

Học nhanh C# cho người mới bắt đầu

(<https://o7planning.org/vi/10333/hoc-nhanh-csharp-cho-nguoi-moi-bat-dau>)

- 1- Giới thiệu
- 2- Tạo Project C# đầu tiên của bạn
- 3- Giải thích cấu trúc của một class
- 4- Giải thích cấu trúc Project
- 5- Chú ý quan trọng với một chương trình C#
- 6- Thêm mới class
- 7- Các kiểu dữ liệu trong C#
- 8- Biến và khai báo
- 9- Câu lệnh rẽ nhánh
 - 9.1- Câu lệnh If-else
 - 9.2- Câu lệnh Switch-Case
- 10- Vòng lặp trong C#
 - 10.1- Vòng lặp for
 - 10.2- Vòng lặp while
 - 10.3- Vòng lặp do-while
 - 10.4- Lệnh break trong vòng lặp
 - 10.5- Lệnh continue trong vòng lặp
- 11- Mảng trong C#
 - 11.1- Mảng một chiều
 - 11.2- Mảng hai chiều
 - 11.3- Mảng của mảng
- 12- Class, Constructor và đối tượng
- 13- Trường (Field)
- 14- Phương thức (Method)
- 15- Thừa kế trong C#
- 16- Thừa kế và đa hình trong C#

1- Giới thiệu

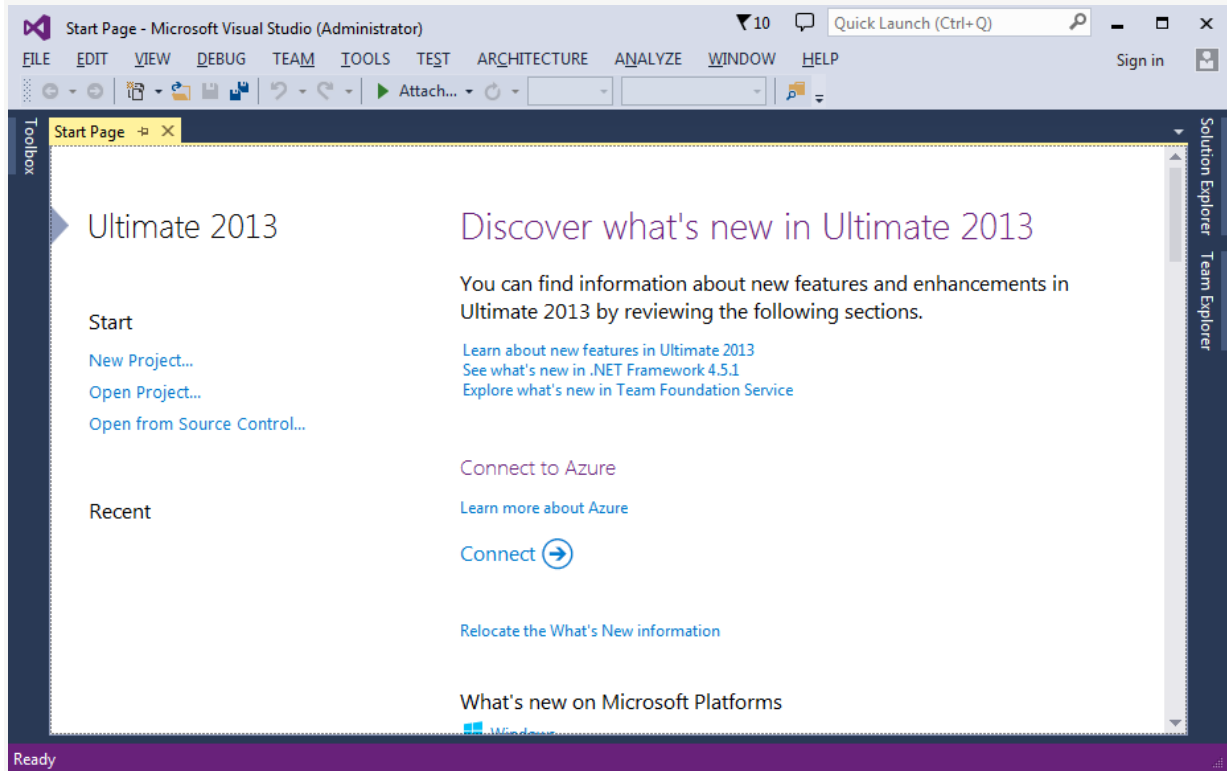
Đây là tài liệu hướng dẫn học C# cho người mới bắt đầu. Để lập trình C# bạn phải cài đặt công cụ lập trình **Visual Studio**. Bạn có thể xem hướng dẫn download và cài đặt tại:

- [Bắt đầu với C# cần những gì?](#)

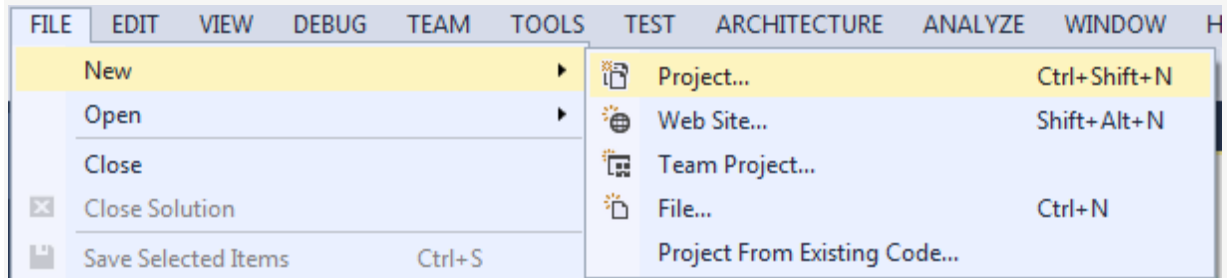
Nếu bạn mới bắt đầu với C#, bạn nên đọc bài viết này lần lượt từ trên xuống, nó sẽ giúp bạn có cái nhìn tổng quan trước khi đi vào các tài liệu chi tiết khác.

2- Tạo Project C# đầu tiên của bạn

Đây là hình ảnh đầu tiên khi bạn mở **Visual Studio**.

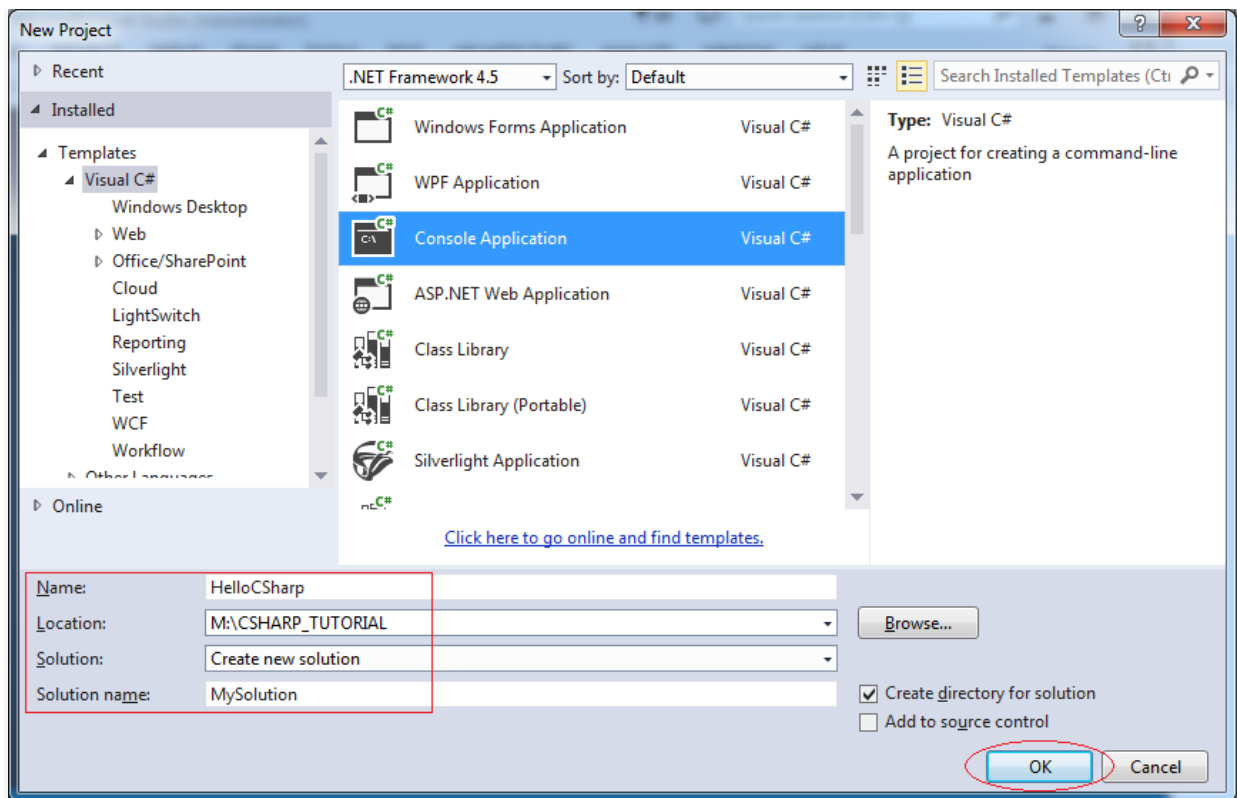


Tạo mới một Project:

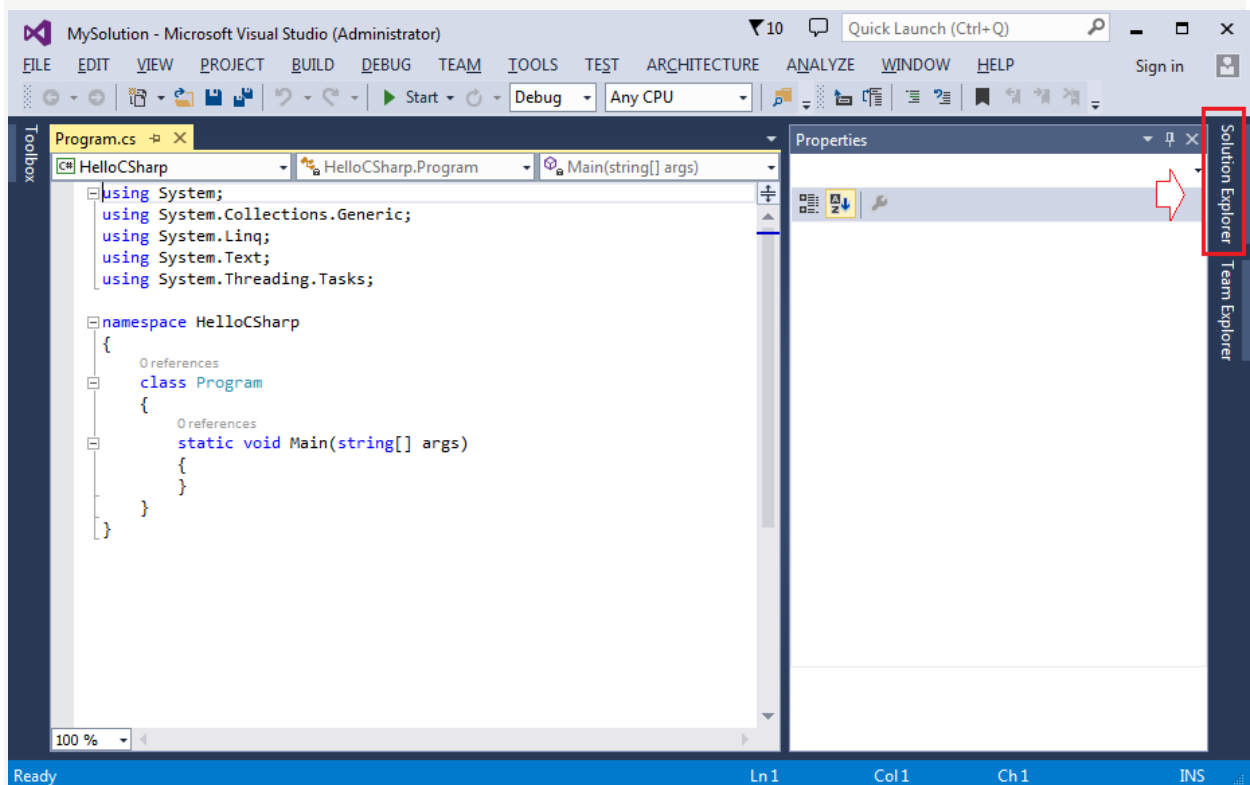


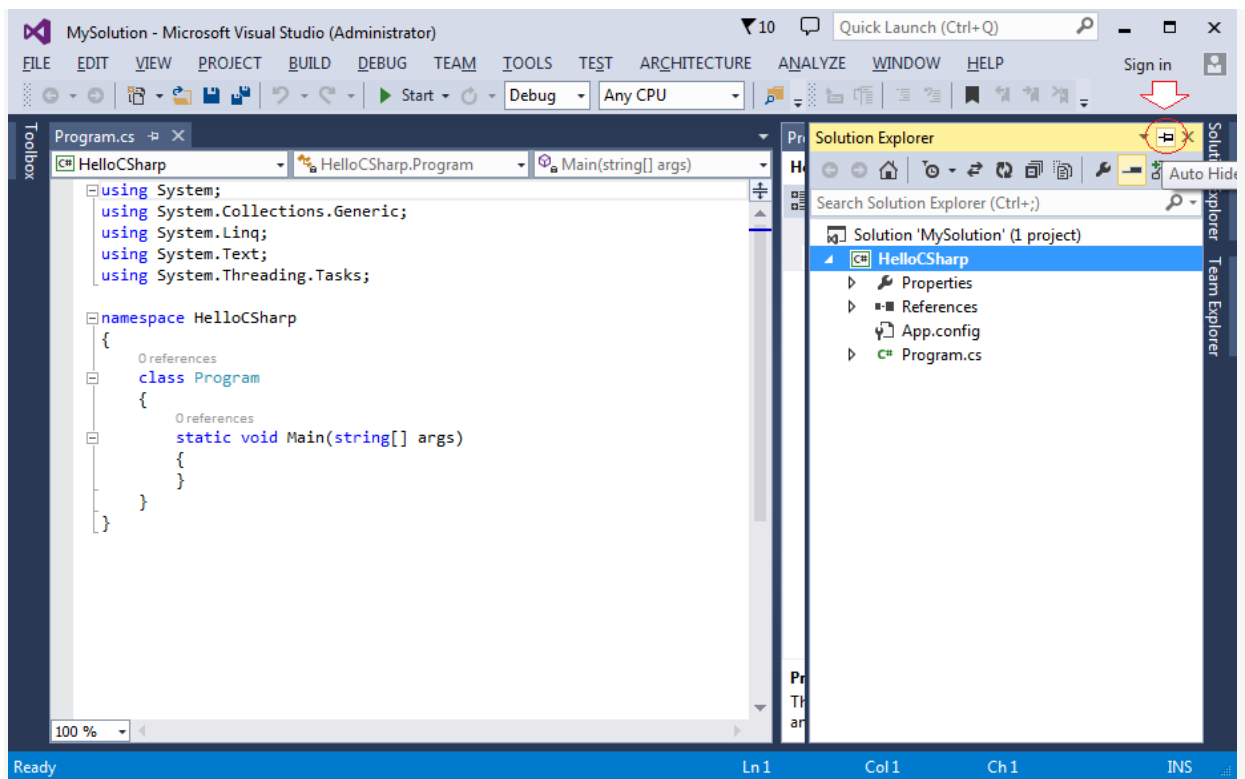
Chúng ta tạo một Project đơn giản (Ứng dụng **Console**, là ứng dụng không có giao diện). Nhập vào:

- **Name:** HelloCSharp
- **Solution:** Create new solution
- **Solution Name:** MySolution



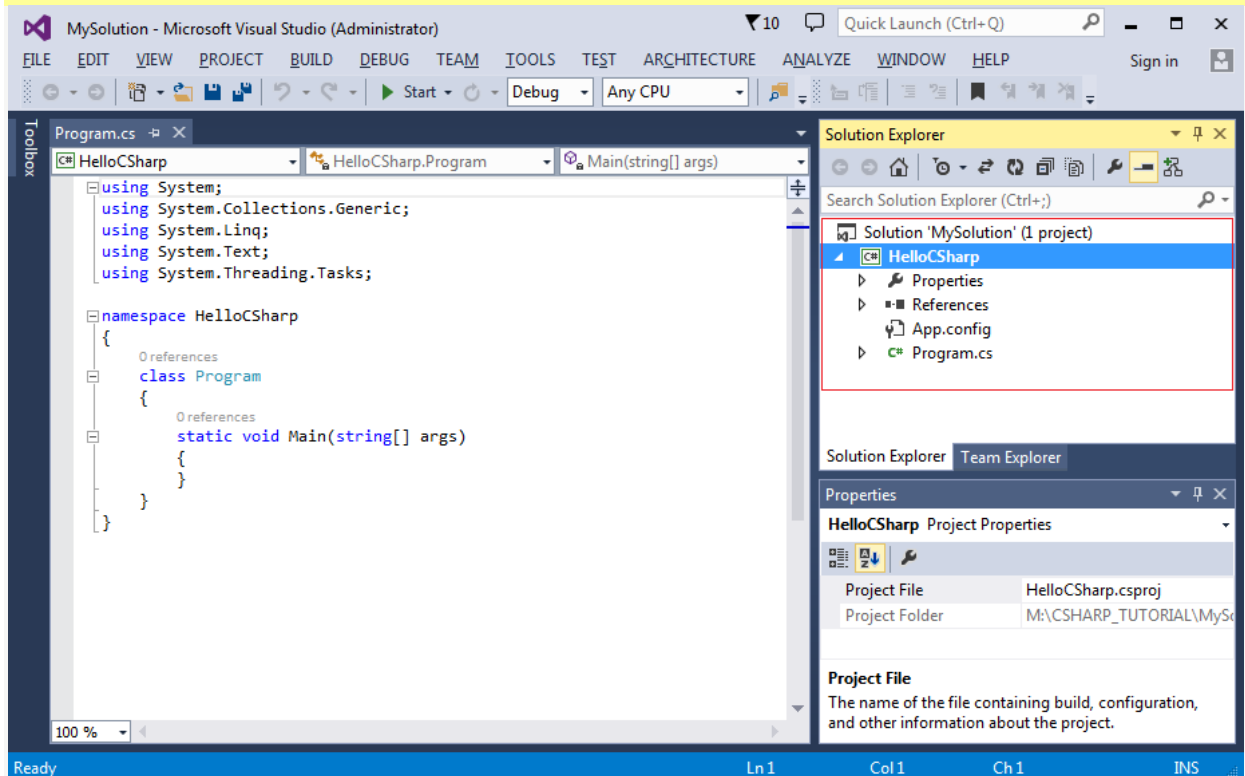
Đây là hình ảnh Project của bạn đã được tạo ra. Bạn cần nhấn vào **Solution Explorer** để xem cấu trúc của Project vừa được tạo ra đó.





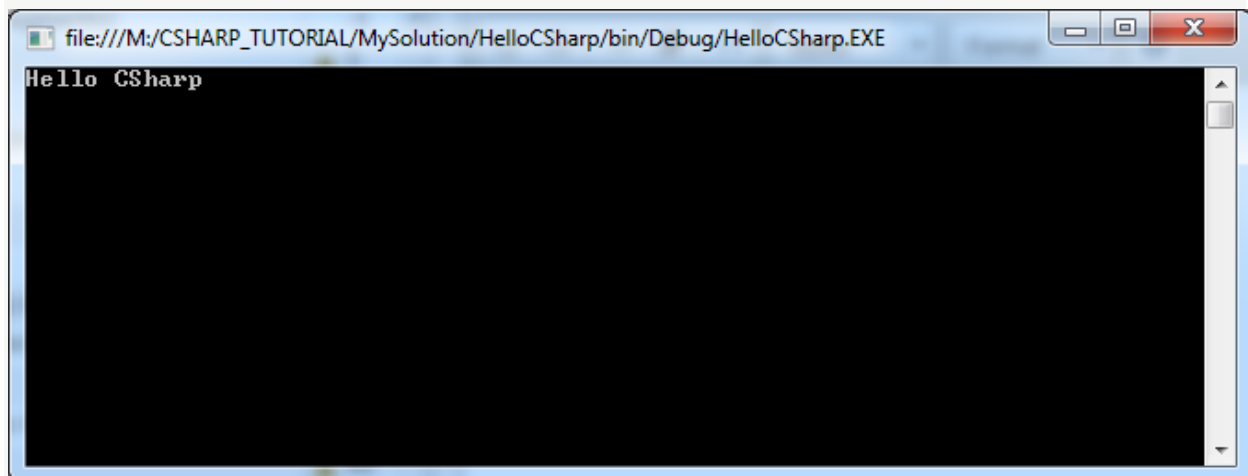
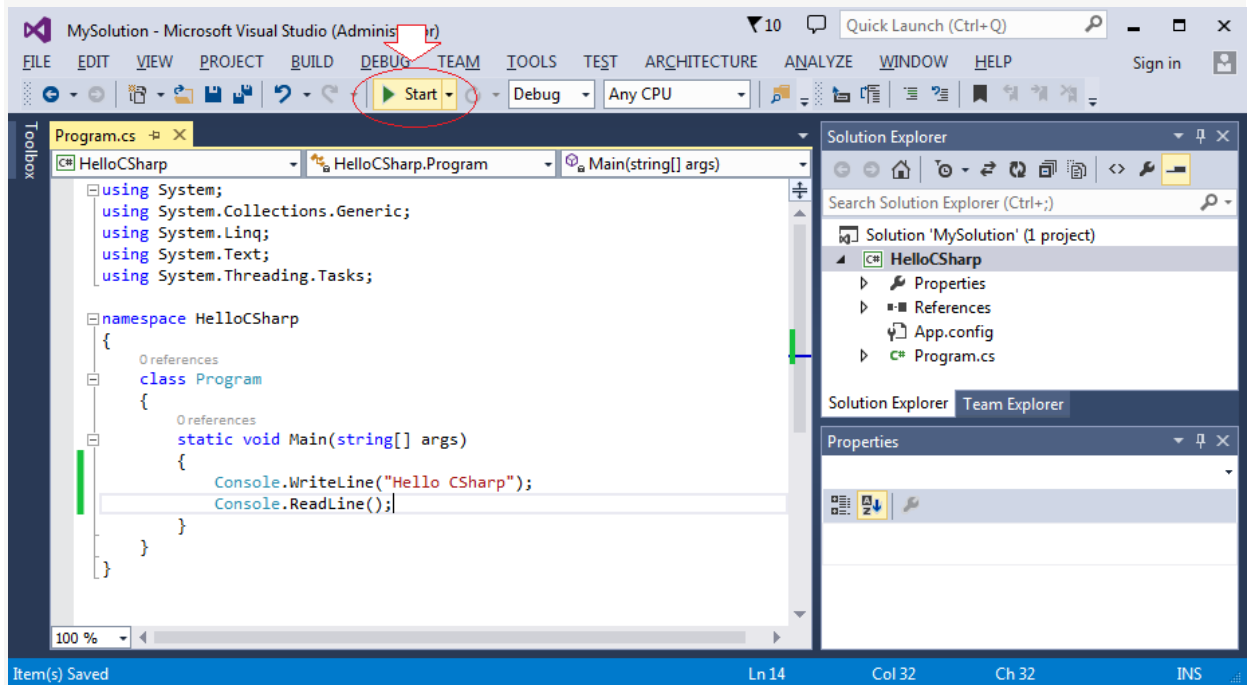
Visual Studio tạo ra một **Solution** (Giải pháp) có tên là **MySolution** và chứa bên trong nó là một **Project** có tên **HelloCSharp**. Và tạo mặc định một lớp có tên **Program** (Ứng với file **Program.cs**).

