Bài: Vòng lặp For trong C++ (For statements)

Xem bài học trên website để ủng hộ Kteam: Vòng lặp For trong C++ (For statements)

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 bài học trước, mình đã chia cho các bạn về VÒNG LẶP DO WHILE TRONG C++ (Do while statements). Vòng lặp **do-while** sẽ được **thực thi ít nhất 1 lần** trước khi kiểm tra điều kiện. Ngoài ra, nó tương tự như vòng lặp while.

Trong bài hôm nay, mình sẽ giới thiệu cho các bạn về cấu trúc vòng lặp thứ 3, và cũng là một cấu trúc vòng lặp được sử dụng nhiều nhất trong C++, đó là **Vòng lặp For trong C++ (For statements).**

Nội dung

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

- VÒNG LẶP WHILE TRONG C++ (While statements)
- VÒNG LẶP DO WHILE TRONG C++ (Do while statements)

Trong bài ta sẽ cùng tìm hiểu các vấn đề:

- Vòng lặp for (for statements)

Vòng lặp for (for statements)

Vòng lặp for là một cấu trúc lặp được sử dụng nhiều nhất trong ngôn ngữ C++, nó hoàn toàn có thể thay thế cho vòng lặp **while**. Lập trình viên thường sử dụng vòng lặp **for** khi **biết trước số lần lặp** của vòng lặp.

Cú pháp của vòng lặp for:

```
for (init-statement; condition-expression; end-expression)
{
   statements;
}
```

Để dễ hình dung, vòng lặp for sẽ **tương đương với vòng lặp while** bên dưới:

C++:

```
{ // note the block here
        init-statement;
        while (condition-expression)
        {
                statements;
                end-expression;
        }
} // variables defined inside the loop go out of scope here
```

Nguyên lý hoạt động của vòng lặp for gồm 3 bước thực thi:

1. ***init*-statement**: phần này có mục đích **định nghĩa và khởi tạo biến**, chỉ được **thực thi 1 lần duy nhất** trong lần lặp đầu tiên.
2. **condition-expression**: phần này gồm các **biểu thức điều kiện**, nếu biểu thức điều kiện đúng, các câu lệnh trong vòng lặp sẽ được thực thi.

3. ***end*-expression**: phần này được thực thi **cuối mỗi lần lặp**, sau khi các câu lệnh trong vòng lặp for được thực thi. Phần này thường có mục đích **tăng hoặc giảm giá trị các biến vòng lặp**. Sau khi thực thi xong, vòng lặp **quay lại kiểm tra điều kiện lặp ở bước 2**.

Ví dụ: bên dưới là vòng lặp for có mục đích xuất các số từ **0** đến **9**.

C++:

```
for (int count = 0; count < 10; ++count)
	cout << count << " ";
```

Chuyển ví dụ trên về vòng lặp while:

C++:

```
{ // dấu ngoặc này là bắt buộc
	int count = 0;
	while (count < 10)
	{
		cout << count << " ";
		++count;
	}
}
```

Output: 0 1 2 3 4 5 6 7 8 9

Trong ví dụ trên, vòng lặp for và while là như nhau, nhưng cách trình bày của vòng lặp while dài và phức tạp hơn. Cặp dấu ngoặc nhọn bên ngoài vòng lặp while là bắt buộc, vì biến **count** có phạm vi vòng lặp, nó sẽ bị hủy khi vòng lặp kết thúc.

Một số ví dụ khác về vòng lặp for:

C++:

```
#include <iostream>
using namespace std;

int main()
{
	for (int i = 10; i > 0; i--) {
		cout << i << ' ';
	}
	// Output: 10 9 8 7 6 5 4 3 2 1

	for (int i = 10; i < 20; i = i + 2) {
		cout << i << ' ';
	}
	// Output: 10 12 14 16 18

	return 0;
}
```

Lược bỏ các thành phần trong vòng lặp for

> Vòng lặp for cho phép **bỏ qua một hoặc tất cả các thành phần** nếu không sử dụng chúng.

