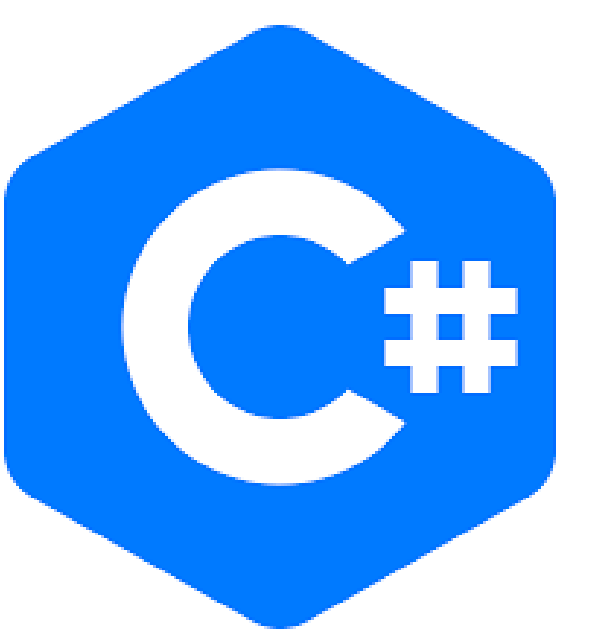
Tổng Quan Về C#



Nội dung

1	Chương trình Hello World
2	Nhập/Xuất trong C#
3	Nội suy chuỗi (string interpolation)
4	Cấu trúc điều khiển
5	Cấu trúc lặp

Chương trình “Hello world”

```
using System;

namespace SampleApplication
{
    class HelloWorld
    {
        static void Main()
        {
            Console.WriteLine("Hello world!");
        }
    }
}
```

Chương trình “Hello world”

using System;

using là một dẫn hướng biên dịch. Lệnh trên báo cho trình biên dịch tìm những class (hoặc những thứ khác) từ namespace có tên System, cụ thể nó sử dụng class Console từ System. Nếu không có dẫn hướng này, ta phải viết đầy đủ System.Console thay vì chỉ viết Console

Chương trình “Hello world”

namespace SampleApplication

Khai báo một namespace có tên là SampleApplication. Namespace là cách ta nhóm tất cả các class và những thứ khác có liên quan nhau thành một “không gian tên” để dễ quản lý và tránh trùng tên khi dự án lớn. Cú pháp khai báo namespace:

```
namespace Tên_Namespace
{
    // Khai báo các class, các namespace con, …
}
```

Chương trình "Hello world"

Một số lưu ý về namespace:

- Tên file chứa khai báo namespace nên đặt giống tên của namespace và ở dạng PascalCase

  Ví dụ: SampleApplication.cs

- Việc bao bọc các class và những thứ liên quan vào một namespace là không bắt buộc, nhưng khuyến nghị nên làm
- Ta có thể dùng bí danh cho những namespace "dài", cú pháp:

  using Tên_Bí_Danh = Tên_Namespace;

  Ví dụ: using Printing = System.Drawing.Printing;
- Để định vị tới một thành phần trong namespace, ta dùng dấu chấm .

Chương trình “Hello world”

class HelloWorld

Khai báo một class có tên HelloWorld. Mỗi ứng dụng phải có ít nhất một khai báo class

Chương trình "Hello world"

static void Main()

- Đây là nơi ứng dụng bắt đầu chạy (còn gọi là entry point của ứng dụng)
- Đối với mỗi ứng dụng, một trong những phương thức (method) trong một class phải có tên là Main, nếu không, ứng dụng sẽ không chạy
- static: phương thức này thuộc class, không cần tạo đối tượng để gọi
- void: nghĩa là phương thức không trả về giá trị

Chương trình "Hello world"

Các dạng hàm Main hợp lệ:

- ✓ static void Main()
- ✓ static void Main(string[] args)
- ✓ static int Main()
- ✓ static int Main(string[] args)

string[] args được dùng khi ta muốn nhận tham số từ dòng lệnh

Chương trình “Hello world”

```
Console.WriteLine("Hello world!");
```

Lệnh này gọi phương thức WriteLine của lớp Console (trong namespace System), in ra dòng chữ Hello world! trên màn hình console và xuống dòng

Nhập/Xuất trong C#

- Sử dụng các phương thức của lớp Console
- Các phương thức hay dùng:
  - ✓ Console.WriteLine(): hiển thị ra màn hình và xuống dòng
  - ✓ Console.Write(): hiển thị ra màn hình
  - ✓ Console.ReadLine(): đọc một chuỗi ký tự từ bàn phím
  - ✓ Console.Read(): đọc một ký tự từ bàn phím

