Bài: Mảng 1 chiều trong C#

Xem bài học trên website để ủng hộ Kteam: Mảng 1 chiều trong C#

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Ở các bài học trước, chúng ta đã cùng nhau tìm hiểu về TỪ KHÓA REF & OUT C#. Hôm nay chúng ta sẽ cùng tìm hiểu về **mảng 1 chiều trong C#.**

Nội dung

Để đọc hiểu bài này tốt nhất các bạn nên có kiến thức cơ bản về các phần:

- CẤU TRÚC CƠ BẢN MỘT CHƯƠNG TRÌNH TRONG C# console application
- BIẾN và KIỂU DỮ LIỆU trong C#
- TOÁN TỬ TRONG C#
- CÂU ĐIỀU KIỆN TRONG C#
- CẤU TRÚC CƠ BẢN VÒNG LẶP
- CẤU TRÚC CƠ BẢN CỦA HÀM TRONG C#

Trong bài học này, chúng ta sẽ cùng tìm hiểu các vấn đề:

- Khái niệm về mảng trong C#. Tại sao phải sử dụng mảng?
- Khai báo, khởi tạo và sử dụng mảng 1 chiều trong C#

Khái niệm về mảng trong C#. Tại sao phải sử dụng mảng?

Khái niệm về mảng

Mảng là

- Tập hợp các đối tượng có cùng kiểu dữ liệu.
- Mỗi đối tượng trong mảng được gọi là một phần tử.
- Các phần tử phân biệt với nhau bằng chỉ số phần tử. Trong C# chỉ số phần tử là các số nguyên không âm và bắt đầu từ 0 1 2 3...

Đặc điểm của mảng:

- Các phần tử trong mảng dùng chung một tên và được truy xuất thông qua chỉ số phần tử.
- Một mảng cần có giới hạn số phần tử mà mảng có thể chứa.
- Phải cấp phát vùng nhớ mới có thể sử dụng mảng.
- Vị trí ô nhớ của các phần tử trong mảng được cấp phát liền kề nhau.

Tại sao phải sử dụng mảng?

Để trả lời câu hỏi này ta thử xét trường hợp chúng ta cần chứa họ tên của 10 sinh viên trong trường.

- Nếu sử dụng biến bình thường thì ta cần khai báo 10 biến kiểu string để chứa họ tên của 10 sinh viên. Đến đây vẫn chưa có vấn đề gì lớn!
- Nhưng thực tế thì một trường không phải chỉ có 10 sinh viên, nó có thể là 1000 sinh viên, 10000 sinh viên, 30000 sinh viên. Lúc này vấn đề đã xuất hiện, chúng ta không thể khai báo vài nghìn biến kiểu string được như vậy code sẽ rất dài dòng, rất khó kiểm soát. Khi đó việc sử dụng mảng để lưu trữ là lựa chọn tốt nhất.

Những **lợi ích** khi sử dụng mảng:

- Gom nhóm các đối tượng có chung tính chất lại với nhau giúp code gọn gàng hơn.
- Để thao tác, dễ quản lý, nâng cấp sửa chữa. Vì lúc này việc thay đổi số lượng sinh viên ta chỉ cần thay đổi số phần tử của mảng là được.
- Dễ dàng áp dụng các cấu trúc lặp vào để xử lý dữ liệu.

Khai báo, khởi tạo và sử dụng mảng 1 chiều trong C#

Khai báo mảng 1 chiều

Cú pháp:

> **<kiểu dữ liệu> [] <tên mảng>**;

Trong đó:

- <kiểu dữ liệu> là kiểu dữ liệu của các phần tử trong mảng.
- Cặp dấu **[]** là ký hiệu cho khai báo mảng 1 chiều.
- <tên mảng> là tên của mảng, cách đặt tên mảng cũng như cách đặt tên biến (quy tắc đặt tên biến đã trình bày trong BIẾN TRONG C#).

Để sử dụng được mảng ta phải khởi tạo giá trị hoặc cấp phát vùng nhớ cho mảng. Cấp phát vùng nhớ:

- Được thực hiện thông qua toán tử new (đã trình bày trong bài TOÁN TỬ TRONG C#).
- Lưu ý là khi cấp phát vùng nhớ cho mảng 1 chiều ta cần chỉ ra số phần tử tối đa của mảng.
- **Ví dụ:**

C#:

```
/*
            * Khai báo mảng 1 chiều kiểu string và có tên là Kteam.
            * Sau đó thực hiện cấp phát vùng nhớ với số phần tử tối đa của mảng là 3.
            */

string[] Kteam = new string[3];
```

