Bài: Vòng lặp foreach trong lập trình C# cơ bản

Xem bài học trên website để ủng hộ Kteam: Vòng lặp foreach trong lập trình C# cơ bản

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 MẢNG NHIỀU CHIỀU TRONG C#. Hôm nay chúng ta sẽ cùng tìm hiểu về **cấu trúc lặp foreach 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 HÀM CƠ BẢN TRONG C#
- MẢNG 1 CHIỀU TRONG C#
- MẢNG 2 CHIỀU TRONG C#

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

- Cú pháp và nguyên tắc hoạt động của foreach trong C#
- Sử dụng foreach trong C#
- So sánh for và foreach trong C#

Cú pháp và nguyên tắc hoạt động của foreach trong C#

Cấu trúc lặp foreach cho phép chúng ta duyệt 1 mảng hoặc 1 tập hợp (sẽ được trình bày trong bài TỔNG QUAN VỀ COLLECTION TRONG C#).

Một số đặc trưng của foreach:

- Foreach không duyệt mảng hoặc tập hợp thông qua chỉ số phần tử như cấu trúc lặp for.
- Foreach duyệt tuần tự các phần tử trong mảng hoặc tập hợp.
- Foreach chỉ dùng để duyệt mảng hoặc tập hợp ngoài ra không thể làm gì khác.

Cú pháp

```
foreach (<kiểu dữ liệu> <tên biến tạm> in <tên mảng hoặc tập hợp>)
{
        // Code xử lý
}
```

Trong đó:

- Các từ khoá **foreach**, **in** là từ khoá bắt buộc.

Copyright © Howkteam.com

- <kiểu dữ liệu> là kiểu dữ liệu của các phần tử trong mảng hoặc tập hợp.
- <tên biến tạm> là tên 1 biến tạm đại diện cho phần tử đang xét khi duyệt mảng hoặc tập hợp.
- <tên mảng hoặc tập hợp> là tên của mảng hoặc tập hợp cần duyệt.

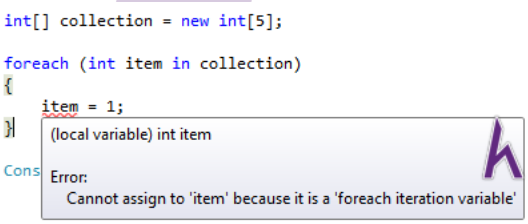
Nguyên tắc hoạt động

Foreach cũng có nguyên tắc hoạt động tương tự như các cấu trúc lặp khác cụ thể như sau:

- Ở vòng lặp đầu tiên sẽ gán giá trị của phần tử đầu tiên trong mảng vào biến tạm.
- Thực hiện khối lệnh bên trong vòng lặp foreach.
- Qua mỗi vòng lặp tiếp theo sẽ thực hiện kiểm tra xem đã duyệt hết mảng hoặc tập hợp chưa. Nếu chưa thì tiếp gán giá trị của phần tử hiện tại vào biến tạm và tiếp tục thực hiện khối lệnh bên trong.
- Nếu đã duyệt qua hết các phần tử thì vòng lặp sẽ kết thúc.

Qua nguyên tắc hoạt động trên ta có thể thấy:

- Biến tạm trong vòng lặp foreach sẽ tương đương với phần tử i trong cách duyệt của vòng lặp for (đã trình bày trong bài CẤU TRÚC VÒNG LẶP FOR TRONG C#).
- Qua mỗi bước lặp ta chỉ có thể thao tác với giá trị của phần tử đang xét mà không thể tương tác với các phần tử đứng trước nó hay đứng sau nó (trong CẤU TRÚC VÒNG LẶP FOR TRONG C# thì hoàn toàn được).
- Bằng cách duyệt của foreach ta không thể thay đổi giá trị của các phần tử vì lúc này giá trị của nó đã được sao chép ra một 1 biến tạm và ta chỉ có thể thao tác với biến tạm.
- Thậm chí việc thay đổi giá trị của biến tạm cũng không được phép. Nếu ta cố làm điều đó thì sẽ gặp lỗi sau:



Sử dụng foreach trong C#

Trong C#, có những danh sách, tập hợp mà ta không thể truy xuất đến các phần tử của nó thông qua chỉ số phần tử được (ví dụ như kiểu **List** – sẽ được trình bày trong bài LIST TRONG C# hoặc các **collection**, **generic** – sẽ được trình bày trong bài COLLECTION TRONG C# và bài GENERIC TRONG C#).