Nhập/Xuất trong C#

Lưu ý:

Hàm Console.ReadLine() trả về chuỗi nên để nhập một số nguyên, ta cần chuyển từ chuỗi về số nguyên:

```
int number = int.Parse(Console.ReadLine());
```

Nội suy chuỗi (string interpolation)

Nội suy chuỗi được sử dụng trong trường hợp cần chèn giá trị của biến/biểu thức vào trong chuỗi để xuất ra, tránh phải dùng đến phép toán nối chuỗi dài dòng

Cú pháp: bắt đầu chuỗi bằng dấu $, những vị trí cần chèn bỏ vào dấu ngoặc {}

Ví dụ:

```
string myName = “Mary”;
int myAge = 20;
Console.WriteLine($”Tôi là {myName}. Tôi {myAge} tuổi”);
```

Cấu trúc điều khiển if

```
if (Biểu_Thức_Điều_Kiện)
{
    Khối_Lệnh
}
```

Ví dụ:

```
if (score>=50)
{
    Console.WriteLine(“Passed”);
}
```

Cấu trúc điều khiển if…else…

```
if (Biểu_Thức_Điều_Kiện)
{
   Khối_Lệnh
}
else
{
    Khối_Lệnh
}
```

Ví dụ:

```
If (score>=50)
{
    Console.WriteLine(“Passed”);
}
else
{
    Console.WriteLine(“Failed”);
}
```

Cấu trúc điều khiển if…else…

Ta có thể lồng if else nhiều cấp. Ví dụ:

```
if (score>=90)
{
    Console.WriteLine(“A”);
}
else if (score>=80)
{
    Console.WriteLine(“B”);
}
else if (score>=70)
{
    Console.WriteLINE(“C”);
}
else
{
    Console.WriteLine(“D”);
}
```

Cấu trúc điều khiển if…else…

Có thể thay thế if…else… bằng toán tử điều kiện ?:

Biểu_Thức_Điều_Kiện ? Lệnh_Khi_Đúng : Lệnh_Khi_Sai;

Ví dụ:

Console.WriteLine(score>=50?”Passed”:”Failed”);

string kq = (i % 2) == 0 ? “chẵn” : “lẻ”;

Cấu trúc điều khiển switch…case…

```
switch (Biểu_Thức)
{
   case Giá_Trị_1:
      Các_câu_lệnh_1;
      break;
    case Giá_Trị_2:
      Các_câu_lệnh_2;
      break;
                ……..
     default:
        Các_câu_lệnh_mặc_ định;

}
```

Cấu trúc lặp for

```
for (Khởi_Tạo;Điều_Kiện;Bước_Nhảy)
{
    Khối_Lệnh
}
```

Ví dụ:

```
for (int i=1; i<=5; i++)
{
    Console.WriteLine(“Số “ + i);
}
```

Cấu trúc lặp while

```
while (Biểu_Thức_Điều_Kiện)
{
    Khối_Lệnh
}
```

Ví dụ:

```
int pow = 3;
while (pow<=100)
{
    pow = 3 * pow;
}
```

Cấu trúc lặp do … while

```
do
{
   Khối_Lệnh
}
while (Biểu_Thức_Điều_Kiện)
```

Ví dụ:

```
int i = 1;
do
{
    Console.WriteLine($“Số {i}“);
    i++;
}
while (i<=5);
```

Cấu trúc lặp foreach

foreach dùng để duyệt qua các phần tử trong một tập hợp (mảng, danh sách, collection, …) thực hiện những lệnh tương ứng mà không quan tâm đến chỉ số (index) của phần tử

```
foreach (Kiểu_Dữ_Liệu Tên_Biến in Mảng)
{
    Khối_Lệnh (cần thực hiện với Biến)
}
```

Ví dụ:

```
int[] numbers = { 1, 2, 3, 4, 5 };
foreach (int num in numbers)
{
      Console.WriteLine("Số: " + num);
}
```

Cấu trúc lặp foreach

```
string[] names = { "An", "Bình", "Chi", "Dũng" };
foreach (string name in names)
{
    Console.WriteLine("Xin chào, " + name);
}

List<string> fruits = new List<string>() { "Táo",
"Cam", "Xoài" };
foreach (string fruit in fruits)
{
     Console.WriteLine("Trái cây: " + fruit);
}
```