TRƯỜNG CAO ĐẲNG CÔNG NGHỆ BÁCH KHOA HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

- - - - - -

BÁO CÁO BÀI TẬP LỚN
MÔN LẬP TRÌNH ỨNG DỤNG

Đề tài:
PHẦN MỀM QUẢN LÝ SẢN PHẨM

Giáo viên hướng dẫn:Thầy Trần Thanh

Nhóm thực hiện: Nhóm 5 – Lớp K20-IT04

Phạm Như Huấn

Nguyễn Văn Toàn

Nguyễn Văn Linh

Nguyễn Huy Nhu

Nguyễn Huỳnh Đức

Năm học 2021-2022

LỜI NÓI ĐẦU

.NET là nền tảng lập trình ứng dụng tiện lợi, cung cấp cho người lập trình môi trường làm việc trực quan, dễ dàng trong việc phát triển các ứng dụng, thuận lợi trong việc kết nối và làm việc với cơ sở dữ liệu. Winform là một kiểu ứng dụng được xây dựng trên môi trường .NET, được các lập trình viên và các nhà nghiên cứu ứng dụng rộng rãi vào công việc. Hiện nay Lập trình Windows Form với C#.NET là một trong những học phần quan trọng đối với các bạn sinh viên chuyên ngành Công nghệ thông tin trong các trường đại học và cao đẳng. Với mục đích cung cấp cho các bạn sinh viên những kiến thức khá toàn diện về lập trình ứng dụng Windows Form với C# 5.0, phiên bản mới nhất và nền tảng của C# về lập trình cơ sở, lập trình hướng đối tượng, cần thiết cho lập trình ứng dụng Windows Form C#; nhóm tác giả giảng viên, Trường Đại học Duy Tân phối hợp với Nhà xuất bản Thông tin và Truyền thông xuất bản cuốn giáo trình "***Lập trình Windows Form với C#.NET***

Ngày nay , Công Nghệ Thông Tin đã và đang đóng vai trò quan trọng trong đời sống kinh tế , xã hội của nhiều quốc gia trên thế giới , là một phần quan trọng không thể thiếu trong xã hội ngày càng hiện đại hóa . Vì vậy , việc tin học hóa vào một số lĩnh vực là hoàn toàn có thể và phù hợp với xu hướng hiện nay . Xuất phát từ nhu cầu thực tế đó , trong công việc kiểm tra thông tin sản phẩm , việc quản lý sản phẩm là một việc không thể thiếu . Nhằm thay thế một số công việc mà trước đó phải thao tác trên bằng tay trên giấy tờ đạt hiệu quả không cao, mất nhiều thời gian . Vì vậy , chúng em đã thực hiện báo cáo với đề tài " Xây dựng hệ thống quản lý sản phẩm ''
Do trong khuôn khổ ngắn , trình độ chuyên môn , kinh nghiệm và kiến thức bản thân còn hạn chế, nên chúng em rất mong được sự góp ý của thầy và các ban trong lớp , để đề tài nghiên cứu của chúng em ngày càng hoàn thiện hơn và được ứng dụng trong thực tế

PHẦN 1 : GIỚI THIỆU CÁC CÔNG CỤ SỬ DỤNG

1.HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

+Cơ Sở Dữ Liệu SQL SERVER

SQL Server là một hệ thống quản trị cơ sở dữ liệu quan hệ (Relational Database Management System (RDBMS)) sử dụng Transact – SQL để trao đổi dữ liệu giữa Client computer và SQL Server computer . Một RDBMS bao gồm databases , database engine và các ứng dụng để quản lý dữ liệu và các bộ phận khác nhau trong RDBMS .

SQL Server được tối ưu để có thể chạy trên môi trường cơ sở dữ liệu rất lớn (Very Large Database Environment) lên đến Tera-Byte và có thể phục vụ cùng lúc cho hàng ngàn user. SQL Server có thể kết hợp "ăn ý" với các server khác

như Microsoft Internet Information Server (IIS), E-Commerce Server, Proxy Server....

MS SQL có thể làm những gì? :

• Thực thi các truy vấn đối với CSDL

• Lấy dữ liệu từ CSDL Chèn các bản ghi vào CSDL

• Cập nhật các bản ghi trong CSDL

• Xoá các bản ghi từ CSDL

• Tạo ra CSDL mới

• Tạo ra các bảng mới trong CSDL

• Tạo ra các thủ tục ru trữ (Stored procedures) trong 1 CSDL

• Tạo được các Vew (bảng hiển thị hình thức)

• Thiết lập quyền truy cập vào các bằng các thủ tục và các vew.

2.CÔNG CỤ LẬP TRÌNH VISUAL STUDIO C#

+Giới thiệu về ngôn ngữ lập trình C#

Ngôn ngữ C# khá đơn giản, chỉ khoảng 80 từ khóa và hơn mười mấy kiểu dữ liệu được xây dựng sẵn. Tuy nhiên, ngôn ngữ C# có ý nghĩa cao khi nó thực thi những khái niệm lập trình hiện đại. C# bao gồm tất cả những hỗ trợ cho cấu trúc, thành phần component, lập trình hướng đối tượng.

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

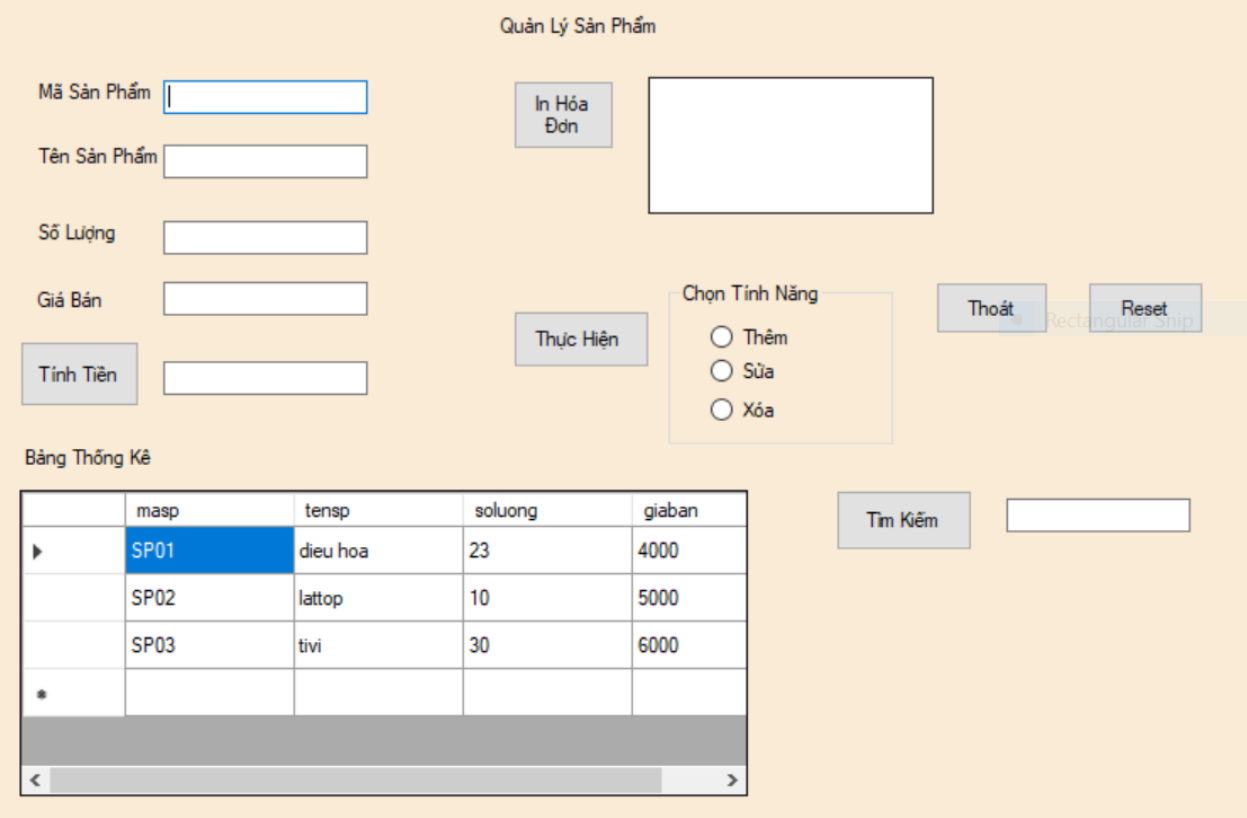
Trong ngôn ngữ C# mọi thứ liên quan đến khai báo lớp điều được tìm thấy trong phần khai báo của nó. Định nghĩa một lớp trong ngôn ngữ C# không đòi hỏi phải chia ra tập tin header và tập tin nguồn giống như trong ngôn ngữ C++. Hơn thế nữa, ngôn ngữ C# hỗ trợ kiểu XML, cho phép chèn các tag XML để phát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

+Giới thiệu về Windown Form

Windows Forms hay viết tắt là WinForm là thuật ngữ chỉ việc phát triển các ứng dụng giao diện người dùng bằng cách sử dụng các thành phần xây dựng sẵn (buil in Component) còn được gọi là các điều khiển. Hay nói cách khác Windows Forms là một API (Application Programming Interface) cho phép tạo GUI (Graphical User Interface) cho các ứng dụng chạy trên desktop.

