

TỔNG QUAN (TT)

Biên soạn: **ThS. Nguyễn Thị Anh Thư**

1

NỘI DUNG

1. Tính đóng gói
2. Phương thức
3. Mảng (Array)
4. Chuỗi (String)
5. Structure
6. Lớp (Class)

1. TÍNH ĐÓNG GÓI

- **Encapsulation (Tính đóng gói)**, trong phương pháp lập trình hướng đối tượng, ngăn cản việc truy cập tới chi tiết của trình triển khai (Implementation Detail).
- Tính trừu tượng và tính đóng gói là hai tính chất có liên quan đến nhau trong lập trình hướng đối tượng. Tính trừu tượng cho phép các thông tin liên quan hiển thị và tính đóng gói cho phép một lập trình viên triển khai các mức độ mong muốn của trừu tượng hóa.
- **Điều đó có nghĩa các phần của class được ẩn đi như là Đóng gói và hiển thị ra như là trừu tượng hóa.**

1. TÍNH ĐÓNG GÓI

- Tính đóng gói được triển khai bởi sử dụng **Access Specifier**. Một Access Specifier định nghĩa phạm vi và tính nhìn thấy của một thành viên lớp. C# hỗ trợ các Access Specifier sau:
 - Public
 - Private
 - Protected
 - Internal
 - Protected internal

1. TÍNH ĐÓNG GÓI

Độ truy cập (Modifier)	Mô tả
private	Truy cập bị hạn chế trong phạm vi của định nghĩa Class. Đây là loại phạm vi truy cập mặc định nếu không được chính thức chỉ định
protected	Truy cập bị giới hạn trong phạm vi định nghĩa của Class và bất kỳ các class con thừa kế từ class này.
internal	Truy cập bị giới hạn trong phạm vi Assembly của dự án hiện tại.
protected internal	Truy cập bị giới hạn trong phạm vi Assembly hiện tại và trong class định nghĩa hoặc các class con.
public	Không có bất kỳ giới hạn nào khi truy cập vào các thành viên công khai (public).

1. TÍNH ĐỒNG GÓI

	Cùng Assembly			Khác Assembly	
	Trong class định nghĩa?	Trong class con	Ngoài class định nghĩa, ngoài class con	Trong class con	Ngoài class con
private	Y				
protected	Y	Y		Y	
internal	Y	Y	Y		
protected internal	Y	Y	Y		
public	Y	Y	Y	Y	Y

1. TÍNH ĐỒNG GÓI

▪ **Public Access Specifier trong C#**

- Public Access Specifier trong C# cho phép một lớp trưng bày các biến thành viên và các hàm thành viên của nó tới các hàm và đối tượng khác. Bất kỳ thành viên public nào trong C# có thể được truy cập từ bên ngoài lớp đó.
- Public access modifier là mạnh mẽ nhất và có thể truy cập ở mọi nơi. Nó có phạm vi truy cập rộng nhất so với các modifier khác.

1. TÍNH ĐỒNG GÓI

- **Public Access Specifier trong C#**

- Ví dụ:

- Giả sử ta có hai lớp có tên lần lượt là: **Rectangle** và **ExecuteRectangle**.
- Lớp **Rectangle**: chứa các thuộc tính, phương thức.

```
class Rectangle
{
    //cac bien thanh vien
    public double length;
    public double width;

    //cac phuong thuc
    1reference
    public double GetArea()
    {
        return length * width;
    }
    1reference
    public void Display()
    {
        Console.WriteLine("Chieu dai: {0}", length);
        Console.WriteLine("Chieu rong: {0}", width);
        Console.WriteLine("Dien tich: {0}", GetArea());
    }
}
```


1. TÍNH ĐỒNG GÓI

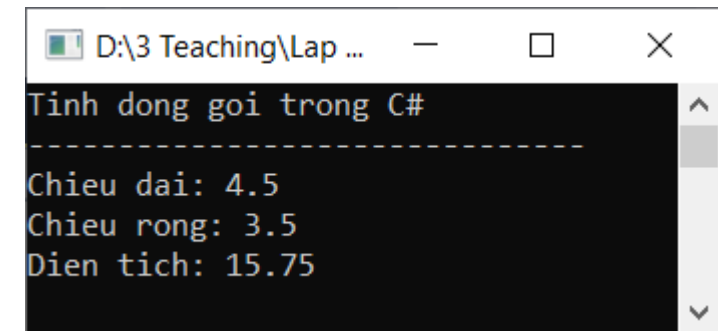
▪ Public Access Specifier trong C#

▪ Ví dụ:

- Giả sử ta có hai lớp có tên lần lượt là: **Rectangle** và **ExecuteRectangle**.
- Lớp **ExecuteRectangle**: chứa phương thức **main()** để thao tác trên đối tượng **Rectangle**.

```
class ExecuteRectangle
{
    0 references
    static void Main(string[] args)
    {
        Console.WriteLine("Tinh dong goi trong C#");
        Console.WriteLine("-----");

        //tao doi tuong Rectangle
        Rectangle r = new Rectangle();
        //thiet lap cac thuoc tinh
        r.length = 4.5;
        r.width = 3.5;
        //goi phuong thuc
        r.Display();
        Console.ReadLine();
    }
}
```



The screenshot shows a console window titled "D:\3 Teaching\Lap ..." with the following output:

```
Tinh dong goi trong C#
-----
Chieu dai: 4.5
Chieu rong: 3.5
Dien tich: 15.75
```

1. TÍNH ĐÓNG GÓI

- **Private Access Specifier trong C#**
- Private Access Specifier trong C# cho phép một lớp ẩn các biến thành viên và các hàm thành viên của nó với các hàm và đối tượng khác. Chỉ có các hàm trong cùng lớp đó có thể truy cập tới các thành viên private. Ngay cả khi một Instance của một lớp cũng không thể truy cập các thành viên private của nó.

1. TÍNH ĐỒNG GÓI

- Private Access Specifier trong C#

- Ví dụ:

- Lớp Rectangle

```
class Rectangle
{
    //cac bien thanh vien
    private double length;
    private double width;

    //cac phuong thuc
    1 reference
    public void Acceptdetails()
    {
        Console.WriteLine("Nhap chieu dai: ");
        length = Convert.ToDouble(Console.ReadLine());
        Console.WriteLine("Nhap chieu rong: ");
        width = Convert.ToDouble(Console.ReadLine());
    }
    1 reference
    public double GetArea()
    {
        return length * width;
    }
    1 reference
    public void Display()
    {
        Console.WriteLine("Chieu dai: {0}", length);
        Console.WriteLine("Chieu rong: {0}", width);
        Console.WriteLine("Dien tich: {0}", GetArea());
    }
}
```

1. TÍNH ĐỒNG GÓI

- **Private Access Specifier trong C#**
- Ví dụ:
 - Lớp **ExecuteRectangle**

```
class ExecuteRectangle
{
    0 references
    static void Main(string[] args)
    {
        Console.WriteLine("Tinh dong goi trong C#");
        Console.WriteLine("-----");

        //tao doi tuong Rectangle
        Rectangle r = new Rectangle();
        //thiet lap cac thuoc tinh
        r.length = 4.5;
        r.width = 3.5;
        //goi phuong thuc
        r.Acceptdetails();
        r.Display();
        Console.ReadLine();

        Console.ReadKey();
    }
}
```

1. TÍNH ĐÓNG GÓI

- **Private Access Specifier trong C#**

- Ví dụ:

- Trong ví dụ, các biến thành viên `length` và `width` được khai báo **private**, vì thế chúng không thể được truy cập từ hàm **Main()**. Trong Visual Studio sẽ báo hiệu một dấu gạch đỏ bên dưới hai biến này, giống như:

```
//tao doi tuong Rectangle
Rectangle r = new Rectangle();
//thiet lap cac thuoc tinh
r.length = 4.5;
r.width = 3.5;
//goi phuong thuc
r.Acceptdetails();
r.Display();
```

1. TÍNH ĐÓNG GÓI

- **Protected Access Specifier trong C#**
- Protected Access Specifier trong C# cho phép một lớp con truy cập các biến thành viên và các hàm thành viên của lớp cơ sở của nó. Cách này giúp triển khai tính kế thừa.

1. TÍNH ĐỒNG GÓI

- **Internal Access Specifier trong C#**
- Internal Access Specifier trong C# cho phép một lớp trưng bày các biến thành viên và các hàm thành viên của nó tới các hàm và đối tượng khác trong *Assembly* hiện tại.
- Nói cách khác, bất kỳ thành viên nào với Internal Access Specifier trong C# có thể được truy cập từ bất kỳ lớp hoặc phương thức được định nghĩa bên trong ứng dụng mà thành viên đó được định nghĩa.

