê danh sách các tệp tin



# 4.2. Điều khiển ListView
## Ví dụ: Liệt kê danh sách các tệp tin

- Chú ý khi viết Code
  - Khai báo: using System.IO;
  - Khai báo sử dụng đối tượng DirectoryInfo để lấy thông tin của thư mục:

    DirectoryInfo dir = new DirectoryInfo("C:\\Windows\");
  - dir.GetFiles("*.*"): Lấy ra danh sách các File trong thư mục "dir"
  - FileInfo f: Khai báo đối tượng f chứa thông tin về các tệp tin

    - f.Name: Tên tệp tin
    - f.Length: Dung lượng tệp tin (byte)
    - f.Attributes: Thuộc tính của tệp tin
    - f.CreationTime: Ngày giờ tạp ra tệp tin

# 4.2. Điều khiển ListView

## Ví dụ: Liệt kê danh sách các tệp tin

- Chú ý khi viết Code
  - Khai báo cột trên Listview

```
this.listView1.Columns.Add("Name",200,
H
```

# 4.2. Điều khiển ListView
## Ví dụ: Liệt kê danh sách các tệp tin

- Chú ý khi viết Code
  - Chế độ hiển thị

    listView1.View = View.Details;
  - Thêm các tệp tin vào List view1

```
ListViewItem item1;  // Khai báo Item1 thuộc đối tượng ListViewItem
    foreach (FileInfo f in dir.GetFiles("*.*"))  // Lấy thông tin của tệp tin
        {                                                    // đưa vào Listview1
            i++;
            item1 = new ListViewItem(i.ToString());
            item1.SubItems.Add(f.Name);
            item1.SubItems.Add(f.Length.ToString());
            item1.SubItems.Add( f.Attributes.ToString());
            listView1.Items.Add(item1);
        }
```

# 4.2. Điều khiển ListView

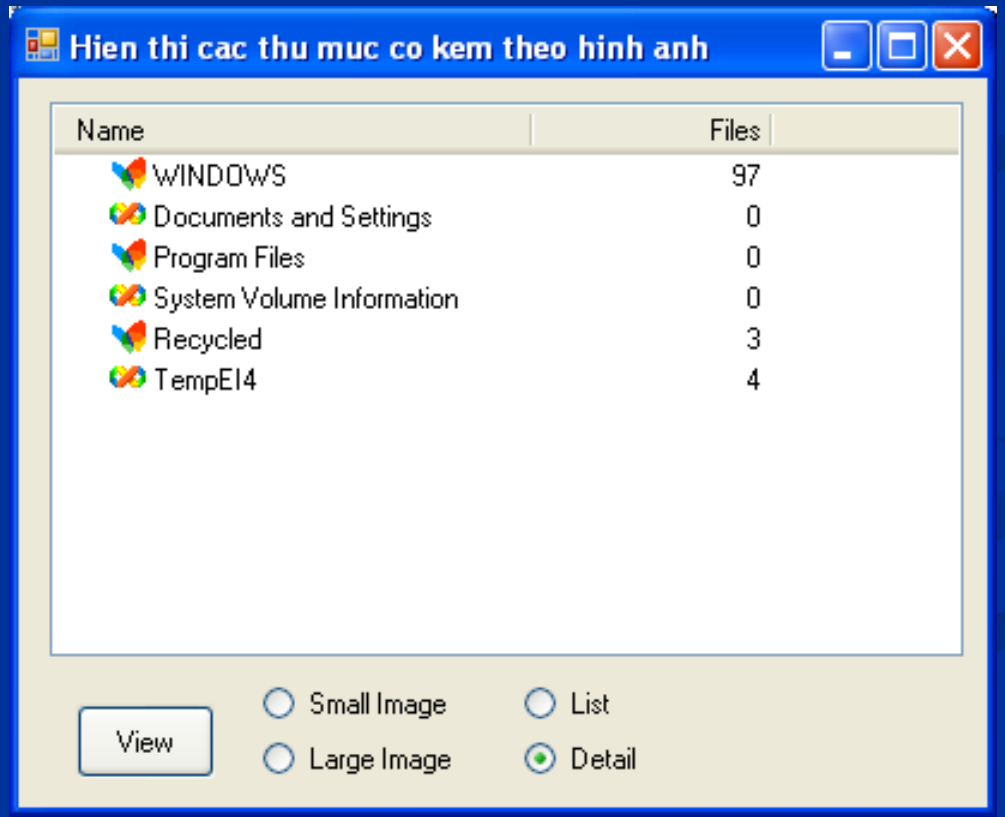
## Bài tập

SV tạo Listview để chứa danh sách các tệp tin lấy từ ổ đĩa D, tương tự như ví dụ trên