Các điều khiển này dùng để hiển thị thông tin cho người dùng cũng như cho người dùng nhập thông tin vào Windows Forms ra đời đáp ứng nhu cầu tạo ứng dụng nhanh (Rapid Application Development).

PHẦN 2 :THIẾT KẾ GIAO DIỆN HỆ THỐNG



SQL SeVer

	masp	tensp	soluong	giaban
1	SP01	dieu hoa	23	4000
2	SP02	lattop	10	5000
3	SP03	tivi	30	6000

PHẦN 3 : ƯU, NHƯỢC ĐIỂM VÀ HƯỚNG PHÁT TRIỂN CỦA HỆ THỐNG

1.Ưu điểm

- Hệ thống được xây dựng gọn nhẹ, dễ sử dụng.

- Hệ thống ràng buộc dữ liệu được đảm bảo.

- Giao diện thân thiện với người sử dụng.

2. Nhược điểm

- Cách tổ chức dữ liệu và kỹ thuật lập trình chưa tốt
- Bắt lỗi hạn chế, chưa hoàn thiện
- Các chức năng của chương trình chưa linh động, có thể gây khó khăn khi sử dụng.
- Chưa bảo mật được dữ liệu.

3. Hướng Phát Triển

- Khả năng xử lý được tất cả các sự kiện, các lỗi ngoài ý muốn tốt hơn của chương trình và dùng thao tác lên chương trình.
- Hoàn thiện tốt hơn về lập trình C# và ràng buộc dữ liệu.
- Nâng cao tính linh động của chương trình
- Thêm các chức năng mới để đáp ứng điều kiện của người dùng.
- Nâng cao kỷ thuật lập trình và hoàn chỉnh các thành phần còn thiếu theo hướng chuyên nghiệp, chạy thử, khả năng đưa vào áp dụng thức tế khả quan.
- Bảo mật dữ liệu tốt hơ
-Windows Form cho phép người phát triển tạo ra các giao diện người dùng sử dụng các thành phần khác nhau (components). Các thành phần này còn được gọi là các điều khiển (controls). Những điều khiển này cho phép chúng ta thu thập thông tin từ người dùng cũng như trình bày các thông tin để người dùng có thể xem.

- Một Form được chạy trên một máy tính cục bộ (local machine) và một form có thể truy cập đến các tài nguyên khác nhau như bộ nhớ, các thư mục, các tệp tin, các cơ sở dữ liệu…

- Do đó Windows Form phù hợp cho các ứng dụng desktop như các ứng dụng quản lý thông tin, các ứng dụng tương tác trực tiếp với người dùng.

- Vai trò của Windows Form:

- Các Form có thể chứa các điều khiển (các thành phần) khác nhau.
- Xử lý dữ liệu được nhập bởi người dùng.
- Hiển thị (trình bày) các thông tin tới người dùng.
- Kết nối đến các nguồn CSDL khác nhau trên các máy tính cục bộ hoặc máy tính khác

MỤC LỤC

NỘI DUNG

Nội dung giáo trình gồm 3 chương, cụ thể như sau:
Chương 1. Microsoft .Net và C#
Chương 2. Cơ bản về C#
Chương 3. Lập trình hướng đối tượng với C#
Cuốn giáo trình sẽ mang lại cho các bạn sinh viên, những người yêu thích lập trình những kiến thức nền tảng về dịch vụ ADO.NET truy cập nhiều nguồn dữ liệu thông dụng hiện nay như Microsoft SQL, Microsoft Access, Oracle, MySQL, DB2 và XML; Kiến thức về dự án Excel Workbook mở rộng chức năng cho bảng tính Microsoft Excel sử dụng dịch vụ Excel Interop. Ngoài ra, giáo trình còn cung cấp kiến thức lập trình song song sử dụng tuyến đoạn, lập trình ứng dụng sử dụng mô hình đối tượng thành phần COM và thành phần phân tán mới nhất của Microsoft như Web service, WCF

Chương 1
MICROSOFT .NET VÀ C#

Chương này giới thiệu tổng quan về Microsoft .NET, ngôn ngữ C#, ý nghĩa và sức mạnh của Microsoft .NET và ưu điểm của ngôn ngữ C# so với các ngôn ngữ khác. Ngoài ra chương này còn trình bày về cách thức bắt đầu lập trình với C# trong môi trường Visual Studio .NET như thế nào và cách thức biên dịch và thực hiện ứng dụng Console C# đơn giản.

Tổng quan về Microsoft .NET
- Lịch sử phát triển của Microsoft .NET
- Đặc điểm và kiến trúc .NET framework: Common language runtime và Thư viện .NET framework

Microsoft Visual C#
- Giới thiệu ngôn ngữ C#
- Các đặc trưng của C#
- Biên dịch và thực hiện một chương trình C# đơn giản

1.1. TỔNG QUAN VỀ MICROSOFT .NET
1.1.1. Lịch sử phát triển của Microsoft .NET

Vào đầu năm 1998, sau khi hoàn thành phiên bản Version 4 của IIS (Internet Information Server), công ty Microsoft bắt đầu xây dựng một kiến trúc mới trên nền tảng IIS và đặt tên là NGWS (Next Generation Windows Services).

Sau khi Visual Basic ra đời vào cuối 1998, dự án kế tiếp là Visual Studio 7 được xác nhập vào NGWS. Đến tháng 11/2000 thì Microsoft đã phát hành phiên bản Beta của .NET 1.0.

Tháng 2/2002, phiên bản 1.0 của .NET framework ra đời tích hợp cùng với bộ Visual Studio .NET, được mặc định cài đặt trong Windows XP, tiếp theo phiên bản 1.1 phát hành với Visual Studio

.NET 2003 và Windows Server 2003.
Sau đó, phiên bản 2.0 của .NET framework tích hợp cùng với bộ Visual Studio .NET 2005, và Windows Server 2003.
Phiên bản 3.0 của .NET framework bao gồm với Visual Studio.NET 2008 và Windows Server 2008, Windows Vista, tiếp theo, phiên bản 3.5 với Windows 7 và Windows Server 2008.
Ngày 12 tháng 4 năm 2010, NET framework 4.0 được phát hành cùng với Visual Studio 2010.
Gia đình .NET framework cũng bao gồm hai phiên bản sử dụng cho điện thoại di động và hệ thống nhúng. Một phiên bản .NET compact framework, bao gồm trong Windows CE, một hệ điều hành nguồn mở 32 bit, sử dụng cho các thiết bị thông minh như các vi điều khiển dùng trong công nghiệp, các thiết bị đầu cuối như camera, điện thoại di động, các thiết bị giải trí gia đình. Phiên bản thứ hai là .NET micro framework nhằm phát triển các hệ thống nhúng trên các thiết bị nhỏ, hạn chế tài nguyên.

1.1.2. Đặc điểm và kiến trúc .NET framework

Microsoft .NET là nền tảng cho việc xây dựng và thực thi các ứng dụng từ các ứng dụng truyền thống giao diện người dùng ký tự CUI (Character User Interface), các ứng dụng giao diện người dùng đồ họa GUI (Graphics User Interface) Windows Form đến ứng dụng web, ứng dụng thiết bị di động và hệ thống nhúng, ứng dụng mạng và ứng dụng hướng dịch vụ trong môi trường phân tán Internet.
.NET framework là một nền tảng mới làm đơn giản việc phát triển ứng dụng trong môi trường phân tán của Internet với các đặc điểm:

- Cung cấp môi trường lập trình hướng đối tượng thuần túy và mạnh mẽ.
- Quản lý cài đặt phần mềm đảm bảo không tranh chấp phiên bản, thực thi an toàn mã nguồn
- Cung cấp mô hình bảo mật chung cho các ứng dụng
- .NET độc lập ngôn ngữ nghĩa là bạn có thể sử dụng nhiều ngôn ngữ lập trình được hỗ trợ bởi .NET để xây dựng và tích hợp các ứng dụng .NET. .NET framework cung cấp hệ thống kiểu chung CTS (Common Type System), định nghĩa các kiểu dữ liệu và các cấu trúc hỗ trợ bởi môi trường thực hiện và quản lý mã nguồn CLR (Common Language Runtime), và làm thế nào chúng có thể tương tác với nhau tuân theo đặc tả hạ tầng ngôn ngữ chung CLI (Common Language Infrastructure). Vì vậy, .NET hỗ trợ trao đổi các kiểu giữa các thư viện và ứng dụng sử dụng bất kỳ ngôn ngữ .NET. Có đến hơn hai mươi ngôn ngữ lập trình hiện nay được hỗ trợ