*Chú ý: Một **Solution** (Giải pháp) có thể có một hoặc nhiều **Project**.*



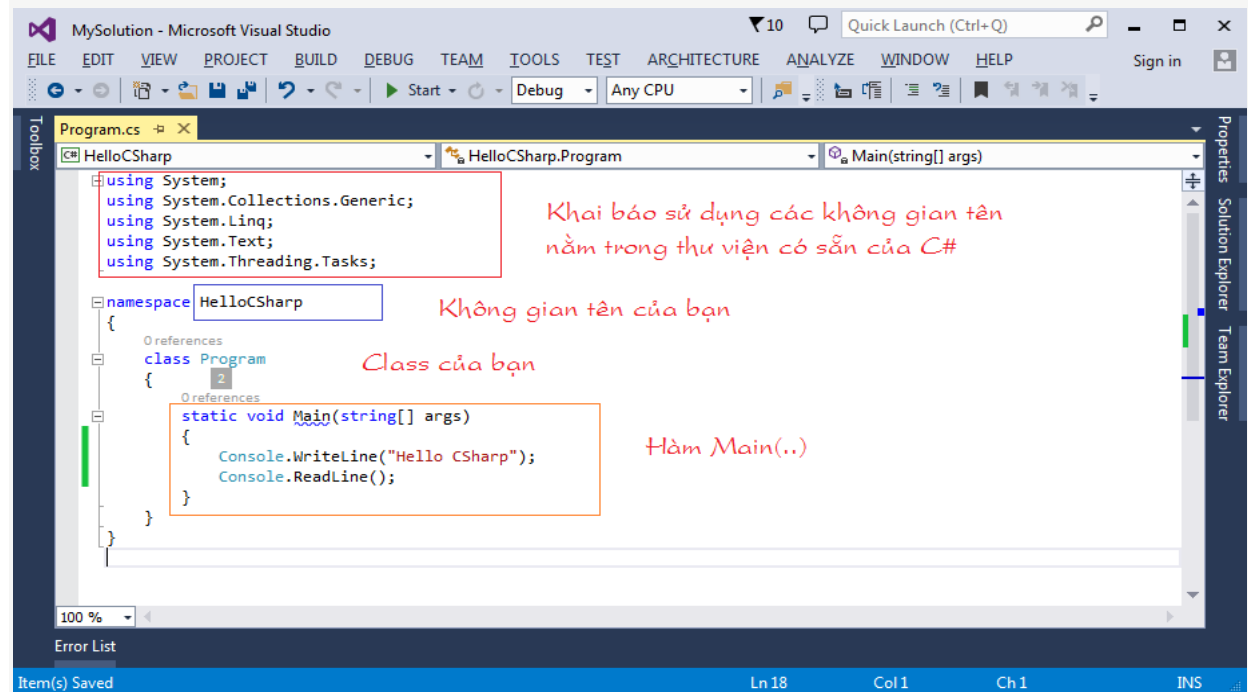
Sửa code của lớp **Program**, để khi chạy nó in ra màn hình **Console** một dòng chữ "**Hello CSharp**", và chờ đợi người dùng nhập vào một dòng văn bản bất kỳ trước khi kết thúc.

Nhấn vào **Start** để chạy lớp **Program**.



3- Giải thích cấu trúc của một class

Hình minh họa dưới đây là cấu trúc của lớp **Program**, nó nằm trong không gian tên (namespace) **HelloCSharp**. Một không gian tên có thể chứa một hoặc nhiều lớp.



Nếu bạn muốn sử dụng một lớp nào đó, bạn phải khai báo sử dụng lớp đó, hoặc khai báo sử dụng không gian tên (namespace) chứa lớp đó.

?

```
1 // Khai báo sử dụng không gian tên (namespace) System.  
2 // (Nghĩa là có thể sử dụng tất cả các lớp,..  
3 // có trong namespace này).  
4 using System;
```

Khi chương trình được chạy, phương thức **Main(string[])** sẽ được gọi thực thi.

1. **static** là từ khóa thông báo rằng đây là phương thức tĩnh.
2. **void** là từ khóa thông báo rằng phương thức này không trả về gì cả.
3. **args** là tham số của phương thức nó là một mảng các **string** (chuỗi) - **string[]**.

?

```
1 static void Main(string[] args)  
2 {  
3     // Ghi ra màn hình Console một dòng chữ.  
4     Console.WriteLine("Hello CSharp");  
5  
6     // Đợi người dùng gõ vào bất kỳ và nhấn Enter trước  
7     //khi kết thúc chương trình.  
8     Console.ReadLine();  
9 }
```

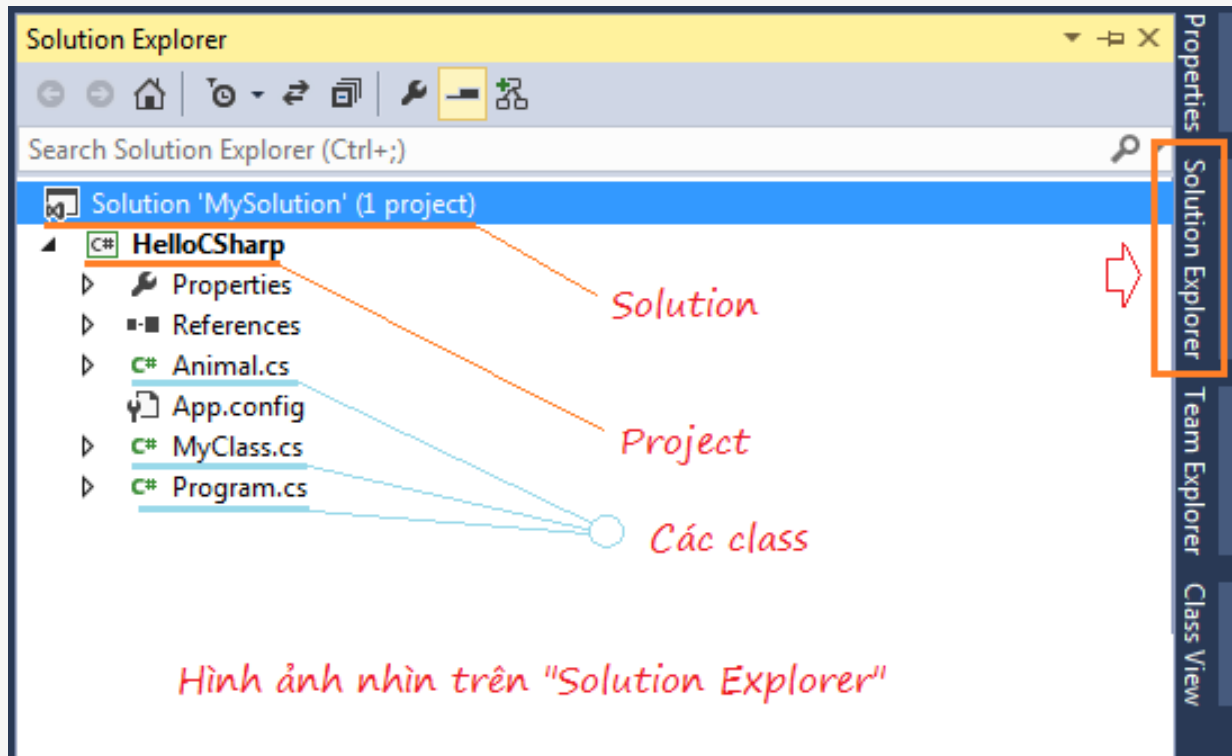
Sau khi đã khai báo sử dụng không gian tên (namespace) **System**, bạn có thể sử dụng lớp **Console** nằm trong **namespace** này. **WriteLine(string)** là một phương thức tĩnh của lớp **Console**, nó ghi ra màn hình một chuỗi.

?

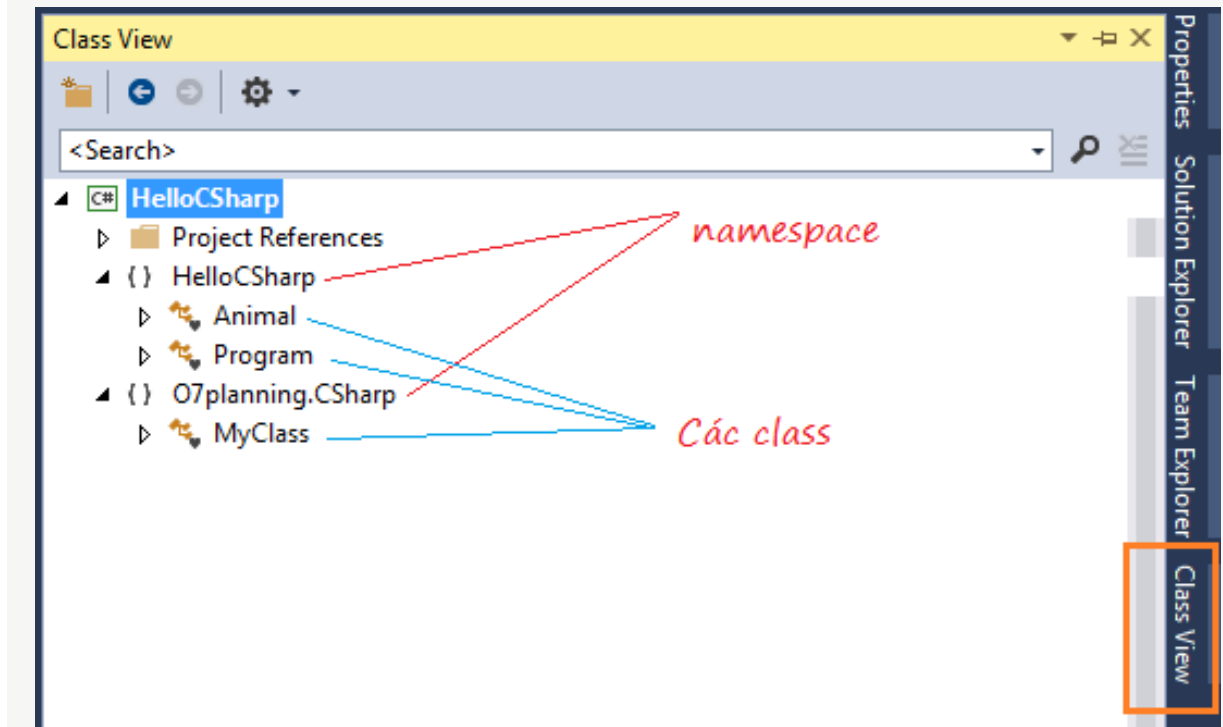
```
1 // Khai báo để sử dụng không gian tên System
2 //(System namespace).
3 // (Nó chứa lớp Console).
4 using System;
5
6 // Và bạn có thể sử dụng lớp Console:
7 Console.WriteLine("Hello CSharp");
8
9 // Nếu bạn không khai báo sử dụng không gian tên System.
10 // Nhưng muốn sử dụng lớp Console, bạn phải viết dài hơn:
11 System.Console.WriteLine("Hello CSharp");
```

4- Giải thích cấu trúc Project

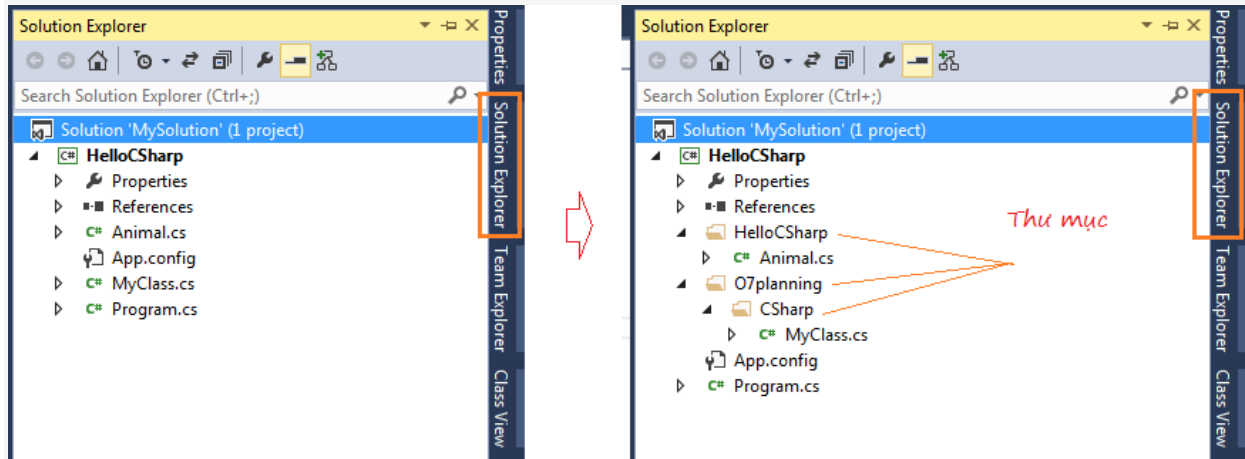
Một giải pháp (**Solution**) có thể chứa trong nó nhiều **Project**. Trong các **Project** chứa các lớp (class).



Khi nhìn trên "**Class view**" bạn có thể thấy được các lớp của bạn thuộc vào không gian tên (namespace) nào.



Trong **CSharp** bạn tạo ra một lớp **Animal** với không gian tên là **HelloCSharp**, lớp này mặc định sẽ nằm tại thư mục gốc của project. Bạn tạo ra một lớp khác là **MyClass** với không gian tên là **O7planning.CSharp**, lớp này cũng nằm tại thư mục gốc của project. Với một project lớn có nhiều class, cách tổ chức các file như vậy gây khó khăn cho bạn. Bạn có thể tạo ra các thư mục khác nhau để chứa các class, quy tắc do bạn quyết định, tuy nhiên tốt nhất bạn tạo ra các thư mục có tên là tên của **namespace**. Hãy học tập cách tổ chức của **Java**.

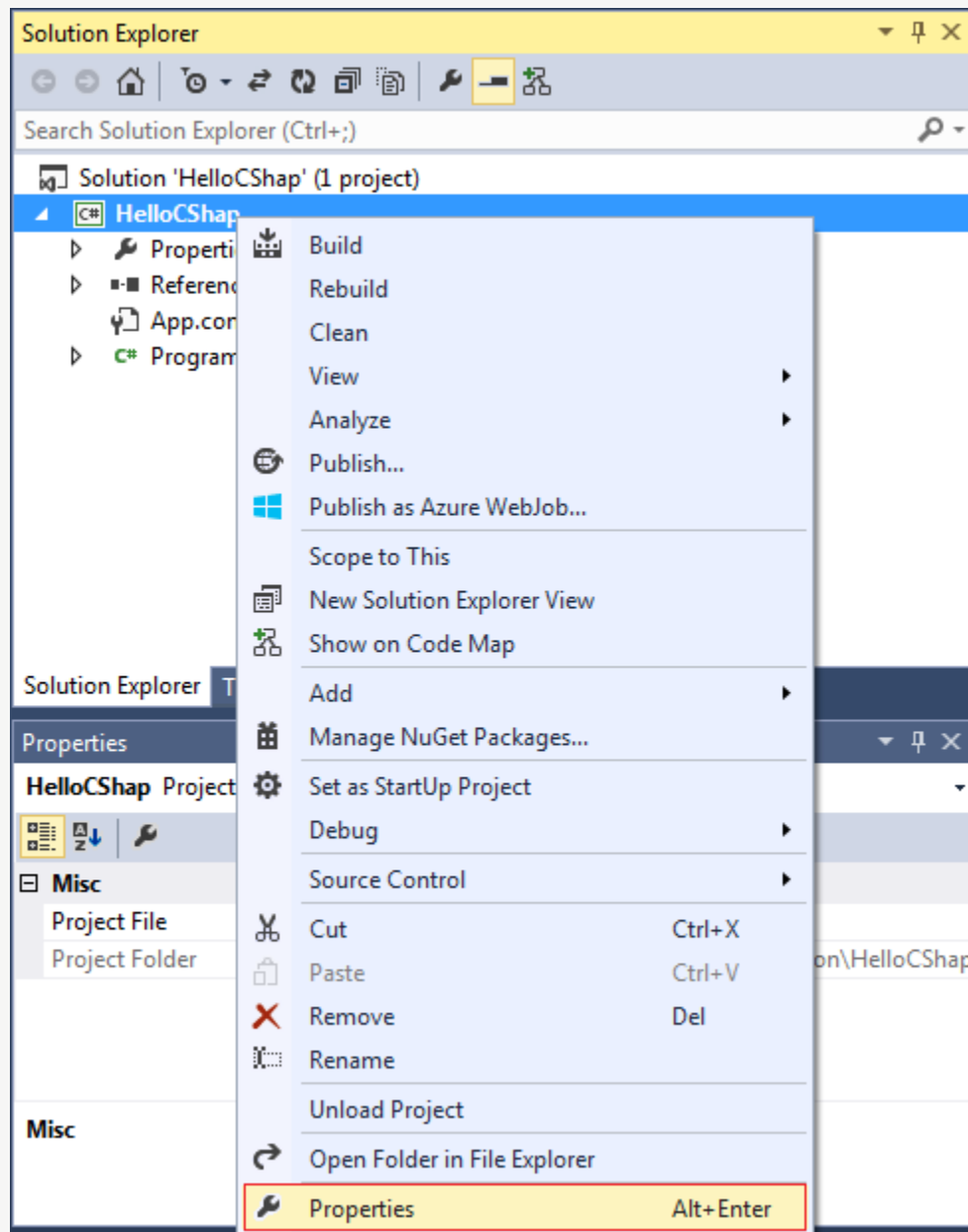


5- Chú ý quan trọng với một chương trình C#

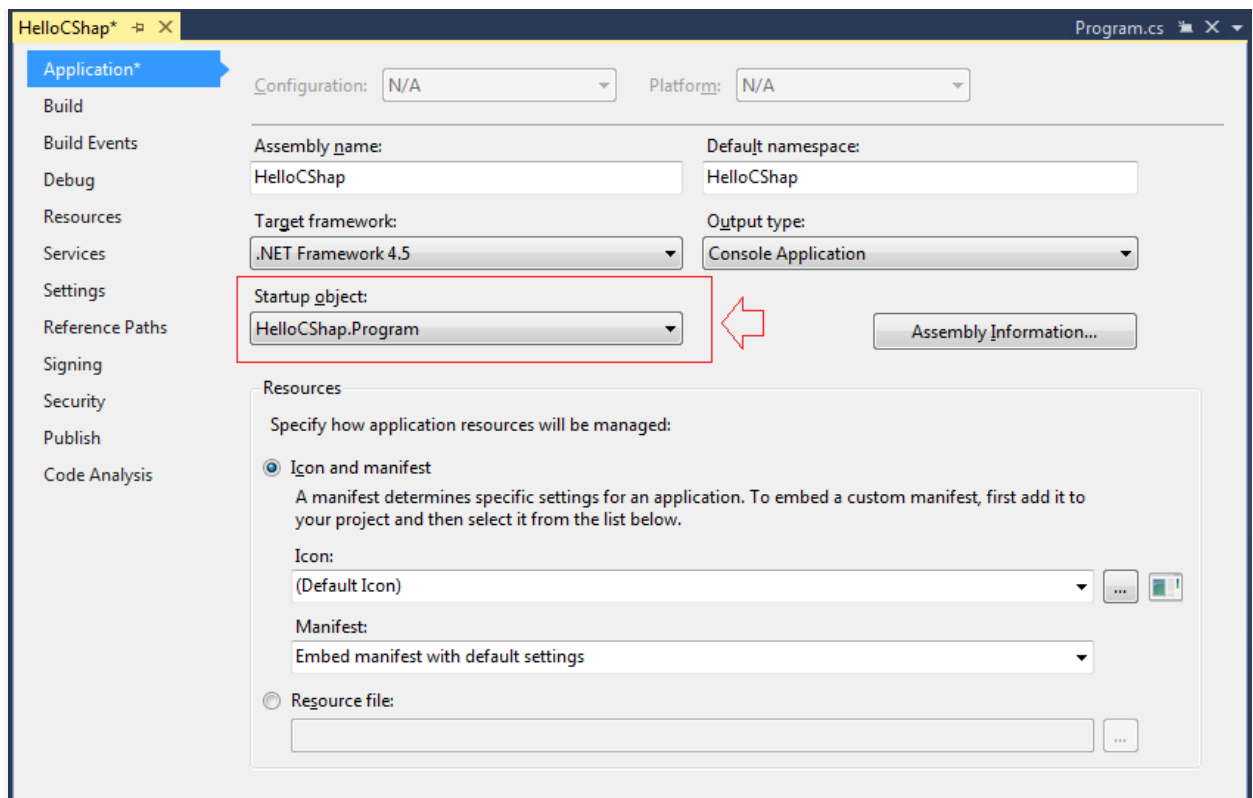
Trong một ứng dụng C# bạn cần khai báo rõ ràng một lớp có phương thức **Main(string[])** dùng làm điểm bắt đầu để chạy ứng dụng của bạn, điều này không bắt buộc nếu toàn bộ ứng dụng của bạn có duy nhất một lớp có phương thức **Main(string[])**, nhưng trong trường hợp có 2 lớp có phương thức **Main** nếu bạn không chỉ định rõ, một thông báo lỗi sẽ xuất hiện ra trong quá trình biên dịch.

Vì vậy tốt nhất bạn hãy khai báo rõ ràng lớp có phương thức **Main(string[])**, bạn có thể khai báo lại cho một lớp khác nếu muốn.