1. TÍNH ĐỒNG GÓI

- **Internal Access Specifier trong C#**

- Ví dụ:

- **Lớp Rectangle**

```
class Rectangle
{
    //cac bien thanh vien
    internal double length;
    internal double width;

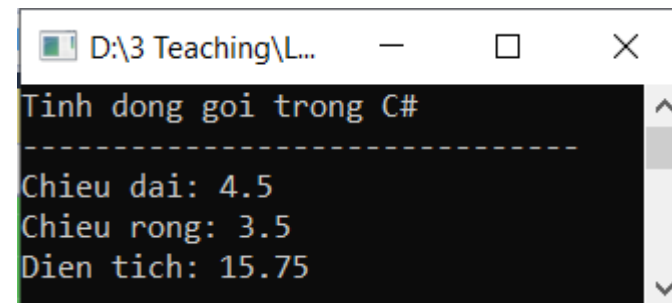
    //cac phuong thuc
    1 reference
    double GetArea()
    {
        return length * width;
    }
    1 reference
    public void Display()
    {
        Console.WriteLine("Chieu dai: {0}", length);
        Console.WriteLine("Chieu rong: {0}", width);
        Console.WriteLine("Dien tich: {0}", GetArea());
    }
}
```


1. TÍNH ĐỒNG GÓI

- **Internal Access Specifier trong C#**
- Ví dụ:
 - Lớp **ExecuteRectangle**

```
class ExecuteRectangle
{
    0 references
    static void Main(string[] args)
    {
        Console.WriteLine("Tinh dong goi trong C#");
        Console.WriteLine("-----");

        //tao doi tuong Rectangle
        Rectangle r = new Rectangle();
        //thiet lap cac thuoc tinh
        r.length = 4.5;
        r.width = 3.5;
        //goi phuong thuc
        r.Display();
        Console.ReadLine();
    }
}
```



The screenshot shows a console window titled "D:\3 Teaching\L..." with the following output:

```
Tinh dong goi trong C#
-----
Chieu dai: 4.5
Chieu rong: 3.5
Dien tich: 15.75
```

1. TÍNH ĐÓNG GÓI

- **Protected Internal Access Specifier trong C#**
- Protected Internal Access Specifier trong C# cho phép một lớp ẩn các biến thành viên và các hàm thành viên của nó với các hàm và đối tượng khác, ngoại trừ một lớp con bên trong cùng ứng dụng đó.
- Điều này cũng được sử dụng trong khi triển khai tính kế thừa trong C#.

2. PHƯƠNG THỨC

- Một phương thức là một nhóm lệnh cùng nhau thực hiện một tác vụ.
- **Mỗi chương trình C# có ít nhất một lớp với một phương thức là Main.**
- Để sử dụng một phương thức trong C# cần:
 - Định nghĩa phương thức
 - Gọi phương thức

2. PHƯƠNG THỨC

▪ Định nghĩa phương thức trong C#

```
<Access Specifier> <Kiểu_trả_về> <tên_phương_thức>(danh_sách_tham_số)  
{  
    phần thân phương thức  
}
```

2. PHƯƠNG THỨC

Định nghĩa phương thức trong C#

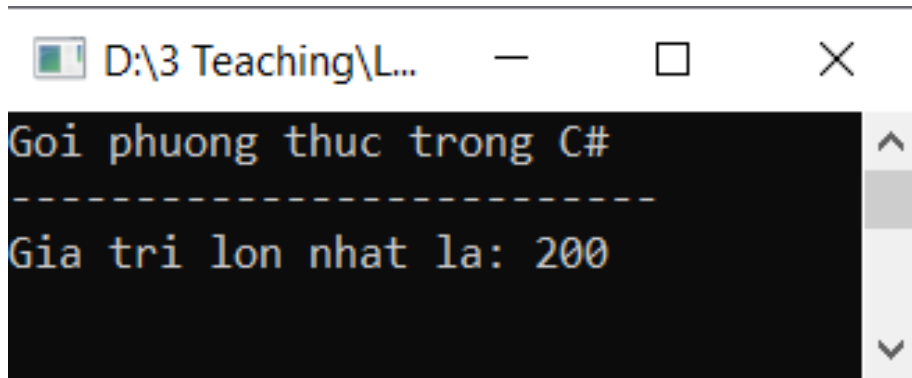
- **Access Specifier:** Định nghĩa tính nhìn thấy của một biến hoặc một phương thức với lớp khác.
- **Kiểu trả về:** Một phương thức có thể trả về một giá trị. Kiểu trả về là kiểu dữ liệu của giá trị mà phương thức trả về. Nếu phương thức không trả về bất kỳ giá trị nào, thì kiểu trả về là **void**.
- **tên_phương_thức:** Tên phương thức là một định danh duy nhất và nó là phân biệt kiểu chữ. Nó không thể giống bất kỳ định danh nào khác đã được khai báo trong lớp đó.
- **danh_sách_tham_số:** Danh sách tham số được bao quanh trong dấu ngoặc đơn, các tham số này được sử dụng để truyền và nhận dữ liệu từ một phương thức. Danh sách tham số liên quan tới kiểu, thứ tự và số tham số của một phương thức. Các tham số là tùy ý, tức là một phương thức có thể không chứa tham số nào.
- **phần thân phương thức:** Phần thân phương thức chứa tập hợp các chỉ thị cần thiết để hoàn thành hoạt động đã yêu cầu.

2. PHƯƠNG THỨC

- **Ví dụ:**
- Chương trình sau minh họa một hàm *FindMax* nhận hai giá trị integer và trả về số nào lớn hơn trong hai số. Nó có Access Specifier, vì thế nó có thể được truy cập từ bên ngoài lớp bởi sử dụng một Instance (sự thể hiện) của lớp đó.

2. PHƯƠNG THỨC

■ Ví dụ:



The screenshot shows a Windows command prompt window with the title bar "D:\3 Teaching\L...". The window contains the following text:

```
Goi phuong thuc trong C#
-----
Gia tri lon nhat la: 200
```

```
class Program
{
    1 reference
    public int FindMax(int num1, int num2)
    {
        /* khai bao bien cuc bo */
        int result;

        if (num1 > num2)
            result = num1;
        else
            result = num2;
        return result;
    }
    0 references
    static void Main(string[] args)
    {
        Console.WriteLine("Goi phuong thuc trong C#");
        Console.WriteLine("-----");
        /* phan dinh nghia bien cuc bo */
        int a = 100;
        int b = 200;
        int ret;
        Program n = new Program();

        //goi phuong thuc FindMax
        ret = n.FindMax(a, b);
        Console.WriteLine("Gia tri lon nhat la: {0}", ret);
        Console.ReadLine();

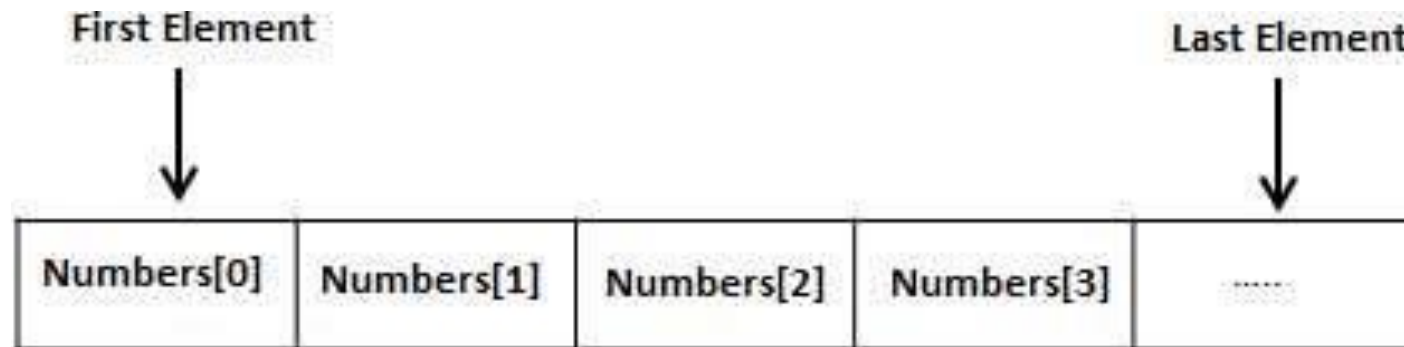
        Console.ReadKey();
    }
}
```

3. MẢNG (ARRAY)

- **Một mảng lưu giữ một tập hợp các phần tử có kích cỡ cố định trong cùng kiểu.**
- Một mảng được sử dụng để lưu giữ một tập hợp dữ liệu, nhưng nó thường hữu ích hơn khi nghĩ về một mảng như là một tập hợp các biến cùng kiểu được lưu giữ tại các vị trí bộ nhớ kề nhau.
- Thay vì khai báo biến một cách rời rạc, như biến `number0`, `number1`,... và `number99`, chúng ta có thể khai báo một mảng các giá trị như `numbers[0]`, `numbers[1]` và ... `numbers[99]` để biểu diễn các giá trị riêng biệt.
- **Một thành viên cụ thể của mảng có thể được truy cập qua index (chỉ số).**

3. MẢNG (ARRAY)

- **Tất cả mảng đều bao gồm các vị trí nhớ liền kề nhau.** Địa chỉ thấp nhất tương ứng với thành viên đầu tiên và địa chỉ cao nhất tương ứng với thành viên cuối cùng của mảng.