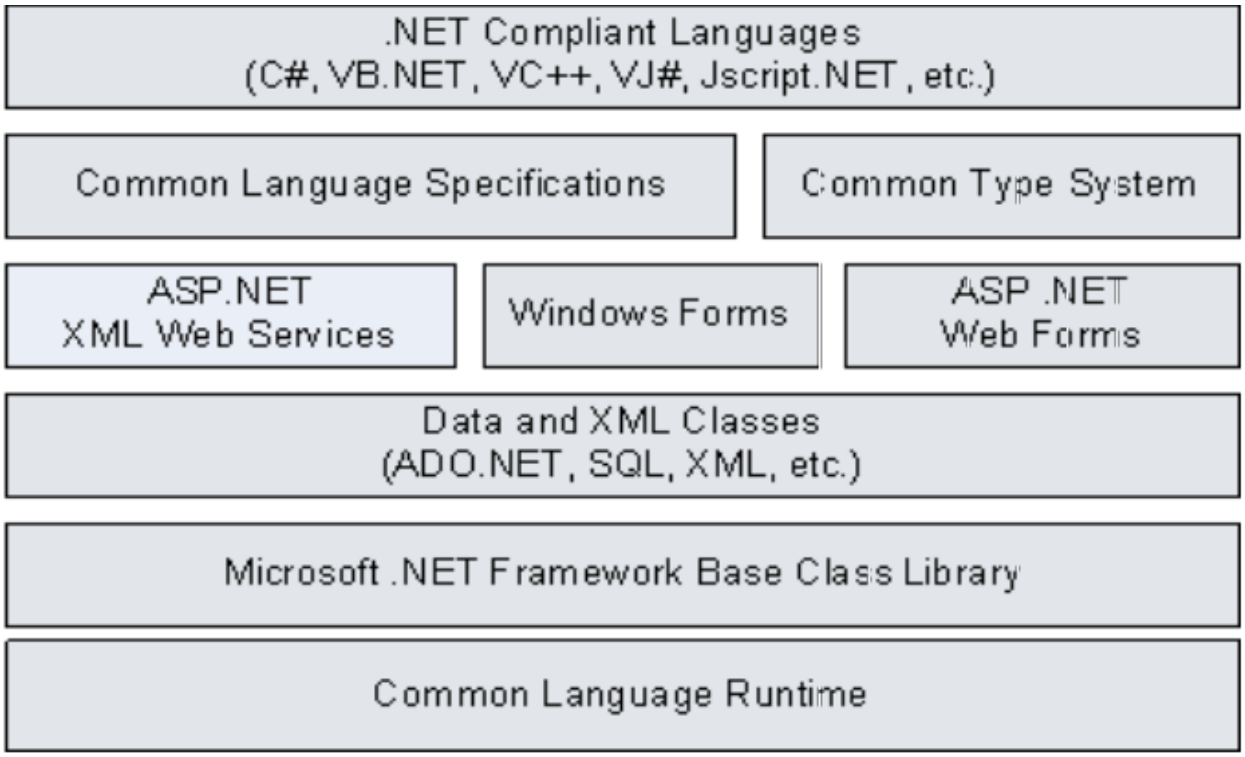
trên nền tảng .NET như là Fortran, Pascal, Cobol, Visual Basic, C#, C++, Java, JScript, Python, Eiffel, Perl, Small Talk, Scheme, APL, Mercury... Trong đó, các ngôn ngữ lập trình phổ biến nhất để phát triển các ứng dụng trên nền tảng .NET hiện nay là C# và VB.NET.

.NET framework có hai thành phần chính:

- CLR (Common Language Runtime): là nền tảng của .NET framework, môi trường thực hiện và quản lý mã nguồn mà được biên dịch thành một ngôn ngữ trung gian của Microsoft gọi là MSIL (Microsoft Intermediate Language) và lưu trữ trong tập tin gọi là assembly. Ngôn ngữ trung gian MSIL này là ngôn ngữ chung cho tất cả các ngôn ngữ .NET hiện có. Trong khi biên dịch như vậy, các ứng dụng cũng tạo ra

những thông tin cần thiết để giới thiệu ứng dụng, ta gọi những thông tin này là metadata. CLR sử dụng trình biên dịch JIT (Just-In-Time) để biên dịch mã IL một lần nữa thành ngôn ngữ gốc của máy tính (mã nhị phân) trước khi thực hiện. Các đặc điểm chính khác của CLR là quản lý phiên bản, quản lý bộ nhớ, tích hợp hệ thống độc lập ngôn ngữ và cung cấp hệ thống kiểu dữ liệu chung.

- .NET framework class library: thư viện lớp .NET, là tập hợp hướng đối tượng của các kiểu dữ liệu (lớp, giao tiếp, kiểu liệt kê…) tái sử dụng. Các kiểu dữ liệu đó được tổ chức lại thành từng nhóm riêng biệt như trong một thư viện để ta dễ dàng sử dụng. Ta gọi các nhóm như vậy là không gian tên (namespaces), và ta sẽ dùng những namespace này để gọi hay nhập (import) các kiểu dữ liệu cần thiết cho ứng dụng của mình.



Hình 1.1. Kiến trúc .NET framework

1.2. NGÔN NGỮ C#

1.2.1. Giới thiệu C#

Ngôn ngữ C# (C sharp) do đội ngũ kỹ sư của Microsoft thiết kế, trong đó người dẫn đầu là Anders Hejlsberg và Scott Wiltamuth. Anders Hejlsberg là tác giả của Turbo Pascal và là người lãnh đạo nhóm thiết kế Borland Delphi. Visual C# là một cài đặt của ngôn ngữ C# bởi Microsoft.

C# là một ngôn ngữ lập trình hướng đối tượng đầy năng lực, được dẫn xuất từ Java và C++, và thêm vào những đặc tính mới để làm cho ngôn ngữ này dễ sử dụng hơn. Nhiều trong số những đặc tính này khá giống với những đặc tính có trong ngôn ngữ Java. Nếu bạn đã sử dụng ngôn ngữ C++ hay Java, bạn sẽ thấy rằng cú pháp của C# rõ ràng tương thích.

C# đóng vai trò quan trọng trong kiến trúc của Microsoft .NET framework, và nhiều người đã so sánh với vai trò của C trong việc phát triển UNIX.

Ngôn ngữ C# được thiết kế cho việc xây dựng các ứng dụng dựa trên nền tảng .NET như ứng dụng Windows Form, ứng dụng web truy xuất cơ sở dữ liệu sử dụng công nghệ ADO.NET (ActiveX Data Objects for .NET) hay LINQ (Language Integrated Query), ứng dụng mạng, ứng dụng phân tán và hướng dịch vụ sử dụng các dịch vụ

COM+, .NET remoting, Web service, WCF (Windows Communication Foundation), ứng dụng xử lý song song đa tuyến đoạn (multi-threading), ứng dụng cho các thiết bị kỹ thuật số cá nhân PDA (Personal Digital Assistant) như pocket PC, SmartPhone, IPhone…

Lịch sử phát triển của C# và các đặc điểm chính của từng phiên bản mô tả trong bảng sau:

Bảng 1.1. Lịch sử phát triển của C#

Phiên bản	Ngày ra đời	Visual Studio	Đặc điểm mới
C# 1.0	01/2002	2002	
C# 2.0	11/2005	2005	Kiểu tổng quát (generic type), phương thức nặc danh (anonymous method), kiểu Nullable (nullable type)
C# 3.0	11/2007	2008	Định nghĩa biến kiểu không tường minh (implicitly typed local variable), kiểu nặc danh (anonymous type), phương thức mở rộng (extension method), khởi tạo đối tượng và danh sách (object and collection initializer), thuộc tính được tự động cài đặt (auto-implemented properties), biểu thức lambda (lambda expression), cây biểu thức (expression tree), biểu thức truy vấn (query expression) LINQ (Language-Integrated Query)
C# 4.0	04/2010	2010	Tham đối đặt tên (named argument), tham đối tùy chọn (optional argument), cải tiến hơn giao diện thành phần COM (more COM interface), kiểu dynamic và ràng buộc trễ (dynamic type and late binding)
C# 5.0	02/2012	2012	Lập trình bất đồng bộ (async programming), thông tin lời gọi phương thức (caller information)

1.2.2. Đặc trưng của ngôn ngữ C#

Mục tiêu của C# là cung cấp một ngôn ngữ hướng đối tượng, đơn giản, an toàn và hiện đại, với nhiều tính năng mạnh mẽ và mềm dẻo.