Nhấn phải chuột vào **HelloCSharp** project, chọn **Properties**:



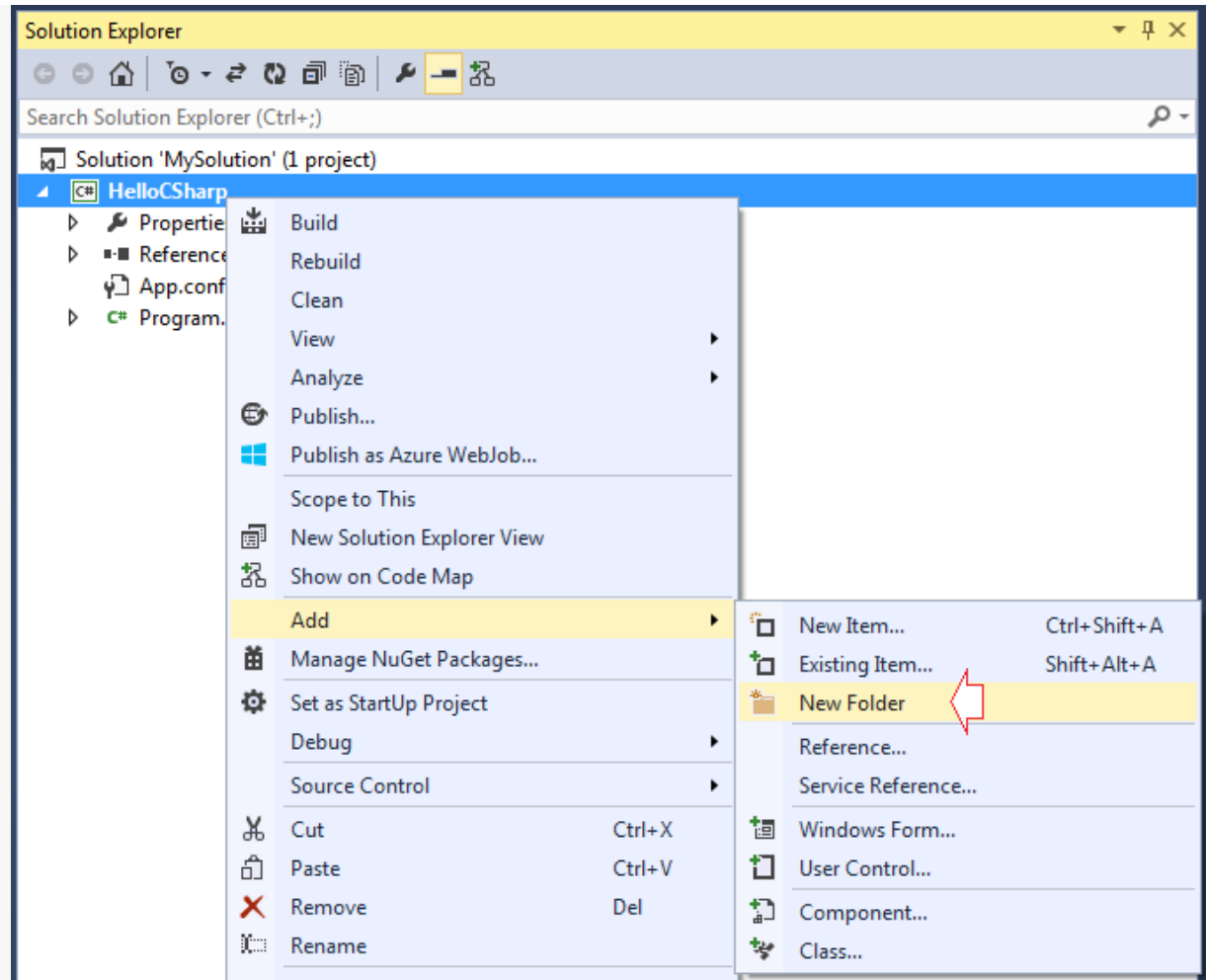
Chọn "**Startup object**" là một lớp có phương thức **Main(string[])** và **Save** lại.



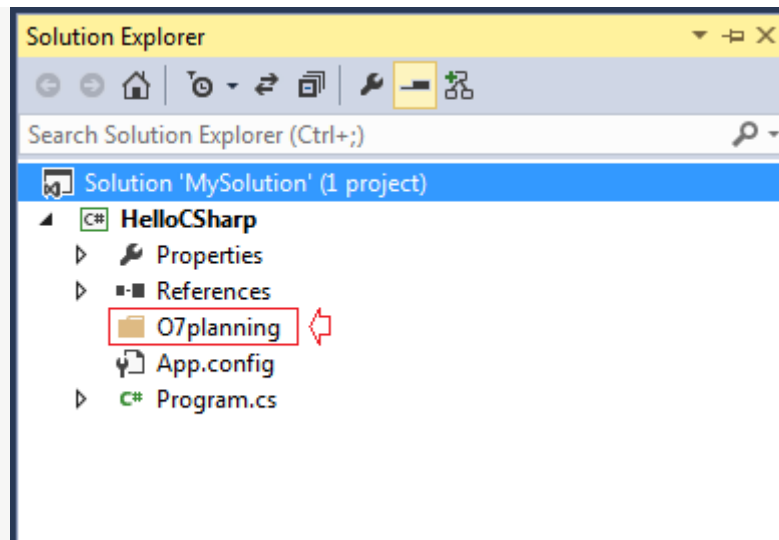
6- Thêm mới class

Bây giờ tôi thêm mới một lớp có tên **MyClass** với không gian tên **O7planning.CSharp**.
Trên "**Solution Explorer**" nhấn phải chuột vào project chọn:

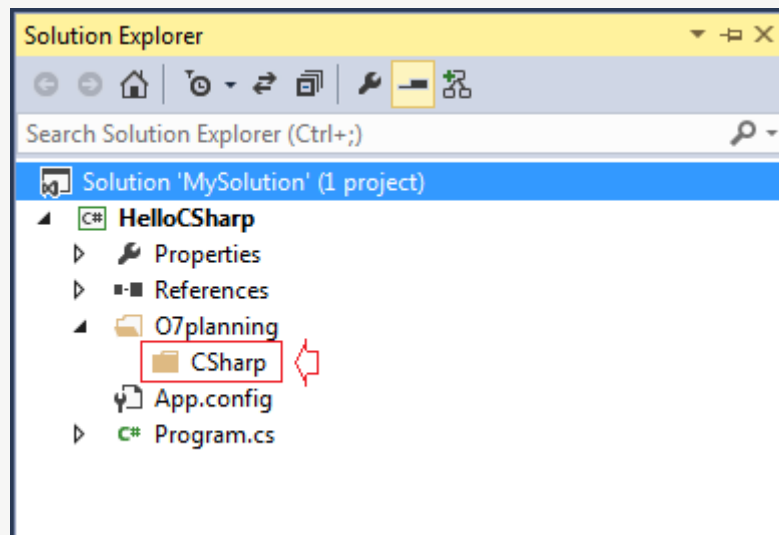
- **Add/New Folder**



Đặt tên cho thư mục là **O7planning**.

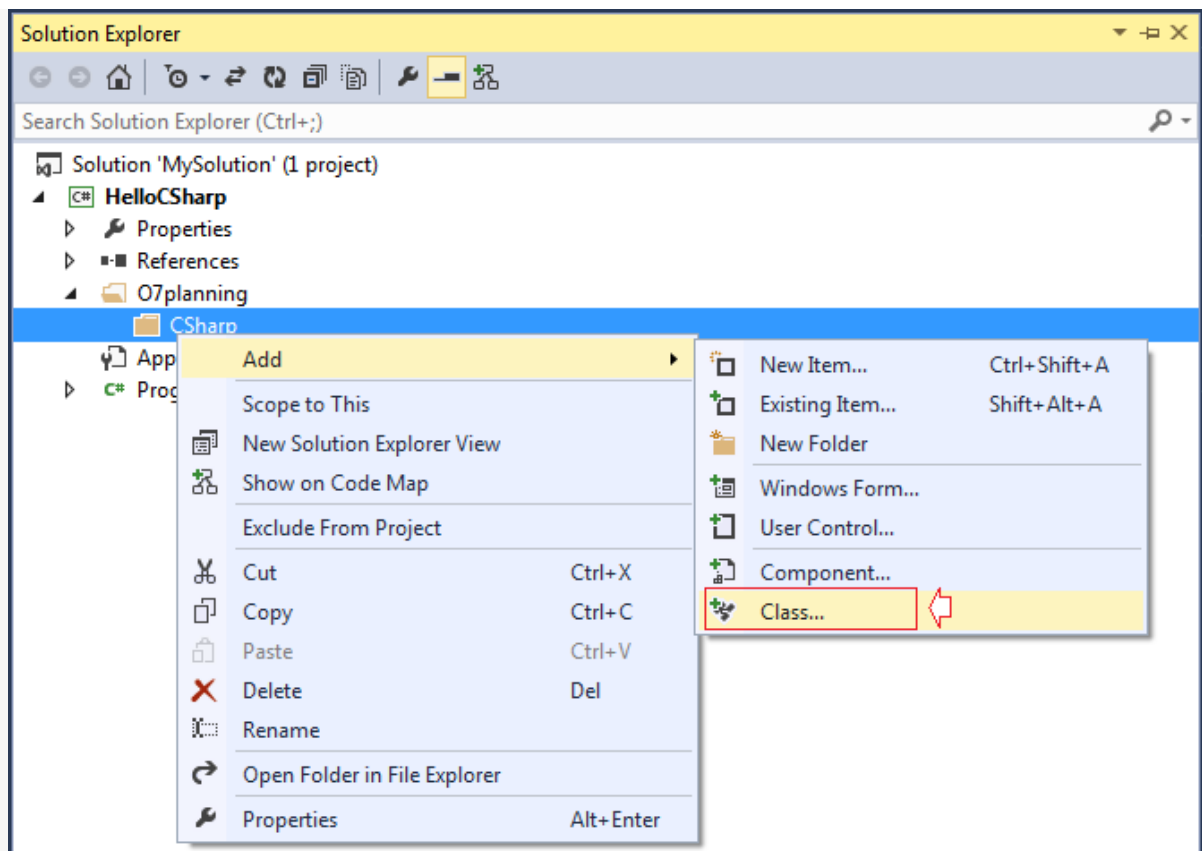


Tiếp tục tạo một thư mục **"CSharp"** là con của thư mục **"O7planning"**.

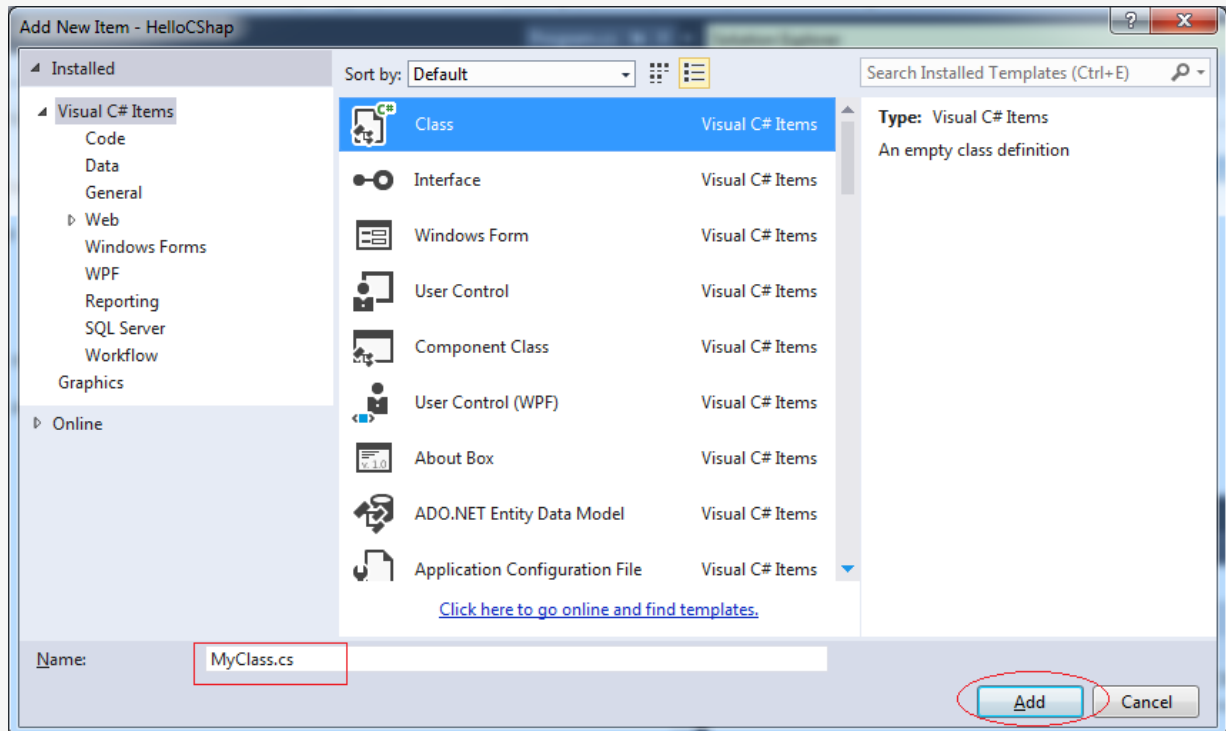


Nhấn phải chuột vào thư mục **"CSharp"** chọn:

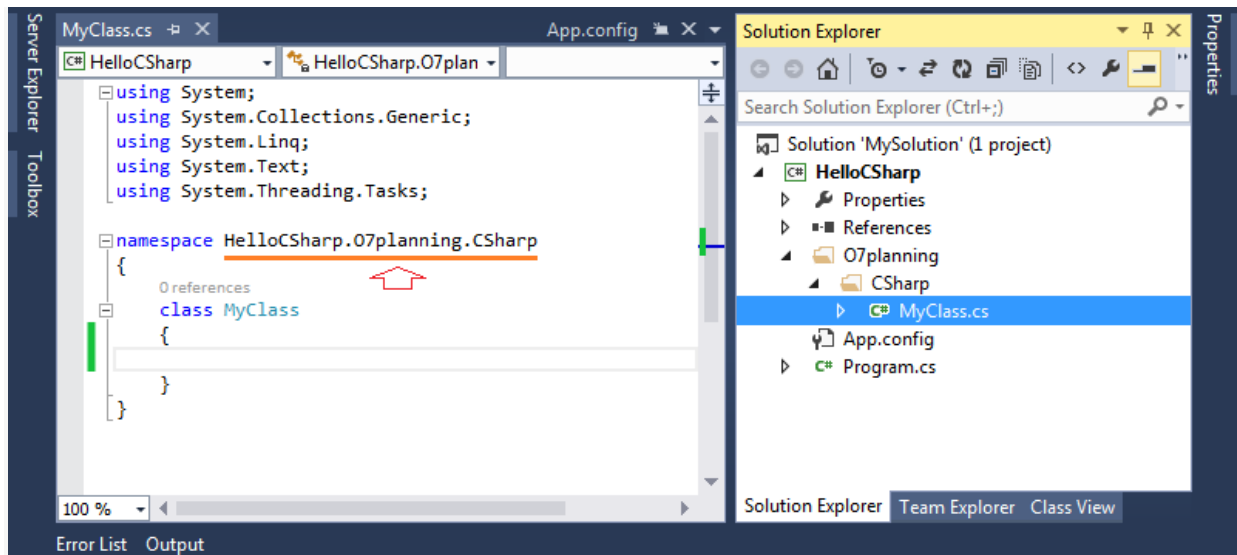
- Add/Class



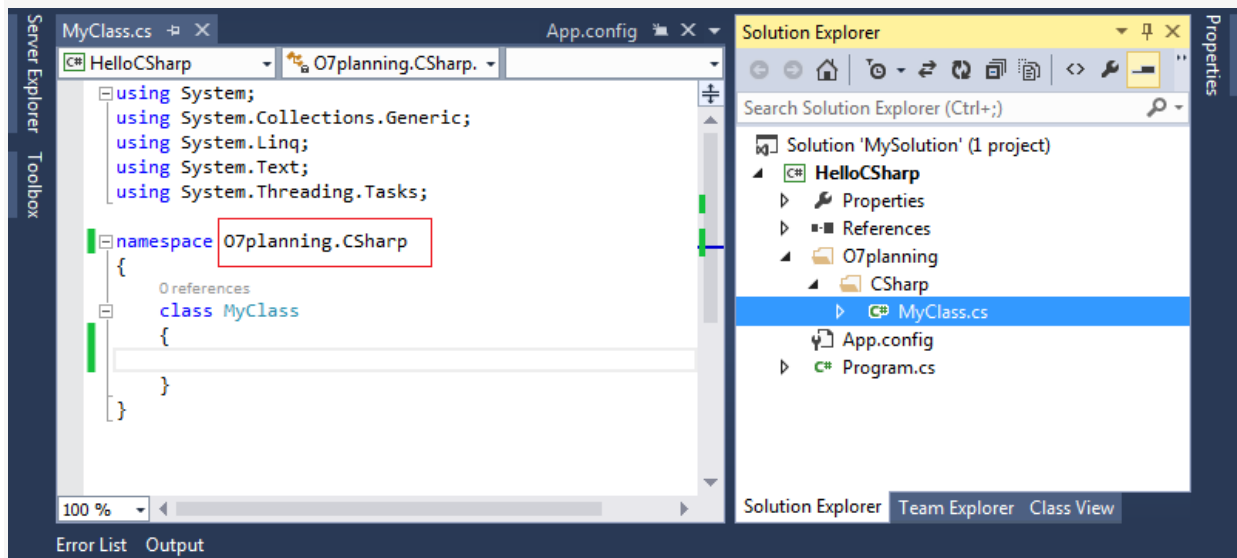
Chọn kiểu item là **Class**, và nhập tên lớp.



Lớp đã được tạo ra, nó nằm trong không gian tên "**HelloCSharp.O7planning.CSharp**". Bạn có thể đổi tên cho **namespace** thành "**O7planning.CSharp**".



Đổi tên namespace thành "O7planning.CSharp".



Bạn có thể thay đổi nội dung của lớp:

MyClass.cs

?

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace O7planning.CSharp
8  {
9      class MyClass
10     {
11         static void Main(string[] args)
12     {

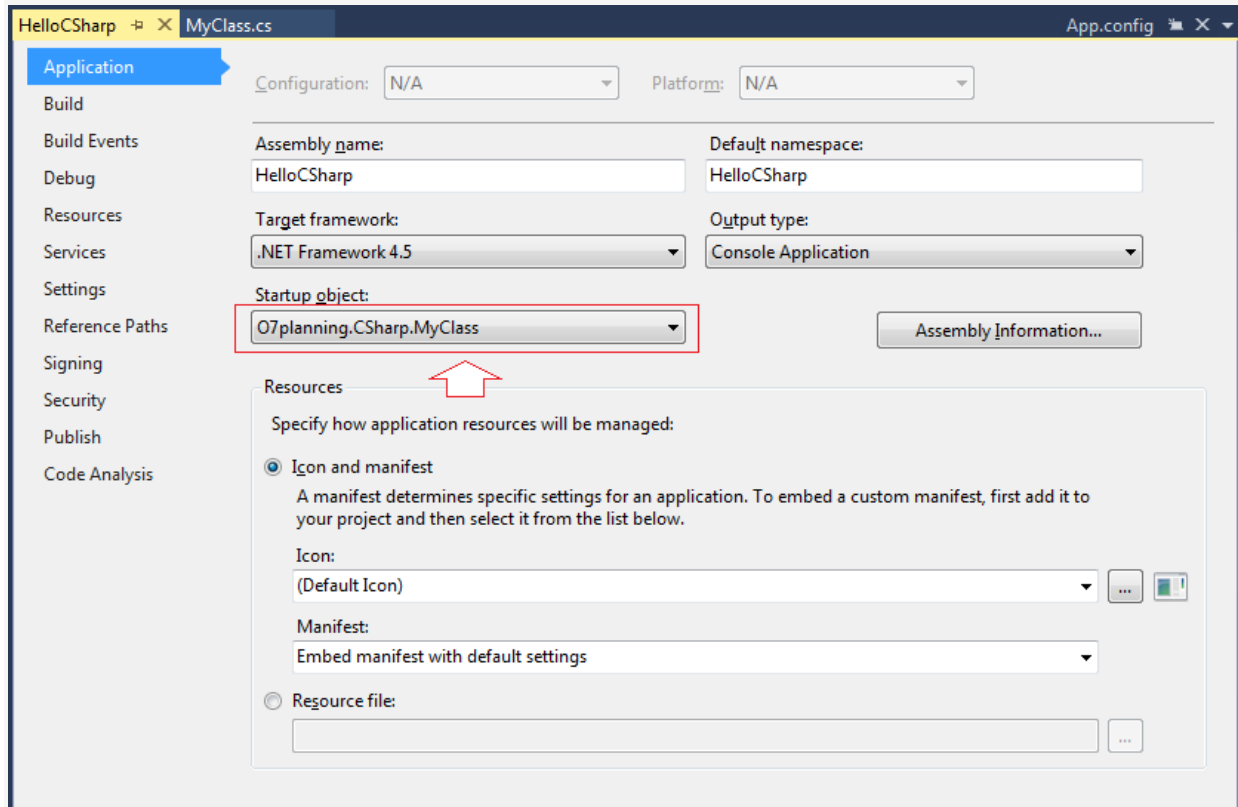
```

```

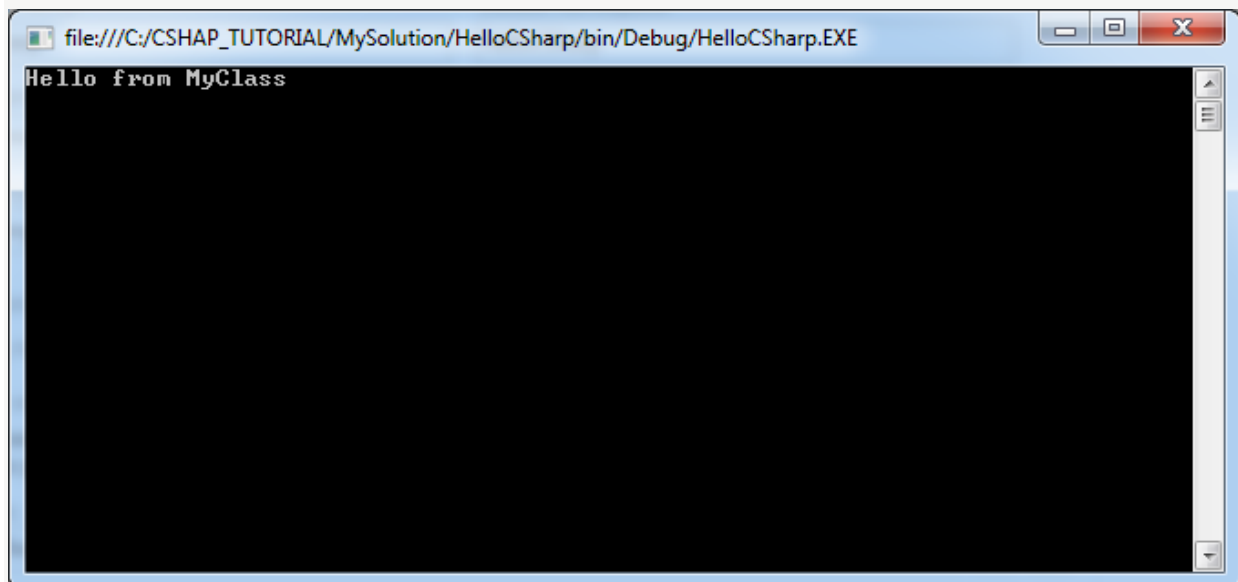
13         Console.WriteLine("Hello from MyClass");
14         Console.ReadLine();
15     }
16 }
17 }

```

Khai báo lớp **MyClass** là điểm bắt đầu để chạy. Nhấn phải chuột vào **Project** chọn **Properties**



Chạy ví dụ:



7- Các kiểu dữ liệu trong C#

Kiểu	Mô tả	Phạm vi	Giá trị mặc định
bool	Giá trị Boolean (Đúng hoặc sai).	True hoặc False	False
byte	Số tự nhiên không dấu 8-bit	0 tới 255	0
char	Ký tự unicode 16-bit	U +0000 tới U +ffff	'\0'
decimal	Có độ chính xác đến 28 con số và giá trị thập phân (Sử dụng 128-bit)	$(-7.9 \times 10^{28}$ tới $7.9 \times 10^{28}) / 100$ tới 28	0.0M
double	Kiểu dấu chấm động có độ chính xác gấp đôi (Sử dụng 64-bit)	$(+/-)5.0 \times 10^{-324}$ tới $(+/-)1.7 \times 10^{308}$	0.0D
float	Kiểu dấu chấm động (Sử dụng 32-bit)	-3.4×10^{38} to $+ 3.4 \times 10^{38}$	0.0F
int	Số nguyên có dấu 32-bit	-2,147,483,648 tới 2,147,483,647	0
long	64-bit signed integer type	-923,372,036,854,775,808 tới 9,223,372,036,854,775,807	0L
sbyte	Số nguyên có dấu 8-bit	-128 tới 127	0
short	Số nguyên có dấu 16-bit	-32,768 tới 32,767	0
uint	Số nguyên không dấu 32-bit	0 tới 4,294,967,295	0
ulong	Số nguyên không dấu 64-bit	0 tới 18,446,744,073,709,551,615	0
ushort	Số nguyên không dấu 16-bit	0 tới 65,535	0

8- Biến và khai báo

Một biến xác định bởi một cái tên cho một nơi lưu trữ dữ liệu mà chương trình của bạn có thể thao tác. Mỗi biến trong C# có một kiểu dữ liệu cụ thể, trong đó xác định kích thước và phạm vi giá trị có thể được lưu trữ trong bộ nhớ, và tập hợp các toán tử có thể áp dụng cho biến.

Biến có thể thay đổi giá trị trong quá trình tồn tại của nó trong chương trình. Các biến có giá trị cố định được gọi là các hằng số. Sử dụng từ khóa **const** để khai báo một biến là hằng số.

Khai báo một biến:

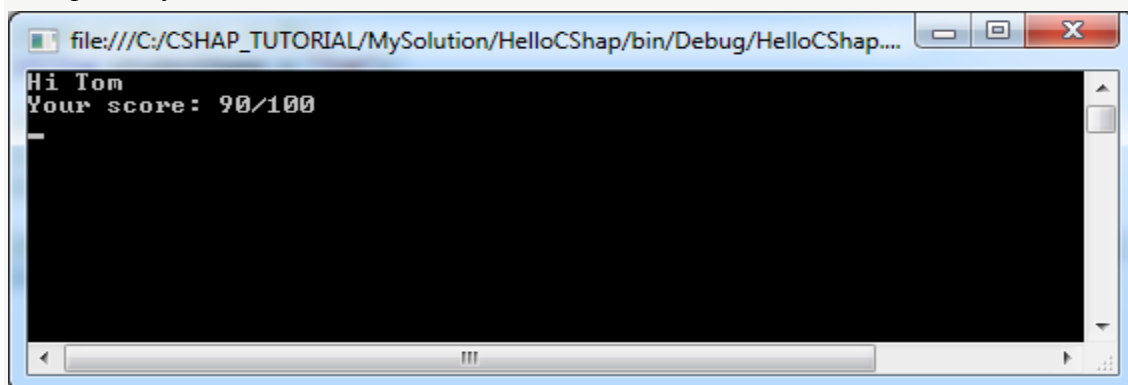
?