Trong trường hợp như vậy, để duyệt các danh sách, tập hợp có tính chất như trên thì foreach là lựa chọn tốt nhất.

Chúng ta sẽ tìm hiểu sức mạnh của foreach qua các bài học sau. Còn trong bài học này mình chỉ ví dụ đơn giản để các bạn có thể nắm cú pháp cũng như cách sử dụng foreach.

- Xét chương trình sau:

:

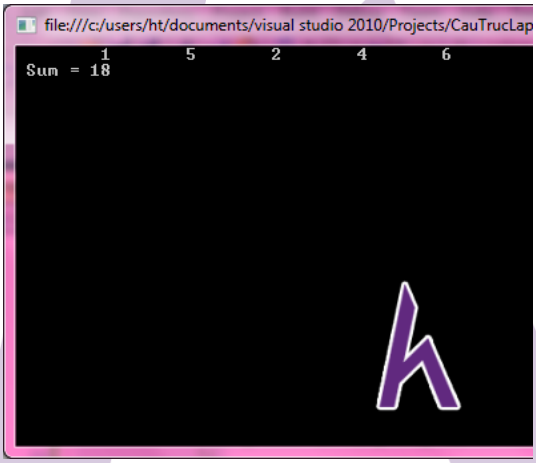
Copyright © Howkteam.com

```
/*
             * Khai báo mảng 1 chiều IntArray và khởi tạo giá trị.
             * Các bạn có thể xem lại cú pháp này ở bài Mảng 1 chiều trong C#
             * Khai báo 1 biến Sum để chứa giá trị tổng các phần tử trong mảng IntArray.
             */
            int[] IntArray = { 1, 5, 2, 4, 6 };
            int Sum = 0;

            /*
             * Sử dụng foreach để duyệt mảng và in giá trị của các phần tử trong mảng.
             * Đồng thời tận dụng vòng lặp để tính tổng các phần tử trong mảng.
             */
            foreach (int item in IntArray)
            {
                Console.Write("\t" + item);
                Sum += item;
            }

            Console.WriteLine("\n Sum = " + Sum);
```

Có lẽ chúng ta đã không mấy xa lạ với đoạn chương trình trên. Đoạn chương trình trên sẽ duyệt mảng để in ra các giá trị của mảng và tính tổng các phần tử trong mảng.

Nhưng thay vì sử dụng for thì mình sử dụng foreach để các bạn có thể thấy được sự tương đồng giữa các thành phần trong **cấu trúc foreach** và CẤU TRÚC VÒNG LẶP FOR TRONG C#.

Kết quả khi chạy đoạn chương trình trên:



Ví dụ thứ 2: ta thử sử dụng foreach để duyệt mảng jagged

:

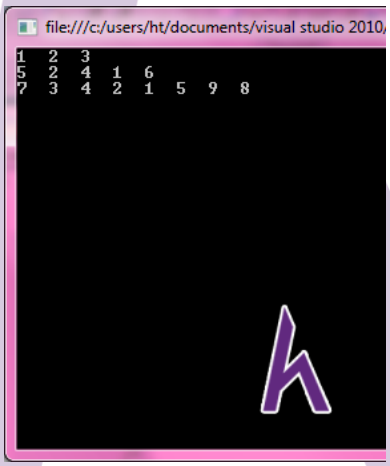
Copyright © Howkteam.com

```
/*
             * Khai báo 1 mảng jagged tên là JaggedArray và khởi tạo giá trị.
             * Các bạn có thể xem lại cú pháp khai báo này ở bài Mảng nhiều chiều trong C#.
             */
            int[][] JaggedArray =
                            {
                                new int[] { 1, 2, 3 },
                                new int[] { 5, 2, 4, 1, 6},
                                new int[] { 7, 3, 4, 2, 1, 5, 9, 8}
                            };

            /*
             * Tương tự như dùng for, ta cũng dùng 2 vòng foreach lồng vào nhau để duyệt mảng.
             */
            foreach (int[] Element in JaggedArray)
            {
                foreach (int item in Element)
                {
                    Console.Write(item + "  ");
                }
                Console.WriteLine();
            }
```

Ta có thể thấy cách duyệt foreach ngắn gọn hơn nhiều so với cách duyệt bằng vòng lặp for thông thường.

Ta cũng chả quan tâm đến việc phải xử lý độ dài mảng hay chỉ số phần tử để truy xuất 1 phần tử nào đó.