1.2.2.1. Đơn giản

Ngôn ngữ C# đơn giản, chỉ khoảng 80 từ khóa và hơn mười mấy kiểu dữ liệu được xây dựng sẵn. C# khá giống về cú pháp, biểu thức, toán tử và những chức năng khác được lấy trực tiếp từ ngôn ngữ

Java và C++, nhưng nó đã được cải tiến để làm cho ngôn ngữ đơn giản hơn.

Cũng như Java, C# loại bỏ một vài sự phức tạp của C++, bao gồm đa kế thừa, và lớp cơ sở ảo (virtual base class), chúng là những nguyên nhân gây ra những vấn đề cho người phát triển C++.
C# chỉ sử dụng dấu chấm để truy xuất các thành viên của lớp như Java, thay vì sử dụng dấu ::, . và → như C++. Đối với người mới tìm hiểu về công nghệ này thì các cải tiến này làm bớt nhầm lẫn và đơn giản hơn.
C# cũng đưa ra khái niệm thuộc tính (property) đơn giản thay cho phương thức truy cập và thiết lập giá trị biến thành viên. Thuộc tính cung cấp khả năng bảo vệ các biến dữ liệu bên trong một lớp, bằng việc đọc và ghi chúng thông qua thuộc tính.
Các phương thức nhập xuất dữ liệu là thành viên của lớp Console giúp người mới học dễ dàng nhớ các thao tác nhập xuất dữ liệu.
Các danh sách (Collection) phong phú thuộc không gian tên System.Collections và System.Collections.Generic, các kiểu liệt kê (enum) xây dựng sẵn giúp người dùng thuận tiện hơn trong xử lý các danh sách dữ liệu.

1.2.2.2. An toàn và hiện đại

C# chứa tất cả những đặc tính như là xử lý ngoại lệ, thu gom bộ nhớ tự động, những kiểu dữ liệu mở rộng, bảo mật mã nguồn, kiểu tổng quát (generic type), hỗ trợ cho cấu trúc, thành phần (component), lập trình hướng đối tượng là những đặc tính được mong đợi trong một ngôn ngữ lập trình hiện đại.
Trong C#, bộ thu gom bộ nhớ tự động và kiểu dữ liệu an toàn được tích hợp vào ngôn ngữ, sẽ loại bỏ những vấn đề rắc rối của C++.

1.2.2.3. Hướng đối tượng

Những đặc điểm chính của ngôn ngữ lập trình hướng đối tượng (object-oriented programming language) là đóng gói (encapsulation), kế thừa (inheritance), và đa hình (polymorphism) được hỗ trợ trong C#.

1.2.2.4. Mạnh mẽ và cũng mềm dẻo

C# được sử dụng cho nhiều ứng dụng từ client đến server trong môi trường phân tán internet như các ứng dụng Windows Form, ứng dụng web truy xuất cơ sở dữ liệu sử dụng công nghệ ADO.NET hay LINQ, ứng dụng mạng, ứng dụng phân tán và dịch vụ web sử dụng các dịch vụ COM+, .NET remoting, web service, WCF service, ứng dụng xử lý song song đa tuyến đoạn, ứng dụng cho các thiết bị kỹ thuật số cá nhân PDA như pocket PC, SmartPhone, IPhone…
C# là một trong những ngôn ngữ lập trình mới nhất và đang được sử dụng phổ biến. Nhiều sản phẩm của công ty Microsoft đã được chuyển đổi và viết lại bằng C#. Bằng cách sử dụng ngôn ngữ này, Microsoft đã xác nhận khả năng của C#.

1.2.2.5. Ngôn ngữ C# và những ngôn ngữ khác

Có nhiều ngôn ngữ lập trình phổ biến trước đây như Visual Basic, C++ và Java. Chúng ta hãy tìm hiểu sự khác nhau giữa ngôn ngữ C# và những ngôn ngữ này.
Nếu chúng ta đã học Java, chúng ta sẽ tìm thấy nhiều tương thích trong C#. Microsoft nói rằng C# mang đến sức mạnh của ngôn ngữ C++ với sự dễ dàng của ngôn ngữ Visual Basic.
C# loại bỏ một vài đặc tính phức tạp của C++ bao gồm đa kế thừa và lớp cơ sở ảo, chúng là những nguyên nhân gây ra những vấn đề cho người phát triển C++.
Trong ngôn ngữ C# mọi thứ liên quan đến khai báo lớp đều được tìm thấy trong phần khai báo của nó. Định nghĩa một lớp trong ngôn ngữ C# không yêu cầu phải chia ra tập tin header và tập tin nguồn giống như trong ngôn ngữ C++. Hơn thế nữa, ngôn ngữ C# hỗ trợ kiểu XML (eXtensible Markup Language), cho phép chèn các thẻ XML để phát sinh tự động các tài liệu cho lớp.

Như đã nói ở bên trên .NET runtime trong C# thực hiện việc thu gom bộ nhớ (garbage collection) tự động như Java. Do điều này nên việc sử dụng con trỏ trong C# ít quan trọng hơn trong C++. Nhưng con trỏ và phép toán con trỏ cũng có thể được sử dụng trong C#, khi đó những đoạn mã nguồn này sẽ được đánh dấu là không an toàn (unsafe code). Và bộ giải phóng bộ nhớ tự động của CLR sẽ không thực hiện việc giải phóng những đối tượng được tham chiếu bằng sử dụng con trỏ cho đến khi chúng được giải phóng.
Một điểm giống nhau giữa C# và Java là cả hai cùng biên dịch ra mã trung gian: C# biên dịch ra MSIL còn Java biên dịch ra bytecode. Sau đó chúng được thực hiện bằng cách thông dịch hoặc biên dịch trong từng máy ảo tương ứng. Tuy nhiên, trong ngôn ngữ C# nhiều hỗ trợ được đưa ra để biên dịch mã ngôn ngữ trung gian sang mã máy.
C# chứa nhiều kiểu dữ liệu cơ sở hơn Java và cũng cho phép nhiều mở rộng với kiểu dữ liệu giá trị. Ví dụ, ngôn ngữ C# hỗ trợ kiểu liệt kê (enumeration), kiểu này được giới hạn đến một tập hằng được định nghĩa trước, và kiểu dữ liệu cấu trúc là kiểu dữ liệu giá trị do người dùng định nghĩa.
C# hỗ trợ ngoại lệ và xử lý ngoại lệ giống Java và C++. Khái niệm không gian tên (namespace) của C# tương tự C++ và khái niệm gói (package) trong Java.
Ngoài việc truyền tham đối phụ thuộc vào kiểu dữ liệu tham đối như Java và C++, C# còn hỗ trợ từ khóa ref, out, cho phép truyền dữ liệu kiểu giá trị bằng tham chiếu. C# cũng hỗ trợ kiểu tổng quát (generic type) cho lớp và phương thức như C++ và Java.
Trong C#, nạp chồng toán tử được hỗ trợ tương tự C++.

Những thành viên của lớp được gọi duy nhất bằng toán tử "."
như Java, và khác với C++ có nhiều cách gọi trong các tình huống khác nhau.

C# cũng hỗ trợ giao tiếp (interface). Trong ngôn ngữ C#, một lớp chỉ có thể kế thừa từ duy nhất một lớp cha, tức là không cho đa kế thừa như trong ngôn ngữ C++, tuy nhiên một lớp có thể cài đặt nhiều giao tiếp như Java và C++.
Trong ngôn ngữ C#, cấu trúc cũng được hỗ trợ, nhưng khái niệm và ngữ nghĩa của nó thay đổi khác với C++. Trong C#, một cấu trúc được giới hạn, là kiểu dữ liệu nhỏ gọn, và khi tạo thể hiện yêu cầu ít hơn về hệ điều hành và bộ nhớ so với một lớp. Một cấu trúc thì không thể kế thừa từ một lớp hay được kế thừa nhưng một cấu trúc có thể hiện thực một giao tiếp.
Và nhiều đặc tính mới mạnh mẽ khác được bổ sung trong C# như các khái niệm lệnh lặp foreach, property, event, delegate, indexer…

1.2.3. Biên dịch và thực hiện ứng dụng Console C#.NET đơn giản

Để bắt đầu cho việc tìm hiểu ngôn ngữ C#, chương đầu tiên trình bày cách soạn thảo, biên dịch và thực thi một chương trình C# đơn giản nhất.

