I. C# CƠ BẢN

Thông tin về Trung Tâm Lập Trình TP

Thông tin về khóa học

Đối tượng

Nền tảng cần có

C# là gì? Vì sao học c#?

1. C# là gì?

C# là một ngôn ngữ lập trình đơn giản, hiện đại, mục đích tổng quát, hướng đối tượng được phát triển bởi Microsoft và được phê chuẩn bởi European Computer Manufacturers Association (ECMA) và International Standards Organization (ISO).

C# được phát triển bởi Anders Hejlsberg và team của ông trong khi phát triển .Net Framework.

C# được thiết kế cho Common Language Infrastructure (CLI), mà gồm Executable Code và Runtime Environment, cho phép chúng ta sử dụng các ngôn ngữ high-level đa dạng trên các nền tảng và cấu trúc máy tính khác nhau.

1. Dưới đây là các lý do làm C# là ngôn ngữ lập trình chuyên nghiệp được sử dụng rộng rãi:

* C# là ngôn ngữ lập trình hiện đại, mục đích tổng quát.
* C# là ngôn ngữ hướng đối tượng tốt nhất(OOP).
* Dễ dàng để học.
* Là một ngôn ngữ được cấu trúc rõ ràng, có nhiều tài liệu hướng dẫn.
* Cung cấp môi trường để tạo ra các chương trình có hiệu quả.
* Có thể được biên dịch trên nhiều nền tảng máy tính khác nhau(Cross platforms).
* Là một phần của .Net Framework.

Các đặc điểm lập trình mạnh mẽ của C#

Cấu trúc C# khá gần với các ngôn ngữ high-level truyền thống, C và C++, và là một ngôn ngữ lập trình hướng đối tượng. Nó có sự giống nhau mạnh mẽ với Java, nó có nhiều đặc điểm lập trình mạnh mẽ mà làm cho nó trở nên ưa thích với các lập trình viên trên toàn thế giới.

Dưới đây là các đặc điểm quan trọng của C#:

* Điều kiện Boolean
* Tự động dọn rác bởi Garbage-Collector (GC)
* Thư viện chuẩn (Standard Library)
* Assembly
* Property và sự kiện (Event)
* Delegate Quản lý sự kiện
* Dễ dàng để sử dụng Generic
* Indexer
* Biên dịch có điều kiện (Conditional Compilation)
* Đa luồng dễ dàng (Multithreading)
* LINQ và Lambda Expression
* Tích hợp với Windows

Cài đặt môi trường (Enviroment)

**Giới thiệu .Net Framework**

.Net Framework là một nền tảng mang tính cách mạng giúp bạn viết các kiểu ứng dụng sau:

* Windows application
* Web application
* Dịch vụ Web

Các ứng dụng .Net Framework là các ứng dụng đa nền tảng. Nền tảng đã được thiết kế theo cách như vậy có thể được sử dụng từ bất kỳ ngôn ngữ lập trình nào: C#, C++, Visual Basic, Jscript, COBOL… Tất cả các ngôn ngữ này có thể truy cập tới nền tảng này cũng như giao tiếp với nhau.

.Net Framework gồm một thư viện code khổng lồ được sử dụng bởi các ngôn ngữ client như C#. Dưới đây là một số thành phần của .Net Framework.

* Common Language Runtime (CLR)
* .Net Framework Class Library
* Common Language Specification
* Common Type System
* Metadata và Assembly
* Windows Form
* ASP.Net và ASP.Net AJAX
* ADO.Net
* Windows Workflow Foundation (WF)
* Windows Presentation Foundation
* Windows Communication Foundation (WCF)
* LINQ

**Integrated Development Environment (IDE) cho C#**

Microsoft cung cấp các công cụ phát triển sau cho lập trình C#:

* Visual Studio 2015(Bảng đề nghị)
* Visual Studio 2013
* Visual Studio 2012

Cú pháp cơ bản (Basic Syntax)

C# là một ngôn ngữ lập trình hướng đối tượng. Trong phương pháp lập trình hướng đối tượng, một chương trình gồm các đối tượng đa dạng mà tương tác với nhau các cách thức của action. Các action mà một đối tượng có thể nhận được gọi là các phương thức. Các đối tượng cùng loại được xem như là có cùng kiểu hoặc, được xem như là trong cùng lớp.

Ví dụ, xét đối tượng Rectangle. Nó có các thuộc tính như length và width. Phụ thuộc vào thiết kế trên. Nó có thể cần những cách để cấp nhận các giá trị của thuộc tính này, tính toán diện tích và hiển thị chi tiết.