Ví dụ: bên dưới là chương trình xuất các số từ **0** đến **9**.

C++:

```
#include <iostream>
using namespace std;

int main()
{
	int count = 0;
	for (; count < 10; )
	{
		cout << count << " ";
		++count;
	}

	return 0;
}
```

Output: 0 1 2 3 4 5 6 7 8 9

Trong ví dụ trên, biến vòng lặp được khởi tạo bên ngoài, và được tăng giá trị bên trong vòng lặp. Phần khởi tạo và cập nhật giá trị của vòng lặp được để trống.

Không như những vòng lặp khác, vòng lặp for **có thể bỏ qua cả 3 thành phần**, nó sẽ tạo ra 1 **vòng lặp vô hạn**:

```
for (;;)
{
   statements;
}
```

Cấu trúc trên tương đương với:

C++:

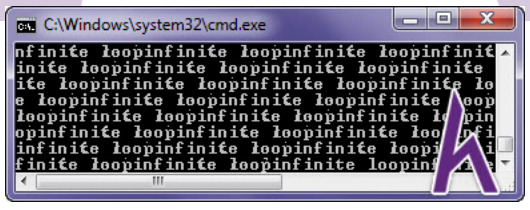
```
while (true)
{
	statements;
}
```

Ví dụ về vòng lặp for vô hạn:

C++:

```
for (;;)
{
	cout << "infinite loop";
}
```

Output:



Nhiều thành phần trong mỗi phần của vòng lặp for

Trong trường hợp vòng lặp for **cần làm việc với nhiều biến**, có **nhiều điều kiện dừng**, hoặc có **nhiều biến cần cập nhật giá trị**, lập trình viên có thể sử dụng **toán tử phẩy ","** để tạo thêm nhiều thành phần trong mỗi phần.

Ví dụ:

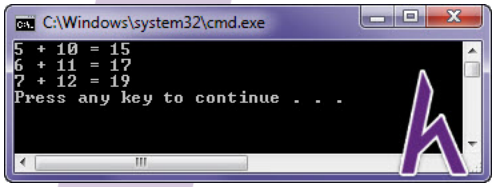
C++:

```
#include <iostream>
using namespace std;

int main()
{
	for (int i = 5, j = 10; i + j < 20; i++, j++)
	{
		cout << i << " + " << j << " = " << (i + j) << '\n';
	}

	return 0;
}
```

Output:



Trong ví dụ trên, vòng lặp for có thể khai báo 2 biến i và j, sau đó cập nhật giá trị j và j sau mỗi lần lặp.

> **Chú ý:** các biến khai báo trong phần ***init*-statement** phải có **cùng kiểu** dữ liệu, cách nhau bởi **dấu phẩy ","**.

Vòng lặp lồng nhau (Nested for loops)

Tương tự như những vòng lặp khác, vòng lặp for có thể lồng vào trong một vòng lặp khác.

Ví dụ:

C++:

```
#include <iostream>
using namespace std;

int main()
{
	for (int i = 0; i <= 5; i++)
	{
		for (int j = 0; j <= i; j++)
		{
			cout << j << " ";
		}
		cout << endl;
	}

	return 0;
}
```

Output:

Kết luận

Qua bài học này, bạn đã nắm rõ về Vòng lặp For trong C++ (For statements). **Vòng lặp for** là một cấu trúc lặp được sử dụng nhiều nhất trong ngôn ngữ C++, nó hoàn toàn có thể thay thế cho vòng lặp **while**. Lập trình viên thường sử dụng vòng lặp **for** khi **biết trước số lần lặp** của vòng lặp.

Trong bài tiếp theo, mình sẽ giới thiệu cho các bạn về 2 từ khóa liên quan mật thiết với cấu trúc vòng lặp, đó là TỪ KHÓA BREAK & CONTINUE.

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**".