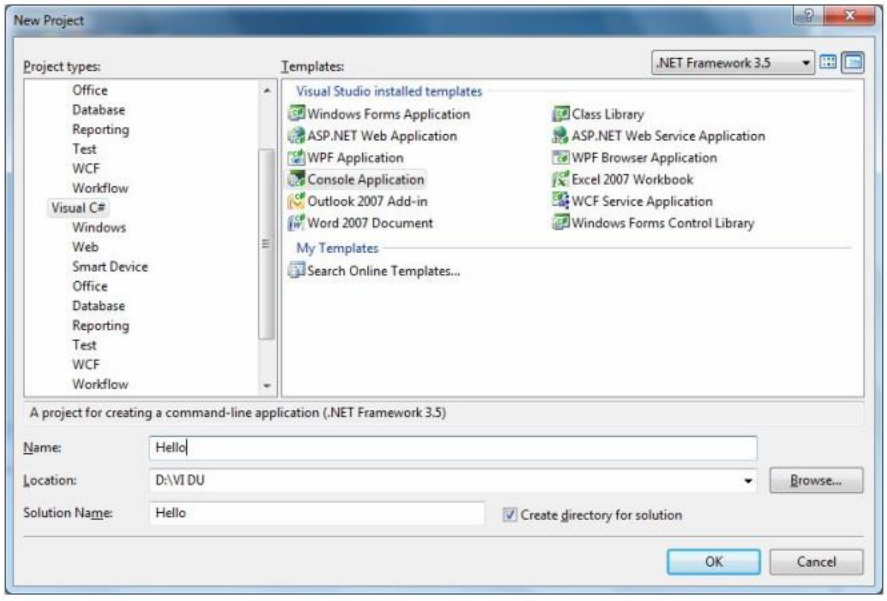
Đây là một định nghĩa lớp Hello gồm có một phương thức tên là Main(). Tất cả các chương trình C# phải chứa một lớp có phương thức là Main() là phương thức được gọi thực hiện đầu tiên mỗi khi thực hiện chương trình. Phương thức Console.WriteLine() sử dụng để xuất một chuỗi trên màn hình console và kết thúc dòng.
C# là một ngôn ngữ phân biệt hoa-thường. Bạn cần phải chú ý là các từ khóa trong C# đều được viết thường ví dụ public, class, static… trong khi các tên namespace, tên lớp, phương thức… sẽ được viết hoa đầu từ, ví dụ System, Console.WriteLine, Main…
Có hai cách để soạn thảo, biên dịch và thực thi chương trình:

1.2.3.1. Sử dụng trình biên dịch dòng lệnh C#

Sử dụng chương trình soạn thảo văn bản bất kỳ như Notepad soạn thảo mã nguồn chương trình, rồi lưu vào tập tin có phần mở rộng là *.cs, trong ví dụ này là Hello.cs.
Bước tiếp theo là biên dịch tập tin nguồn vừa tạo ra, sử dụng trình biên dịch dòng lệnh C# (C# command-line compiler) csc.exe :
Chọn Start/ Programs/ Microsoft Visual Studio/ Visual Studio Tools/ Visual Studio Command Prompt.
Chuyển đến thư mục chứa tập tin nguồn, rồi gõ lệnh sau:
csc.exe [/out: TậpTinThựcThi] TậpTinNguồn
Ví dụ: csc /out:Hello.exe Hello.cs

hay csc Hello.cs
Kết quả tập tin Hello.exe sẽ xuất hiện trong cùng thư mục chứa tập tin nguồn.
Có thể sử dụng tùy chọn /out, theo sau là tên của tập tin chương trình thực thi hay chính là kết quả biên dịch tập tin nguồn.
Các tham số tùy chọn có rất nhiều nếu muốn tìm hiểu chúng ta có thể dùng lệnh: csc.exe /? Lệnh này hiển thị toàn bộ các tùy chọn biên dịch và các hướng dẫn sử dụng.
Cuối cùng, thực hiện tập tin Hello.exe bằng cách gõ: Hello

1.2.3.2. Sử dụng môi trường phát triển tích hợp IDE (Integrated Development Environment) Visual Studio .NET



Hình 1.2. Hộp đối thoại New project

Để tạo dự án Hello.cs trong Microsoft Visual Studio, khởi động Visual Studio, chọn menu File/ New/ Project. Chức năng này sẽ mở cửa sổ New Project. Chọn mục Visual C# trong vùng Project types bên trái, ở vùng Templates bên phải, chọn Console Application. Lúc này chúng ta có thể nhập tên cho ứng dụng ở mục Name, lựa chọn thư mục lưu trữ các tập tin này ở mục Location, và nhập tên solution chứa dự án ở mục Solution Name. Visual Studio .NET sẽ tạo ra một solution Hello chứa project Hello. Một không gian tên (namespace) phát sinh dựa trên tên của project Hello để chứa project. Một lớp tên là Program.cs phát sinh, có thể tùy ý đổi tên của chúng trong cửa sổ Solution Explorer. Khi đổi tên tập tin chứa lớp là Hello.cs, tên lớp cũng thay đổi thành Hello.
Để biên dịch chương trình, chọn menu Build/ Build Solution.
Để chạy chương trình có hay không sử dụng chế độ debug, chọn

Debug/ Start Debugging hay Start Without Debugging.

Sau khi biên dịch và chạy chương trình, kết quả dòng chữ "Welcome to C#.NET" hiển thị ra màn hình.
Chúng ta cần tìm hiểu các khái niệm solution, project và assembly.
Dự án (project) sử dụng để quản lý, xây dựng, biên dịch và thực hiện hiệu quả các thành viên cần thiết để tạo nên ứng dụng như các tham chiếu (references), các kết nối cơ sở dữ liệu (data connections), các thư mục (folders) và các tập tin (files). Tập tin dự án sau khi biên dịch là một tập tin khả thi .exe (executable file) hay một thư viện liên kết động .dll (dynamic link library).
Solution có thể chứa một hay nhiều project. Visual Studio .NET lưu trữ định nghĩa solution trong hai tập tin: .sln và .suo. Tập tin .sln lưu trữ dữ liệu định nghĩa solution như các thành viên và cấu hình ở cấp solution. Tập tin .suo lưu trữ dữ liệu thiết lập tùy chọn IDE. Để một dự án trong solution trở thành dự án khởi động, kích phải trên tên dự án trong cửa sổ Solution Explorer, chọn Set as StartUp Project.
Một cửa sổ giao diện cho việc xem và quản lý các solution, project và các thành viên của project là Solution Explorer, được cung cấp bởi Visual Studio .NET bằng cách chọn View/ Solution Explorer.
Assembly là thành phần cơ bản nhất của bất kỳ ứng dụng .NET. Khi biên dịch một ứng dụng .NET, Visual Studio sẽ tạo ra một assembly được lưu trữ dạng tập tin khả thi .exe hay tập tin thư viện liên kết động .dll. Assembly chứa mã biên dịch MSIL và các thành phần khác như tập các kiểu và tài nguyên, với mục đích bảo mật, định danh kiểu, chia sẻ, phiên bản.

Chương 2
CƠ BẢN VỀ C#

Chương này trình bày cơ bản về C# và so sánh ngôn ngữ C# với Java và C++. Bạn đọc sẽ khám phá hệ thống kiểu dữ liệu, phân biệt giữa kiểu dữ liệu giá trị và tham chiếu, kiểu dữ liệu định nghĩa sẵn và kiểu dữ liệu do người dùng định nghĩa. Bên cạnh đó bạn đọc sẽ tìm hiểu cách khai báo biến, cách sử dụng các phép toán, câu lệnh điều kiện, lệnh lặp. Cuối cùng, bạn đọc sẽ tìm hiểu C# sử dụng ngoại lệ để xử lý lỗi dễ dàng như thế nào. Tất cả những kiến thức cơ bản này là nền tảng cho lập trình ứng dụng Windows Form.

Các thành phần cơ bản của ngôn ngữ C#
Kiểu dữ liệu
- Kiểu dữ liệu giá trị (Value data type)
  - ♣ Kiểu dữ liệu cơ sở (Primitive data type)
  - ♣ Kiểu cấu trúc (Struct)
  - ♣ Kiểu liệt kê (Enumeration)

♣ Kiểu Nullable
• Kiểu dữ liệu tham chiếu (Reference data type)
♣ Chuỗi (String)
♣ Kiểu con trỏ (Pointer)
♣ Mảng (Array)
♣ Kiểu tham chiếu phương thức (Delegate)
♣ Lớp (Class)

♣ Các lớp danh sách (Collection) xây dựng sẵn
♣ Biến (Variable) và hằng (constant và literal)
♣ Phép toán (Operator) và các quy tắc trên các phép toán
♣ Cấu trúc điều khiển (Control structures)
♣ Ngoại lệ (Exception) và xử lý ngoại lệ
♣ Nhập/ xuất dữ liệu từ bàn phím

2.1. CÁC THÀNH PHẦN CƠ BẢN CỦA NGÔN NGỮ C#

2.1.1. Bộ ký tự dùng trong C# (Character set)

Ngôn ngữ C# được xây dựng trên bộ ký tự gồm 26 chữ cái hoa A...Z và 26 chữ cái thường a...z, 10 chữ số 0...9, các ký hiệu toán học + - * / % = ()..., dấu nối _, các ký hiệu đặc biệt khác ; : {} [ ] ? \ & | # $...