# 4.2. Điều khiển ListView

## Ví dụ 2

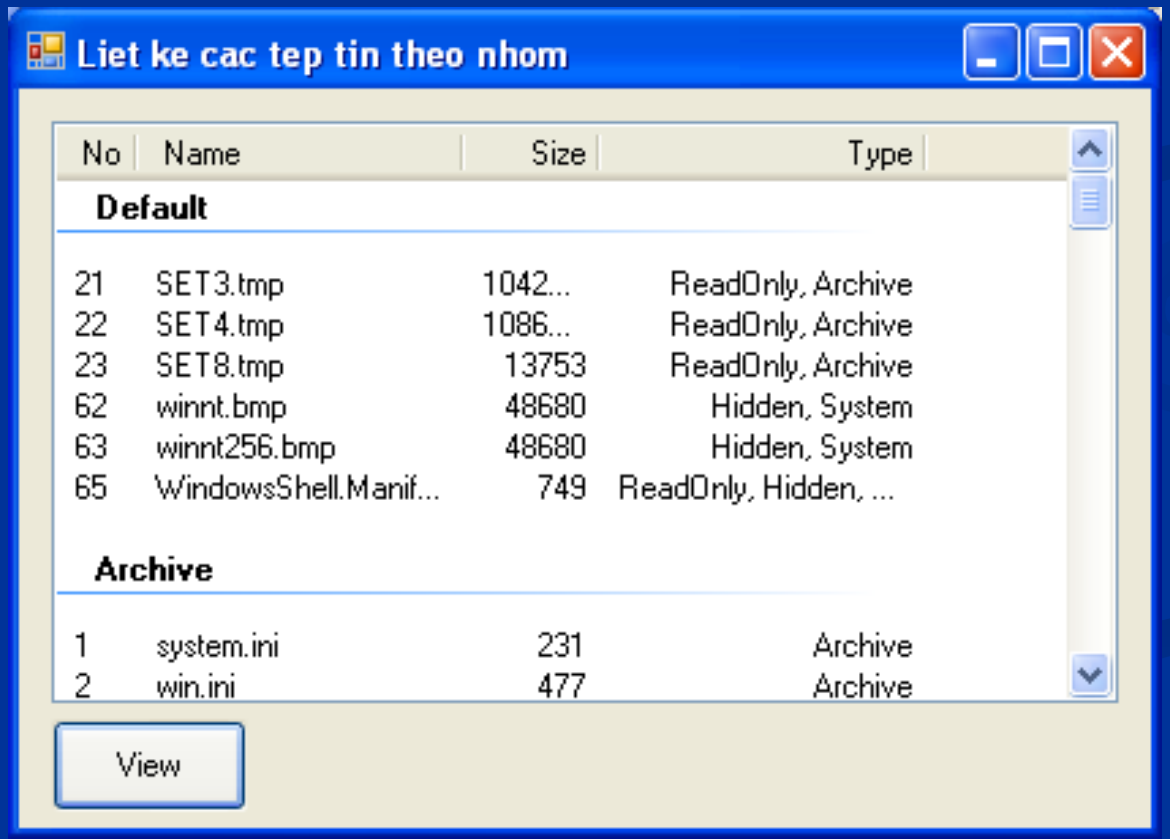
Tạo List view liệt kê các thư mục con, có chứa hình ảnh như sau:



# 4.2. Điều khiển ListView

## Ví dụ 3

Tạo List view liệt kê các thư mục con theo 4 nhóm (Archieve, System, Normal, Default) như sau:



# 4.3. Điều khiển TreeView

- Trình bày danh sách phần tử phân cấp theo từng nút (Giống Windows Explorer của Windows)

# 4.3. Điều khiển TreeView

## Một số thuộc tính

- CheckBoxes: Xuất hiện Checkbox bên cạnh từng nút của Treeview
- Nodes: Khai báo số Node (có header) của Listview
- FullRowSelect: là true – cho phép tô màu ứng với hàng của phần tử được chọn, giá trị mặc định là False
- ShowLine: Cho phép có đường viền ứng với từng nút, mặc định là True
- LabelEdit: là true nếu cho phép thay đổi chuỗi của mỗi nút