```
1 // Khai báo một biến
2 // Data Type: Kiểu dữ liệu.
3 // Variable Name: Tên biến.
4 <Data Type> <Variable name>;
5
6 // Khai báo một biến và gán giá trị cho nó.
7 <Data Type> <Variable name> = <value>;
8
9 // Khai báo một hằng số (constants)
10 const <Data Type> <Variable name> = <value>;
```

VariableExample.cs

```
?  
1 using System;  
2 using System.Collections.Generic;  
3 using System.Linq;  
4 using System.Text;  
5 using System.Threading.Tasks;  
6  
7 namespace HelloCSharp  
8 {  
9     class VariableExample  
10    {  
11        static void Main(string[] args)  
12        {  
13            // Khai báo một hằng số có kiểu int.  
14            // Bạn không thể gán giá trị mới cho hằng số.  
15            const int MAX_SCORE = 100;  
16  
17            // Khai báo một biến có kiểu int.  
18            int score = 0;  
19  
20            // Gán giá trị mới cho biến score.  
21            score = 90;  
22  
23            // Khai báo một chuỗi (string).  
24            string studentName = "Tom";  
25  
26            // In giá trị của biến ra màn hình Console.  
27            Console.WriteLine("Hi {0}", studentName);  
28            Console.WriteLine("Your score: {0}/{1}", score, MAX_SCORE);  
29  
30            // Chờ người dùng nhập vào gì đó và nhấn Enter trước  
31            // khi kết thúc chương trình.  
32            Console.ReadLine();  
33        }  
34    }  
35 }  
36 }
```

Kết quả chạy ví dụ:



9- Câu lệnh rẽ nhánh

9.1- Câu lệnh If-else

if là một câu lệnh kiểm tra một điều kiện gì đó trong **C#**. Chẳng hạn: Nếu **a > b** thì làm gì đó

Các toán tử so sánh thông dụng:

Toán tử	Ý nghĩa	Ví dụ
>	Lớn hơn	5 > 4 là đúng (true)
<	Nhỏ hơn	4 < 5 là đúng (true)
>=	Lớn hơn hoặc bằng	4 >= 4 là đúng (true)
<=	Nhỏ hơn hoặc bằng	3 <= 4 là đúng (true)
==	Bằng nhau	1 == 1 là đúng (true)
!=	Không bằng nhau	1 != 2 là đúng (true)
&&	Và	a > 4 && a < 10
	Hoặc	a == 1 a == 4

?

```
1 // Cú pháp?
2 //
3 // condition: Điều kiện để kiểm tra.
4 if ( condition )
5 {
6     // Làm gì đó ở đây.
7 }
```

Ví dụ:

?

```
1 // Ví dụ 1:
2 if ( 5 < 10 )
3 {
4     Console.WriteLine( "Five is now less than ten");
5 }
6
7 // Ví dụ 2:
8 if ( true )
9 {
10     Console.WriteLine( "Do something here");
11 }
```

Cấu trúc đầy đủ của if - else if - else:

?

```
1 // Chú ý rằng sẽ chỉ có nhiều nhất một khối lệnh được thực thi.
2 // Chương trình kiểm tra các điều kiện từ trên xuống dưới.
3 // Khi bắt gặp một điều kiện đúng, khối lệnh đó sẽ được thực thi.
4 // Các điều kiện còn lại sẽ bị bỏ qua.
5 ...
6
7 // Nếu condition1 là đúng thì ..
8 if (condition1 )
9 {
10     // Làm gì đó nếu condition1 là đúng (true).
11 }
12 // Ngược lại nếu condition2 là đúng thì ...
13 else if(condition2 )
14 {
15     // Làm gì đó tại đây nếu condition2 là đúng
16     // (condition1 là sai).
17 }
18 // Ngược lại nếu conditionN là đúng thì ...
19 else if(conditionN )
20 {
21     // Làm gì đó ở đây nếu conditionN là đúng
22     // (Tất cả các điều kiện ở trên là sai).
23 }
24 // Ngược lại (Khi tất cả các điều kiện ở trên là sai).
25 else {
26     // Làm gì đó.
27 }
```

IfElseExample.cs

?

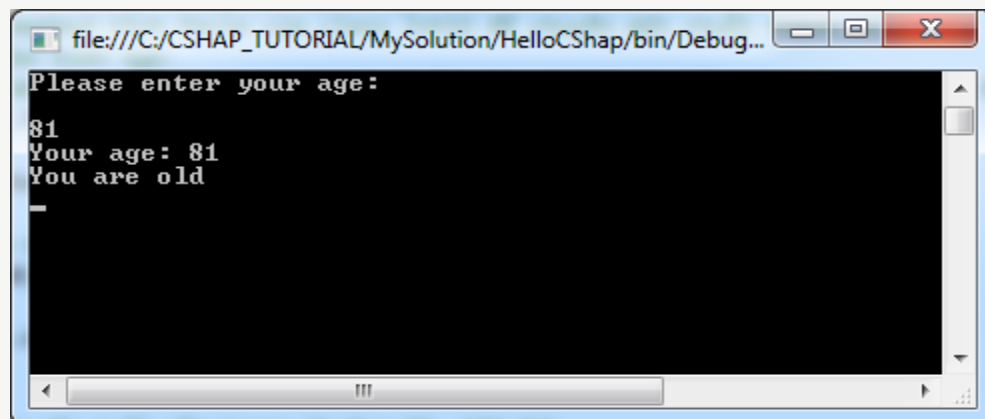
```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace HelloCSharp
8 {
9     class IfElseExample
10     {
11         static void Main(string[] args)
12         {
13             // Khai báo một số kiểu int, mô tả tuổi của bạn.
14             int age;
15
16             Console.WriteLine("Please enter your age: \n");
```

```

17
18 // Khai báo một biến, để lưu trữ đoạn text
19 // người dùng nhập vào từ bàn phím.
20 string inputStr = Console.ReadLine();
21
22 // Int32 là một lớp nằm trong namespace System.
23 // Sử dụng phương thức tĩnh Parse của lớp Int32
24 // để chuyển đổi một chuỗi thành một số
25 // Và gán vào biến age.
26 // (Chú ý: Nếu 'inputStr' không phải là chuỗi số,
27 // có thể gây lỗi chương trình tại đây).
28 age = Int32.Parse(inputStr);
29
30 Console.WriteLine("Your age: {0}", age);
31
32 // Kiểm tra nếu age nhỏ hơn 80 thì ...
33 if (age < 80)
34 {
35     Console.WriteLine("You are pretty young");
36 }
37
38 // Ngược lại nếu tuổi nằm trong khoảng 80, 100 thì
39 else if (age >= 80 && age <= 100)
40 {
41     Console.WriteLine("You are old");
42 }
43 // Ngược lại (Các trường hợp còn lại)
44 else
45 {
46     Console.WriteLine("You are verry old");
47 }
48
49 Console.ReadLine();
50 }
51 }
52 }

```

Chạy ví dụ, và nhập vào 81, và xem kết quả:



9.2- Câu lệnh Switch-Case

Cú pháp câu lệnh rẽ nhánh **switch**:

```
?
1 // Sử dụng từ khóa 'switch' để kiểm tra giá trị của một biến.
2 switch ( <variable> )
3 {
4     case value1:
5         // Làm gì đó tại đây nếu giá trị của biến bằng value1.
6         break;
7     case value2:
8         // Làm gì đó tại đây nếu giá trị của biến bằng value2.
9         break;
10
11     ...
12
13     default:
14         // Các trường hợp còn lại.
15         break;
16 }
```

BreakExample.cs

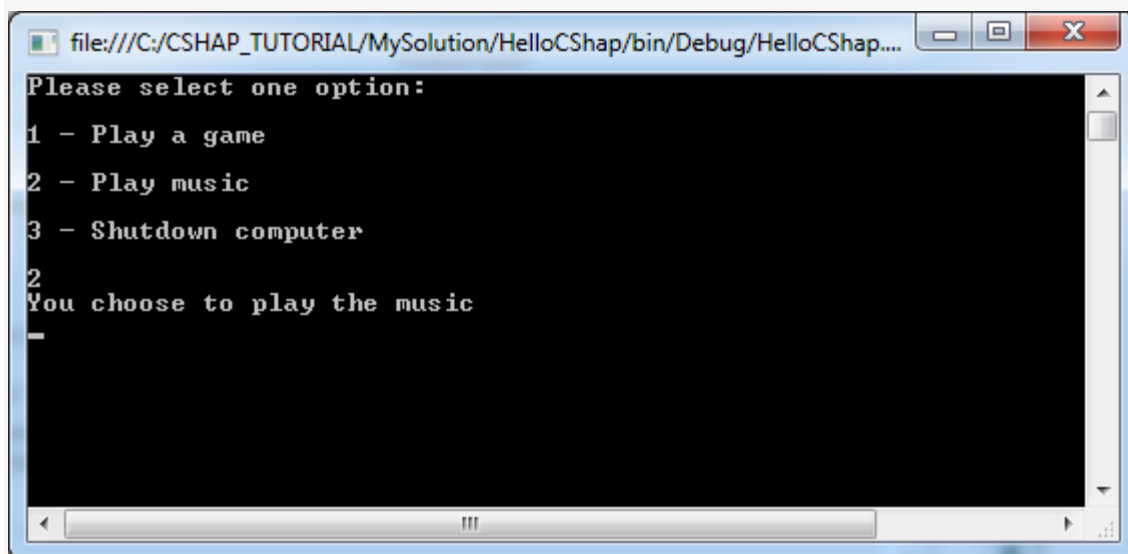
```
?
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace HelloCSharp
8 {
9     class BreakExample
10     {
11         static void Main(string[] args)
12         {
13             // Đề nghị người dùng chọn 1 lựa chọn.
14             Console.WriteLine("Please select one option:\n");
15
16             Console.WriteLine("1 - Play a game \n");
17             Console.WriteLine("2 - Play music \n");
18             Console.WriteLine("3 - Shutdown computer \n");
19
20
21             // Khai báo một biến option
22             int option;
23
24             // Chuỗi người dùng nhập vào từ bàn phím
25             string inputStr = Console.ReadLine();
26
27             // Chuyển chuỗi thành số nguyên.
28             option = Int32.Parse(inputStr);
```

```

29
30         // Kiểm tra giá trị của 'option'.
31         switch (option)
32         {
33
34             case 1:
35                 Console.WriteLine("You choose to play the game");
36                 break;
37             case 2:
38                 Console.WriteLine("You choose to play the music");
39                 break;
40             case 3:
41                 Console.WriteLine("You choose to shutdown the computer");
42                 break;
43             default:
44                 Console.WriteLine("Nothing to do...");
45                 break;
46         }
47
48         Console.ReadLine();
49     }
50 }
51 }

```

Chạy ví dụ và nhập vào 2:



Chú ý:

Lệnh **break** trong trường hợp này nói với chương trình rằng thoát ra khỏi **switch**.

Bạn có thể gộp nhiều trường hợp (**case**) để thực thi cùng một khối lệnh.

BreakExample2.cs

```

?
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;

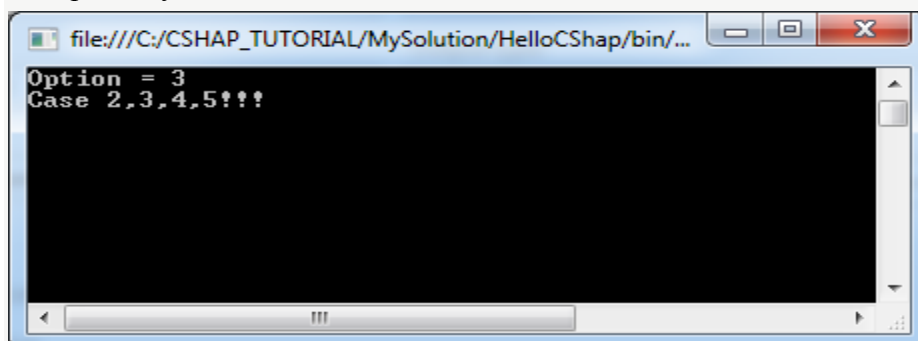
```

```

4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace HelloCSharp
8  {
9      class BreakExample2
10     {
11         static void Main(string[] args)
12         {
13             // Khai báo biến option và gán giá trị 3 cho nó.
14             int option = 3;
15
16             Console.WriteLine("Option = {0}", option);
17
18             // Kiểm tra giá trị của 'option'.
19             switch (option)
20             {
21
22                 case 1:
23                     Console.WriteLine("Case 1");
24                     break;
25
26                 // Trường hợp option = 2,3,4,5 xử lý giống nhau.
27                 case 2:
28                 case 3:
29                 case 4:
30                 case 5:
31                     Console.WriteLine("Case 2,3,4,5!!!");
32                     break;
33                 default:
34                     Console.WriteLine("Nothing to do...");
35                     break;
36             }
37
38             Console.ReadLine();
39         }
40     }
41 }

```

Kết quả chạy ví dụ:



10- Vòng lặp trong C#

Vòng lặp (loop) được sử dụng để chạy lặp lại một khối lệnh. Nó làm chương trình của bạn thực thi lặp đi lặp lại một khối lệnh nhiều lần, đây là một trong các nhiệm vụ cơ bản trong lập trình.

C# hỗ trợ 3 loại vòng lặp khác nhau:

- **FOR**
- **WHILE**
- **DO WHILE**

10.1- Vòng lặp for

Cấu trúc của vòng lặp **for**:

?

```
1 // initialize variable: Khởi tạo một biến.
2 // condition: Điều kiện.
3 // updates new value for variable: Cập nhật giá trị mới cho biến.
4 for (initialize variable; condition; updates new value for variable)
5 {
6     // Thực thi khối lệnh nếu điều kiện là đúng (true).
7 }
```

Ví dụ:

?

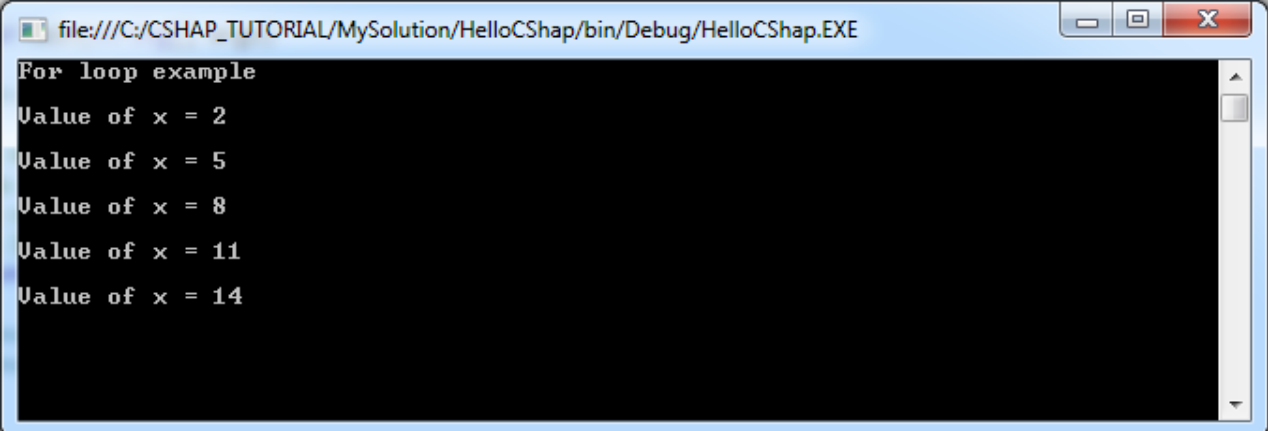
```
1 // Ví dụ 1:
2 // Tạo một biến x và gán giá trị 0 cho nó.
3 // Điều kiện kiểm tra là x < 5
4 // Nếu x < 5 đúng thì khối lệnh được thực thi.
5 // Sau mỗi bước lặp (iteration), giá trị của x được tăng lên 1.
6 for (int x = 0; x < 5 ; x = x + 1)
7 {
8     // Làm gì đó tại đây khi x < 5 là đúng (true).
9 }
10
11
12 // Ví dụ 2:
13 // Tạo một biến x và gán giá trị ban đầu của nó là 2
14 // Điều kiện kiểm tra là x < 15
15 // Nếu x < 15 đúng thì khối lệnh được chạy
16 // Sau mỗi bước lặp (iteration), giá trị của x được tăng lên 3.
17 for (int x = 2; x < 15 ; x = x + 3)
18 {
19     // Làm gì đó tại đây khi x < 15 là đúng (true).
20 }
```

ForLoopExample.cs

?

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace HelloCSharp
8  {
9      class ForLoopExample
10     {
11         static void Main(string[] args)
12         {
13             Console.WriteLine("For loop example");
14
15             // Tạo một biến x và gán giá trị ban đầu của nó là 2
16             // Điều kiện kiểm tra là x < 15
17             // Nếu x < 15 đúng thì khối lệnh được chạy
18             // Mỗi bước lặp (iteration), giá trị x lại được
19             // cập nhập mới, tăng giá trị lên 3.
20             for (int x = 2; x < 15; x = x + 3)
21             {
22                 Console.WriteLine( );
23                 Console.WriteLine("Value of x = {0}", x);
24             }
25
26             Console.ReadLine();
27         }
28     }
29 }
```

Kết quả chạy ví dụ:



The screenshot shows a Windows command prompt window titled "file:///C:/CSHAP_TUTORIAL/MySolution/HelloCSharp/bin/Debug/HelloCSharp.EXE". The output of the program is displayed as follows:

```
For loop example
Value of x = 2
Value of x = 5
Value of x = 8
Value of x = 11
Value of x = 14
```

10.2- Vòng lặp while

Cú pháp của vòng lặp **while**:

?

```
1 // condition: Điều kiện
2 while (condition)
3 {
4     // Trong khi 'condition' là đúng, thì thực thi khối lệnh.
5 }
```

Ví dụ:

?

```
1 // Khai báo một biến x.
2 int x = 2;
3
4 while ( x < 10)
5 {
6     // Làm gì đó tại đây khi x < 10 còn đúng.
7     ...
8
9     // Cập nhập giá trị mới cho biến x.
10    x = x + 3;
11 }
```

WhileLoopExample.cs

?

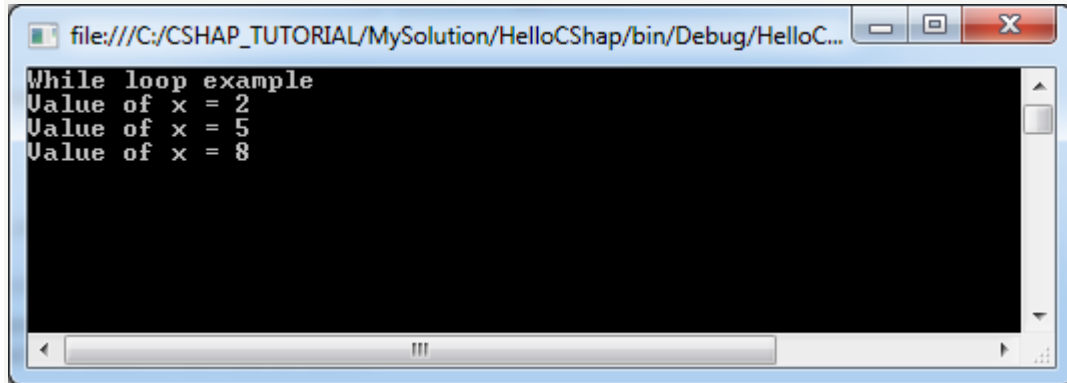
```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace HelloCSharp
8 {
9     class WhileLoopExample
10    {
11        static void Main(string[] args)
12        {
13
14            Console.WriteLine("While loop example");
15
16            // Tạo một biến x và gán giá trị 2 cho nó.
17            int x = 2;
18
19            // Điều kiện là x < 10.
20            // Nếu x < 10 đúng thì khối lệnh được thực thi.
21            while (x < 10)
22            {
```

```

23             Console.WriteLine("Value of x = {0}", x);
24
25             x = x + 3;
26         }
27
28         Console.ReadLine();
29     }
30 }
31 }

```

Kết quả chạy ví dụ:



10.3- Vòng lặp do-while

Cú pháp của vòng lặp **do-while**

?

```

1 // Đặc điểm của vòng lặp 'do-while' là nó sẽ thực khi khối lệnh ít nhất
2 // Sau mỗi bước lặp (iteration), nó sẽ kiểm tra lại điều kiện,
3 // Nếu điều kiện đúng, khối lệnh sẽ được thực thi tiếp.
4 do {
5     // Làm gì đó tại đây.
6 } while (condition);
7 // Chú ý: Cần có dấu chấm phẩy (;) tại đây.