3. MẢNG (ARRAY)

Khai báo mảng trong C#

- Để khai báo một mảng trong ngôn ngữ C#, chúng ta có thể sử dụng cú pháp:

kiểu_dữ_liệu[] tên_mảng;

- Tại đây:
 - kiểu_dữ_liệu được sử dụng để xác định kiểu của phần tử trong mảng.
 - [] xác định rank hay kích cỡ của mảng.
 - tên_mảng xác định tên mảng.
- Ví dụ:
 - double[] balance;

3. MẢNG (ARRAY)

Khởi tạo mảng trong C#

- Việc khai báo một mảng không khởi tạo mảng trong bộ nhớ.
- Khi biến mảng được khởi tạo, chúng ta có thể gán giá trị cho mảng đó.
- **Mảng là một kiểu tham chiếu**, vì thế chúng ta cần sử dụng từ khóa **new** trong C# để tạo một Instance (sự thể hiện) của mảng đó.
- Ví dụ:

```
double[] balance = new double[10];
```

3. MẢNG (ARRAY)

Gán giá trị cho một mảng trong C#

- Có thể gán giá trị cho các phần tử mảng riêng biệt bởi sử dụng chỉ số mảng, như:

```
double[] balance = new double[10];
```

```
balance[0] = 4500.0;
```

- Có thể gán giá trị cho mảng tại thời điểm khai báo mảng, như sau:

```
double[] balance = { 2340.0, 4523.69, 3421.0};
```

- Có thể tạo và khai báo một mảng, như sau:

```
int [] marks = new int[5] { 99, 98, 92, 97, 95};
```

3. MẢNG (ARRAY)

Gán giá trị cho một mảng trong C#

- Có thể sao chép một biến mảng vào trong biến mảng mục tiêu khác. Trong tình huống này, cả biến mục tiêu và biến nguồn đều trở tới cùng vị trí bộ nhớ:

```
int [] marks = new int[] { 99, 98, 92, 97, 95};
```

```
int[] score = marks;
```

- Khi tạo một mảng, C# compiler ngầm định khởi tạo mỗi phần tử mảng thành một giá trị mặc định phụ thuộc vào kiểu mảng.
- Ví dụ: Với một mảng int, thì tất cả phần tử được khởi tạo là 0.

3. MẢNG (ARRAY)

Truy cập các phần tử mảng trong C#

■ Ví dụ 1:

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        Console.WriteLine("Mang trong C#");
        Console.WriteLine("-----");
        int[] n = new int[10]; /* n la mot mang gom 10 so nguyen */
        int i, j;
        /* khoi tao cac phan tu cua mang n */
        for (i = 0; i < 10; i++)
        {
            n[i] = i + 100;
        }

        /* hien thi gia tri cac phan tu cua mang n */
        for (j = 0; j < 10; j++)
        {
            Console.WriteLine("Phan tu [{0}] = {1}", j, n[j]);
        }
        Console.ReadKey();
    }
}
```

```
D:\3 Teach... Mang trong C#
-----
Phan tu [0] = 100
Phan tu [1] = 101
Phan tu [2] = 102
Phan tu [3] = 103
Phan tu [4] = 104
Phan tu [5] = 105
Phan tu [6] = 106
Phan tu [7] = 107
Phan tu [8] = 108
Phan tu [9] = 109
```

3. MẢNG (ARRAY)

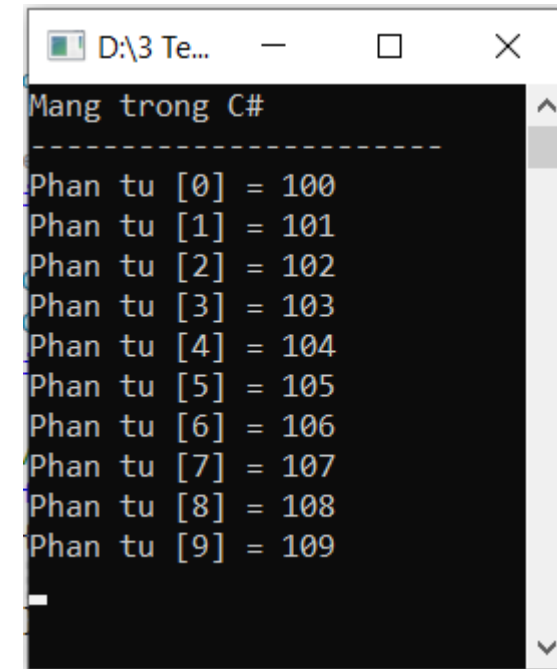
Truy cập các phần tử mảng trong C#

■ Ví dụ 2:

```
public static void Main(string[] args)
{
    Console.WriteLine("Mang trong C#");
    Console.WriteLine("-----");
    int[] n = new int[10]; /* n là một mảng gồm 10 số nguyên */

    /* khởi tạo các phần tử trong mảng n */
    for (int i = 0; i < 10; i++)
    {
        n[i] = i + 100;
    }

    /* hiển thị các giá trị của phần tử trong mảng n */
    foreach (int j in n)
    {
        int i = j - 100;
        Console.WriteLine("Phan tu [{0}] = {1}", i, j);
    }
    Console.ReadKey();
}
```



```
Mang trong C#
-----
Phan tu [0] = 100
Phan tu [1] = 101
Phan tu [2] = 102
Phan tu [3] = 103
Phan tu [4] = 104
Phan tu [5] = 105
Phan tu [6] = 106
Phan tu [7] = 107
Phan tu [8] = 108
Phan tu [9] = 109
```

3. MẢNG (ARRAY)

Chi tiết về mảng trong C#

- Mảng là một phần rất quan trọng trong ngôn ngữ C#.
- Dưới đây là những định nghĩa quan trọng liên quan đến mảng mà được trình bày rõ ràng hơn cho các lập trình viên C#:

Khái niệm	Miêu tả
Mảng đa chiều trong C#	C# hỗ trợ mảng đa chiều. Mẫu đơn giản nhất của mảng đa chiều là mảng hai chiều
Truyền mảng tới hàm trong C#	Có thể truyền cho hàm một con trỏ tới một mảng bằng việc xác định tên mảng mà không cần chỉ số của mảng
Mảng tham số trong C#	Được sử dụng để truyền một số lượng chưa biết của các tham số tới một hàm
Lớp Array trong C#	Được định nghĩa trong System namespace, nó là lớp cơ sở cho tất cả mảng cung cấp các thuộc tính và phương thức để làm việc với mảng

4. CHUỖI (STRING)

- Trong C#, chúng ta có thể sử dụng các chuỗi (string) như là mảng các ký tự.
- Tuy nhiên, phổ biến hơn là **sử dụng từ khóa string để khai báo một biến chuỗi**.
- Tạo một đối tượng String trong C# bởi sử dụng một trong các phương thức sau:
 1. Bằng việc gán một hằng chuỗi cho một biến String.
 2. Sử dụng một constructor của lớp String.
 3. Sử dụng toán tử nối chuỗi (+).
 4. Bởi việc thu nhận một thuộc tính hoặc gọi một phương thức mà trả về một chuỗi.
 5. Bằng việc gọi một phương thức định dạng để chuyển đổi một giá trị hoặc một đối tượng thành biểu diễn chuỗi của nó.

4. CHUỖI (STRING)

- Ví dụ sau minh họa các phương thức để tạo một chuỗi trong C#:

```
static void Main(string[] args)
{
    Console.WriteLine("Cac cach tao chuoi trong C#");
    Console.WriteLine("-----");

    //su dung phep gan hang chuoi va toan tu noi chuoi
    string fname, lname;
    fname = "Nguyen Thi Anh";
    lname = "Thu";

    string fullname = fname + " " + lname;
    Console.WriteLine("Ho va ten: {0}", fullname);

    //su dung constructor cua lop string
    char[] letters = { 'H', 'e', 'l', 'l', 'o' };
    string greetings = new string(letters);
    Console.WriteLine("\nLoi chao bang tieng Anh: {0}", greetings);

