6. Mảng một chiều

Follow me:

GitHub Logo **GitHub** Codeforces Logo **Codeforces** LinkedIn Logo **LinkedIn**

December, 2024

Giới thiệu

Có kiểu dữ liệu **tham chiếu** với **3 loại**:

Mảng một chiều:

```
int[] numbers = { 1, 2, 3, 4, 5 };
Console.WriteLine(numbers[0]); // Outputs: 1
```

Mảng đa chiều: Hình dung như là một ma trận

```
int[,] matrix = new int[2, 3] { { 1, 2, 3 }, { 4, 5, 6 } };
Console.WriteLine(matrix[0, 0]); // Outputs: 1
```

Jagged Arrays (lười dịch): Mảng răng cưa là các mảng của mảng. Mỗi phần tử của một mảng răng cưa tự nó là một mảng.

```
int[][] jaggedArray = new int[3][];
jaggedArray[0] = new int[] { 1, 2 };
jaggedArray[1] = new int[] { 3, 4, 5 };
jaggedArray[2] = new int[] { 6, 7, 8, 9 };
Console.WriteLine(jaggedArray[0][1]); // Outputs: 2
```

Truy cập các phần tử của mảng

Bạn có thể truy cập bằng index. Số thứ tự index bắt đầu từ 0, các ngôn ngữ cấp thấp như `c` hoặc `Pascal` sẽ bắt đầu từ 1

```
string[] cars = { "Volvo", "BMW", "Ford", "Mazda" };
Console.WriteLine(cars[0]); // Outputs: Volvo
Array Length
```

Cách tìm ra độ dài của mảng `.Length` :

```
string[] cars = { "Volvo", "BMW", "Ford", "Mazda" };
Console.WriteLine(cars.Length); // Outputs: 4
```

Với **mảng đa chiều** thì sử dụng `.GetLength(number)` sẽ được học thêm ở buổi sau.

Nhập mảng một chiều

Sử dụng vòng lặp để nhập mảng

```
for (int i = 0; i < arr.Length; i++)
 {
   Console.Write($"arr[{i}] = ");
   arr[i] = int.Parse(Console.ReadLine());
 }
```

Xuất mảng một chiều

```
 for (int i = 0; i < arr.Length; i++)
 {
   Console.Write(arr[i] + " ");
 }
 Console.WriteLine());
```

Truyền tham trị

Vì lười nhập ảnh nên đây là code ví dụ của ChatGPT:

```
using System;

class Program
{
    static void Change(int[] a)
    {
        a[0] = 9; // Thay đổi giá trị phần tử đầu tiên của mảng
        Console.WriteLine($"Inside method: a[0] = {a[0]}");
    }

    static void Main(string[] args)
    {
        // Khởi tạo mảng một chiều
        int[] a = new int[4] { 5, 8, 2, 7 };

        Console.WriteLine($"Before calling method: a[0] = {a[0]}");
```

```
        // Gọi hàm Change
        Change(a);

        Console.WriteLine($"After calling method: a[0] = {a[0]}");
    }
}
```

Kết quả đầu ra:

```
Before calling method: a[0] = 5
Inside method: a[0] = 9
After calling method: a[0] = 9
```

Giải thích:

- Trước khi gọi phương thức `Change`, giá trị phần tử đầu tiên của mảng là `5`.
- Trong phương thức `Change`, phần tử đầu tiên (`a[0]`) được thay đổi thành `9`. Vì mảng là một kiểu tham chiếu, sự thay đổi này áp dụng trực tiếp lên mảng gốc.
- Sau khi gọi phương thức, giá trị phần tử đầu tiên của mảng trong `Main` cũng là `9`. Điều này cho thấy mảng được truyền theo **tham chiếu ngầm định** trong C#.

Nhận xét:

- Đối số **phải là một mảng** đã được khởi tạo;
- Tham số nhận giá trị là **tham chiếu đến vùng giá trị của đối số**-> tham số và đối số cùng tham chiếu đến 1 vùng giá trị.

Một trường hợp khác

```
using System;

class Program
{
    static void Change(int[] a)
    {
        a[0] = 9; // Thay đổi giá trị phần tử đầu tiên của mảng gốc
        a = new int[2] { -3, -7 }; // Gán một mảng mới cho tham số 'a' (chỉ áp dụng trong pl
        Console.WriteLine($"Inside method: a[0] = {a[0]}");
    }

    static void Main(string[] args)
    {
        // Khởi tạo mảng một chiều
```

```
        int[] a = new int[4] { 5, 8, 2, 7 };

        Console.WriteLine($"Before calling method: a[0] = {a[0]}");

        // Gọi phương thức Change
        Change(a);

        Console.WriteLine($"After calling method: a[0] = {a[0]}");
    }
}
```

Kết quả đầu ra:

```
Before calling method: a[0] = 5
Inside method: a[0] = -3
After calling method: a[0] = 9
```

Giải thích:

1. **Trước khi gọi phương thức `Change`:**
   - Phần tử đầu tiên của mảng `a` là `5`.
   - Đây là giá trị ban đầu của mảng.
2. **Trong phương thức `Change`:**
   - `a[0] = 9`: Phần tử đầu tiên của mảng gốc được thay đổi thành `9`. Vì mảng là kiểu tham chiếu, sự thay đổi này ảnh hưởng trực tiếp đến mảng gốc.
   - `a = new int[2] { -3, -7 }`: Gán một mảng mới cho tham số `a`. Điều này chỉ thay đổi tham số cục bộ `a` trong phương thức, không ảnh hưởng đến biến `a` trong `Main`.
3. **Sau khi gọi phương thức `Change`:**
   - Mặc dù trong phương thức, `a` được gán một mảng mới, nhưng tham số `a` cục bộ không ảnh hưởng đến mảng gốc. Do đó, giá trị phần tử đầu tiên của mảng `a` trong `Main` vẫn là `9`.

Nhận xét:

- Đối số **phải là một mảng** đã được khởi tạo;
- Tham số nhận giá trị là **tham chiếu đến vùng giá trị của đối số** -> tham số và đối số cùng tham chiếu đến 1 vùng giá trị;
- Khi có câu lệnh **thay đổi tham chiếu của tham số thì không ảnh hưởng đến đối số.**

Truyền tham chiếu

Dưới đây là chương trình hoàn chỉnh với cách sử dụng từ khóa `ref` trong phương thức `Change`:

```
using System;

class Program
{
    static void Change(ref int[] a)
    {
        a[0] = 9; // Thay đổi giá trị phần tử đầu tiên của mảng
        a = new int[2] { -3, -7 }; // Gán một mảng mới cho tham số 'a' (ảnh hưởng đến mảng g
        Console.WriteLine($"Inside method: a[0] = {a[0]}");
    }

    static void Main(string[] args)
    {
        // Khởi tạo mảng một chiều
        int[] a = new int[4] { 5, 8, 2, 7 };

        Console.WriteLine($"Before calling method: a[0] = {a[0]}");

        // Gọi phương thức Change với từ khóa ref
        Change(ref a);

        Console.WriteLine($"After calling method: a[0] = {a[0]}");
    }
}
```

Kết quả đầu ra:

```
Before calling method: a[0] = 5
Inside method: a[0] = -3
After calling method: a[0] = -3
```

Giải thích:

1. **Trước khi gọi phương thức `Change`:**
   - Mảng `a` được khởi tạo với các giá trị `{5, 8, 2, 7}`, và giá trị `a[0]` là `5`.
2. **Trong phương thức `Change`:**
   - `a[0] = 9`: Thay đổi giá trị phần tử đầu tiên của mảng thành `9`. Vì tham số `a` được truyền theo tham chiếu với từ khóa `ref`, sự thay đổi này sẽ ảnh hưởng đến mảng gốc.
   - `a = new int[2] { -3, -7 }`: Sau khi thay đổi giá trị phần tử đầu tiên, mảng `a` được gán

một mảng mới `{ -3, -7 }`. Với từ khóa `ref`, sự thay đổi này sẽ ảnh hưởng đến biến `a` trong `Main`, làm cho mảng `a` trong `Main` trỏ đến mảng mới.

3. **Sau khi gọi phương thức `Change`:**
    - Sau khi gọi phương thức, mảng `a` trong `Main` đã bị thay đổi thành mảng mới `{ -3, -7 }`. Vì tham số được truyền theo tham chiếu (ref), mảng trong `Main` bị thay thế hoàn toàn bằng mảng mới, và `a[0]` có giá trị `-3`.

Nhận xét:

- Thao tác trên tham số cũng chính là thao tác trên đối số **trong lời gọi hàm**
- Tham chiếu **ref**: đối số **phải là mảng đã được khởi tạo**
- Tham chiếu **out**: đối số **có thể là mảng chưa được khởi tạo**

Kiểu dữ liệu `List<T>`

`List<T>`: có thể hiểu là mảng động.

Cú pháp (Syntax):

- Khởi tạo danh sách rỗng với kích thước mặc định

```
List<int> myList = new List<int>();
```

- Khởi tạo danh sách rỗng với kích thước cụ thể

```
List<int> myList = new List<int>(10);
```

Thuộc tính:

- **Capacity**: trả về tổng số phần tử khi tạo biến.
- **Count**: trả về số phần tử thực tế.

```
List<int> myList1 = new List<int>();
Console.WriteLine(myList1.Count);   // 0
Console.WriteLine(myList1.Capacity); // 0

List<int> myList2 = new List<int>(10);
Console.WriteLine(myList2.Count); //0
Console.WriteLine(myList2.Capacity); //10
```

Một số thao tác cơ bản:

- Add(T): thêm phần tử vào cuối List
- Clear(): xóa toàn bộ phần tử trong List
- ToArray(): sao chép các phần tử từ List sang Array