

**ỦY BAN NHÂN DÂN TP. HỒ CHÍ MINH
TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC
KHOA CÔNG NGHỆ THÔNG TIN**

**GIÁO TRÌNH
HỌC PHẦN: LẬP TRÌNH ỨNG DỤNG
NGÀNH/NGHỀ: CÔNG NGHỆ THÔNG TIN
TRÌNH ĐỘ: CAO ĐẲNG**

Ban hành kèm theo Quyết định số:... .../QĐ-CNTĐ-CN
Ngày....tháng....năm...của.....

Năm 2019

Lời Giới Thiệu

Giáo trình Lập trình ứng dụng nhằm trang bị cho sinh viên kiến thức cơ bản về ngôn ngữ lập trình C#, từ duy về lập trình và những kỹ thuật cần thiết trong lập trình để sinh viên có thể xây dựng được ứng dụng phần mềm bằng C#.

Qua các hoạt động học tập sinh viên có thể hoàn thiện dần tư duy lập trình và thói quen tuân thủ các qui định trong môi trường lập trình chuyên nghiệp.

Giáo trình gồm có các chương chính sau:

Chương 1: Tổng quan về lập trình ứng dụng

Chương 2: Nền tảng C# cơ bản

Chương 3: Lập trình hướng đối tượng trong C#

Chương 4: Lập trình ứng dụng Windows Forms

Chương 5: Sử dụng ADO.NET

Chương 6: Mô hình 3 lớp

GIÁO TRÌNH HỌC PHẦN

Tên học phần: Lập trình ứng dụng

Mã học phần:

Vị trí, tính chất, ý nghĩa và vai trò của học phần

- Vị trí: Đây là học phần bắt buộc, hệ cao đẳng ngành Công nghệ thông tin
- Tính chất: Học phần này cung cấp cho sinh viên về ngôn ngữ lập trình C#, tư duy lập trình và những kỹ thuật cần thiết trong lập trình để sinh viên có thể xây dựng được ứng dụng phần mềm bằng C#.
- Ý nghĩa và vai trò của học phần: Thông qua các hoạt động học tập sinh viên có thể hoàn thiện dần tư duy lập trình và thói quen tuân thủ các qui định trong môi trường lập trình chuyên nghiệp.

Mục tiêu của học phần:

- Kiến thức:
 - Tổng hợp các kiến thức cơ bản về ngôn ngữ lập trình C#
 - Trình bày được cấu trúc chương trình trong ngôn ngữ lập trình C#
- Kỹ năng:
 - Thành thạo công cụ IDE để phát triển ứng dụng
 - Tạo được ứng dụng console và sử dụng giao diện đồ họa để thiết kế ứng dụng
 - Phân tích thiết kế và xây dựng được ứng dụng tương tác với file hoặc với cơ sở dữ liệu.
- Năng lực tự chủ và trách nhiệm:
 - Rèn luyện thói quen tư duy hệ thống, học tập tích cực, chủ động
 - Tuân thủ các yêu cầu về chuẩn viết code

MỤC LỤC

| | |
|---|----------|
| Chương 1. Tổng quan về lập trình ứng dụng | 1 |
| 1.1 Tổng quan về lập trình ứng dụng | 1 |
| 1.2 Tổng quan về .NET Framework | 1 |
| 1.3 Kiến trúc .NET Framework | 1 |
| 1.3.1 Common Language Runtime (CLR)..... | 2 |
| 1.3.2 Thư viện lớp .NET Framework..... | 3 |
| 1.4 Biên dịch và MSIL | 4 |
| 1.5 Giới thiệu ứng dụng loại Windows Application | 5 |
| Chương 2. NỀN TẢNG C# CƠ BẢN..... | 1 |
| 2.1 Kiểu dữ liệu | 2 |
| 2.1.1 Các kiểu dữ liệu cơ bản trong c#..... | 2 |
| 2.1.2 Khai báo biến và khởi tạo giá trị | 3 |
| 2.2 Toán tử và biểu thức..... | 3 |
| 2.3 Câu lệnh điều khiển..... | 6 |
| 2.3.1 Cấu trúc rẽ nhánh | 6 |
| 2.3.2 Cấu trúc switch..... | 8 |
| 2.3.3 Cấu trúc lặp | 9 |
| 2.4 Kiểu mảng | 14 |
| 2.4.1 Khái niệm mảng một chiều: | 14 |
| 2.4.2 Đặc điểm của mảng:..... | 14 |
| 2.4.3 Cú pháp chung khai báo mảng:..... | 15 |
| 2.4.4 Khởi tạo các giá trị cho mảng | 15 |
| 2.4.5 Sử dụng mảng | 15 |
| 2.4.6 Một số phương thức đặt trưng cho mảng một chiều | 15 |
| 2.4.7 Cách duyệt mảng một chiều..... | 15 |
| 2.4.8 Ví dụ mảng một chiều | 16 |
| 2.5 Collections..... | 17 |

| | |
|---|-----------|
| 2.5.1 Một số Collections thông dụng | 18 |
| 2.5.2 Sử dụng các phương thức và thuộc tính có trong lớp ArrayList..... | 18 |
| 2.5.3 Một số ví dụ | 19 |
| 2.6 Xử lý chuỗi..... | 23 |
| 2.7 Xử lý ngoại lệ..... | 26 |
| 2.7.1 Xử lý ngoại lệ (Exception)..... | 26 |
| 2.7.2 Cú pháp chung khi xử lý ngoại lệ | 26 |
| 2.7.3 Lớp Exception trong C#..... | 27 |
| 2.7.4 Một số lớp Exception kế thừa từ lớp SystemException..... | 27 |
| 2.8 Đọc ghi tập tin | 30 |
| 2.8.1 Đọc tập tin | 30 |
| 2.8.2 Ghi tập tin..... | 31 |
| 2.9 Bài tập | 35 |
| Chương 3. Lập trình hướng đối tượng trong C# | 37 |
| 3.1 Tổng quan lập trình hướng đối tượng..... | 38 |
| 3.2 Đối tượng..... | 38 |
| 3.3 Lớp đối tượng | 38 |
| 3.4 Các đặc trưng cơ bản | 38 |
| 3.5 Định nghĩa lớp và đối tượng..... | 39 |
| 3.6 Xây dựng lớp trong C# | 40 |
| 3.7 Bài tập áp dụng | 41 |
| 3.8 Bài tập | 51 |
| Chương 4. Lập trình windows forms | 54 |
| 4.1 Giới thiệu thành phần Windows FoRmS | 54 |
| 4.2 Các thành phần của Windows Forms..... | 54 |
| 4.3 Những đặc điểm cơ bản của Windows Forms | 55 |
| 4.3.1 Một số phương thức của Form..... | 58 |
| 4.3.2 Một số sự kiện của Form (Form Event)..... | 59 |
| 4.4 Các điều khiển chuẩn | 60 |
| Button Control..... | 60 |
| Label Control | 61 |

| | |
|--|------------|
| TextBox Control..... | 62 |
| CheckBox Control..... | 62 |
| RadioButton Control | 64 |
| ListBox Control..... | 65 |
| CheckListBox..... | 69 |
| ComboBox Control | 70 |
| GroupBox, Panel | 71 |
| PictureBox..... | 72 |
| ImageList..... | 72 |
| 4.5 Điều khiển đặc biệt..... | 73 |
| DateTimePicker và MonthCalender..... | 73 |
| ListView | 74 |
| TreeView..... | 79 |
| 4.6 Điều khiển menu | 81 |
| 4.6.1 Một số thuộc tính của Menu | 81 |
| 4.6.2 Tạo ContextMenu | 82 |
| 4.7 Sử dụng hộp thoại | 84 |
| 4.7.1 Hộp thoại màu ColorDialog | 84 |
| 4.7.2 Hộp thoại Font | 85 |
| 4.7.3 Hộp thoại OpenFileDialog và SaveDialog..... | 86 |
| 4.7.4 Hộp thoại MessageBox | 91 |
| 4.8 Unit test và Function test..... | 92 |
| 4.8.1 Kiểm tra đơn vị (Unit test): | 92 |
| 4.8.2 Kiểm tra chức năng (Function test): | 93 |
| 4.9 Bài tập | 94 |
| Chương 5. Sử dụng ADO.NET..... | 105 |
| 5.1 Giới thiệu..... | 106 |
| 5.2 Kiến trúc tổng quan của ADO.NET | 107 |
| 5.3 Tổng quan về các mô hình xử lý dữ liệu trong ADO.NET | 109 |
| 5.3.1 Mô hình Kết nối | 109 |
| 5.3.2 Mô hình Ngắt Kết nối | 110 |

| | | |
|------------------|---|------------|
| 5.4 | Làm việc với các lớp trong ADO.NET | 112 |
| 5.4.1 | Lớp Connection..... | 112 |
| 5.4.2 | Đối tượng Command..... | 114 |
| 5.4.3 | Đối tượng DataAdapter..... | 115 |
| 5.5 | Thực thi StoredProcedure (thủ tục lưu trữ sẵn) với đối tượng Command | 118 |
| 5.6 | Bài tập | 125 |
| Chương 6. | Mô hình 3 lớp..... | 128 |
| 6.1 | Giới thiệu tổng quan về mô hình 3 lớp..... | 128 |
| 6.2 | Xây dựng ứng dụng theo mô hình 3 lớp..... | 129 |
| 6.3 | Bài tập | 149 |

CHƯƠNG 1. TỔNG QUAN VỀ LẬP TRÌNH ỨNG DỤNG

1.1 Tổng quan về lập trình ứng dụng

Ngôn ngữ lập trình C# là ngôn ngữ lập trình phổ biến thứ 4 sau Java, PHP và Python với khoảng 31% các nhà phát triển sử dụng nó thường xuyên. Đây cũng là cộng đồng lớn thứ 3 trên StackOverflow với hơn 1,1 triệu chủ đề. Sự phổ biến này giúp thị trường nhân sự C# phát triển với hơn 17.000 công việc C# vào năm 2017 (Theo thống kê của vietnamworks). Nó đã tạo ra những cơ hội rất lớn cho những nhà lập trình viên.

Ngôn ngữ C# được sử dụng để xây dựng các ứng dụng từ đơn giản đến phức tạp với bộ phát triển của C# làm việc chủ yếu trên bộ khung .NET Framework. Đây là ngôn ngữ lập trình có khả năng tạo ra nhiều ứng dụng mạnh mẽ và an toàn cho nền tảng Windows với thành phần máy chủ, dịch vụ web, ứng dụng di động và nhiều khả năng khác nữa.

1.2 Tổng quan về .NET Framework

Microsoft .NET được công bố lần đầu tiên vào 7/2000 tại hội nghị Professional Developers' Conference ở Orlando. Nó gồm hai phần chính là Framework và Integrated Development Environment (IDE).

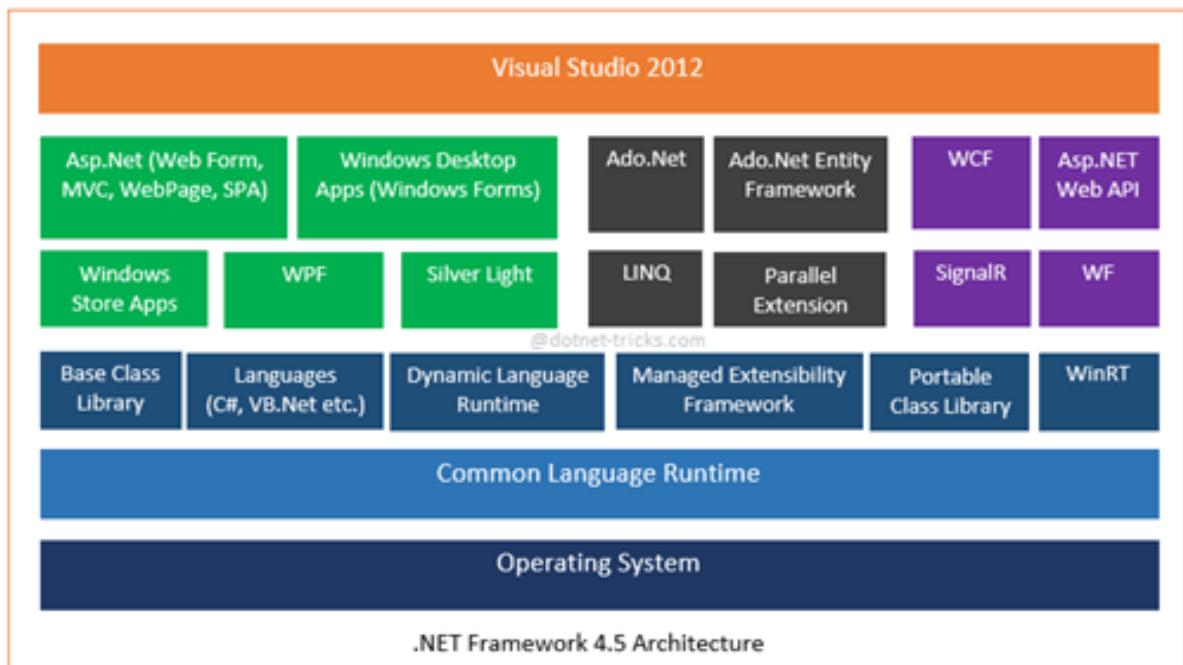
Framework là cốt lõi và là tinh hoa của .NET. Đồng thời, nó cũng là nền tảng mà các ứng dụng .NET chạy trên đó.

IDE cung cấp một môi trường giúp triển khai và nhanh chóng các ứng dụng trên .NET.

1.3 Kiến trúc .NET Framework

.NET Framework có hai thành phần chính: Common Language Runtime (CLR) và thư viện lớp .NET Framework.

- CLR là nền tảng của .NET Framework.
- Thư viện lớp, một thành phần chính khác của .NET Framework là một tập hợp hướng đối tượng của các kiểu dữ liệu được dùng lại, nó cho phép chúng ta có thể phát triển những ứng dụng từ những ứng dụng truyền thống command-line hay những ứng dụng có giao diện đồ họa (GUI) đến những ứng dụng mới nhất được cung cấp bởi ASP.NET, như là Web Form và dịch vụ XML Web.



Hình 1.1 Các thành phần trong .NET Framework

1.3.1 Common Language Runtime (CLR)

CLR là thành phần kết nối giữa các phần khác trong .NET Framework với hệ điều hành. CLR là chương trình viết bằng .NET, không được biên dịch ra mã máy mà nó được dịch ra một ngôn ngữ trung gian Microsoft Intermediate Language (MSIL). Khi chạy chương trình, CLR sẽ dịch MSIL ra mã máy để thực thi các tính năng, đảm bảo ứng dụng không chiếm dụng và sử dụng tràn lan tài nguyên của hệ thống. Sau đây là một số đặc điểm quan trọng của CLR:

- Là thành phần quan trọng nhất của .NET Framework
- Quản lý và thực thi các managed code và là nền tảng của kiến trúc .NET.
- Quản lý bộ nhớ, quản lý tiến trình (thread), kiểm tra security....
- CLR được thiết kế để làm tăng hiệu xuất thực hiện.
- Gần giống như Java Virtual Machine (JVM)

CLR thúc đẩy cho mã nguồn được thực thi mạnh mẽ hơn bằng việc thực thi mã nguồn chính xác và sự xác nhận mã nguồn. Nền tảng của việc thực hiện này là Common Type System (CTS). CTS đảm bảo rằng những mã nguồn được quản lý thì được tự mô tả (self- describing). Sự khác nhau giữa Microsoft và các trình biên

dịch ngôn ngữ của hằng thứ ba là việc tạo ra các mã nguồn được quản lý có thể thích hợp với CTS. Điều này thì mã nguồn được quản lý có thể sử dụng những kiểu được quản lý khác và những thể hiện, trong khi thúc đẩy nghiêm ngặt việc sử dụng kiểu dữ liệu chính xác và an toàn.

Thêm vào đó, môi trường được quản lý của runtime sẽ thực hiện việc tự động xử lý layout của đối tượng và quản lý những tham chiếu đến đối tượng, giải phóng chúng khi chúng không còn được sử dụng nữa. Việc quản lý bộ nhớ tự động này còn giải quyết hai lỗi chung của ứng dụng: thiếu bộ nhớ và tham chiếu bộ nhớ không hợp lệ.

Runtime được thiết kế để cải tiến hiệu suất thực hiện. Mặc dù CLR cung cấp nhiều các tiêu chuẩn dịch vụ runtime, nhưng mã nguồn được quản lý không bao giờ được dịch. Có một đặc tính gọi là Just-in-Time (JIT) biên dịch tất cả những mã nguồn được quản lý vào trong ngôn ngữ máy của hệ thống vào lúc mà nó được thực thi. Khi đó, trình quản lý bộ nhớ xóa bỏ những phân mảnh bộ nhớ nếu có thể được và gia tăng tham chiếu bộ nhớ cục bộ, và kết quả gia tăng hiệu quả thực thi.

1.3.2 Thư viện lớp .NET Framework

Thư viện lớp .NET Framework là một tập hợp những kiểu dữ liệu được dùng lại và được kết hợp chặt chẽ với Common Language Runtime. Thư viện lớp là hướng đối tượng cung cấp những kiểu dữ liệu mà mã nguồn được quản lý của chúng ta có thể dẫn xuất. Điều này không chỉ làm cho những kiểu dữ liệu của .NET Framework dễ sử dụng mà còn làm giảm thời gian liên quan đến việc học đặc tính mới của .NET Framework.Thêm vào đó, các thành phần của các hằng thứ ba có thể tích hợp với những lớp trong .NET Framework.

Cũng như mong đợi của người phát triển với thư viện lớp hướng đối tượng, kiểu dữ liệu .NET Framework cho phép người phát triển thiết lập nhiều mức độ thông dụng của việc lập trình, bao gồm các nhiệm vụ như: quản lý chuỗi, thu thập hay chọn lọc dữ liệu, kết nối với cơ sở dữ liệu, và truy cập tập tin. Ngoài những nhiệm vụ thông dụng trên. Thư viện lớp còn đưa vào những kiểu dữ liệu để hỗ trợ cho những kịch bản phát triển chuyên biệt khác. Ví dụ người phát triển có thể sử dụng .NET Framework để phát triển những kiểu ứng dụng và dịch vụ như sau:

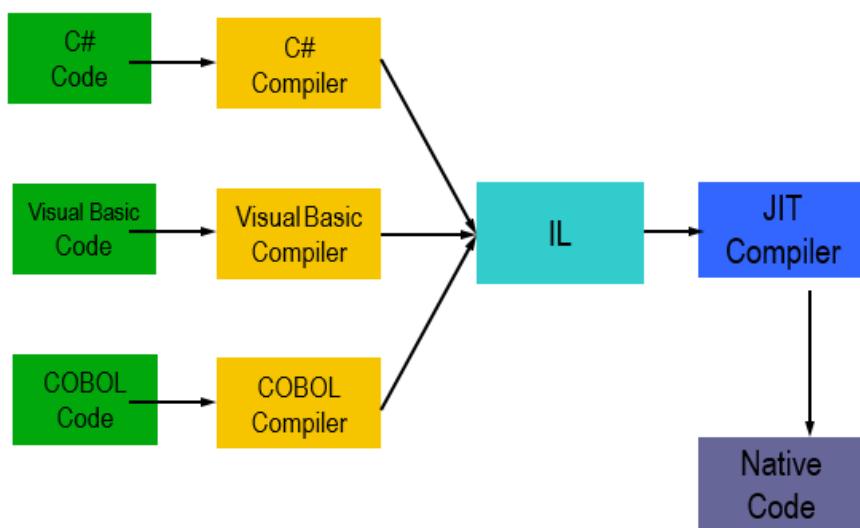
- Ứng dụng Console
- Ứng dụng giao diện GUI trên Windows (Windows Forms)

- Ứng dụng ASP.NET
- Dịch vụ XML Web
- Dịch vụ Windows

Trong đó những lớp Windows Forms cung cấp một tập hợp lớn các kiểu dữ liệu nhằm làm đơn giản việc phát triển các ứng dụng GUI chạy trên Windows. Nếu như viết các ứng dụng ASP.NET thì có thể sử dụng các lớp Web Forms trong thư viện .NET Framework.

1.4 Biên dịch và MSIL

Trong .NET Framework, chương trình không được biên dịch vào các tập tin thực thi mà thay vào đó chúng được biên dịch vào những tập tin trung gian gọi là Microsoft Intermediate Language (MSIL). Những tập tin MSIL được tạo ra từ C# cũng tương tự như các tập tin MSIL được tạo ra từ những ngôn ngữ khác của .NET, platform ở đây không cần biết ngôn ngữ của mã nguồn. Điều quan trọng chính yếu của CLR là chung (common), cùng một runtime hỗ trợ phát triển trong C# cũng như trong VB.NET.



Hình 1.3 Biên dịch trong .NetFramework

Mã nguồn C# được biên dịch vào MSIL khi chúng ta build project. Mã MSIL này được lưu vào trong một tập tin trên đĩa. Khi chúng ta chạy chương trình, thì MSIL được biên dịch một lần nữa, sử dụng trình biên dịch Just-In-Time (JIT). Kết quả là mã máy được thực thi bởi bộ xử lý của máy.

Trình biên dịch JIT tiêu chuẩn thì thực hiện theo yêu cầu. Khi một phương thức được gọi, trình biên dịch JIT phân tích MSIL và tạo ra sản phẩm mã máy có hiệu quả

cao, mã này có thể chạy rất nhanh. Trình biên dịch JIT đủ thông minh để nhận ra khi một mã đã được biên dịch, do vậy khi ứng dụng chạy thì việc biên dịch chỉ xảy ra khi cần thiết, tức là chỉ biên dịch mã MSIL chưa biên dịch ra mã máy. Khi đó một ứng dụng .NET thực hiện, chúng có xu hướng là chạy nhanh và nhanh hơn nữa, cũng như là những mã nguồn được biên dịch rồi thì được dùng lại.

Do tất cả các ngôn ngữ .NET Framework cùng tạo ra sản phẩm MSIL giống nhau, nên kết quả là một đối tượng được tạo ra từ ngôn ngữ này có thể được truy cập hay được dẫn xuất từ một đối tượng của ngôn ngữ khác trong .NET. Ví dụ, người phát triển có thể tạo một lớp cơ sở trong VB.NET và sau đó dẫn xuất nó trong C# một cách dễ dàng.

1.5 Giới thiệu ứng dụng loại Windows Application

Thông thường, trong việc phát triển phần mềm, người phát triển phải tuân thủ theo quy trình phát triển phần mềm một cách nghiêm ngặt và quy trình này đã được chuẩn hóa. Tuy nhiên trong phạm vi của chúng ta là tìm hiểu một ngôn ngữ mới và viết những chương trình nhỏ thì không đòi hỏi khắt khe việc thực hiện theo quy trình. Nhưng để giải quyết được những vấn đề thì chúng ta cũng cần phải thực hiện đúng theo các bước sau. Đầu tiên là phải xác định vấn đề cần giải quyết. Nếu không biết rõ vấn đề thì ta không thể tìm được phương pháp giải quyết. Sau khi xác định được vấn đề, thì chúng ta có thể nghĩ ra các kế hoạch để thực hiện. Sau khi có một kế hoạch, thì có thể thực thi kế hoạch này. Sau khi kế hoạch được thực thi, chúng ta phải kiểm tra lại kết quả để xem vấn đề được giải quyết xong chưa. Logic này thường được áp dụng trong nhiều lĩnh vực khác nhau, trong đó có lập trình.

Khi tạo một chương trình trong C# hay bất cứ ngôn ngữ nào, chúng ta nên theo những bước tuần tự sau:

- Xác định mục tiêu của chương trình.
- Xác định những phương pháp giải quyết vấn đề.
- Tạo một chương trình để giải quyết vấn đề.
- Thực thi chương trình để xem kết quả.

Ví dụ mục tiêu để viết chương trình xử lý văn bản đơn giản, mục tiêu chính là xây dựng chương trình cho phép soạn thảo và lưu trữ những chuỗi ký tự hay văn bản. Nếu không có mục tiêu thì không thể viết được chương trình hiệu quả.

Bước thứ hai là quyết định đến phương pháp để viết chương trình. Bước này xác định những thông tin nào cần thiết được sử dụng trong chương trình, các hình thức nào được sử dụng. Từ những thông tin này chúng ta rút ra được phương pháp để giải quyết vấn đề.

Bước thứ ba là bước cài đặt, ở bước này có thể dùng các ngôn ngữ khác nhau để cài đặt, tuy nhiên, ngôn ngữ phù hợp để giải quyết vấn đề một cách tốt nhất sẽ được chọn. Trong phạm vi của sách này chúng ta mặc định là dùng C#, đơn giản là chúng ta đang tìm hiểu nó! Và bước cuối cùng là phần thực thi chương trình để xem kết quả.

Chương trình C# đơn giản

Để bắt đầu cho việc tìm hiểu ngôn ngữ C# và tạo tiền đề cho các chương sau, chương đầu tiên trình bày một chương trình C# đơn giản nhất.

Ví dụ : Chương trình C# đầu tiên.

```
class ChaoMung
{
    static void Main( )
    {
        // Xuat ra man hinh
        System.Console.WriteLine("Chao Mung");
    }
}
```

Kết quả:

Chao Mung

Sau khi viết xong chúng ta lưu dưới dạng tập tin có phần mở rộng *.cs (C sharp). Sau đó biên dịch và chạy chương trình. Kết quả là một chuỗi “Chao Mung” sẽ xuất hiện trong màn hình console.

Xây dựng chương trình đơn giản với Windows Form sẽ được trình bày ở chương 4 của tài liệu này.

CHƯƠNG 2. NỀN TẢNG C# CƠ BẢN

Học xong chương này, người học có thể:

- + Trình bày kiến thức cơ bản về ngôn ngữ C#.
- + Sử dụng được các kiểu dữ liệu, cấu trúc của ngôn ngữ lập trình C# để giải quyết bài toán vừa và nhỏ.

2.1 Kiểu dữ liệu

Ngôn ngữ C# sử dụng các kiểu dữ liệu cơ bản như những ngôn ngữ lập trình khác. Ngoài ra trong C# có các kiểu dữ liệu giá trị tương ứng như int, double, ... và kiểu dữ liệu tham chiếu như String, Object,...

Các kiểu dữ liệu trong C# là tập hợp của các giá trị có các đặc điểm tương tự. ví dụ kiểu byte chỉ định số bộ số nguyên trong phạm vi [0..255].

Đặc điểm kiểu dữ liệu: Tên kiểu dữ liệu, kích thước bộ nhớ sử dụng và giá trị mặc định.

2.1.1 Các kiểu dữ liệu cơ bản trong c#

- Kiểu số nguyên: sbyte, byte, short, ushort, int , uint, long, ulong
- Kiểu số thực: float, double
- Kiểu bool: True hoặc False
- Kiểu ký tự: char
- Kiểu chuỗi: string
- Kiểu tham chiếu: object

Bảng các kiểu dữ liệu cơ bản:

| Kiểu dữ liệu | Mô tả | Kích thước (bit) | Vùng dữ liệu | Ví dụ |
|--------------|--------------|------------------|--|-----------------|
| int | Số nguyên | 32 | -2^{31} đến $2^{31} - 1$ | int nSize; |
| long | Số nguyên | 64 | -2^{63} đến $2^{63} - 1$ | long lSize; |
| float | Số thực | 32 | $\pm 1.5 \times 10^{-45}$ đến $\pm 3.4 \times 10^{38}$ | float fDelta; |
| double | Số thực | 64 | $\pm 5.0 \times 10^{-324}$ đến $\pm 1.7 \times 10^{308}$ | double dDelta; |
| decimal | Số thập phân | 128 | | decimal decKe; |
| string | Chuỗi | 16 | | string strName; |

| | | | | |
|------|------------|----|--------------------|--------------|
| char | Ký tự | 16 | 0 đến $2^{16} - 1$ | char chrAns; |
| bool | Đúng / Sai | 8 | True hoặc False | bool bRet; |

2.1.2 Khai báo biến và khởi tạo giá trị

<Kiểu dữ liệu> tenbien;

tenbien = giatri;

Khai báo một biến kiểu số nguyên x và khởi tạo giá trị ban đầu là 1.

int iX;

iX= 1;

hoặc khai báo và gán giá trị trên cùng một dòng:

int iX=1;

Khai báo và khởi tạo giá trị cho hằng số:

const <kieudulieu> tenHang = giaTri;

ví dụ: Khai báo và khởi tạo giá trị Max

const int iMax = 100;

2.2 Toán tử và biểu thức

Toán tử trong C# thực hiện các phép toán về số học, toán tử quan hệ, toán tử logic

Các phép toán tử số học được hỗ trợ trong C#, ví dụ ta có 2 biến A và B có giá trị là 10 và 20.

| Toán tử | Mô tả | Ví dụ | Kết quả |
|---------|--|-------|---------|
| + | Cộng hai toán hạng | A + B | 30 |
| - | Toán hạng trước trừ toán hạng sau | A - B | -10 |
| * | Nhân hai toán hạng | A * B | 200 |
| / | Chia tử số cho mẫu số | B / A | 2 |
| % | Phép toán lấy phần dư | B % A | 0 |
| ++ | Toán tử tăng, tăng giá trị nguyên lên 1 đơn vị | A++ | 11 |
| -- | Toán tử giảm, giảm giá trị nguyên xuống 1 đơn vị | A-- | 9 |

Ví dụ chương trình thực hiện các phép toán:

```

1  using System;
2  namespace Toantu
3  {
4      class Program
5      {
6          static void Main(string[] args)
7          {
8              // khai báo 3 biến số nguyên và khởi tạo giá trị
9              int iA = 22, iB= 10, iC;
10             iC = iA + iB;
11             Console.WriteLine("Đòng 1 - giá trị của C là :{0}", iC);
12             iC = iA - iB;
13             Console.WriteLine("Đòng 2 - giá trị của C là :{0}", iC);
14             iC = iA * iB;
15             Console.WriteLine("Đòng 3 - giá trị của C là :{0}", iC);
16             iC = iA / iB;
17             Console.WriteLine("Đòng 4 - giá trị của C là :{0}", iC);
18             iC = iA % iB;
19             Console.WriteLine("Đòng 5 - giá trị của C là :{0}", iC);
20             iC = iA++;
21             Console.WriteLine("Đòng 6 - giá trị của C là :{0}", iC);
22             iC = iA--;
23             Console.WriteLine("Đòng 7 - giá trị của C là :{0}", iC);
24             Console.Read();
25         }
26     }
27 }

```

Kết quả của chương trình

```

Đòng 1 - giá trị của C là :32
Đòng 2 - giá trị của C là :12
Đòng 3 - giá trị của C là :220
Đòng 4 - giá trị của C là :2
Đòng 5 - giá trị của C là :2
Đòng 6 - giá trị của C là :22
Đòng 7 - giá trị của C là :23

```

Toán tử quan hệ trong C# sẽ cho giá trị True hoặc False, với giá trị của 2 biến, A có giá trị là 10 và B có giá trị là 20, ta có bảng như sau:

| Toán tử | Mô tả | Ví dụ | Kết quả |
|---------|---|--------|---------|
| == | Kiểm tra nếu giá trị 2 biến bằng nhau thì trả về giá trị True, ngược lại False | A == B | False |
| != | Kiểm tra nếu giá trị 2 biến khác nhau thì trả về giá trị True, ngược lại False | A != B | True |
| > | Kiểm tra nếu giá trị biến bên trái lớn hơn biến bên phải thì trả về giá trị True, ngược lại False | A > B | False |
| < | Kiểm tra nếu giá trị biến bên trái bé hơn biến bên phải thì trả về giá trị True, ngược lại False | A < B | True |

| | | | |
|--------|---|------------|-------|
| \geq | Kiểm tra nếu giá trị biến bên trái lớn hơn hoặc bằng biến bên phải trả về giá trị True, ngược lại False | $A \geq B$ | False |
| \leq | Kiểm tra nếu giá trị biến bên trái bé hơn hoặc bằng biến bên phải trả về giá trị True, ngược lại False | $A \leq B$ | True |

Ví dụ

```

1  using System;
2  namespace ToantuQuanhe
3  {
4      class Program
5      {
6          static void Main(string[] args)
7          {
8              {
9
10             // khai bao 2 bien a,b kieu so nguyen
11             int iA = 21;
12             int iB = 10;
13             if (iA == iB) // so sanh bang
14             {
15                 // xuat ket qua sau khi so sanh
16                 Console.WriteLine("Line 1 - {0} is equal to {1}", iA, iB);
17             }
18             else
19             {
20                 Console.WriteLine("Line 1 - {0} is not equal to {1}", iA, iB);
21             }
22             if (a < b)// so sanh nho hon so a nho hon so b
23             {
24                 Console.WriteLine("Line 2 - {0} is less than {1}", iA, iB);
25             }
26             else
27             {
28                 Console.WriteLine("Line 2 - {0} is not less than {1}", iA, iB);
29             }
30             if (iA > iB) // so sanh so a lon hon so b
31             {
32                 Console.WriteLine("Line 3 - {0} is greater than {1}", iA, iB);
33             }
34             else
35             {
36                 Console.WriteLine("Line 3 - {0} is not greater than {1}", iA, iB);
37             }
38             /* thay doi gia tri cua A va B */
39             iA = 5;
40             iB = 20;
41             if (iA <= iB) // so sanh 2 so nho hay bang
42             {
43                 Console.WriteLine("Line 4 - {0} is either less than or equal to {1}", iA, iB);
44             }
45             if (iB >= iA) // so sanh 2 so lon hon hay bang
46             {
47                 Console.WriteLine("Line 5-{0} is either greater than or equal to {1}", iA, iB);
48             }
49         }
50     }
51 }
52 }
```

Kết quả của chương trình

```
Line 1 - 21 is not equal to 10
Line 2 - 21 is not less than 10
Line 3 - 21 is greater than 10
Line 4 - 5 is either less than or equal to 20
Line 5-20 is either greater than or equal to 5
```

Cách chuyển chuỗi ký tự số sang số trong ngôn ngữ C#

Cú pháp chung: <kieudulieu> tenBien=System.DataTypeObject.Parse("chuỗi số");

Ví dụ:

```
int iSo = System.Int16.Parse("123"); // chuyển chuỗi số "123" thành số 123.
```

```
double dSo = System.Double.Parse("1123455634");
```

2.3 Câu lệnh điều khiển

Trong ngôn ngữ C# các cấu trúc điều khiển cũng giống như ngôn ngữ C++.

2.3.1 Cấu trúc rẽ nhánh

a. Cấu trúc đơn giản

```
if (điều_kiện)
{
    <lệnh điều kiện>
}
```

Ý nghĩa của cấu trúc

Đây là cấu trúc đơn giản nhất của rẽ nhánh, nó có nghĩa là: nếu <điều kiện> đúng, thực hiện <lệnh điều kiện>

b. Cấu trúc đầy đủ

```
if (điều kiện 1)
{
    <lệnh điều kiện 1>
}
else if (điều kiện 2)
{
    <lệnh điều kiện 2>
}
else if (điều kiện n)
```

```
{  
    <lệnh điều kiện n>  
}  
else  
{  
    <lệnh>  
}
```

Ý nghĩa của cấu trúc

Nếu <điều kiện 1> đúng, thực hiện <lệnh điều kiện 1>, nếu không, tiếp tục kiểm tra <điều kiện 2>

Nếu <điều kiện 2> đúng, thực hiện <lệnh điều kiện 2>, nếu không, tiếp tục kiểm tra <điều kiện n>

Nếu <điều kiện n> đúng, thực hiện <lệnh điều kiện n>

Nếu tất cả điều kiện trên đều sai, thực hiện <lệnh>

Ví dụ dùng cấu trúc rẽ nhánh

Nhập điểm cho sinh viên và cho biết kết quả xếp loại học lực như sau.

Xếp loại học lực sinh viên:

- Nếu điểm ≥ 8 học lực Giỏi,
- Nếu $6.5 \leq$ điểm < 8 học lực Khá,
- Nếu $5.0 \leq$ điểm < 6.5 học lực Trung bình,
- Còn lại là học lực yếu.

```
1  using System;
2  namespace RenhanhIf
3  {
4      class Program
5      {
6          static void Main(string[] args)
7          {
8              // dong nhac nhap diem tu ban phim
9              Console.WriteLine("Nhập điểm sinh viên :");
10             // khai bao bien va nhap diem vao
11             float fDiem = float.Parse(Console.ReadLine());
12             if (fDiem >= 8.0) // kiem tra diem
13                 Console.WriteLine("Học lực: Gioi");
14             else if (fDiem >= 6.5) // kiem tra diem
15                 Console.WriteLine("Học lực: Kha");
16             else if (fDiem >= 5.0) // kiem tra diem
17                 Console.WriteLine("Học lực: Trung bình");
18             else
19                 Console.WriteLine("Học lực: Yeu");
20             Console.Read();
21         }
22     }
23 }
```

Kết quả chương trình như sau:

```
Nhập điểm sinh viên :8.5
Học lực: Gioi
```

2.3.2 Cấu trúc switch

Cú pháp chung

```
switch(biểu thức)
{
    case <b>iêu thức hằng 1</b>:
        // các lệnh cần thực thi
        break;

    case <b>iêu thức hằng 2</b>:
        // các lệnh cần thực thi
        break;
    ....
    case <b>iêu thức hằng n</b>:
        // các lệnh cần thực thi
        break;

    default:
        // các lệnh cần thực thi
}
```

Các quy tắc sau được áp dụng tới một lệnh switch:

- Biểu thức được sử dụng trong switch phải có kiểu là số nguyên hoặc liệt kê.
- Biểu thức **hằng** cho một case phải cùng kiểu dữ liệu với biến trong switch và nó phải là **hằng số**.
- Mỗi một case có lệnh break để ngắt ra khỏi cấu trúc switch
- Nếu không thực hiện các lựa chọn thì sẽ thực hiện các lệnh trong default.

Ví dụ: Viết chương trình nhập vào một số từ 1 đến 5 và đọc số đó bằng chữ

```
4  using System.Text;
5  namespace CauTrucSwitch
6  {
7      class Program
8      {
9          static void Main(string[] args)
10         {
11             int iN;// khai bao bien so nguyen
12             Console.WriteLine("Nhập vào số nguyên từ 1 đến 5: "); // dòng nhac
13             iN = System.Int32.Parse(Console.ReadLine());// nhap va chuyen sang so
14             switch (iN)
15             {
16                 case 1: Console.WriteLine("So Mot");
17                     break;
18                 case 2: Console.WriteLine("So Hai");
19                     break;
20                 case 3: Console.WriteLine("So Ba");
21                     break;
22                 case 4: Console.WriteLine("So Bon");
23                     break;
24                 case 5: Console.WriteLine("So Nam");
25                     break;
26                 default :
27                     Console.WriteLine("Không đọc được");
28                     break;
29             }
30             Console.Read();
31         }
32     }
33 }
```

Kết quả chạy chương trình

```
Nhap vao so nguyen tu 1 den 5: 2
So Hai
```

```
Nhap vao so nguyen tu 1 den 5: 8
Khong doc duoc
```

2.3.3 Cấu trúc lặp

Vòng lặp trong C# cũng giống như vòng lặp trong C++, nhằm lặp lại nhiều lần khi còn thỏa điều kiện

2.3.3.1 Cấu trúc lặp for

Cấu trúc lặp for được dùng cho trong trường hợp số lần lặp được xác định.

Cú pháp chung for

for (giá trị khởi tạo; điều kiện; tăng / giảm)

```
{  
    // khối lệnh  
}
```

Ví dụ: Viết chương trình tính tổng từ 1 đến n, n nhập từ bàn phím.

```
1  using System;  
2  using System.Collections.Generic;  
3  using System.Linq;  
4  using System.Text;  
5  using System.Threading.Tasks;  
6  namespace VongLapFor  
7  {  
8      class Program  
9      {  
10         static void Main(string[] args)  
11         {  
12             int iN;// khai báo biến số nguyên  
13             double dTong = 0;  
14             Console.Write("Nhập vào số nguyên n: "); // dòng nhắc  
15             iN = System.Int32.Parse(Console.ReadLine());// nhập và chuyển sang số  
16             // vòng lặp  
17             for (int i = 1; i <= iN; i++)  
18             {  
19                 dTong = dTong + i;  
20             }  
21             Console.WriteLine("Tổng các số từ 1 ->{0}: {1}", iN, dTong);  
22             Console.Read();  
23         }  
24     }  
25 }
```

Kết quả chương trình:

```
Nhap vao so nguyen n: 10  
Tong cac so tu 1 ->10: 55
```

2.3.3.2 Câu trúc lặp foreach

Dùng để lặp trong mảng hoặc một tập hợp

```
Cú pháp chung foreach  
foreach(<kiểu dữ liệu> tên in <mảng / tập hợp>)  
{  
    // lệnh thông qua tên trong câu trúc foreach  
}
```

Ví dụ: Cho một mảng các số nguyên và dùng **foreach** để duyệt và in ra các phần tử trong mảng.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  namespace LapForach
7  {
8      class Program
9      {
10         static void Main(string[] args)
11         {
12             int [] intArray ={1,2,3,4,5,6,7};
13             Console.WriteLine("In gia tri cac phan ti trong mang: ");
14             foreach(int item in intArray)
15             {
16                 Console.Write( "{0} " , item);
17             }
18             Console.Read();
19         }
20     }
21 }
```

Kết quả chương trình:

```
In gia tri cac phan ti trong mang:
1 2 3 4 5 6 7
```

2.3.3.3 Cấu trúc lặp while

Kiểm tra điều kiện trước khi thực hiện các lệnh trong vòng lặp. Nếu điều kiện không thỏa sẽ không thực hiện lần nào.