C# sử dụng bộ ký tự chuẩn quốc tế Unicode. Khác với ký tự ASCII dài 8 bít, ký tự Unicode dài 16 bit hay 32 bit. Nó không chỉ bao gồm những ký tự trong bộ ký tự ASCII mà còn có vài triệu ký tự khác tương ứng với hầu hết các bảng chữ cái trên thế giới.

2.1.2. Từ khoá (Keyword)

Là những từ định nghĩa trước trong C#, có ý nghĩa xác định, phải dùng đúng cú pháp, đều viết bằng chữ thường, không dùng vào việc khác hay đặt tên mới trùng từ khoá.

Từ khoá gồm có từ khoá khai báo, điều khiển, kiểu dữ liệu, toán tử:

• Từ khoá khai báo: namespace, public, private, static, const, class, new…
• Từ khoá điều khiển: switch, case, break, if, return, for, while, continue, try, catch…
• Từ khoá toán tử: is
• Từ khoá kiểu dữ liệu: bool, byte, sbyte, short, ushort, float, double, null, void…

2.1.3. Định danh (Identifier)

Định danh là tên được đặt cho kiểu dữ liệu, phương thức, biến, hằng, đối tượng, không gian tên… Định danh là một dãy ký tự có phân biệt chữ hoa, thường. Tên bắt đầu bằng ký tự chữ cái hoặc dấu

gạch nối dưới _, các ký tự còn lại phải là ký tự chữ cái, chữ số, dấu gạch nối dưới. Các định danh không được đặt tên trùng với từ khóa của C#.

Khi đặt tên nên theo quy tắc đặt tên định nghĩa sẵn của C# cho dễ nhớ là tất cả các tên của lớp, phương thức, biến, giao tiếp, không gian tên… đều viết chữ hoa đầu từ, ngoại trừ tên hằng viết chữ hoa.

2.1.4. Lời chú thích (Comment)

Lời chú thích thêm vào chương trình với mục đích giải thích, giúp cho người lập trình dễ dàng bổ sung, sửa chữa, nâng cấp chương trình. Khi chạy chương trình, trình dịch sẽ bỏ qua chú thích.

Các lời chú thích có thể được thêm vào trong mã nguồn C# giống như trong Java, C++ và có thể đặt tại bất kỳ vị trí nào:

- Chú thích gồm một hay nhiều dòng: bắt đầu chú thích với dấu /* và kết thúc bởi */
- Chú thích một dòng: bắt đầu dòng chú thích bởi dấu //
- Ngoài hai kiểu chú thích trên giống trong Java, C++ thì C# còn hỗ trợ thêm kiểu thứ ba, kiểu này chứa các định dạng XML nhằm xuất ra tập tin XML khi biên dịch để tạo tài liệu chú thích cho mã nguồn. Các chú thích XML rất đơn giản, được tạo bằng cách đặt dấu /// trước các lớp, giao tiếp, phương thức hoặc trước phần khai báo thuộc tính… Có hơn 20 loại thẻ khác nhau có thể được sử dụng trong các chú thích XML, và được chia làm 2 loại chính: thẻ chính và thẻ phụ.

Sau khi tạo chú thích, mở cửa sổ Properties của ứng dụng, chọn trang Build, rồi kích chọn mục XML documentation file. Biên dịch ứng dụng, sẽ phát sinh tập tin tài liệu chú thích .XML.

2.2. KIỂU DỮ LIỆU, BIẾN, HẰNG

2.2.1. Kiểu dữ liệu (Data type)

C# là một ngôn ngữ định kiểu mạnh mẽ. Kiểu dữ liệu là tập hợp các giá trị mà một biến thuộc kiểu đó có thể nhận được. Mỗi biến phải khai báo thuộc một kiểu dữ liệu.

Các kiểu dữ liệu trong ngôn ngữ C# chia làm hai loại, kiểu dữ liệu xây dựng sẵn và kiểu dữ liệu do người dùng định nghĩa. C# cũng phân loại các kiểu dữ liệu thành hai loại: kiểu dữ liệu giá trị (value data type), và kiểu dữ liệu tham chiếu (reference data type). Việc phân chia này do sự khác nhau khi lưu kiểu dữ liệu giá trị và kiểu dữ liệu tham chiếu trong bộ nhớ. Biến kiểu dữ liệu giá trị lưu giữ giá trị của nó trong bộ nhớ tĩnh (stack). Trong khi đó biến kiểu tham chiếu, không chứa đối tượng thuộc kiểu này, mà lưu một địa chỉ hay tham chiếu đến đối tượng trong bộ nhớ stack, còn đối tượng thật sự lưu trong bộ nhớ động (heap). Nếu chúng ta có một đối tượng có kích thước rất lớn thì việc lưu giữ chúng trên bộ nhớ heap rất có ích. Tất cả các kiểu dữ liệu cơ sở

xây dựng sẵn là kiểu dữ liệu giá trị. Kiểu chuỗi và các lớp định nghĩa sẵn là kiểu dữ liệu tham chiếu. Và tất cả các kiểu do người dùng định nghĩa ngoại trừ kiểu cấu trúc, liệt kê đều là kiểu dữ liệu tham chiếu. Ngoài ra C# cũng hỗ trợ kiểu con trỏ như C++, nhưng ít khi được sử dụng, khi làm việc với những đoạn mã lệnh có sử dụng kiểu con trỏ, sẽ được đánh dấu không an toàn (unsafe).

2.2.1.1. Kiểu dữ liệu giá trị

Bao gồm các kiểu dữ liệu cơ sở định nghĩa sẵn, kiểu liệt kê và kiểu cấu trúc. Kiểu dữ liệu cấu trúc người dùng định nghĩa sẽ được tìm hiểu trong chương kế tiếp cùng với kiểu lớp và kiểu tham chiếu phương thức (delegate).

Kiểu cơ sở

Ngôn ngữ C# đưa ra các kiểu dữ liệu cơ sở rất hữu dụng, mỗi kiểu dữ liệu được ánh xạ đến một kiểu dữ liệu hỗ trợ bởi .NET. Việc ánh xạ các kiểu dữ liệu cơ sở của C# đến các kiểu dữ liệu của .NET sẽ đảm bảo các đối tượng được tạo ra trong C# có thể được sử dụng đồng thời với các đối tượng được tạo bởi bất kỳ ngôn ngữ khác được biên dịch bởi .NET.

Các kiểu dữ liệu cơ sở là kiểu dữ liệu struct xây dựng sẵn chứa nhiều thuộc tính và phương thức cần thiết, trong đó phương thức thường dùng là: Parse(string s): chuyển chuỗi s thành kiểu struct tương ứng

Kiểu liệt kê

Kiểu liệt kê đơn giản là tập hợp các tên hằng có giá trị không thay đổi, giúp bạn tổ chức dữ liệu khoa học hơn, mã trong sáng dễ hiểu hơn. C# cung cấp rất nhiều kiểu liệt kê định nghĩa sẵn, sẽ được đề cập đến trong các chương kế tiếp.

Kiểu Nullable (Nullable type)

C# 2.0 cung cấp kiểu struct System.Nullable<T> thuộc không gian tên System cho phép kiểu giá trị có giá trị null. Tất cả kiểu dữ liệu có giá trị null tương ứng. Một kiểu Nullable có thể biểu diễn một dãy giá trị của kiểu tương ứng và giá trị null. Chẳng hạn như biến kiểu Nullable<Int32> có thể gán giá trị từ -2147483648 đến 2147483647 hay giá trị null.

Kiểu Nullable khai báo cho các biến kiểu giá trị mà có thể gán giá trị null. Bạn không thể tạo một kiểu Nullable dựa trên kiểu tham chiếu. Kiểu tham chiếu đã hỗ trợ giá trị null là tham chiếu rỗng.

Cú pháp T? sử dụng để khai báo kiểu Nullable<T>, trong đó T là kiểu giá trị. Phép toán ?? để gán giá trị mặc định của biến kiểu Nullable có giá trị là null, cho biến không có kiểu Nullable.

2.2.1.2. Kiểu dữ liệu tham chiếu

Kiểu chuỗi

Kiểu dữ liệu chuỗi (string hay String) là kiểu tham chiếu xây

dựng sẵn, lưu giữ một dãy ký tự. Chuỗi trong C# là không thay đổi, mỗi khi chuỗi thay đổi, một chuỗi mới được tạo ra để chứa chuỗi kết quả.

