Bài: Cấu trúc lặp Goto trong C#.

Xem bài học trên website để ủng hộ Kteam: Cấu trúc lặp Goto 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 CẤU TRÚC CƠ BẢN CỦA VÒNG LẶP. Ở bài này chúng ta sẽ cùng đi sâu vào chi tiết cách sử dụng **vòng lặp goto**

> Khuyến cáo không sử dụng Goto trong ngôn ngữ 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 CỦA MỘT CHƯƠNG TRÌNH C# console application
- BIẾN và KIỂU DỮ LIỆU trong C#
- TOÁN TỬ TRONG C#
- CÂU ĐIỀU KIỆN TRONG C#
- CẤU TRÚC CƠ BẢN CỦA VÒNG LẶP

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

- Cấu trúc của một vòng lặp goto
- Các ví dụ sử dụng goto

Cấu trúc của một vòng lặp goto

Cái tên goto có thể hiểu là đi đến đâu đó. Thường sử dụng cấu trúc goto người ta sẽ đi kèm một câu điều kiện (có thể không cần).

Cấu trúc:

```
    goto <label>;
```

- Trong đó label là một nhãn đích đến trong code. Nơi mà code sẽ tiếp tục được thực thi từ đó. Cấu trúc của một label: <Tên label>:
- goto là từ khóa thông báo cho trình biên dịch biết sẽ đi đến nhãn ngay sau để tiếp tục thực thi code.

Các ví dụ sử dụng goto:

C#:

```
static void Main(string[] args)
        {
            int a = 1;
            switch(a)
            {
                case 1:             // label case 1
                    Console.WriteLine("Case 1");
                    break;
                case 2:             // label case 2
                    Console.WriteLine("Case 2");
                    goto case 1;    // dịch chuyển tới label case 1
                    break;          // Đoạn code này thừa vì sẽ không bao giờ thực thi
                case 3:             // label case 3
                    Console.WriteLine("Case 3");
                    break;
            }
      Console.ReadKey();
        }
```

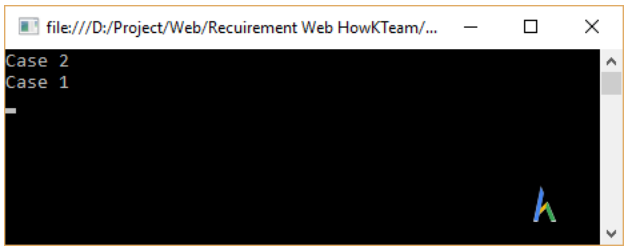
- Trong đoạn code này chúng ta biên dịch sẽ thấy kết quả xuất ra màn hình dòng chữ "Case 1" vì a == 1.

- Giờ chúng ta thay giá trị a = 2 xem kết quả ra sao nhé

C#:

```
static void Main(string[] args)
        {
            int a = 2;
            switch(a)
            {
                case 1:             // label case 1
                    Console.WriteLine("Case 1");
                    break;
                case 2:             // label case 2
                    Console.WriteLine("Case 2");
                    goto case 1;    // dịch chuyển tới label case 1
                    break;          // Đoạn code này thừa vì sẽ không bao giờ thực thi
                case 3:             // label case 3
                    Console.WriteLine("Case 3");
                    break;
            }
            Console.ReadKey();
        }
```

- Kết quả:



Ta thấy chương trình đi vào case 2 sau đó gặp dòng code goto case 1; nên đã nhảy lên thực thi dòng code

C#:

```
case 1:             // label case 1
Console.WriteLine("Case 1");
break;
```

rồi mới kết thúc chương trình.

Một ví dụ khác kết hợp với câu điều kiện:

C#:

```
static void Main(string[] args)
        {
            int a = 1;
            // nếu a == 2
            if (a == 2)
            {
                // dịch chuyển tới vị trí label a_Is_2
                goto a_Is_2;
            }

            Console.WriteLine("A == 1");
            a_Is_2:
            Console.WriteLine("A == 2");

            Console.ReadKey();
        }
```

- Ở trường hợp này kết quả màn hình sẽ xuất ra dòng chữ "A == 1" và dòng chữ "A == 2" ra màn hình.