```
Cú pháp chung
while(điều kiện)
{
    // thực hiện lệnh khi thỏa điều kiện
}
```

Ví dụ: Viết chương trình in ra các số nhỏ hơn 10

```
4  using System.Text;
5  using System.Threading.Tasks;
6  namespace LapWhile
7  {
8      class Program
9      {
10         static void Main(string[] args)
11         {
12             int iA = 0; // khai báo giá trị iA là 0
13             Console.WriteLine("Chương trình in các số nhỏ hơn 10: ");
14
15             while (iA < 10) // kiểm tra điều kiện
16             {
17                 Console.Write("{0} ", iA);
18                 iA++; // tăng giá trị iA lên 1 đơn vị
19             }
20             Console.Read();
21         }
22     }
23 }
```

Kết quả chương trình

```
Chương trình in các số nhỏ hơn 10:
0 1 2 3 4 5 6 7 8 9
```

2.3.3.4 Câu trúc lặp do ... while

Câu trúc lặp do ...while chương trình thực hiện ít nhất được một lần sau đó mới kiểm tra điều kiện lặp.

Cú pháp chung

```
do{
    // thực hiện lệnh
}while(điều kiện);
```

Ví dụ: Viết chương trình in ra các số nhỏ hơn 10

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  namespace LapWhile
7  {
8      class Program
9      {
10         static void Main(string[] args)
11         {
12             int iA = 0; // khai tao gia tri iA = 0
13             Console.WriteLine("Chuong trinh in cac so nho hon 10: ");
14             do
15             {
16                 Console.Write("{0} ", iA);
17                 iA++; // tang gia tri iA len 1
18             } while (iA < 10);
19             Console.Read();
20         }
21     }
22 }
```

Kết quả chương trình:

```
Chuong trinh in cac so nho hon 10:
0 1 2 3 4 5 6 7 8 9 -
```

2.3.3.5 Câu lệnh break và continue:

Câu lệnh **break**: Dùng để thoát khỏi vòng lặp

Ví dụ: in ra các số nhỏ hơn 10 nhưng dùng lệnh break để ngắt ra khỏi vòng lặp khi chạy đến 5.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace BreakContinue
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             for (int i = 0; i < 10; i++)
14             {
15                 Console.Write("{0} ", i);
16                 if (i == 5) // kiem tra i bang 5
17                 {
18                     break; // ngat ra khoi vong lap
19                 }
20             }
21             Console.Read();
22         }
23     }
24 }
```

Kết quả chương trình

```
0 1 2 3 4 5
```

Câu lệnh **continue**: Sử dụng để bỏ qua phần còn lại của câu lệnh và bắt đầu vòng lặp ở câu lệnh đầu tiên.

Ví dụ: in ra các số nhỏ hơn 10 nhưng dùng lệnh continue để bỏ qua khi chạy đến 5 và tiếp tục lệnh kế tiếp.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace BreakContinue
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             for (int i = 0; i < 10; i++)
14             {
15                 if (i == 5) // kiểm tra i bằng 5
16                 {
17                     continue; // bỏ qua khi giá trị là 5
18                 }
19                 Console.WriteLine("{0} ", i);
20             }
21             Console.Read();
22         }
23     }
24 }
```

Kết quả chương trình:

```
0 1 2 3 4 6 7 8 9 -
```

2.4 Kiểu mảng

2.4.1 Khái niệm mảng một chiều:

- Tập hợp các đối tượng có cùng kiểu dữ liệu.
- Mỗi đối tượng trong mảng được gọi là một phần tử mảng.
- Các phần tử được truy xuất qua chỉ số mảng, trong C# chỉ số phần tử là các số nguyên không âm và bắt đầu từ 0 đến kích thước mảng -1.

2.4.2 Đặc điểm của mảng:

- Các phần tử mảng dùng chung một tên và truy xuất qua chỉ số.
- Một mảng cần có giới hạn số phần tử mà mảng có thể chứa.
- Phải cấp phát vùng nhớ mới có thể sử dụng mảng.
- Vị trí các ô nhớ của các phần tử được cấp phát liền kề nhau.

2.4.3 Cú pháp chung khai báo mảng:

| | |
|--|--|
| <p><kiểu dữ liệu>[] tenMang</p> <p>Kiểu dữ liệu: Xác định kiểu dữ liệu phần tử trong mảng</p> <p>[]: Số trong ngoặc xác định kích thước mảng một chiều</p> <p>Tên mảng: Xác định tên của mảng</p> | <p>Ví dụ khai báo mảng số nguyên có 10 phần tử</p> <pre>int [] mangA; mangA= new int [10];</pre> |
|--|--|

2.4.4 Khởi tạo các giá trị cho mảng

```
// khai báo và khởi tạo mảng các số nguyên
int [] mangA ={1,2,3,4};

// khai báo và khởi tạo mảng kiểu chuỗi
string [] hoTen = new string [] {"Nguyen Van An", "Tran Van On"};
```

2.4.5 Sử dụng mảng

- Kiểu mảng có thể dùng làm kiểu dữ liệu cho biến
- Kiểu dữ liệu trả về cho hàm
- Tham số truyền vào cho hàm

2.4.6 Một số phương thức đặt trưng cho mảng một chiều

| Tên thuộc tính hoặc phương thức mảng | Ý nghĩa |
|--------------------------------------|--|
| Length | Thuộc tính trả về số nguyên kiểu int là số phần tử tối đa của mảng |
| LongLength | Thuộc tính trả về số nguyên kiểu long là số phần tử tối đa của mảng |
| Sort() | Phương thức thực hiện sắp xếp mảng theo thứ tự |
| Clear() | Phương thức xóa hết dữ liệu trong mảng và đưa mảng về giá trị mặc định của kiểu, chỉ xóa giá trị của phần tử mảng. |
| Reverse() | Phương thức thực hiện đảo ngược thứ tự mảng một chiều |

2.4.7 Cách duyệt mảng một chiều

- Dùng để truy xuất đến các giá trị phần tử mảng thông qua chỉ số mảng.
- Chỉ số mảng là số nguyên tăng dần bắt đầu từ 0
- Dùng vòng lặp để duyệt qua các giá trị phần tử mảng

2.4.8 Ví dụ mảng một chiều

Viết chương trình nhập vào mảng số nguyên có kích thước là 10 và nhập giá trị cho các phần tử của mảng từ bàn phím. In các giá trị của các phần tử mảng vừa nhập ra màn hình.

```
static void Main(string[] args)
{
    int[] mangA; // khai báo mảng số nguyên
    mangA = new int[10]; // khởi tạo mảng có kích thước là 10
    // nhap giá trị cho các phần tử mảng
    for (int i = 0; i < 10; i++)
    {
        Console.WriteLine("Nhập giá trị cho mảng tại vị trí mangA[{0}] = ", i);
        mangA[i] = Int16.Parse(Console.ReadLine());
    }
    // in giá trị mảng vừa nhập ra màn hình
    Console.WriteLine("Giá trị mảng vừa nhập: \n");
    for (int i = 0; i < 10; i++)
    {
        Console.WriteLine("vị trí {0} : mangA[{1}] = {2}", i, i, mangA[i]);
    }
    Console.Read();
}
```

Kết quả chương trình

```
*Nhập giá trị cho mảng tại vị trí 0: 1
Nhập giá trị cho mảng tại vị trí 1: 2
Nhập giá trị cho mảng tại vị trí 2: 3
Nhập giá trị cho mảng tại vị trí 3: 4
Nhập giá trị cho mảng tại vị trí 4: 5
Nhập giá trị cho mảng tại vị trí 5: 6
Nhập giá trị cho mảng tại vị trí 6: 7
Nhập giá trị cho mảng tại vị trí 7: 8
Nhập giá trị cho mảng tại vị trí 8: 9
Nhập giá trị cho mảng tại vị trí 9: 10
Giá trị mảng vừa nhập:
vị trí 0 : MangA[0]:= 1
vị trí 1 : MangA[1]:= 2
vị trí 2 : MangA[2]:= 3
vị trí 3 : MangA[3]:= 4
vị trí 4 : MangA[4]:= 5
vị trí 5 : MangA[5]:= 6
vị trí 6 : MangA[6]:= 7
vị trí 7 : MangA[7]:= 8
vị trí 8 : MangA[8]:= 9
vị trí 9 : MangA[9]:= 10
```

Chúng ta sử dụng vòng lặp for để truy cập giá trị phần tử trong mảng. Trong C# chúng ta có thể dùng lệnh **foreach** để duyệt qua các giá trị phần tử trong mảng.

Chúng ta sử dụng lại ví dụ trên dùng foreach để in ra giá trị các phần tử của mảng.

Chương trình

```
static void Main(string[] args)
{
    int[] mangA; // khai báo mảng số nguyên
    mangA = new int[10]; // khởi tạo mảng có kích thước là 10
    // nhập giá trị cho các phần tử mảng
    for (int i = 0; i < 10; i++)
    {
        Console.WriteLine("Nhập giá trị cho mảng tại vị trí mangA[{0}] = ", i);
        mangA[i] = Int16.Parse(Console.ReadLine());
    }
    // in giá trị mảng vừa nhập ra màn hình
    Console.WriteLine("Giá trị mảng vừa nhập: \n");
    int k = 0;
    foreach (int j in mangA)
    {
        Console.WriteLine("vị trí {0} : mangA[{1}] = {2}", k, k, j);
        k++;
    }
    // dừng màn hình
    Console.Read();
}
```

Kết quả chương trình dùng vòng lặp foreach

```
Nhap gia tri cho mang tai vi tri 0: 1
Nhap gia tri cho mang tai vi tri 1: 2
Nhap gia tri cho mang tai vi tri 2: 3
Nhap gia tri cho mang tai vi tri 3: 4
Nhap gia tri cho mang tai vi tri 4: 5
Nhap gia tri cho mang tai vi tri 5: 6
Nhap gia tri cho mang tai vi tri 6: 7
Nhap gia tri cho mang tai vi tri 7: 8
Nhap gia tri cho mang tai vi tri 8: 9
Nhap gia tri cho mang tai vi tri 9: 10
Gia tri mang vua nhap:
vi tri 0 : MangA[0]:= 1
vi tri 1 : MangA[1]:= 2
vi tri 2 : MangA[2]:= 3
vi tri 3 : MangA[3]:= 4
vi tri 4 : MangA[4]:= 5
vi tri 5 : MangA[5]:= 6
vi tri 6 : MangA[6]:= 7
vi tri 7 : MangA[7]:= 8
vi tri 8 : MangA[8]:= 9
vi tri 9 : MangA[9]:= 10
```

2.5 Collections

Collections có một số đặc điểm là một mảng có kích thước động, không cần khai báo kích thước khi khởi tạo, có thể tăng giảm số lượng phần tử trong mảng, có thể lưu trữ một tập hợp các đối tượng thuộc nhiều kiểu dữ liệu khác nhau, hỗ trợ nhiều phương thức để thao tác với tập hợp, mỗi Collections được tổ chức thành một lớp nên cần khởi tạo khi sử dụng.

Khai báo thư viện để sử dụng các lớp có trong Collections

System.collections

2.5.1 Một số Collections thông dụng

| Lớp | Mô tả |
|------------|--|
| Array | Cung cấp phương thức để khởi tạo, thao tác tìm kiếm, sắp xếp,... |
| ArrayList | Lớp cho phép lưu trữ và quản lý các phần tử mảng, tuy nhiên không giống mảng, ta có thể thêm hoặc xóa phần tử một cách linh hoạt và có thể điều chỉnh kích cỡ một cách tự động |
| SortList | Lớp lưu trữ dưới dạng cặp Key – Value, khi đó ta có thể truy xuất các phần tử trong danh sách thông qua Key, thay vì thông qua chỉ số mảng như mảng bình thường |
| Stack | Cho phép lưu trữ và thao tác dữ liệu theo cấu trúc LIFO |
| Queue | Cho phép lưu trữ và thao tác dữ liệu theo cấu trúc FIFO |
| Dictionary | Biểu diễn tập hợp Keys và Values trong Dictionary |

Khai báo và sử dụng các phương thức ArrayList: Khi sử dụng lớp ArrayList chúng ta phải khởi tạo lớp với các loại Constructor của lớp ArrayList đã được định nghĩa.

2.5.2 Sử dụng các phương thức và thuộc tính có trong lớp ArrayList

| Phương thức hoặc thuộc tính | Mô tả |
|--|--|
| Count | Thuộc tính trả về 1 số nguyên là số phần tử có trong ArrayList |
| Add(object value) | Phương thức thêm một đối tượng vào cuối ArrayList |
| AddRange(Collection ListObject) | Phương thức thêm danh sách từ ListObject vào cuối ArrayList |
| Clear() | Phương thức xóa hết các phần tử trong ArrayList |
| Contains(Object value) | Kiểm tra đối tượng value có tồn tại trong ArrayList hay không |
| GetRange(int startIndex, int EndIndex) | Phương thức trả về 1 ArrayList bao gồm các phần tử từ vị trí startIndex đến EndIndex trong ArrayList ban đầu |

| | |
|--|--|
| IndexOf(object value) | Phương thức trả về giá trị đầu tiên xuất hiện đối tượng value trong ArrayList, Không tìm thấy trả về -1 |
| Insert(int Index, object value) | Phương thức chèn đối tượng value vào vị trí Index trong ArrayList |
| InsertRange(int index, Icollection Listobject) | Phương thức chèn danh sách ListObject vào vị trí Index trong ArrayList |
| LastIndexOf(object value) | Phương thức trả về vị trí xuất hiện cuối cùng của đối tượng value trong ArrayList, không tìm thấy trả về -1 |
| Remove(object value) | Phương thức xóa đối tượng xuất hiện đầu tiên trong ArrayList |
| RemoveAt(int index) | Phương thức xóa phần tử tại vị trí index trong ArrayList |
| RemoveRange(int index, int count) | Phương thức xóa bỏ một danh sách trong ArrayList từ vị trí Index và số phần tử count cần xóa trong ArrayList |
| Reverse() | Phương thức đảo ngược các phần tử trong ArrayList |
| Sort() | Phương thức sắp xếp các phần tử trong ArrayList theo thứ tự tăng dần |
| ToArray() | Phương thức trả về một mảng object chứa các phần tử được sao chép từ ArrayList |
| BinarySort(object value) | Phương thức tìm kiếm đối tượng value trong ArrayList theo thuật toán tìm kiếm nhị phân Nếu tìm thấy sẽ trả về vị trí của phần tử, ngược lại trả về giá trị âm Lưu ý: ArrayList phải được sắp xếp trước khi sử dụng |

2.5.3 Một số ví dụ

Ví dụ 1: Áp dụng Lớp ArrayList sử dụng phương thức Add để thêm các phần tử vào ArrayList

```
static void Main(string[] args)
{
    // khai báo ArrayList
    ArrayList arrlItem = new ArrayList();
    arrlItem.Add(1); // them phan tu vao arrayList
    arrlItem.Add("a"); // them phan tu vao arrayList
    arrlItem.Add(5); // them phan tu vao arrayList
    // in các phần tử trong ArrayList
    Console.Write("Gia tri co trong ArrayList la: ");
    foreach (object item in arrlItem)
        Console.Write(item + " ");
    Console.Read();
}
```

Kết quả chương trình

```
Gia tri co trong ArrayList la: 1 a 5
```

Ví dụ 2: Sử dụng phương thức AddRange để thêm các phần tử cho ArrayList

```
static void Main(string[] args)
{
    // khai báo ArrayList
    ArrayList arrlItem = new ArrayList();
    arrlItem.AddRange(new int[] { 1, 1, 5, 4 });
    // in các phần tử trong ArrayList
    Console.Write("Gia tri co trong ArrayList la: ");
    foreach (int value in arrlItem)
        Console.Write(value + " ");
    Console.Read();
}
```

Kết quả chương trình

```
Gia tri co trong ArrayList la: 1 1 5 4
```

Ví dụ 3: Sử dụng phương thức Remove và RemoveAt để xóa bỏ một giá trị trong ArrayList

```
static void Main(string[] args)
{
    // khai báo ArrayList
    ArrayList arrlItem = new ArrayList();
    arrlItem.AddRange(new int[] { 1, 1, 5, 4 });
    // xóa bỏ phần tử trong arraylist
    arrlItem.Remove(5); // xóa bỏ giá trị phần tử là 5
    // in các phần tử trong ArrayList
    Console.Write("ArrayList sau khi xóa là: ");
    foreach (int value in arrlItem)
        Console.Write(value + " ");
    Console.Read();
}
```

Kết quả chương trình sau khi xóa giá trị phần tử 5 trong ArrayList

```
ArrayList sau khi xoa la: 1 1 4
```

Ví dụ 4: Xóa giá trị tại một vị trí trong ArrayList dùng phương thức RemoveAt

```
static void Main(string[] args)
{
    // khai báo ArrayList
    ArrayList arrlItem = new ArrayList();
    arrlItem.AddRange(new int[] { 1, 1, 5, 4 });
    // xóa bỏ phần tử trong arraylist
    arrlItem.RemoveAt(3); // xóa bỏ giá trị phần tử tại vị trí 3
    /// in các phần tử trong ArrayList
    Console.WriteLine("ArrayList sau khi xoa la: ");
    foreach (int value in arrlItem)
        Console.Write(value + " ");
    Console.Read();
}
```

Kết quả chương trình

```
ArrayList sau khi xoa la: 1 1 5
```

Ví dụ 5: Xóa bỏ một số giá trị từ vị trí bắt đầu và số phần tử trong ArrayList

```
static void Main(string[] args)
{
    // khai báo ArrayList
    ArrayList arrlItem = new ArrayList();
    arrlItem.AddRange(new int[] { 1, 1, 5, 4 });
    Console.WriteLine("ArrayList trước khi xóa là: ");
    foreach (int value in arrlItem)
        Console.Write(value + " ");
    // xóa bỏ phần tử trong arraylist
    arrlItem.RemoveRange(1, 2); // xóa bỏ giá trị từ vị trí 1 và xóa 2 phần tử
    // in các phần tử trong ArrayList
    Console.WriteLine("\nArrayList sau khi xoa la: ");
    foreach (int value in arrlItem)
        Console.Write(value + " ");
    Console.Read();
}
```

Kết quả chương trình

```
ArrayList trước khi xóa là: 1 1 5 4
ArrayList sau khi xoa la: 1 4
```

Ví dụ 6: Tìm giá trị 1 có trong ArrayList sử dụng phương thức Contains

```
static void Main(string[] args)
{
    // khai báo ArrayList
    ArrayList arrlItem = new ArrayList();
    arrlItem.AddRange(new int[] { 1, 1, 5, 4 });
    // tìm phần tử có chứa trong ArrayList không
    bool Found = arrlItem.Contains(1);
    if (Found == true)
        Console.WriteLine("Tim thay gia tri trong ArrayList");
    else
        Console.WriteLine("Khong tim thay gia tri trong ArrayList");
    Console.Read();
}
```

Kết quả trả về tìm thấy hoặc không tìm thấy

```
Tim thay gia tri trong ArrayList
```

Ví dụ 7: Tìm giá trị phần tử là 5 có trong ArrayList không và trả về vị trí xuất hiện đầu tiên của đối tượng trong ArrayList sử dụng phương thức IndexOf.

```
static void Main(string[] args)
{
    // khai báo ArrayList
    ArrayList arrlItem = new ArrayList();
    arrlItem.AddRange(new int[] { 1, 1, 5, 4, 5 });
    // tìm phần tử đầu tiên xuất hiện trong ArrayList không
    int pos = arrlItem.IndexOf(5);
    if (pos > 0)
        Console.WriteLine("Tim thay gia tri tai vi tri dau tien {0}", pos);
    else
        Console.WriteLine("Khong tim thay gia tri trong ArrayList");
    Console.Read();
}
```

Kết quả chương trình

```
Tim thay gia tri tai vi tri dau tien 2
```

Ví dụ 8: Tìm giá trị phần tử là 5 có trong ArrayList và trả về vị trí xuất hiện cuối cùng của đối tượng trong ArrayList sử dụng phương thức LastIndexOf

```

static void Main(string[] args)
{
    // khai báo ArrayList
    ArrayList arrlItem = new ArrayList();
    arrlItem.AddRange(new int[] { 1, 1, 5, 4, 5 });
    // tìm phần tử bắt đầu từ vị trí có chứa trong ArrayList không
    int pos = arrlItem.LastIndexOf(5);
    if (pos > 0)
        Console.WriteLine("Tim thay gia tri tai vi tri {0}", pos);
    else
        Console.WriteLine("Khong tim thay gia tri trong ArrayList");
    Console.Read();
}

```

Kết quả chương trình

Tim thay gia tri tai vi tri 4

2.6 Xử lý chuỗi

Trong C# bạn có thể sử dụng chuỗi là mảng các ký tự, tuy nhiên ta thường dùng từ khóa string để khai báo một chuỗi.

Trong C# String được khai báo là một lớp, chúng ta có các thuộc tính và phương thức để xử lý chuỗi.

Các thuộc tính trong lớp String

- Chars: Xác định đối tượng char tại vị trí xác định trong string là một ký tự
- Length: Trả về một số nguyên là số ký tự trong một chuỗi

Các phương thức trong lớp String

| Phương thức | Ý nghĩa |
|---|---|
| public static int Compare(string strA, string strB) | So sánh 2 đối tượng chuỗi được chỉ định và trả về một số nguyên cho biết vị trí tương đối của chúng <ul style="list-style-type: none"> ✓ 0 strA bằng strB ✓ 1 strA lớn hơn strB ✓ -1 strA nhỏ hơn strB |
| public static int Compare(string strA, string strB, bool ignoreCase) | So sánh 2 đối tượng string và tùy thuộc có phân biệt chữ Hoa và chữ thường. |
| public int CompareTo(string strB) | So sánh một đối tượng string với đối tượng string được chỉ định, |

| | |
|---|---|
| | kết quả trả về là một số nguyên: 0,1,-1 |
| public static string Copy(string str) | Sao chép một chuỗi với giá trị giống chuỗi đã được quy định trước |
| public bool Contains(string value) | Trả về giá trị cho biết chuỗi con có trong chuỗi được chỉ định. Kết quả True hoặc False |
| public bool EndsWith(string value) | Xác định chuỗi kết thúc có phù hợp với chuỗi đã chỉ định không, kết quả trả về True hoặc False |
| public static string Format (string format, params Object[] args) | Định dạng chuỗi theo quy định với cách biểu diễn chuỗi của một đối tượng tương ứng |
| public int IndexOf(string value) | Trả về vị trí đầu tiên khi tìm thấy giá trị chuỗi cần tìm trong một chuỗi được chỉ định |
| public int IndexOf(string value, StringComparison comparisonType) | Trả về vị trí đầu tiên khi tìm thấy giá trị chuỗi cần tìm trong một chuỗi được chỉ định. Có xác định loại tìm kiếm đã chỉ định |
| public int LastIndexOf(char value) | Trả về chỉ mục (dựa trên cơ sở 0) cho sự xuất hiện cuối cùng của ký tự Unicode đã cho bên trong đối tượng String hiện tại |
| public int LastIndexOf(string value) | Trả về chỉ mục (dựa trên cơ sở 0) cho sự xuất hiện cuối cùng của một chuỗi đã cho bên trong đối tượng String hiện tại |
| public string Remove(int startIndex) | Gỡ bỏ tất cả ký tự trong Instance hiện tại, bắt đầu tại vị trí đã xác định và tiếp tục tới vị trí cuối cùng, và trả về chuỗi đó |

| | |
|--|--|
| <code>public string Remove(int startIndex, int count)</code> | Gỡ bỏ số ký tự đã cho trong chuỗi hiện tại bắt đầu tại một vị trí đã xác định và trả về chuỗi đó |
| <code>public string Replace(char oldChar, char newChar)</code> | Thay thế tất cả ký tự Unicode đã cho xuất hiện trong đối tượng String hiện tại với ký tự Unicode đã xác định và trả về chuỗi mới |
| <code>public string Replace(string oldValue, string newValue)</code> | Thay thế tất cả chuỗi đã cho xuất hiện trong đối tượng String hiện tại với đối tượng string đã xác định và trả về chuỗi mới |
| <code>public string[] Split(params char[] separator)</code> | Trả về một mảng chuỗi mà chứa các chuỗi phụ trong đối tượng String hiện tại, được giới hạn bởi các phần tử của một mảng ký tự Unicode đã cho |
| <code>public string[] Split(char[] separator, int count)</code> | Trả về một mảng chuỗi mà chứa các chuỗi phụ trong đối tượng String hiện tại, được giới hạn bởi các phần tử của một mảng ký tự Unicode đã cho. Tham số int xác định số chuỗi phụ tối đa để trả về |
| <code>public bool StartsWith(string value)</code> | Xác định có hay không phần bắt đầu của instance của chuỗi này khớp với chuỗi đã cho |
| <code>public char[] ToCharArray()</code> | Trả về một mảng ký tự Unicode với tất cả ký tự trong đối tượng String hiện tại |
| <code>public string ToLower()</code> | Trả về một bản sao của chuỗi này đã được biến đổi thành chữ thường |

| | |
|-------------------------|---|
| public string ToUpper() | Trả về một bản sao của chuỗi này đã được biến đổi thành chữ hoa |
| public string Trim() | Gỡ bỏ tất cả ký tự whitespace từ đối tượng String hiện tại |

2.7 Xử lý ngoại lệ

2.7.1 Xử lý ngoại lệ (Exception)

Xử lý ngoại lệ là một vấn đề xuất hiện trong khi thực thi chương trình. Một **Exception** trong C# là phản hồi về một tình huống ngoại lệ xuất hiện trong khi chương trình đang chạy. Ví dụ trong chương trình có phép toán chia cho Zero(không), trường hợp này sẽ phát sinh ngoại lệ không chia được cho Zero và chúng ta muốn chương trình vẫn tiếp tục chạy thì chúng ta phải xử lý trường hợp ngoại lệ khi chia cho Zero.

Xử lý ngoại lệ: Khi chia cho Zero. Xử lý ngoại lệ trong C# được xây dựng trên những từ khóa: **try, catch, finally, throw**.

- **try:** Khởi lệnh chương trình mà ở đó có thể phát sinh ra các ngoại lệ.
- **catch:** Xử lý ngoại lệ tại vị trí trong chương trình phát sinh ngoại lệ cần được xử lý, từ khóa **catch** trong C# chỉ dẫn bắt một xử lý ngoại lệ.
- **finally:** Thực thi một tập lệnh đã cho.
- **throw:** Ném lỗi khi xuất hiện vấn đề, được dùng từ khóa **throw** trong C#.

2.7.2 Cú pháp chung khi xử lý ngoại lệ

```

try
{
    // các lệnh thực thi chương trình có thể gây ra ngoại lệ (Exception)
}
catch( tên các ngoại lệ e1 )
{
    // các lệnh xử lý ngoại lệ
}
catch( tên các ngoại lệ e2 )
{
    // các lệnh xử lý ngoại lệ
}
...
catch( tên các ngoại lệ en )
{
}

```

```

        // các lệnh xử lý ngoại lệ
    }
finally
{
    // các lệnh thực thi
}

```

2.7.3 Lớp Exception trong C#

Các Exception trong C# được biểu diễn bởi các lớp, các lớp Exception trong C# chủ yếu kế thừa từ lớp **System.Exception**. Một số lớp kế thừa từ lớp Exception là ApplicationException và SystemException.

Lớp ApplicationException hỗ trợ các Exception được tạo bởi chương trình ứng dụng.

SystemException: Là lớp cơ sở cho tất cả SystemException

2.7.4 Một số lớp Exception kế thừa từ lớp SystemException

| Lớp Exception | Mô tả |
|-----------------------------------|---|
| System.IO.IOException | Xử lý ngoại lệ về nhập/ xuất |
| System.IndexOutOfRangeException | Xử lý ngoại lệ khi một phương thức tham chiếu tới một chỉ mục bên ngoài dãy mảng. |
| System.ArrayTypeMismatchException | Xử lý ngoại lệ khi kiểu là không phù hợp với kiểu mảng |
| System.NullReferenceException | Xử lý ngoại lệ từ việc tham chiếu một đối tượng null. |
| System.DivideByZeroException | Xử lý ngoại lệ khi chia cho số 0. |
| System.InvalidCastException | Xử lý ngoại lệ trong khi ép kiểu. |
| System.OutOfMemoryException | Xử lý ngoại lệ việc thiếu bộ nhớ |
| System.StackOverflowException | Xử lý ngoại lệ việc tràn ngăn xếp (stack) |

Ví dụ 1: Xử lý ngoại lệ khi chương trình chia cho zero

```
1  using System;
2  namespace ChiaChoZero
3  {
4      class Program
5      {
6          static void Main(string[] args)
7          {
8              int iA, iB; // khai bao bien
9              float fKetqua;// khai bao bien
10             try
11             {
12                 Console.WriteLine("Nhập giá trị a: ");
13                 iA = int.Parse(Console.ReadLine());
14                 Console.WriteLine("Nhập giá trị b: ");
15                 iB = int.Parse(Console.ReadLine());
16                 // chia hai số
17                 fKetqua = iA / iB;
18                 Console.WriteLine("Kết quả : {0}", fKetqua);
19             }
20             catch (DivideByZeroException ex) // xử lý khi chia cho zero
21             {
22                 Console.WriteLine("{0}", ex.Message);
23             }
24             catch (Exception) // xử lý ngoại lệ khác
25             {
26                 Console.WriteLine("Xử lý ngoại lệ khác");
27             }
28             finally // xử lý trước khi kết thúc chương trình
29             {
30                 Console.WriteLine("Xử lý trước khi kết thúc chương trình ");
31                 Console.ReadKey();
32             }
33         }
34     }
35 }
```

Kết quả chương trình

| | |
|---------------------------------------|---------------------------------------|
| Nhập giá trị a: 4 | Nhập giá trị a: 4 |
| Nhập giá trị b: 2 | Nhập giá trị b: 0 |
| Kết quả : 2 | Attempted to divide by zero. |
| Xử lý trước khi kết thúc chương trình | Xử lý trước khi kết thúc chương trình |

Ví dụ 2: Đọc tập tin từ thư mục và xử lý các trường hợp ngoại lệ

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.IO;
7
8  namespace XuLyNgoaiLe
9  {
10    class Program
11    {
12      static void Main(string[] args)
13      {
14        string FileName = "d:\\file.txt";
15        try
16        {
17          FileStream fs = new FileStream(FileName, FileMode.Open);
18          StreamReader reader = new StreamReader(fs, Encoding.UTF8);
19          Console.WriteLine("File {0} successfully opened.", FileName);
20          Console.WriteLine("File contents:");
21          using (reader)
22          {
23            Console.WriteLine(reader.ReadToEnd());
24          }
25        }
26        catch (FileNotFoundException ex) // Xu ly ngoai le khi khong co ten tap tin
27        {
28          Console.Error.WriteLine("{0}", ex.Message);
29        }
30        catch (DirectoryNotFoundException dr)// Xu ly ngoai le khi khong co ten thu muc
31        {
32          Console.Error.WriteLine(" {0}", dr.Message);
33        }
34        catch (IOException e)// Xu ly ngoai le khac
35        {
36          Console.Error.WriteLine("{0}", e.Message);
37        }
38        finally // ket thuc chuong trinh
39        {
40          Console.Read();
41        }
42      }
43    }
44  }
```

Kết quả chương trình: Nếu có tập tin thì chương trình sẽ đọc nội dung của tập tin ghi ra màn hình

```
File d:\\file.txt successfully opened.
File contents:
Chuong trinh demo
Xu ly ngoai le
```

Nếu không ghi đúng đường dẫn chương trình sẽ ghi ra trên màn hình

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.IO;
7
8  namespace XuLyNgoaiLe
9  {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14             string FileName = "h:\\file.txt";
15             try
16             {
17                 FileStream fs = new FileStream(FileName, FileMode.Open);
18                 StreamReader reader = new StreamReader(fs, Encoding.UTF8);
19                 Console.WriteLine("File {0} successfully opened.", FileName);
20                 Console.WriteLine("File contents:");
21                 using (reader)
22                 {
23                     Console.WriteLine(reader.ReadToEnd());
24                 }
25             }
26             catch (FileNotFoundException ex) // Xu ly ngoai le khi khong co ten tap tin
27             {
28                 Console.Error.WriteLine("{0}", ex.Message);
29             }
30             catch (DirectoryNotFoundException dr)// Xu ly ngoai le khi khong co ten thu muc
31             {
32                 Console.Error.WriteLine(" {0}", dr.Message);
33             }
34             catch (IOException e)// Xu ly ngoai le khac
35             {
36                 Console.Error.WriteLine("{0}", e.Message);
37             }
38             finally // ket thuc chuong trinh
39             {
40
41                 Console.Read();
42             }
43         }
44     }
45 }
46 }
```

Kết quả chương trình khi đường dẫn không đúng

```
Could not find a part of the path 'H:\\file.txt'.
```

2.8 Đọc ghi tập tin

Sử dụng lớp I/O trong thư viện System.IO, để đọc dữ liệu hoặc ghi dữ liệu xuống một tập tin text, trong lớp File cung cấp một số phương thức dùng để đọc ghi tập tin.

2.8.1 Đọc tập tin

Đọc tập tin chúng ta dùng 2 cách:

Sử dụng *StreamReader*:

Cú pháp chung:

```
FileStream fs = new FileStream(FileName, FileMode.Open);
```

StreamReader read = new StreamReader(fs, Encoding.Unicode);

Để đọc nội dung của tập tin từ read chúng ta có thể dùng 2 phương thức được xây dựng sẵn:

- + *ReadToEnd()*: Dùng để đọc toàn bộ dữ liệu từ read
- + *ReadLine()*: Dùng để đọc từng dòng trong read
- Sử dụng đọc trực tiếp từ File cú pháp chung
string []lines = File.ReadAllLines(FileName);
FileName: Là tên tập tin được chỉ dẫn trên đĩa.