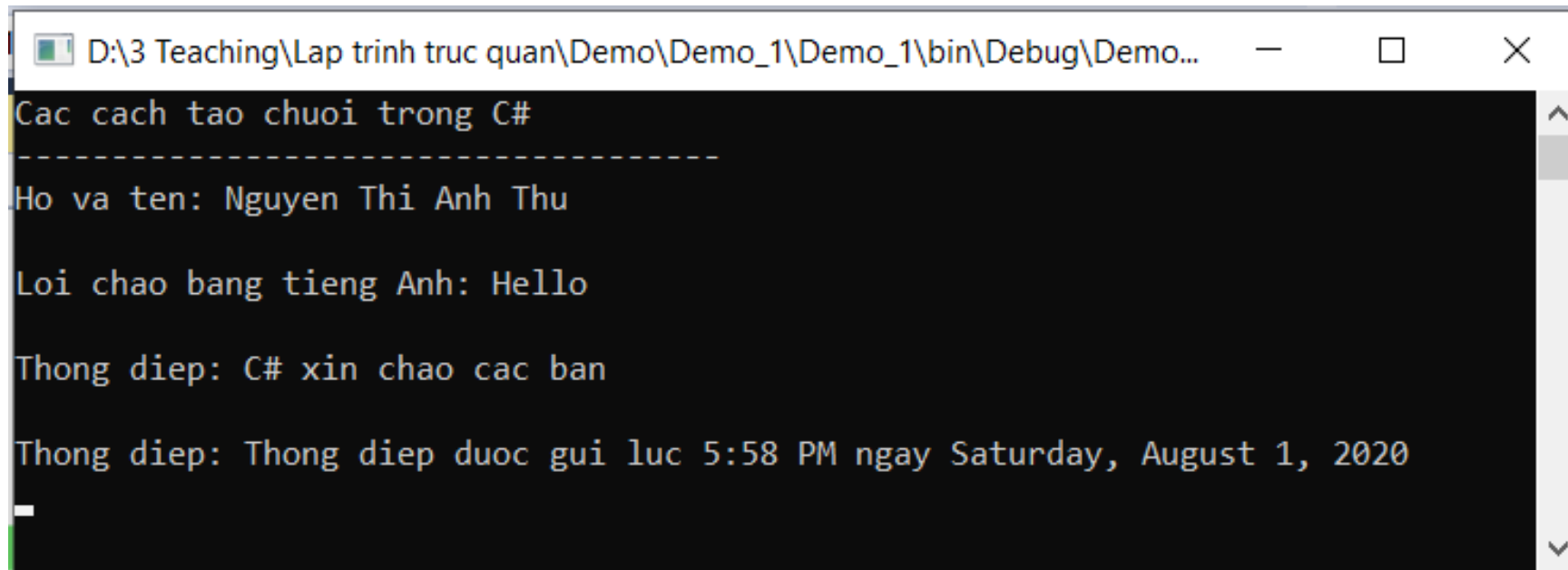
    //tu cac phuong thuc ma tra ve mot chuoi
    string[] sarray = { "C#", "xin", "chao", "cac", "ban" };
    string message = String.Join(" ", sarray);
    Console.WriteLine("\nThong diep: {0}", message);

    //dinh dang phuong thuc de chuyen doi mot gia tri
    DateTime waiting = new DateTime(2020, 8, 1, 17, 58, 1);
    string chat = String.Format("Thong diep duoc gui luc {0:t} ngay {0:D}", waiting);
    Console.WriteLine("\nThong diep: {0}", chat);

    Console.ReadKey();
}
```

4. CHUỖI (STRING)

- Ví dụ sau minh họa các phương thức để tạo một chuỗi trong C#:



```
D:\3 Teaching\Lap trinh truc quan\Demo\Demo_1\Demo_1\bin\Debug\Demo...
Cac cach tao chuoai trong C#
-----
Ho va ten: Nguyen Thi Anh Thu

Loi chao bang tieng Anh: Hello

Thong diep: C# xin chao cac ban

Thong diep: Thong diep duoc gui luc 5:58 PM ngay Saturday, August 1, 2020
```

4. CHUỖI (STRING)

Các thuộc tính của lớp String trong C#

- Lớp String trong C# có hai thuộc tính:

STT	Thuộc tính
1	Chars Lấy đối tượng <i>Char</i> tại một vị trí cụ thể trong đối tượng <i>String</i> hiện tại
2	Length Lấy số ký tự của đối tượng String hiện tại

4. CHUỖI (STRING)

Phương thức của lớp String trong C#

- Lớp String có một số phương thức mà hữu ích trong khi làm việc với các đối tượng String trong C#. Bảng dưới đây liệt kê các phương thức được sử dụng phổ biến nhất:

STT	Phương thức
1	public static int Compare(string strA, string strB) So sánh hai đối tượng String cụ thể và trả về một integer mà chỉ vị trí có liên quan của chúng trong thứ tự sắp xếp
2	public bool Contains(string value) Trả về một giá trị chỉ dẫn có hay không đối tượng String đã cho xuất hiện bên trong chuỗi này
3	public static string Join(string separator, params string[] value) Nối chuỗi tất cả phần tử của một mảng chuỗi, bởi sử dụng Separator (bộ tách) đã cho giữa mỗi phần tử

4. CHUỖI (STRING)

- Ví dụ: So sánh chuỗi trong C#

```
static void Main(string[] args)
{
    Console.WriteLine("So sanh chuoi trong C#");
    Console.WriteLine("-----");

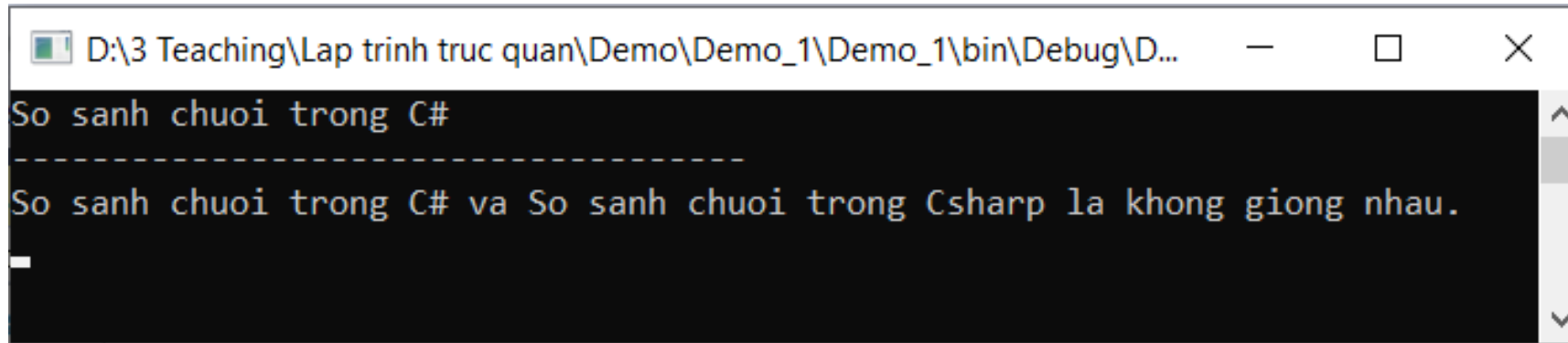
    string str1 = "So sanh chuoi trong C#";
    string str2 = "So sanh chuoi trong Csharp";

    if (String.Compare(str1, str2) == 0)
    {
        Console.WriteLine(str1 + " va " + str2 + " la giong nhau.");
    }
    else
    {
        Console.WriteLine(str1 + " va " + str2 + " la khong giong nhau.");
    }

    Console.ReadKey();
}
```

4. CHUỖI (STRING)

- Ví dụ: So sánh chuỗi trong C#



```
D:\3 Teaching\Lap trình trực quan\Demo\Demo_1\Demo_1\bin\Debug\D...
So sanh chuoi trong C#
-----
So sanh chuoi trong C# va So sanh chuoi trong Csharp la khong giiong nhau.
_
```

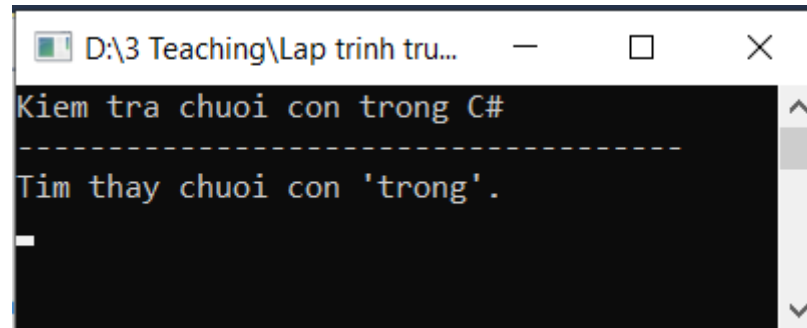
4. CHUỖI (STRING)

- Ví dụ: Kiểm tra chuỗi con trong C#

```
static void Main(string[] args)
{
    Console.WriteLine("Kiem tra chuoi con trong C#");
    Console.WriteLine("-----");

    string str = "Chuoi con trong C#";
    if (str.Contains("trong"))
    {
        Console.WriteLine("Tim thay chuoi con 'trong'.");
    }