- Sau khi mảng được cấp phát vùng nhớ thì các phần tử trong mảng sẽ mang giá trị mặc định:
  - Đối với số nguyên là 0
  - Đối với số thực là 0.0
  - Đối với kiểu ký tự là '' (ký tự rỗng)
  - Đối với kiểu tham chiếu là null
- Chúng ta có thể khởi tạo giá trị khác mà chúng ta mong muốn ngay khi cấp phát vùng nhớ bằng cú pháp sau:

> **<kiểu dữ liệu>**[] **<tên mảng>** = **new** <kiểu dữ liệu>**[]** { <giá trị 1>, ..., <giá trị n> };

- Các giá trị khởi tạo nằm trong cặp dấu ngoặc ngọn {} và cách nhau bởi dấu phẩy.
- Chúng ta không cần cung cấp số phần tử tối đa mà trình biên dịch sẽ tự đếm xem bạn đã khởi tạo bao nhiêu giá trị và xem nó như số phần tử tối đa. Vì thế dù việc khai báo số phần tử tối đa không lỗi nhưng trong trường hợp này nó không có ý nghĩa lắm!

Khởi tạo giá trị

Cú pháp:

> **<kiểu dữ liệu>[] <tên mảng>** = { <giá trị 1>, ..., <giá trị n> };

Ví dụ:

C#:

```
int[] IntArray = { 3, 9, 10 };
```

Về bản chất thì cách này trình biên dịch vẫn xem xét số phần tử khởi tạo và cấp phát vùng nhớ cho biến mảng sau đó thực khởi tạo giá trị cho các phần tử trong mảng. Nhưng cách viết này có vẻ nhanh và gọn hơn so với cách cấp phát vùng nhớ rồi mới khởi tạo giá trị.

Tóm lại ta có 3 cách khai báo và khởi tạo sau:

- Khai báo và cấp phát vùng nhớ

C#:

```
string[] Array = new string[3];
```

- Khai báo, cấp phát và khởi tạo giá trị cho mảng

C#:

```
string[] Kteam = new string[] { "HowKteam", "Free Education" };
```

- Khởi tạo giá trị cho mảng

C#:

```
int[] IntArray = { 3, 9, 10 };
```

Sử dụng mảng

Kiểu mảng có thể dùng làm:

- Kiểu dữ liệu cho biến.
- Kiểu trả về cho hàm.
- Tham số truyền vào cho hàm.
    - Các phần tử của mảng được truy xuất thông qua chỉ số phần tử và cặp dấu **[]**. Có thể xem các phần tử của mảng như là các biến đơn và thao tác như thao tác với biến bình thường.

C#:

```
// Khai báo, cấp phát và khởi tạo mảng kiểu string và tên là Kteam
string[] Kteam = new string[] { "HowKteam", "Free Education" };
/*
 * Vì chỉ số phần tử được đánh số từ 0 nên muốn truy xuất đến phần tử thứ 2 của mảng
thì chỉ số phần tử là 1
*/
            Console.WriteLine(Kteam[1]);
```

Một số thuộc tính và phương thức đặc trưng của mảng 1 chiều:

Tên thuộc tính hoặc phương thức	Ý nghĩa
Length	Thuộc tính trả về số nguyên kiểu int là số phần tử tối đa của mảng
LongLength	Thuộc tính trả về số nguyên kiểu long là số phần tử tối đa của mảng

GetLength(<số chiều>)	Trả về số nguyên kiểu int là số phần tử trong chiều đã xác định. Lưu ý chiều của mảng là các số nguyên và được đánh số từ 0. Cho nên đối với mảng 1 chiều thì số chiều là 0.
GetLongLength(<số chiều>)	Tương tự GetLength nhưng trả về số nguyên kiểu long
Sort()	Phương thức thực hiện sắp xếp mảng theo một thứ tự
Clear()	Phương thức xóa hết dữ liệu trong mảng và đưa về giá trị mặc định của kiểu. Lưu ý là chỉ xóa giá trị, vùng nhớ vẫn còn đó và có thể tiếp tục sử dụng mà không cần cấp phát.
Copy()	Thực hiện copy giá trị của mảng ra một vùng nhớ mới (phép gán thông thường thì 2 đối tượng sẽ dùng chung vùng nhớ rất nguy hiểm vì đối tượng này thay đổi dẫn đến đối tượng kia cũng thay đổi)
Reverse()	Phương thức thực hiện đảo ngược thứ tự của mảng 1 chiều

Còn rất nhiều thuộc tính và phương thức khác, mình chỉ giới thiệu một số cái hay dùng còn lại các bạn có thể tự khám phá.