2.8.2 Ghi tập tin

- Sử dụng StreamWriter: Dùng để ghi dữ liệu xuống tập tin, cú pháp chung
FileStream fs = new FileStream(FileName, FileMode.Create);
FileMode: Dùng để chọn Mode để ghi mới hoặc ghi đè lên tập tin, có các Mode để ghi tập tin:
 - Create*: Tạo mới nếu tập tin chưa tồn tại
 - Append*: Ghi tiếp vào tập tin đã tồn tại
 - CreateNew*: Tạo mới tập tin

StreamWriter sWrite = new StreamWriter(fs, Encoding.Unicode);

- Sử dụng *sWrite.WriteLine()*: Dùng để ghi vào file và xuống dòng
- Sử dụng *sWrite.Write()*: Dùng ghi tiếp nhưng không xuống dòng
- Sử dụng ghi trực tiếp từ File:
 - *WriteAllLines(string filepath, string[] line)*: Dùng để ghi mảng các dòng xuống tập tin. Mỗi chuỗi trong mảng được ghi thành từng dòng trong tập tin.
 - *WriteAllText(string filepath, string str)*: Hàm ghi chuỗi vào tập tin
 - *filepath*: Đường dẫn đến tên tập tin để ghi

Ví dụ 1: Sử dụng lớp File để đọc và ghi tập tin

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.IO;
7  namespace DocGhiFile
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.OutputEncoding = Encoding.Unicode;
14             string fileLPath = @"d:\Line.txt";
15             string fileSPath = @"d:\string.txt";
16             // ghi tap tin
17             try
18             {
19                 string[] lines = new string[2];
20                 lines[0] = "Write data to file with C#.";
21                 lines[1] = "STDIO.VN";
22                 // ghi vao tap tin su dung phuong thuc WriteAllLines
23                 File.WriteAllLines(fileLPath, lines);
24             }
25             catch (Exception ex)
26             {
27                 Console.WriteLine("Loi: " + ex.Message);
28             }
29
30             string str;
31             // ghi tap tin
32             try
33             {
34                 str = "Write data to file with C#.\\r\\nSTDIO.VN";
35                 File.WriteAllText(fileSPath, str);
36             }
37             catch (Exception ex)
38             {
39                 Console.WriteLine("Loi: " + ex.Message);
40             }
41             try
42             {
43                 // doc tap tin tu file va ghi ra man hinh
44                 string[] li = File.ReadAllLines(fileLPath);
45                 foreach (string item in li)
46                 {
47                     Console.WriteLine("{0}", item);
48                 }
49             }
50             catch (Exception ex)
51             {
52                 Console.WriteLine("Loi " + ex.Message);
53             }
54             Console.Read();
55         }
56     }
}
```

Kết quả chương trình

```
Write data to file with C#.
STDIO.VN
```

Ví dụ 2: Sử dụng StreamReader để đọc tập tin

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.IO;
7  namespace DocGhiStreamWriter
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             FileStream fs = new FileStream("d:\\\\test.txt", FileMode.Open);
14             StreamReader rd = new StreamReader(fs, Encoding.UTF8);
15             try
16             {
17                 String giatri = rd.ReadToEnd();
18                 Console.WriteLine(giatri);
19             }
20             catch (Exception ex)
21             {
22                 Console.WriteLine("Loi " + ex.Message);
23             }
24             finally
25             {
26                 rd.Close();
27                 Console.ReadLine();
28             }
29         }
30     }
31 }
32
```

Kết quả chương trình

```
Day la chuong trinh dung StreamReader
Su dung de doc noi dung tap tin tu dia
```

Ví dụ 3: Sử dụng StreamWriter để ghi dữ liệu xuống tập tin và đọc lại và ghi ra màn hình

Code chương trình

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.IO;
7  namespace DocGhiStreamWriter
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.WriteLine("Ban nhap du lieu can luu: ");
14             String dulieu = Console.ReadLine();
15             String filepath = "D:\\test.txt";// đường dẫn của file muốn tạo
16             FileStream fs = new FileStream(filepath, FileMode.Create);//Tạo file mới tên là test.txt
17             try
18             {
19                 StreamWriter sWriter = new StreamWriter(fs, Encoding.UTF8);//fs là 1 FileStream
20                 sWriter.WriteLine("Chuong trinh luu file dung StreamWriter");
21                 sWriter.WriteLine(dulieu ); // ghi xuống file du lieu nhap tu ban phim
22                 // Ghi và đóng file
23                 sWriter.Flush();
24             }
25             catch (Exception ex)
26             {
27                 Console.WriteLine("Loi " + ex.Message);
28             }
29             finally
30             {
31                 fs.Close();// dong file
32             }
33             FileStream fr = new FileStream("d:\\test.txt", FileMode.Open);
34             StreamReader rd = new StreamReader(fr, Encoding.UTF8);
35             try|
36             {
37                 String giatri = rd.ReadToEnd();
38                 Console.WriteLine(giatri);
39             }
40             catch (Exception ex)
41             {
42                 Console.WriteLine("Loi " + ex.Message);
43             }
44             finally
45             {
46                 rd.Close();
47                 Console.ReadLine();
48             }
49         }
50     }
51 }
52 }
```

Kết quả chương trình

```
Ban nhap du lieu can luu:
Chuong trinh demo doc ghi file bang StreamReader va StreamWriter
Chuong trinh luu file dung StreamWriter
Chuong trinh demo doc ghi file bang StreamReader va StreamWriter
```

2.9 Bài tập

HỌC PHẦN LẬP TRÌNH ỨNG DỤNG BÀI TẬP THỰC HÀNH SỐ 1

I. Thông tin chung:

- Mã số bài tập : BT-LTUD-01
- Hình thức nộp bài : Nộp qua Moodle môn học
- Thời hạn nộp bài : ... / ... /
- Nội dung : Chương 2: Nền tảng C# cơ bản

Chuẩn đầu ra cần đạt:

L.O.1 Sử dụng được các kiểu dữ liệu, cấu trúc của ngôn ngữ lập trình C# để giải quyết bài toán vừa và nhỏ

L.O.4 Viết code đúng chuẩn

A. CẤU TRÚC RÊ NHÁNH

Bài tập 1: Viết chương trình giải phương trình bậc nhất $ax + b = 0$

Bài tập 2: Viết chương trình giải phương trình bậc 2.

Bài tập 3: Xây dựng ứng dụng Tìm số lớn nhất giữa 3 số

Bài tập 4: Viết chương trình chào theo giờ

Tìm câu chào bằng tiếng Anh tương ứng với giờ. Biết rằng

- Giờ từ 20h đến trước 4h: Good night!
- Giờ từ 4h đến 12h: Good morning!
- Giờ từ sau 12h đến 17h: Good afternoon!
- Giờ từ sau 17h đến 20h: Good evening!
- Xuất câu chào ra màn hình

Bài tập 5: Viết chương trình nhập vào năm dương lịch và xuất ra năm âm lịch tương ứng.

Biết rằng: Năm âm lịch = Can + Chi

Can = (Năm dương) % 10;

| | | | | | | | | | |
|-------------|------------|-------------|------------|-------------|-----------|-------------|-------------|------------|-----------|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Canh | Tân | Nhâm | Quý | Giáp | Ất | Bính | Đinh | Mậu | Kỷ |

Chi = (Năm dương) % 12

| | | | | | | | | | | | |
|-------------|------------|-------------|------------|-----------|------------|------------|------------|-------------|-----------|------------|------------|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Thân | Dậu | Tuất | Hợi | Tý | Sửu | Dần | Mẹo | Thìn | Tỵ | Ngọ | Mùi |

B. CẤU TRÚC LẶP

Bài tập 1: Viết chương trình cho phép nhập vào một số n và in ra màn hình giá trị [-n..n].
Viết với các cấu trúc lặp for, while, và do while.

Bài tập 2: Viết chương trình cho phép nhập vào một số n và in ra màn hình số chẵn từ [-n..n].
Viết với các cấu trúc lặp for, while, và do while

Bài tập 3: Viết chương trình nhập vào số n và kiểm tra xem n có phải số nguyên tố hay không?

C. MẢNG MỘT CHIỀU

Bài tập 1: Nhập vào dãy số, các số ngăn cách nhau bởi dấu phẩy. In ra:

1. Tổng các số thực âm và tổng số thực dương;
2. Tổng số thực lớn hơn X và tổng số thực nhỏ hơn X (X được nhập từ bàn phím);
3. Tổng các số nguyên chẵn và các số nguyên lẻ;
4. Tổng số thực âm lớn nhất và số thực dương nhỏ nhất

BÀI TẬP XỬ LÝ CHUỖI

Bài tập 1: Viết chương trình nhập vào 2 chuỗi và thực hiện các chức năng như sau:

1. So sánh hai chuỗi vừa nhập (có phân biệt chữ thường và chữ Hoa) xuất kết quả ra màn hình (sử dụng Compare)
2. So sánh hai chuỗi vừa nhập (không phân biệt chữ thường và chữ Hoa) xuất kết quả ra màn hình (sử dụng Compare)
3. Nối chuỗi 1 và chuỗi 2 và xuất kết quả ra màn hình (sử dụng Concat)
4. Cho biết vị trí xuất hiện chuỗi 2 trong chuỗi 1 (sử dụng indexof) xuất vị trí bắt đầu xuất hiện chuỗi 2 ra màn hình
5. Chèn chuỗi 2 vào sau cuối của chuỗi 1 (sử dụng Insert) và xuất kết quả ra màn hình

CHƯƠNG 3. LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG C#

Học xong chương này, người học có thể:

- + Trình bày các đặc điểm về lập trình hướng đối tượng.
- + Xây dựng ứng dụng theo hướng đối tượng.

3.1 Tổng quan lập trình hướng đối tượng

Lập trình hướng đối tượng là một kỹ thuật hỗ trợ đối tượng, điểm mạnh của lập trình hướng đối tượng là tái sử dụng lại code, cung cấp một cấu trúc rõ ràng, che dấu được dữ liệu bên trong, giúp lập trình viên đơn giản hóa độ phức tạp của bài toán.

Lập trình hướng đối tượng là một phương pháp lập trình mới nhằm làm cho chương trình trở nên linh hoạt, tin cậy và dễ phát triển, dễ bảo trì và nâng cấp.

Sự trừu tượng hóa dữ liệu là phương pháp biểu diễn dữ liệu giúp người sử dụng có thể thao tác trên dữ liệu một cách dễ dàng mà không cần quan tâm đến các chi tiết của dữ liệu.

3.2 Đối tượng

Mỗi phần tử trong thế giới thực xung quanh chúng ta là một đối tượng, nó được hiểu như là những thực thể.

Đối tượng là sự gắn kết giữa các thành phần thông tin mô tả đối tượng (dữ liệu của đối tượng gọi là thuộc tính) và các hành vi của đối tượng (gọi là các phương thức) thao tác trên các dữ liệu này.

Ví dụ: Đối tượng Sinh viên có các thuộc tính như MSSV, Họ tên, Địa chỉ,và các phương thức tính điểm trung bình, ...

3.3 Lớp đối tượng

Lớp (class) là một khái niệm trong lập trình hướng đối tượng mô tả cho chúng những thực thể có chung tính chất và hành vi. Lớp định nghĩa những thuộc tính và hành vi được dùng cho những đối tượng của lớp đó.

Kết quả của sự trừu tượng hóa các đối tượng có: Cùng loại, cùng các thông tin mô tả về đối tượng thì được gọi là Lớp đối tượng.

Ví dụ: Từ những đối tượng xe ô tô cụ thể ta có thể trừu tượng hóa thành lớp xe Oto

3.4 Các đặc trưng cơ bản

Các thành phần cơ bản của lớp: Biến thành viên lưu trữ thông tin mô tả về đối tượng

Thuộc tính và phương thức dùng để cập nhật, tính toán, cung cấp và xử lý thông tin của đối tượng. Ví dụ thuộc tính MaXe, SoXe..., phương thức Lai, DungLai, NoMay,...

Tính trừu tượng (Abstraction)

Tính trừu tượng là một trong những khái niệm quan trọng trên nền tảng .NET. Muốn tạo ra các lớp cơ sở (base class). Để cài đặt lớp trừu tượng trong C# sử dụng từ khóa abstract. Một lớp trừu tượng không khởi tạo được đối tượng của lớp này, nhưng cho phép thừa kế để tạo ra lớp con.

Tính đóng gói (Encapsulation)

Tính đóng gói là kỹ thuật ràng buộc dữ liệu và phương thức thao tác trên dữ liệu đó vào trong lớp để kiểm soát. Làm tăng tính trùu tượng của dữ liệu. Lớp đối tượng chỉ cung cấp một số phương thức giao tiếp với môi trường bên ngoài, che dấu cài đặt bên trong lớp.

Tính kế thừa (Inheritance)

Tính kế thừa là quá trình định nghĩa một lớp đối tượng (Lớp dẫn xuất) dựa trên lớp khác đã định nghĩa gọi là lớp cơ sở nhằm tận dụng mã của lớp đó. Lớp mới chỉ bổ sung các thành phần riêng của chính nó hoặc định nghĩa lại các hàm của lớp cơ sở.

Tính đa hình (Polymorphism)

Tính đa hình là ý tưởng sử dụng giao diện chung cho nhiều phương thức khác nhau. Phương thức cụ thể sẽ được xác định lúc thực thi chương trình, tùy thuộc vào đối tượng đang thực thi giao diện đó. Làm giảm đáng kể độ phức tạp của chương trình.

3.5 Định nghĩa lớp và đối tượng

Để định nghĩa một kiểu dữ liệu mới hay một lớp đầu tiên phải khai báo rồi sau đó mới định nghĩa các thuộc tính và phương thức của kiểu dữ liệu đó. Khai báo một lớp bằng cách sử dụng từ khóa class.

Cú pháp chung

[mức độ truy cập] class Tenlop [:lớp cơ sở]

{

Khai báo các thành phần dữ liệu

Định nghĩa các phương thức, thuộc tính của lớp

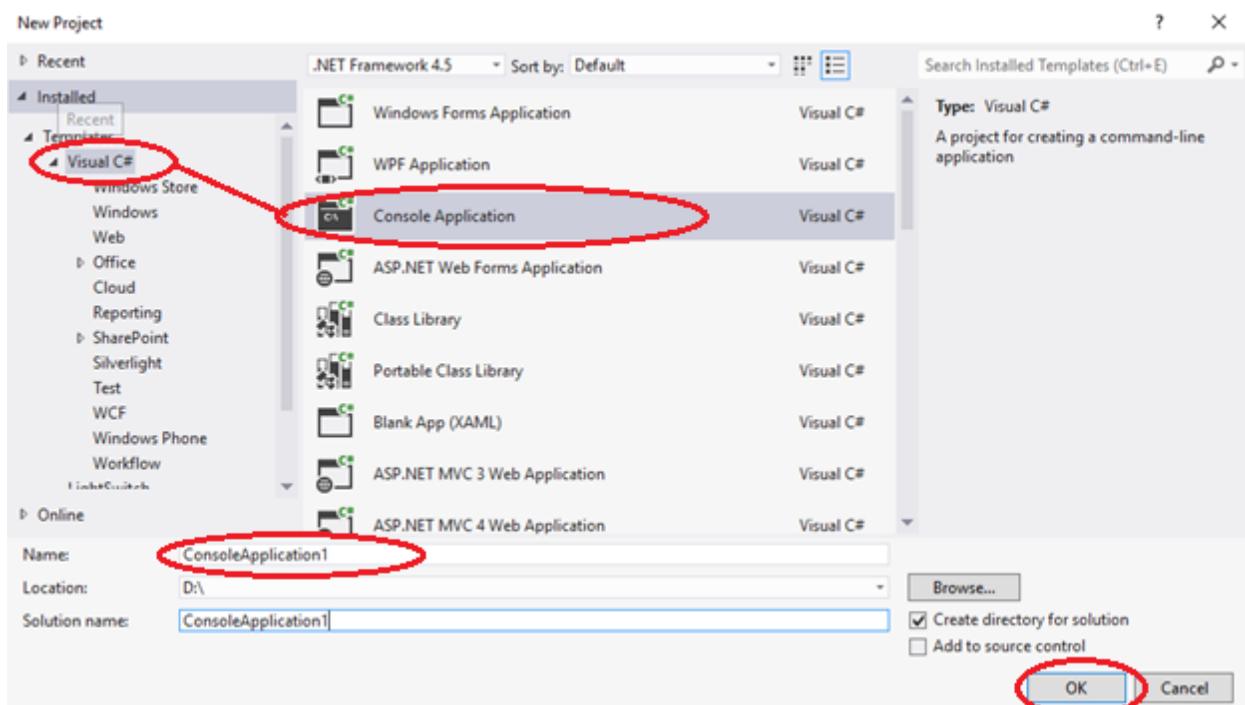
}

| Từ khóa | Mô tả |
|-----------|--|
| public | Không giới hạn, những thành viên được đánh dấu public có thể được dùng bởi bất kỳ các phương thức của lớp bao gồm những lớp khác |
| private | Thành viên trong một lớp A được đánh dấu private thì chỉ được truy cập bởi các phương thức của lớp A |
| protected | Thành viên trong lớp A được đánh dấu là protected thì chỉ được các phương thức bên trong lớp A và những phương thức dẫn xuất từ lớp A truy cập |

| | |
|--------------------|---|
| internal | Thành viên trong lớp A được đánh là internal thì được truy cập bởi những phương thức của bất cứ lớp nào trong cùng khối hợp ngữ với A |
| protected internal | Thành viên trong lớp A được đánh dấu là protected internal được truy cập bởi các phương thức của lớp A, các phương thức của lớp dẫn xuất của A, và bất cứ lớp nào trong cùng khối hợp ngữ của A |

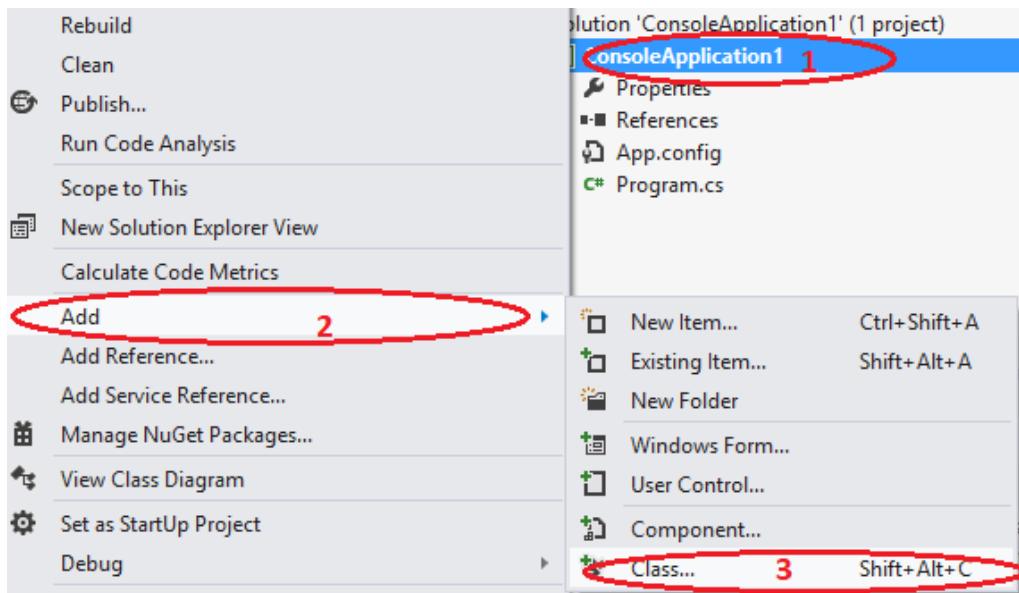
3.6 Xây dựng lớp trong C#

Khởi tạo chương trình Visual Studio chọn C# chọn menu File → New → Project



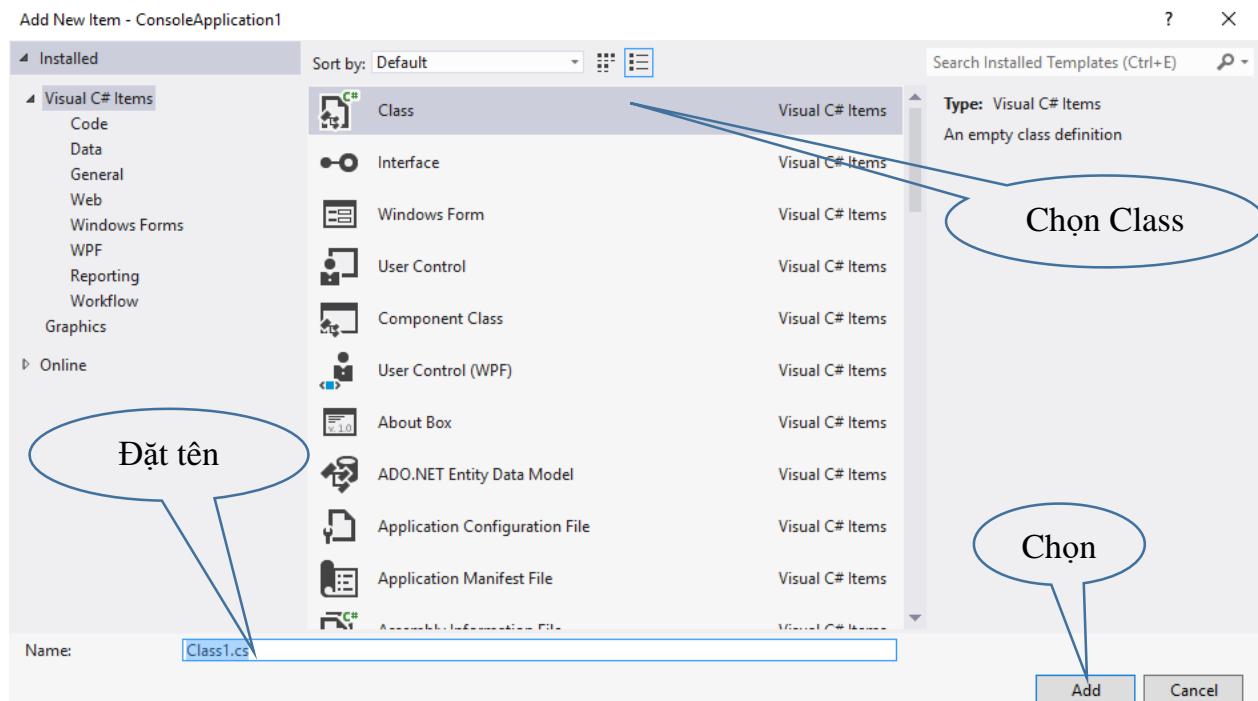
Hình 3.3 Tạo project

Chọn Right click vào project vừa tạo và thêm vào các lớp cho project



Hình 3.4 Thêm vào class cho project

Đặt tên cho class và chọn Add



Hình 3.5 Đặt tên cho class

3.7 Bài tập áp dụng

Bài tập áp dụng 1

Xây dựng chương trình quản lý nhân viên, xây dựng lớp, thuộc tính và phương thức thực hiện.

Xây dựng class Nhân viên – Tạo class clsNhanvien với các thuộc tính và các phương thức như sau: Thông tin về mã nhân viên, họ tên nhân viên, giới tính, hệ số lương,...

Xây dựng class clsNhanvien

```
class clsNhanVien
{
    // khai bao cac thuoc tinh va properties
    private string _MaNV;
    private string _HoNV;
    private string _TenNV;
    private double _Hesoluong;
    private bool _Giotinh;
    // properties
    public bool Giotinh...
    // properties
    public double Hesoluong...
    // properties
    public string MaNV...
    // properties
    public string HoNV...
    // properties
    public string TenNV...
    ...
}
```

Xây dựng các phương thức

```
// khai bao constructor
// constructor mac dinh
public clsNhanVien()
{
    MaNV = "0000";
    HoNV = "";
    TenNV = "";
}
//construtor co tham so
public clsNhanVien(string ma, string ho, string ten)
{
    this.MaNV = ma;
    this.HoNV = ho;
    this.TenNV = ten;
}
// phuong thuc tinh luong
public double TinhLuong()
{
    return Hesoluong * 1390000;
}
// phuong thuc xuat thong tin
public void XuatThongTin()
{
    string chuoi = string.Format("MaNV: {0}\nHoten: {1} {2}\nTien luong: {3}",
        MaNV, HoNV, TenNV, TinhLuong());
    Console.WriteLine(chuoi);
}
public void XuatThongTin(string ma, string ho, string ten)
{
    string chuoi = string.Format("\nMaNV: {0}\nHoten: {1} {2}\nTien luong: {3}", ma, ho, ten);
    Console.WriteLine(chuoi);
}
```

Xây dựng lớp chứa hàm chính để thực hiện chương trình

```
class Program
{
    static void Main(string[] args)
    {
        // khai bao khoi tao lop nhan vien voi gia tri mac dinh
        clsNhanVien nv1 = new clsNhanVien();
        // gan gia tri cho cac thuoc tinh
        nv1.MaNV = "NV001";
        nv1.HoNV = "Nguyen Van";
        nv1.TenNV = "Troi";
        nv1.Giotinh = true;
        nv1.Hesoluong = 3;
        // xuat thong tin nhan vien
        nv1.XuatThongTin();

        // khai bao va khoi tao nhan vien voi gia tri khai tao constructor co tham so
        clsNhanVien nv2 = new clsNhanVien("NV02", "Vo Van", "Ngan");
        nv2.Hesoluong = 1.5;
        // xuat thong tin nhan vien
        nv2.XuatThongTin();

        Console.ReadKey();
    }
}
```

Kết quả chương trình

```
MaNV: NV001
Hoten: Nguyen Van Troi
Tien luong: 4170000
MaNV: NV02
Hoten: Vo Van Ngan
Tien luong: 2085000
```

Sinh viên xây dựng tiếp các phương thức nhập thông tin cho nhân viên và gọi thực thi trong chương trình chính.

Bài tập áp dụng 2

Xây dựng lớp Xe có các thuộc tính: tốc độ, biển số, hãng sản xuất, xây dựng lớp xe khách có thuộc tính số khách và xe tải có thuộc tính trọng tải kế thừa từ lớp Xe.

Xây dựng class Xe

```
class Xe
{
    // cac bien thanh phan trong lop xe
    // protected de cac lop con truy cap duoc
    protected int tocDo;
    protected string bienSo;
    protected string hangSX;
    // khai bao constructor co tham so
    public Xe(int td, string bs, string hsx)
    {
        tocDo = td;
        bienSo = bs;
        hangSX = hsx;
    }
    // phuong thuc xuat thong tin
    public void Xuat()
    {
        Console.WriteLine("Xe: {0}\n Bien so: {1}\n Toc do: {2} kmh",
                           hangSX, bienSo, tocDo);
    }
}
```

Xây dựng class XeKhach kế thừa từ lớp Xe

```
class XeKhach:Xe
{
    private int soKhach;
    // khai bao contructor
    public XeKhach(int td, string bs, string hsx, int shk) : base(td, bs, hsx)
    {
        soKhach = shk;
    }
    // phuong thuc xuat
    public void Xuat()
    {
        base.Xuat();
        Console.WriteLine("so cho ngoi : {0} khach", soKhach);
    }
}
```

Xây dựng class XeTai kế thừa từ lớp Xe

```
class XeTai:Xe
{
    int trongTai;
    // constructor
    public XeTai(int td, string bs, string hsx, int trgtai)
        : base(td, bs, hsx)
    {
        trongTai = trgtai;
    }
    // phuong thuc xuat
    public void Xuat()
    {
        base.Xuat();
        Console.WriteLine("Trong tai {0} tan ", trongTai);
    }
}
```

Xây dựng hàm chính

```
class Program
{
    static void Main(string[] args)
    {

        // khai bao va khoi tao lop Xe khach
        XeKhach xk = new XeKhach(130, "51A-4532", "HuynhDai", 45);
        // khai bao va khoi tao lop Xe tai
        XeTai xt = new XeTai(50, "55B- 7896", "Merscedes", 100);
        // goi phuong thuc Xuat cua xe khach
        xk.Xuat();
        // goi phuong thuc Xuat cua xe tai
        xt.Xuat();
        Console.ReadKey();
    }
}
```

Kết quả thực thi chương trình

```
Xe: HuynhDai
    Bien so: 51A-4532
    Toc do: 130 kmh
    so cho ngoi : 45 khach
Xe: Merscedes
    Bien so: 55B- 7896
    Toc do: 50 kmh
    Trong tai 100 tan
```

Bài tập áp dụng 3

Một điểm dịch vụ cần quản lý các thông tin cho thuê xe đạp và xe máy. Với xe đạp cần lưu họ tên người thuê, số giờ thuê.

Tiền thuê xe đạp được tính như sau: 10000 cho giờ đầu tiên, 8000 cho mỗi giờ tiếp theo.

Với mỗi xe máy cần lưu họ tên người thuê, số giờ thuê, loại xe (100 phân khối, 250 phân khối), biển số.

Tiền thuê xe máy được tính như sau: Đối với giờ đầu tiên, loại xe 100 phân khối tính 15000; loại xe 250 phân khối tính 20000. Đối với những giờ tiếp theo tính 10000 cho cả hai loại xe máy.

Viết chương trình xây dựng các lớp cần thiết sau đó nhập danh sách các thông tin thuê xe đạp và xe máy, sau đó xuất ra các thông tin sau:

1. Xuất tất cả các thông tin thuê xe (cả số tiền thuê tương ứng).
2. Tính tổng số tiền cho thuê xe đạp và xe máy.
3. Xuất tất cả các thông tin liên quan đến việc thuê xe đạp.
4. Tính tổng số tiền cho thuê xe máy loại 250 phân khối.

Xây dựng lớp Xe có các phương thức ảo

```
class Xe
{
    // khai báo biến thành viên
    protected string _Hoten;
    protected int _GioThue;
    //
    public virtual int TienThue
    {
        get { return 0; }
    }
    // Xây dựng phương thức ảo
    public virtual void Xuat()
    {
    }
    // Xây dựng phương thức ảo
    public virtual string ID()
    {
        return "X";
    }
}
```

Xây dựng lớp XeDap kế thừa lớp Xe

```
class XeDap:Xe // ke thua tu lop xe
{
    // khai bao constructor
    public XeDap(string ht, int gt)
    {
        _Hoten = ht;
        _GioThue = gt;
    }

    public override string ID()
    {
        return "XeDap";
    }
    // dinh nghia lai tien thue
    public override int TienThue
    {
        get
        {
            int iTinhTien = 10000;
            if (_GioThue > 0)
                iTinhTien = iTinhTien + 8000 * (_GioThue - 1);
            return iTinhTien;
        }
    }
    // dinh nghia lai phuong thuc Xuat cho xe dep
    public override void Xuat()
    {
        Console.WriteLine("XeDap: Ho Ten: " + _Hoten + "\tGio Thue:"
            + _GioThue + "\tTien Thue:" + TienThue);
    }
}
```

Xây dựng lớp XeMay kế thừa lớp Xe

```
class XeMay:Xe // xe may ke thu tu lop xe
{
    // khai bao bien thanh vien
    string Bienso;
    string Loaixe;
    // khai bao constructor co tham so
    public XeMay(string ht, int gt, string lx, string bs)
    {
        _Hoten = ht;
        _GioThue = gt;
        Loaixe= lx;
        Bienso = bs;

    }
    // dinh nghia lai ID
    public override string ID()
    {
        if (Loaixe == "100")
            return "XeMay_100";
        else
            return "XeMay_250";
    }
}
```

```
// dinh nghia lai tinh tien thue cho xe may
public override int TienThue
{
    get
    {
        int iTien=0;
        if(LoaiXe=="100")
            iTien = 15000;
        else
            iTien = 20000;
        if(_GioThue > 0)
            iTien = iTien + 10000 * (_GioThue-1);
        return iTien;
    }
}
// dinh nghia lai phuong thuc Xuat
public override void Xuat()
{
    Console.WriteLine("Xe may: Ho Ten:" + _Hoten + "\tGio Thue:" + _GioThue +
                      "\tLoai xe:" + LoaiXe + "\tBien So:" + BienSo + "\tTien Thue:" + TienThue);
}
}
```

Xây dựng lớp CuaHangXe

```
class CuaHangXe
{
    public int iN;
    Xe[] XeThue; // khai bao mang xe cho thue
    // khai bao constructor co tham so
    public CuaHangXe(int iSize)
    {
        iN = iSize;
        XeThue = new Xe[iSize];
    }
    // xây dựng phương thức menu
    public int Menu()
    {
        int iChon;
        do
        {
            Console.Write("\nBan chon nhap xe nao:\n");
            Console.WriteLine("1.Ban chon nhap xe dap: ");
            Console.WriteLine("2.Ban chon nhap xe may: ");
            Console.Write("Ban chon so: ");
            iChon = int.Parse(Console.ReadLine());

        } while ((iChon!= 1) && (iChon != 2));
        return iChon;
    }
}
```

```

// xây dựng hàm nhập danh sách xe cho thuê
public void NhapDSXeThue()
{
    int iChon;
    for (int i = 0; i < iN; i++)
    {
        iChon = Menu();
        if (iChon == 1) // nhap thong tin xe dap
        {
            Console.WriteLine("\n Ban nhap thong tin xe dap");
            Console.Write("\n Nhập vào tên người thuê: ");
            string hoTen = Console.ReadLine();
            Console.Write("\n Số giờ thuê: ");
            int gt = int.Parse(Console.ReadLine());
            XeThue[i] = new XeDap(hoTen, gt);
        }
        else // nhap thong tin xe may
        {
            Console.WriteLine("\n Ban nhap thong tin xe may");
            Console.Write("\n Nhập vào tên người thuê: ");
            string hoTen = Console.ReadLine();
            Console.Write("\n Số giờ thuê: ");
            int iGioThue = int.Parse(Console.ReadLine());
            Console.Write("\n Biển số xe: ");
            string bienSo = Console.ReadLine();
            Console.WriteLine("\n Loại xe phân khối (100 - 250): ");
            string loaiXe = Console.ReadLine();
            XeThue[i] = new XeMay(hoTen, iGioThue, loaiXe, bienSo);
        }
    }
}

// Xuất danh sách xe thuê
public void XuatDSxeThue()
{
    for (int i = 0; i < iN; i++)
    {
        XeThue[i].Xuat();
    }
}

// tính tổng tiền thuê
public long TongTienChoThue()
{
    long lTongTien = 0;
    for (int i = 0; i < iN; i++)
    {
        lTongTien = lTongTien + XeThue[i].TienThue;
    }
    return lTongTien;
}

// xuất danh sách xe đạp
public void XuatXeDap()
{
    for (int i = 0; i < iN; i++)
    {
        if (XeThue[i].ID() == "XeDap")// kiểm tra xe đạp
        {
            XeThue[i].Xuat(); // gọi xuất thông tin xe đạp
        }
    }
}

```

```
// tinh tong tien xe may 250
public long TongTienXemay250()
{
    long lTongTien = 0;
    for (int i = 0; i < iN; i++)
    {
        if (XeThue[i].ID() == "XeMay_250") // kiem tra xe may 250
        {
            lTongTien = lTongTien + XeThue[i].TienThue; // tinh tien cong lai
        }
    }
    return lTongTien;
}
```

Xây dựng hàm chính thực thi ứng dụng

```
class Program
{
    static void Main(string[] args)
    {
        CuaHangXe ch = new CuaHangXe(1);
        ch.NhapDSXeThue();
        Console.WriteLine("\n Danh sach xe cho thue: ");
        ch.XuatDSxeThue();
        Console.WriteLine("\n Tong tien thue xe: {0}", ch.TongTienChoThue());
        Console.WriteLine("Thong tin xe dap: ");
        ch.XuatXeDap();
        Console.WriteLine("\n Tong tien thue xe may 250:{0} ", ch.TongTienXemay250());
        Console.ReadKey();
    }
}
```

Kết quả chương trình

```
Ban chon nhap xe nao:
1.Ban chon nhap xe dap:
2.Ban chon nhap xe may:
Ban chon so: 1

Ban nhap thong tin xe dap

Nhap vao ten nguoi thue: Nguyen van an

So gio thue: 3

Danh sach xe cho thue:
XeDap: Ho Ten: Nguyen van an    Gio Thue:3      Tien Thue:26000

Tong tien thue xe: 26000
Thong tin xe dap:
XeDap: Ho Ten: Nguyen van an    Gio Thue:3      Tien Thue:26000

Tong tien thue xe may_250:  0
```

3.8 Bài tập

HỌC PHẦN LẬP TRÌNH ỨNG DỤNG BÀI TẬP THỰC HÀNH SỐ 3

I. Thông tin chung:

- Mã số bài tập : BT-LTUD - 03
- Hình thức nộp bài : Nộp qua Moodle môn học
- Thời hạn nộp bài : ... / ... /
- Nội dung: Chương 3: Lập trình hướng đối tượng

Chuẩn đầu ra cần đạt:

L.O.2 Vận dụng lập trình hướng đối tượng để xây dựng được các lớp

L.O.7 Rèn luyện các kỹ năng tìm kiếm thông tin để tự giải quyết vấn đề

L.O.4 Viết code đúng chuẩn

Bài tập 1: Viết lớp giải phương trình bậc hai.

Lớp này có các thuộc tính a, b, c và nghiệm x1, x2. Hãy xây dựng theo hướng đối tượng lớp trên. Lớp cho phép bên ngoài xem được các nghiệm của phương trình và cho phép thiết lập giá trị hoặc lấy giá trị a, b, c.

Bài tập 2: Viết chương trình quản lý kho.

Hãy lưu trữ mã số, tên hàng, giá và số lượng đang có của mỗi món hàng trong một lớp. Nhập chi tiết của N (N nhập từ bàn phím) món hàng hiển thị tên từng món hàng và tổng giá trị của nó.

Bài tập 3: Viết một chương trình để lưu trữ các sinh viên gồm: mã sinh viên, họ và tên và điểm trung bình của N (N nhập từ bàn phím) sinh viên.

Hãy sắp xếp danh sách sinh viên này theo thứ tự điểm trung bình giảm dần. Hiển thị 3 sinh viên có điểm trung bình cao nhất

Bài tập 4: Tạo một lớp CD gồm: CDName, CDTypre và CDPrice và các phương thức cần thiết khác.

Viết một ứng dụng có dạng menu như sau:



Thêm CD: Cho phép thêm một đĩa CD vào danh mục các đĩa CD hiện có. Hãy cung cấp khả năng quản lý khoảng 1000 đĩa CD. Cần có sự kiểm soát nếu số lượng đĩa CD được nhập vượt quá 1000.

Search CD: Nhập vào tên của đĩa CD, nếu không tìm thấy thì báo lỗi, nếu tìm thấy thì in các thông tin liên quan đến đĩa CD đó.

| CD No. | CD Name | CD Type | CD Price |
|--------|---------------|----------|-----------|
| 1 | Ngay khong em | Ca nhac | 70K(VND) |
| 2 | Thoi xa vang | Phim | 320K(VND) |
| 3 | De che | Tro choi | 6K(VND) |

Display catalog: Hiển thị tất cả các đĩa CD hiện có trong danh mục, danh mục đĩa CD được hiển thị ở dạng bảng, có chứa cột tiêu đề.

Bài tập nâng cao

Bài tập 1: Cho thiết kế lớp Employee (nhân viên) như sau:

Các thành phần dữ liệu:

- id: Định danh, kiểu int. Định danh này được sinh tự động và tăng dần bắt đầu từ 1.
- name: Họ tên nhân viên, kiểu String.
- yearOfBirth: Năm sinh nhân viên, kiểu int.
- salaryLevel: Bậc lương, kiểu double.
- basicSalary: Lương cơ bản, kiểu double. (Chú ý lương cơ bản là thuộc tính được sử dụng chung cho mọi đối tượng của lớp Employee).

Các phương thức:

- GetId(): Trả lại định danh của nhân viên.
- GetName(): Trả lại tên của nhân viên.
- GetYearOfBirth(): Trả lại năm sinh của nhân viên.
- GetIncome(): Trả lại thu nhập của nhân viên. Thu nhập được tính bằng bậc lương nhân lương cơ bản ($\text{salaryLevel} * \text{basicSalary}$).
- Input(): Nhập thông tin nhân viên.
- Display(): Hiển thị thông tin về nhân viên. Bao gồm các thông tin: định danh, tên, năm sinh, lương cơ bản, thu nhập.
- SetSalaryLevel(): Thiết lập bậc lương cho nhân viên.
- SetBasicSalary(): Thiết lập lương cơ bản.

Hãy viết chương trình cài đặt lớp Employee và lớp sử dụng Employee.

Bài tập 2: Xây dựng lớp có tên là TienDien với các thông tin bao gồm:

+ Dữ liệu:

- Họ tên chủ hộ
- Địa chỉ
- Số công tơ tháng trước
- Số công tơ tháng này

+ Phương thức

- Phương thức thiết lập không tham số và 4 tham số
- Phương thức nhập dữ liệu
- Phương thức hiển thị dữ liệu
- Thuộc tính tính số công tơ điện đã dùng(=Số công tơ tháng này- Số công tơ tháng trước)

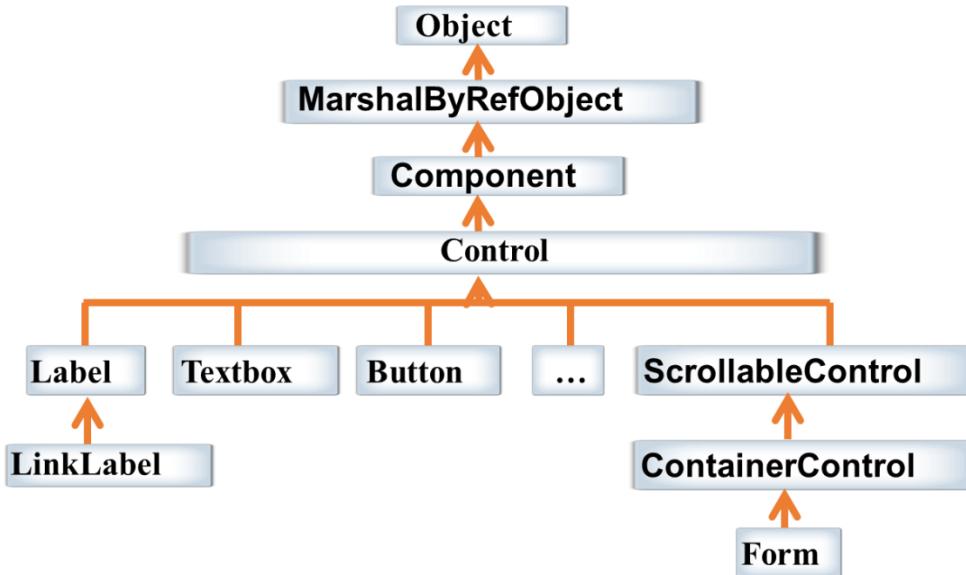
Phương thức tính tiền điện được tính theo công thức: Số điện đã dùng*1240
Sau đó xây dựng lớp TienDienMoi bằng việc kế thừa lớp TienDien để tính tiền
điện theo một quy định mới. Việc tính tiền điện lúc này căn cứ vào định mức quy
định. Nếu trong định mức là 1240, ngoài định mức là 1600

CHƯƠNG 4. LẬP TRÌNH WINDOWS FORMS

4.1 Giới thiệu thành phần Windows Forms

Lập trình Windows Forms là cách cơ bản để cung cấp các thành phần giao diện (GUI components) cho môi trường .NET Framework. Windows Forms được xây dựng trên thư viện API (Application Programming Interface). Windows Forms cơ bản bao gồm: Một Form là khung hiển thị thông tin người dùng và các control được đặt trong Form và được lập trình để đáp ứng các sự kiện.

4.2 Các thành phần của Windows Forms



Hình 4.1 Các thành phần của Windows Forms

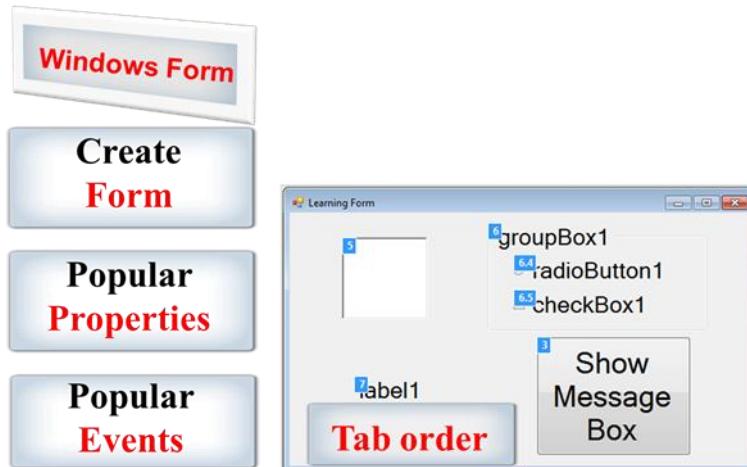
Các thành phần ứng dụng Windows Forms

Một số thuộc tính chung của các control trong Windows Forms

| Thuộc tính | Điễn giải |
|------------|---|
| Text | Mô tả text xuất hiện trên control |
| Focus | Phương thức chuyển Focus của control |
| TabIndex | Thứ tự của Control nhận Focus, mặc định do .Net thiết lập |
| Enable | Thiết lập trạng thái truy cập của control |
| Visible | Ẩn control trên Form, có thể dùng phương thức Hide |
| Anchor | Neo giữ control ở vị trí xác định, cho phép control di chuyển theo vị trí |

| | |
|-----------------|------------------------------------|
| Size | Xác nhận kích thước của control |
| Backcolor | Màu nền của control |
| BackGroundImage | Ảnh nền của control |
| ForeColor | Màu hiển thị text trên Form |
| Enable | Xác định control trạng thái Enable |
| Focused | Xác định control khi nhận Focus |
| Font | Font hiển thị text trên control |

Windows Forms là cửa sổ một màn hình ứng dụng, control trên giao diện là cửa sổ giao tiếp giữa người sử dụng và máy tính.

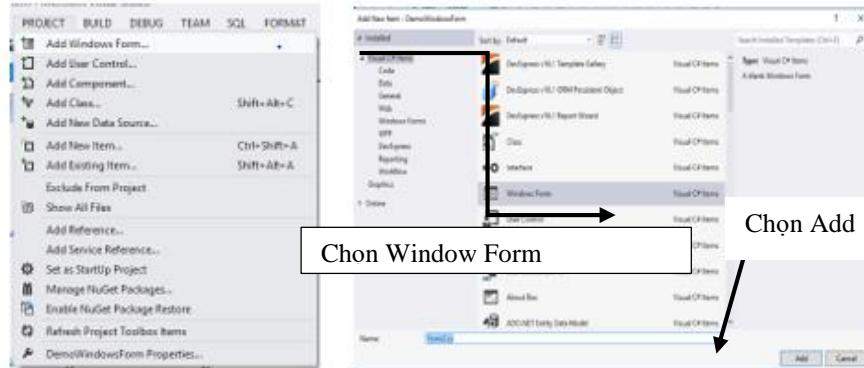


4.3 Những đặc điểm cơ bản của Windows Forms

Một Windows Forms giống như một class, tất cả các Forms đều thừa kế từ lớp **System.Windows.Forms.Form** khi sử dụng cần phải khai báo và khởi tạo để sử dụng.

Tạo một Forms mới

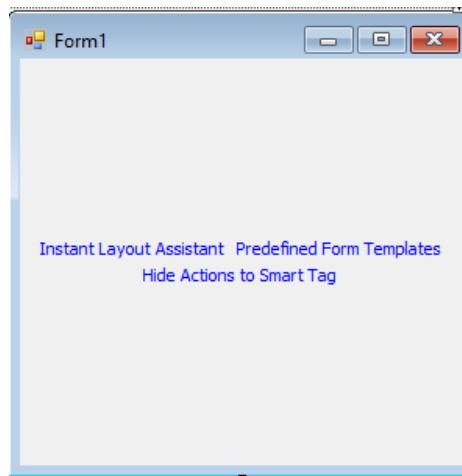
Khi tạo một Project thì 1 Forms mới sẽ được tự động được thêm vào với tên Forms mặc định là Form1, để thêm mới một Forms khác vào project bằng cách vào menu Project -> Add Windows Form -> chọn Windows Form -> đặt tên cho Form và chọn Add



Hình 4.2 Tạo một ứng dụng Windows Forms

Trong thiết kế giao diện Windows Forms có 2 thành phần: cửa sổ Design và cửa sổ code để viết code cho các sự kiện xử lý trên Forms.

Cửa sổ Windows Forms Design: Dùng để thiết kế bằng cách kéo các control để lên Forms.



Hình 4.3 Cửa sổ Form Design

Cửa sổ Code: Dùng để viết code cho các xử lý trên Form.