# 4.3. Điều khiển TreeView

## Một số thuộc tính

- ShowPlusMinus: là true thì có biểu tượng dấu + và - xuất hiện trên mỗi nút
- ShowRootLine: Chọn giá trị true nếu cho trình bày nút gốc
- ImageList: Chỉ ra đối tượng ImageList được đưa vào làm ảnh trên các nút của Treeview theo thứ tự chỉ mục từ 0 đến n-1 (giả sử ImageList có n ảnh)

# 4.3. Điều khiển TreeView

## Một số Phương thức

- CollapseAll: Trình bày tất cả các nút trên Treeview
- ExpandAll: Thu gọn tất cả các nút trên Treeview

# 4.3. Điều khiển TreeView

## Thêm nút vào Treeview

this.Treeview1.Nodes.Add(……..)

7 hàm nạp chồng

```
int TreeNodeCollection.Add(TreeNode node)  (+ 6 overload(s))
Adds a previously created tree node to the end of the tree node collection.

Exceptions:
   System.ArgumentException
```

# 4.3. Điều khiển TreeView

1 of 7 TreeNode TreeNodeCollection.Add (**string text**)
**text:** The label text displayed by the System.Windows.Forms.TreeNode.

this.Treeview1.Nodes.Add("My Computer")

2 of 7 int TreeNodeCollection.Add (**TreeNode node**)
**node:** The System.Windows.Forms.TreeNode to add to the collection.

this.Treeview1.Nodes[level1].Nodes.Add("Computer")

3 of 7 TreeNode TreeNodeCollection.Add (**string key**, string text)
**key:** The name of the tree node.

this.Treeview1.Nodes.Add("Root","My Computer")

# 4.3. Điều khiển TreeView

4 of 7 TreeNode TreeNodeCollection.Add (**string key**, string text, int imageIndex)
**key:** The name of the tree node.

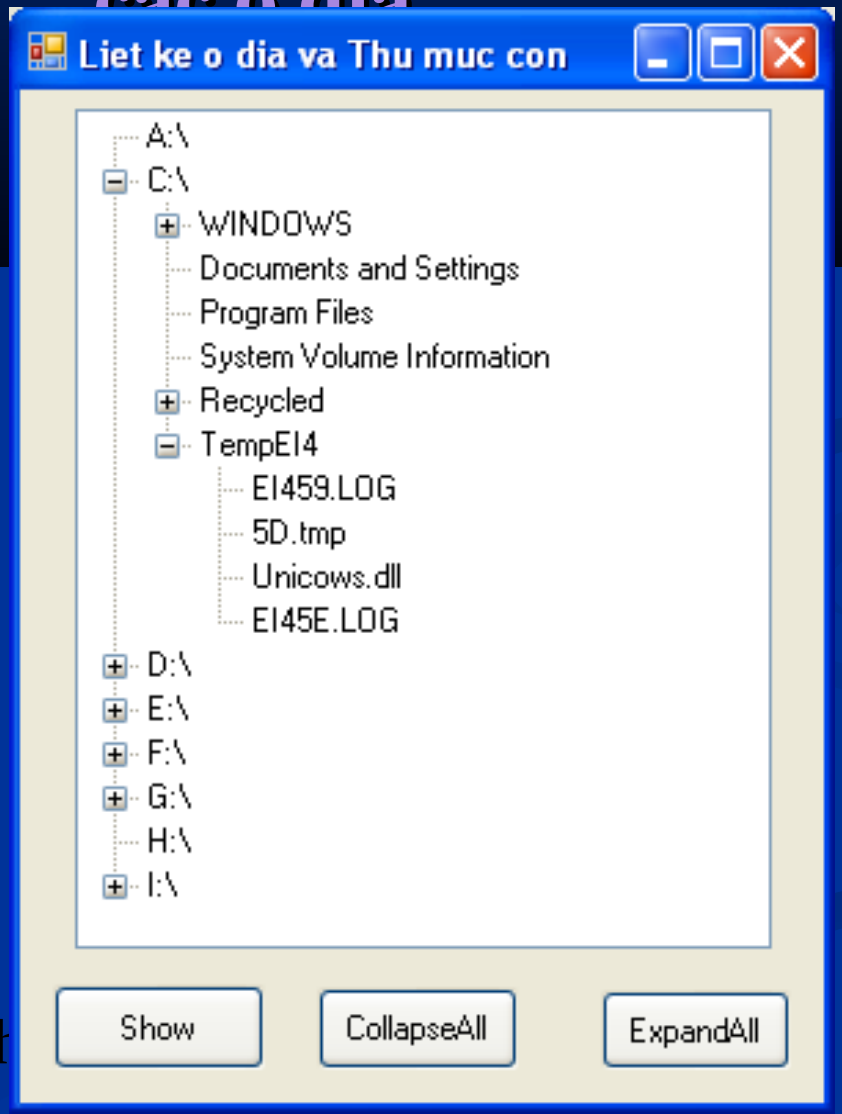
this.Treeview1.Nodes.Add("Root","My Computer",1)

