Bài 7: Loop (Vòng Lặp)

Chúng ta đã biết làm cách nào để thiết lập những conditions, bây giờ chúng ta sẽ học cách thực hiện các vòng lặp. Vậy loop (vòng lặp) là gì ?

Là một phương pháp giúp ta có thể lặp lại nhiều lần một nhóm các instructions. Rất tiện lợi, đặc biệt là trong bài thực hành đầu tiên đang đợi bạn sau khi kết thúc phần này
Hãy thư giãn, không có gì phức tạp cả. Ở phần trước, chúng ta đã thấy thế nào là những Boolean, tương đối khó nuốt.

Nhưng bây giờ thì trôi chảy hơn rồi, và bài thực hành sắp tới cũng không khiến bạn gặp quá nhiều rắc rối đâu.

Nói chung là hãy cố hiểu rõ nó, vì chúng ta chuẩn bị bước vào chương thứ II, và sẽ có nhiều điều hứng thú hơn rất nhiều.

Thế nào là một vòng lặp?

Tương tự như các conditions, có nhiều cách để thực hiện một vòng lặp. Nhưng dù thực hiện bằng cách nào, thì chúng đều thực hiện một chức năng: **lặp lại nhiều lần các instruction**.

Chúng ta sẽ tìm hiểu 3 dạng vòng lặp thường sử dụng trong C:

1. while
2. do... while
3. for

Biểu đồ thể hiện cách hoạt động của các vòng lặp

Máy tính thực hiện những instruction từ cao xuống thấp (giống như mọi khi)

i. Khi đã đến cuối của vòng lặp, nó quay trở lại instruction đầu tiên.
ii. Nó lại đọc những instruction từ cao xuống thấp...
iii. ... Và lại bắt đầu từ instruction đầu tiên.

Vấn đề xảy ra là nếu ta không dừng nó lại thì máy tính sẽ không ngừng lặp lại những instruction này. Điều này không khiến ta bận tâm vì nó lặp lại theo yêu cầu của chúng ta.

Và tại đây lại xuất hiện ... **những điều kiện**!

Khi chúng ta tạo một vòng lặp, chúng ta luôn phải tạo một điều kiện. Điều kiện này có nghĩa là "Lặp lại vòng lặp nếu điều kiện này vẫn đúng."
Có rất nhiều cách để thực hiện nó giống như tôi đã nói. Và sau đây là cách thực hiện một vòng lặp while trên C

While loop (vòng lặp While)

Cách tạo ra một vòng lặp while.

Code C:

```
while (/* Dieukien */)
{
    // Cac instructions duoc lap lai
}
```

Chỉ đơn giản như vậy
While có nghĩa là "hễ còn". Chúng ta sẽ nói với máy tính "*Hễ điều kiện vẫn đúng thì lặp lại các instruction được viết trong dấu gộp.*"
Tôi thực hiện một ví dụ đơn giản: Yêu cầu người sử dụng nhập vào số 47. Hễ người sử dụng không thực hiện đúng, thì máy tính sẽ tiếp tục yêu cầu nhập vào số 47. Chương trình sẽ không dừng lại nếu như người sử dụng vẫn không nhập vào số 47.

Code C:

```
int giatriCanNhap = 0;
while (giatriCanNhap != 47)
{
    printf("Hay nhap vao so 47 ! ");
    scanf("%d", &giatriCanNhap);
}
```

Và đây là cách mà ví dụ trên thực hiện. Ghi thêm rằng tôi đã ép mình đánh sai 2-3 lần trước khi đánh vào số chính xác.

Console:

```
Hay nhap vao so 47 ! 10
Hay nhap vao so 47 ! 27
Hay nhap vao so 47 ! 40
Hay nhap vao so 47 ! 47
```

Chương trình tự dừng lại cho đến khi số 47 được nhập vào.
Vòng lặp sẽ lặp lại các instruction nếu như người sử dụng vẫn không nhập vào đúng số 47. Chỉ đơn giản như vậy.

Bây giờ, chúng ta sẽ thử làm một vài điều thú vị hơn: chúng ta muốn vòng lặp sẽ lặp lại nhiều lần một instruction.

Chúng ta sẽ tạo một biến số "counter" có giá trị 0 lúc bắt đầu, chúng ta sẽ tăng dần giá trị đó lên. Bạn còn nhớ **increment** ko? Chúng ta sẽ cộng thêm 1 vào biến số bằng cách viết "bienso++".