```

1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10
11 namespace DemoWindowsForm
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19
20         private void Form1_Load(object sender, EventArgs e)
21         {
22             // code
23         }
24     }
25 }
26

```

Một số thuộc tính của Forms

| Thuộc tính | Giá trị | Diễn giải |
|--------------|--------------|---|
| ControlBox | True/ False | Có hoặc không 3 nút min, max, close trên tiêu đề góc phải |
| MaximizeBox | True / False | Có hay không nút Max |
| MinimizeBox | True / False | Có hay không nút Min |
| TopMost | True / False | Form này có luôn nằm lên trên hết các Forms khác hay không |
| Caption | Text | Text nằm trên thanh tiêu đề của Form |
| Name | Text | Tên của Form |
| AcceptButton | Event | Chọn sự kiện của control trên Form khi click tương ứng với phím Enter |
| CancelButton | Event | Chọn sự kiện của control trên Form khi click tương ứng với phím Esc |

| | | |
|-----------------|------------------------|--|
| FormBorderStyle | None | Không có đường biên |
| | FixedSingle | Tương tự FixedDialog |
| | Fixed3D | Trong giống 3 chiều |
| | FixedDialog | Như hộp thoại |
| | Sizeable | Mặc định |
| | FixedToolWindows | Thanh caption nhỏ và không có nút Close |
| | SizeableToolWindow | Giống FixedToolWindow nhưng có đường biên mỏng |
| StartPosition | Manual | Hiển thị Form ở vị trí theo giá trị của Property Location của Form |
| | CenterScreen | Hiển thị ở ngay giữa màn hình |
| | CenterParent | Hiển thị Form ở ngay giữa Form chính của nó |
| | WindowsDefaultLocation | Hiển thị ở vị trí default của cửa sổ |
| | WindowsDefaultBounds | Hiển thị Form ở vị trí default của cửa sổ, với kích thước default của cửa sổ |
| AutoSize | True /False | Cho phép kéo dãn Form |
| CauseValidation | True/ False | Cho phép các control con của nó có thể phát sinh sự kiện Validation. Dùng để kiểm tra dữ liệu hợp lệ |
| Icon | Icon | Thiết lập Icon cho Form |

4.3.1 Một số phương thức của Form

| Phương thức | Điễn giải |
|-------------|--|
| Show() | Sau khi Dialog hiện lên, người dùng có thể chọn Focus vào control khác |

| | |
|--------------|---|
| ShowDialog() | Sau khi Dialog hiển thị lên người dùng không thể Focus vào các control khác. Kiểu dữ liệu trả về là DialogResult. |
| Close() | Đóng Dialog |

4.3.2 Một số sự kiện của Form (Form Event)

| Sự kiện | Diễn giải |
|-------------------------|---|
| Load() | Sự kiện được kích hoạt khi Form đã Load xong Control. Sự kiện này được bắt khi người dùng muốn khởi tạo biến hoặc giá trị trong Form |
| Paint() | Sự kiện này được gọi khi Form thực hiện vẽ lại |
| KeyPress, KeyDown,... | Các sự kiện về phím |
| MouseDown, MouseMove... | Các sự kiện về chuột |
| FormClosed | Sự kiện khi đóng Form |
| FormClosing | Sự kiện phát sinh trước khi đóng Form |

Các sự kiện của Form: Trong lập trình visual điều quan trọng là xử lý các sự kiện, khi lập trình thường chỉ thực hiện các thao tác kéo thả là có thể tạo được một giao diện hoàn chỉnh. Để giao diện đó hoạt động theo đúng các yêu cầu của chúng ta thì phải lập trình cho các sự kiện của một hay nhiều control trên Form.

Form có rất nhiều sự kiện, ở đây giới thiệu một số sự kiện quan trọng của Form:

Sự kiện Form_Load

```
private void Form1_Load(object sender, EventArgs e)
{
    // viết code xử lý khi form được gọi
}
```

Sự kiện khi Form đóng:

```
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    // xử lý khi form đóng
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    // xử lý trước khi form đóng
}
```

Sự kiện khi click trên Form:

```
private void Form1_Click(object sender, EventArgs e)
{
    // xử lý khi click chuột trên form
}
```

Trong một ứng dụng có thể sử dụng nhiều Form, để thực hiện Form nào khi chạy chương trình (startup) ta chọn cho Form đó được thực hiện trong hàm chính trong tập tin Program.

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Form1()); // chọn form1 để chạy
}
```

Ví dụ: Xử lý cho Form sử dụng sự kiện khi click trên Form thì Form sẽ thay đổi kích thước được tăng lên 50 pixel.

```
private void Form1_Click(object sender, EventArgs e)
{
    // xử lý khi click chuột trên form
    MessageBox.Show("Bạn mới click chuột lên trên form!\n" +
        "Click Ok và form sẽ lớn hơn 50 pixels.", "Thông báo");
    this.Height = this.Height + 50;
    this.Width = this.Width + 50;
}
```

4.4 Các điều khiển chuẩn

Button Control

Là một đối tượng nút nhấn, cho phép người dùng click chuột vào nó và nó sẽ thực thi một hay nhiều đoạn lệnh mà người lập trình chỉ định qua các event mà nó nhận được.

Các thuộc tính của Button:

| | |
|----------------|-----------|
| Tên thuộc tính | Điễn giải |
|----------------|-----------|

| | |
|-----------------|--|
| Name | Tên của button |
| Text | Chữ hiển thị trên Button |
| Font | Chọn Font chữ cho Button |
| Backcolor | Màu nền cho Button |
| ForeColor | Màu cho chữ |
| TextAlign | Canh lè chữ trên Button |
| Image | Hình sẽ làm nền cho Button (giống như bạn chọn Center khi set hình Background cho Window). |
| BackGroundImage | Hình chỉ định sẽ làm nền cho Button |
| Anchor | Chỉ định control được neo lại theo Form |

Các sự kiện của Button: Trong các sự kiện của Button thì có sự kiện Click là quan trọng nhất, sự kiện mặc định của Button là click khi ta nhấn double trên control thì visual sẽ chuyển sang của số code và tạo cho chúng ta một sự kiện click và chúng ta viết code để xử lý.

```
private void button1_Click(object sender, EventArgs e)
{
    //code xử lý khi click chuột lên control
}
```

Label Control

Dùng để trình bày một chuỗi văn bản thông thường nhằm mục đích thêm thông tin cho đối tượng khác. Ta cũng có thể dùng nhãn để làm công cụ đưa kết quả ra màn hình dưới dạng một chuỗi.

Các thuộc tính cơ bản của nhãn:

| | |
|----------------|-----------------------------|
| Tên thuộc tính | Điễn giải |
| Name | Tên của control |
| Borderstyle | Kiểu đường viền cho control |
| Text | Hiển thị text trên Form |
| Font | Font hiển thị trên control |
| TextAlign | Canh lè text |

Có 2 loại Label và LinkLabel thường dùng.

TextBox Control

Là đối tượng dùng để nhập dữ liệu vào. Có 2 dạng TextBox và MaskedTextBox.

Các thuộc tính cơ bản của TextBox

| Tên thuộc tính | Điễn giải |
|----------------|---|
| MaxLength | Chiều dài tối đa khi nhập dữ liệu |
| Multiline | Cho/ không cho phép nhập nhiều dòng trong Textbox |
| PasswordChar | Hiển thị ký hiệu trong ô nhập |
| ReadOnly | Cho hay không cho phép nhập dữ liệu vào |
| Name | Tên của control |
| Font | Font cho control |

Các sự kiện của TextBox

| Tên Event | Điễn giải |
|----------------------|--|
| TextChanged | Sự kiện phát sinh khi người dùng thay đổi nội dung text |
| Validating | Khi mất Focus thì sự kiện này phát sinh để kiểm tra dữ liệu hợp lệ |
| KeyPress, KeyDown... | Bắt sự kiện khi gõ phím |

MaskedTextBox có nhiều dạng sẵn cho chúng ta nhập dữ liệu như: Numeric, PhoneNumber, Date, Time...

CheckBox Control

Control thể hiện trạng thái chọn và không chọn tương ứng với giá trị Yes/No. Khi ta chọn khi CheckBox đang có Focus thì sẽ đổi trạng thái chọn hoặc không.

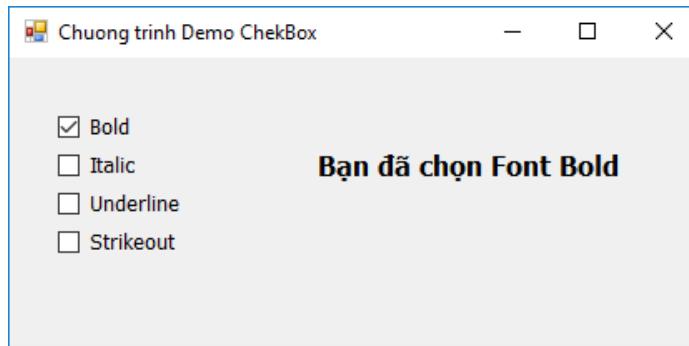
Các thuộc tính của CheckBox

| Thuộc tính | Điễn giải |
|------------|------------------------------|
| Checked | Chọn đánh dấu hay không chọn |
| CheckState | CheckBox đang được đánh dấu |
| | CheckBox không được chọn |

| | |
|------------|---|
| | CheckBox ở trạng thái không hoạt động |
| ThreeState | Cho phép CheckBox có 3 trạng thái: chọn, không chọn, không tác dụng |

Các sự kiện của CheckBox: CheckBox cũng có nhiều sự kiện nhưng chúng ta chỉ xét sự kiện quan trọng nhất là Click. Sự kiện click của CheckBox cũng giống như Button nhưng chúng ta phải kiểm tra trạng thái của CheckBox là True hay False.

Ví dụ sử dụng CheckBox: Thiết kế giao diện



Hình 4.4 Control CheckBox

Code cho các sự kiện:

```

19 // sự kiện khi CheckBox Bold thay đổi
20 private void chkFontBold_CheckedChanged(object sender, EventArgs e)
21 {
22     if (chkFontBold.Checked)
23     {
24         lblMsg.Font = new Font(lblMsg.Font, FontStyle.Bold);
25         lblMsg.Text = "Bạn đã chọn Font Bold";
26     }
27     else
28     {
29         lblMsg.Text = "Bạn đã bỏ chọn Font";
30         lblMsg.Font = new Font(lblMsg.Font, FontStyle.Regular);
31     }
32 }
```

```

33 // sự kiện khi CheckBox Italic thay đổi
34 private void chkFontItalic_CheckedChanged(object sender, EventArgs e)
35 {
36     if (chkFontItalic.Checked)
37     {
38         lblMsg.Font = new Font(lblMsg.Font, FontStyle.Italic);
39         lblMsg.Text = "Bạn đã chọn Font Italic";
40     }
41     else
42     {
43         lblMsg.Text = "Bạn đã bỏ chọn Font";
44         lblMsg.Font = new Font(lblMsg.Font, FontStyle.Regular);
45     }
46 }
```

Tương tự sinh viên viết tiếp cho các CheckBox còn lại.

RadioButton Control

Nút chọn cho thay đổi trạng thái Yes/No, nhưng với RadioButton chỉ cho chọn một trong nhiều RadioButton trong cùng một nhóm, RadioButton cũng có những thuộc tính giống như CheckBox chỉ khác CheckBox là chỉ chọn được 1 trong nhiều RadioButton.

Ví dụ sử dụng RadioButon để chọn màu cho Form.

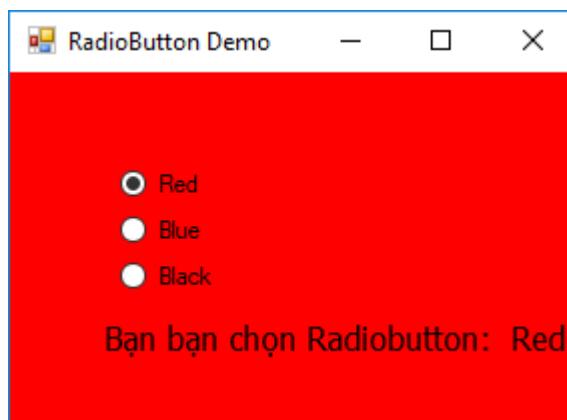


Hình 4.4 Control RadioButton

Code sự kiện Red chương trình

```
private void rdoRed_CheckedChanged(object sender, EventArgs e)
{
    lblColor.Text = "Bạn bạn chọn Radiobutton: " + rdoRed.Text;
    frmRadioButton.ActiveForm.BackColor = Color.Red;
}
```

Kết quả chương trình khi chọn RadioButton Red



Hình 4.5 Kết quả chương trình khi chọn Control RadioButton

ListBox Control

List cho phép tạo ra một danh sách các phần tử để lựa chọn, đối với List có số lượng lớn các phần tử thì xuất hiện thanh trượt để di chuyển cho ta thấy được các phần tử trong List.

Các thuộc tính của List

| Tên thuộc tính | Giá trị | Điễn giải |
|----------------|---------------|--|
| MultiColumn | True / False | Cho phép hiển thị một hay nhiều cột |
| SelectionMode | None | Không có phần tử nào được chọn |
| | One | Chỉ cho phép chọn một phần tử |
| | MultiSimple | Cho phép chọn nhiều phần tử cùng lúc |
| | MultiExtended | Giống như MultiSimple nhưng cho kết hợp dùng phím để chọn |
| Sort | True / False | Sắp xếp thứ tự trong List |
| Items | Collection | Một hộp thoại cho phép nhập giá trị của các phần tử trong List |

Một số phương thức của ListBox

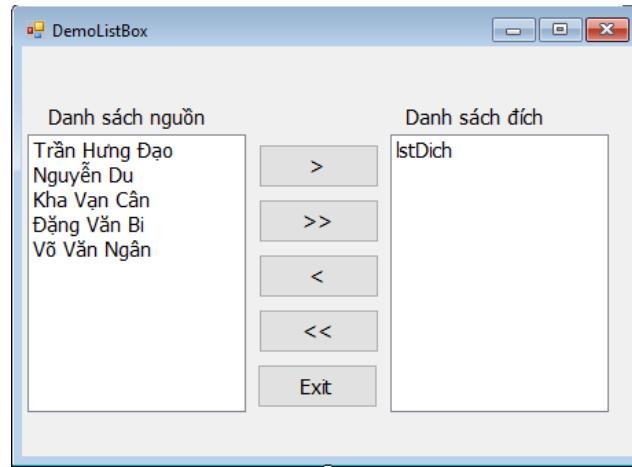
| Phương thức | Điễn giải |
|-------------|---|
| Add | Thêm phần tử vào List |
| AddRange | Thêm mảng phần tử vào List |
| Remove | Loại bỏ một phần tử |
| RemoveAt | Loại bỏ một phần tử tại vị trí trong List |
| Clear | Xóa hết các giá trị trong List |
| Count | Đếm số phần tử có trong List |

Các sự kiện của ListBox: Sự kiện quan trọng của list là `SelectIndexChanged`, sự kiện này được phát sinh khi bạn click chọn vào một phần tử khác hay dùng phím di chuyển qua các phần tử khác trong ListBox.

Ví dụ: Sử dụng Control thiết kế chương trình Demo ListBox trong Windows Forms:

| Đối tượng | Thuộc tính | Sự kiện |
|-----------|---|--------------------|
| Form | Name: frmDemoListBox Text: DemoListBox | FormClosing |
| ListBox | Name: lstNguon Items: danh sách collection | |
| ListBox | Name: lstNguon | |
| Button | Name: btnAdd Text: > | btnAdd_Click |
| Button | Name: btnAddAll Text: >> | btnAddAll_Click |
| Button | Name: btnRemove Text: < | btnRemove_Click |
| Button | Name: btnRemoveAll Text: << | btnRemoveAll_Click |
| Button | Name: btnExit Text: Exit | btnExit_Click |
| Label | Name: Default Text: Danh sách nguồn | |
| Label | Name: Default Text: Danh sách đích | |

Giao diện chương trình sau thiết kế



Hình 4.6 Control ListBox

Code cho các sự kiện của các control

| Sự kiện | Code cho các sự kiện |
|--------------|---|
| FormClosing | <pre>private void frmDemoListBox_FormClosing(object sender, FormClosingEventArgs e) { DialogResult dl = MessageBox.Show("Bạn có muốn thoát không?", "Thông báo", MessageBoxButtons.YesNo, MessageBoxIcon.Question); if (dl == DialogResult.No) { e.Cancel = true; } }</pre> |
| btnAdd_Click | <pre>private void btnAdd_Click(object sender, EventArgs e) { try { lstDich.Items.Add(lstNguon.SelectedItem); lstNguon.Items.Remove(lstNguon.SelectedItem); } catch (ArgumentException ex) { MessageBox.Show("Bạn chưa chọn phần tử nào trong nguồn" , "Thông báo"); } }</pre> |

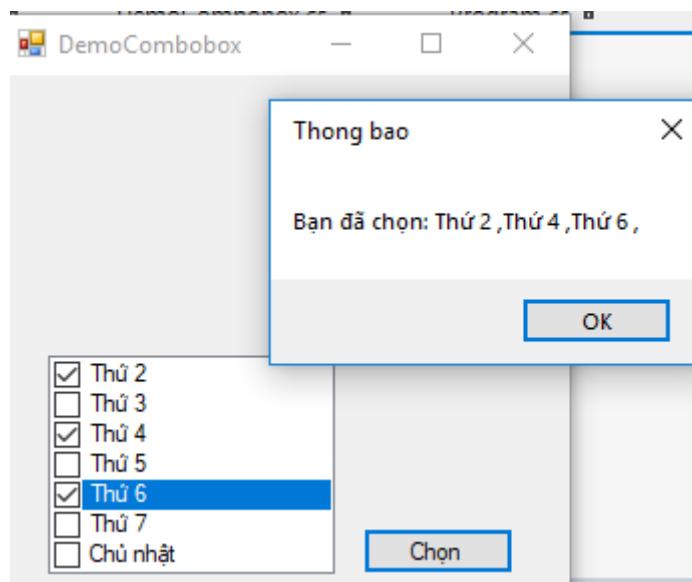
| | |
|--------------------|--|
| | <pre> } catch(Exception ex) { MessageBox.Show(ex.Message); } } } </pre> |
| btnAddAll_Click | <pre> private void btnAddAll_Click(object sender, EventArgs e) { foreach (string item in lstNguon.Items) { lstDich.Items.Add(item); } lstNguon.Items.Clear(); } </pre> |
| btnRemove_Click | <pre> private void btnRemove_Click(object sender, EventArgs e) { try { lstNguon.Items.Add(lstDich.SelectedItem); lstDich.Items.Remove(lstDich.SelectedItem); } catch (ArgumentException ex) { MessageBox.Show("Bạn chưa chọn phần tử nào trong Đích", "Thông báo"); } catch (Exception ex) { MessageBox.Show(ex.Message); } } </pre> |
| btnRemoveAll_Click | <pre> private void btnRemoveAll_Click(object sender, EventArgs e) { foreach (string item in lstDich.Items) { lstNguon.Items.Add(item); } } </pre> |

| | |
|---------------|--|
| | <pre> } lstDich.Items.Clear(); } } </pre> |
| btnExit_Click | <pre> private void btnExit_Click(object sender, EventArgs e) { Close(); } </pre> |

CheckListBox

CheckListBox cung cấp cho bạn tất cả khả năng của ListBox đồng thời cho phép hiển thị CheckBox bên cạnh các phần tử trong ListBox. Người dùng có thể đánh dấu Check cho các phần tử trong CheckListBox và có thể được duyệt qua bằng điều hướng CheckedListBox.CheckedCollection hoặc CheckedListBox.SelectedIndexCollection .

Trường hợp muốn kiểm tra một mục trong CheckListBox có được chọn hay không chúng ta kiểm tra Checked của phần tử trong List.



Hình 4.7 Control CheckListBox

Sự kiện khi Click nút chọn

```

private void button1_Click(object sender, EventArgs e)
{
    string str = "";
    foreach (string item in checkedListBox1.CheckedItems)
    {
        str += item + " ,";
    }
    MessageBox.Show("Bạn đã chọn " + str, "Thông báo");
}

```

ComboBox Control

Tương ứng với một trình đơn đưa ra một danh sách các mục và ta chỉ có chọn một trong các mục đó mà thôi. Các thuộc tính và sự kiện giống như ListBox nhưng chỉ khác ListBox là trong ComboBox chỉ chọn được một phần tử duy nhất.

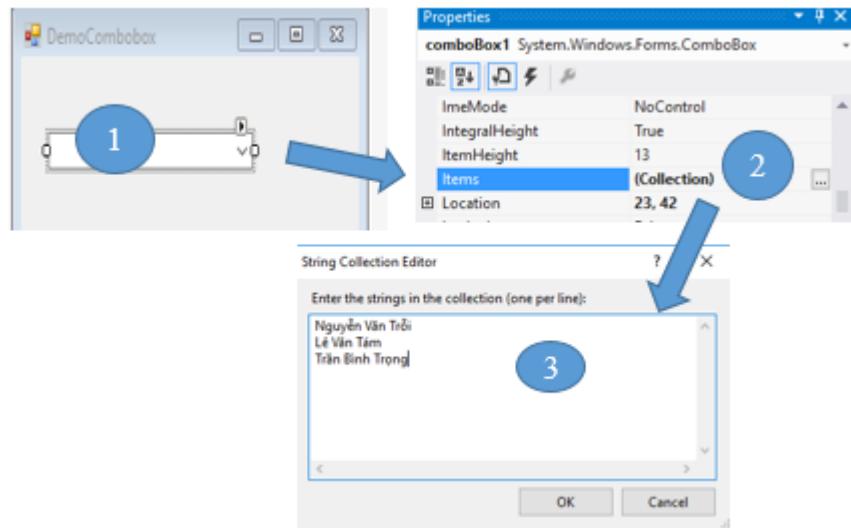
Các thuộc tính của ComboBox:

| Thuộc tính | Giá trị | Điễn giải |
|---------------|-----------------------|---|
| DropDownStyle | DropDown | Popup có mũi tên xuống và có thể cho nhập vào ô text |
| | DropDownList | Popup có mũi tên xuống và không cho nhập vào ô text, chỉ chọn 1 giá trị trong Droplist |
| | Simple | Nó giống TextBox và ListBox nằm liền kề nhau, có thể nhập dữ liệu vào ô text, nếu chọn giá trị trong List bên dưới sẽ hiển thị lên trên TextBox |
| ValueMember | Text | Giá trị text kết nối với dữ liệu nguồn |
| DisplayMember | Text | Giá trị text hiển thị với kết nối với dữ liệu nguồn |
| DataSource | Kết nối dữ liệu nguồn | Kết nối với dữ liệu nguồn |

Một số phương thức của ComboBox: Một số phương thức của ComboBox cũng giống ListBox: phương thức Add, AddRange, Remove, RemoveAt, Clear.

Một số sự kiện của ComboBox: Sự kiện mặc định của ComboBox là ComboBox_SelectedIndexChanged, xảy ra khi chúng ta chọn một phần tử mới trong ComboBox.

Ví dụ Demo sử dụng ComboBox



Hình 4.8 ComboBox

GroupBox, Panel

GroupBox là một control thuộc lớp chứa, thiết kế để hiển thị một khung bao quanh các control, có thể hiển thị tiêu đề với thuộc tính Text, khi thiết lập các giá trị của GroupBox nó sẽ ảnh hưởng đến các control mà nó chứa hoặc khi chúng ta xóa GroupBox thì các control chứa trong nó sẽ bị xóa theo, Lớp GroupBox kế thừa từ System.Windows.Forms.Control

Thuộc tính thường dùng của GroupBox:

| | |
|------------|---|
| Thuộc tính | Mô tả |
| Controls | Danh sách các control chứa trong GroupBox |
| Text | Caption của GroupBox |

Panel là một control chứa nhóm các control, Panel không có thuộc tính text, nhưng Panel có thanh cuộn (Scrollbar) khi chúng ta thiết kế có chứa nhiều control khi kích thước Panel bị giới hạn,

Thuộc tính thường dùng của Panel

| | |
|-------------|--|
| Thuộc tính | Mô tả |
| Controls | Danh sách các control chứa trong Panel |
| AutoScroll | Xuất hiện khi Panel bị giới hạn để hiển thị hết các control, mặc định là False |
| BorderStyle | Khung viền của Panel, mặc định là None, các tham số khác như: Fixed3D, FixedSingle |

Ví dụ minh họa sử dụng GroupBox và Panel



Hình 4.9 GroupBox và Panel

PictureBox

PictureBox là control dùng để hiển thị hình ảnh, sử dụng thuộc tính Image để thiết lập ảnh lúc thiết kế hoặc lúc chạy chương trình.

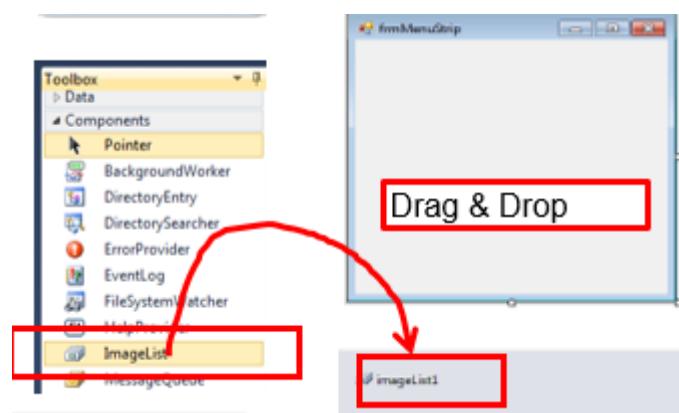
Các thuộc tính của PictureBox

| Thuộc tính | Điễn giải |
|------------|--|
| Image | Ảnh cần hiển thị |
| SizeMode | <ul style="list-style-type: none"> - Normal: Ảnh chuẩn - StretchImage: Giảm kích thước theo PictureBox - AutoSize: Tự động điều chỉnh cỡ ảnh - CenterImage: Căn giữa ảnh |

ImageList

ImageList là control dùng để quản lý các đối tượng hình ảnh, ImageList thường được sử dụng kết hợp với control khác, như ListView, TreeView, MenuStrip, toolBar....

Tạo ImageList



Hình 4.10 Image List

Thiết lập thuộc tính và lấy hình ảnh cho Images trong Properties của ImageList

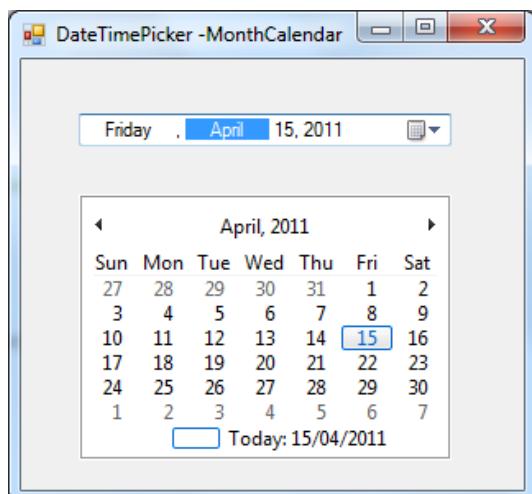


Hình 4.11 Images trong Properties của ImageList

4.5 Điều khiển đặc biệt

DatePicker và MonthCalendar

Control dùng để thiết lập cho người dùng chọn ngày tháng năm từ control.



Hình 4.12 DateTimePicker và MonthCalendar

Một số thuộc tính chính của DateTimePicker và MonthCalender

| Thuộc tính | Điễn giải |
|--------------------|---|
| Name | Thuộc tính tên của control |
| CustomFormat | Thiết lập chuỗi ngày giờ tùy chỉnh |
| Value | Thiết lập giá trị thuộc tính trước khi điều khiển hiển thị |
| MinDate MaxDate | Xác định phạm vi ngày và giờ trong một khoảng ngày nhỏ nhất và lớn nhất |
| | Thiết lập ngày lớn nhất trong control |

Sự kiện chính cho DateTimePicker là ValueChanged: Sự kiện ValueChanged được kích hoạt khi giá trị của control bị thay đổi.

ListView

ListView là dạng control phổ biến hiển thị một danh sách item, các item có thể có các item con được gọi là SubItem.

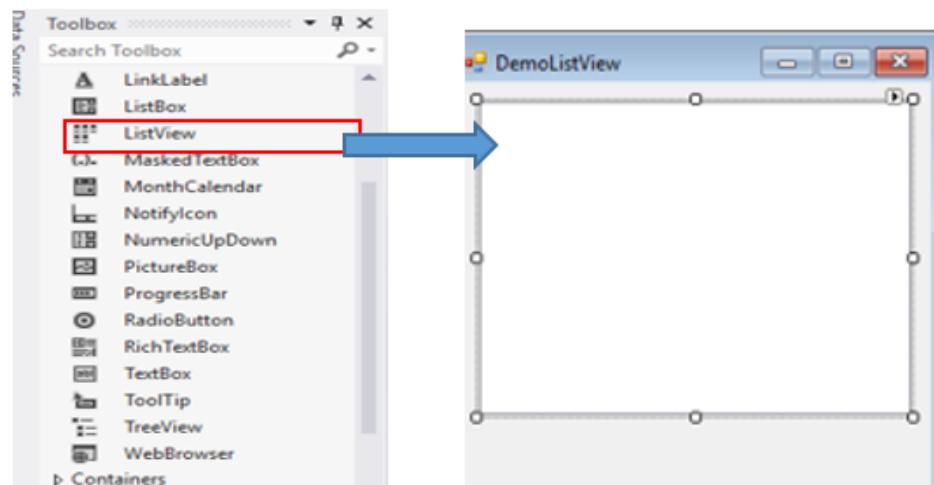
| Thuộc tính | Điễn giải |
|------------------|--|
| View | SmallImage |
| | Hiển thị dạng Icon nhỏ |
| | LargeImage |
| | Hiển thị dạng Icon Lớn |
| | Details |
| | Hiển thị dạng chi tiết |
| | List |
| | Hiển thị dạng danh sách |
| | Title |
| | Hiển thị dạng Icon tiêu đề |
| Columns | Danh sách các cột tiêu đề |
| Items | Danh sách các phần tử trong ListView |
| MultiSelect | True / False cho chọn nhiều Item |
| FullRowSelect | True / False cho chọn một dòng |
| ContextMenuStrip | Cho chọn ContextMenuStrip |
| Sorting | - Ascending: Sắp xếp theo thứ tăng dần - Descending: Sắp xếp theo thứ tự giảm dần |

| | |
|----------------|---|
| LargeImageList | Danh sách ImageList khi chọn LargeImage |
| SmallImageList | Danh sách ImageList khi chọn SmallImage |

Thêm một ListView control vào ứng dụng

Có 2 cách để thêm ListView vào ứng dụng:

Cách 1: Kéo ListView control từ ToolBox vào Forms

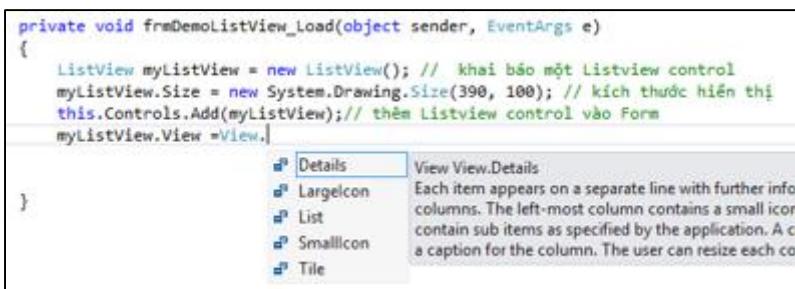


Hình 4.13 DemoListView

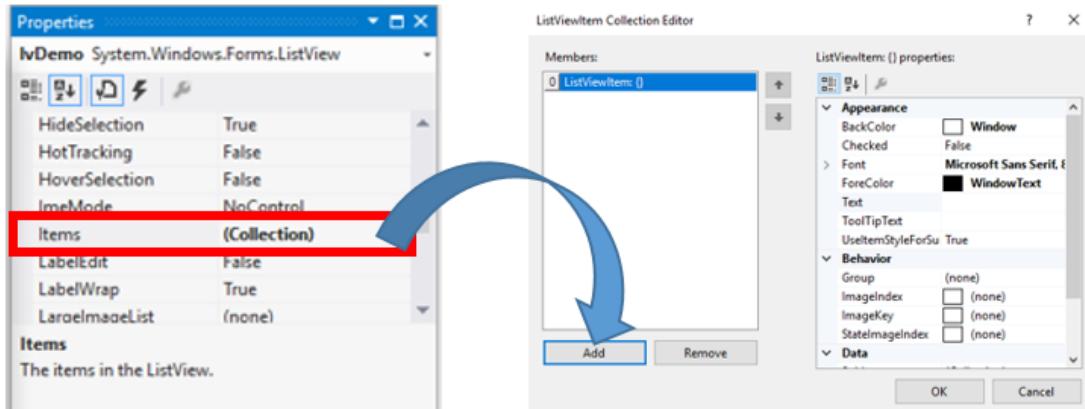
Cách 2: Sử dụng code để thêm vào một ListView trên Forms

```
private void frmDemoListView_Load(object sender, EventArgs e)
{
    ListView myListview = new ListView(); // khai báo một Listview control
    myListview.Size = new System.Drawing.Size(390, 100); // kích thước hiển thị
    this.Controls.Add(myListview); // thêm Listview control vào Form
}
```

Thay đổi chế độ xem ListView: Có thể tùy chỉnh trong thuộc tính của ListView hoặc chúng ta có thể viết code để thay đổi hiển thị của ListView.



Thêm các phần tử vào ListView sử dụng thuộc tính Items trong cửa sổ Properties.



Hình 4.14 Thuộc tính của ListView

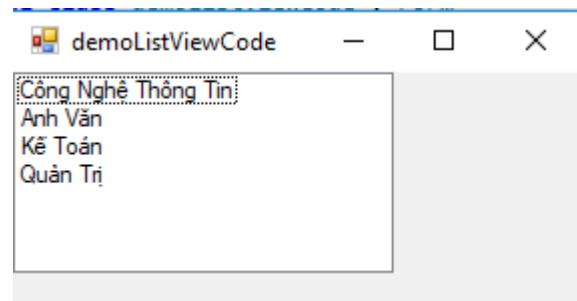
Mỗi Item Add sẽ có các thuộc tính như: Text, ForeColor, ImageIndex

Chúng ta cũng có thể viết code để thêm các item vào ListView với mục đích tương tự như thiết kế bằng control Design.

Viết code để tạo

```
myListView.View = View.List;
myListView.Items.Add("Công Nghệ Thông Tin");
myListView.Items.Add("Anh Văn");
myListView.Items.Add("Kế Toán ");
myListView.Items.Add("Quản Trị");
```

Kết quả



Thêm các Column vào ListView chúng ta cũng có thể thêm bằng 2 cách, thêm các Column từ Design hoặc thêm các cột bằng code

```
// thêm các colum bằng code
myListView.Columns.Add("Tên khoa", 190);
myListView.Columns.Add("Số lượng", 90);
myListView.View = View.Details;
```

Thêm vào các item và SubItem trong ListView,

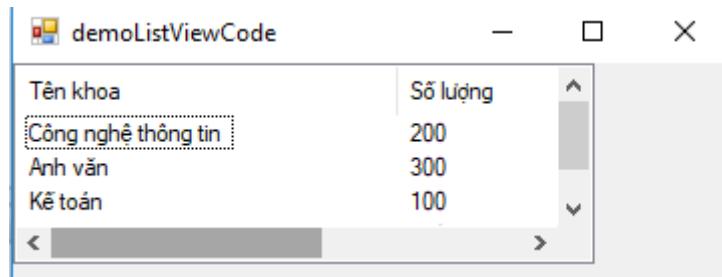
```

21     private void demoListViewCode_Load(object sender, EventArgs e)
22     {
23         ListView myListview = new ListView(); // khai báo một Listview control
24         myListview.Size = new System.Drawing.Size(290, 100); // kích thước hiển thị
25         this.Controls.Add(myListview); // thêm Listview control vào Form
26         // thêm các colum bằng code
27         myListview.Columns.Add("Tên khoa", 190);
28         myListview.Columns.Add("Số lượng", 90);
29         myListview.View = View.Details;
30
31         //thêm item và subitem vào listview
32         ListViewItem cttt = new ListViewItem("Công nghệ thông tin");
33         ListViewItem.ListViewSubItem slctt = new ListViewItem.ListViewSubItem(cttt, "200");
34         cttt.SubItems.Add(slctt);
35         // thêm vào listview
36         myListview.Items.Add(cttt);

37
38         ListViewItem av = new ListViewItem("Anh văn");
39         ListViewItem.ListViewSubItem slav = new ListViewItem.ListViewSubItem(av, "300");
40         av.SubItems.Add(slav);
41         // thêm vào listview
42         myListview.Items.Add(av);
43         ListViewItem kt = new ListViewItem("Kế toán");
44         ListViewItem.ListViewSubItem slkt = new ListViewItem.ListViewSubItem(kt, "100");
45         kt.SubItems.Add(slkt);
46         // thêm vào listview
47         myListview.Items.Add(kt);
48         ListViewItem qt = new ListViewItem("Quản Trị");
49         ListViewItem.ListViewSubItem slqt = new ListViewItem.ListViewSubItem(qt, "150");
50         qt.SubItems.Add(slqt);
51         // thêm vào listview
52         myListview.Items.Add(qt);
53

```

Kết quả chương trình



Để xóa hết tất cả các item trong ListView sử dụng method Clear.

```
// xóa hết các phần tử trong Listview
myListview.Clear();
```

Để xóa một phần tử trong ListView hoặc xóa một phần tử tại một vị trí trong ListView

```
//xóa phần tử trong listview
cttt.Remove(); // xóa phần tử công nghệ thông tin
```

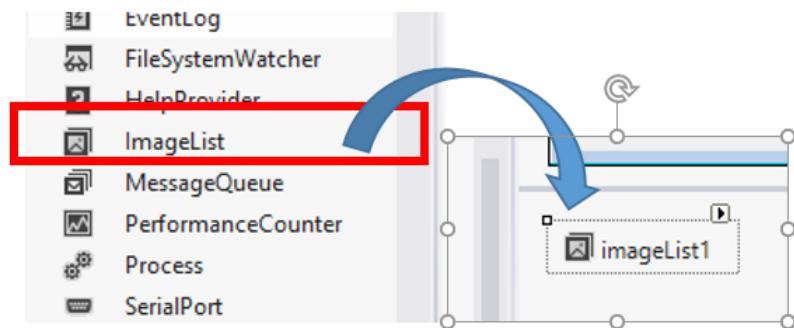
Xóa phần tử tại vị trí số 2

```
//xóa phần tử trong listView tại vị trí  
myListView.Items.RemoveAt(2); // xóa phần tử tại vị trí 2
```

Liên kết hình ảnh với danh sách các Items của ListView

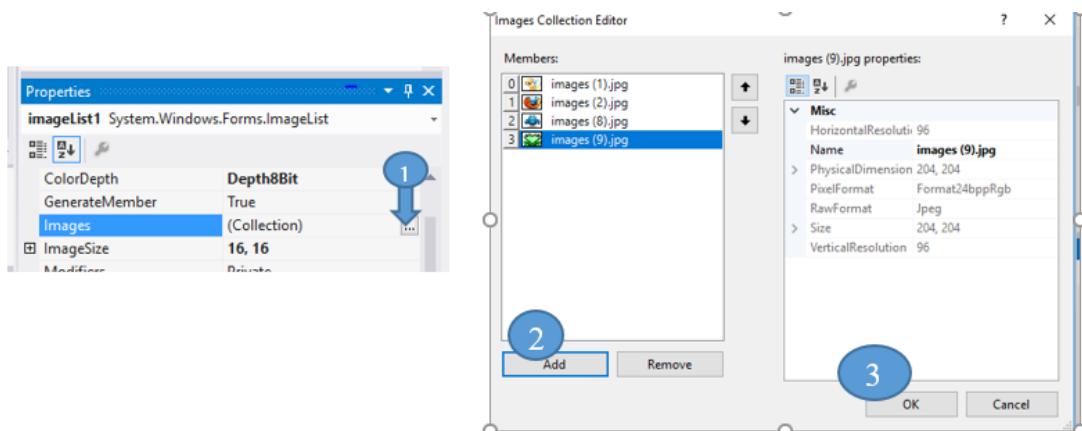
Để liên kết hình ảnh chúng ta sử dụng ImageList với tập các hình ảnh, cùng với ListViewItem để sử dụng ImageIndex là chỉ mục trong ImageList.