    Console.ReadKey();
}
```



The screenshot shows a console window titled "D:\3 Teaching\Lap trinh tru...". The output of the program is displayed in a monospaced font on a black background. The first line is "Kiem tra chuoi con trong C#", followed by a dashed line "-----". The third line is "Tim thay chuoi con 'trong'.". A white cursor is visible on the line following the output.

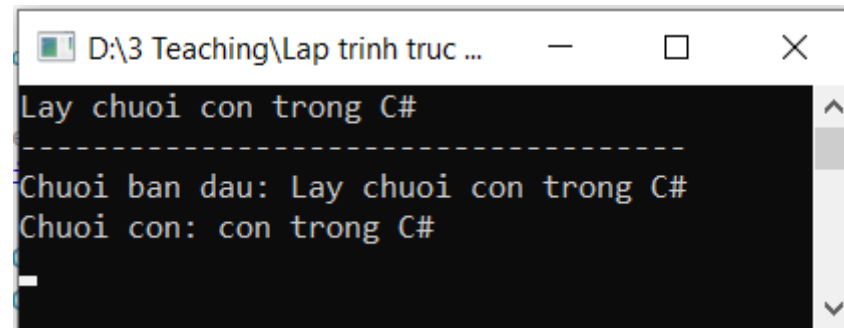
4. CHUỖI (STRING)

- Ví dụ: Lấy chuỗi con trong C#

```
static void Main(string[] args)
{
    Console.WriteLine("Lay chuoi con trong C#");
    Console.WriteLine("-----");

    string str = "Lay chuoi con trong C#";
    Console.WriteLine("Chuoi ban dau: " + str);
    string substr = str.Substring(10);
    Console.WriteLine("Chuoi con: " + substr);

    Console.ReadKey();
}
```



The screenshot shows a console window titled "D:\3 Teaching\Lap trinh truc ...". The output of the program is displayed as follows:

```
Lay chuoi con trong C#
-----
Chuoi ban dau: Lay chuoi con trong C#
Chuoi con: con trong C#
```

4. CHUỖI (STRING)

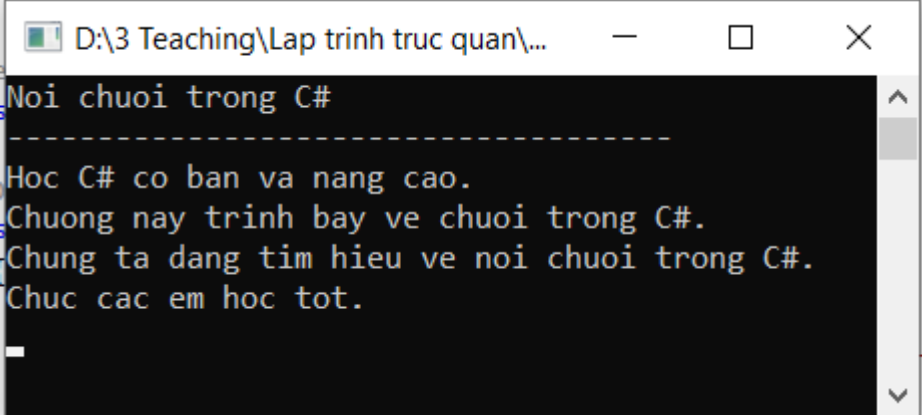
■ Ví dụ: Nối chuỗi trong C#

```
static void Main(string[] args)
{
    Console.WriteLine("Noi chuoi trong C#");
    Console.WriteLine("-----");

    string[] starray = new string[]{"Hoc C# co ban va nang cao.",
    "Chuong nay trinh bay ve chuoi trong C#.",
    "Chung ta dang tim hieu ve noi chuoi trong C#.",
    "Chuc cac em hoc tot."};

    string str = String.Join("\n", starray);
    Console.WriteLine(str);

    Console.ReadKey();
}
```



The screenshot shows a console window titled "D:\3 Teaching\Lap trinh truc quan\...". The output of the program is displayed as follows:

```
Noi chuoi trong C#
-----
Hoc C# co ban va nang cao.
Chuong nay trinh bay ve chuoi trong C#.
Chung ta dang tim hieu ve noi chuoi trong C#.
Chuc cac em hoc tot.
```

5. STRUCTURE

- Trong C#, một cấu trúc (structure) là một kiểu dữ liệu. Nó tạo một biến đơn mà giữ dữ liệu liên quan của các kiểu dữ liệu đa dạng. Từ khóa **struct** trong C# được sử dụng để tạo một cấu trúc (structure).
- Các cấu trúc được sử dụng để biểu diễn một bản ghi (record). Giả sử, chúng ta muốn theo dõi các cuốn sách trong một thư viện. Chúng ta có thể muốn theo dõi các thuộc tính sau của mỗi cuốn sách:
 - Tên sách
 - Tác giả
 - Thể loại
 - ID (mã sách)

5. STRUCTURE

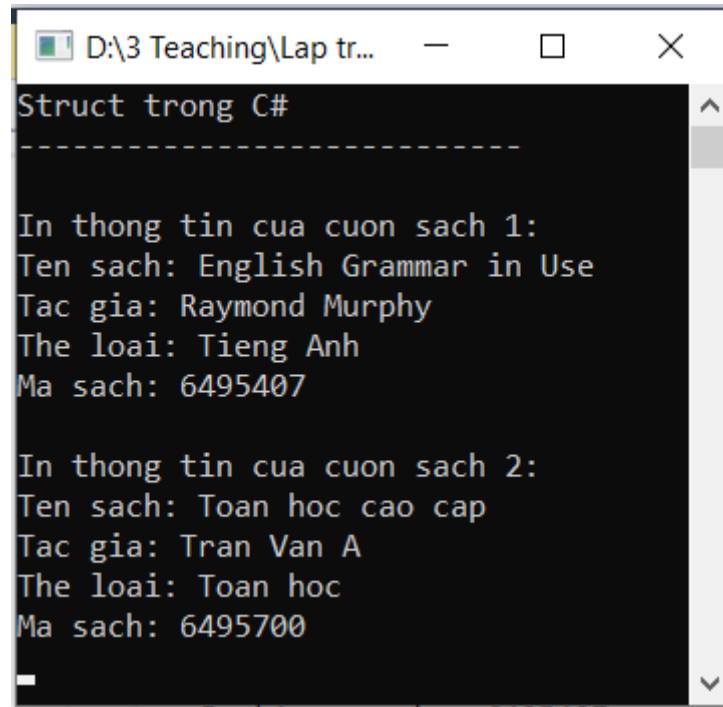
Định nghĩa cấu trúc trong C#

- Để định nghĩa cấu trúc, chúng ta phải sử dụng lệnh **struct**. Câu lệnh struct định nghĩa một kiểu dữ liệu mới, với hơn một thành viên trong chương trình.
- Ví dụ dưới đây là cách khai báo cấu trúc Book:

```
struct Book
{
    public string ten_sach;
    public string tac_gia;
    public string the_loai;
    public int ma_sach;
};
```

5. STRUCTURE

- Cách sử dụng cấu trúc trong class:



```
Struct trong C#
-----

In thông tin của cuốn sách 1:
Ten sach: English Grammar in Use
Tac gia: Raymond Murphy
The loai: Tieng Anh
Ma sach: 6495407

In thông tin của cuốn sách 2:
Ten sach: Toan hoc cao cap
Tac gia: Tran Van A
The loai: Toan hoc
Ma sach: 6495700
```

```
static void Main(string[] args)
{
    Console.WriteLine("Struct trong C#");
    Console.WriteLine("-----\n");

    Book Book1;    /* khai bao Book1 thuoc kieu cau truc Book */
    Book Book2;    /* khai bao Book2 thuoc kieu cau truc Book */

    /* thông tin chi tiết về Book1 */
    Book1.ten_sach = "English Grammar in Use";
    Book1.tac_gia = "Raymond Murphy";
    Book1.the_loai = "Tieng Anh";
    Book1.ma_sach = 6495407;
    /* thông tin chi tiết về Book2 */
    Book2.ten_sach = "Toan hoc cao cap";
    Book2.tac_gia = "Tran Van A";
    Book2.the_loai = "Toan hoc";
    Book2.ma_sach = 6495700;

    /* in các thông tin của Book1 */
    Console.WriteLine("In thông tin của cuốn sách 1:");
    Console.WriteLine("Ten sach: {0}", Book1.ten_sach);
    Console.WriteLine("Tac gia: {0}", Book1.tac_gia);
    Console.WriteLine("The loai: {0}", Book1.the_loai);
    Console.WriteLine("Ma sach: {0}", Book1.ma_sach);
    /* in các thông tin của Book2 */
    Console.WriteLine("\nIn thông tin của cuốn sách 2:");
    Console.WriteLine("Ten sach: {0}", Book2.ten_sach);
    Console.WriteLine("Tac gia: {0}", Book2.tac_gia);
    Console.WriteLine("The loai: {0}", Book2.the_loai);
    Console.WriteLine("Ma sach: {0}", Book2.ma_sach);