Tạo một đối tượng chuỗi: Nhiều string được tạo từ các hằng chuỗi. Khi trình dịch bắt gặp một chuỗi ký tự bao giữa cặp nháy kép, nó tạo ra một đối tượng chuỗi có giá trị là chuỗi này. Bạn có thể dùng hằng chuỗi ở bất kỳ đâu bạn dùng đối tượng string

Kiểu mảng

Mảng là một cấu trúc lưu giữ các thành phần cùng kiểu. Mỗi thành phần của mảng được truy xuất bởi chỉ số của nó trong mảng.

Mảng 1 chiều:

- Khai báo một biến dùng để tham chiếu đến mảng, nhưng không có mảng nào thật sự tồn tại

KiểuPhầnTử[] TênMảng;

Kiểu phần tử là kiểu dữ liệu của các thành phần của mảng, có thể là bất kỳ kiểu giá trị hay tham chiếu.

- Truy xuất thành phần của mảng: TênMảng[ChỉMục]
- Lấy kích thước mảng: TênMảng.**Length**
- Khai báo, cấp phát bộ nhớ và khởi tạo giá trị đầu của mảng:

Mảng có thể khởi tạo khi khai báo sử dụng toán tử new hay không. Mảng khởi tạo là danh sách các biểu thức cách nhau dấu phẩy, đặt trong ngoặc móc. Chiều dài mảng là số thành phần giữa hai dấu { và }

Kiểu con trỏ

C# cũng hỗ trợ kiểu con trỏ như C++, nhưng ít khi được sử dụng, khi đoạn mã lệnh sử dụng kiểu con trỏ, sẽ được đánh dấu không an toàn (unsafe). Khai báo biến kiểu con trỏ như sau:

Kiểu* TênBiến;

void* TênBiến; //cho phép nhưng không nên dùng

2.3. BIỂU THỨC VÀ TOÁN TỬ

2.3.1. Biểu thức (Expression)

Biểu thức là một công thức tính toán một giá trị theo quy tắc toán học, cấu tạo từ các toán hạng (operand): hằng, phương thức, biến và nối kết với nhau bởi các toán tử (operator). Các toán hạng trong

mọi biểu thức phải tương thích với nhau về kiểu. Một biểu thức có thể là biểu thức số học, logic, ký tự, chuỗi khi kết quả của biểu thức có kiểu tương ứng.

Ví dụ: Biểu thức đơn giản nhất là biến hay hằng

22 a

"Hello" true

2.3.2. Toán tử (Operator)

Toán tử là phép toán dùng để kết hợp các toán hạng, dùng trong

các biểu thức.

2.3.2.1. Các toán tử số học (Arithmetic operator)

Phép toán số học sử dụng trên các toán hạng có kiểu số và trả về kết quả kiểu số:

Toán tử	**Ý nghĩa**	**Cách dùng**
+	Cộng hai toán hạng	op1 + op2
-	Trừ	op1 – op2
*	Nhân	op1 * op2
/	Chia	op1 / op2
%	Chia lấy số dư	op1 % op2

(op1, op2 là các toán hạng kiểu số)

2.3.2.2. Các toán tử tăng, giảm (Increment, decrement operator)

Phép toán tăng giảm sử dụng cho các biến kiểu số nguyên và kiểu ký tự:

- op ++ hay ++ op: Tăng biến số nguyên lên một đơn vị, hay cho ký tự Unicode kế tiếp
- op -- hay -- op: Giảm biến số nguyên xuống một đơn vị, hay cho ký tự Unicode kế trước.

2.3.2.3. Toán tử quan hệ hay so sánh (Comparison operator)

Phép toán so sánh áp dụng trên các kiểu dữ liệu cơ bản và luôn cho kết quả có kiểu logic.

Toán tử	**Ý nghĩa**	**Cách dùng**
==	Bằng	op1 = = op2
!=	Không bằng	op1 != op2
<	Nhỏ hơn	op1 < op2
>	Lớn hơn	op1 > op2
<=	Nhỏ hơn hoặc bằng	op1 <= op2
>=	Lớn hơn hoặc bằng	op1 >= op2

2.3.2.4. Toán tử logic (Logical operator)

Phép toán logic chỉ áp dụng cho các toán hạng kiểu logic, cho kết quả kiểu logic

Toán tử	**Ý nghĩa**	**Cách dùng**
& hay &&	Và (And)	op1 && op2
| hay ||	Hoặc (Or)	op1 || op2
^	Hoặc loại trừ (Xor)	op1 ^ op2
!	Phủ định (Not)	! op

- Phép & hay && cho kết quả true chỉ khi 2 toán hạng op1, op2 có giá trị true
- Phép | hay || cho kết quả là false chỉ khi 2 toán hạng op1, op2 đều có giá trị false
- Phép ^ cho kết quả là true chỉ khi 2 toán hạng op1, op2 khác

giá trị nhau

- Phép ! cho kết quả là true nếu op có trị false, và ngược lại

2.3.3. Các quy tắc thực hiện phép toán và chuyển kiểu

2.3.3.1. Quy tắc thứ tự ưu tiên (Precedence order rule)

Độ ưu tiên của các phép toán trong biểu thức thực hiện theo thứ tự từ trên xuống như trong bảng sau. Có thể chủ động quy định thứ tự thực hiện phép toán trong biểu thức bằng những cặp ngoặc tròn ( ), trong ngoặc sẽ thực hiện trước, và ngoặc trong thực hiện trước, ngoặc ngoài thực hiện sau. Các phép toán trên cùng hàng sẽ có cùng độ ưu tiên.

2.3.3.2. Quy tắc kết hợp (Associativity rule)

Quy tắc kết hợp thực hiện trên các phép toán có cùng độ ưu tiên theo thứ tự từ trái qua phải hay từ phải qua trái như trong bảng 2.4

Ví dụ:

int x = a + b - c; s += a += b += c;	//Thứ tự thực hiện phép toán + - từ trái qua phải //Thứ tự thực hiện phép gán từ phải sang trái

2.3.3.3. Quy tắc nâng cấp toán hạng (Operand promotion rule)

C# tự động nâng cấp kiểu cơ sở nào đó thành kiểu cơ sở khác lớn hơn, khi có nhu cầu. Chẳng hạn, nâng cấp kiểu char thành int, short hoặc int thành long, nâng cấp kiểu float thành double, hay kiểu nguyên thành kiểu thực. Các trường hợp chuyển đổi kiểu như vậy không làm mất mát thông tin.

2.3.3.4. Quy tắc chuyển kiểu (Casting rule)

Nếu bạn muốn chuyển đổi giữa các kiểu dữ liệu, sử dụng cú pháp như sau:

(KiểuMới) Value

2.4. LỆNH (STATEMENT)

Lệnh có thể là lệnh đơn, khối lệnh hay cấu trúc điều khiển.

2.4.1. Lệnh đơn (Simple statement)

Lệnh đơn là thành phần cơ bản nhất trong chương trình C#, diễn đạt một thao tác riêng lẻ nào đó phải làm, được kết thúc bằng dấu chấm phẩy.

2.4.2. Khối lệnh (Compound statement hay block)

Nhiều câu lệnh đơn có thể nhóm lại thành một câu lệnh phức hợp gọi là khối lệnh. Khối lệnh được mở đầu bằng dấu { và kết thúc bằng dấu }, có thể đặt trong một khối lệnh khác.

2.4.3. Cấu trúc rẽ nhánh có điều kiện (Conditional structure)

2.4.3.1. Cấu trúc if

if là câu lệnh lựa chọn cho phép chương trình rẻ nhánh thực hiện lệnh theo hai hướng khác nhau căn cứ trên giá trị true, false của biểu thức điều kiện kiểu logic.

Cú pháp:

Dạng 1:	**if** (BiểuThứcĐiềuKiện)

	Lệnh; **if** (BiểuThứcĐiềuKiện)
Dạng 2:	

Lệnh1;
else Lệnh2;

- Biểu thức điều kiện (Conditional expression) có kiểu logic bool.
- Lệnh (Statement) có thể là lệnh đơn, khối lệnh, hay cấu trúc điều khiển

Ý nghĩa: Nếu biểu thức điều kiện thoả mãn (có giá trị true) thì lệnh 1 thực hiện, ngược lại nếu biểu thức điều kiện không thoả mãn (có giá trị false) thì không làm gì cả (với dạng 1) hay thực hiện lệnh 2 (với dạng 2)

2.4.3.2. Cấu trúc switch

switch là câu lệnh lựa chọn, cho phép rẻ nhánh thực hiện lệnh theo nhiều hướng khác nhau căn cứ trên giá trị của một biểu thức.