-Tạo ImageList với danh sách các hình ảnh.



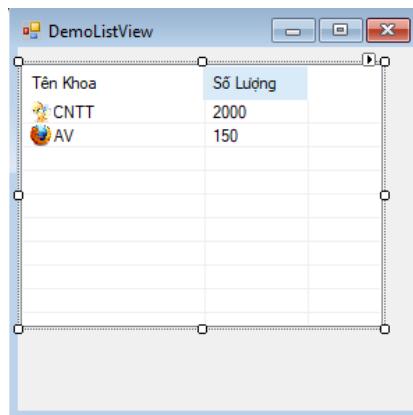
Hình 4.15 ImageList

Thêm hình ảnh vào ImageList, chọn thuộc tính Images trong Properties trong ImageList



Hình 4.16 Thuộc tính của ImageList

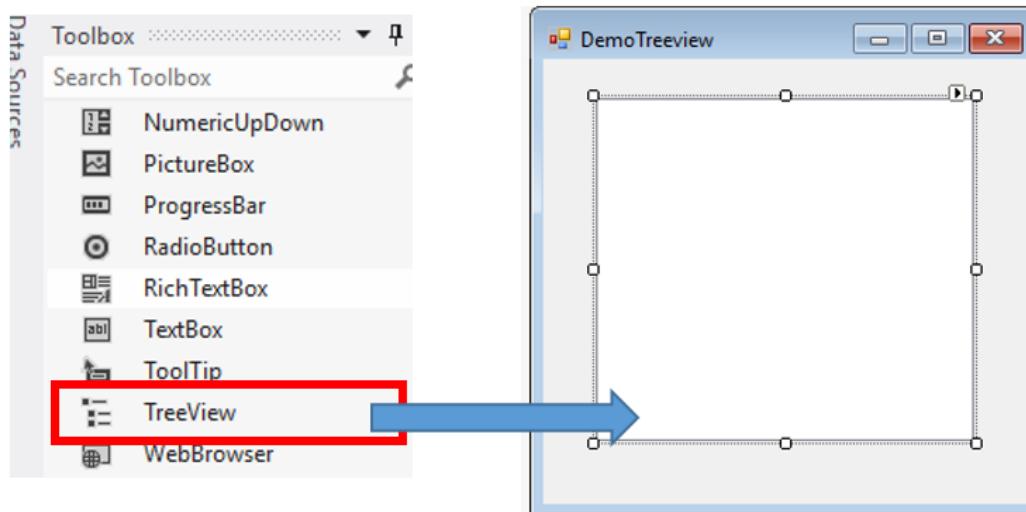
Sử dụng ImageList cho ListView bằng code hoặc chọn thuộc tính trong Properties của ListView để chọn danh sách hình ảnh.



Hình 4.17 Sử dụng ImageList cho ListView

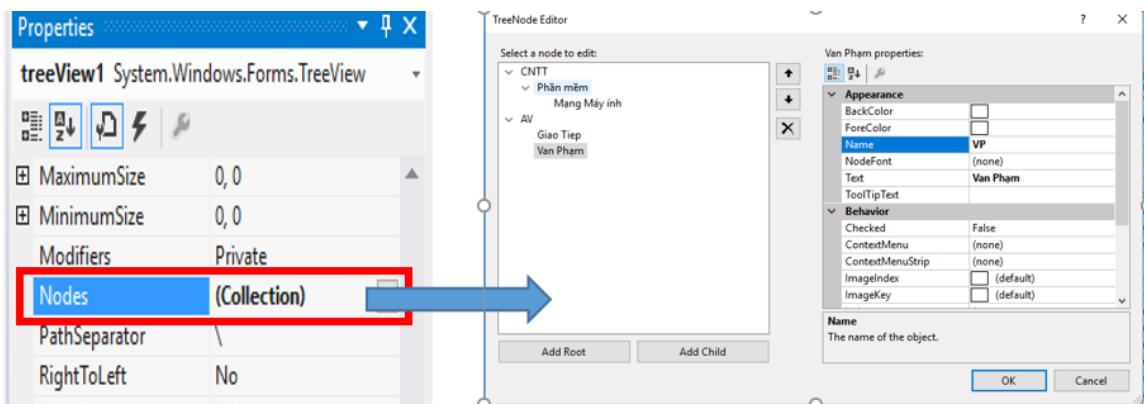
TreeView

Điều khiển TreeView dùng để trình bày danh sách phân tử phân cấp theo từng Node hình cây. Thực hiện tạo TreeView ta kéo điều khiển vào Forms từ hộp ToolBox.



Hình 4.18 Hình sử dụng TreeView

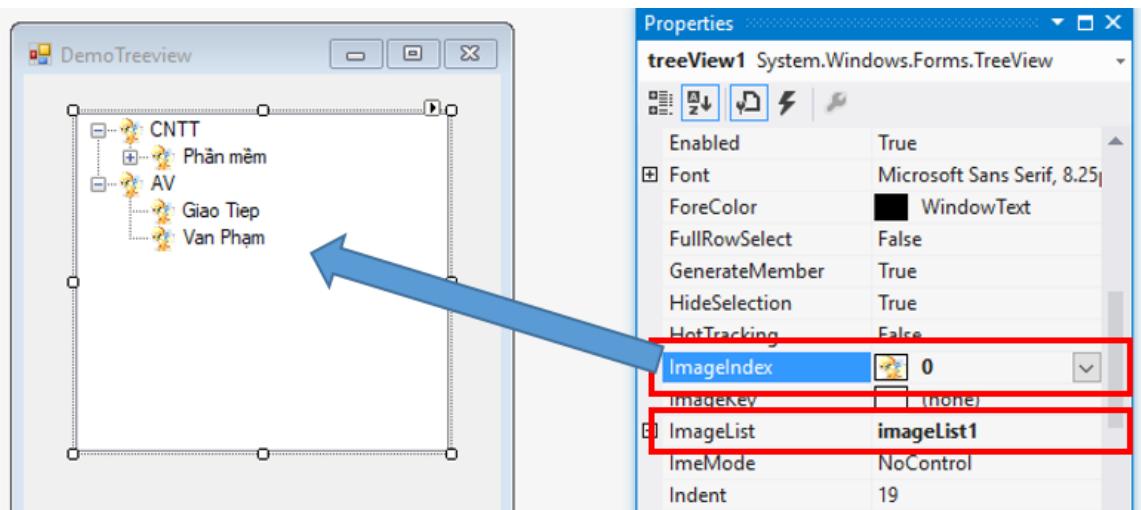
Khai báo các thuộc tính Nodes: Khai báo số node của điều khiển TreeView



Hình 4.19 Thuộc tính sử dụng TreeView

Một số thuộc tính của TreeView

| Thuộc tính | Diễn giải |
|--------------------|--|
| ShowPlusMinus | Chọn giá trị True thì biểu tượng dấu + và - xuất hiện trên mỗi node, mặc định là True. |
| ImageList | Chứa đối tượng ImageList |
| ImageIndex | Tất cả các node được thể hiện bởi hình ảnh mà thuộc tính này đã chọn |
| SelectedImageIndex | Node được chọn sẽ có hình ảnh được chọn |
| Checkboxes | Xuất hiện CheckBox bên cạnh node |



Hình 4.20 Thuộc tính ImageIndex trong TreeView

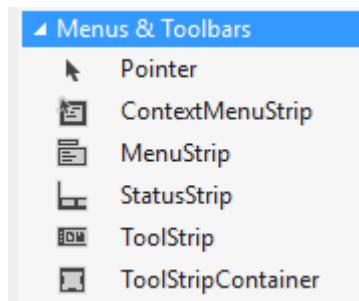
Một số phương thức của TreeView

| Phương thức | Diễn giải |
|------------------------|---|
| CollapseAll() | Hiển thị tất cả các node trên điều khiển TreeView |
| ExpandAll() | Thu gọn tất cả các node trên điều khiển TreeView |
| Clear() | Xóa tất cả các node đang tồn tại |
| Các biến cố thông dụng | |
| AfterCheck: | Xảy ra khi người dùng click vào CheckBox |

| | |
|---------------|------------------------------------|
| AfterCollapse | Xảy ra khi thu gọn tất cả các node |
| AfterExpand | Xảy ra khi mở rộng tất cả các node |
| AfterSelect | Xảy ra khi Click vào node |

4.6 Điều khiển menu

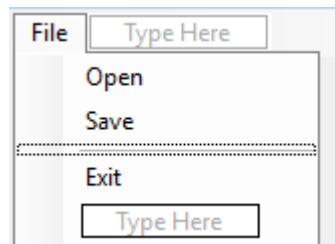
Menu dùng để thiết kế giao diện có menu, menu là control trên Forms, chúng ta có thể tạo menu bằng 2 cách, cách 1 dùng code để tạo menu khi chạy chương trình gọi là Runtime, cách 2 chọn menu cần thiết kế từ ToolBox kéo vào Forms để thiết kế.



Hình 4.21 Menu và Toolbars

Có 2 loại menu: Menu truyền thống (MenuStrip), menu ngữ cảnh (ContextMenu)

Tạo MenuStrip như sau: Kéo MenuStrip vào Forms và nhập tên menu vào ô TypeHere,



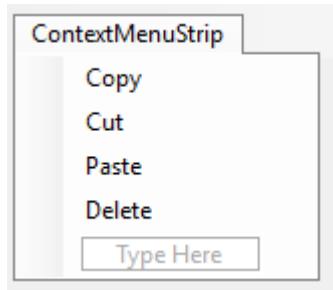
Hình 4.21 Tạo Menu File

4.6.1 Một số thuộc tính của Menu

| | |
|-------------------|---|
| Thuộc tính | Diễn giải |
| Items | Chứa những top menu |
| MDIWindowListItem | Chọn Top menu hiển thị tên các cửa sổ con |

Sự kiện cho Menu Item thường dùng là Click

4.6.2 Tạo ContextMenu



Hình 4.22 Tạo ContextMenuStrip

ContextMenu xuất hiện khi ta click chuột phải, thông thường ContextMenu áp dụng cho control nào trên Forms, hoặc áp dụng cho Form khi chúng ta Click chuột phải. Để tạo ContextMenu chúng ta chọn control trong ToolBox và kéo thả vào trên Form sau đó ta thiết kế như hình trên.

Đối với các Control đều có thuộc tính ContextMenuStrip, chúng ta chọn menu strip cho control cần đặt ContextMenu, khi chúng ta chạy chương trình nhấn chuột phải trên control sẽ xuất hiện ContextMenu.

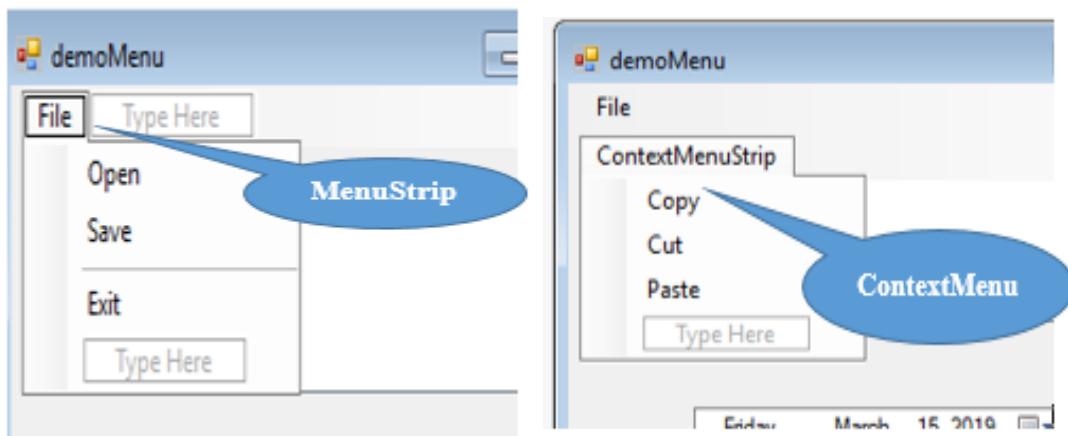
Các sự kiện cho ContextMenu chủ yếu là click

Ví dụ: Sử dụng Menustrip và ContextMenu cho chương trình sau:

Thiết kế chương trình có các control sau:

| Đối tượng | Thuộc tính |
|-------------|--------------------------------------|
| Form | Name: frmDemoMEnu Text: Demo Menu |
| Mainmenu | Name: mnuMain |
| MenuItem | Name: mnuFile Text: File |
| MenuItem | Name: mnuOpen Text: Open |
| MenuItem | Name: mnuCach Text: - |
| MenuItem | Name: mnuExit Text: Exit |
| ContextMenu | Name: ContextMenu |

| | |
|-----------------|---|
| MenuItemContext | Name: mnuCopy Text: Copy |
| MenuItemContext | Name: mnuCut Text: Cut |
| MenuItemContext | Name: mnuCut Text: Paste |
| ListBox | Name: lstDemo Items: Nguyen Trai Nguyen Khuyen ContextMenuStrip: menutextMenu |



Hình 4.23 Demo Menu File và ContextMenu

4.7 Sử dụng hộp thoại

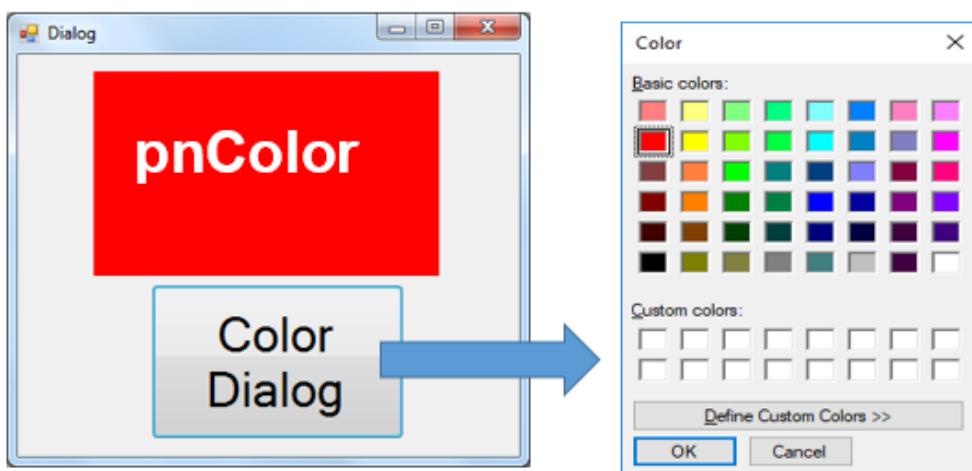
Trong Windows Forms đã xây dựng sẵn một số lớp về hộp thoại cho người lập trình sử dụng, khi chúng ta sử dụng hộp thoại thì giá trị trả về là giá trị được chọn trong hộp thoại.

4.7.1 Hộp thoại màu ColorDialog



Hình 4.24 Tạo ColorDialog

Ví dụ thiết kế chương trình lấy màu cho cho một đối tượng trên Forms như sau:



Hình 4.25 Demo ColorDialog

Xử lý sự kiện khi click trên nút lệnh ColorDialog

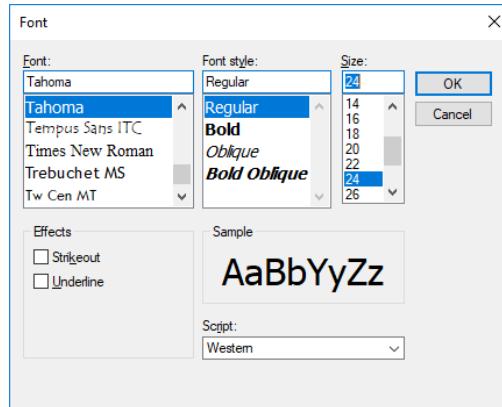
```
private void btnColor_Click(object sender, EventArgs e)
{
    ColorDialog cldlg = new ColorDialog();
    cldlg.Color = pnColor.BackColor;

    if (cldlg.ShowDialog() == DialogResult.OK)
        pnColor.BackColor = cldlg.Color;
}
```

4.7.2 Hộp thoại Font

Hộp thoại FontDialog thường được dùng trên C# để xây dựng các phần mềm soạn thảo văn bản, hoặc thiết lập Font cho các control khi chạy chương trình ứng dụng.

Hộp thoại Font cung cấp danh sách các Font chữ được cài đặt trên hệ thống, cho phép người dùng chọn thuộc tính cho Font chữ, ví dụ cho thay đổi Font, kích thước Font, và các hiệu ứng.



Hình 4.25 Tạo Font

Các thuộc tính, phương thức và sự kiện của hộp thoại Font

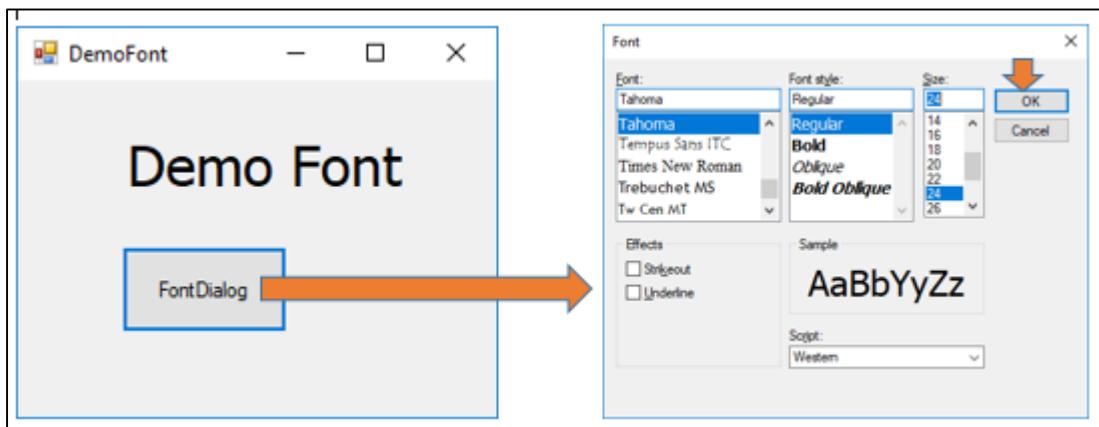
| | |
|-------------|-------------------------------------|
| Thuộc tính | Diễn giải |
| Font | Get/Set Font được chọn |
| Color | Get/Set màu của Font |
| ShowColor | Hiển thị / không hiển thị chọn màu |
| ShowApply | Hiển thị/ không Button Apply |
| Phương thức | |
| ShowDialog | Hiển thị hộp thoại ra màn hình |
| Sự kiện | |
| Apply | Kích hoạt khi người dùng chọn Apply |

Ví dụ về thiết kế Forms có 2 control để sử dụng hộp thoại Font như sau:

| Control | Thuộc tính |
|---------|----------------------------------|
| Button | Name: btnFont Text:FontDialog |
| Label | Name: lblFont |



Thiết kế Form và chương trình



Hình 4.26 Demo Font

Sự kiện khi click trên Button FontDialog

```
private void btnFont_Click(object sender, EventArgs e)
{
    FontDialog ftDialog = new FontDialog();
    ftDialog.Font = lblFont.Font;
    if (ftDialog.ShowDialog() == DialogResult.OK)
    {
        lblFont.Font = ftDialog.Font;
    }
}
```

4.7.3 Hộp thoại OpenFileDialog và SaveDialog

Hộp thoại OpenFileDialog chọn tập tin để mở hoặc lưu dữ liệu từ hộp thoại SaveDialog

Thuộc tính, phương thức và sự kiện của hộp thoại OpenFileDialog

| Thuộc tính | Điễn giải |
|------------------|-------------------------------|
| FileName | Lấy tên file được chọn |
| FileNames | Lấy tất cả các file được chọn |
| Filter | Xác định kiểu file để mở |
| InitialDirectory | Thư mục khởi tạo |
| MultiSelect | Cho phép chọn nhiều file |
| Title | Tiêu đề hộp thoại |
| Phương thức | |
| ShowDialog | Hiển thị hộp thoại |

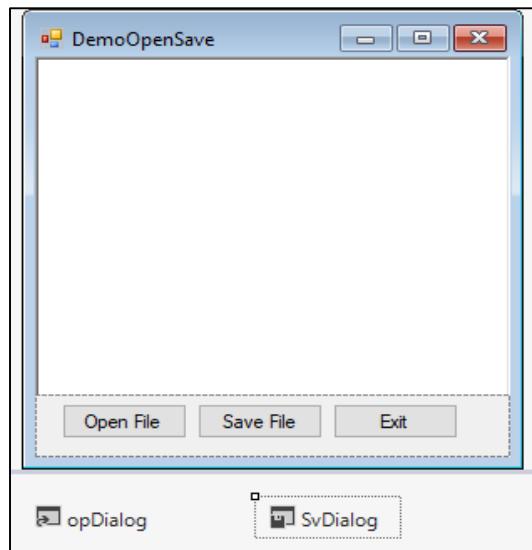
| Sự kiện | |
|---------|----------------------------------|
| FileOK | Xuất hiện khi người dùng chọn OK |

Ví dụ : Viết chương trình đọc và ghi file có kiểu dữ liệu .txt

Sử dụng các control sau:

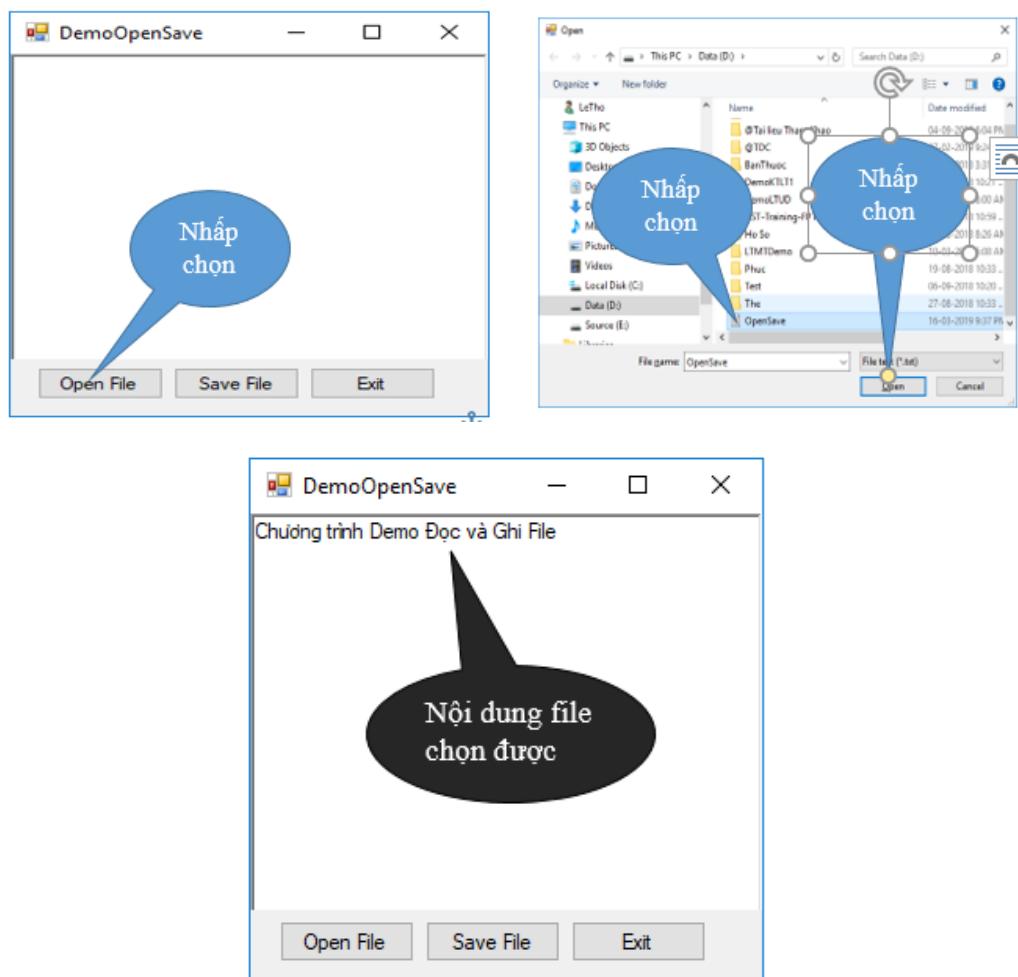
| Control | Thuộc tính |
|----------------|---|
| Panel | Name: pnButton Dock: Bottom |
| RichTextBox | Name: rtfNoiDung Dock: Fill |
| Button | Name:btnOpen Text: Open File |
| Button | Name:btnSave Text: Save File |
| Button | Name:btnExit Text: Exit |
| Form | Name:frm DemoOpenSave Text: DemoOpenSave |
| OpenFileDialog | Name: opDialog |
| saveFileDialog | Name: svDialog |

Thiết kế giao diện chương trình như sau:



Hình 4.27 Tạo giao diện OpenSave

Kết quả khi chạy chương trình và chọn nút Open File

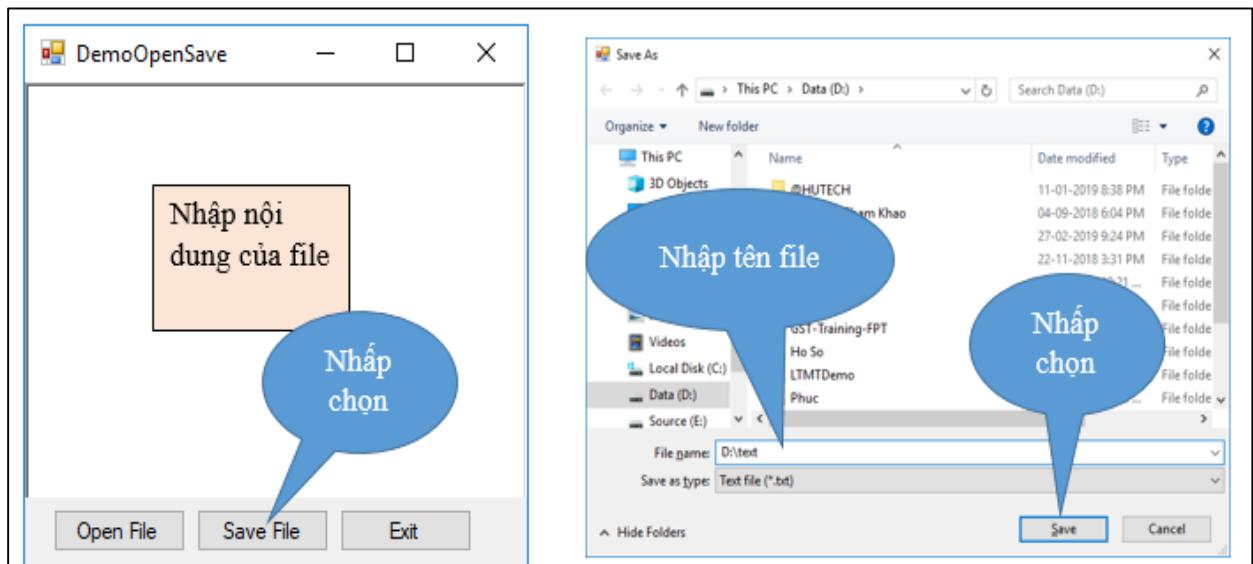


Hình 4.28 Demo hộp thoại thao tác với File

Code cho sự kiện khi click chọn nút lệnh Open File

```
private void btnOpen_Click(object sender, EventArgs e)
{
    opDialog.Multiselect = true; // cho phép chọn nhiều file
    // lọc những file để mở
    opDialog.Filter = "File text (*.txt)|*.txt";
    DialogResult result;
    // hiển thị và lấy giá trị trả về
    result = opDialog.ShowDialog();
    if (result == DialogResult.OK)
    {
        // tạo stream để đọc
        // sử dụng thư viện System.IO
        Stream streamRead = opDialog.OpenFile();
        // tạo đối tượng để đọc
        StreamReader read = new StreamReader(streamRead);
        // đọc nội dung lên richtextbox
        rtfNoiDung.Text = read.ReadToEnd();
        // đóng đối tượng đọc
        read.Close();
        // đóng file
        streamRead.Close();
    }
}
```

Khi chọn nút lệnh Save File



Hình 4.29 Demo thực hiện Save File

Code cho sự kiện khi chọn nút lệnh Save File

```
private void btnSave_Click(object sender, EventArgs e)
{
    // lưu file cho nút chọn Save File
    // lưu file dạng txt
    SvDialog.Filter= "Text file (*.txt)|*.txt";
    if (SvDialog.ShowDialog() == DialogResult.OK)
    {
        //tao stream để đọc ghi file
        Stream sw = SvDialog.OpenFile();
        // tao đối tượng ghi file
        StreamWriter writer = new StreamWriter(sw);
        // lưu nội dung từ ricktextbox xuống file
        writer.Write(rtfNoiDung.Text);
        writer.Close();
        sw.Close();
    }
}
```

Code cho sự kiện nút lệnh Exit

```
private void btnExit_Click(object sender, EventArgs e)
{
    DialogResult result;
    result = MessageBox.Show("Bạn có muốn thoát không? ", "Thông báo",
                           MessageBoxButtons.YesNo,
                           MessageBoxIcon.Question);
    if (result == DialogResult.Yes)
    {
        Close();
    }
}
```

4.7.4 Hộp thoại MessageBox

Hộp thoại MessageBox là hộp thoại thông báo chỉ dẫn cho người dùng, thông tin hộp thoại bao gồm Text, Button, Symbol, trong C# xây dựng một lớp MessageBox có các phương thức tĩnh Show để hiển thị dialog, lấy giá trị trả về để biết kết quả của người dùng lựa chọn. Trong lớp của MessageBox có 21 phương thức Show với các tham số truyền vào khác nhau.

Các Button hiển thị cho người dùng lựa chọn với MessageBox được định nghĩa trong MessageBoxButtons.

| Member | Value |
|------------------|-------|
| Ok | 0 |
| OkCancel | 1 |
| AbortRetryIgnore | 2 |
| YesNoCancel | 3 |
| YesNo | 4 |
| RetryCancel | 5 |

Hình 4.30 Giá trị của MessageBoxButtons

MessageBoxIcon

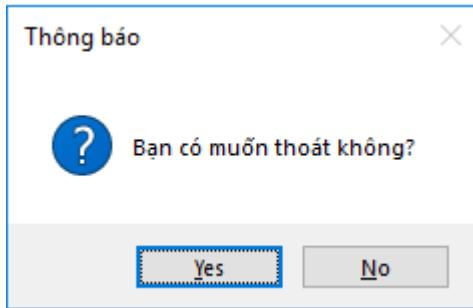
| Member | Value | Image |
|-------------|-------|-------|
| None | 0x00 | |
| Hand | 0x10 | |
| Stop | 0x10 | |
| Error | 0x10 | |
| Question | 0x20 | |
| Exclamation | 0x30 | |
| Warning | 0x30 | |
| Asterisk | 0x40 | |
| Information | 0x40 | |

Hình 4.31 MessageBoxIcon

Một số MessageBox minh họa

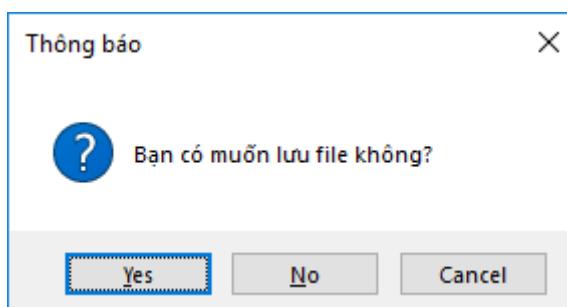
Ví dụ 1:

```
MessageBox.Show("Bạn có muốn thoát không? ", "Thông báo",
    MessageBoxButtons.YesNo,
    MessageBoxIcon.Question);
```



Ví dụ 2:

```
MessageBox.Show("Bạn có muốn lưu file không? ", "Thông báo",
    MessageBoxButtons.YesNoCancel,
    MessageBoxIcon.Question);
```



Ví dụ 3: MessageBox nhận kết quả Yes/No từ người dùng

```
DialogResult result;
result = MessageBox.Show("Bạn có muốn thoát không? ", "Thông báo",
    MessageBoxButtons.YesNo,
    MessageBoxIcon.Question);
if (result == DialogResult.Yes)
{
    MessageBox.Show("Bạn chọn Yes");
}
else
    MessageBox.Show("Bạn chọn No");
```

4.8 Unit test và Function test

4.8.1 Kiểm tra đơn vị (Unit test):

Kiểm tra đơn vị (unit test) là một kiểm thử phần mềm trong đó các đơn vị hoặc các thành phần riêng lẻ của phần mềm được kiểm tra.

Kiểm thử đơn vị các ứng dụng phần mềm được thực hiện trong quá trình phát triển của một ứng dụng. Mục tiêu của kiểm thử đơn vị là cô lập một phần mã và xác minh tính chính xác của nó. Trong lập trình thủ tục, một đơn vị có thể là một chức năng riêng lẻ, kiểm thử đơn vị thường được kiểm tra bởi nhà phát triển.

Kiểm thử đơn vị là cấp độ thử nghiệm đầu tiên được thực hiện trước khi thực hiện tích hợp. Kiểm thử đơn vị là một kỹ thuật kiểm tra hộp trắng (WhiteBox) thường được thực hiện bởi nhà phát triển.

- + Kiểm tra đơn vị sửa lỗi sớm trong chu kỳ phát triển và tiết kiệm chi phí
- + Giúp các nhà phát triển và cho phép thay đổi nhanh chóng
- + Kiểm tra đơn vị giúp sử dụng lại mã. Di chuyển mã và các thử nghiệm sang dự án mới, tinh chỉnh mã khi kiểm tra chạy lại.

Kiểm tra đơn vị thường tự động nhưng vẫn có thể thực hiện thủ công.

4.8.2 Kiểm tra chức năng (Function test):

Kiểm thử chức năng được định nghĩa là một loại kiểm thử xác minh rằng mỗi chức năng của ứng dụng phần mềm hoạt động phù hợp với đặc tả yêu cầu. Thử nghiệm này liên quan đến hộp đen và nó không quan tâm đến mã nguồn của ứng dụng.

Mỗi chức năng của hệ thống đều được kiểm tra bằng cách cung cấp đầu vào phù hợp, xác minh đầu ra và so sánh với kết quả mong đợi.

Thử nghiệm này bao gồm kiểm tra giao diện người dùng, API, cơ sở dữ liệu, bảo mật, ứng dụng đang được thử nghiệm.

Mục tiêu chính của kiểm thử chức năng là kiểm tra các chức năng hệ thống phần mềm bao gồm:

- + Các chức năng chính: Kiểm tra chức năng chính của chương trình
- + Khả năng sử dụng cơ bản: Thử nghiệm khả năng sử dụng cơ bản của hệ thống, kiểm tra người dùng có thể tự do điều hướng qua màn hình mà không gặp bất kỳ khó khăn nào.
- + Khả năng truy cập: Kiểm tra khả năng truy cập hệ thống của người dùng
- + Điều kiện lỗi: Sử dụng các kỹ thuật kiểm tra để kiểm tra lỗi, kiểm tra xem các thông báo lỗi phù hợp hiển thị.

Cách bước thực hiện kiểm tra chức năng:

- + Xác định bộ dữ liệu nhập: Identify test input
- + Tính toán kết quả mong đợi với các giá trị đầu vào thử nghiệm được chọn
- + Thực hiện các trường hợp thử nghiệm
- + So sánh kết quả dự kiến thực tế và tính toán

Trong kiểm thử phần mềm, kiểm tra chức năng là một quá trình kiểm tra các chức năng của hệ thống và đảm bảo rằng hệ thống đang hoạt động theo các chức năng được chỉ định trong tài liệu, mục tiêu của thử nghiệm là kiểm tra các chức năng hệ thống có chạy tốt không.

4.9 Bài tập

BÀI TẬP THỰC HÀNH SỐ 4

I. Thông tin chung:

- Mã số bài tập : BT-LTUD - 04
- Hình thức nộp bài : Nộp qua Moodle môn học
- Thời hạn nộp bài : ... / ... /
- Nội dung: Chương 4: Lập trình ứng dụng Windows Forms

Chuẩn đầu ra cần đạt:

L.O.3 Sử dụng các Control để thiết kế giao diện của chương trình

L.O.4 Viết code đúng chuẩn

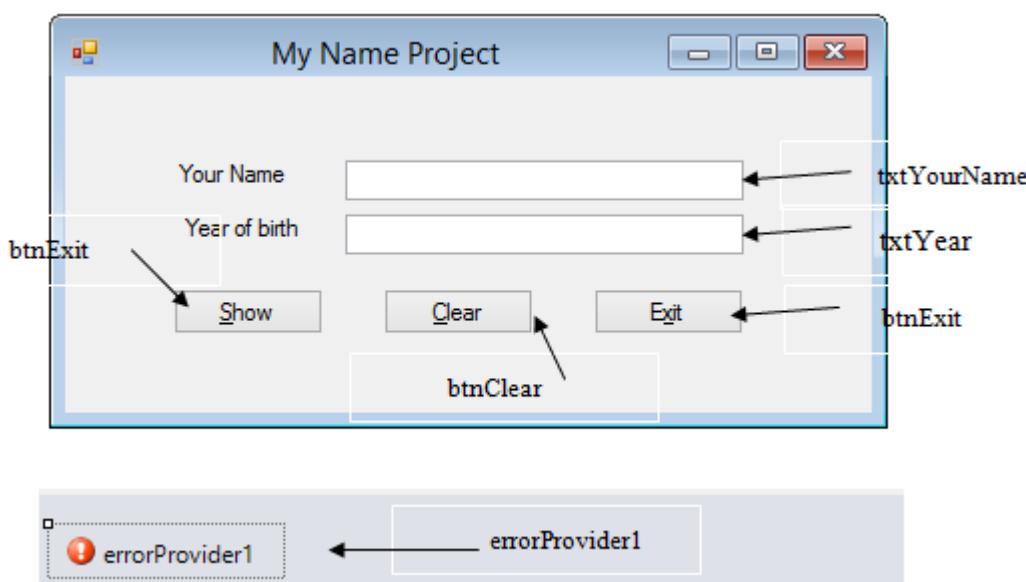
L.O.5 Sử dụng TestCase để kiểm tra phần mềm

L.O.6 Hiện thực được các chương trình vừa và nhỏ

L.O.7 Rèn luyện các kỹ năng tìm kiếm thông tin để tự giải quyết vấn đề

L.O.8 Tự tổ chức và quản lý hoạt động nhóm

Bài tập 1: Thiết kế Form sau



- Chương trình cho phép nhập tên, năm sinh vào Textbox YourName và Year of birth tương ứng. Nếu YourName không nhập dữ liệu, Year of birth không phải là số thì phải thông báo lỗi (dùng ErrorProvider). Người dùng nhấn nút Show sẽ hiển thị thông tin nhập vào MessageBox bao gồm: tên, tuổi (năm hiện tại – năm sinh).