    Console.ReadKey();
}
```

5. STRUCTURE

Đặc điểm của cấu trúc trong C#

- Cấu trúc có thể có các phương thức, các trường, indexer, thuộc tính, phương thức operator và sự kiện.
- Cấu trúc có thể có các constructor đã được định nghĩa, nhưng không có destructor. Tuy nhiên, chúng ta **không thể định nghĩa một constructor mặc định cho một cấu trúc**. Constructor mặc định được định nghĩa tự động và không thể bị thay đổi.
- Không giống các Lớp, **cấu trúc không thể kế thừa từ cấu trúc hoặc lớp khác**.
- Cấu trúc không thể được sử dụng như là một cơ sở cho cấu trúc hoặc lớp khác.

5. STRUCTURE

Đặc điểm của cấu trúc trong C#

- Một cấu trúc có thể triển khai một hoặc nhiều Interface.
- Thành viên cấu trúc không thể được xác định ở dạng abstract, virtual hoặc protected.
- Khi chúng ta tạo một đối tượng Struct bởi sử dụng toán tử **new**, nó lấy đối tượng đã tạo và constructor thích hợp được gọi. Không giống Lớp, **cấu trúc có thể được khởi tạo mà không cần sử dụng toán tử new**.
- Nếu toán tử new không được sử dụng, thì các trường chưa được gán và đối tượng không thể được sử dụng tới khi tất cả trường đó được khởi tạo.

5. STRUCTURE

Phân biệt Class và Structure trong C#

- Các Lớp là các kiểu tham chiếu, còn cấu trúc là các kiểu giá trị.
- Cấu trúc không hỗ trợ tính kế thừa.
- Cấu trúc không cho phép định nghĩa constructor mặc định.

5. STRUCTURE

- Từ các điểm trên, chúng ta viết lại ví dụ trên:

```
struct Book
{
    private string ten_sach;
    private string tac_gia;
    private string the_loai;
    private int ma_sach;
    2 references
    public void nhapGiaTri(string t, string a, string s, int id)
    {
        ten_sach = t;
        tac_gia = a;
        the_loai = s;
        ma_sach = id;
    }
    2 references
    public void display()
    {
        Console.WriteLine("Tieu de: {0}", ten_sach);
        Console.WriteLine("Tac gia: {0}", tac_gia);
        Console.WriteLine("The loai: {0}", the_loai);
        Console.WriteLine("Ma sach: {0}", ma_sach);
    }
};
```

5. STRUCTURE

- Từ các điểm trên, chúng ta viết lại ví dụ trên:

```
static void Main(string[] args)
{
    Console.WriteLine("Struct trong C#");
    Console.WriteLine("-----\n");
    Book Book1 = new Book(); /* Khai bao Book1 thuoc kieu cau truc Book */
    Book Book2 = new Book(); /* Khai bao Book2 thuoc kieu cau truc Book */

    /* thong tin Book1 */
    Book1.nhapGiaTri("English Grammer in Use",
        "Raymond Murphy", "Tieng Anh", 6495407);

    /* thong tin book2 */
    Book2.nhapGiaTri("Toan hoc cao cap",
        "Tran Van A", "Toan hoc", 6495700);

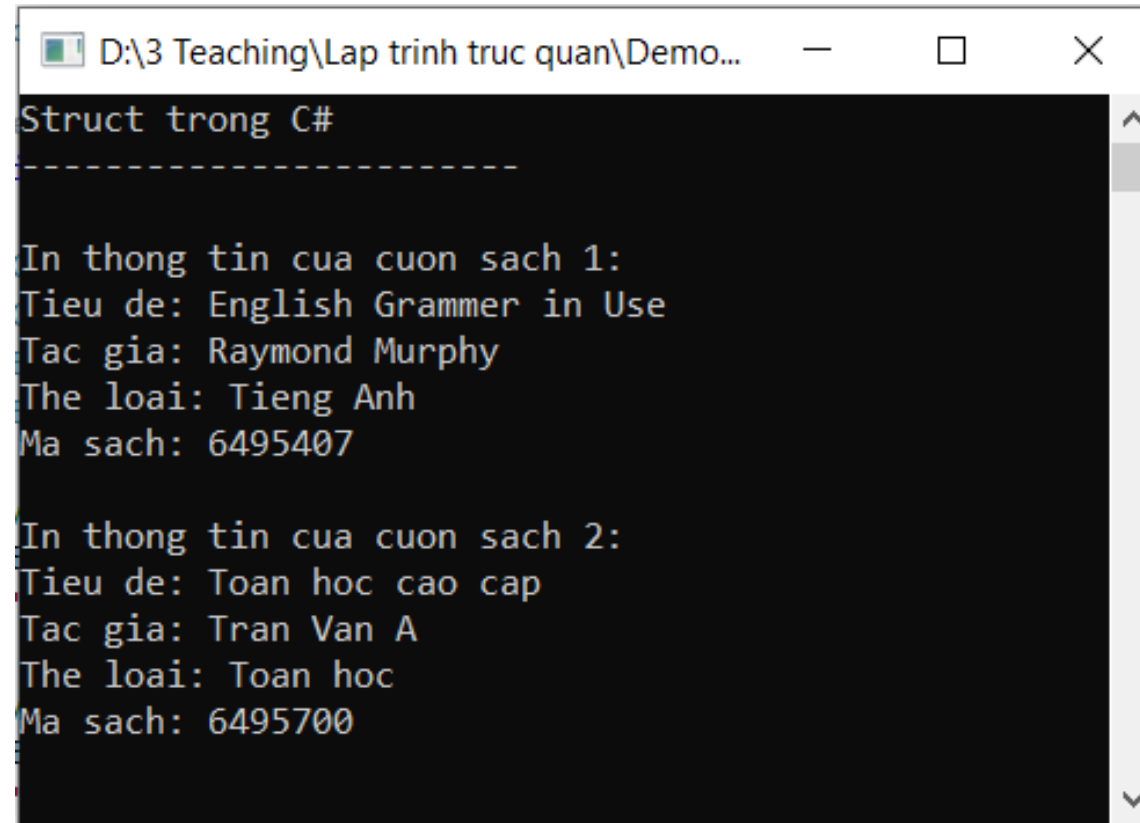
    /* In thong tin Book1 */
    Console.WriteLine("In thong tin cua cuon sach 1:");
    Book1.display();

    /* In thong tin Book2 */
    Console.WriteLine("\nIn thong tin cua cuon sach 2:");
    Book2.display();

    Console.ReadKey();
}
```

5. STRUCTURE

- Từ các điểm trên, chúng ta viết lại ví dụ trên:



```
Struct trong C#
-----

In thông tin của cuốn sách 1:
Tieu de: English Grammer in Use
Tac gia: Raymond Murphy
The loai: Tiếng Anh
Ma sach: 6495407

In thông tin của cuốn sách 2:
Tieu de: Toán học cao cấp
Tac gia: Trần Văn A
The loai: Toán học
Ma sach: 6495700
```

6. LỚP (CLASS)

- Khi định nghĩa một lớp (class) trong C#, nghĩa là ta **định nghĩa một blueprint cho một kiểu dữ liệu**.
- Điều này không thực sự định nghĩa bất kỳ dữ liệu nào, nhưng nó định nghĩa ý nghĩa của tên lớp đó.
- Tức là, một đối tượng của lớp đó gồm những cái gì, các hoạt động nào có thể được thực hiện trên đối tượng đó.
- Các đối tượng là instance (sự thể hiện) của một lớp. Các phương thức và các biến mà cấu tạo nên một lớp được gọi là các thành viên của lớp đó.

6. LỚP (CLASS)

Định nghĩa một Class trong C#

- Một định nghĩa lớp trong C# bắt đầu với từ khóa **class** được theo sau bởi tên lớp và phần thân lớp được bao quanh bởi các dấu ngoặc ôm.