Chúng ta cùng theo dõi ví dụ sau triển khai một lớp Rectangle và cú pháp C# cơ bản của chương trình này. Để minh họa rõ ràng tính hướng đối tượng của C#, mình sẽ tạo hai class trong hai file riêng rẽ nhau.

Ví dụ lớp Rectangle: chứa các thuộc tính, phương thức chính

class Rectangle

{

// cac bien thanh vien

double \_length;

double \_width;

//phuong thuc

public void Acceptdetails()

{

\_length = 4.5;

\_width = 3.5;

}

//phuong thuc

public double GetArea()

{

return \_length \* \_width;

}

//phuong thuc

public void Display()

{

Console.WriteLine("Chieu dai: {0}", \_length);

Console.WriteLine("Chieu rong: {0}", \_width);

Console.WriteLine("Dien tich: {0}", GetArea());

}

}

Lớp ExecuteRectangle: là lớp chứa phương thức main() để tiến hành thao tác trên đối tượng Rectangle của lớp Rectangle. Đến đây, có lẽ bạn chưa hiểu gì, nhưng không sao vì đây chỉ là chương giúp bạn làm quen với C# thôi.

using System;

namespace TPCenter

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Chuong trinh minh hoa tinh huong doi tuong trong C#");

Console.WriteLine("--------------------------------------------------\n");

//tao doi tuong Rectangle

Rectangle r = new Rectangle();

//goi cac phuong thuc cua doi tuong nay

r.Acceptdetails();

r.Display();

Console.ReadLine();

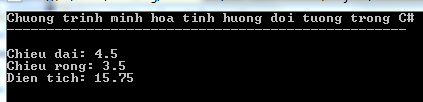
Console.ReadKey();

}

}

}

Nhấn phím **F5** để biên dịch và chạy chương trình C# trên sẽ cho kết quả sau:



**Từ khóa using trong C#**

Lệnh đầu tiên trong bất kỳ chương trình C# nào là:

using System;

Từ khóa using được sử dụng để bao namespace trong chương trình. Một chương trình C# có thể bao nhiều lệnh using.

**Từ khóa class trong C#**

Từ khóa class được sử dụng để khai báo một lớp trong C#.

**Comments trong C#**

Comment được sử dụng để khởi tạo code. Compiler bỏ qua các comment. Các comment đa dòng trong các chương trình C# bắt đầu với /\* và kết thúc với \*/ như sau:

/\* dong nay minh hoa comment nhieu dong trong C#.

Cu phap co ban C#

Ngon ngu lap trinh C# \*/

Comment đơn dòng được chỉ dẫn bởi ký hiệu '//'. Ví dụ:

// vi du comment don dong trong C#

**Biến thành viên**

Các biến là các thuộc tính hoặc thành viên dữ liệu của một lớp, được sử dụng để lưu giữ dữ liệu. Trong chương trình trước đó, lớp Rectangle có hai biến thành viên là length và width.

**Hàm thành viên**

Hàm là tập hợp các lệnh mà thực hiện một tác vụ cụ thể. Các hàm thành viên của một lớp được khai báo bên trong lớp đó. Lớp Rectangle chứa 3 hàm thành viên là: AcceptDetails, GetArea và Display.

**Thuyết minh một Class**

Trong chương trình trên, lớp ExecuteRectangle chứa phương thức Main() và khởi tạo lớp Rectangle.

**Định danh (Identifier)**

Một định danh là một tên được sử dụng để nhận diện một lớp, biến, hàm hoặc bất kỳ mục tự định nghĩa (user-defined).

Một tên phải bắt đầu với một chữ cái mà có thể được theo sau bởi một dãy các chữ cái, chữ số (0-9) hoặc dấu gạch dưới (\_). Ký tự đầu tiên của một định danh không thể là một chữ số.

Nó phải không chứa bất kỳ khoảng trống hoặc ký tự như ? - + ! @ # % ^ & \* ( ) [ ] { } . ; : " ' / và \. Tuy nhiên, dấu gạch dưới có thể được sử dụng.

Nó không nên là một từ khóa trong C#.

**Từ khóa trong C#**

Từ khóa là các từ dành riêng (Reserved Keyword) được định nghĩa trước cho C# compiler. Những từ khóa này không thể được sử dụng làm định danh. Tuy nhiên, nếu bạn muốn sử dụng các từ khóa này để làm định danh, bạn có thể đặt ký tự @ ở trước chúng.