- Kết quả khi chạy đoạn chương trình trên:



So sánh for và foreach trong C#

Foreach mang trong mình một số ưu điểm như:

- Câu lệnh ngắn gọn, sẽ sử dụng.
- Rất có ích khi duyệt danh sách, tập hợp mà không thể truy xuất thông qua chỉ số phần tử.
- Duyệt các danh sách, tập hợp có số phần tử không xác định hoặc số phần tử thay đổi liên tục.

Mặc dù có nhiều ưu điểm nhưng không hẳn là foreach hơn hẳn for. Cùng điểm qua một vài tiêu chí để xem 2 đối thủ này ai hơn ai nhé.

Tiêu chí	For	Foreach

Copyright © Howkteam.com

Khả năng truy xuất phần tử	Truy xuất ngẫu nhiên (có thể gọi bất kỳ phần tử nào trong mảng để sử dụng)	Truy xuất tuần tự (chỉ sử dụng được giá trị phần tử đang xét)
Thay đổi được giá trị của các phần tử	Có	Không
Duyệt mảng, tập hợp khi không biết được số phần tử của mảng, tập hợp	Không	Có
Hiệu suất (tốc độ xử lý) (*)	Đối với mảng, danh sách hoặc tập hợp có khả năng truy xuất ngẫu nhiên thì for sẽ chiếm ưu thế	Đối với mảng, danh sách hoặc tập hợp không có khả năng truy xuất ngẫu nhiên thì foreach chiếm ưu thế

(*) Nhìn chung hiệu suất của for và foreach còn phụ thuộc vào cấu trúc dữ liệu đang xét cho nên việc so sánh này chỉ mang tính chất tham khảo.

Sau đây là 2 đoạn chương trình kiểm tra tốc độ của for và foreach đối với 2 cấu trúc dữ liệu là mảng 1 chiều (có khả năng truy xuất ngẫu nhiên) và danh sách liên kết **LinkedList** (không có khả năng truy xuất ngẫu nhiên):

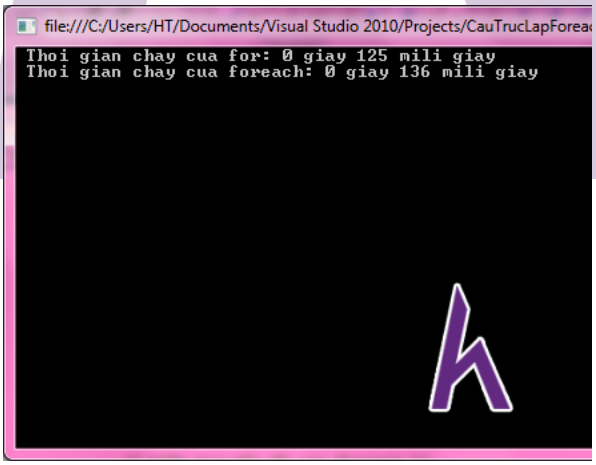
Đầu tiên là mảng 1 chiều:

:

Copyright © Howkteam.com

```
/* Kiểm tra tốc độ của for */

/*
             * Sử dụng 1 cái đồng hồ để đo thời gian chạy của 2 vòng lặp for và foreach
             * Ở đây mình chỉ kiểm tra tốc độ chứ không tập trung giải thích cú pháp
             * Các bạn có thể tìm hiểu thêm.
             */
            Stopwatch start = new Stopwatch();
            start.Start();

            int[] IntArray = new int[Int32.MaxValue / 100];
            int s = 0;
            int Length = IntArray.Length;
            for (int i = 0; i < Length; i++)
            {
                s += IntArray[i];
            }

            start.Stop();
            Console.WriteLine(" Thoi gian chay cua for: {0} giay {1} mili giay", start.Elapsed.Seconds,
start.Elapsed.Milliseconds);

            /* Kiểm tra tốc độ của foreach */
            Stopwatch start2 = new Stopwatch();
            start2.Start();

            int[] IntArray2 = new int[Int32.MaxValue / 100];
            int s2 = 0;

            foreach (int item in IntArray2)
            {
                s2 += item;
            }

            start2.Stop();
            Console.WriteLine(" Thoi gian chay cua foreach: {0} giay {1} mili giay", start2.Elapsed.Seconds,
start2.Elapsed.Milliseconds);
```

- Đoạn chương trình mình thực hiện:
  - Khai báo 1 mảng 1 chiều có 20 triệu phần tử (khai báo số phần tử lớn để có thể thấy được sự chêch lệch về tốc độ)
  - Lần lượt dùng for, foreach để duyệt mảng đó và thực hiện 1 câu lệnh nào đó.
  - Cuối cùng là xuất ra thời gian thực thi của từng trường hợp dưới dạng giây và mili giây.
- Kết quả khi chạy đoạn chương trình trên:



Copyright © Howkteam.com

- Dựa vào kết quả ta có thể thấy được sự chêch lệch nhỏ về tốc độ, nếu kiểm tra với số phần tử lớn hơn hoặc cấu trúc dữ liệu phức tạp hơn thì chêch lệch này càng lớn.