```
<access specifier> class tên_lớp
{
    // các biến thành viên
    <access specifier> <kiểu_dữ_liệu> biến1;
    <access specifier> <kiểu_dữ_liệu> biến2;
    ...
    <access specifier> <kiểu_dữ_liệu> biếnN;

    // các phương thức thành viên
    <access specifier> <kiểu_trả_về> tên_phương_thức1(danh_sách_tham_số)
    {
        // phần thân phương thức
    }
    <access specifier> <kiểu_trả_về> tên_phương_thức2(danh_sách_tham_số)
    {
        // phần thân phương thức
    }
    ...
    <access specifier> <kiểu_trả_về> tên_phương_thứcN(danh_sách_tham_số)
    {
        // phần thân phương thức
    }
}
```

6. LỚP (CLASS)

Ghi chú:

- *Access specifier* xác định các qui tắc truy cập cho các thành viên cũng như chính lớp đó. Nếu không được đề cập, thì Access Specifier mặc định cho một kiểu lớp là **internal**. Chế độ truy cập mặc định cho các thành viên là **private**.
- *kiểu_dữ_liệu* xác định kiểu biến và trả về kiểu dữ liệu mà phương thức trả về.
- Để truy cập các thành viên lớp, chúng ta sử dụng toán tử **dot (.)**.
- Toán tử dot (.) liên kết với tên của một đối tượng với tên của một thành viên.

6. LỚP (CLASS)

Ví dụ:

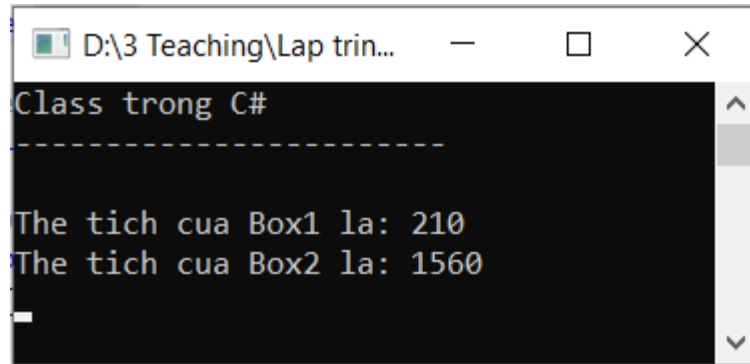
- Tạo hai class có tên lần lượt là **Box** và **TestCsharp** trong hai file riêng biệt.
- Lớp **Box**: chứa các thuộc tính của một hộp.

```
class Box
{
    public double chieu_dai;
    public double chieu_rong;
    public double chieu_cao;
}
```

6. LỚP (CLASS)

Ví dụ:

- Lớp **TestCsharp**: chứa phương thức **main()** để thao tác trên đối tượng **Box**.



```
D:\3 Teaching\Lap trin...
Class trong C#
-----
The tích của Box1 là: 210
The tích của Box2 là: 1560
```

```
public class TestCsharp
{
    0 references
    public static void Main(string[] args)
    {
        Console.WriteLine("Class trong C#");
        Console.WriteLine("-----\n");

        Box Box1 = new Box();    // tao doi tuong Box1
        Box Box2 = new Box();    // tao doi tuong Box2
        double the_tich = 0.0;    // the tích của box

        // thông tin của box1
        Box1.chieu_cao = 5.0;
        Box1.chieu_dai = 6.0;
        Box1.chieu_rong = 7.0;

        // thông tin của box2
        Box2.chieu_cao = 10.0;
        Box2.chieu_dai = 12.0;
        Box2.chieu_rong = 13.0;

        // Tính và in the tích của box1
        the_tich = Box1.chieu_cao * Box1.chieu_dai * Box1.chieu_rong;
        Console.WriteLine("The tích của Box1 là: {0}", the_tich);

        // Tính và in the tích của box2
        the_tich = Box2.chieu_cao * Box2.chieu_dai * Box2.chieu_rong;
        Console.WriteLine("The tích của Box2 là: {0}", the_tich);

        Console.ReadKey();
    }
}
```


6. LỚP (CLASS)

Hàm thành viên và tính đóng gói trong C#

- Một hàm thành viên trong C# của một lớp là một hàm mà có định nghĩa và nguyên mẫu (prototype) của nó bên trong định nghĩa lớp tương tự như bất kỳ biến nào khác. Nó hoạt động trên bất kỳ đối tượng nào của lớp mà nó là thành viên và có truy cập tới tất cả thành viên của một lớp cho đối tượng đó.
- **Các biến thành viên là các thuộc tính của một đối tượng (từ bối cảnh thiết kế) và chúng được giữ private để triển khai tính đóng gói.** Những biến này chỉ có thể được truy cập bởi sử dụng các hàm thành viên public.

6. LỚP (CLASS)

Ví dụ:

- Tạo hai lớp có tên lần lượt là **Box** và **TestCsharp** trong hai file riêng biệt.
- Lớp **Box**: chứa các thuộc tính và phương thức thành viên.

```
class Box
{
    private double chieu_dai;
    private double chieu_rong;
    private double chieu_cao;
    2 references
    public void setChieuDai(double len)
    {
        chieu_dai = len;
    }

    2 references
    public void setChieuRong(double bre)
    {
        chieu_rong = bre;
    }

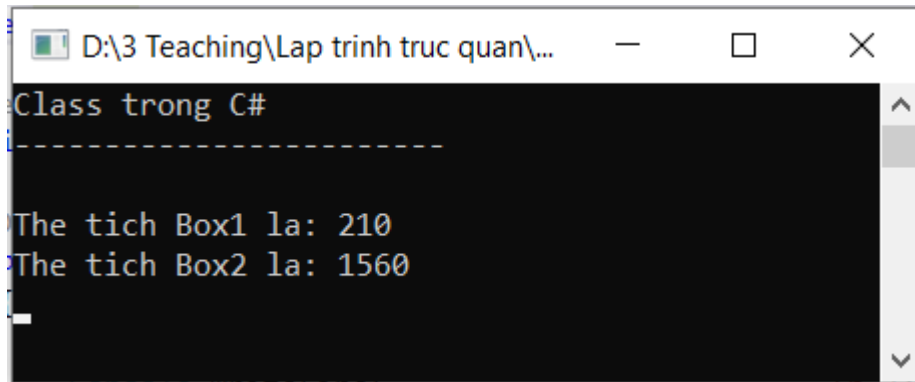
    2 references
    public void setChieuCao(double hei)
    {
        chieu_cao = hei;
    }

    2 references
    public double tinhTheTich()
    {
        return chieu_dai * chieu_rong * chieu_cao;
    }
}
```

6. LỚP (CLASS)

Ví dụ:

- Lớp **TestCsharp**: chứa phương thức **main()** để thao tác trên đối tượng **Box**.



```
D:\3 Teaching\Lap trinh truc quan\...
Class trong C#
-----
The tich Box1 la: 210
The tich Box2 la: 1560
```

```
public class TestCsharp
{
    0 references
    public static void Main(string[] args)
    {
        Console.WriteLine("Class trong C#");
        Console.WriteLine("-----\n");

        Box Box1 = new Box();    // tao doi tuong Box1
        Box Box2 = new Box();    // tao doi tuong Box2
        double the_tich;

        // nhap thong tin cho Box1
        Box1.setChieuDai(6.0);
        Box1.setChieuRong(7.0);
        Box1.setChieuCao(5.0);

        // nhap thong tin cho Box2
        Box2.setChieuDai(12.0);
        Box2.setChieuRong(13.0);
        Box2.setChieuCao(10.0);

        // tinh va in the tich Box1
        the_tich = Box1.tinhTheTich();
        Console.WriteLine("The tich Box1 la: {0}", the_tich);

        // tinh va in the tich Box2
        the_tich = Box2.tinhTheTich();
        Console.WriteLine("The tich Box2 la: {0}", the_tich);

        Console.ReadKey();
    }
}
```

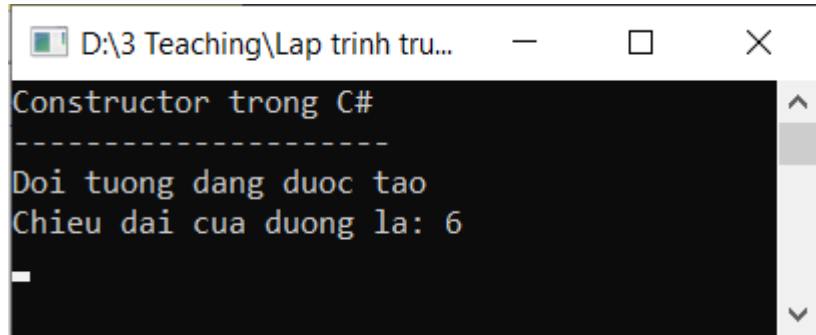
6. LỚP (CLASS)

Constructor trong C#

- Một constructor của một Class, là một hàm thành viên đặc biệt của một lớp, được thực thi bất cứ khi nào chúng ta tạo các đối tượng mới của lớp đó.
- **Một constructor có tên giống như tên lớp và nó không có bất kỳ kiểu trả về nào.**

6. LỚP (CLASS)

■ Ví dụ:



```
Constructor trong C#
-----
Doi tuong dang duoc tao
Chieu dai cua duong la: 6
```

```
class Line
{
    private double chieu_dai;
    1 reference
    public Line()
    {
        Console.WriteLine("Doi tuong dang duoc tao");
    }

    1 reference
    public void setChieuDai(double len)
    {
        chieu_dai = len;
    }