```

DoWhileLoopExample.cs

?

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace HelloCSharp
8 {
9     class DoWhileLoopExample
10    {
11        static void Main(string[] args)

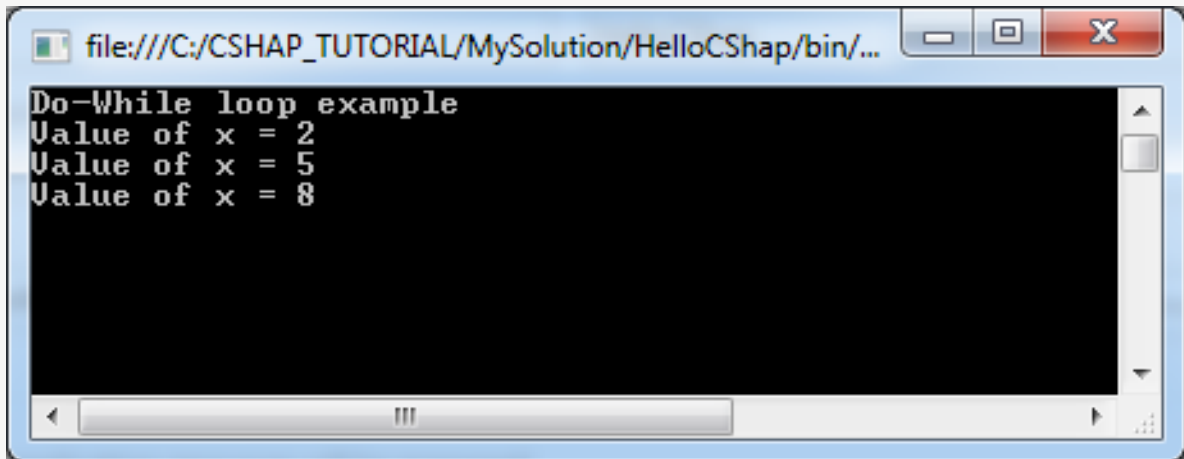
```

```

12         {
13
14             Console.WriteLine("Do-While loop example");
15
16             // Tạo một biến x và gán giá trị 2 cho nó.
17             int x = 2;
18
19             // Thực hiện khối lệnh ít nhất 1 lần.
20             // Sau mỗi bước lặp (iteration), nó sẽ kiểm tra
21             // lại điều kiện,
22             // Nếu điều kiện đúng, khối lệnh sẽ được thực
23             // thi tiếp.
24             do
25             {
26                 Console.WriteLine("Value of x = {0}", x);
27
28                 x = x + 3;
29
30             } while (x < 10);
31             // Chú ý: Cần có dấu chấm phẩy tại đây.
32             Console.ReadLine();
33         }
34     }
35 }

```

Kết quả chạy ví dụ:



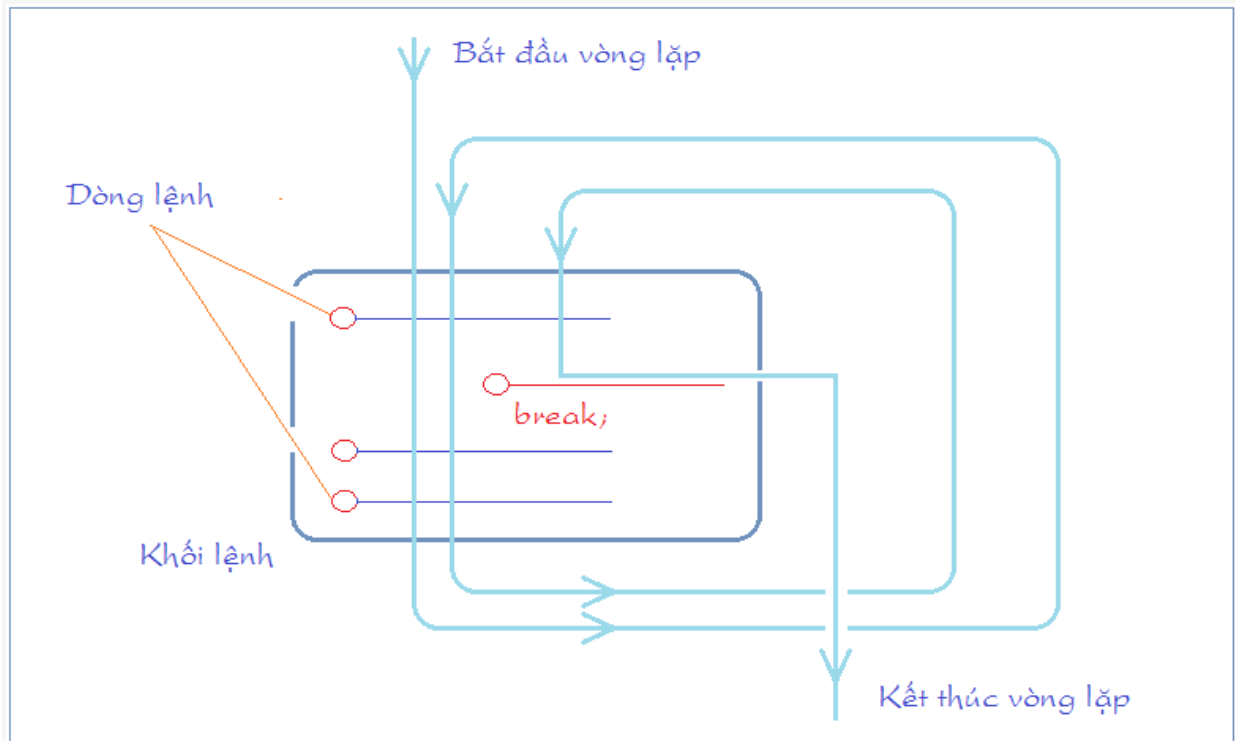
```

file:///C:/CSHAP_TUTORIAL/MySolution/HelloCSharp/bin/...
Do-While loop example
Value of x = 2
Value of x = 5
Value of x = 8

```

10.4- Lệnh break trong vòng lặp

break là một lệnh nó có thể nằm trong một khối lệnh của một vòng lặp. Đây là lệnh kết thúc vòng lặp vô điều kiện.



LoopBreakExample.cs

```
?
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace HelloCSharp
8  {
9      class LoopBreakExample
10     {
11         static void Main(string[] args)
12         {
13             Console.WriteLine("Break example");
14
15             // Tạo một biến x và gán giá trị 2 cho nó.
16             int x = 2;
17
18             while (x < 15)
19             {
20
```

```

21         Console.WriteLine("-----\n");
22         Console.WriteLine("x = {0}", x);
23
24         // Kiểm tra nếu x = 5 thì thoát ra khỏi
25         // vòng lặp.
26         if (x == 5)
27         {
28             break;
29         }
30         // Tăng giá trị của x lên 1
31         // (Viết ngắn gọn cho x = x + 1;).
32         x++;
33         Console.WriteLine("x after ++ = {0}", x);
34
35     }
36
37     Console.ReadLine();
38 }
39 }
40 }

```

Kết quả chạy ví dụ:

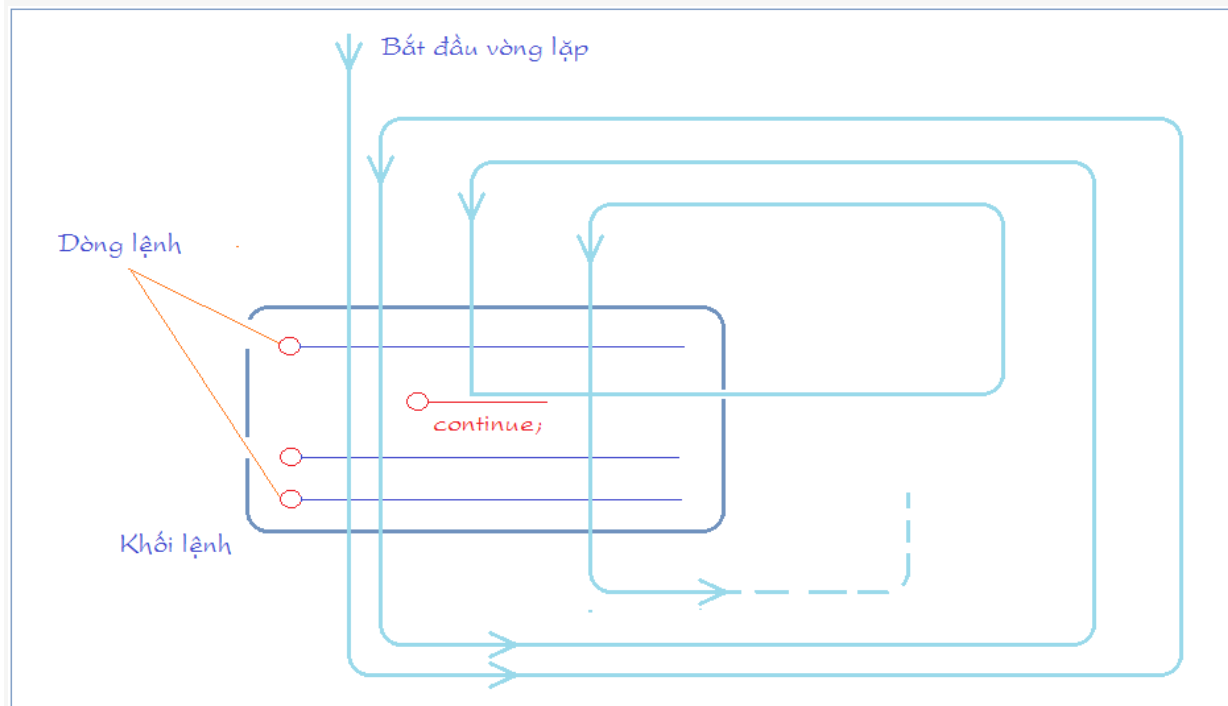
```

file:///C:/CSHAP_TUTORIAL/MySolution/HelloCShap/bin/Debug/HelloCShap.EXE
Break example
-----
x = 2
x after ++ = 3
-----
x = 3
x after ++ = 4
-----
x = 4
x after ++ = 5
-----
x = 5
_

```

10.5- Lệnh continue trong vòng lặp

continue là một lệnh, nó có thể nằm trong một vòng lặp, khi bắt gặp lệnh **continue** chương trình sẽ bỏ qua các dòng lệnh trong khối phía dưới của **continue** và bắt đầu một vòng lặp mới.



LoopContinueExample.cs

```
?
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace HelloCSharp
8  {
9      class LoopContinueExample
10     {
11         static void Main(string[] args)
12         {
13
14             Console.WriteLine("Continue example");
15
16             // Tạo một biến x và gán giá 2 cho nó.
17             int x = 2;
18
19             while (x < 7)
20             {
21
```



```

22         Console.WriteLine("-----\n");
23         Console.WriteLine("x = {0}", x);
24
25         // Toán tử % là phép chia lấy số dư.
26         // Nếu x chẵn, thì bỏ qua các dòng lệnh phía
27         // dưới của 'continue', và tiếp tục bước lặp
28         // (iteration) mới (nếu điều kiện vẫn đúng).
29         if (x % 2 == 0)
30         {
31             // Tăng giá trị của x lên 1 (x = x + 1;).
32             x++;
33             continue;
34         }
35         else
36         {
37             // Tăng giá trị của x lên 1 (x = x + 1;).
38             x++;
39         }
40         Console.WriteLine("x after ++ = {0}", x);
41
42     }
43
44     Console.ReadLine();
45 }
46 }
47 }

```

Kết quả chạy ví dụ:

```

file:///C:/CSHAP_TUTORIAL/MySolution/HelloCShap/bin/Debug/HelloCShap.EXE
Continue example
-----
x = 2
-----
x = 3
x after ++ = 4
-----
x = 4
-----
x = 5
x after ++ = 6
-----
x = 6
-

```

11- Mảng trong C#

11.1- Mảng một chiều

Đây là hình minh họa về mảng một chiều có 5 phần tử, các phần tử được đánh chỉ số từ 0 tới 4.

```
int[] a;
```



Cú pháp khai báo mảng một chiều:

?

```
1 // Cách 1:
2 // Khai báo một mảng các số int, chỉ định giá trị cho các phần tử.
3 int[] years = { 2001, 2003, 2005, 1980, 2003 };
4
5 // Cách 2:
6 // Khai báo một mảng các số float, chỉ rõ số phần tử.
7 // (3 phần tử).
8 float[] salaries = new float[3];
```

ArrayExample1.cs

?

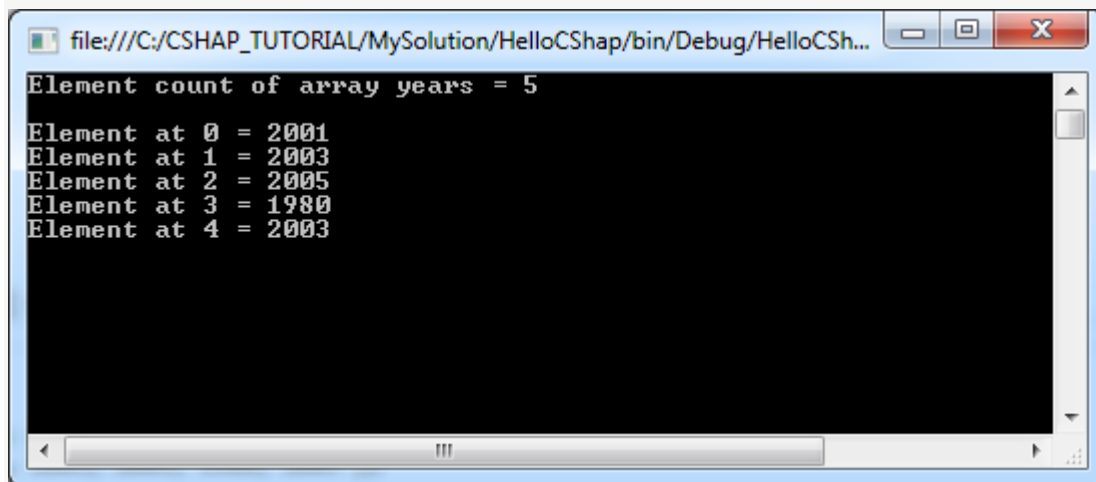
```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace HelloCSharp
8 {
9     class ArrayExample1
10     {
11         static void Main(string[] args)
12         {
13             // Cách 1:
14             // Khai báo một mảng, chỉ định giá trị cho các phần tử.
15             int[] years = { 2001, 2003, 2005, 1980, 2003 };
16
17             // Length là một Property của mảng, nó trả về số phần tử
18             // của mảng.
19             Console.WriteLine("Length of years = {0}\n", years.Length);
20
21             // Sử dụng vòng lặp for để in ra các phần tử của mảng.
22             for (int i = 0; i < years.Length; i++) {
```

```

23         Console.WriteLine("Element at {0} = {1}", i, years[i]);
24     }
25
26     // Cách 2:
27     // Khai báo một mảng có 3 phần tử.
28     float[] salaries = new float[3];
29
30     // Gán các giá trị cho các phần tử.
31     salaries[0] = 1000;
32     salaries[1] = 1200;
33     salaries[2] = 1100;
34
35     Console.ReadLine();
36 }
37 }

```

Kết quả chạy ví dụ:



11.2- Mảng hai chiều

Đây là hình minh họa một mảng 2 chiều

		Column				
		0	1	2	3	4
Row	0	a[0,0]	a[0,1]	a[0,2]	a[0,3]	a[0,4]
	1	a[1,0]	a[1,1]	a[1,2]	a[1,3]	a[1,4]
	2	a[2,0]	a[2,1]	a[2,2]	a[2,3]	a[2,4]

Cú pháp khai báo một mảng 2 chiều:

?

```
1 // Khai báo một mảng 2 chiều, 3 dòng & 5 cột.
2 // Chỉ định giá trị cho các phần tử.
3 int[,] a = new int[,] {
4     {1, 2, 3, 4, 5},
5     {0, 3, 4, 5, 7},
6     {0, 3, 4, 0, 0}
7 };
8
9 // Khai báo một mảng 2 chiều, 3 dòng & 5 cột.
10 // Không chỉ định giá trị các phần tử.
11 int[,] a = new int[3,5];
```

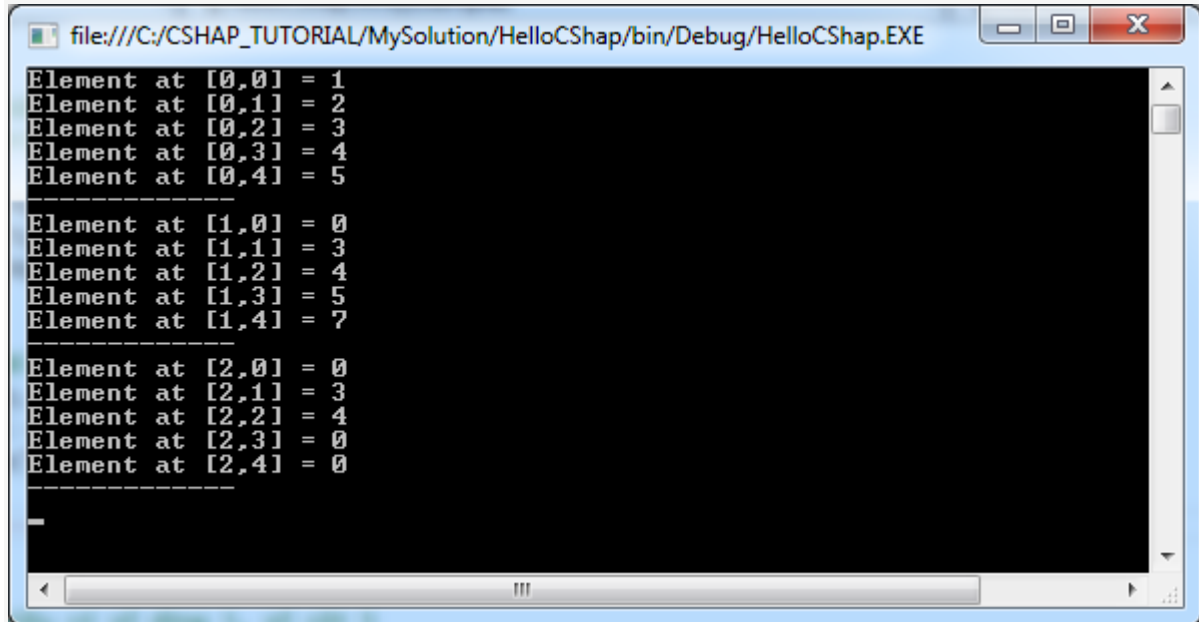
ArrayExample2.cs

?

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace HelloCSharp
8 {
9     class ArrayExample2
10     {
11         static void Main(string[] args)
12         {
13             // Khai báo một mảng 2 chiều, 3 dòng 5 cột.
14             // Chỉ định các giá trị cho các phần tử.
15             int[,] a = {
16                 { 1, 2, 3, 4, 5 },
17                 { 0, 3, 4, 5, 7 },
18                 { 0, 3, 4, 0, 0 }
19             };
20
21             // Sử dụng vòng lặp for để in ra các phần tử của mảng.
22             for (int row = 0; row < 3; row++) {
23                 for (int col = 0; col < 5; col++) {
24                     Console.WriteLine("Element at [{0},{1}] = {2}",
25                         row, col, a[row,col]);
26                 }
27                 Console.WriteLine("-----");
28             }
29
30             // Khai báo một mảng 2 chiều có số dòng 3, số cột 5
31             // Các phần tử chưa được gán giá trị.
32             int[,] b = new int[3, 5];
33
34             Console.ReadLine();
35         }
36     }
37 }
```

```
36     }
37 }
```

Kết quả chạy ví dụ:



```
file:///C:/CSHAP_TUTORIAL/MySolution/HelloCSharp/bin/Debug/HelloCSharp.EXE
Element at [0,0] = 1
Element at [0,1] = 2
Element at [0,2] = 3
Element at [0,3] = 4
Element at [0,4] = 5
-----
Element at [1,0] = 0
Element at [1,1] = 3
Element at [1,2] = 4
Element at [1,3] = 5
Element at [1,4] = 7
-----
Element at [2,0] = 0
Element at [2,1] = 3
Element at [2,2] = 4
Element at [2,3] = 0
Element at [2,4] = 0
-----
```

11.3- Mảng của mảng

ArrayOfArrayExample.cs

```
?
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace HelloCSharp
8  {
9      class ArrayOfArrayExample
10     {
11         static void Main(string[] args)
12         {
13             // Khai báo một mảng 3 phần tử.
14             // Mỗi phần tử là một mảng khác.
15             string[][] teams = new string[3][];
16
17             string[] mu = { "Beckham", "Giggs" };
18             string[] asenal = { "Oezil", "Szczęsny", "Walcott" };
19             string[] chelsea = { "Oscar", "Hazard", "Drogba" };
20
21             teams[0] = mu;
22             teams[1] = asenal;
23             teams[2] = chelsea;
```

```

24
25         // Sử dụng vòng lặp for để in ra các phần tử của mảng.
26         for (int row = 0; row < teams.Length; row++)
27         {
28             for (int col = 0; col < teams[row].Length ; col++)
29             {
30                 Console.WriteLine("Element at [{0}],[{1}] = {2}",
31                     row, col, teams[row][col]);
32             }
33             Console.WriteLine("-----");
34         }
35
36
37         Console.ReadLine();
38     }
39 }
40 }