Trong C#, một số định danh có ý nghĩa đặc biệt trong ngữ cảnh của code, ví dụ như get và set được gọi là các contextual keyword (từ khóa thuộc ngữ cảnh).

|  |
| --- |
|  |
|  |
| **Reserved Keyword** | | | | | | |
| abstract | as | base | bool | break | byte | case |
| catch | char | checked | class | const | continue | decimal |
| default | delegate | do | double | else | enum | event |
| explicit | extern | false | finally | fixed | float | for |
| foreach | goto | if | implicit | in | in (generic modifier) | int |
| interface | internal | is | lock | long | namespace | new |
| null | object | operator | out | out (generic modifier) | override | params |
| private | protected | public | readonly | ref | return | sbyte |
| sealed | short | sizeof | stackalloc | static | string | struct |
| switch | this | throw | true | try | typeof | uint |
| ulong | unchecked | unsafe | ushort | using | virtual | void |
| volatile | while |  |  |  |  |  |
| **Contextual Keyword trong C#** | | | | | | |
| add | alias | ascending | descending | dynamic | from | get |
| global | group | into | join | let | orderby | partial (type) |
| partial (method) | remove | select | set |  |  |  |

Kiểu dữ liệu (Data Types)

Các biến trong C# được phân chia thành các kiểu sau:

* + Kiểu giá trị (Value type)
  + Kiểu tham chiếu (Reference type)
  + Kiểu con trỏ (Pointer type)

**Kiểu giá trị trong C#**

Các biến kiểu giá trị có thể được gán một giá trị một cách trực tiếp. Chúng được kế thừa từ lớp System.ValueType.

Kiểu giá trị trực tiếp chứa dữ liệu. Một số ví dụ là int, char, và float, tương ứng giữ số nguyên, chữ cái, và số thực. Khi bạn khai báo một kiểu int, hệ thống cấp phát bộ nhớ để lưu giá trị đó.

Bảng sau liệt kê các kiểu giá trị có sẵn trong C# 2010:

|  |  |  |  |
| --- | --- | --- | --- |
| **Kiểu** | **Biểu diễn** | **Dãy giá trị** | **Giá trị mặc định** |
| bool | Giá trị Boolean | True hoặc False | False |
| byte | Kiểu unsigned integer (8 bit) | 0 tới 255 | 0 |
| char | Kiểu Unicode character (16 bit) | U +0000 tới U +ffff | '\0' |
| decimal | Kiểu thập phân (128 bit) | (-7.9 x 1028 tới 7.9 x 1028) / 100 to 28 | 0.0M |
| double | Kiểu double (64 bit) | (+/-)5.0 x 10-324 tới (+/-)1.7 x 10308 | 0.0D |
| float | Kiểu float (32 bit) | -3.4 x 1038 tới + 3.4 x 1038 | 0.0F |
| int | Kiểu integer (32 bit) | -2,147,483,648 tới 2,147,483,647 | 0 |
| long | Kiểu signed integer (64 bit) | -9,223,372,036,854,775,808 tới 9,223,372,036,854,775,807 | 0L |
| sbyte | Kiểu signed integer (8 bit) | -128 tới 127 | 0 |
| short | Kiểu signed integer (16 bit) | -32,768 tới 32,767 | 0 |
| uint | Kiểu unsigned integer (32 bit) | 0 tới 4,294,967,295 | 0 |
| ulong | Kiểu unsigned integer (64 bit) | 0 tới 18,446,744,073,709,551,615 | 0 |
| ushort | Kiểu unsigned integer (16 bit) | 0 tới 65,535 | 0 |

**Từ khóa sizeof trong C#**

Để lấy kích cỡ chính xác của một kiểu hoặc một biến trên một nền tảng cụ thể, bạn có thể sử dụng phương thức **sizeof**. Biểu thức *sizeof(type)* hiển thị kích cỡ của đối tượng hoặc kiểu bằng giá trị byte. Ví dụ dưới đây để lấy kích cỡ của kiểu **int** trên bất kỳ máy tính:

using System;

namespace TPCenter

{

class TestCsharp

{

static void Main(string[] args)

{

Console.WriteLine("Tu khoa sizeof trong C#");

Console.WriteLine("-------------------------");

Console.WriteLine("Kich co cua kieu du lieu int la: {0}", sizeof(int));

Console.WriteLine("Kich co cua kieu du lieu float la: {0}", sizeof(float));

Console.WriteLine("Kich co cua kieu du lieu double la: {0}", sizeof(double));

Console.WriteLine("Kich co cua kieu du lieu char la: {0}", sizeof(char));

Console.ReadLine();

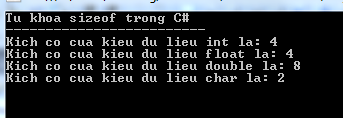
Console.ReadKey();

}

}

}

Biên dịch và chạy chương trình C# trên sẽ cho kết quả sau:



Kiểu tham chiếu trong C#

Kiểu tham chiếu không chứa dữ liệu thực sự được lưu giữ trong một biến, nhưng chúng chứa một tham chiếu tới các biến.