5 of 7 TreeNode TreeNodeCollection.Add (**string key**, string text, string imageKey)
**key:** The name of the tree node.

this.Treeview1.Nodes.Add("Root","My Computer",
"C:\\Picture\\computer1.ico")

# 4.3. Điều khiển TreeView

## Ví dụ: Liệt kê các ổ đĩa và các thư mục con trên các ổ đĩa




Pł

# 4.3. Điều khiển TreeView

## Ví dụ: Liệt kê các ổ đĩa và các thư mục con trên các ổ đĩa

- Chú ý khi viết Code
  - Khai báo: using System.IO;
  - Khai báo sử dụng đối tượng Directory
    - Directory.GetLogicalDrives(): Lấy ds cách ổ đĩa logic
    - Directory.GetDirectories(F): Lấy danh sách các thư mục con của thư mục F
    - Directory.GetFile(F): Lấy danh sách các

# 4.3. Điều khiển TreeView

## Ví dụ: Liệt kê các ổ đĩa và các thư mục con trên các ổ đĩa

- Chú ý khi viết Code
  - Thêm nút vào TreeView như sau:

    this.Treeview1.Nodes.Add(TreeNode node)

VD:

this.Treeview1.Nodes.Add(“Root,”My Computer”,1)

# 4.3. Điều khiển TreeView

## Ví dụ: Liệt kê các ổ đĩa và các thư mục con trên các ổ đĩa

- Liệt kê các ổ Logic đặt lên Treevie
- Nút Show gọi hàm GetDisk()

```
void GetDisk()
    {
        foreach (string d in Directory.GetLogicalDrives())
        {
            this.treeView1.Nodes.Add(d);
        }
```

# 4.3. Điều khiển TreeView

- Liệt kê các Thư mục đặt lên Treeview

```
void GetFolder(string name, int level)
    {
        try
        {
          foreach (string d in Directory.GetDirectories(name))
           {
              this.treeView1.Nodes[level].Nodes.Add(d.Substring(3));
          }   //Cắt đi 3 ký tự đầu tiên VD: C:\TP\Bin thì còn TP\Bin
        }
        catch (Exception ex)
        {
           MessageBox.Show(ex.Message, "Error",
MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Warning);
        }
    }
```

# 4.3. Điều khiển TreeView

- Để liệt kê tất cả các thư mục trên các ổ đĩa, ta sửa lại hàm GetDisk như sau:

```
void GetDisk()
      {
       int i = 0;
          foreach (string d in Directory.GetLogicalDrives())
          {
             this.treeView1.Nodes.Add(d);
           GetFolder(d, i);
            i++;
          }
      }
```

# 4.3. Điều khiển TreeView

- Liệt kê các File có trong 1 thư mục đặt lên Treeview

```
void GetFile(string name, int level, int level1)
    {
        try
        {
            foreach (string d in Directory.GetFiles(name))
            {
                    this.treeView1.Nodes[level].Nodes[level1].
                    Nodes.Add(d.Substring(name.Length + 1));
            }
        }
        catch (Exception ex)

{           MessageBox.Show(ex.Message,"Error",MessageBoxButtons.AbortRetry
Ignore,
MessageBoxIcon.Warning);
        }
    }
```

# 4.3. Điều khiển TreeView

- Để Liệt kê các File, các thư mục con của các ổ Logic đặt lên Treeview ta viết lại GetFolder như sau:

```
void GetFolder(string name, int level)
    {
        try
        {   int level1 = 0;
            foreach (string d in Directory.GetDirectories(name))
            {
                this.treeView1.Nodes[level].Nodes.Add(d.Substring(3));
                GetFile(d, level, level1);                level1++;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Error",
MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Warning);
        }
    }
```

# 4.3. Điều khiển TreeView

- Viết Code cho Nút CollapseAl và  ExpandAll

```
private void button2_Click(object sender, EventArgs e)
    {
        treeView1.CollapseAll();
    }

  private void button3_Click(object sender, EventArgs e)
    {
        treeView1.ExpandAll();
    }
```