Hãy đọc kĩ đoạn mã này và hãy thử hiểu cách hoạt động:

Code C:

```c
long counter = 0;
while (counter < 10)
{
    printf ("Xin chao cac ban !\n");
    counter++;
}
```

Kết quả :

Console:

```
Xin chao cac ban !
Xin chao cac ban !
Xin chao cac ban !
Xin chao cac ban !
Xin chao cac ban !
Xin chao cac ban !
Xin chao cac ban !
Xin chao cac ban !
Xin chao cac ban !
Xin chao cac ban !
```

Đoạn mã này sẽ lặp lại 10 lần câu "Xin chao cac ban !".

Chính xác nó hoạt động như thế nào?

Trình tự hoạt động của vòng lặp như sau:

1. Lúc bắt đầu, chúng ta có một biến số *counter* có giá trị là 0.
2. Vòng lặp **while** vẫn lặp lại hễ giá trị counter vẫn bé hơn 10, vì counter có giá trị là 0 lúc bắt đầu, máy tính sẽ đi vào vòng lặp
3. Hàm *printf* sẽ hiển thị ra màn hình câu "Xin chao cac ban !"
4. Máy tính sẽ **tăng giá trị của biến số counter lên 1**, nhờ vào instruction "*counter++;*". Bây giờ counter có giá trị là 1
5. Đã đến cuối của vòng lặp (dấu **}**), bây giờ chúng ta quay lại từ khi bắt đầu, từ while. Chúng ta sẽ kiểm tra lại điều kiện: "*có phải giá trị của counter vẫn bé hơn 10* ?". Vâng, giá trị của nó hiện giờ là 1, vậy hãy lặp lại các instruction của vòng lặp.

Và cứ thế tiếp tục... Counter tăng dần các giá trị 0, 1 ,2 ,3 ,...,8 ,9 ,10. Đến khi counter có giá trị là 10, điều kiện counter < 10 không còn chính xác, nên chúng ta sẽ ra khỏi vòng lặp

Mặt khác, chúng ta có thể thấy giá trị của biến *counter* tăng dần theo kích cỡ của vòng lặp
Nếu bạn đã hiểu vấn đề này, thì xem như bạn đã hiểu tất cả về vòng lặp while.

Bạn có thể tăng giới hạn của vòng lặp ("<100" thay vì "<10"). Việc này rất hữu ích nếu bỗng dưng bạn bị chép phạt 100 lần

Chú ý những vòng lặp không giới hạn

Khi bạn tạo một vòng lặp,hãy chắc chắn rằng nó có thể dừng lại tại một thời điểm nào đó ! Nếu điều kiện luôn luôn đúng, chương trình của bạn sẽ không bao giờ dừng lại !
Đây là một ví dụ về một vòng lặp không giới hạn:

Code C:

```
while (1)
{
    printf("Vong lap khong gioi han\n");
}
```

Bạn có nhớ những Boolean: 1 = đúng, 0 = sai. Tại đây, điều kiện luôn luôn đúng, và máy tính sẽ hiển thị "vong lap khong gioi han" liên tục và không ngừng !

 Để dừng lại chương trình, trên Windows bạn không có lựa chọn nào khác ngoài việc nhấn vào dấu X ở góc phải bên trên của console. Trên Linux, bạn có thể nhấn Ctrl + C để dừng chương trình.

Tóm lại hãy chú ý: bằng mọi cách phải tránh rơi vào vòng lặp không giới hạn.

Đôi khi vòng lặp không giới hạn sẽ có lợi, nhất là trong các game điện tử ta sẽ thấy sau này.

Do… while loop (vòng lặp do... while)

Dạng vòng lặp này tương tự như while, chỉ khác một điều là nó ít được sử dụng hơn.
Điều khác biệt so với while là vị trí của điều kiện. Đối với *while* điều kiện nằm ở vị trí bắt đầu vòng lặp, còn ở *do... while*, **điều kiện nằm ở cuối cùng**:

Code C:

```
long counter = 0;
do
{
    printf ("Xin chao cac ban !\n");
    counter++;
} while (counter < 10);
```

Có điều gì khác biệt ở đây ?