Cú pháp:

switch (BiểuThứcĐiềuKiện)
{
case Biểu thức 1:
Lệnh 1;
break;
…
case Biểu thức n:
Lệnh n;
break;
default:
Lệnh n + 1;
break;
}

2.4.4. Cấu trúc lặp (Repeat structure hay loop)

2.4.4.1. Cấu trúc for

Ngoài for, while và do… while, C# còn bổ sung thêm cấu trúc lặp foreach.

Cú pháp:

for (BiểuThứcKhởiTạo; BiểuThứcĐiềuKiện; BiểuThứcTăng)
Lệnh;

- Lệnh (Statement) có thể là lệnh đơn, khối lệnh, hay cấu trúc điều khiển.
- Biểu thức khởi tạo (Initialization expressions): là các biểu thức khởi tạo giá trị cho các biến điều khiển vòng lặp, cách nhau dấu phẩy.
- Biểu thức điều kiện (Boolean expression): có kiểu logic bool,

thường là biểu thức so sánh giá trị của biến điều khiển với giá trị kết thúc vòng lặp.

- Biểu thức tăng (Increment expressions): là các biểu thức tăng giảm giá trị biến điều khiển vòng lặp, cách nhau dấu phẩy.

2.4.4.2. Cấu trúc while

Cú pháp:

while (BiểuThứcĐiềuKiện)
Lệnh;

- Lệnh (Statement): có thể là lệnh đơn, khối lệnh, hay cấu trúc điều khiển
- Biểu thức điều kiện (Conditional expression): có kiểu logic

Ý nghĩa: Kiểm tra chừng nào biểu thức điều kiện còn thoả mãn thì còn thực hiện lệnh.

2.4.4.3. Cấu trúc do… while

Cú pháp:

do
Lệnh;
while (BiểuThứcĐiềuKiện);

- Lệnh (Statement): Có thể là lệnh đơn, khối lệnh, hay cấu trúc điều khiển
- Biểu thức điều kiện (Conditional expression): Có kiểu logic

Ý nghĩa: Đầu tiên sẽ thực hiện lệnh, sau đó kiểm tra biểu thức điều kiện, nếu biểu thức đúng thì vòng lặp tiếp tục, nếu biểu thức sai thì thoát vòng lặp

2.4.4.4. Cấu trúc foreach

Cấu trúc foreach cho phép tạo vòng lặp duyệt qua tất cả phần tử của một danh sách (collection) hay một mảng. Đây là một câu lệnh lặp mới không có trong ngôn ngữ C/C++. Câu lệnh foreach có cú pháp chung như sau: **foreach** (KiểuPhầnTử TênPhầnTử **in** TênTậpHợp)
Lệnh;

- Lệnh (Statement): Có thể là lệnh đơn, khối lệnh, hay cấu trúc điều khiển
- Kiểu phần tử (Item type): Là kiểu phần tử trong danh sách
- Tên phần tử (Item name): Là tên biến dùng để truy cập phần tử
- Tên danh sách (Collection name): Là tên biến danh sách

2.5. NGOẠI LỆ (EXCEPTION) VÀ XỬ LÝ NGOẠI LỆ

Ngoại lệ trong C# là các đối tượng có kiểu lớp định nghĩa sẵn, biểu diễn trạng thái lỗi phát sinh trong trường hợp nào đó khi gọi phương thức hay trong trường hợp chia cho 0, cảnh báo bộ nhớ thấp… Một số phương thức định nghĩa sẵn trong trường hợp nào đó thì ngoại lệ sẽ phát sinh. Khi một ngoại lệ phát sinh, phải bắt ngoại lệ bởi lệnh try… catch, nếu không chương trình sẽ kết thúc thực

hiện. Lệnh xử lý ngoại lệ thực hiện thông qua các từ khoá: try, catch, finally và throw

2.5.3. Các lớp ngoại lệ

Exception là lớp cha của tất cả các ngoại lệ, cung cấp một số các phương thức và thuộc tính:

- Thuộc tính Message trả về thông tin phát sinh ngoại lệ.
- Thuộc tính HelpLink cung cấp một liên kết để trợ giúp cho các tập tin liên quan đến các ngoại lệ.
- Thuộc tính StackTrace cung cấp thông tin về câu lệnh lỗi.

Có hai kiểu ngoại lệ:

- Ngoại lệ phát sinh bởi ứng dụng dẫn xuất từ lớp ApplicationException
- Ngoại lệ phát sinh bởi hệ thống dẫn xuất từ lớp SystemException.

Hai lớp này kế thừa trực tiếp từ lớp Exception. Lớp ngoại lệ bạn định nghĩa nên dẫn xuất từ lớp ApplicationException.

2.6. LỚP SYSTEM.CONSOLE

Lớp Console bao gồm các phương thức nhập, xuất và định dạng kết xuất, các luồng xử lý lỗi cho các ứng dụng dựa trên console. Lớp Console trong phiên bản .NET 2.0 đã được cải tiến thêm các đặc tính mới.

2.6.1. Định dạng kết xuất

Lớp Console cung cấp một số đặc tính định dạng kết xuất sau:

- BackgroundColor, ForegroundColor: hai thuộc tính này cho phép bạn thiết lập màu nền và màu chữ cho kết xuất ra màn hình.
- BufferHeight, BufferWidth: hai thuộc tính này điều chỉnh chiều cao và chiều rộng cho buffer của console.
- Clear(): phương thức xóa buffer và vùng hiển thị của console.
- WindowHeight, WindowWidth, WindowTop, WindowLeft: các thuộc tính này điều khiển các chiều cao, rộng, lề trên, lề trái của console so với buffer.

2.6.2. Nhập và xuất với lớp Console

Lớp Console định nghĩa một tập các phương thức tĩnh để thực hiện việc nhập dữ liệu từ bàn phím và xuất ra màn hình.

- Phương thức Write() xuất một giá trị hay đối tượng ra màn hình. Nếu xuất đối tượng, phương thức ToSring() của đối tượng sẽ chuyển đối tượng thành chuỗi và xuất ra màn hình.
- Phương thức WriteLine() xuất một giá trị hay đối tượng ra màn hình, sau đó xuống dòng.
- Phương thức string ReadLine() trả về chuỗi nhập từ bàn

phím, kết thúc bởi phím xuống dòng.
• Phương thức int Read() trả về ký tự nhập từ bàn phím.
Chương 3*: Lập trình hướng đối tượng với C#*
• Phương thức khởi tạo danh sách nhân viên vừa biên chế vừa hợp đồng
• Phương thức thêm một nhân viên vào cuối danh sách
• Phương thức in danh sách nhân viên
• Phương thức sắp xếp danh sách nhân viên theo lương giảm dần
• Phương thức đếm số nhân viên biên chế
• Phương thức tìm nhân viên có mã nhân viên chỉ rõ
• Phương thức tìm nhân viên có lương nhỏ nhất đầu tiên trong mảng
• Phương thức tính tổng quỹ lương phải trả
• Phương thức xóa nhân viên có mã nhân viên chỉ rõ

TÀI LIỆU THAM KHẢO

[1] **Nguyễn Ngọc Bình Phương**, **Thái Thanh Phong**, *Các giải pháp lập trình C#*, NXB Giao thông vận tải, 2006
[2] **Erik brown**, *Windows Forms Programming with C#*, Manning Publications Co., 2002
[3] **Eric Gunnerson**, *A Programmer's Introduction to C#,* Apress publisher, 2000
[4] **Jack Xu, Ph.D**, *Practical C# Charts and Graphics*, UniCAD Publishing, 2007
[5] **James W. Cooper**, *C# Design Patterns,* Addison-Wesley publisher, 2006
[6] **Joseph Albahari**, *C# 4.0 in a Nutshell,* O'Reilly Media Inc., 2006
[7] **Matthew MacDonald**, *Pro .NET 2.0 Windows Forms and Custom Controls in C#,* Apress publisher, 2006
[8] Một số tài liệu trên Internet

LỜI CẢM ƠN

Nhóm em xin chân thành gửi lời cám ơn trường Cao Đẳng Công Nghệ Bách Khoa Hà Nội đã tạo điều kiện cho chúng em có cơ hội thực hành, tiếp xúc để chúng em có thể tránh được những vướng mắc và bỡ ngỡ trong môi trường học tập-thực hành sắp tới. Nhóm em xin chân thành cảm ơn Thầy Trần Thanh đã tận tình hướng dẫn , giải đáp thắc mắc và chỉ bảo nhóm em trong suốt thời gian nhóm em hoàn thành bài tập lớn . Mặc dù đã cố gắng hoàn thành đề tài tốt nhất nhưng do thời gian và kiến thức còn có hạn nên em sẽ không thể tránh khỏi những thiếu sót nhất định, rất mong nhận được sự cảm thông, chia sẻ và tận tình đóng góp chỉ bảo của quý thầy cô cũng như các bạn.