- Người dùng nhấn nút Clear sẽ xóa hết thông tin đã nhập trên các Textbox, đồng thời đặt con trỏ vào Textbox YourName.

- Nút Exit xác nhận người dùng có thực sự muốn thoát khỏi chương trình không? (Yes: thoát, No: không).

Hướng dẫn:

Danh sách các thuộc tính của các object:

| Object | Properties | Events |
|---------------|--|--------------------------|
| frmMain | Name: frmMain Text: My name Project FontName: Tahoma FontSize: 11 AcceptButton: btnShow (nhận sự kiện click chuột khi nhấn Enter) CancelButton: btnExit (nhận sự kiện click chuột khi nhấn Esc) | FormClosing |
| txtYourName | Name: txtYourName BorderStyle: FixSingle | Leave (mất tiêu điểm) |
| TxtYear | Name: txtYear BorderStyle: FixSingle | TextChanged |
| btnShow | Name: btnShow Text: &Show | Click |
| btnClear | Name: btnClear Text: &Clear | Click |
| BtnExit | Name: btnExit Text: E&xit | Click |
| errorProvider | Name: errorProvider1 | |

Các sự kiện:

```

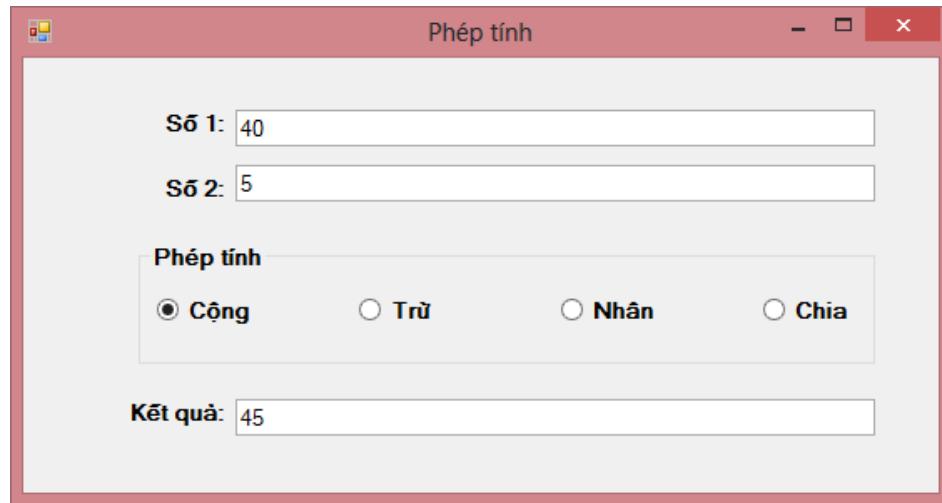
private void btnClear_Click(object sender, EventArgs e)
{
    txtYourName.Clear();
    txtYear.Clear();
    txtYourName.Focus();
}
private void btnShow_Click(object sender, EventArgs e)
{
    int age = DateTime.Now.Year -
Convert.ToInt32(txtYear.Text);
    string s = "My name is: " + txtYourName.Text + "\n" +
age.ToString();
    MessageBox.Show(s);
}
private void txtYourName_Leave(object sender, EventArgs e)
{
    Control ctr = (Control)sender;
    if (ctr.Text.Trim().Length == 0)

```

```
        this.errorProvider1.SetError(txtYourName, "You
must enter Your name");
    else
        this.errorProvider1.Clear();
}
private void txtYear_TextChanged(object sender, EventArgs
e)
{
    Control ctr = (Control)sender;
    if (ctr.Text.Trim().Length > 0 &&
!char.IsDigit(ctr.Text, ctr.Text.Length - 1))
        this.errorProvider1.SetError(txtYear, "This is
not invalid number");
    else
        this.errorProvider1.Clear();
}
private void frmMain_FormClosing(object sender,
FormClosingEventArgs e)
{
    DialogResult r;
    r = MessageBox.Show("Do you want to close?", "Exit",
    MessageBoxButtons.YesNo,
    MessageBoxIcon.Question,
    MessageBoxDefaultButton.Button1);
    if (r == DialogResult.No)
        e.Cancel = true;
}
private void btnExit_Click(object sender, EventArgs e)
{
    this.Close();
}
```

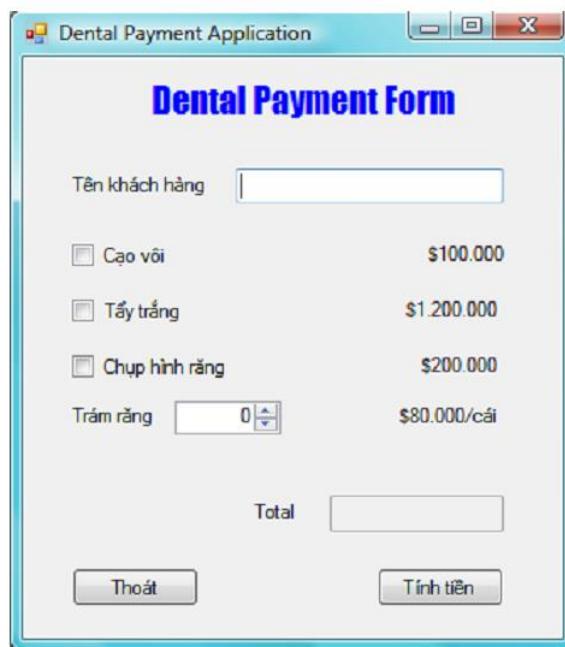
Bài tập 2: Thiết kế giao diện như hình, khi nhấn chọn vào phép tính nào thì sẽ hiện kết quả của phép tính đó vào ô kết quả.

Trước khi tính cần kiểm tra dữ liệu nhập phải là số.



Bài tập 3:

- ✓ Tạo một ứng dụng Windows Form cơ bản tính tiền công dịch vụ tại phòng nha.
- ✓ Với mỗi khách hàng, các dịch vụ cung cấp gồm: tẩy răng, cạo vôi, chụp hình răng và trám răng. Mỗi loại sẽ có chi phí riêng. Cuối cùng tính tổng các chi phí mà người khách phải trả. Lưu ý: chỉ tính tiền khi phần thông tin tên khách hàng đã được nhập (nếu thông tin này chưa có thì chương trình phát sinh MessageBox cảnh báo).
- ✓ Ứng dụng có giao diện đơn giản như hình 1 bên dưới.



Bài tập 4: Từ bài tập 3 bổ sung thêm chức năng:

- ✓ Bổ sung một ListBox vào form tính tiền, ListBox này dùng để lưu trữ các thông tin tính tiền của khách hàng. Mỗi thông tin tính tiền sẽ được lưu trên một dòng trong ListBox (một item của listbox). Một item gồm hai thông tin: <tên khách hàng> - <tổng số tiền thanh toán>
- ✓ Bổ sung chức năng Lưu: cho phép lưu trữ các thông tin tính tiền của khách hàng trong một file text. File text này có định dạng mỗi dòng là một thông tin tính tiền: gồm tên khách hàng + tổng số tiền.
- ✓ Bổ sung chức năng Đọc file: Cho phép load thông tin tính tiền khách hàng từ một file lưu trữ
- ✓ Tạo chức năng tùy chọn: Cho phép người tính tiền phòng nha có thể chỉnh lại đơn giá cho từng dịch vụ. Hiện tại ứng dụng trên các đơn giá là fix, ta sẽ cải tiến lại chức năng này. Sinh viên nên tạo file chứa đơn giá từng dịch vụ, tạo form cho phép người quản lý phòng nha có thể hiệu chỉnh lại giá tiền này, lưu lại file đó, và mỗi lần ứng dụng chạy thì đọc file đó để lấy đơn giá.

Bài tập 5: Viết chương trình nhập danh sách sinh viên theo yêu cầu sau: (xem hình bên dưới).

Quy định Form hiển thị giữa màn hình. Không cho người sử dụng thay đổi kích thước Form.

Quy định việc di chuyển tab hợp lý.

Các Listbox được phép chọn nhiều mục (kết hợp giữa phím Shift, Ctrl và chuột)

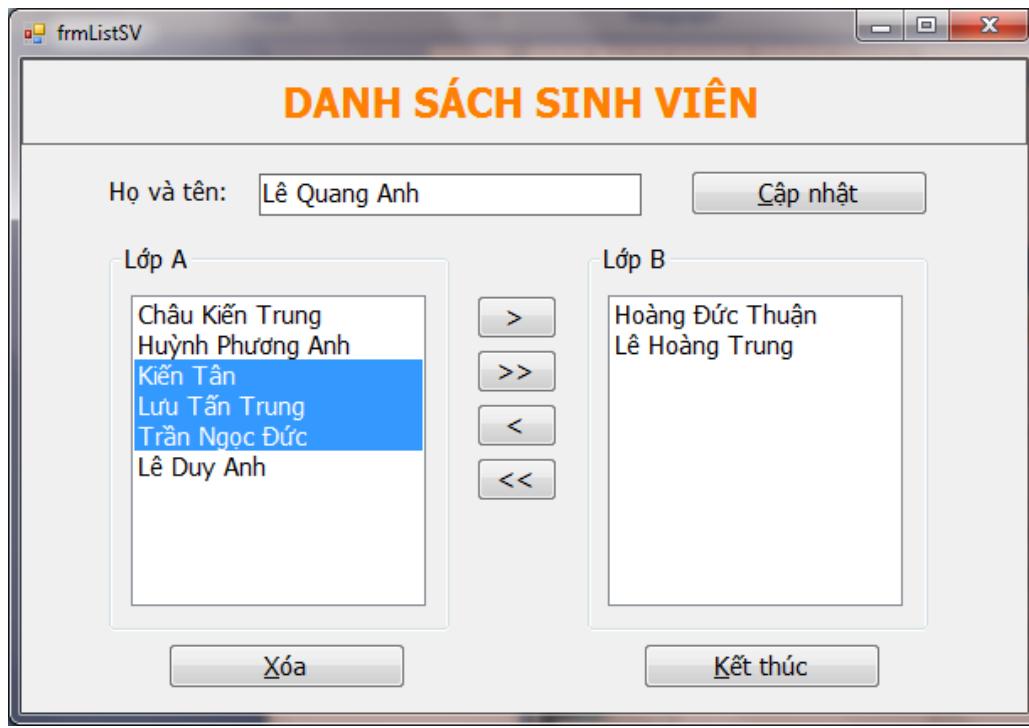
Khi người dùng nhập Họ và tên của sinh viên vào Textbox, click nút Cập Nhật (hoặc Enter) thì tên sinh viên đó sẽ được đưa vào danh sách lớp A (không chấp nhận dữ liệu rỗng).

chuyển các tên đang chọn từ Listbox trái sang Listbox phải và ngược lại.

chuyển hết toàn bộ các tên từ Listbox trái sang Listbox phải và ngược lại.

Nút Xóa: cho phép xóa các tên đang chọn trong danh sách lớp A.

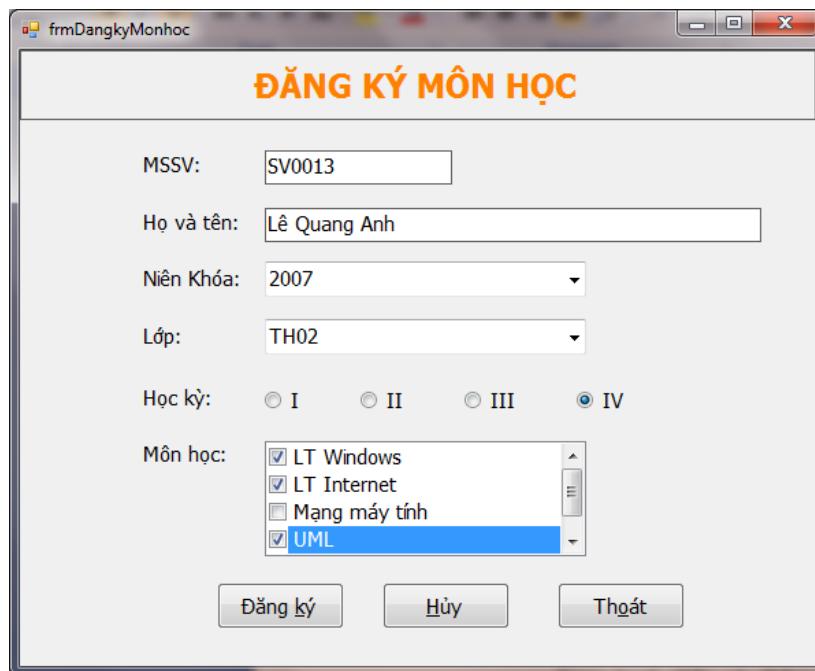
Thêm vào giao diện 1 combobox Lớp, trong đó có 2 lớp: Lớp A, Lớp B, theo đó người sử dụng có thể chọn lớp để cập nhật sinh viên vào lớp mong muốn.

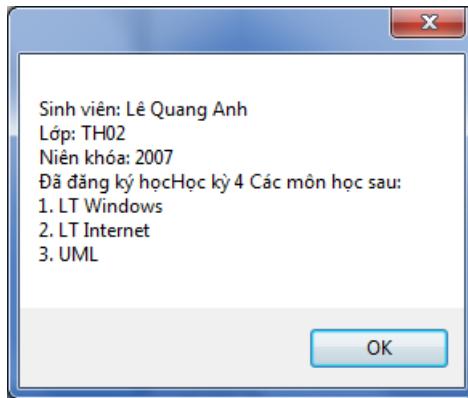


Bài tập 6: Viết chương trình cho phép sinh viên đăng ký học các môn học trong học kỳ.

Khi Form hiện lên, các ô nhập đều để trống (thiết lập tab hợp lý).

Nút Đăng ký: Hiển thị các thông tin mà sinh viên đã đăng ký lên Messagebox như hình:

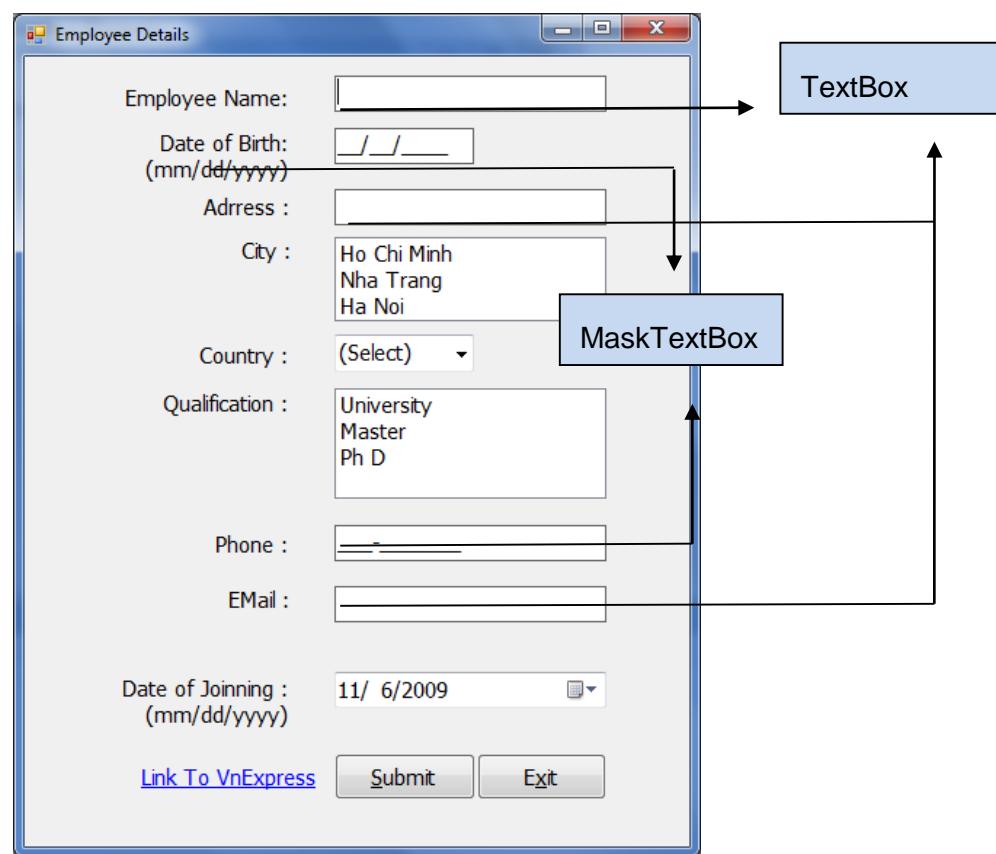




Nút Hủy: trả lại trạng thái ban đầu của Form.

Nút Thoát: thoát khỏi ứng dụng.

Bài tập 7: Thiết kế giao diện như sau



Quy định của Masktextbox Phone là 000-0000000.

ComboBox Country chỉ chứa 2 nước VietNam và Thailan (chứa 3 thành phố Pattaya,ChiengMai và Bangkok).

Khi đang nhập 1 ô mà bỏ trống và focus đến ô khác thì sẽ có thông báo lỗi và cho focus về ô cần nhập.

Khi nhấn Submit sẽ có một Messagebox hiển thị đầy đủ thông tin vừa nhập.

Bài tập 8: Viết chương trình nhập dữ liệu vào Listview như hình:



Người sử dụng nhập thông tin Last name, First name, Phone và sử dụng nút, AddName để nhập vào ListView.

Các dòng trong ListView có biểu tượng (icon) hiển thị như hình.

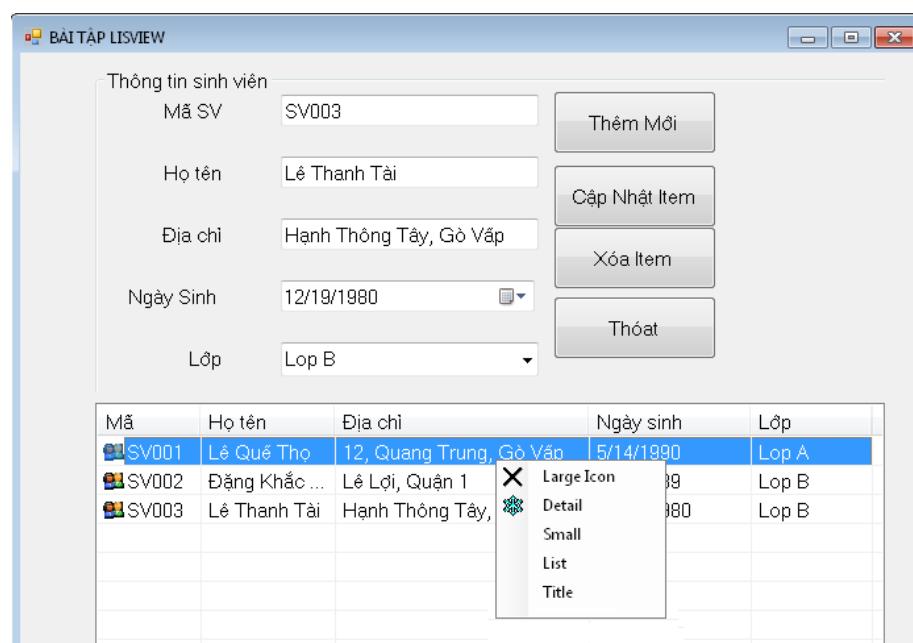
Người sử dụng có thể thay đổi chế độ view của ListView bằng menu View.

Menu FormatListview hiển thị hộp thoại chọn màu dùng để thay đổi dạng grid của ListView.

Sử dụng ListViewItem để thêm một dòng mới cho Listview.

Sử dụng ImageList để chứa thư viện icon cho Listview. Kết nối Listview với ImageList.

Bài tập 9: Thiết kế chương trình quản lý SV, cho phép nhập thông tin SV vào các TextBox như hình



Thêm vào Form hai Imagelist là ilsNho có kích thước mặc định 16 x 16, ilsLon có kích thước mặc 48 x 48 phục vụ cho ListView.

Nhấn nút Cập Nhật Item thì đưa thông tin sinh viên vào ListView theo các cột như hình.

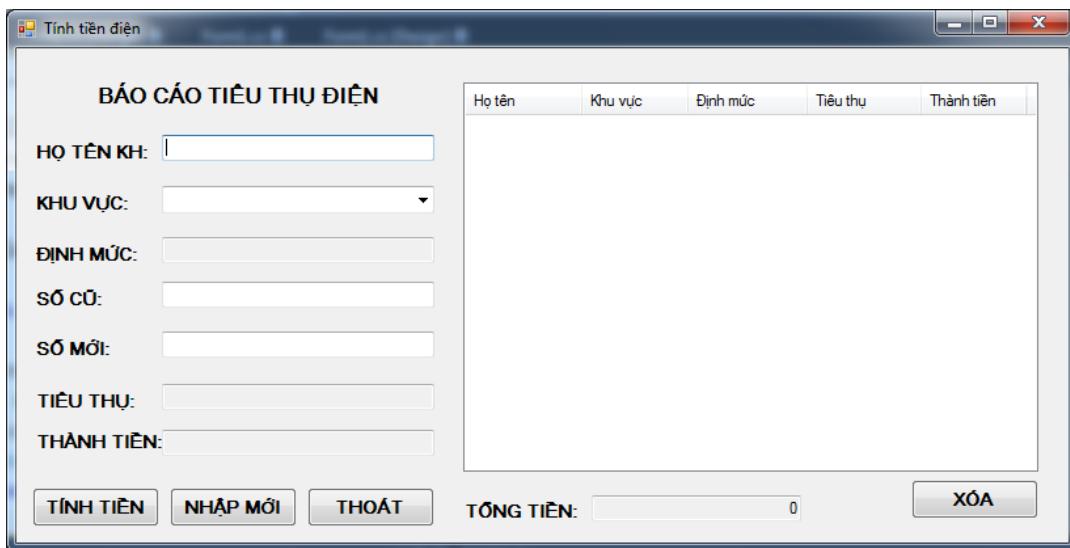
Nhấn nút Xóa Item là xóa item đang chọn trên ListView (có thể chọn nhiều). Trước khi xóa cần xác nhận đã chọn Item nào chưa, xác nhận có chắc xóa không.

Nhấn nút Thêm Mới thì xóa thông tin sinh viên đang nhập và cho phép nhập thông tin sinh viên mới.

Nếu chọn một sinh viên nào trong ListView thì hiện lại thông tin Sinh Viên đó lên các TextBox tương ứng.

Click phải vào ListView cho phép hiện menu ngữ cảnh để chọn chức năng view.

Bài tập 10: Thiết kế giao diện như sau



Thực hiện các yêu cầu sau:

Thiết lập thuộc tính cho phép chọn nhiều dòng trên ListView.

ComboBox có 3 khu vực: Khu vực 1 (định mức là 50), khu vực 2 (định mức là 100), khu vực 3 (định mức là 150). Khi chọn khu vực nào thì hiện định mức tương ứng.

Nút TÍNH TIỀN (hoặc Enter trên các TextBox): Kiểm tra dữ liệu nhập, nếu hợp lệ thì tính và xuất kết quả ra ô Tiêu thụ và Thành tiền, đồng thời thêm một dòng tương ứng vào ListView và cập nhật ô tổng tiền.

Đơn giá điện: Trong định mức là 500, ngoài định mức là 1000.

Nút NHẬP MỚI: Xoá nội dung các TextBox và Label, đồng thời đặt con trỏ vào TextBox đầu tiên

Nút XÓA: Cho phép xóa 1 dòng đang chọn trong ListView, phải xác nhận lại trước khi xoá và cập nhật lại ô tổng tiền.

Nút THOÁT (hoặc nhấn Esc): Thoát chương trình

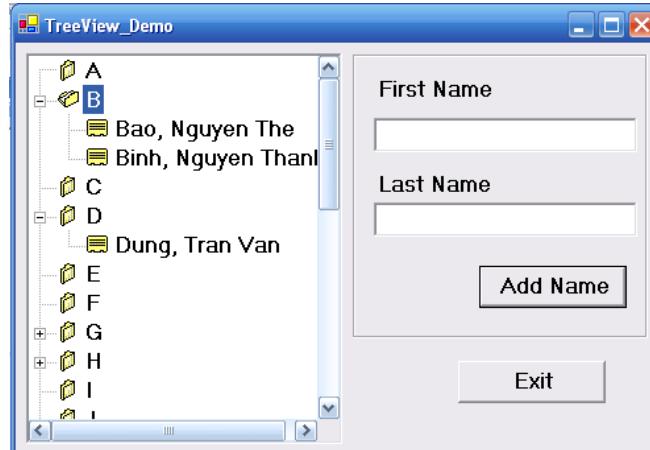
Quy định Form hiển thị giữa màn hình.

Quy định việc di chuyển tab hợp lý.

Thiết lập thuộc tính Anchor hợp lý cho các control.

Thiết lập MinimumSize cho form.

Bài tập 11: Viết chương trình nhập danh sách danh bạ với yêu cầu giao diện như hình dưới.



Khi chương trình vừa hiển thị, TreeView chưa tất cả các chữ cái từ A->Z.

Nhằm mục đích tiện lợi cho người sử dụng khi tìm tên, khi người sử dụng nhập tên của một người nào đó, chương trình sẽ đưa tên người này vào Treeview ở vị trí node có tương ứng với chữ các đầu của tên (xem hình).

Bài tập 12: Viết chương trình xem danh sách SV của Khoa Tin học như hình



Khi Forms hiện lên, TreeView hiển thị danh sách các lớp – sinh viên như hình, chưa có nút nào được chọn. Con trỏ đặt tại ô Nhập tên.

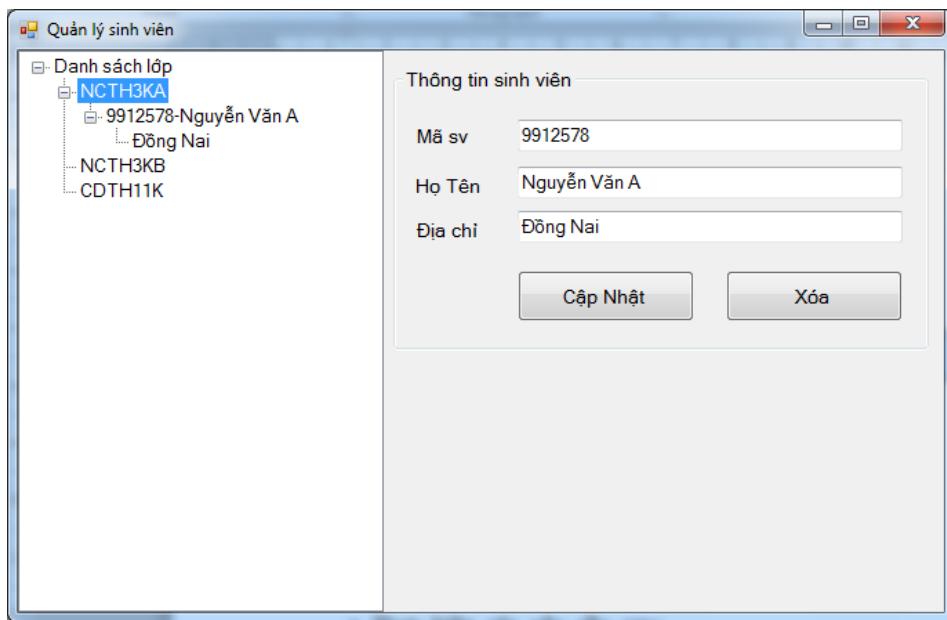
Khi người dùng chọn nút cấp Khoa, chương trình hiện toàn bộ danh sách SV thuộc Khoa vào ListView.

Khi người dùng chọn một lớp bất kỳ thì chương trình hiện toàn bộ danh sách SV thuộc lớp đang chọn vào ListView.

Khi chọn 1 SV bất kỳ thì chỉ hiện thị SV đó vào ListView.

Nút Tìm: cho phép tìm SV (trong cấp đang chọn trên TreeView) có họ tên chứa chuỗi nhập trong TextBox. Hiện kết quả ra ListView.

Bài tập 13: Thiết kế giao diện như sau



Thực hiện các yêu cầu sau:

Thiết lập HideSelection = False.

Khi Form hiện lên, đã có sẵn 1 số lớp trong danh sách lớp ở TreeView.

Nút Cập Nhật: Thêm 1 SV vào lớp đang chọn trên TreeView với nội dung các nút như hình. Trước khi thêm phải kiểm tra thông tin nhập gồm: các ô nhập không được để trống, không được trùng mã SV. Ngoài ra còn phải kiểm tra nút chọn trên TreeView có phải là nút lớp không (chỉ được thêm vào nút lớp).

Nút Xóa: cho phép xóa nút đang chọn trong TreeView, phải xác nhận lại trước khi xoá và chỉ được xoá khi chọn nút chứa mã SV.

Khi click chọn nút mã SV hoặc địa chỉ thì hiện thông tin sv đó qua các TextBox.

Quy định Forms hiển thị giữa màn hình.

Quy định việc di chuyển tab hợp lý.

Thiết lập thuộc tính Dock hợp lý cho TreeView.

Thiết lập MinimumSize cho Forms.

BÀI TẬP NHÓM LÀM PROJECT

- I. Sinh viên sẽ chọn nhóm và thực hiện đăng ký Project.
- II. Sinh viên viết testcase để kiểm tra chương trình thực hiện
- III. Viết code đúng chuẩn và nộp bài theo lịch trình của giáo viên

CHƯƠNG 5. SỬ DỤNG ADO.NET

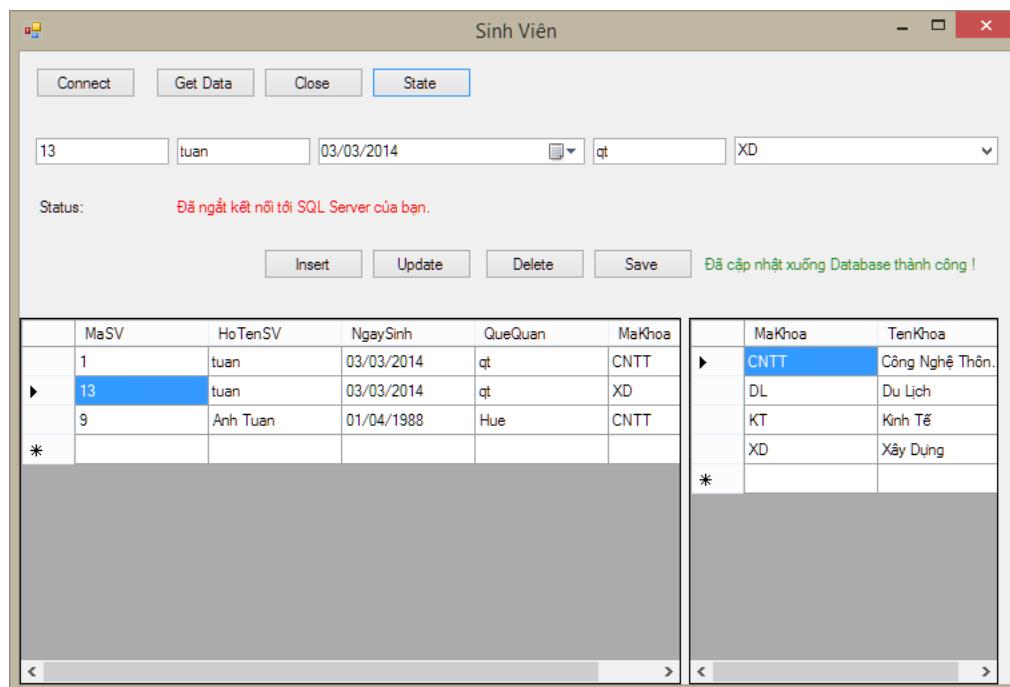
Học xong chương này, người học có thể:

- Trình bày được kiến trúc của ADO.NET.
- Trình bày được cách kết nối CSDL
- Trình bày được đối tượng SQL Connection, Command, DataAdapter.
- Thực hiện được kết nối với CSDL; Thao tác được với CSDL qua các StoreProcedure sử dụng ADO.NET
- Xây dựng được ứng dụng đơn giản có kết nối CSDL

5.1 Giới thiệu

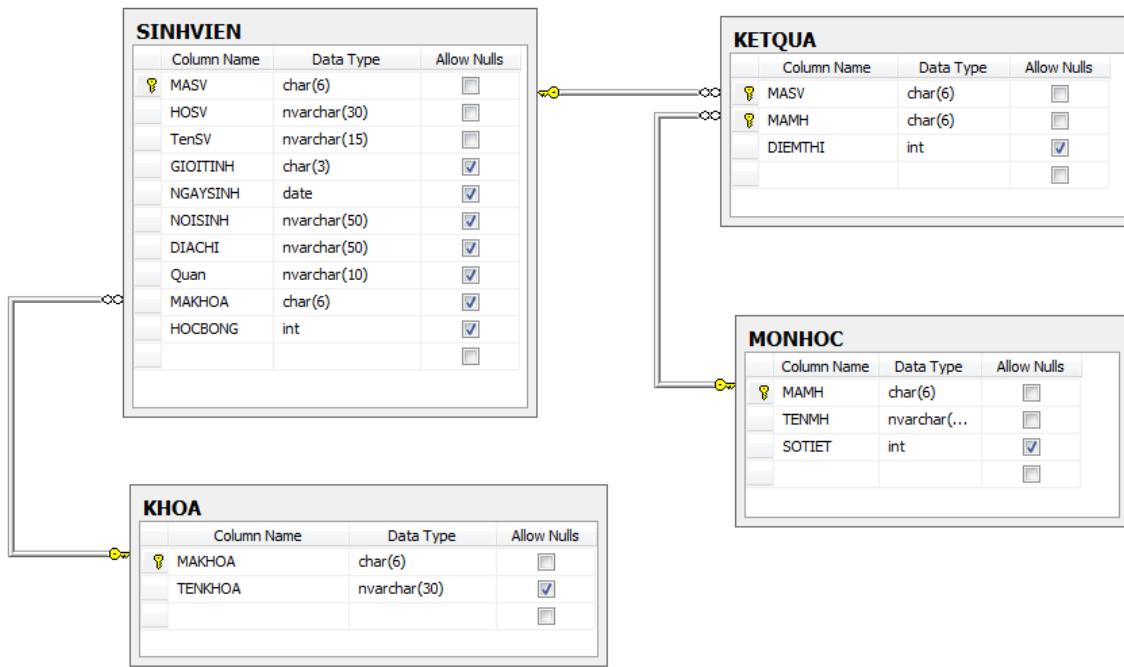
Xử lý dữ liệu là nhiệm vụ phổ biến và quan trọng của nhiều chương trình ứng dụng. Dữ liệu được truy xuất, xử lý của một chương trình ứng dụng có thể là một tập tin văn bản, tập tin các bản ghi, hay là một nguồn dữ liệu từ CSDL nào đó. .NET Framework cung cấp một lượng lớn các thành phần giao diện (Win Forms, Web Forms) hỗ trợ cho việc trình bày, nối kết (bind) và xử lý dữ liệu. Cùng với đó là nền tảng xử lý dữ liệu ADO.NET cung cấp cách thức làm việc với nhiều loại nguồn dữ liệu khác nhau một cách linh động.

Chương này trình bày cách sử dụng ADO.net để thực hiện kết nối và xử lý dữ liệu với cơ sở dữ liệu SQL.Server. Tài liệu cũng sử dụng một số cơ sở dữ liệu cơ bản như “Quản lý Sinh viên”, “Quản lý bán hàng”, ...để minh họa cho phần lý thuyết.



Hình 5.1 Giới thiệu ứng dụng

Trong chương này, một số ví dụ sẽ thực hiện trên hệ thống cơ sở dữ liệu QuanLySinhVien sau:

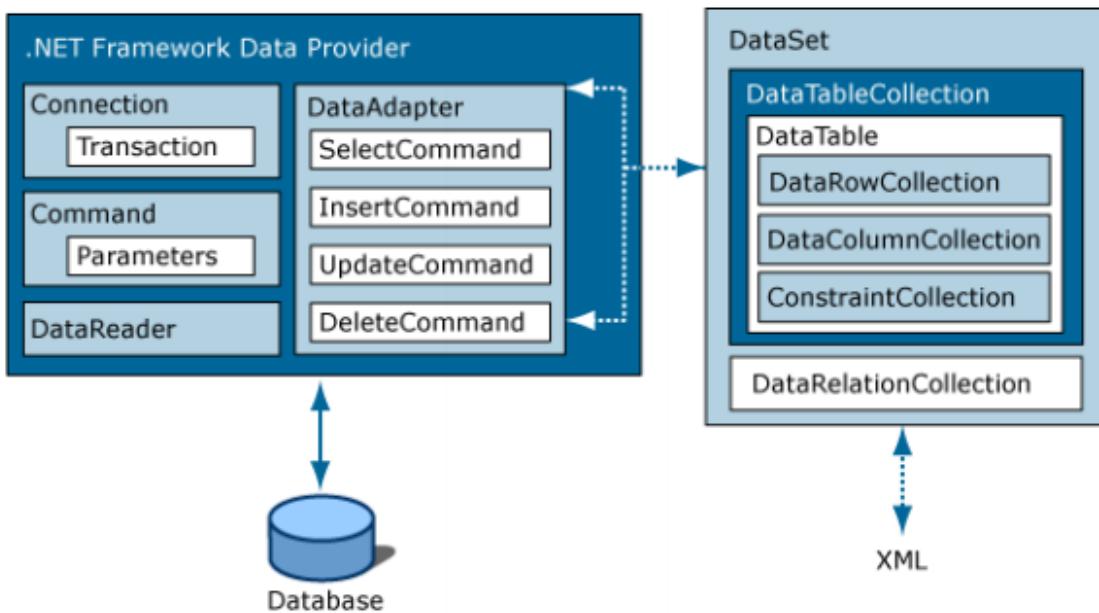


Hình 5.2 Mô hình cơ sở dữ liệu QLSV

Do tính chất quan trọng của việc xử lý dữ liệu trong một ứng dụng cùng với sự phức tạp của ADO.NET, phần tiếp theo sẽ trình bày một số điểm lý thuyết về ADO.NET.

5.2 Kiến trúc tổng quan của ADO.NET

Gồm có 2 phần chính sau:



Hình 5.3 Kiến trúc của ADO.net

Thành phần thứ nhất: .NET Framework Data Provider được thiết kế để thực hiện các thao tác kết nối, gửi các lệnh xử lý đến CSDL (thành phần này còn được gọi với một tên khác là lớp kết nối – *Connectectivity Layer*).

Trong ADO.NET, có 4 đối tượng chính với các chức năng cơ bản như sau:

- Connection: giúp thực hiện kết nối đến các CSDL
- Command: giúp truy cập đến CSDL và thực hiện các phát biểu SQL hay thủ tục lưu trữ sẵn (stored procedure) của CSDL
- DataReader: dùng để đọc nhanh nguồn dữ liệu, chỉ được duyệt tuần tự theo chiều tiến của các record
- DataAdapter: dùng để chuyển dữ liệu truy vấn được cho các đối tượng lưu trữ và xử lý (DataSet, DataTable). DataAdapter chủ yếu thực hiện các thao tác như SELECT, INSERT, UPDATE, DELETE.

Thành phần .NET Framework Data Provider cung cấp giao diện lập trình chung để làm việc với các nguồn dữ liệu. Dưới đây là bảng mô tả giao diện lập trình cùng với các lớp cơ bản của các data provider mà ADO.NET cung cấp sẵn:

| Interface | SQL Server Provider | Oracle Provider | OLEDB Provider | ODBC Provider |
|-----------------------|---------------------|-------------------|------------------|-----------------|
| IDbConnection | SqlConnection | OracleConnection | OleDbConnection | OdbcConnection |
| IDbDataAdapter | SqlDataAdapter | OracleDataAdapter | OleDbDataAdapter | OdbcDataAdapter |
| IDbCommand | SqlCommand | OracleCommand | OleDbCommand | OdbcCommand |
| IDbDataReader | SqlDataReader | OracleDataReader | OleDbDataReader | OdbcDataReader |

Hình 5.4 Các DataProvider

Để sử dụng Data Provider nào, chúng ta phải tiến hành khai báo using namespace tương ứng.