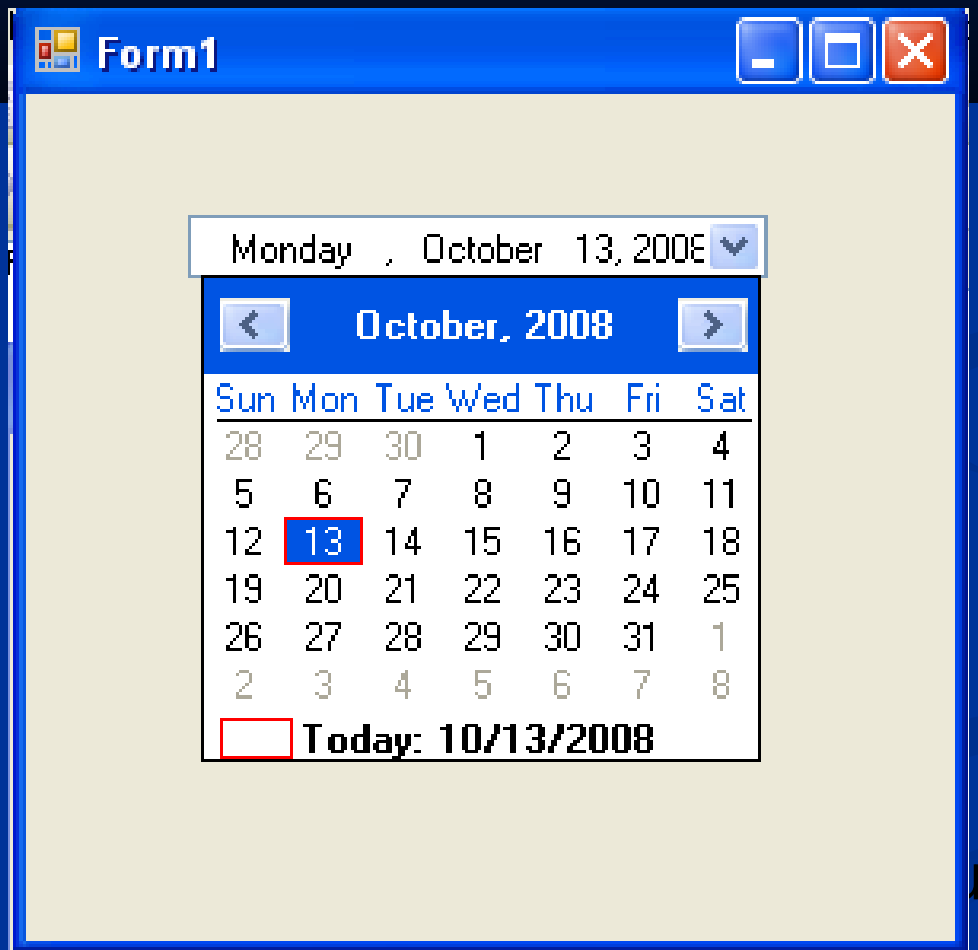
# 4.3. Điều khiển TreeView

- Bài tập

SV làm lại ví dụ trên

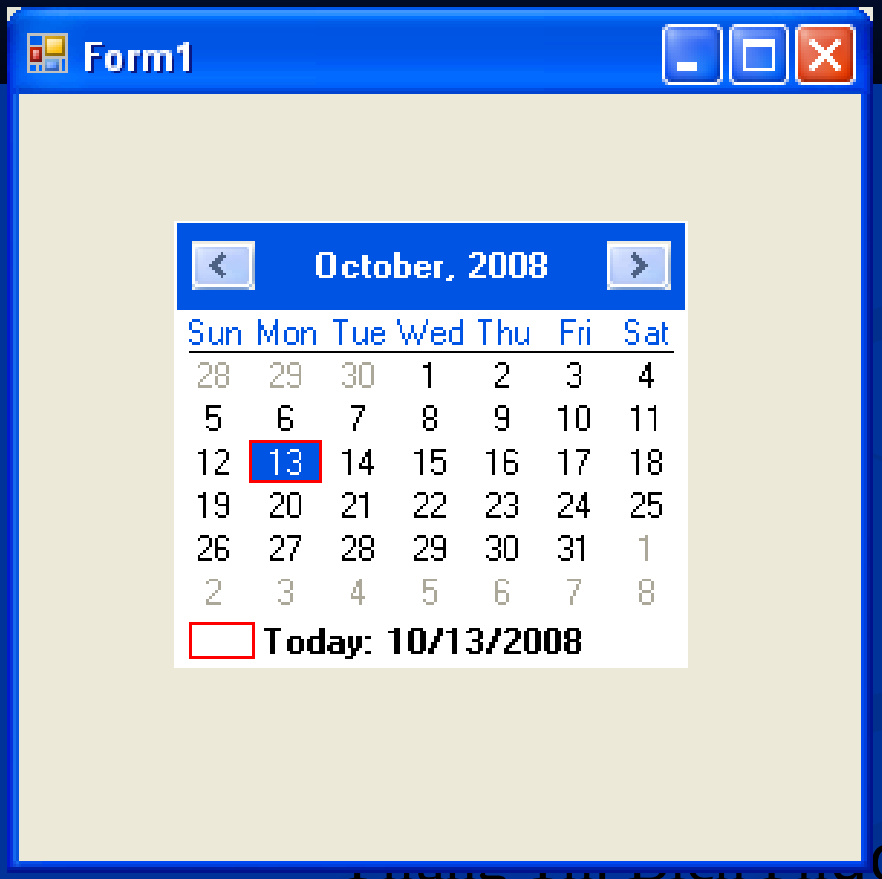
# 4.4. Điều khiển DateTimePicker

- Giống VB 6.0



# 4.5. Điều khiển MonthCalendar