```

Kết quả chạy ví dụ

```

file:///C:/CSHAP_TUTORIAL/MySolution/HelloCShap/bin/Debug/HelloCShap.EXE
Element at [0],[0] = Beckham
Element at [0],[1] = Giggs
-----
Element at [1],[0] = Oezil
Element at [1],[1] = Szczesny
Element at [1],[2] = Walcott
-----
Element at [2],[0] = Oscar
Element at [2],[1] = Hazard
Element at [2],[2] = Drogba
-----

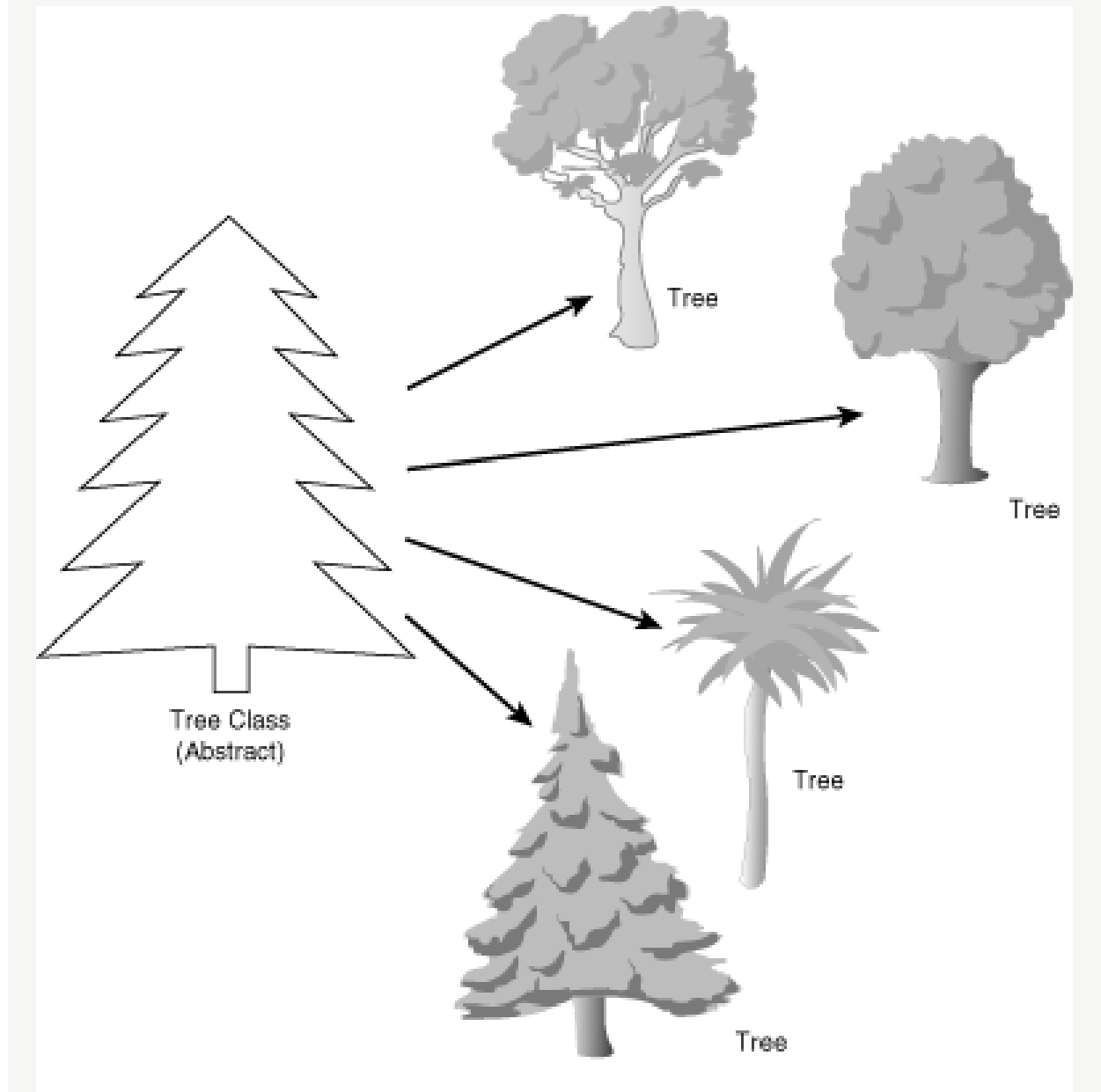
```

12- Class, Constructor và đối tượng

Bạn cần có sự phân biệt giữa 3 khái niệm:

- Class
- Cấu tử (Constructor)
- Đối tượng (Instance)

Khi chúng ta nói về Cây, nó là một thứ gì đó trừu tượng, nó là một lớp (class). Nhưng khi chúng ta chỉ thẳng vào một cái cây cụ thể thì lúc đó đã rõ ràng và đó là đối tượng (instance).



Hoặc khi chúng ta nói về người (Person) thì đó cũng trừu tượng, nó là một lớp. Nhưng khi chỉ thẳng vào bạn hoặc tôi thì đó là 2 đối tượng khác nhau, cùng thuộc lớp người.



Thomas Edison



Bill Gates

Person (Class)

Person.cs

?

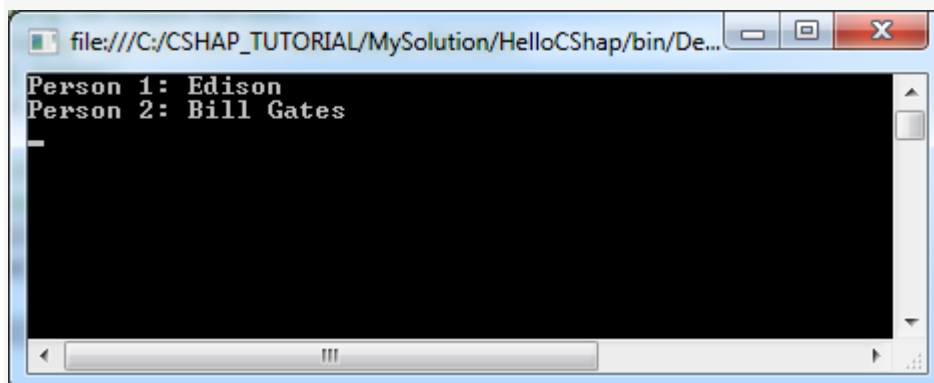
```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace HelloCSharp
8  {
9      class Person
10     {
11         // Đây là một trường (field) .
12         // Lưu trữ tên người.
13         public string Name;
14
15         // Đây là một Constructor.
16         // Nó dùng để tạo ra một đối tượng.
17         // Constructor này này có một tham số.
18         // Constructr luôn có tên giống tên của lớp.
19         public Person(string persionName)
20         {
21             // Gán giá trị của tham số cho trường name.
22             this.Name = persionName;
23         }
24
25         // Đây là một phương thức trả về kiểu string.
26         public string GetName()
27         {
28             return this.Name;
29         }
30     }
31 }
```


Như trên class **Person** không có phương thức **Main**. Tiếp theo class **PersonTest** là ví dụ khởi tạo các đối tượng của **Person** thông qua các câu từ.

PersonTest.cs

```
?
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7
8  namespace HelloCSharp
9  {
10     class PersonTest
11     {
12         static void Main(string[] args)
13         {
14             // Tạo một đối tượng của lớp Person.
15             // Khởi tạo đối tượng này từ Constructor của lớp Person.
16             Person edison = new Person("Edison");
17
18             // Lớp Person có phương thức GetName().
19             // Sử dụng đối tượng để gọi phương thức GetName():
20             String name = edison.GetName();
21             Console.WriteLine("Person 1: " + name);
22
23             // Tạo một đối tượng từ của lớp Person.
24             // Khởi tạo đối tượng này từ Constructor của lớp Person.
25             Person billGate = new Person("Bill Gates");
26
27             // Lớp Person có trường 'name' là công khai (public).
28             // Bạn có thể sử dụng đối tượng để truy cập trường 'name'.
29             String name2 = billGate.Name;
30             Console.WriteLine("Person 2: " + name2);
31
32             Console.ReadLine();
33         }
34     }
35 }
```

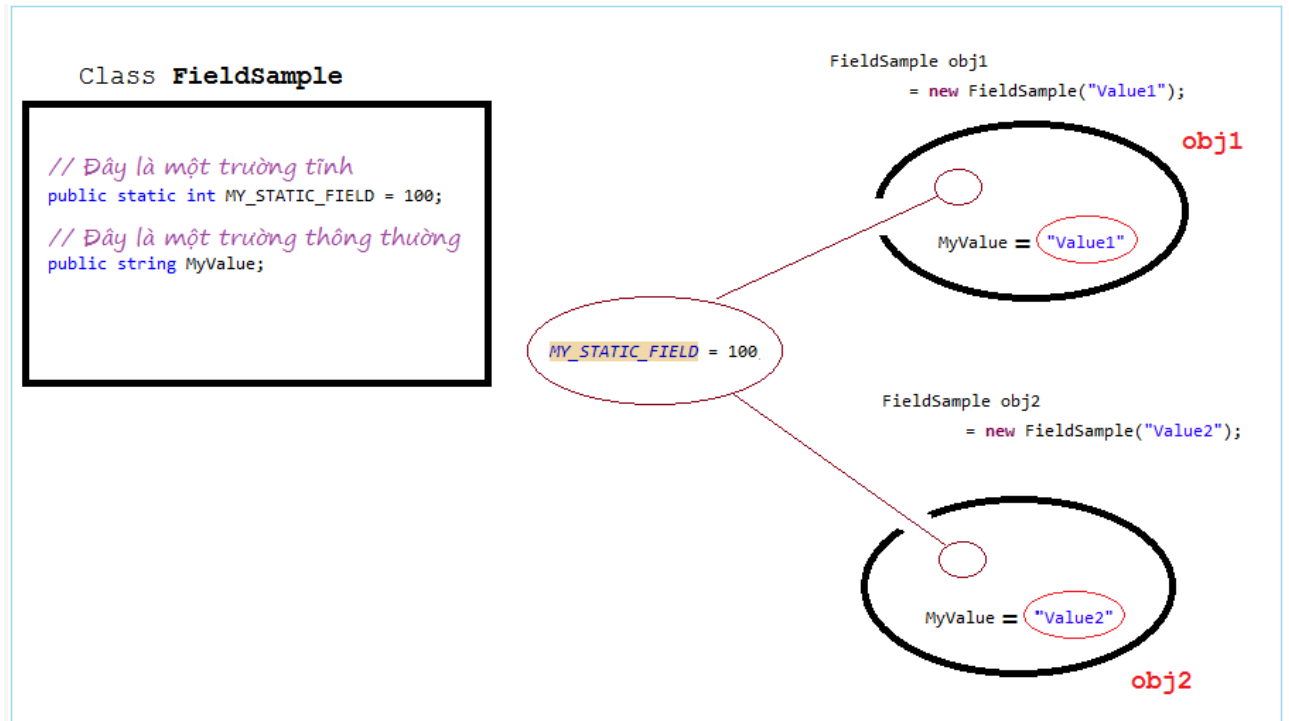
Kết quả chạy ví dụ:



13- Trường (Field)

Trong phần tiếp theo này chúng ta sẽ thảo luận về một số khái niệm:

- Trường (Field)
 - Trường thông thường
 - Trường tĩnh (static Field)
 - Trường const (const Field)
 - Trường tĩnh và readonly (static readonly Field)



FieldSample.cs

```
?  
1 using System;  
2 using System.Collections.Generic;  
3 using System.Linq;  
4 using System.Text;  
5 using System.Threading.Tasks;  
6  
7 namespace HelloCSharp  
8 {  
9     class FieldSample  
10    {  
11        // Đây là một trường tĩnh (static field).  
12        public static int MY_STATIC_FIELD = 100;  
13  
14        // Đây là một trường thông thường.  
15        public string MyValue;
```

```

16
17         // Đây là một Constructor của lớp FieldSample.
18         public FieldSample(string value)
19         {
20             this.MyValue = value;
21         }
22     }
23 }

```

FieldSampleTest.cs

?

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace HelloCSharp
8  {
9      class FieldSampleTest
10     {
11         static void Main(string[] args)
12         {
13             // In ra giá trị của trường tĩnh (static field).
14             // Với các trường tĩnh, bạn phải truy cập nó
15             // thông qua tên lớp.
16             Console.WriteLine("FieldSample.MY_STATIC_FIELD= {0}",
17                               FieldSample.MY_STATIC_FIELD);
18
19             // Bạn có thể thay đổi giá trị của trường tĩnh.
20             FieldSample.MY_STATIC_FIELD = 200;
21
22             Console.WriteLine(" ----- ");
23
24             // Tạo đối tượng thứ nhất.
25             FieldSample obj1 = new FieldSample("Value1");
26
27             // Các trường không tĩnh bạn phải truy cập thông
28             // qua đối tượng.
29             Console.WriteLine("obj1.MyValue= {0}", obj1.MyValue);
30
31             // Tạo đối tượng thứ 2:
32             FieldSample obj2 = new FieldSample("Value2");
33
34
35             Console.WriteLine("obj2.MyValue= {0}", obj2.MyValue);
36
37             // Bạn có thể thay đổi giá trị của trường.
38             obj2.MyValue = "Value2-2";

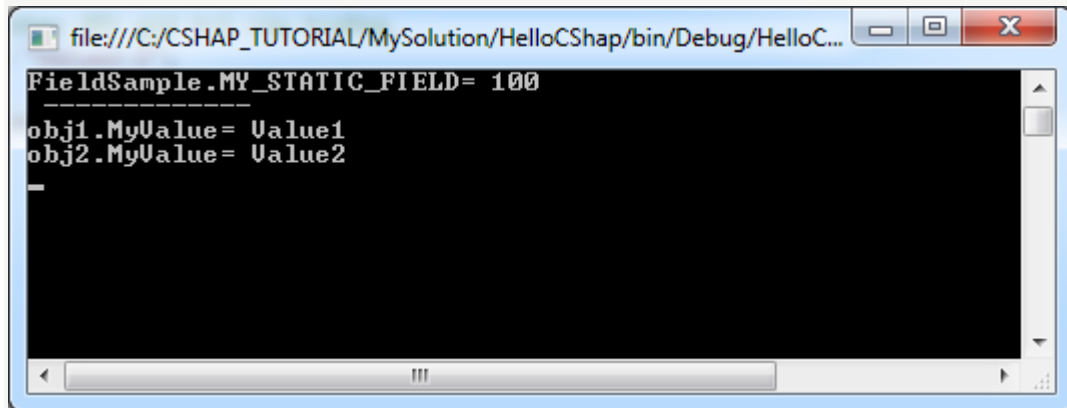
```

```

39
40         Console.ReadLine();
41     }
42
43 }
44 }

```

Kết quả chạy ví dụ:



Ví dụ readonly & static readonly.

ConstFieldExample.cs

```

?
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace HelloCSharp
8  {
9      class ConstFieldExample
10     {
11         // Một trường hằng số, giá trị của nó được xác định
12         // tại thời điểm biên dịch.
13         // Bạn không thể gán giá trị mới cho các trường const.
14         // Chú ý: Trường const luôn luôn là static (Tĩnh).
15         public const int MY_VALUE = 100;
16
17         // Một trường tĩnh (static field) và readonly.
18         // Giá trị của nó có thể gán sẵn, hoặc chỉ được
19         // gán 1 lần trong Constructor.
20         public static readonly DateTime INIT_DATE_TIME1 = DateTime.Now;
21
22         // Một trường readonly và không tĩnh (none-static)
23         // Giá trị của nó có thể gán sẵn, hoặc chỉ được
24         // gán một lần trong Constructor.
25         public readonly DateTime INIT_DATE_TIME2 ;
26

```

```

27         public ConstFieldExample()
28         {
29             // Giá trị của nó được gán 1 lần, tại lần chạy đầu tiên.
30             INIT_DATE_TIME2 = DateTime.Now;
31         }
32     }
33 }

```

14- Phương thức (Method)

Phương thức (Method)

- Phương thức thông thường.
- Phương thức tĩnh
- Phương thức sealed. (Sẽ được đề cập trong phần thừa kế của class).

MethodSample.cs

?

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace HelloCSharp
8  {
9      class MethodSample
10     {
11         public string text = "Some text";
12
13         // Một Constructor mặc định.
14         // Nghĩa là Constructor không có tham số.
15         public MethodSample()
16         {
17
18         }
19
20         // Đây là một phương thức trả về kiểu String.
21         // Phương thức này không có tham số.
22         public string GetText()
23         {
24             return this.text;
25         }
26
27         // Đây là một phương thức có 1 tham số String.
28         // Phương thức này trả về void (Hay gọi là ko trả về gì)
29         public void SetText(string text)
30         {

```

```

31         // this.text: tham chiếu tới trường text.
32         // Để phân biệt với tham số text.
33         this.text = text;
34     }
35
36     // Đây là một phương thức tĩnh.
37     // Trả về kiểu int, và có 3 tham số.
38     public static int Sum(int a, int b, int c)
39     {
40         int d = a + b + c;
41         return d;
42     }
43 }
44 }

```

MethodSampleTest.cs

?

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace HelloCSharp
8  {
9      class MethodSampleTest
10     {
11         static void Main(string[] args)
12         {
13             // Tạo đối tượng MethodSample
14             MethodSample obj = new MethodSample();
15
16             // Gọi phương thức GetText().
17             // Các phương thức không tĩnh cần phải được gọi
18             // thông qua đối tượng.
19             String text = obj.GetText();
20
21             Console.WriteLine("Text = " + text);
22
23             // Gọi phương thức SetText(string)
24             // Các phương thức không tĩnh cần phải được gọi
25             // thông qua đối tượng.
26             obj.SetText("New Text");
27
28             Console.WriteLine("Text = " + obj.GetText());
29
30             // Các phương thức tĩnh cần phải được gọi
31             // thông qua tên lớp.
32             int sum = MethodSample.Sum(10, 20, 30);
33

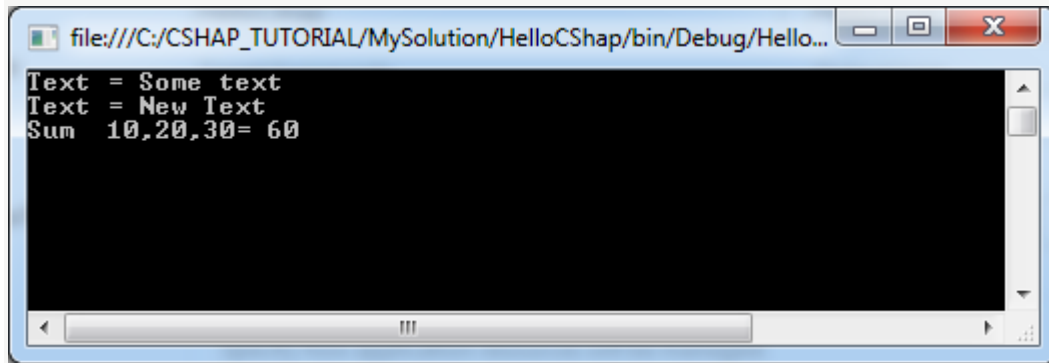
```

```

34         Console.WriteLine("Sum 10,20,30= " + sum);
35
36         Console.ReadLine();
37     }
38 }
39 }

```

Kết quả chạy ví dụ:



15- Thừa kế trong C#

CSharp cho phép viết class mở rộng từ một class khác. Class mở rộng từ một class khác được gọi là class con. Class con có được thừa kế các trường, thuộc tính và các method từ class cha.

Hãy xem một ví dụ minh họa về thừa kế trong **CSharp**:

Animal.cs

```

?
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace HelloCSharp
8  {
9      // Mô phỏng một lớp động vật.
10     class Animal
11     {
12         public Animal()
13         {
14
15         }
16
17         public void Move()
18         {
19             Console.WriteLine("Move ...!");
20         }
21     }
22 }

```

23 }

Cat.cs

?

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace HelloCSharp
8 {
9     class Cat : Animal
10    {
11
12        public void Say()
13        {
14            Console.WriteLine("Meo");
15        }
16
17        // Một phương thức của lớp Cat.
18        public void Catch()
19        {
20            Console.WriteLine("Catch Mouse");
21        }
22    }
23 }
```

Ant.cs

?

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace HelloCSharp
8 {
9     // Con kiến
10    class Ant : Animal
11    {
12    }
13 }
```

AnimalTest.cs

?

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
```

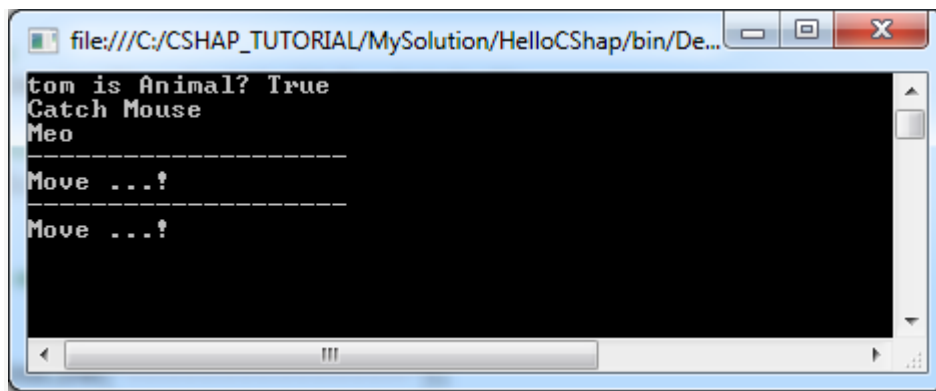


```

5  using System.Threading.Tasks;
6
7  namespace HelloCSharp
8  {
9      class AnimalTest
10     {
11
12         static void Main(string[] args)
13         {
14             // Khai báo một đối tượng Cat.
15             Cat tom = new Cat();
16
17             // Kiểm tra xem 'tom' có phải là đối tượng Animal
18             // hay không.
19             // Kết quả rõ ràng là true.
20             bool isAnimal = tom is Animal;
21
22             // ==> true
23             Console.WriteLine("tom is Animal? " + isAnimal);
24
25             // Gọi phương thức Catch
26             tom.Catch();
27
28             // Gọi vào phương thức Say() của Cat.
29             // ==> Meo
30             tom.Say();
31
32             Console.WriteLine("-----");
33
34             // Khai báo một đối tượng Animal
35             // Khởi tạo đối tượng thông qua cấu tử của Cat.
36             Animal tom2 = new Cat();
37
38             // Gọi phương thức Move()
39             tom2.Move();
40
41             Console.WriteLine("-----");
42
43             // Tạo một đối tượng Ant.
44             Ant ant = new Ant();
45
46             // Gọi phương thức Move() thừa kế được từ Animal.
47             ant.Move();
48
49             Console.ReadLine();
50         }
51     }
52 }

```

Kết quả chạy class **AnimalTest**:



```
file:///C:/CSHAP_TUTORIAL/MySolution/HelloCShap/bin/De...
tom is Animal? True
Catch Mouse
Meo
-----
Move ...?
-----
Move ...?
```

16- Thừa kế và đa hình trong C#

- 1- Giới thiệu
- 2- Lớp, đối tượng và Constructor
- 3- Thừa kế trong CSharp
- 4- Đa hình trong CSharp




67Shares

1- Giới thiệu

Thừa kế và đa hình - đây là một khái niệm vô cùng quan trọng trong **CSharp**. Mà bạn bắt buộc phải hiểu nó.

2- Lớp, đối tượng và Constructor

Bạn cần hiểu một cách rạch ròi về class, cấu tử (constructor) và đối tượng trước khi bắt đầu tìm hiểu quan hệ thừa kế trong **CSharp**. Chúng ta xem lớp **Person**, mô tả một con người với các thông tin liên quan.