Chẳng hạn, `using System.Data.SqlClient;`

Ngoài những data provider mô tả ở bảng trên, chúng ta có thể reference đến các data provider khác không được tích hợp sẵn bởi ADO.NET trong Visual Studio .NET, chẳng hạn như data provider dùng cho MySQL, PostgreSQL, ...

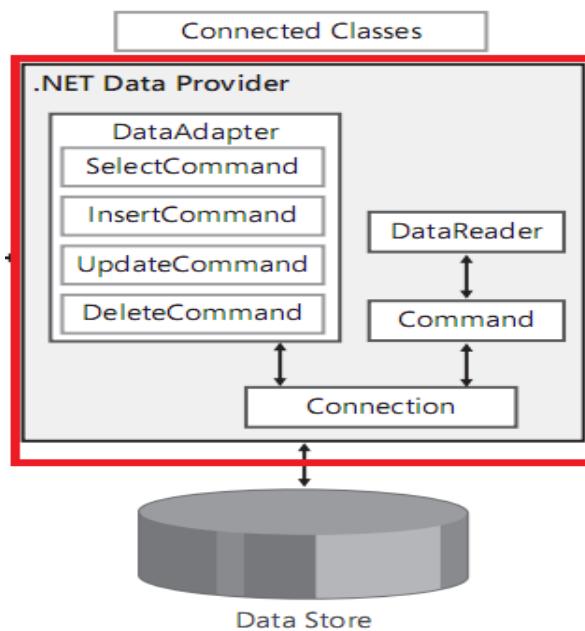
Thành phần thứ hai trong kiến trúc ADO.NET – DataSet – được xem như là container dùng để lưu trữ đối tượng liên quan đến dữ liệu như DataTable, DataRelation, DataView. Thành phần này còn được gọi là lớp không kết nối (disconected layer)

DataSet như là một CSDL thu nhỏ tại máy client, có thể chứa các đối tượng table, view, constraint, relationship giữa các table, ... Tất cả dữ liệu từ nguồn dữ liệu thực sẽ được nạp vào DataSet dưới dạng các DataTable, đó là một snapshot của nguồn dữ liệu thực. Khôi dữ liệu này sẽ được chỉnh sửa độc lập, sau đó nếu cần sẽ được cập nhật trở lại nguồn dữ liệu thực. Theo nguyên tắc này, hệ thống không cần duy trì kết nối liên tục một cách không cần thiết với nguồn dữ liệu thực trong suốt quá trình thao tác với nó.

5.3 Tổng quan về các mô hình xử lý dữ liệu trong ADO.NET

5.3.1 Mô hình Kết nối

Trong mô hình kết nối của ADO.NET, có một connection hoạt động được duy trì giữa đối tượng *DataReader* của ứng dụng và một *Data source* (nguồn dữ liệu). Một dòng dữ liệu (data row) được trả về từ data source từ *DataTable*. Điểm quan trọng nhất của mô hình kết nối đó là dữ liệu được lấy từ tập dữ liệu (các record được trả về bởi một lệnh SQL nào đó) theo kiểu từng record một cho một lần đọc, chỉ đọc (read-only), và chỉ theo một hướng tiến (forward-only).



Hình 5.5 Mô hình ngắn kết nối

Mô hình kết nối sử dụng một số lớp sau:

- `SqlConnection`: Quản lý kết nối tới Server
- `SqlCommand`: Truyền các lệnh SQL tới Server
- `SqlDataReader`: Duyệt dữ liệu trên Server, chỉ duyệt từng dòng

Các bước điển hình để làm việc với đối tượng *DataReader* là như sau:

2. Tạo đối tượng *Connection* bằng cách truyền một chuỗi *Connection String* cho hàm khởi tạo của nó.
3. Khởi tạo một biến chuỗi và gán cho câu lệnh SQL dựa theo yêu cầu dữ liệu muốn nạp về.
4. Khởi tạo một đối tượng *Command* từ với nội dung câu lệnh SQL đã xác định ở trên.
5. Tạo đối tượng *DataReader* bằng cách thực thi phương thức *Command.ExecuteReader()*. Đối tượng này sau đó sẽ được dùng để đọc kết quả của câu truy vấn mỗi dòng một lần.

Đoạn code sau minh họa các bước trên với Data Provider SqlClient. Đoạn code sẽ đọc danh sách họ tên các sinh viên trong một bảng SinhVien của cơ sở dữ liệu và hiển thị lên một điều khiển ListBox.

```

using System.Data.SqlClient;
//(1) Tao Connection
    SqlConnection conSV = new SqlConnection(chuoiKetNoi);
    cnSQL.Open();
// (2) Chuoi SQL thuc hien lay danh sach ten cac sinh vien xep tang dan theo
    // NgaySinh
    string sqlSV = "SELECT HoSV, TenSV FROM SinhVien ORDER BY NgaySinh";
// (3) Tao doi tuong Command
    SqlCommand cmdSQL = new SqlCommand(sqlSV, conSV);
//(4) Tao doi tuong DataReader
    DataReader rdr = cmd.ExecuteReader();
    while (rdr.Read())
        lstSV.Items.Add(rdr["HoTen"]);
    // Fill ListBox
    rdr.Close(); // Dong datareader

```

5.3.2 Mô hình Ngắt Kết nối



Hình 5.6 Mô hình ngắt kết nối

Trong mô hình dữ liệu được nạp bằng cách sử dụng một lệnh SQL từ nguồn dữ liệu bên ngoài vào bộ nhớ đệm tại máy client; tập kết quả được xử lý tại máy cục bộ; mọi cập nhật sau đó sẽ được truyền từ dữ liệu trong bộ nhớ ngược trở lại nguồn dữ liệu.

Mô hình “ngắt kết nối” do đối tượng kết nối chỉ được mở đủ để đọc dữ liệu từ nguồn dữ liệu và tiến hành các thao tác cập nhật. Bằng cách đưa dữ liệu về phía máy Client, tài nguyên của Server – chẳng hạn như thông tin dữ liệu Connection, bộ nhớ, thời gian xử lý – sẽ được giải phóng bớt. Tuy vậy, mô hình này cũng có nhược điểm về thời gian cần để nạp tập dữ liệu và bộ nhớ dùng để chứa dữ liệu tại máy client.

Như hình 5.6, các thành phần chính của mô hình ngắt kết nối đó là DataAdapter và DataSet. DataAdapter làm nhiệm vụ như là cầu nối giữa nguồn dữ liệu và DataSet, nạp dữ

liệu vào các bảng của DataSet và đẩy các thay đổi ngược trở lại nguồn dữ liệu. Một DataSet đóng vai trò như là một cơ sở dữ liệu quan hệ nằm trong bộ nhớ, chứa một hay nhiều DataTables, giữa các DataTable này cũng có thể có các mối quan hệ với nhau như trong một cơ sở dữ liệu quan hệ thực. Một DataTable chứa các dòng và các cột dữ liệu thường được lấy từ cơ sở dữ liệu nguồn.

Trong số các phương thức và thuộc tính của DataAdapter thì Fill() và Update() là hai phương thức quan trọng nhất. Fill() chuyển một query đến cơ sở dữ liệu và lưu tập kết quả trả về trong một DataTable nào đó; phương thức Update() thực hiện một thao tác thêm, xóa, cập nhật dựa trên những thay đổi của đối tượng DataSet. Các lệnh cập nhật thực sự được chứa trong các thuộc tính của DataAdapter. Chi tiết về DataAdapter sẽ được đề cập ở phần sau.

Để minh họa cách thức làm việc với DataAdapter và DataSet, đoạn code dưới đây giới thiệu cách tạo ra một đối tượng DataTable, nạp dữ liệu từ một cơ sở dữ liệu, và đưa nó vào một DataSet.

```
string sqlSV = "SELECT MaSV, HoSV, NgaySV, NgaySinh FROM SinhVien";  
string connStr = "Data Source=MYSERVER;Initial Catalog=qlsinhvien;  
User Id=k28;Password=k28;";  
  
// (1) Tao doi tuong Data Adapter  
SqlDataAdapter daSV = new SqlDataAdapter(sql, connStr);  
  
// (2) Tao doi tuong Dataset  
DataSet dsSV = new DataSet();  
  
// (3) Tao mot Table co ten "SinhVien" trong Dataset va nap du lieu cho no  
daSV.Fill(dsSV, "SinhVien");  
  
// (4) Hien thi danh sach ten sinh vien ra list box  
  
DataTable dtSV = ds.Tables["SinhVien"];  
  
for (int i=0; i< dtSV.Rows.Count;i++)  
{  
    DataRow row = dtSV.Rows[i];  
    lstbSV.Items.Add(row["TenSV"]);  
}
```

5.4 Làm việc với các lớp trong ADO.NET

5.4.1 Lớp Connection

Có nhiều lớp Connection trong ADO.NET – mỗi lớp tương ứng với một Data Provider bao gồm SqlConnection, OracleConnection, OleDbConnection, OdbcConnection. Mặc dù mỗi lớp có thể gồm những đặc tính riêng, nhưng các lớp này đều phải implement interface IDbConnection. Bảng dưới đây tóm tắt các thành phần được định nghĩa bởi interface này. Bao gồm các thuộc tính và phương thức:

- Các thuộc tính (Property)

| Loại | Tên | Mô tả |
|-----------------|-------------------|---|
| Property | ConnectionString | Get/Sets chuỗi kết nối đến data source. |
| Property | ConnectionTimeout | Khoảng thời gian tối đa tính bằng giây để chờ thực hiện việc kết nối đến data source |
| Property | Database | Tên CSDL ứng với Connection hiện tại |
| Property | State | Trạng thái hiện tại của Connection. Trả về một giá trị kiểu liệt kê (enumeration): Broken, Closed, Connecting, Executing, Fetching, hoặc Open |

- Các phương thức (Method)

| Loại | Tên | Mô tả |
|---------------|------------------|---|
| Method | Open | Mở một Connection. Roll back mọi thao tác đang làm dở. |
| | Close | Đóng Connection – trả Connection cho Connection Pool nếu như có sử dụng Connection Pool |
| Method | BeginTransaction | Khởi tạo một database transaction |
| Method | ChangeDatabase | Thay đổi CSDL hiện tại cho Connection đang mở. Chuỗi mô tả tên CSDL mới được truyền cho phương thức này |
| Method | CreateCommand | Tạo ra một đối tượng Command ứng với Connection |

Connection string

Thuộc tính ConnectionString xác định Data source và các thông tin cần thiết để truy xuất data source, chẳng hạn như User ID và Password, ... Ngoài những thông tin cơ bản này, Connection string còn có thể chứa các giá trị cho các trường dữ liệu đặc trưng cho Data provider. Ví dụ, Connection string cho Ms Sql Server có thể chứa các giá trị để quy định Connection Timeout và Packet Size.

Dưới đây là các ví dụ về cách thành lập chuỗi kết nối cho các Data provider thường gặp.

SqlConnection sử dụng cơ chế xác thực kiểu SQL Server:

“server=(1);database=(2);uid=(3);pwd=(4)” hoặc
“Data Source=(1);Initial Catalog=(2);User ID=(3);Password=(4)”
SqlConnection sử dụng cơ chế xác thực kiểu Windows:
“Server=(1);Database=(2);Trusted_Connection=yes”

Trong đó:

- (1) là tên/máy chủ chứa CSDL
- (2) là tên CSDL
- (3) là tên đăng nhập
- (4) là mật khẩu tương ứng.

Ví dụ:

“server=192.168.0.1; database=qlSinhvien; uid=tdc;pwd=tdc”

Các Connection String được dùng để tạo ra đối tượng Connection. Cách thực hiện thông thường là truyền chuỗi này cho hàm khởi tạo như ví dụ dưới đây:

```
string stConnection =  
    "Data Source=192.168.0.1;" +  
    "Initial Catalog=quanlySinhvien;" +  
    "User Id=tdc;" +  
    "Password=tdc";  
    SqlConnection cn = new SqlConnection(stConnection);  
    cn.Open(); //Open connection
```

Sau đây là tóm tắt các bước tạo kết nối đến CSDL SQL Server:

Bước 1: Khai báo đối tượng SqlConnection

SqlConnection connSV= null;

Bước 2: Tạo chuỗi kết nối tới máy chủ SQL Server

```
string conStrSV = "Data Source=Server; Initial Catalog=QuanLySinhvien; uid=tdc;  
Password =tdc;
```

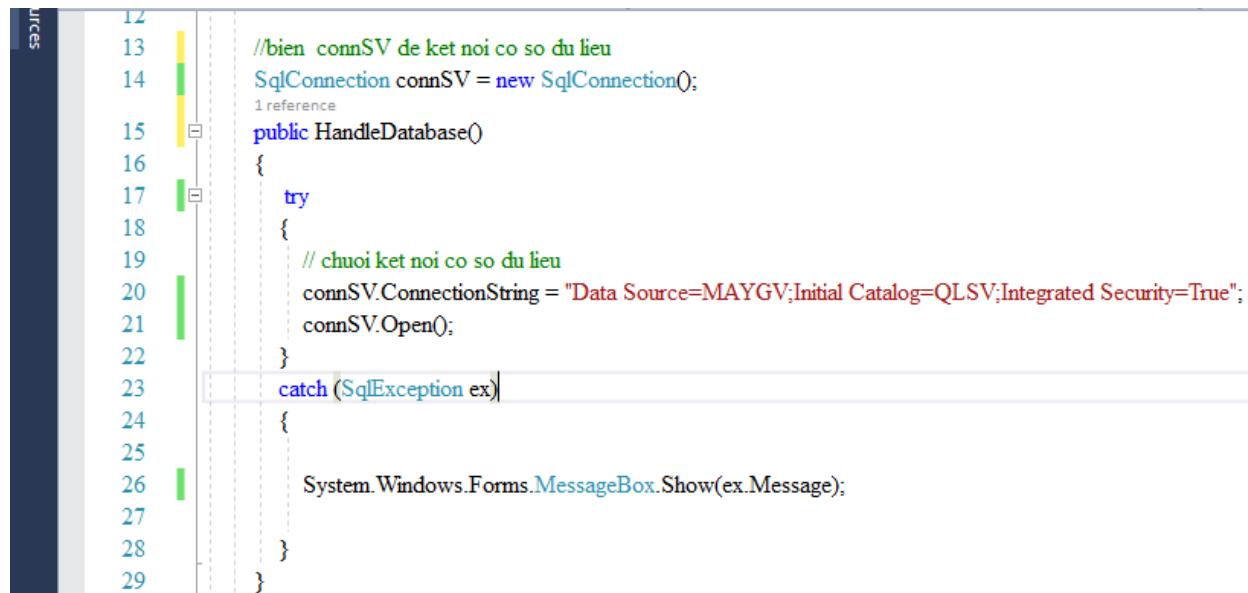
Bước 3: Tạo đối tượng Connection

connSV = new SqlConnection(conStrSV);

Bước 4: Mở kết nối qua đối tượng Connection

connSV.Open();

Đoạn code sau được viết trong Class handleDatabase.cs



```

12
13 // bien connSV de ket noi co so du lieu
14 SqlConnection connSV = new SqlConnection();
15 1 reference
16 public HandleDatabase()
17 {
18     try
19     {
20         // chuo ket noi co so du lieu
21         connSV.ConnectionString = "Data Source=MAYGV;Initial Catalog=QLSV;Integrated Security=True";
22         connSV.Open();
23     }
24     catch (SqlException ex)
25     {
26         System.Windows.Forms.MessageBox.Show(ex.Message);
27     }
28 }
29

```

5.4.2 Đối tượng Command

Sau khi một đối tượng connection được tạo ra, bước tiếp theo trong quá trình truy xuất CSDL – đối với mô hình kết nối – đó là tạo ra một đối tượng Command để gửi một query (Select) hay một action Command (thêm, xóa, sửa) đến Data Source. Có nhiều loại lớp Command ứng với các Data Provider;

Tạo và thực thi đối tượng Command

Chúng ta có thể dùng một trong nhiều hàm khởi dựng để tạo đối tượng Command một cách trực tiếp.

Lệnh SQL được gán trong thuộc tính CommandText của đối tượng Command sẽ được thực thi bằng một trong các phương thức được chỉ ra sau đây.

Các thuộc tính của đối tượng Command

- Thuộc tính CommandText: truyền vào câu lệnh SQL hoặc tên Stored Procedure
- Thuộc tính CommandType: kiểu câu lệnh SQL là StoredProcedure, TableDirect hay Text
- Thuộc tính Connection: truyền vào đối tượng Connection

Các phương thức của đối tượng Command

- Phương thức ExecuteReader: thực thi với câu lệnh SQL Select và trả về luồng dữ liệu qua đối tượng DataReader
- Phương thức ExecuteScalar: thực thi câu lệnh SQL Select và trả về một giá trị ở hàng đầu tiên và cột đầu tiên. Thích hợp với câu lệnh Select dạng thống kê như Count, Sum, AVG ...

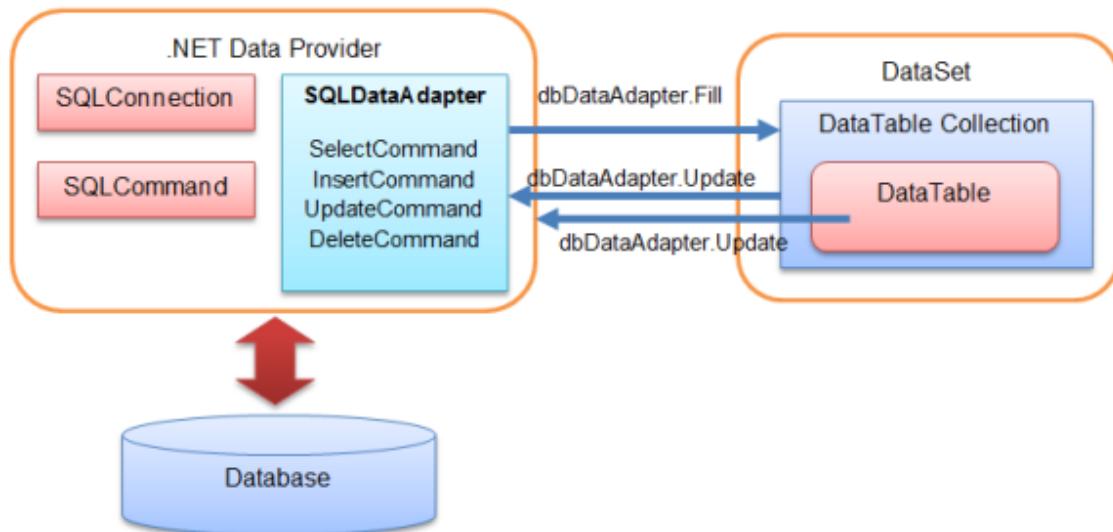
- Phương thức ExecuteNonQuery: thực thi câu lệnh SQL như Insert, Update, Delete ... và không trả về kết quả
- Phương thức ExecuteXMLReader: thực thi câu lệnh truy vấn với các câu lệnh cho XML

Đoạn code dưới đây minh họa cách tạo ra một đối tượng Command và thiết lập các thuộc tính của nó.

```
SqlConnection conn = new SqlConnection(connstr);
conn.open();
string sqlSV =
"INSERT INTO SinhVien (MaSinhVien, HoTen) VALUES (@pMaSinhVien,
@pHoTen)";
SqlCommand cmdSV = new SqlCommand();// Tạo đối tượng Command
cmd.Connection = connstr;
cmd.CommandText = sqlSV;//gán thuộc tính CommandText bằng câu truy vấn SQL
cmd.Parameters.AddWithValue ("@pMaSinhVien", 12);
cmd.Parameters.AddWithValue ("@pHoTen", "tnv spider");
```

5.4.3 Đối tượng DataAdapter

Được sử dụng như một liên kết giữa DataSource và các Table lưu trữ trong Cache.



Hình 5.7 Đối tượng DataAdapter

Trong đó:

- Phương thức Fill: đổ dữ liệu từ nguồn dữ liệu vào DataSet
- Phương thức Update: cập nhật dữ liệu từ DataSet tới nguồn dữ liệu

Ví dụ sau đây thực hiện câu lệnh truy vấn SQL Select lấy dữ liệu từ bảng SinhVien trong CSDL Quản lý Sinh viên và hiển thị dữ liệu lên DataGridView.

Sau đây là một số bước thực hiện đưa dữ liệu lên trên DataGridView:

-Bước 1: Xác định chuỗi kết nối và câu lệnh SQL cần thực hiện

```
connStr = @"Data Source=.;Initial Catalog=qlSinhVien;User ID=sa;password=123456";
// chuỗi kết nối đến CSDL

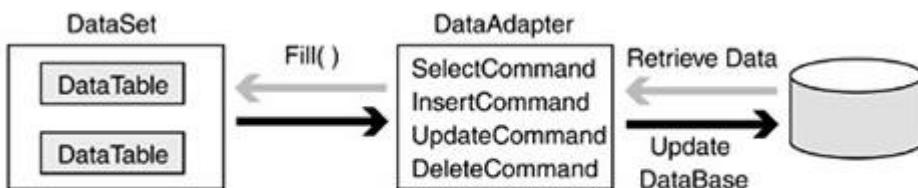
String sql = "Select * from Sinhvien"// câu lệnh select cần thực hiện
```

-Bước 2: Tạo đối tượng connection kết nối giữa ứng dụng và CSDL

```
SqlConnection connSV = new SqlConnection(); // khởi tạo một đối tượng kết nối
connSV.ConnectionString = connStr; // lấy đường dẫn đến Cơ sở dữ liệu
connSV.Open(); // mở kết nối
```

-Bước 3 : Tạo đối tượng SqlDataAdapter là cầu nối giữa Dataset và Datasource để thực hiện công việc như đọc hay cập nhật dữ liệu

```
SqlDataAdapter daSV = new SqlDataAdapter(sql,conn) ;
```



Bước 4 : Dữ liệu đọc ra từ câu lệnh Select được lưu vào 1 Datatable trong dataset

```
DataTable dtSV = new DataTable() ; // khởi tạo đối tượng datatable
daSV.Fill(dtSV) ; // fill dữ liệu vào datatable
```

Bước 5: Hiển thị dữ liệu lên DataGridView

```
dtGridViewSV.DataSource = dtSV; // gán Datasource cho DataGridView
```

```
dtGridViewSV.AutoResizeColumns(); // cǎn chinh lai chiều rộng các cột của datagridview
```

Bước 6: Đóng kết nối

```
connSV.Close();
```

Sau đây là đoạn code hoàn chỉnh trong File HandleDatabase.cs như sau:

```
12 //Bien connSV de ket noi co so du lieu
13 SqlConnection connSV = new SqlConnection();
14 1 reference
15 public HandleDatabase()
16 {
17     try
18     {
19         // chuoi ket noi co so du lieu
20         connSV.ConnectionString = "Data Source=MAYGV;Initial Catalog=QLSV;Integrated Security=True";
21         connSV.Open();
22     }
23     catch (SqlException ex)
24     {
25
26         System.Windows.Forms.MessageBox.Show(ex.Message);
27     }
28 }
29

30 /// <summary>
31 /// get all data from Table
32 /// </summary>
33 /// <param name="tableName">name of table</param>
34 /// <returns>all records in Table</returns>
35 1 reference
36 public DataTable getDataFromTable(string tableName)
37 {
38
39     DataTable dtTable = new DataTable();
40     string strSql = "Select * from " + tableName + "";
41     SqlCommand sqlCommand = new SqlCommand(strSql, connSV);
42     SqlDataAdapter dtAdapter = new SqlDataAdapter(sqlCommand);
43     dtAdapter.Fill(dtTable);
44     return dtTable;
45 }
```

Trong Form FrmKhoa.cs viết cho sự kiện Form Load để hiển thị dữ liệu lên trên DataGridView

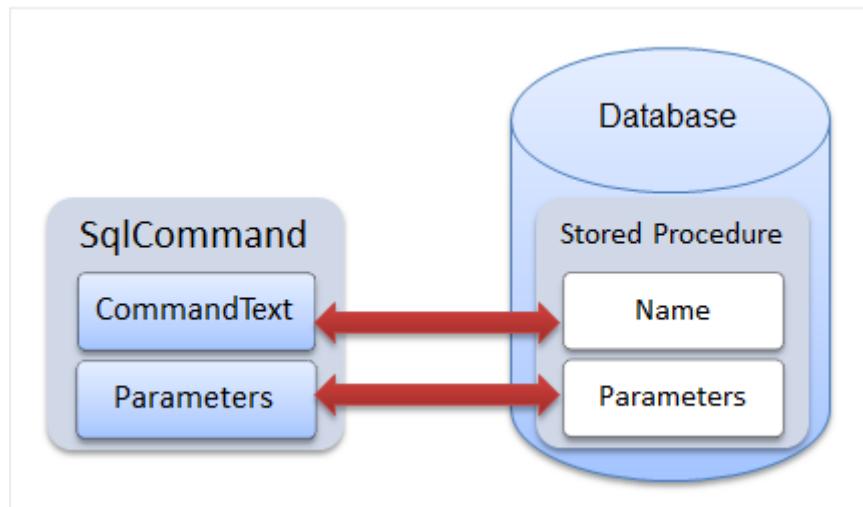
```

20  ...
21  /// get data from Table
22  /// </summary>
23  /// <param name="sender"></param>
24  /// <param name="e"></param>
25  1 reference
26
27  private void frmKhoa_Load(object sender, EventArgs e)
28
29

```

5.5 Thực thi StoredProcedure (thủ tục lưu trữ sẵn) với đối tượng Command

Một StoredProcedure là một đoạn code SQL được lưu sẵn trong CSDL và có thể được thực thi như là một Script. ADO.NET hỗ trợ việc thực thi các StoredProcedure cho các data provider OleDb , SqlClient, ODBC, và OracleClient.

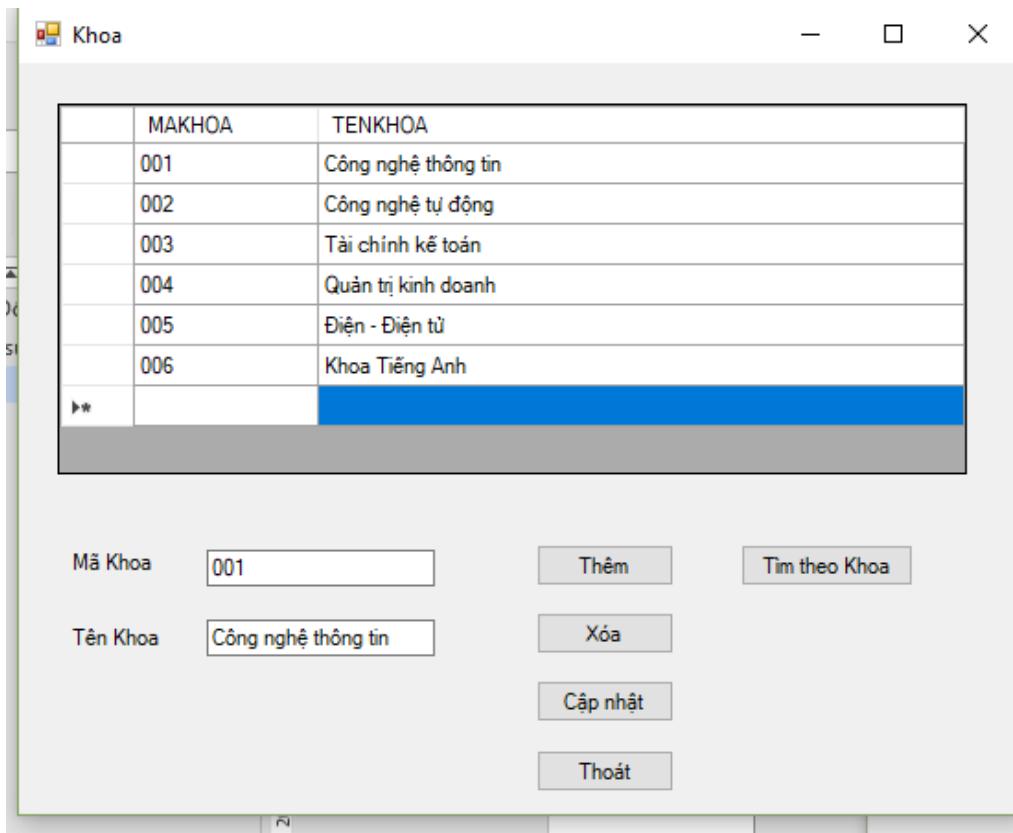


Hình 5.8 Thực thi Stored Procedure

Các bước để thực thi một storedprocedure:

- Thiết lập thuộc tính SqlCommand.CommandText thành tên của StoredProcedure;
- Thiết lập thuộc tính CommandType là CommandType.StoredProcedure;
- Thiết lập các Parameter (nếu có) cho StoredProcedure
- Thực thi phương thức ExecuteNonQuery.

Sử dụng bảng Khoa trong cơ sở dữ liệu quản lý sinh viên để xây dựng ứng dụng để hiển thị dữ liệu Khoa lên DataGridView và thực hiện các thao tác thêm, xóa, cập nhật dữ liệu.



Hình 5.9 Màn hình chương trình Quản lý Khoa

Bước 1: Thiết kế CSDL với bảng Khoa như dưới đây

| Column Name | Data Type | Allow Nulls |
|-------------|--------------|-------------------------------------|
| MAKHOA | char(6) | <input type="checkbox"/> |
| TENKHOA | nvarchar(30) | <input checked="" type="checkbox"/> |
| | | <input type="checkbox"/> |

Bước 2: Viết các thủ tục cho phép thêm, sửa, xóa một Khoa

/* Thủ tục thêm mới Khoa*/

```
CREATE PROCEDURE [dbo].[spInsertKhoa]
```

```
@MaKhoa nchar(6),
```

```
@TenKhoa nvarchar(30)
```

```
AS
```

```
INSERT INTO KHOA(MAKHOA,TENKHOA)
```

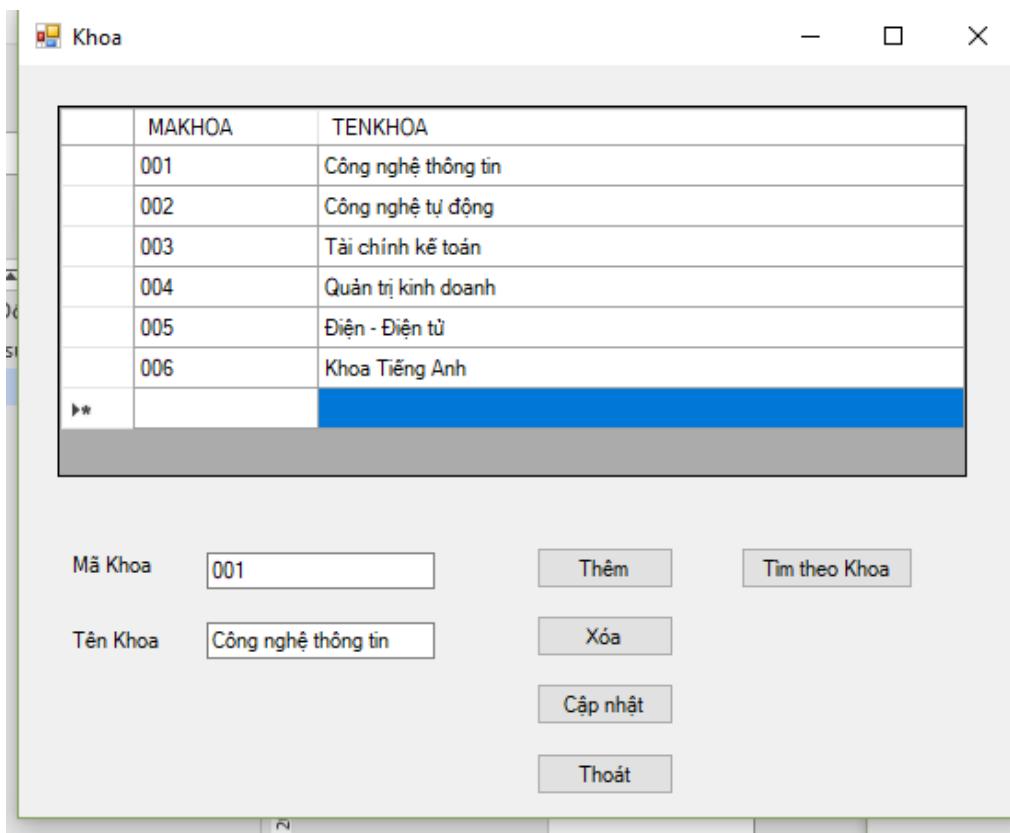
```
VALUES(@MaKhoa,@TenKhoa)
```

```
GO
```

```
/* Thủ tục xóa một Khoa*/
Create Procedure [dbo].[spdeleteKhoa]
@MaKhoa char(6)
as
begin
Delete from KHOA where MAKHOA = @MaKhoa
end

/* Thủ tục sửa thông tin 1 Khoa*/
Create Procedure [dbo].[spupdateKhoa]
@MaKhoa char(6), @TenKhoa varchar(30)
as
begin
Update Khoa Set MAKHOA=@MaKhoa, TENKHOA=@TenKhoa where
MAKHOA = @MaKhoa
end
```

Bước 3: Bước này thực hiện thiết kế giao diện như sau:



Bước 4: Lập trình hiển thị dữ liệu lên DataGridView

Đầu tiên, tạo Class để thực hiện các thao tác với CSDL: khai báo và khởi tạo đối tượng Connection. Sau đó, viết các hàm để thực hiện việc **getDataFromTable()** dùng để lấy dữ liệu đưa lên DataGridView vì hàm này còn được sử dụng lại khi ta thêm, sửa, xóa 1 bản ghi. Đồng thời, cũng viết các hàm thực hiện thao tác thêm, xóa, cập nhật dữ liệu.

```

15  public HandleDatabase()
16  {
17
18
19      try
20      {
21          // chuỗi kết nối cơ sở dữ liệu
22          conn.ConnectionString = "Data Source=.;Initial Catalog=QLSV;Integrated Security=True";
23          conn.Open();
24      }
25      catch (SqlException ex)
26      {
27
28          System.Windows.Forms.MessageBox.Show(ex.Message);
29
30      }
31  }
32  /// <summary>
33  /// get all data from Table
34  /// </summary>
35  /// <param name="tableName">name of table</param>
36  /// <returns>all records in Table</returns>
37  public DataTable getDataFromTable(string tableName)
38  {
39
40      DataTable dtTable = new DataTable();
41      string strSql="Select * from "+tableName+"";
42      SqlCommand sqlCommand = new SqlCommand(strSql, conn);
43      SqlDataAdapter dtAdapter = new SqlDataAdapter(sqlCommand);
44      dtAdapter.Fill(dtTable);
45      return dtTable;
46
47  }

```

Trên Form Khoa, chúng ta viết cho sự kiện Load Form:

```

private void frmKhoa_Load(object sender, EventArgs e)
{
    HandleDatabase dbSV = new HandleDatabase();

    dtGviewGiaovien.DataSource = dbSV.getDataFromTable("KHOA");
}

```

Bước 5: Hiển thị dữ liệu lên TextBox tương ứng khi chọn 1 dòng trong DataGridView

```

private void dtgrViewKhoa_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
    txtmaKhoa.Text=
dtgrViewKhoa.Rows[e.RowIndex].Cells[0].Value.ToString();
}

```

```
txttenKhoa.Text =  
dtgrViewKhoa.Rows[e.RowIndex].Cells[1].Value.ToString();  
  
//txtmaKhoa.Text = dtgrViewKhoa[e.ColumnIndex,  
e.RowIndex].Value.ToString();  
}
```

Bước 6: Thực thi thủ tục thêm mới một Khoa: **spinsertKhoa**. Viết thủ tục thêm 1 Khoa vào trong Class HandleDatabase như sau:

```
public int insertdataKhoaSP(string nameSP, string sMaKhoa, string sTenKhoa)  
{  
    int iResult;  
    try  
    {  
        SqlCommand sqlCmdSV = new SqlCommand(nameSP, connSV);  
        sqlCmdSV.CommandType = CommandType.StoredProcedure;  
        sqlCmdSV.Parameters.AddWithValue("@maKhoa", sMaKhoa);  
        sqlCmdSV.Parameters.AddWithValue("@tenKhoa", sTenKhoa);  
        iResult = sqlCmdSV.ExecuteNonQuery();  
    }catch(SqlException sqlEx)  
    {  
        iResult = 0;  
        return 0;  
    }  
    return iResult;  
}
```

Sau đó, viết tiếp cho nút thêm cho sự kiện Click (btnThem_Click)

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

```
string sMakhoa = txtmaKhoa.Text;
string sTenkhoa = txttenKhoa.Text;
//kiem tra du lieu hop le
if (handleData.insertdataKhoaSP("spinsertKhoa",makhoa, tenkhoa) > 0)
{
    MessageBox.Show("Thêm dữ liệu thành công");
    dtgrViewKhoa.DataSource = handleData.getDataFromtable("KHOA");
}
else
{
    MessageBox.Show("Thêm dữ liệu không thành công");
}
```

Bước 7: Tương tự cho việc thực thi các thủ tục sửa và xóa như sau:

Trong Class HandleDataBase, viết hàm Delete và hàm Update:

```
public int deleteDataKhoa(string maKhoa)
{
    SqlCommand cmdSV= new SqlCommand("sp_", conn);
    cmdSV.CommandType = CommandType.StoredProcedure;
    cmdSV.Parameters.AddWithValue("@maHD", maHD);
    int iResult= command.ExecuteNonQuery();
    return iResult;
}

public int updateDataKhoa(string maKhoa, string tenKhoa)
{
    SqlCommand command = new SqlCommand("spUpdateKhoa",
conn);
```

```
        command.CommandType = CommandType.StoredProcedure;  
  
        command.Parameters.AddWithValue("@maKhoa", maKhoa);  
  
        command.Parameters.AddWithValue("@TenKhoa", tenKhoa);  
  
        int iResult = command.ExecuteNonQuery();  
  
        return iResult;  
  
    }
```

Trong Form Khoa, viết tiếp hai hàm Cập nhật và Xóa cho sự kiện Click của 2 button Update và Delete

```
private void btnDelete_Click(object sender, EventArgs e)  
{  
  
    DialogResult dir = MessageBox.Show("Bạn muốn xóa Khoa này?", "xóa khoa",  
    MessageBoxButtons.YesNo);  
  
    String sMakhoa=txtmaKhoa.Text;  
  
    if (dir == DialogResult.Yes)  
  
    {  
  
        handleData.deleteDataKhoa(sMakhoa);  
  
        dtgrViewKhoa.DataSource = handleData.getDataFromtable("KHOA");  
  
    }  
}  
  
private void btnUpdate_Click(object sender, EventArgs e)  
{  
  
    string sMaKhoa = string.Empty;  
  
    string sTenKhoa = string.Empty;  
  
    sMaKhoa = txtmaKhoa.Text;  
  
    sTenKhoa = txttenKhoa.Text;
```

```
if (sMaKhoa!=null)
{
    DialogResult dir = MessageBox.Show("Bạn có muốn sửa thông tin Khoa
này?", "Sửa Khoa", MessageBoxButtons.YesNo);

    if (dir == DialogResult.Yes)

    {
        handleData.updateDataKhoa(sMaKhoa, sTenKhoa);

        dtgrViewKhoa.DataSource = handleData.getDataFromtable("KHOA");

    }
}

}
```

Trong chương này đã trình bày sử dụng ADO.net và sử dụng Stored Procedure để thực hiện các thao tác với Cơ sở dữ liệu bằng cách tạo các StoredProcedure(SP) trong SQL Server và xây dựng lớp xử lý dữ liệu để thực hiện các thao tác hiển thị dữ liệu lên trên DataGridView và thực hiện thao tác Thêm, Xóa, Sửa dữ liệu.