Rất đơn giản: vòng lặp while luôn chắc chắn rằng nó sẽ không bao giờ hoạt động nếu như điều kiện sai từ khi bắt đầu. Ví dụ, nếu như ta gán cho counter giá trị là 50, nhưng điều kiện sai kể từ khi bắt đầu thì chương trình sẽ không tiến vào vòng lặp.

Đối với vòng lặp do... while thì khác: vòng lặp này luôn luôn thực hiện **ít nhất một lần**. Thực tế, điều kiện sẽ được kiểm tra ở vị trí kết thúc giống như ta đã thấy. Nếu như biến *counter* có giá trị là 50, mặc dù điều kiện bị sai nhưng vòng lặp vẫn sẽ thực hiện ít nhất một lần.

Đôi khi vòng lặp này khá tiện dụng vì luôn chắc rằng chương trình luôn chạy nó một lần. Nhưng dù gì thì nó vẫn khá hiếm.

 Có một điều đặc biệt trong vòng lặp do... while là có một dấu chấm phẩy phía sau while, bạn đừng quên điều đó. Nếu không có nó, chương trình của bạn sẽ không thể biên dịch được

For loop (vòng lặp for)

Về nguyên tắc, vòng lặp while cho phép thực hiện tất cả những vòng lặp mà ta muốn. Nhưng trong nhiều trường hợp, người ta cần một loại vòng lặp khác gọn gàng hơn.

Vòng lặp for được sử dụng khá nhiều trong lập trình. Tôi không thể tính chính xác nhưng tôi chắc rằng nó được sử dụng gấp nhiều lần while, vì vậy bạn cần phải biết rõ 2 loại vòng lặp này. Như tôi đã nói với bạn, vòng lặp for cũng chính là một dạng khác của while. Đây là một ví dụ về while mà ta đã thấy trước đó

Code C:

```
long counter = 0;
while (counter < 10)
{
    printf ("Xin chao cac ban !\n");
    counter++;
}
```

Cũng trong trường hợp tương tự nhưng ta nếu ta dùng vòng lặp for:

Code C:

```
long counter;
for (counter = 0 ; counter < 10 ; counter++)
{
    printf ("Xin chao cac ban !\n");
}
```

Có bao nhiêu điểm khác biệt?

- Bạn có thể thấy rằng chúng ta đã không khai báo giá trị của biến số ngay sau khi tạo ra nó (nhưng chúng ta có quyền làm điều đó).
- Có rất nhiều thứ trong ngoặc sau for (chúng ta sẽ xem xét sau).
- Cũng không có counter++ trong vòng lặp giống như khi dùng vòng lặp while.

Nó được tìm thấy trong ngoặc **( )** và cũng chính điểm này khiến vòng lặp *for* trở nên thú vị.

Có 3 instructions viết ngắn gọn trong ngoặc và chúng **cách nhau bởi những dấu chấm phẩy:**

- instruction đầu tiên **dùng để khai báo**: khai báo biến số *counter*. Trong trường hợp của chúng ta, biến số có giá trị là 0.

- instruction thứ hai là **điều kiện**: giống như vòng lặp while, đây là điều kiện để vòng lặp được thực hiện. Khi điều kiện vẫn còn đúng, thì vòng lặp lại sẽ được tiếp tục.

- Cuối cùng có một **increment** ở đây: instruction cuối cùng sẽ được thực hiện ở cuối mỗi vòng lặp để cập nhật giá trị của biến số counter. Tương tự, chúng ta cũng có thể thực hiện decrement (*counter--;*) hoặc bất kì dạng phép tính nào (*counter += 2);* để tăng hoặc giảm giá trị cho những biến số.

Tóm lại, giống như ta đã thấy vòng lặp *for* không có gì khác biệt ngoài một số thứ được viết ngắn gọn hơn so với vòng lặp *while*

Hãy nắm vững nó, chúng ta sẽ cần sử dụng nó rất nhiều lần! Trong chương tiếp theo, có lẽ chúng ta sẽ mệt mỏi với một ít bài thực hành.

Như bạn đã biết, trong những bài thực hành hầu như sẽ không có thêm kiến thức mới, đây là cơ hội để bạn có thể ứng dụng những gì đã được học trong những bài học trước.

Để kết thúc phần này với một hình vui mà tôi chắc là bạn đã có thể hiểu được ý nghĩa của nó