Chúng ta có thể nhận thấy 2 điều ở ví dụ này:

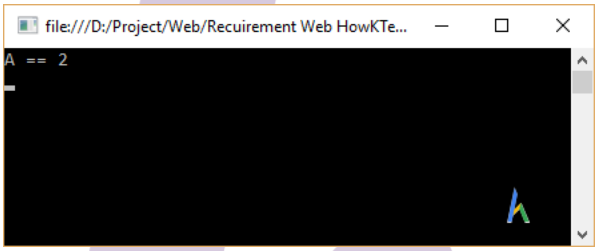
> label có thể được tạo ra một cách dễ dàng và không nhất thiết phải nằm trong cấu trúc switch.
>
> Việc tạo label sẽ **không** ảnh hưởng gì đến code thông thường.

Chúng ta thử cho giá trị của a = 2 và xem kết quả nhé:

C#:

```
static void Main(string[] args)
        {
            int a = 2;
            // nếu a == 2
            if (a == 2)
            {
                // dịch chuyển tới vị trí label a_Is_2
                goto a_Is_2;
            }

            Console.WriteLine("A == 1");
            a_Is_2:
            Console.WriteLine("A == 2");

            Console.ReadKey();
        }
```

- Kết quả sẽ in ra màn hình dòng chữ "A == 2". Dòng chữ "A == 1" không được in ra do đoạn code Console.**WriteLine**("A == 1"); bị bỏ qua vì câu lệnh goto dịch chuyển đến label a_Is_2: nằm bên dưới nó.



> Vậy rõ ràng lần này chương trình không đi theo trình tự từ trên xuống dưới mà bị dịch chuyển ngược lên trên trở lại bởi câu lệnh goto.
>
> Bản chất của goto là **nhảy đến bất cứ nơi đâu có label**. Mọi câu lệnh khác khi goto bỏ qua đều bị trình biên dịch bỏ qua.

Ví dụ về vòng lặp vô tận:

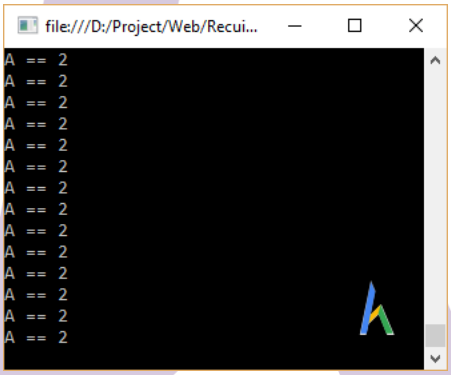
Chỉ cần câu điều kiện lặp luôn thỏa mãn thì vòng lặp sẽ lặp mãi.

C#:

```
static void Main(string[] args)
        {
            int a = 2;

            Ifinity_Loop:
            // nếu a == 2
            if (a == 2)
            {
                // dịch chuyển tới vị trí label a_Is_2
                goto a_Is_2;
            }

            Console.WriteLine("A == 1");
            a_Is_2:
            Console.WriteLine("A == 2");
            goto Ifinity_Loop; // dịch chuyển tới vị trí label Ifinity_Loop

            Console.ReadKey();
        }
```

Chúng ta thấy khi chạy chương trình sẽ in ra hàng loạt dòng "A == 2" do đoạn code Console.WriteLine("A == 2"); được lặp lại liên tục. Đoạn code làm phát sinh vấn đề này là ở đây:

- Ban đầu chương trình sẽ đi đến đoạn code goto a_Is_2;.
- Sau đó lại bị đoạn code goto Ifinity_Loop; đưa ngược lại label Ifinity_Loop.
- Mãi không thể nào kết thúc quá trình lặp lại này dẫn đến một vòng lặp vô tận.



Kết luận

Qua bài này chúng ta đã nắm được cách sử dụng goto và label. Đây là một cấu trúc được khuyến cáo là **hạn chế sử dụng** (không dùng thì tốt hơn) trong lập trình C# vì nó có thể phá vỡ cấu trúc của một chương trình (*đi từ trên xuống*).

Bài sau chúng ta sẽ tiếp tục tìm hiểu về cấu trúc lặp tiếp theo đó là **CẤU TRÚC VÒNG LẶP FOR 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**".