		
Person (Class)	Thomas Edison	Bill Gates
	Name: Thomas Edison Born Year: 1847 Place Of Birth:	Name: Bill Gates Born Year: 1955 Place Of Birth: Seattle, Washington

Person.cs

```
?  
1 using System;  
2 using System.Collections.Generic;  
3 using System.Linq;  
4 using System.Text;  
5 using System.Threading.Tasks;  
6  
7 namespace InheritancePolymorphism  
8 {  
9     class Person
```

```

10     {
11         // Trường name - thông tin tên người
12         public String Name;
13
14         // Trường bornYear - thông tin năm sinh
15         public int BornYear;
16
17         // Nơi sinh
18         public String PlaceOfBirth;
19
20         // Constructor này có 3 tham số.
21         // Mục đích để khởi tạo các giá trị cho các trường
22         // của Person.
23         // Chỉ định rõ tên, năm sinh, nơi sinh.
24         public Person(String Name, int BornYear, String PlaceOfBirth)
25         {
26             this.Name = Name;
27             this.BornYear = BornYear;
28             this.PlaceOfBirth = PlaceOfBirth;
29         }
30
31         // Constructor này có 2 tham số.
32         // Mục đích khởi tạo giá trị cho 2 trường tên và năm sinh
33         // cho Person.
34         // Nơi sinh không được khởi tạo.
35         public Person(String Name, int BornYear)
36         {
37             this.Name = Name;
38             this.BornYear = BornYear;
39         }
40
41     }
42 }

```

PersonDemo.cs

```

?
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace InheritancePolymorphism
8  {
9      class PersonDemo
10     {
11
12         static void Main(string[] args)
13         {
14
15             // Đối tượng: Thomas Edison.
16             // Được tạo ra bởi Constructor có 2 tham số của lớp Person.
17             Person edison = new Person("Thomas Edison", 1847);

```

```

18
19         Console.WriteLine("Info:");
20         Console.WriteLine("Name: " + edison.Name);
21         Console.WriteLine("Born Year: " + edison.BornYear);
22         Console.WriteLine("Place Of Birth: " + edison.PlaceOfBirth);
23
24         // Đối tượng: Bill Gates
25         // Được tạo ra bởi Constructor có 3 tham số của lớp Person.
26         Person billGates = new Person("Bill Gate", 1955,
27         "Seattle, Washington");
28
29         Console.WriteLine("-----");
30
31         Console.WriteLine("Info:");
32         Console.WriteLine("Name: " + billGates.Name);
33         Console.WriteLine("Born Year: " + billGates.BornYear);
34         Console.WriteLine("Place Of Birth: " +
35         billGates.PlaceOfBirth);
36
37         Console.ReadLine();
38     }
}

```

Kết quả chạy lớp **PersonDemo**:

```

Info:
Name: Thomas Edison
Born Year: 1847
Place Of Birth:
-----
Info:
Name: Bill Gate
Born Year: 1955
Place Of Birth: Seattle, Washington
-

```

Phân biệt Lớp, Constructor và đối tượng:

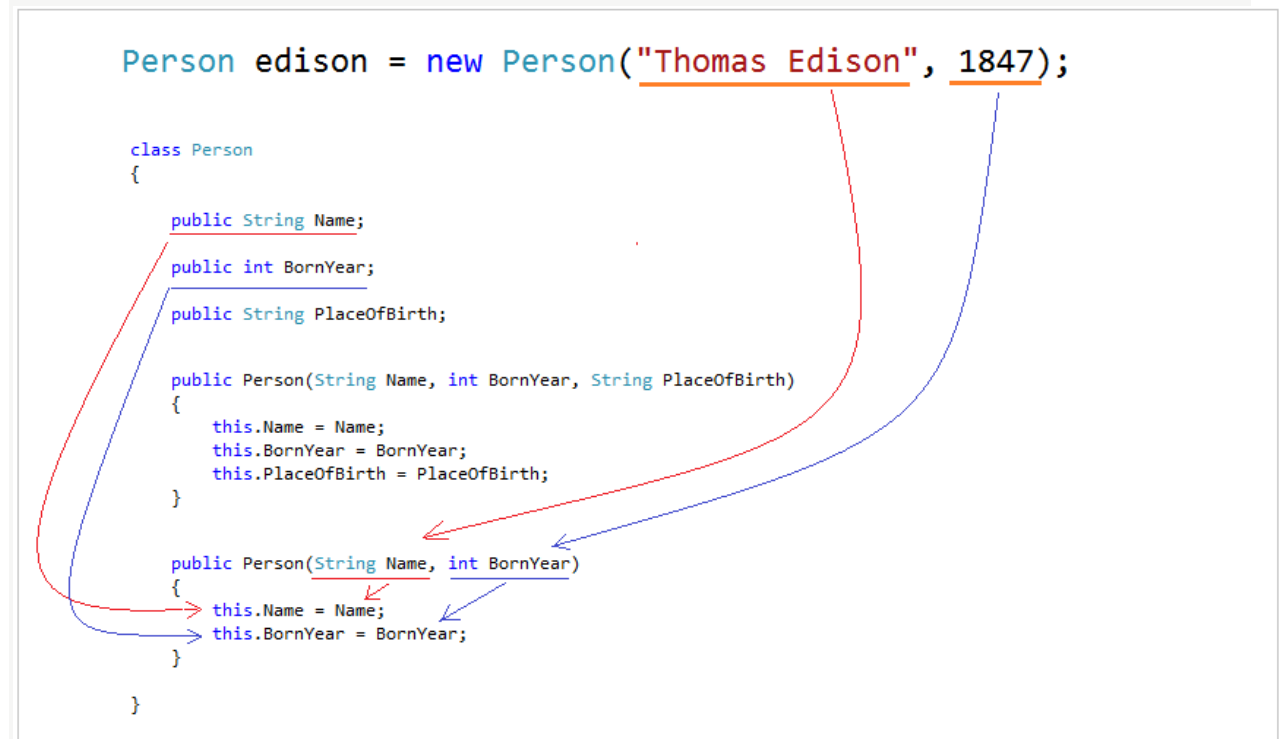
Lớp **Person** mô phỏng một lớp người, nó là một thứ gì đó trừu tượng, nhưng nó có các trường để mang thông tin, trong ví dụ trên là tên, năm sinh, nơi sinh.

Constructor (Cấu tử).

- Constructor luôn có tên giống tên của lớp.
- Một lớp có thể có một hoặc nhiều Constructor.
- Constructor có thể có hoặc không có tham số, Constructor không có tham số còn gọi là Constructor mặc định.
- Constructor được sử dụng để tạo ra một đối tượng của lớp đó.

Như vậy lớp **Person** (Mô tả lớp người) là thứ trừu tượng, nhưng khi chỉ rõ vào bạn hoặc tôi thì đó là 2 đối tượng thuộc lớp **Person**. Và Constructor là một phương thức đặc biệt để tạo ra đối tượng, Constructor sẽ gán các giá trị vào các trường (field) của đối tượng được tạo ra..

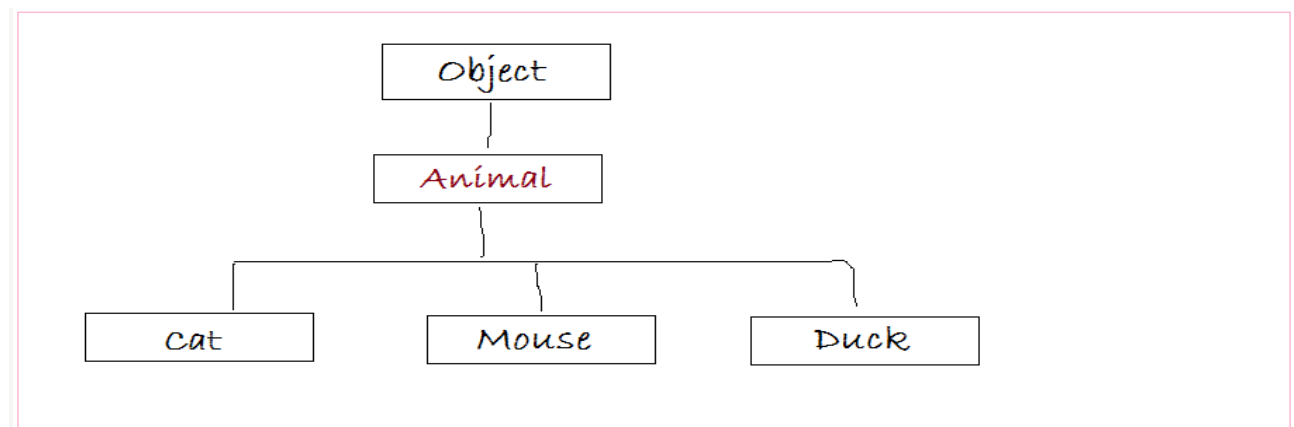
Đây là hình ảnh minh họa các trường (field) của lớp được gán giá trị thế nào, khi bạn tạo một đối tượng từ Constructor.



3- Thừa kế trong CSharp

Chúng ta cần một vài lớp tham gia vào các ví dụ.

- **Animal**: Lớp mô phỏng một động vật.
- **Duck**: Lớp mô phỏng con vịt, là một lớp con của **Animal**.
- **Cat**: Lớp mô phỏng con mèo, là một lớp con của **Animal**.
- **Mouse**: Lớp mô phỏng con chuột, là một lớp con của **Animal**.



Như đã biết ở phần trước, Constructor của lớp được sử dụng để tạo ra một đối tượng, và gán giá trị cho các trường (field).

Constructor của lớp con bao giờ cũng gọi tới một Constructor của lớp cha để khởi tạo giá trị cho các trường của lớp cha, sau đó nó mới khởi tạo giá trị cho các trường của nó.

Hãy xem ví dụ:

Animal.cs

```
?  
1  using System;  
2  using System.Collections.Generic;  
3  using System.Linq;  
4  using System.Text;  
5  using System.Threading.Tasks;  
6  
7  namespace InheritancePolymorphism  
8  {  
9      public abstract class Animal  
10     {  
11         // Đây là Trường Name (Tên).  
12         // ví dụ Mèo Tom, Chuột Jerry.  
13         public string Name;  
14  
15         // Constructor mặc định.  
16         public Animal()  
17         {  
18             Console.WriteLine("- Animal()");  
19         }  
20  
21         public Animal(string Name)  
22         {  
23             // Gán giá trị cho trường Name.  
24             this.Name = Name;  
25             Console.WriteLine("- Animal(string)");  
26         }  
27  
28         // Move(): Con vật di chuyển.  
29         // virtual: Cho phép lớp con có thể ghi đè (override)  
30         // phương thức này.  
31         public virtual void Move()  
32         {  
33             Console.WriteLine("Animal Move");  
34         }  
35  
36         public void Sleep()  
37         {  
38             Console.WriteLine("Sleep");  
39         }  
40  
41     }
```

42 }

Cat là lớp con thừa kế từ lớp **Animal**, nó cũng có các trường của mình.

Cat.cs

?

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace InheritancePolymorphism
8  {
9      public class Cat : Animal
10     {
11
12         public int Age;
13         public int Height;
14
15         // Đây là một Constructor có 3 tham số của lớp Cat.
16         // Sử dụng :base(name) để gọi đến Constructor của lớp
17         // cha: Animal(string).
18         // Các trường của lớp cha sẽ được gán giá trị.
19         // Sau đó các trường của lớp này mới được gán giá trị.
20         public Cat(string name, int Age, int Height)
21             : base(name)
22         {
23             this.Age = Age;
24             this.Height = Height;
25             Console.WriteLine("- Cat(string,int,int)");
26         }
27
28         // Constructor này gọi tới Constructor từ mặc định
29         //(Không tham số) của lớp cha.
30         public Cat(int Age, int Height)
31             : base()
32         {
33             this.Age = Age;
34             this.Height = Height;
35             Console.WriteLine("- Cat(int,int)");
36         }
37
38         public void Say()
39         {
40             Console.WriteLine("Meo");
41         }
42     }
```



```

42
43         // Ghi đè phương thức Move() của lớp cha (Animal).
44         // Viết lại phương thức này,
45         // để mô tả chính xác hành vi di chuyển của loài Mèo.
46         public override void Move()
47         {
48             Console.WriteLine("Cat Move ...");
49         }
50     }
51 }

```

CatTest.cs

?

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace InheritancePolymorphism
8  {
9      class CatTest
10     {
11         static void Main(string[] args)
12         {
13
14             Console.WriteLine("Create object Cat from
15             Cat(string,int,int)");
16
17             // Khởi tạo một đối tượng Cat từ câu từ 3 tham số.
18             // Trường Name của Animal sẽ được gán giá trị "Tom".
19             // Trường Age của Cat sẽ được gán giá trị 3
20             // Trường Height của Cat sẽ được gán giá trị 20.
21             Cat tom = new Cat("Tom",3, 20);
22
23             Console.WriteLine("-----");
24
25             Console.WriteLine("Name = {0}", tom.Name);
26             Console.WriteLine("Age = {0}", tom.Age);
27             Console.WriteLine("Height = {0}", tom.Height);
28
29             Console.WriteLine("-----");
30
31             // Gọi phương thức được thừa kế từ lớp Animal.
32             tom.Sleep();
33
34             // Gọi phương thức Say() (của lớp Cat)
35             tom.Say();
36

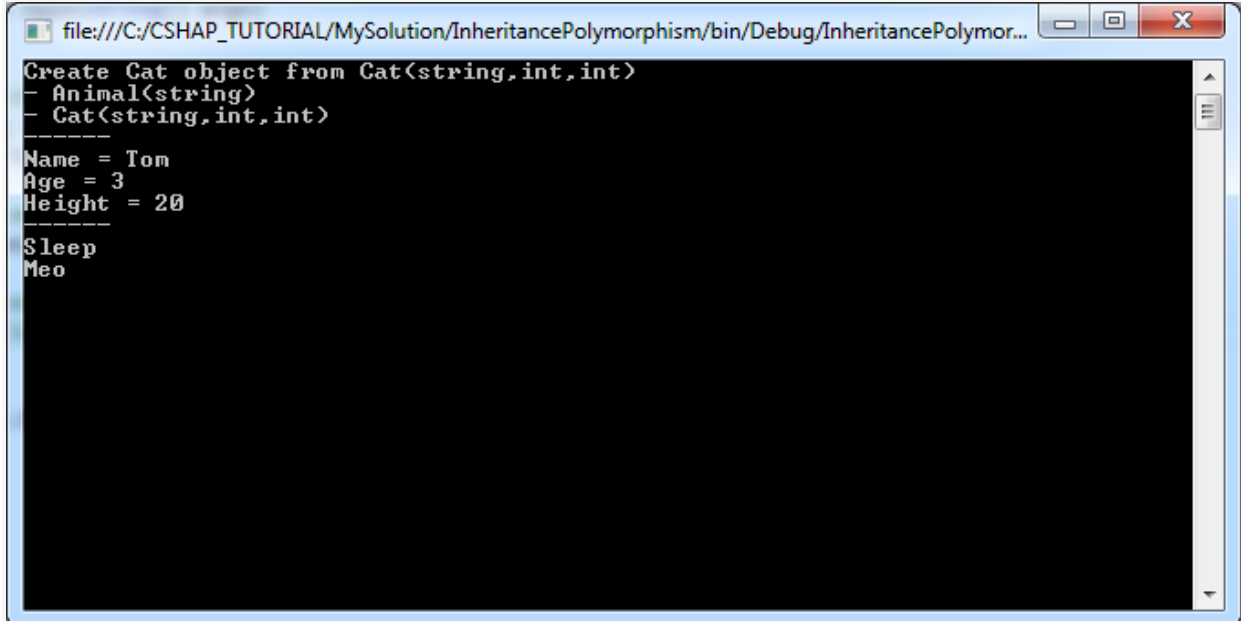
```

```

37
38         Console.ReadLine();
39     }
40 }
41 }

```

Kết quả chạy lớp **CatTest**:



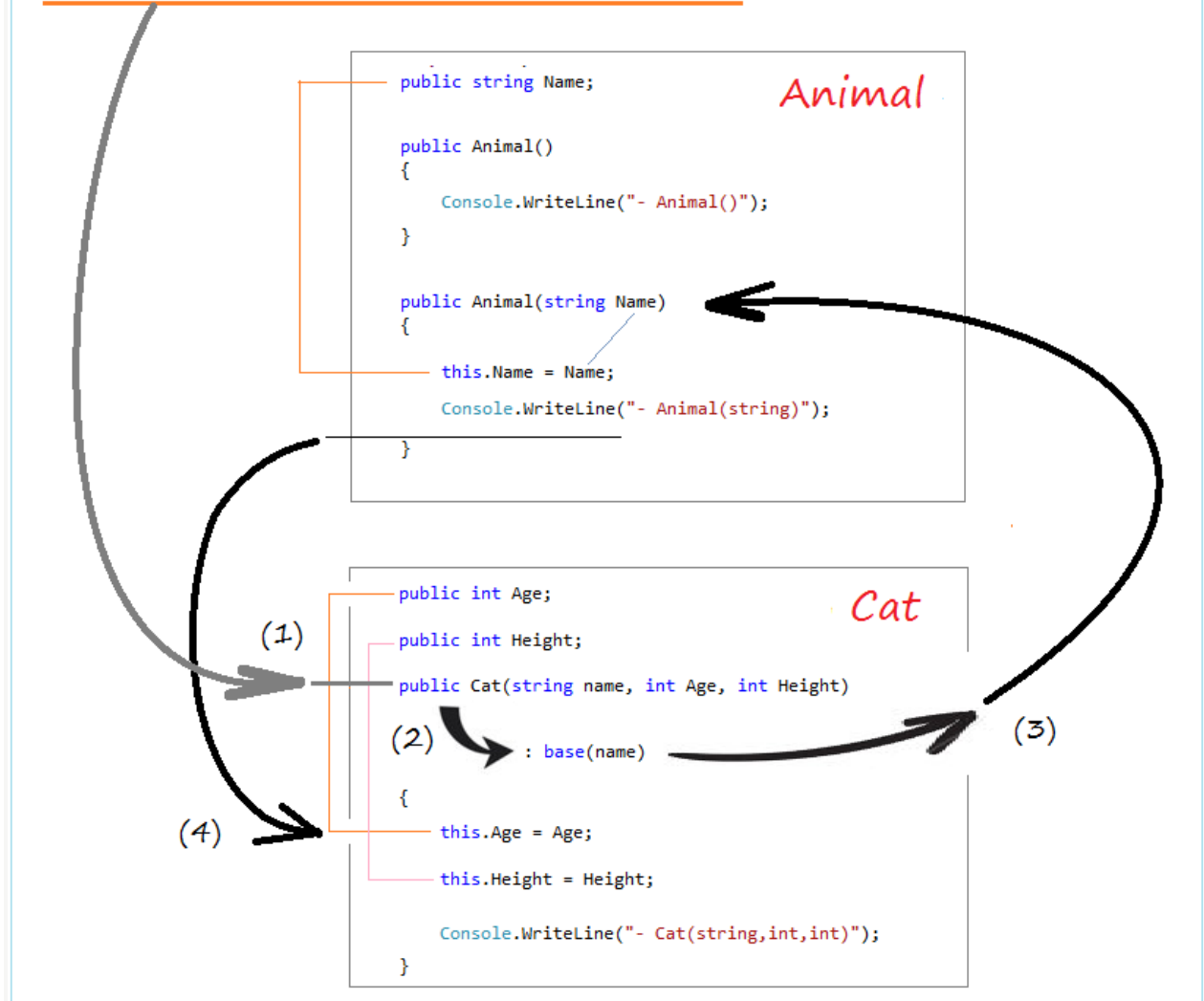
```

file:///C:/CSHAP_TUTORIAL/MySolution/InheritancePolymorphism/bin/Debug/InheritancePolymor...
Create Cat object from Cat(string,int,int)
- Animal(string)
- Cat(string,int,int)
-----
Name = Tom
Age = 3
Height = 20
-----
Sleep
Meo

```

Điều gì đã xảy ra khi bạn khởi tạo một đối tượng từ một Constructor? Nó sẽ gọi lên một Constructor của lớp cha như thế nào? Bạn hãy xem hình minh họa dưới đây:

Cat tom = new Cat("Tom", 3, 20);



Với hình minh họa trên bạn thấy rằng, Constructor của lớp cha bao giờ cũng được gọi trước Constructor của lớp con, nó sẽ gán giá trị cho các trường của lớp cha trước, sau đó các trường của lớp con mới được gán giá trị.

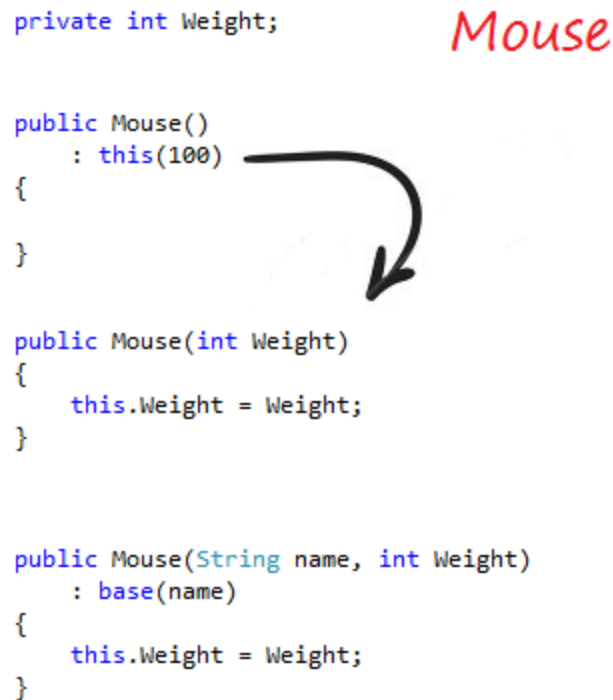
Khi bạn viết một Constructor và không chỉ rõ nó dựa trên (base) Constructor nào của lớp cha, **CSharp** mặc định hiểu là Constructor đó dựa trên Constructor mặc định của lớp cha.