Cách duyệt mảng 1 chiều:

- **Ý tưởng:**
  - Để truy xuất đến các phần tử của mảng cần thông qua chỉ số phần tử.
  - Mà chỉ số phần tử là các số nguyên tăng dần.
  - Từ đó ta có thể tận dụng vòng lặp để tăng giá trị 1 biến lên rồi xem biến đó như là chỉ số phần tử của mảng.
- **Ví dụ:**

C#:

```
int[] Kteam = new int[3];
            for (int i = 0; i < 3; i++) // Vì các phần tử có chỉ số là 0 1 2 nên điều kiện dừng là i < 3
            {
                // Do something
            }
```

- Mọi thứ đều suôn sẻ cho đến một ngày vì lí do nào đó bạn cần nâng cấp mảng Kteam lên 10 phần tử. Khi đó vấn đề sẽ xuất hiện, bạn phải đi thay đổi tất cả những chỗ nào liên quan đến số phần tử từ 3 thành 10 hết. Như vậy mỗi lần có thay đổi về số phần tử bạn đều phải làm lại những việc đó. Vậy tại sao ta không tận dụng một thuộc tính vừa được học xong để giải quyết?

C#:

```
int[] Kteam = new int[3];

            /*
             * Thay số 3 thành thuộc tính Length.
             * Bây giờ bạn có thay đổi số phần tử thì chỉ cần thay đổi ở khai báo thôi là xong!
             */

            for (int i = 0; i < Kteam.Length; i++)
            {
                // Do something
            }
```

- Với cách duyệt mảng như thế này bạn đã có thể làm mọi thứ với mảng từ nhập xuất giá trị cho mảng đến tính toán phức tạp.

Ví dụ chương trình sử dụng mảng 1 chiều:

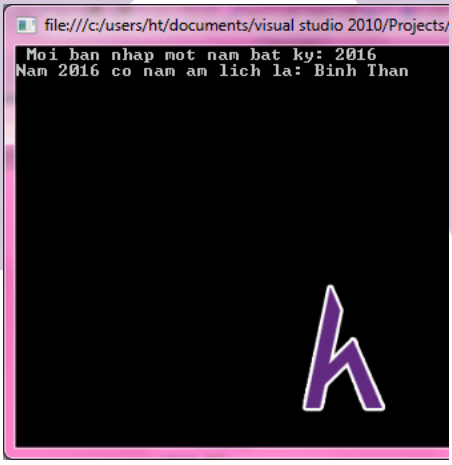
Trước khi vào ví dụ thì các bạn hãy xem lại ví dụ chương trình tính năm âm lịch từ năm dương lịch đã nhập ở bài CẤU TRÚC RẼ NHÁNH SWITCH CASE. Có vẻ code của chúng ta hơi dài, bây giờ mình hãy thử tổ chức lại chương trình đó bằng cách sử dụng mảng.

- **Ý tưởng:**
  - Chúng ta thấy các case là các giá trị nguyên không âm và tăng dần làm ta liên tưởng tới chỉ số của mảng.
  - Như vậy nếu như ta tạo ra 1 mảng có số phần tử bằng số case và giá trị của phần tử tại chỉ số thứ i sẽ tương đương với giá trị của case i.
  - Việc còn lại chỉ là tra cứu theo chỉ số của mảng nữa là xong!
- **Chương trình ví dụ:**

C#:

```
int Year; // Biến chứa giá trị năm cần tính.
            // Mảng Can chứa các giá trị can tương ứng theo bảng can
            string[] Can = { "Canh", "Tan", "Nham", "Quy", "Giap", "At", "Binh", "Dinh", "Mau", "Ky" };

            // Mảng Chi chứa các giá trị chi tương ứng theo bảng chi
            string[] Chi = { "Than", "Dau", "Tuat", "Hoi", "Ty", "Suu", "Dan", "Meo", "Thin", "Ty", "Ngo", "Mui" };


            Console.Write(" Moi ban nhap mot nam bat ky: ");

// Nhập năm dương lịch và ép kiểu về kiểu số nguyên
            Year = Int32.Parse(Console.ReadLine());


            /*
             * Vì kết quả phép chia lấy dư của Year%10 hoặc Year%12 sẽ cho ra số nguyên
             * Nên ta sẽ dùng nó làm chỉ số phần tử để tra cứu ra giá trị can chi tương ứng. Thay vì dùng cách cũ là switch case
             * Như vậy cách này vừa đơn giản vừa dễ hiểu, code rất ít sẽ với cách dùng switch case
             */

            Console.WriteLine("Nam {0} co nam am lich la: {1} {2}", Year, Can[Year%10], Chi[Year%12]); // Nối Can và Chi lại để
được năm âm lịch

            Console.ReadLine();
```

Kết quả khi chạy chương trình trên là:



Kết luận

Nội dung bài này giúp các bạn nắm được:

- Khái niệm về mảng trong C#. Tại sao phải sử dụng mảng?
- Khai báo, khởi tạo và sử dụng mảng 1 chiều trong C#.

Bài sau chúng ta sẽ tìm hiểu về **MẢNG 2 CHIỀU TRONG C#**

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".