5.6 Bài tập

HỌC PHẦN LẬP TRÌNH ỨNG DỤNG BÀI TẬP THỰC HÀNH SỐ 5

I. Thông tin chung:

- Mã số bài tập : BT-LTUD - 05
- Hình thức nộp bài : Nộp qua Moodle môn học
- Thời hạn nộp bài : ... / ... /
- Nội dung: Chương 5: Sử dụng ADO.NET

Chuẩn đầu ra cần đạt:

L.O.3 Sử dụng các Control để thiết kế giao diện của chương trình

L.O.4 Viết code đúng chuẩn

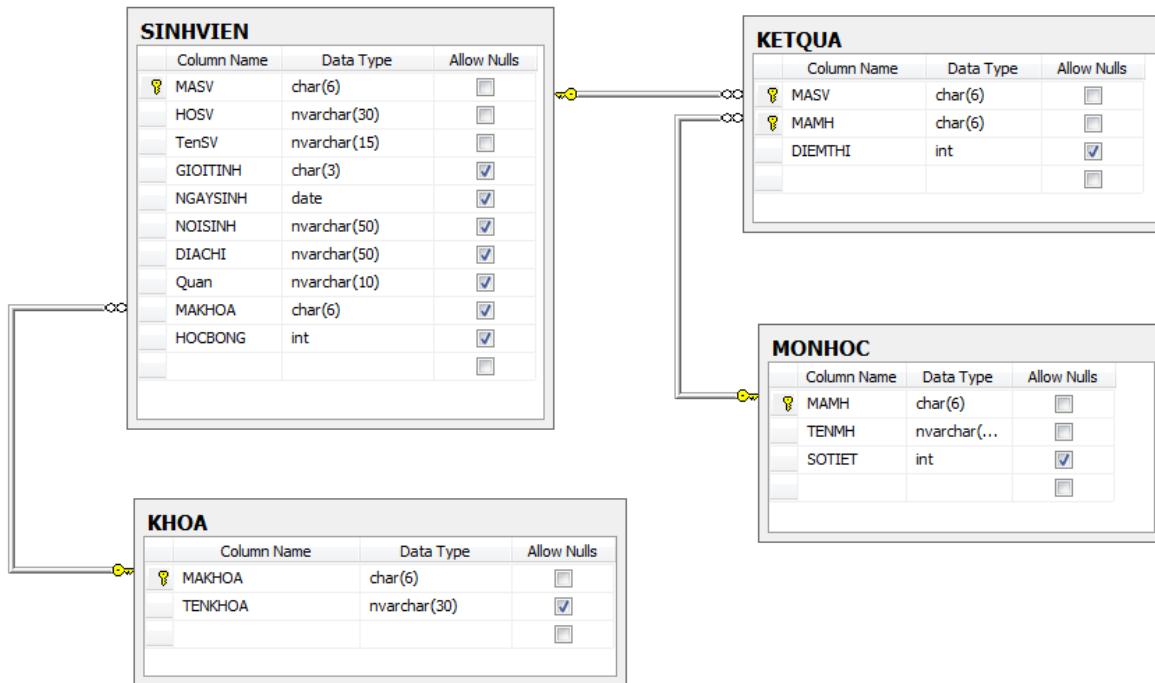
L.O.5 Sử dụng TestCase để kiểm tra phần mềm

L.O.6 Hiện thực được các chương trình vừa và nhỏ

L.O.7 Rèn luyện các kỹ năng tìm kiếm thông tin để tự giải quyết vấn đề

L.O.8 Tự tổ chức và quản lý hoạt động các nhóm

Bài tập 1: Tạo cơ sở dữ liệu Quản lý sinh viên sau:



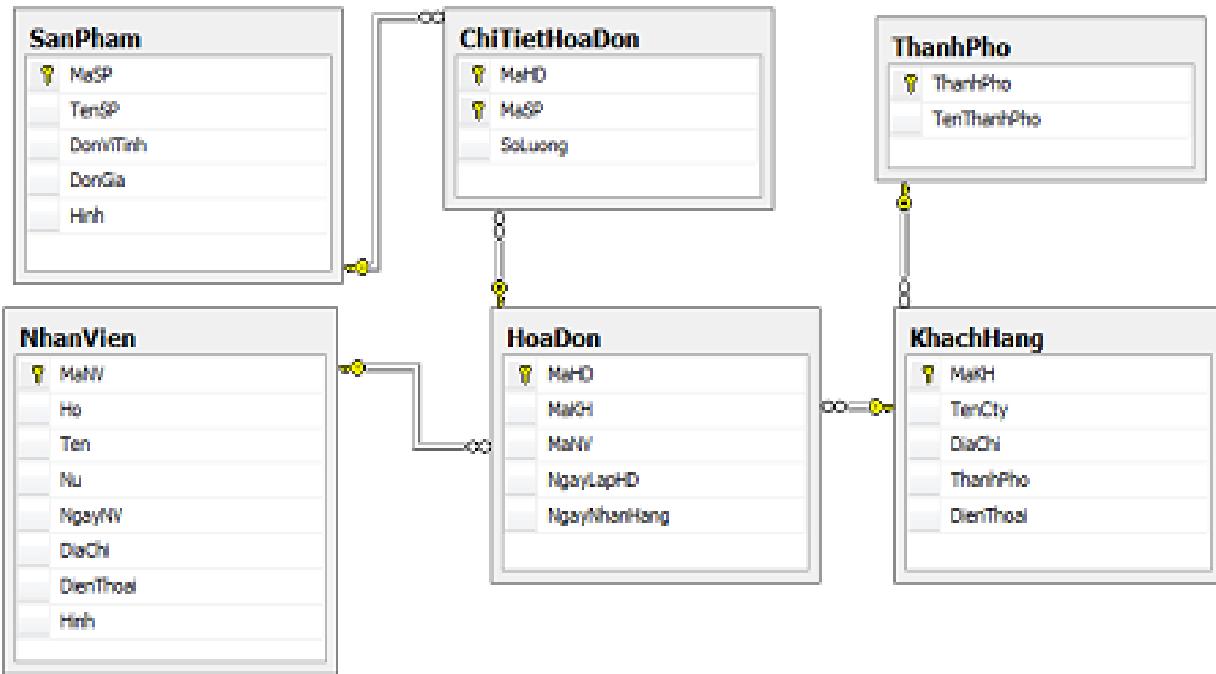
Thực hiện tạo các Storedprocedure:

1. Viết các Storedprocedure thực hiện thêm xóa sửa cho các bảng
2. Viết các Storedprocede thực hiện việc truy xuất dữ liệu các bảng

Thiết kế chương trình ứng dụng để quản lý sinh viên

1. Thiết kế Form nhập thông tin khoa có các chức năng thêm, xóa, sửa, thoát
2. Thiết kế Form nhập thông tin môn học có các chức năng thêm, xóa, sửa, thoát
3. Thiết kế Form nhập thông tin sinh viên có các chức năng thêm, xóa, sửa, thoát
4. Thiết kế Form nhập thông tin kết quả có các chức năng thêm, xóa, sửa, thoát
5. Thiết kế Form nhập thông tin tìm kiếm sinh viên, thoát
6. Thiết kế giao diện chính để gọi các Form trên thực hiện

Bài tập 2: Tạo cơ sở dữ liệu QuanLyBanHang

**Thiết kế cơ sở dữ liệu:**

1. Tạo cơ sở dữ liệu QuanLyBanHang
2. Viết các StoredProcedure thực hiện thêm xóa sửa cho các bảng
3. Viết các StoredProcedure thực hiện việc truy xuất dữ liệu các bảng
4. Viết các Function thực hiện việc truy xuất dữ liệu các bảng

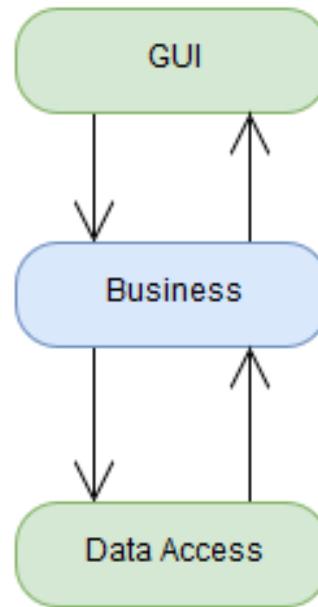
Thiết kế và xây dựng chương trình theo yêu cầu sau

1. Thiết kế Form nhập thông tin Nhân viên có các chức năng thêm, xóa, sửa, thoát
2. Thiết kế Form nhập thông tin Hóa đơn có các chức năng thêm, xóa, sửa, thoát
3. Thiết kế Form nhập thông tin Khách hàng có các chức năng thêm, xóa, sửa, thoát
4. Thiết kế Form nhập thông tin Thành phố có các chức năng thêm, xóa, sửa, thoát
5. Thiết kế Form nhập thông tin Chi tiết hóa đơn có các chức năng thêm, xóa, sửa, thoát
6. Thiết kế Form nhập thông tin Sản phẩm có các chức năng thêm, xóa, sửa, thoát
7. Thiết kế Form tìm kiếm thông tin Sản phẩm, khách hàng, thoát
8. Thiết kế giao diện chính để gọi các Form trên thực hiện

CHƯƠNG 6. MÔ HÌNH 3 LỚP

6.1 Giới thiệu tổng quan về mô hình 3 lớp

Trong phát triển ứng dụng, để dễ quản lý các thành phần của hệ thống, cũng như không bị ảnh hưởng bởi các thay đổi, người phát triển chương trình nhóm các thành phần có cùng chức năng lại với nhau và phân chia trách nhiệm cho từng nhóm để công việc không bị chồng chéo và ảnh hưởng lẫn nhau. Ví dụ trong một công ty có từng phòng ban, mỗi phòng ban sẽ chịu trách nhiệm một công việc cụ thể nào đó, phòng này không được can thiệp vào công việc nội bộ của phòng kia như Phòng tài chính thì chỉ phát lương, còn chuyện lấy tiền đâu phát cho các anh phòng Marketing thì các anh không cần biết. Trong phát triển phần mềm, người ta cũng áp dụng cách phân chia chức năng này. Chúng ta đã biết đến thuật ngữ kiến trúc đa tầng/nhiều lớp, mỗi lớp sẽ thực hiện một chức năng nào đó, trong đó mô hình 3 lớp là phổ biến nhất là **Presentation, Business Logic** và **Data Access**. Các lớp này sẽ giao tiếp với nhau thông qua các dịch vụ (services) mà mỗi lớp cung cấp để tạo nên ứng dụng, lớp này cũng không cần biết bên trong lớp kia làm gì mà chỉ cần biết lớp kia cung cấp dịch vụ gì cho mình và sử dụng nó mà thôi.



Hình 6.1 Mô hình 3 lớp

Mô hình 3-layer gồm có 3 phần chính :

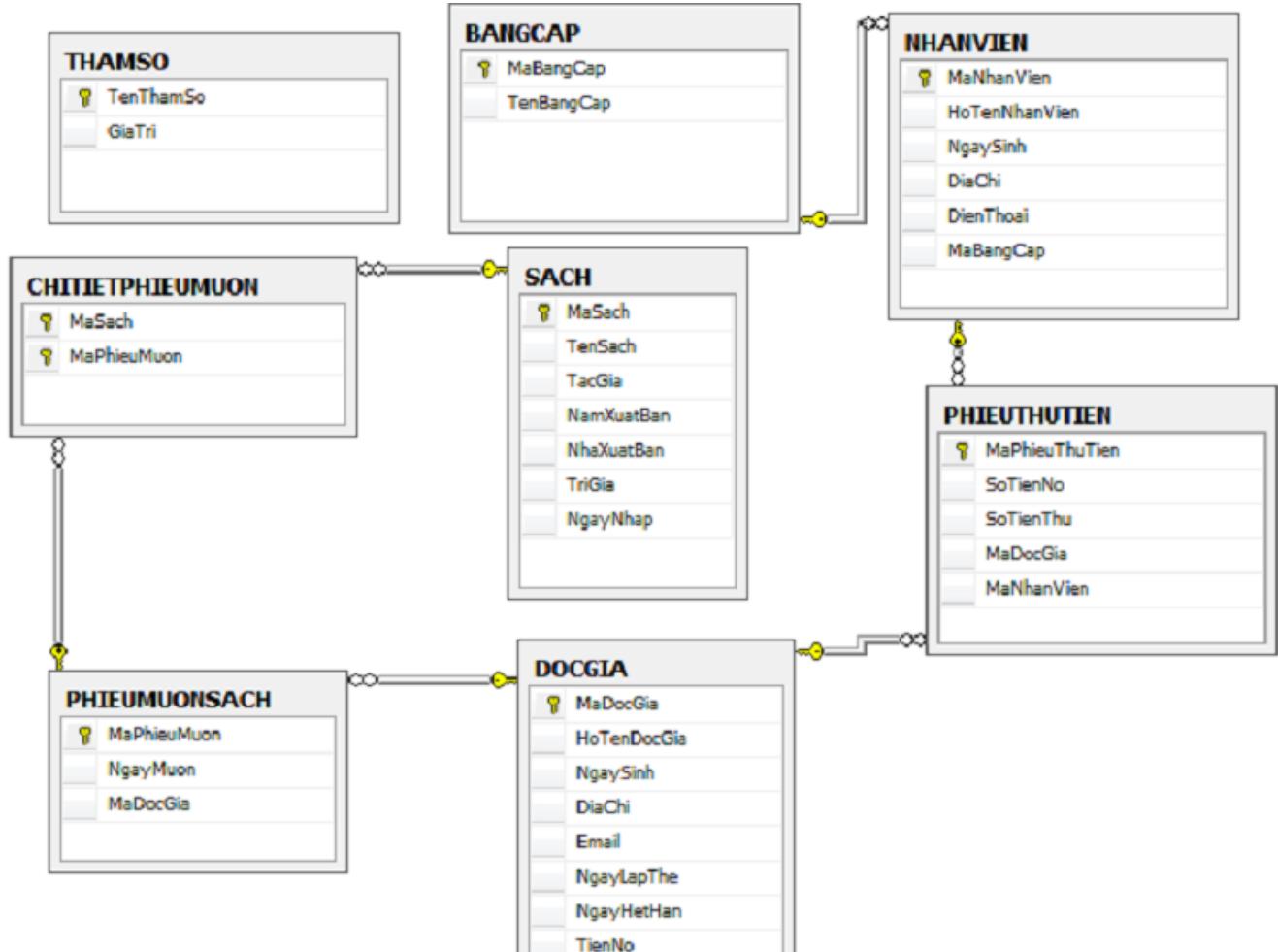
- Presentation Layer (GUI) : Lớp này có nhiệm vụ chính giao tiếp với người dùng. Nó gồm các thành phần giao diện (win form, web form,...) và thực hiện các công việc như nhập liệu, hiển thị dữ liệu, kiểm tra tính đúng đắn dữ liệu trước khi gọi lớp Business Logic Layer (BLL).
- Business Logic Layer (BLL) : Layer này phân ra 2 thành nhiệm vụ :

- Đây là nơi đáp ứng các yêu cầu thao tác dữ liệu của GUI layer, xử lý chính nguồn dữ liệu từ Presentation Layer trước khi truyền xuống Data Access Layer và lưu xuống hệ quản trị CSDL.
 - Đây còn là nơi kiểm tra các ràng buộc, tính toàn vẹn và hợp lệ dữ liệu, thực hiện tính toán và xử lý các yêu cầu nghiệp vụ, trước khi trả kết quả về Presentation Layer.
- Data Access Layer (DAL) : Lớp này có chức năng giao tiếp với hệ quản trị CSDL như thực hiện các công việc liên quan đến lưu trữ và truy vấn dữ liệu (tìm kiếm, thêm, xóa, sửa,...).

6.2 Xây dựng ứng dụng theo mô hình 3 lớp

Phần này sẽ hướng dẫn cách xây dựng ứng dụng sử dụng mô hình 3 lớp dựa trên cơ sở dữ liệu Quản lý thư viện

Tạo Database QLThuvien trên SQLServre có sơ đồ như sau :



Lệnh SQL để tạo DataBase:

```

CREATE DATABASE QLTHUVIEN

USE QLTHUVIEN

GO

--Tạo bảng Tham số

CREATE TABLE [THAMSO](
    [TenThamSo] [nvarchar](40) NOT NULL,
    [GiaTri] [int] NULL,
    CONSTRAINT [PK_THAMSO] PRIMARY KEY (TenThamSo)
)

GO

--Tạo bảng Bằng cấp

CREATE TABLE [BANGCAP](

```

```
[MaBangCap] int Identity(1,1),
[TenBangCap] [nvarchar](40) NULL,
CONSTRAINT [PK_BANGCAP] PRIMARY KEY (MaBangCap)
)
GO
--Tạo bảng Nhân viên
CREATE TABLE [NHANVIEN](
[MaNhanVien] int Identity(1,1),
[HoTenNhanVien] [nvarchar](50) NULL,
[NgaySinh] [datetime] NULL,
[DiaChi] [nvarchar](50) NULL,
[DienThoai] [nvarchar](15) NULL,
[MaBangCap] [int] NULL,
CONSTRAINT [PK_NHANVIEN] PRIMARY KEY (MaNhanVien)
)
GO
--Tạo bảng Độc giả
CREATE TABLE [DOCGIA](
[MaDocGia] int Identity(1,1),
[HoTenDocGia] [nvarchar](40) NULL,
[NgaySinh] [datetime] NULL,
[DiaChi] [nvarchar](50) NULL,
[Email] [nvarchar](30) NULL,
[NgayLapThe] [datetime] NULL,
[NgayHetHan] [datetime] NULL,
[TienNo] [float] NULL,
CONSTRAINT [PK_DOCGIA_1] PRIMARY KEY (MaDocGia)
```

```
)  
GO  
--Tạo bảng Phiếu thu tiền  
CREATE TABLE [PHIEUTHUTIEN](  
    [MaPhieuThuTien] int Identity(1,1),  
    [SoTienNo] [float] NULL,  
    [SoTienThu] [float] NULL,  
    [MaDocGia] [int] NULL,  
    [MaNhanVien] [int] NULL,  
    CONSTRAINT [PK_PHIEUTHUTIEN] PRIMARY KEY (MaPhieuThuTien)  
)  
GO  
--Tạo bảng Sách  
CREATE TABLE [SACH](  
    [MaSach] int Identity(1,1),  
    [TenSach] [nvarchar](40) NULL,  
    [TacGia] [nvarchar](30) NULL,  
    [NamXuatBan] [int] NULL,  
    [NhaXuatBan] [nvarchar](40) NULL,  
    [TriGia] [float] NULL,  
    [NgayNhap] [datetime] NULL,  
    CONSTRAINT [PK_SACH] PRIMARY KEY (MaSach)  
)  
GO  
--Tạo bảng Phiếu mượn sách  
CREATE TABLE [PHIEUMUONSACH](  
    [MaPhieuMuon] int Identity(1,1),
```

```
[NgayMuon] [datetime] NOT NULL,  
[MaDocGia] [int] NULL,  
CONSTRAINT [PK_PHIEUMUONSACH] PRIMARY KEY (MaPhieuMuon)  
)  
--Tạo bảng Chi tiết phiếu mượn  
CREATE TABLE [CHITIETPHIEUMUON](  
[MaSach] [int] NOT NULL,  
[MaPhieuMuon] [int] NOT NULL,  
CONSTRAINT [PK_CHITIETPHIEUMUON] PRIMARY KEY  
(MaSach,MaPhieuMuon)  
)  
GO  
--Tạo khóa ngoại  
GO  
ALTER TABLE [NHANVIEN] WITH NOCHECK ADD CONSTRAINT  
[FK_NHANVIEN_BANGCAP]  
FOREIGN KEY([MaBangCap])  
REFERENCES [BANGCAP] ([MaBangCap])  
ON UPDATE CASCADE  
ON DELETE CASCADE  
GO  
ALTER TABLE [NHANVIEN] CHECK CONSTRAINT  
[FK_NHANVIEN_BANGCAP]  
GO  
ALTER TABLE [PHIEUTHUTIEN] WITH CHECK ADD CONSTRAINT  
[FK_PHIEUTHUTIEN_DOCGIA]  
FOREIGN KEY([MaDocGia])  
REFERENCES [DOCGIA] ([MaDocGia])  
GO
```

```
ALTER TABLE [PHIEUTHUTIEN] CHECK CONSTRAINT  
[FK_PHIEUTHUTIEN_DOCGIA]  
  
GO  
  
ALTER TABLE [PHIEUTHUTIEN] WITH CHECK ADD CONSTRAINT  
[FK_PHIEUTHUTIEN_NHANVIEN] FOREIGN KEY([MaNhanVien])  
REFERENCES [NHANVIEN] ([MaNhanVien])  
ON UPDATE CASCADE  
ON DELETE CASCADE  
  
GO  
  
ALTER TABLE [PHIEUTHUTIEN] CHECK CONSTRAINT  
[FK_PHIEUTHUTIEN_NHANVIEN]  
  
GO  
  
ALTER TABLE [PHIEUMUONSACH] WITH CHECK ADD CONSTRAINT  
[FK_PHIEUMUONSACH_DOCGIA] FOREIGN KEY([MaDocGia])  
REFERENCES [DOCGIA] ([MaDocGia])  
ON UPDATE CASCADE  
ON DELETE CASCADE  
  
GO  
  
ALTER TABLE [PHIEUMUONSACH] CHECK CONSTRAINT  
[FK_PHIEUMUONSACH_DOCGIA]  
  
GO  
  
ALTER TABLE [CHITIETPHIEUMUON] WITH CHECK ADD CONSTRAINT  
[FK_CHITIETPHIEUMUON_PHIEUMUONSACH] FOREIGN  
KEY([MaPhieuMuon])  
REFERENCES [PHIEUMUONSACH] ([MaPhieuMuon])  
  
GO  
  
ALTER TABLE [CHITIETPHIEUMUON] CHECK CONSTRAINT  
[FK_CHITIETPHIEUMUON_PHIEUMUONSACH]  
  
GO
```

```

ALTER TABLE [CHITIETPHIEUMUON] WITH CHECK ADD CONSTRAINT
[FK_CHITIETPHIEUMUON_SACH] FOREIGN KEY([MaSach])
REFERENCES [SACH] ([MaSach])
ON UPDATE CASCADE
ON DELETE CASCADE
GO

ALTER TABLE [CHITIETPHIEUMUON] CHECK CONSTRAINT
[FK_CHITIETPHIEUMUON_SACH]

```

Tạo Project QLThuvien với Form quản lý Danh mục nhân viên frmNhanvien như sau:

The screenshot shows a Windows application window titled "frmNhanVien". The main title is "DANH MỤC NHÂN VIÊN".

Thông tin chi tiết:

| | | | |
|-----------|-------------------|------------|------------------------|
| Họ tên | NGUYỄN MINH THÀNH | Điện thoại | 0908373612 |
| Ngày sinh | 05 Tháng Tư 1983 | Địa chỉ | 41/4 CALMETTE Q1 TPHCM |

Thông tin chung:

| Họ Tên | Ngày sinh | Điện thoại | Địa chỉ |
|-------------------|-----------------------|------------|-----------------------|
| PHẠM MINH VŨ | 24/01/1980 12:00:0... | 090564... | 163/30 Thành Thái ... |
| NGUYỄN MINH THÀNH | 05/04/1983 12:00:0... | 090837... | 41/4 CALMETTE Q... |
| NGUYỄN HÀ MY | 13/04/1985 12:00:0... | 090878... | 178 NAM KỲ KHỚI ... |
| | | | |
| | | | |
| | | | |

Buttons at the bottom: Thêm, Xóa, Sửa, Thoát.

Hình 6.2 Form danh mục nhân viên

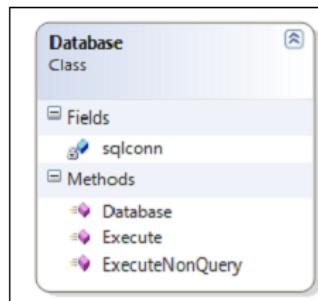
Ta xây dựng 3 lớp như sau cho NHANVIEN:

1. Lớp thao tác CSDL : Database.cs
2. Lớp xử lý nghiệp vụ : NhanVien.cs
3. Lớp xử lý giao diện : frmNhanvien.cs

Bước 1: Xây dựng lớp thao tác CSDL Database.cs

- **Chức năng:** Lớp Database đảm nhiệm việc giao tiếp với cơ sở dữ liệu cho toàn ứng dụng. Tất cả việc tương tác với CSDL dữ liệu diễn ra ở bất cứ nơi nào trong ứng dụng đều được thực hiện thông qua lớp này.
- **Mục đích:** Việc xây dựng lớp này là nhằm mang lại tính dễ bảo trì cũng như tính tiến hóa cho hệ thống. Nếu sau này cần thay đổi môi trường ứng dụng (sang Oracle, Access, Db2 ...) thì chỉ việc chỉnh sửa lớp Database này mà không cần quan tâm đến phần còn lại của ứng dụng.

Lớp Database gồm có các thành phần sau:



| Thuộc tính | Ý nghĩa |
|-----------------|---|
| sqlconn | Thuộc lớp SqlConnection |
| Phương thức | Ý nghĩa |
| Database | Hàm khởi tạo (Constructor) |
| Execute | Thực thi một câu lệnh truy vấn và trả về kết quả là một DataTable. Dùng cho các câu lệnh Select ... |
| ExecuteNonQuery | Thực thi một câu lệnh không quan tâm đến kết quả trả về. Dùng cho các câu lệnh Insert, Delete, Update ... |

Viết code cho Database.cs như sau:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
namespace QLThuvienNTT
{
    class Database
    {
        SqlConnection sqlConn; //Đối tượng kết nối CSDL
        SqlDataAdapter daNV;//Bo dieu phoi du lieu
        DataSet dsNV; //Đối tượng chứa CSDL khi giao tiếp
        public Database()
        {
            string strCnn = "Data Source=.; Database=QLthuvien;
Integrated Security=True";
            sqlConn = new SqlConnection(strCnn);
        }
        //Phuong thuc de thuc hien cau lenh strSQL truy van du lieu
        public DataTable Execute(string sqlStr)
        {
            daNV= new SqlDataAdapter(sqlStr, sqlConn);
            dsNV=new DataSet();
            daNV.Fill(dsNV);
            return dsNV.Tables[0];
        }
    }
}
```

```
}

//Phuong thuc de thuc hien cac lenh Them, Xoa, Sua

public void ExecuteNonQuery(string strSQL)

{

    SqlCommand sqlcmd = new SqlCommand(strSQL, sqlConn);

    sqlConn.Open(); //Mo ket noi

    sqlcmd.ExecuteNonQuery(); //Lenh hien lenh Them/Xoa/Sua

    sqlConn.Close(); //Dong ket noi

}

}

}
```

Bước 2 : Xây dựng lớp xử lý nghiệp vụ cho Nhân viên: Nhanvien.cs

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using System.Data.SqlClient;

namespace QLThuvienNTT

{

    class Nhanvien

    {

        Database dbTV;

        public Nhanvien()

        {

            dbTV = new Database();

        }

    }

}
```

```
}

public DataTable LayDSNhanvien()
{
    string strSQL = "Select Manhanvien, HoTenNhanVien,
Ngaysinh,Diachi,Dienthoai, TenBangcap From Nhanvien N, BANGCAP B Where
N.MaBangCap=B.MaBangCap";

    DataTable dtNV = dbTV.Execute(strSQL);

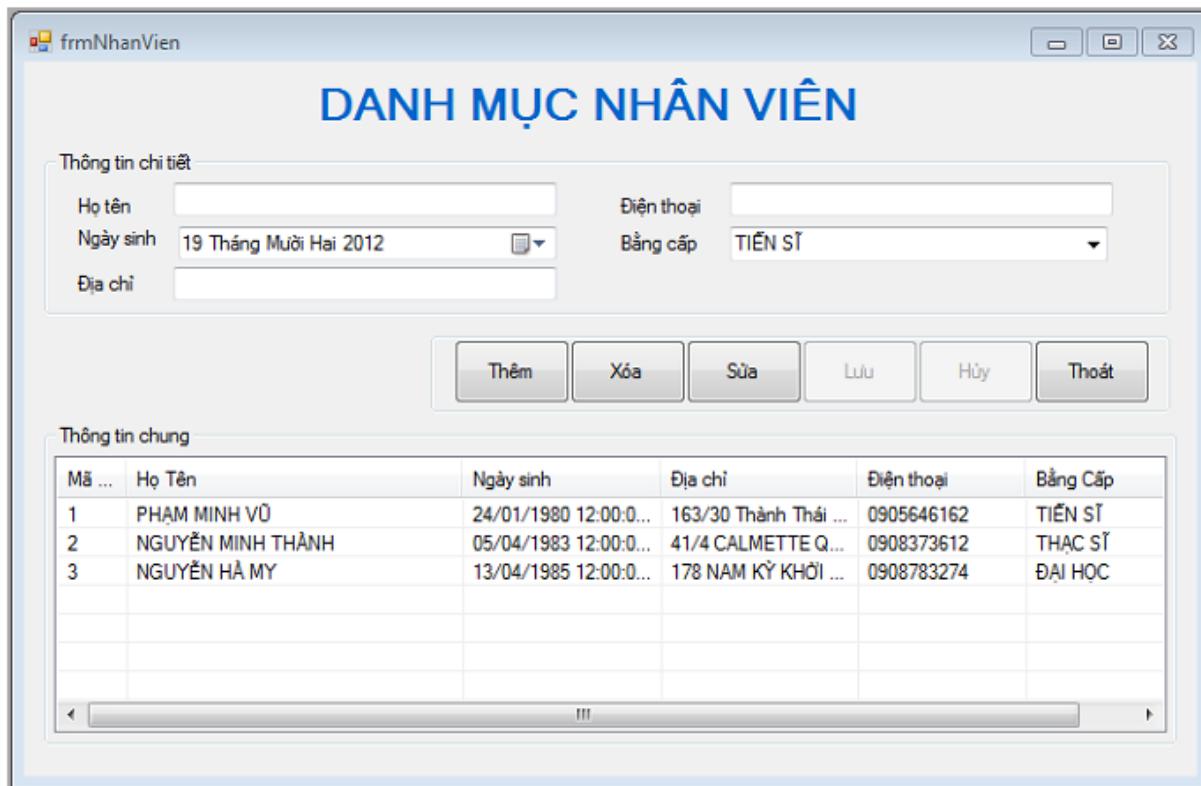
//Gọi phương thức truy xuất dữ liệu

    return dtNV;
}

}
```

Bước 3: Xây dựng lớp xử lý giao diện cho frmNhanVien.cs

Thiết kế giao diện như sau, đặt tên là frmNhanVien



Xử lý Code cho frmNhanVien:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace QLThuvienNTT
{
    public partial class frmNhanvien : Form
    {
        Nhanvien clsNhanvien = new Nhanvien();
        public frmNhanvien()
        {
            InitializeComponent();
        }
        void hienthiNhanvien()
        {
            DataTable dtNV = clsNhanvien.LayDSNhanvien();
            for (int i = 0; i < dtNV.Rows.Count; i++)
            {
                ListViewItem lviNhanvien =
                    lsvNhanVien.Items.Add(dtNV.Rows[i][0].ToString());
                lviNhanvien.SubItems.Add(dtNV.Rows[i][1].ToString());
                lviNhanvien.SubItems.Add(dtNV.Rows[i][2].ToString());
            }
        }
    }
}
```

```
    lviNhanvien.SubItems.Add(dtNV.Rows[i][3].ToString());  
    lviNhanvien.SubItems.Add(dtNV.Rows[i][4].ToString());  
    lviNhanvien.SubItems.Add(dtNV.Rows[i][5].ToString());  
}  
}  
  
private void frmNhanvien_Load(object sender, EventArgs e)  
{  
    hienthiNhanvien();  
}  
}  
}  
} //Còn tiếp cho các sự kiện khác
```

Việc xử lý đã được phân loại rõ ràng các nhiệm vụ xử lý giúp người phát triển chương trình dễ dàng quản lý chương trình hơn.

Bổ sung code đầy đủ cho lớp Nhanvien.cs

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Data;  
using System.Data.SqlClient;  
namespace QLThuvienNTT  
{  
    Database dbTV;  
    public Nhanvien()  
    {  
        dbTV = new Database();  
    }
```

```
}

public DataTable LayDSNhanvien()
{
    string strSQL = "Select Manhanvien, HoTenNhanVien,
Ngaysinh,Diachi,Dienthoai, TenBangcap From Nhanvien N, BANGCAP B Where
N.MaBangCap=B.MaBangCap";
    DataTable dtNV = dbTV.Execute(strSQL);

    //Gọi phương thức truy xuất dữ liệu
    return dtNV;
}
```

```
public DataTable layBangcap()
{
    string strSQL = "Select * from bangcap";
    DataTable dtBangcap = db.Execute(strSQL);
    return dtBangcap;
}

public void xoaNhanVien(string index_nv)
{
    string sqlXoaNhanvien= "Delete from NhanVien where MaNhanVien = " +
index_nv;
    dbTV.ExecuteNonQuery(sqlXoaNhanvien);
}

//Thêm 1 nhân viên mới

public void themNhanVien(string ten, string ngaysinh,
string diachi, string dienthoai, string index_bc)
{
```

```
        string sqlThemNV= string.Format("Insert Into NhanVien  
Values(N'{0}', '{1}', N'{2}', '{3}', {4})",  
ten, ngaysinh, diachi, dienthoai, index_bc);  
  
        dbTV.ExecuteNonQuery(sqlThemNV);  
  
    }  
  
    //Cập nhật nhân viên  
  
    public void capnhatNhanVien(string index_nv, string hoten,  
string ngaysinh, string diachi, string dienthoai, string index_bc)  
  
    {  
  
        //Chuẩn bị câu lệnh truy vấn  
  
        string str = string.Format("Update NHANVIEN  
set HoTenNhanVien = N'{0}', NgaySinh ='{1}',diachi = N'{2}',  
dienthoai = '{3}', MaBangCap = {4} where MaNhanVien = {5}",  
hoten, ngaysinh, diachi, dienthoai, index_bc, index_nv);  
  
        dbTV.ExecuteNonQuery(str);  
  
    }  
  
}
```

Bổ sung đầy đủ cho code trên form frmDMNhanvien

```
using System;  
  
using System.Collections.Generic;  
  
using System.ComponentModel;  
  
using System.Data;  
  
using System.Drawing;  
  
using System.Linq;  
  
using System.Text;  
  
using System.Threading.Tasks;
```

```
using System.Windows.Forms;
namespace QLThuvienNTT
{
    public partial class frmNhanvien : Form
    {
        public bool themmoi = false;
        Nhanvien nv = new Nhanvien();

        public frmNhanvien()
        {
            InitializeComponent();
        }

        void hienthiNhanvien()
        {
            lsvNhanVien.Items.Clear();
            DataTable dt = nv.layDSNhanvien();
            for (int i = 0; i < dt.Rows.Count; i++)
            {
                ListViewItem lvi =
                    lsvNhanVien.Items.Add(dt.Rows[i][0].ToString());
                lvi.SubItems.Add(dt.Rows[i][1].ToString());
                lvi.SubItems.Add(dt.Rows[i][2].ToString());
                lvi.SubItems.Add(dt.Rows[i][3].ToString());
                lvi.SubItems.Add(dt.Rows[i][4].ToString());
                lvi.SubItems.Add(dt.Rows[i][5].ToString());
            }
        }
}
```

```
void setNull()
{
    txtHoten.Text = "";
    txtDiaChi.Text = "";
    txtDienThoai.Text = "";
}

void setButton(bool val)
{
    btnThem.Enabled = val;
    btnXoa.Enabled = val;
    btnSua.Enabled = val;
    btnThoat.Enabled = val;
    btnLuu.Enabled = !val;
    btnHuy.Enabled = !val;
}

void hienthiBangcap()
{
    DataTable dt = nv.layBangcap();
    cboBangCap.DataSource = dt;
    cboBangCap.DisplayMember = "TenBangcap";
    cboBangCap.ValueMember = "MaBangcap";
}

private void frmNhanvien_Load(object sender, EventArgs e)
{
    setNull();
    setButton(true);
    hienthiNhanvien();
}
```

```
hienthiBangcap();  
}  
  
private void lsvNhanVien_SelectedIndexChanged(object sender,  
EventArgs e)  
{  
    if (lsvNhanVien.SelectedIndices.Count > 0)  
    {  
        txtHoten.Text =  
lsvNhanVien.SelectedItems[0].SubItems[1].Text;  
        //Chuyen sang kieu dateTime  
        dtpNgaySinh.Value =  
DateTime.Parse(lsvNhanVien.SelectedItems[0].SubItems[2].Text);  
        txtDiaChi.Text =  
lsvNhanVien.SelectedItems[0].SubItems[3].Text;  
        txtDienThoai.Text =  
lsvNhanVien.SelectedItems[0].SubItems[4].Text;  
        //Tim vi tri cua Ten bang cap trong Combobox  
        cboBangCap.SelectedIndex =  
cboBangCap.FindString(lsvNhanVien.SelectedItems[0].SubItems[5].Text);  
    }  
}  
  
private void btnThem_Click(object sender, EventArgs e)  
{  
    themmoi = true;  
    setButton(false);
```

```
txtHoten.Focus();  
}  
  
private void btnSua_Click(object sender, EventArgs e)  
{  
    if (lsvNhanVien.SelectedIndices.Count > 0)  
    {  
        themmoi = false;  
        setButton(false);  
    }  
    else  
        MessageBox.Show("Bạn phải chọn mẫu tin cập nhật",  
"Sửa mẫu tin");  
  
}  
  
private void btnHuy_Click(object sender, EventArgs e)  
{  
    setButton(true);  
}  
  
private void btnThoat_Click(object sender, EventArgs e)  
{  
    Close();  
}  
  
private void btnXoa_Click(object sender, EventArgs e)
```

```
{  
    if (lsvNhanVien.SelectedIndices.Count > 0)  
    {  
        DialogResult dr = MessageBox.Show("Bạn có chắc xóa  
không?", "Xóa bằng cấp", MessageBoxButtons.YesNo,  
MessageBoxIcon.Question);  
        if (dr == DialogResult.Yes)  
        {  
            nv.XoaNhanVien(  
lsvNhanVien.SelectedItems[0].SubItems[0].Text);  
            lsvNhanVien.Items.RemoveAt(  
lsvNhanVien.SelectedIndices[0]);  
            setNull();  
        }  
    }  
    else  
        MessageBox.Show("Bạn phải chọn mẫu tin cần xóa");  
    }  
  
private void btnLuu_Click(object sender, EventArgs e)  
{  
    string ngay = String.Format("{0:MM/dd/yyyy}",  
dtpNgaySinh.Value);  
    //Định dạng ngày tương ứng với trong CSDL SQLserver  
    if (themmoi)  
    {  
        nv.themNhanVien(txtHoten.Text, ngay, txtDiaChi.Text,
```

```
txtDienThoai.Text, cboBangCap.SelectedValue.ToString());  
    MessageBox.Show("Thêm mới thành công");  
}  
Else  
{  
    nv.CapNhatNhanVien(  
    lsvNhanVien.SelectedItems[0].SubItems[0].Text,  
    txtHoten.Text, ngay, txtDiaChi.Text, txtDienThoai.Text,  
    cboBangCap.SelectedValue.ToString());  
    MessageBox.Show("Cập nhật thành công");  
}  
HienthiNhanvien();  
setNull();  
}  
}  
}
```

6.3 Bài tập

HỌC PHẦN LẬP TRÌNH ỨNG DỤNG BÀI TẬP THỰC HÀNH SỐ 6

I. Thông tin chung:

- Mã số bài tập : BT-LTUD - 06
- Hình thức nộp bài : Nộp qua Moodle môn học
- Thời hạn nộp bài : ... / ... /
- Nội dung: Chương 6: Mô hình 3 lớp

Chuẩn đầu ra cần đạt:

L.O.3 Sử dụng các Control để thiết kế giao diện của chương trình

L.O.4 Viết code đúng chuẩn

L.O.5 Sử dụng TestCase để kiểm tra phần mềm

L.O.6 Hiện thực được các chương trình vừa và nhỏ

L.O.7 Rèn luyện các kỹ năng tìm kiếm thông tin để tự giải quyết vấn đề

L.O.8 Tự tổ chức và quản lý hoạt động nhóm

Bài tập 1: Xây dựng ứng dụng với mô hình ba lớp cho các Form sau như trong ví dụ phần 6.2 cơ sở dữ liệu quản lý thư viện :

| TÊN DANH MỤC | TÊN LỚP | CHỨC NĂNG | GHI CHÚ |
|------------------------|---------------|-----------------|---------|
| Bằng cấp | BangCap.cs | Xử lý nghiệp vụ | |
| | frmBangCap.cs | Xử lý giao diện | |
| Lập thẻ độc giả | Docgia.cs | Xử lý nghiệp vụ | |
| | frmDocGia.cs | Xử lý giao diện | |
| Tiếp nhận sách | Sach.cs | Xử lý nghiệp vụ | |
| | frmSach.cs | Xử lý giao diện | |
| ... | | | |

Bài tập 2: Sử dụng CSDL dữ liệu Quản lý bán hàng trong bài tập 2 chương 5 xây dựng ứng dụng Quản lý bán hàng với mô hình 3 lớp.