Nói cách khác, chúng tham chiếu tới một vị trí bộ nhớ. Việc sử dụng nhiều biến, thì kiểu tham chiếu có thể tham chiếu tới tới một vị trí bộ nhớ. Nếu dữ liệu trong vị trí bộ nhớ bị thay đổi bởi một trong số các biến, thì biến khác tự động phản ánh sự thay đổi về giá trị này. Ví dụ các kiểu tham chiếu **có sẵn** trong C# là: **object**, **dynamic,** và **string**.

Kiểu object trong C#

Kiểu **object** là lớp cơ sở cơ bản cho tất cả kiểu dữ liệu trong C# Common Type System (CTS). Object là một alias cho lớp System.Object. Các kiểu object có thể được gán giá trị của bất kỳ kiểu, kiểu giá trị, kiểu tham chiếu, kiểu tự định nghĩa (user-defined) khác. Tuy nhiên, trước khi gán các giá trị, nó cần một sự chuyển đổi kiểu.

Khi một kiểu giá trị được chuyển đổi thành kiểu object, nó được gọi là **boxing** và ngược lại, khi một kiểu object được chuyển đổi thành kiểu giá trị, nó được gọi là **unboxing**.

object obj;

obj = 100; // day la vi du boxing

Kiểu Dynamic trong C#

Bạn có thể lưu giữ bất kỳ kiểu giá trị nào trong biến kiểu dữ liệu dynamic. Việc kiểm tra các kiểu biến này diễn ra tại run time.

Cú pháp để khai báo một kiểu dynamic trong C# là:

dynamic <tên\_biến> = giá\_trị;

Ví dụ

dynamic d = 20;

Kiểu dynamic là tương tự với kiểu object, ngoại trừ việc kiểm tra cho các biến kiểu object diễn ra tại compile time, trong khi việc kiểm tra các biến kiểu dynamic diễn ra tại run time.

Kiểu string trong C#

Kiểu **string** trong C# cho phép bạn gán bất kỳ giá trị chuỗi nào cho một biến. Kiểu string là một *alias* cho lớp **System.String**. Nó kế thừa từ kiểu object. Giá trị cho một kiểu string có thể được gán bởi sử dụng các hằng chuỗi trong hai mẫu: quoted và @quoted.

Ví dụ:

String str = "Hoc C# co ban va nang cao tai VietJack";

Và một hằng chuỗi @quoted trông như sau:

@"VietJack Team";

Các kiểu tự định nghĩa (user-defined) trong C# là: Class, Interface, hoặc Delegate. Chúng ta sẽ bàn về các kiểu này trong các chương sau.

## Kiểu con trỏ trong C#

Các biến kiểu con trỏ lưu giữ địa chỉ bộ nhớ của kiểu khác. Các con trỏ trong C# có khả năng như con trỏ trong C hoặc C++.

Cú pháp để khai báo một kiểu con trỏ trong C# là:

type\* identifier;

Ví dụ:

char\* cptr;

int\* iptr;

Chúng ta sẽ thảo luận về kiểu con trỏ ở chương: Unsafe Code trong C#.

Chuyển đổi kiểu dữ liệu(Type Conversion)

Biến(Variables)

Hằng(Constant/Literal)

Toán tử(Operators)

Lệnh If, Else, Switch(Decision Making)

Vòng lặp(Loop)

Tính đóng gói(Encapsulation)

Phương thức(Method)

Nullable

Mảng(Array)

Chuỗi(String)

Cấu trúc(Struct)

Enum

Lớp(Classes)

Tính kế thừa(Inheritance)

Tính đa hình(Polymorphism)

Nạp chồng toán tử(Operator Overloading)

Interface

Namespaces

Chỉ thị tiền xử lý(Preprocessor Directives)

Regular Expression

Exception Handling

File I/O

Attributes

Reflection

Properties

Indexer

Delegate

Event

Collection

Generic

Phương thức nặc danh(Anonymous Methods)

Unsafe Code

Multithreading

Tổng kết