?

```
1 // Constructor này không chỉ rõ nó dựa
2 // trên (base) Constructor nào của lớp cha.
3 public Cat(int Age, int Height)
4 {
5
6 }
7
8 // Nó sẽ tương đương với:
9 public Cat(int Age, int Height) : base()
10 {
11 }
```

Một Constructor có thể gọi tới một Constructor khác trong cùng một lớp bằng cách sử dụng **:this**.

```
?
1 private int Weight;
2
3 // Constructor mặc định (Không có tham số).
4 // Nó gọi tới constructor Mouse(int).
5 public Mouse() : this(100)
6 {
7 }
8
9 // Đây là constructor có 1 tham số.
10 // Không chỉ rõ :base
11 // Nghĩa là nó dựa trên (base) Constructor mặc định của lớp cha.
12 public Mouse(int Weight)
13 {
14     this.Weight = Weight;
15 }
```



```
private int Weight;

public Mouse()
    : this(100)
{
}

public Mouse(int Weight)
{
    this.Weight = Weight;
}

public Mouse(String name, int Weight)
    : base(name)
{
    this.Weight = Weight;
}
```

Mouse.cs

```
?
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace InheritancePolymorphism
```

```

8  {
9      public class Mouse : Animal
10     {
11
12         private int Weight;
13
14         // Constructor mặc định (Không có tham số).
15         // Gọi tới Mouse(int)
16         public Mouse()
17             : this(100)
18         {
19         }
20
21         // Constructor này có 1 tham số.
22         // Và không ghi rõ :base
23         // Nghĩa là nó base (dựa trên) Constructor mặc định
24         // của lớp cha.
25         public Mouse(int Weight)
26         {
27             this.Weight = Weight;
28         }
29
30         // Constructor này có 2 tham số.
31         public Mouse(String name, int Weight)
32             : base(name)
33         {
34             this.Weight = Weight;
35         }
36
37         public int GetWeight()
38         {
39             return Weight;
40         }
41
42         public void SetWeight(int Weight)
43         {
44             this.Weight = Weight;
45         }
46
47
48     }
49 }

```

Sử dụng toán tử 'is' bạn có thể kiểm tra một đối tượng có phải là kiểu của một class nào đó hay không. Hãy xem ví dụ dưới đây:

IsOperatorDemo.cs

?

```

1  using System;
2  using System.Collections.Generic;

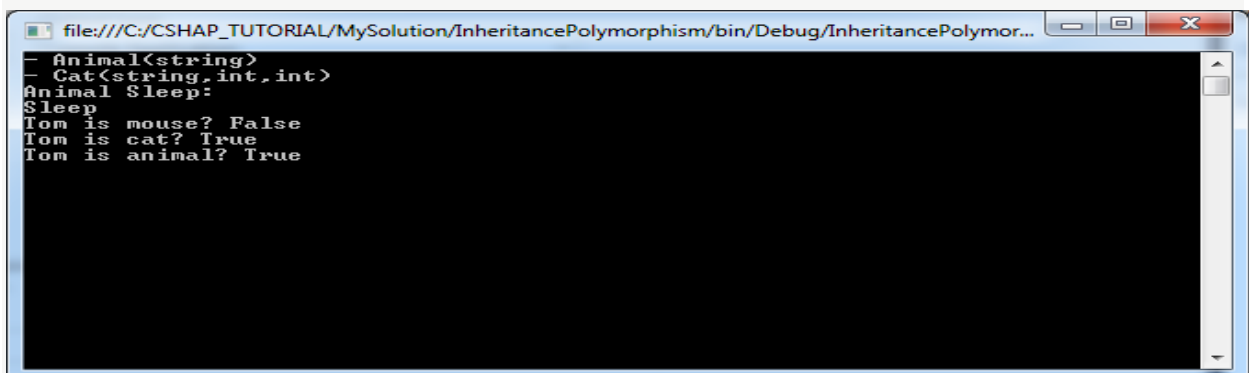
```

```

3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace InheritancePolymorphism
8  {
9      class IsOperatorDemo
10     {
11         static void Main(string[] args)
12         {
13             // Khởi tạo một đối tượng Animal.
14             // Animal là lớp trừu tượng,
15             // Vì vậy bạn không thể tạo đối tượng
16             // từ Constructor của Animal.
17             Animal tom = new Cat("Tom", 3, 20);
18
19             Console.WriteLine("Animal Sleep:");
20
21             // Gọi phương thức Sleep() của Animal
22             tom.Sleep();
23
24             // Sử dụng toán tử 'is' để kiểm tra xem
25             // một đối tượng có phải kiểu nào đó không.
26             bool isMouse = tom is Mouse; // false
27
28             Console.WriteLine("Tom is mouse? " + isMouse);
29
30             bool isCat = tom is Cat; // true
31             Console.WriteLine("Tom is cat? " + isCat);
32
33             bool isAnimal = tom is Animal; // true
34             Console.WriteLine("Tom is animal? " + isAnimal);
35
36             Console.ReadLine();
37         }
38     }
39 }

```

Chạy ví dụ:



```

file:///C:/CSHAP_TUTORIAL/MySolution/InheritancePolymorphism/bin/Debug/InheritancePolymor...
- Animal<string>
- Cat<string,int,int>
Animal Sleep:
Sleep
Tom is mouse? False
Tom is cat? True
Tom is animal? True

```

Ép kiểu (cast) trong CSharp.

CastDemo.cs

?

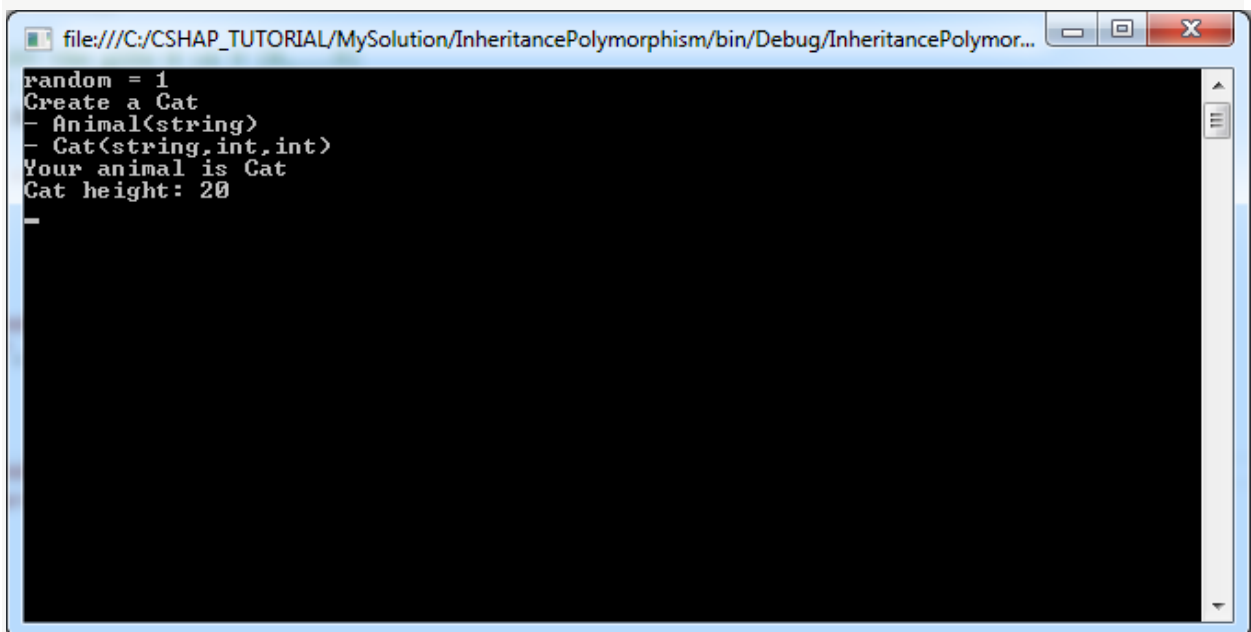
```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace InheritancePolymorphism
8  {
9      class CastDemo
10     {
11         static void Main(string[] args)
12         {
13             // Gọi phương thức, nó trả về một con vật ngẫu nhiên.
14             Animal animal = GetRandomAnimal();
15
16             if (animal is Cat)
17             {
18                 Console.WriteLine("Your animal is Cat");
19
20                 // Ép kiểu (cast) thành đối tượng Cat.
21                 Cat cat = (Cat)animal;
22
23                 // Và truy cập vào trường Height của
24                 // đối tượng Cat.
25                 Console.WriteLine("Cat height: " + cat.Height);
26             }
27             else if (animal is Mouse)
28             {
29                 Console.WriteLine("Your animal is Mouse");
30
31                 // Ép kiểu (cast) thành đối tượng Mouse.
32                 Mouse mouse = (Mouse)animal;
33
34                 // Và gọi một phương thức của lớp Mouse.
35                 Console.WriteLine("Mouse weight: " +
36                 mouse.GetWeight());
37             }
38
39             Console.ReadLine();
40         }
41
42         // Method trả về ngẫu nhiên một con vật.
43         public static Animal GetRandomAnimal()
44         {
```

```

45         // Trả về số ngẫu nhiên nằm giữa 0 và 9 (0,...9)
46         int random = new Random().Next(0, 10);
47
48         Console.WriteLine("random = " + random);
49
50         Animal animal = null;
51         if (random < 5)
52         {
53             Console.WriteLine("Create a Cat");
54             animal = new Cat("Tom", 3, 20);
55         }
56         else
57         {
58             Console.WriteLine("Create a Mouse");
59             animal = new Mouse("Jerry", 5);
60         }
61         return animal;
62     }
63 }

```

Chạy ví dụ:



```

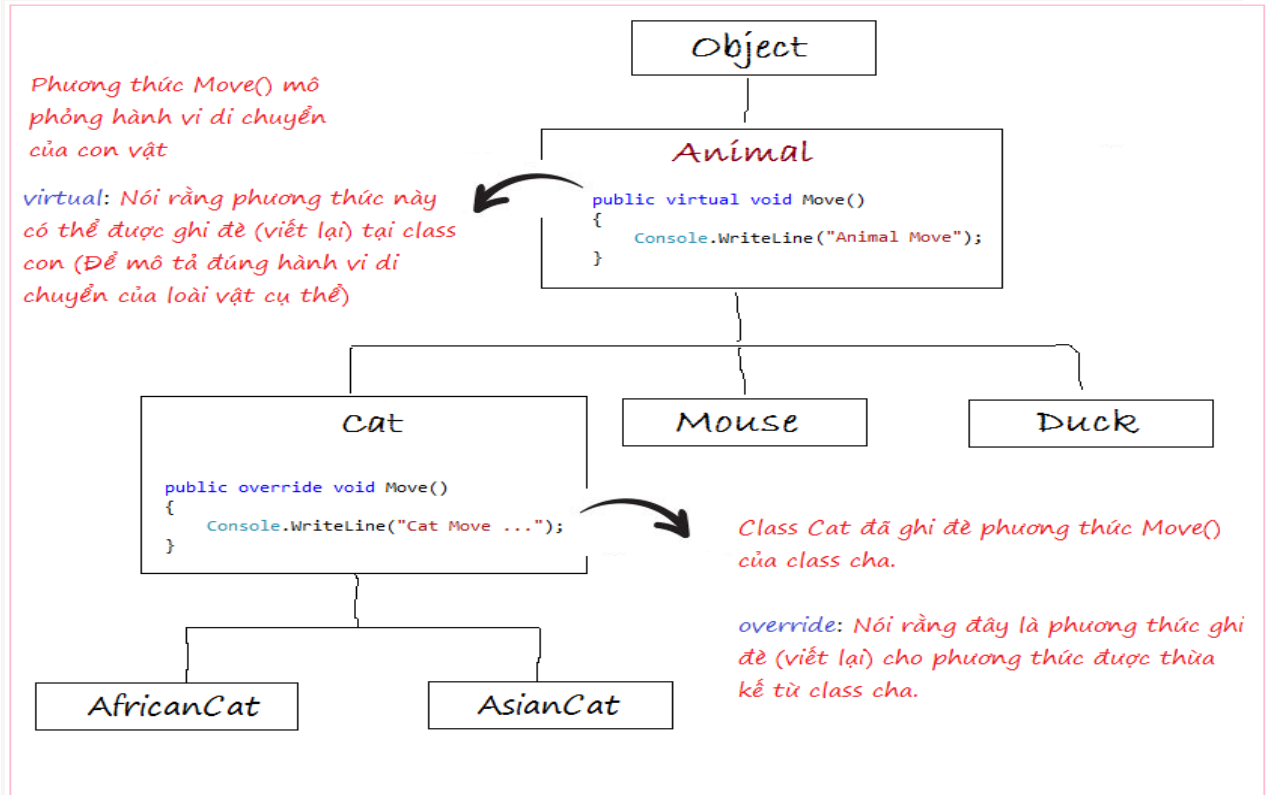
file:///C:/CSHAP_TUTORIAL/MySolution/InheritancePolymorphism/bin/Debug/InheritancePolymor...
random = 1
Create a Cat
- Animal(string)
- Cat(string,int,int)
Your animal is Cat
Cat height: 20
-

```


4- Đa hình trong CSharp

Đa hình (**Polymorphism**) từ này có nghĩa là có nhiều hình thức. Trong mô hình lập trình hướng đối tượng, đa hình thường được diễn tả như là "một giao diện, nhưng nhiều chức năng".

Đa hình có thể là tĩnh hoặc động. Trong đa hình tĩnh, phản ứng với một chức năng được xác định tại thời gian biên dịch. Trong đa hình động, nó được quyết định tại thời gian chạy (run-time).



Bạn có một con mèo nguồn gốc châu Á (AsianCat), bạn có thể nói nó là một con mèo (Cat) hoặc nói nó là một con vật (Animal) đó là một khía cạnh của từ đa hình.

Hoặc một ví dụ khác: Trên lý lịch của bạn ghi rằng bạn là một người châu Á, trong khi đó bạn thực tế là một người Việt Nam. Và tôi cũng có thể nói rằng bạn là người trái đất.

Ví dụ dưới đây cho bạn thấy cách hành xử giữa khai báo và thực tế:

PolymorphismCatDemo.cs

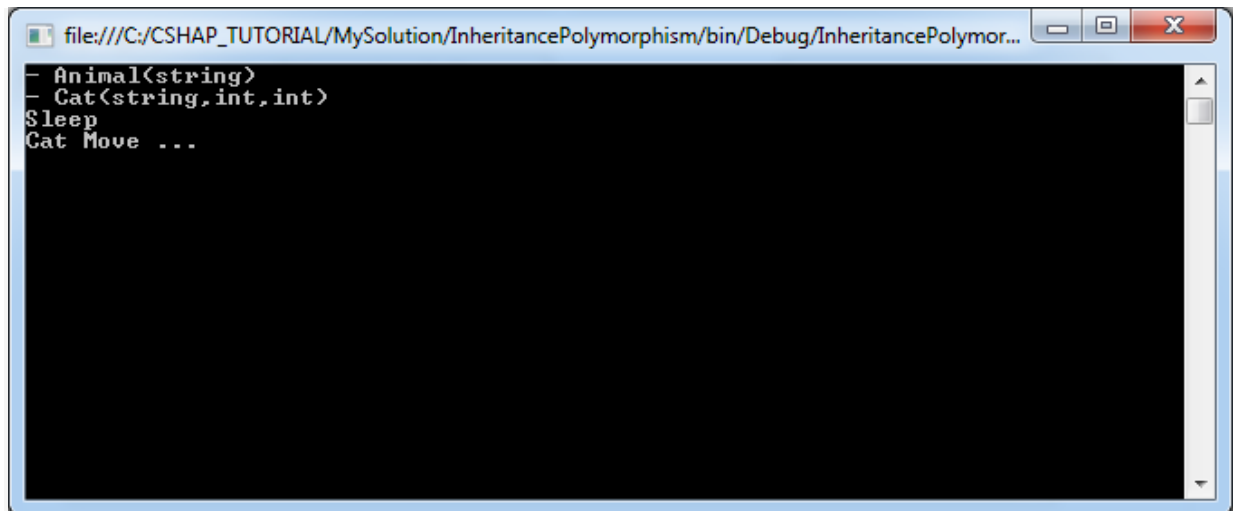
?

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace InheritancePolymorphism
8 {
9     class PolymorphismCatDemo
10    {
11        static void Main(string[] args)
```

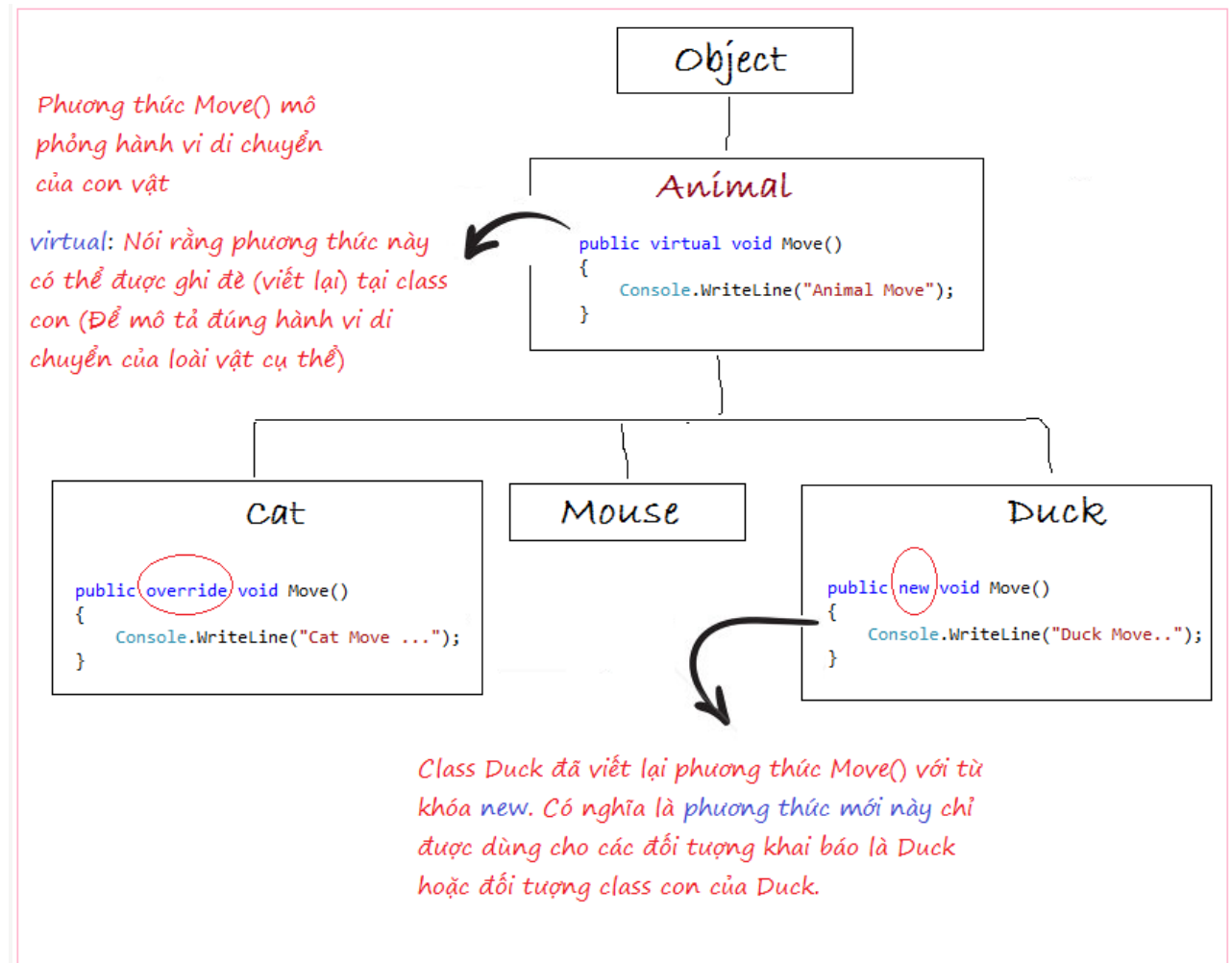
```

12         {
13             // Bạn khai báo một đối tượng Animal (Động vật).
14             // Bằng cách tạo nó bởi Constructor của lớp Cat.
15             // ** Đối tượng 'tom' khai báo là Animal
16             // vì vậy nó chỉ có thể gọi các phương thức của Animal.
17             Animal tom = new Cat("Tom", 3, 20);
18
19             // Gọi phương thức Sleep.
20             tom.Sleep();
21
22             // - Move() là phương thức được định nghĩa trong
23             // lớp Animal.
24             // - Move() được ghi đè (override) trong lớp Cat.
25             // 'tom' thực tế là Cat, mặc dù đang được khai báo
26             // là Animal.
27             // ==> Nó sẽ gọi phương thức Move() định nghĩa
28             // trong lớp Cat.
29             tom.Move(); // ==> Cat Move.
30
31
32
33             Console.ReadLine();
34         }
35     }
36 }

```



Bạn có 2 cách để viết lại một phương thức được định nghĩa ở lớp cha, bằng cách sử dụng từ khóa **override** hoặc **new**. Và chúng rất khác nhau. Hãy xem hình minh họa dưới đây:



Duck.cs

?

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace InheritancePolymorphism
8 {
9     class Duck : Animal
10    {
11
12        public Duck(string name)
13            : base(name)
14        {
15            Console.WriteLine("- Duck(string)");
16        }
17    }
```

```

18
19         public new void Move()
20         {
21             Console.WriteLine("Duck Move..");
22         }
23     }
24 }

```

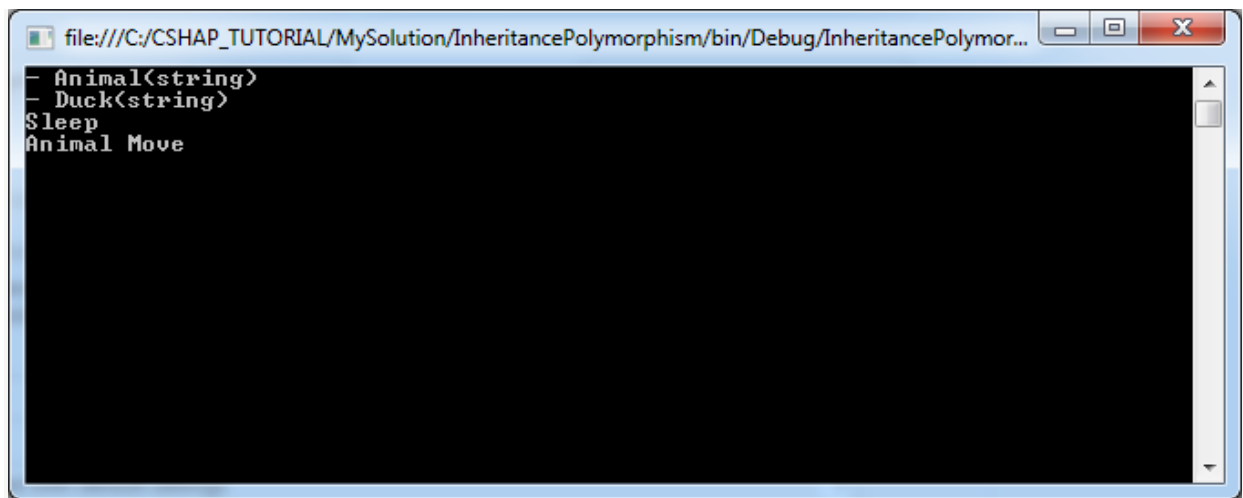
InheritanceDuckDemo.cs

```

?
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace InheritancePolymorphism
8  {
9      class InheritanceDuckDemo
10     {
11         static void Main(string[] args)
12         {
13             // Bạn khai báo một đối tượng Animal.
14             // Nhưng tạo nó bằng Constructor của lớp Duck.
15             Animal donald = new Duck("Donald");
16
17             // Gọi phương thức Sleep() (Được định nghĩa
18             // trong lớp Animal).
19             donald.Sleep();
20
21             // - Move() là phương thức được định nghĩa trong
22             // lớp Animal.
23             // - Move() được "viết lại" với từ khóa 'new'
24             // trong lớp Duck.
25             // ('new' Move() chỉ được dùng cho các đối tượng
26             // được khai báo là Duck hoặc lớp con của Duck).
27             // 'donald' is declared as 'Animal', not 'Duck'.
28             donald.Move(); // ==> Animal Move.
29
30
31             Console.ReadLine();
32         }
33     }
34 }

```

Chạy ví dụ:



A screenshot of a Visual Studio console window. The title bar shows the file path: file:///C:/CSHAP_TUTORIAL/MySolution/InheritancePolymorphism/bin/Debug/InheritancePolymor... The console output is as follows:

```
- Animal<string>  
- Duck<string>  
Sleep  
Animal Move
```

Hết