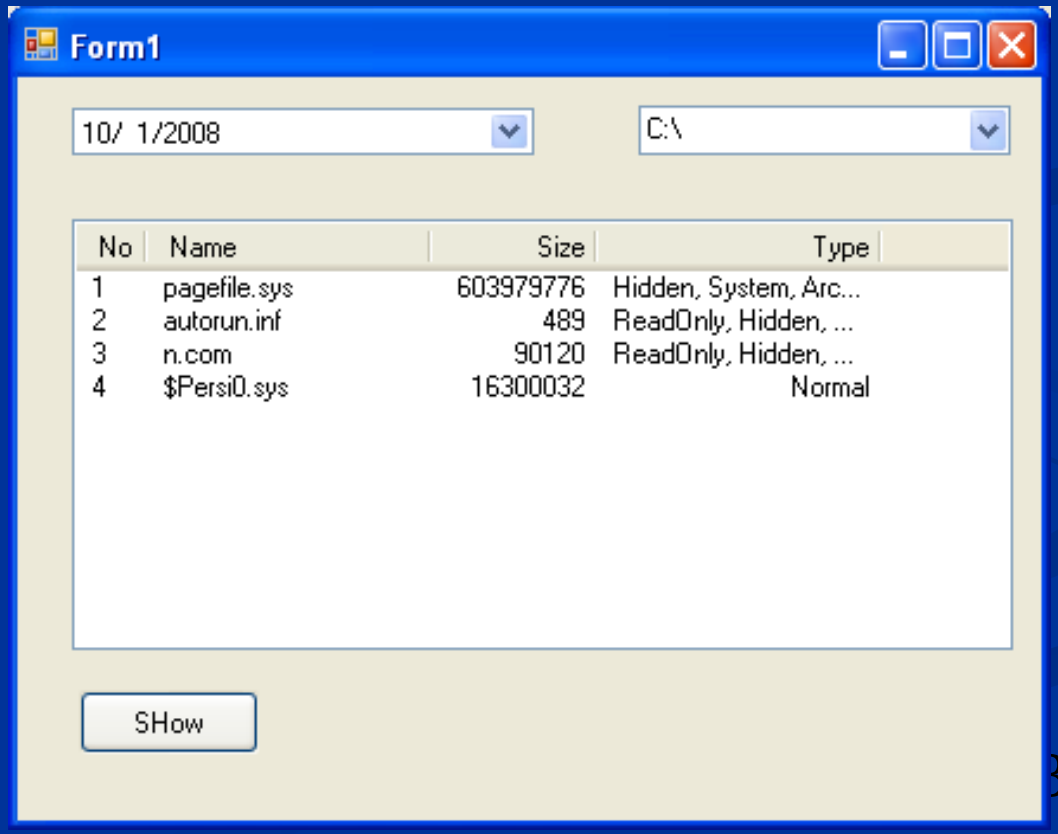
- Giống VB 6.0



# 4.5. Điều khiển MonthCalendar

## Bài tập

- Liệt kê các tệp tin được tạo ra trước ngày chỉ ra trong Combobox1 trong ổ đĩa (Chỉ ra trong ComboBox2)



# CHÚC BẠN THÀNH CÔNG