    1 reference
    public double getChieuDai()
    {
        return chieu_dai;
    }

    0 references
    static void Main(string[] args)
    {
        Console.WriteLine("Constructor trong C#");
        Console.WriteLine("-----");
        //tao doi tuong Line bang constructor
        Line line = new Line();

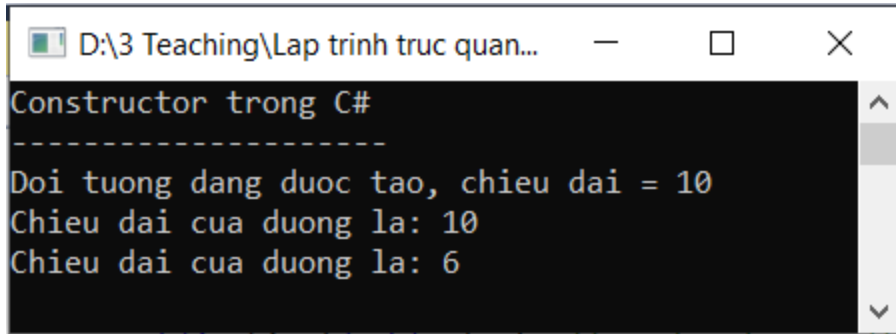
        // thiet lap chieu dai cho duong
        line.setChieuDai(6.0);
        Console.WriteLine("Chieu dai cua duong la: {0}", line.getChieuDai());
        Console.ReadKey();
    }
}
```

6. LỚP (CLASS)

- Một constructor mặc định trong C# không có bất kỳ tham số nào, nhưng nếu cần, một constructor có thể có tham số.
- **Những constructor này được gọi là constructor được tham số hóa.**
- **Kỹ thuật này giúp chúng ta gán giá trị khởi đầu cho một đối tượng tại thời điểm tạo ra nó.**

6. LỚP (CLASS)

■ Ví dụ:



```
Constructor trong C#
-----
Doi tuong dang duoc tao, chieu dai = 10
Chieu dai cua duong la: 10
Chieu dai cua duong la: 6
```

```
class Line
{
    private double chieu_dai;
    1 reference
    public Line(double len) //constructor co tham so
    {
        Console.WriteLine("Doi tuong dang duoc tao, chieu dai = {0}", len);
        chieu_dai = len;
    }
    1 reference
    public void setChieuDai(double len)
    {
        chieu_dai = len;
    }

    2 references
    public double getChieuDai()
    {
        return chieu_dai;
    }

    0 references
    static void Main(string[] args)
    {
        Console.WriteLine("Constructor trong C#");
        Console.WriteLine("-----");
        //tao doi tuong Line bang constructor
        Line line = new Line(10.0);
        Console.WriteLine("Chieu dai cua duong la: {0}", line.getChieuDai());

        // thiet lap chieu dai cho duong
        line.setChieuDai(6.0);
        Console.WriteLine("Chieu dai cua duong la: {0}", line.getChieuDai());
        Console.ReadKey();
    }
}
```

6. LỚP (CLASS)

Destructor trong C#

- Một destructor trong C# là một hàm thành viên đặc biệt của một lớp, được thực thi bất cứ khi nào một đối tượng của lớp đó thoát ra khỏi phạm vi.
- Một destructor có tên giống tên lớp với một dấu ngã (~) ở trước và nó có thể: không trả về một giá trị hoặc không nhận bất kỳ tham số nào.
- Destructor trong C# có thể rất hữu ích để giải phóng tài nguyên bộ nhớ trước khi thoát khỏi chương trình. Destructor không thể bị kế thừa hoặc nạp chồng.

6. LỚP (CLASS)

■ Ví dụ:

```
class Line
{
    private double chieu_dai;
    1 reference
    public Line() //constructor
    {
        Console.WriteLine("Doi tuong dang duoc tao.");
    }

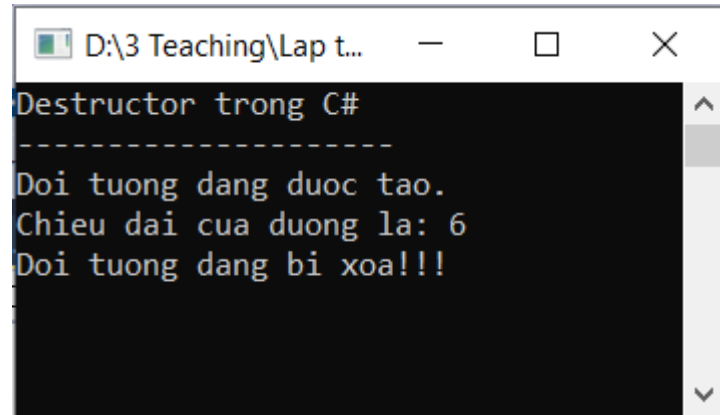
    1 reference
    public void setChieuDai(double len)
    {
        chieu_dai = len;
    }

    1 reference
    public double getChieuDai()
    {
        return chieu_dai;
    }

    0 references
    ~Line() // destructor
    {
        Console.WriteLine("Doi tuong dang bi xoa!!!");
    }
}
```

6. LỚP (CLASS)

■ Ví dụ:



```
Destructor trong C#
-----
Doi tuong dang duoc tao.
Chieu dai cua duong la: 6
Doi tuong dang bi xoa!!!
```

```
class Program
{
    1 reference
    public static void Details()
    {
        //tao doi tuong Line bang constructor
        Line line = new Line();
        // thiet lap chieu dai cho duong
        line.setChieuDai(6.0);
        Console.WriteLine("Chieu dai cua duong la: {0}", line.getChieuDai());
    }
    0 references
    static void Main(string[] args)
    {
        Console.WriteLine("Destructor trong C#");
        Console.WriteLine("-----");
        Details();
        GC.Collect();//don dep bo nho
        Console.ReadKey();
    }
}
```

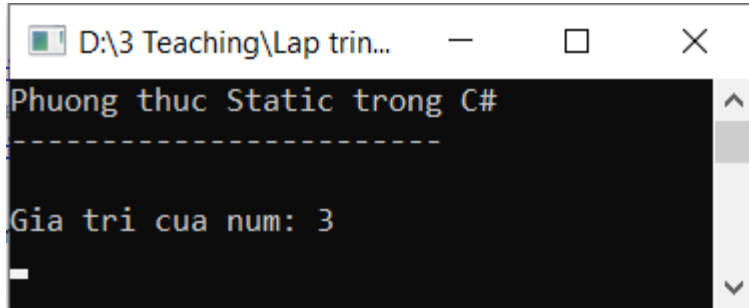
6. LỚP (CLASS)

Thành viên Static của một Class trong C#

- Chúng ta có thể định nghĩa các thành viên lớp là **static** bởi sử dụng từ khóa **static** trong C#.
- Từ khóa **static** ngụ ý rằng chỉ có một **instance** (sự thể hiện) của thành viên tồn tại cho một lớp đó.
- Các biến **static** được sử dụng để định nghĩa các hằng số (**constant**) bởi vì giá trị của chúng có thể được thu nhận bằng việc gọi lớp đó mà không cần tạo một instance của nó.
- Các biến **static** có thể được khởi tạo bên ngoài hàm thành viên hoặc định nghĩa lớp.
- Có thể khai báo một hàm thành viên là **static**. Những hàm này chỉ có thể truy cập các biến **static**. Hàm **static** có thể tồn tại trước cả khi đối tượng được tạo.

6. LỚP (CLASS)

■ Ví dụ:



```
Phuong thuc Static trong C#
-----
Gia tri cua num: 3
```

```
class ThanhVienStatic
{
    public static int num; // thanh vien static
    3 references
    public void count()
    {
        num++;
    }
    //phuong thuc static
    1 reference
    public static int getNum()
    {
        return num;
    }
}
0 references
public class TestCsharp
{
    0 references
    public static void Main(string[] args)
    {
        Console.WriteLine("Phuong thuc Static trong C#");
        Console.WriteLine("-----\n");

        //tao cac doi tuong ThanhVienStatic
        ThanhVienStatic s = new ThanhVienStatic();
        //goi phuong thuc
        s.count();
        s.count();
        s.count();
        Console.WriteLine("Gia tri cua num: {0}", ThanhVienStatic.getNum());

        Console.ReadKey();
    }
}
```

BÀI TẬP

- Xây dựng lớp phân số với:
 - Hai thuộc tính riêng xác định tử số và mẫu số của phân số.
 - Các phương thức:
 - Các phép toán cộng, trừ, nhân, chia các phân số.
 - Phép kiểm tra một phân số có phải tối giản hay không.
 - Phép tìm dạng tối giản của phân số.
- Viết chương trình ứng dụng thực hiện việc nhập vào một dãy các phân số và cho phép người dùng chọn thực hiện các phương thức trên.



Q & A