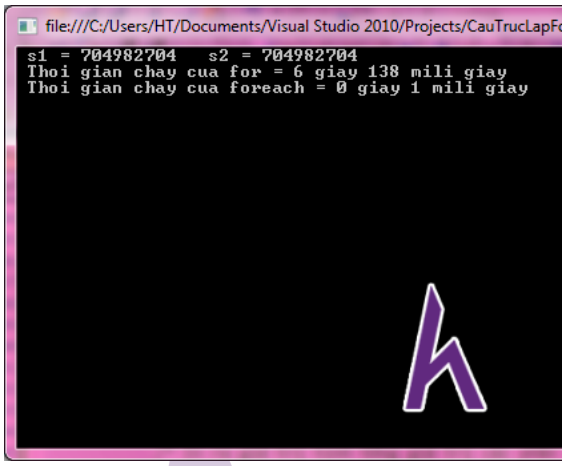
Tiếp theo là đến danh sách liên kết LinkedList:

:

```
/*
             * Khai báo 1 LinkedList chưa các số nguyên int và khởi tạo giá trị cho nó.
             */
            LinkedList<int> list = new LinkedList<int>();

            for (int i = 0; i < 100000; i++)
            {
                list.AddLast(i);
            }

        /* Kiểm tra tốc độ của for */
            Stopwatch st = new Stopwatch();
            int s1 = 0, length = list.Count;
            st.Start();
            for (int i = 0; i < length; i++)
            {
                /*
                 * Vì LinkedList không thể truy xuất thông qua chỉ số phần tử
                 * nên mình phải sử dụng 1 phương thức hỗ trợ làm điều này.
                 * Và đây chính là sự hạn chế của for đối với các cấu trúc dữ liệu tương tự như danh sách liên kết này.
                 */
                s1 += list.ElementAt(i);
            }
            st.Stop();

        /* Kiểm tra tốc độ của foreach */
            Stopwatch st2 = new Stopwatch();
            int s2 = 0;
            st2.Start();
            foreach (int item in list)
            {
                /*
                 * Vì foreach không quan tâm đến chỉ số phần tử nên code viết rất ngắn gọn
                 */
                s2 += item;
            }
            st2.Stop();

            /* In ra giá trị tính tổng giá trị các phần tử khi duyệt bằng for và foreach để chắc chắn rằng cả 2 đều chạy đúng */
            Console.WriteLine(" s1 = {0}   s2 = {1}", s1, s2);
            Console.WriteLine(" Thoi gian chay cua for = {0} giay {1} mili giay", st.Elapsed.Seconds, st.Elapsed.Milliseconds);
            Console.WriteLine(" Thoi gian chay cua foreach = {0} giay {1} mini giay", st2.Elapsed.Seconds,
st2.Elapsed.Milliseconds);
```

- Chương trình mình thực hiện:
  - Tạo 1 **LinkedList** và thêm vào 100000 phần tử.
  - Sau đó lần lượt dùng for, foreach duyệt **LinkedList** trên và tính tổng giá trị các phần tử trong mảng.
  - Cuối cùng in ra thời gian chạy của for, foreach.
- Kết quả khi chạy chương trình trên là:

Copyright © Howkteam.com



- Dựa vào kết quả chạy ta thấy sự chênh lệch là quá lớn, rõ ràng đối với cấu trúc dữ liệu phức tạp, không hỗ trợ truy xuất thông qua chỉ số phần tử nữa thì foreach chiếm ưu thế.

> Tuỳ vào từng trường hợp mà ta nên dùng **for** hay **foreach**. Không nên lạm dụng 1 thứ quá nhiều.

Kết luận

Nội dung bài này giúp các bạn nắm được:

- Cú pháp của foreach trong C#.
- Sử dụng foreach trong C#.
- So sánh for và foreach trong C#.

Bài sau chúng ta sẽ tìm hiểu về **LỚP STRING 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**".

Copyright © Howkteam.com