Bài 17: BIẾN TĨNH TRONG C++ (STATIC VARIABLES IN C++)

Xem bài học trên website để ủng hộ Kteam: Biến tĩnh trong C++ (Static variables in 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 bài học trước, bạn đã nắm được BIẾN TOÀN CỤC TRONG C++ (Global variables) và những kinh nghiệm khi sử dụng biến toàn cục trong lập trình.

Hôm nay, mình sẽ hướng dẫn về phần **Biến tĩnh trong C++ (Static variables)**.

Nội dung

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

- BIẾN TRONG C++ (Variables in C++)
- BIẾN CỤC BỘ TRONG C++ (Local variables)
- BIẾN TOÀN CỤC TRONG C++ (Global variables)

Trong bài ta sẽ cùng tìm hiểu các vấn đề:

- Tổng quan về biến tĩnh (static variables)
- Khi nào nên sử dụng biến tĩnh

Tổng quan về biến tĩnh (static variables)

Trong bài BIẾN CỤC BỘ TRONG C++ (Local variables), bạn đã biết được:

- **Biến được định nghĩa bên trong một khối lệnh (block)** được gọi là các **biến cục bộ (Local variables).**
- **Biến cục bộ có thời gian tự động**, nghĩa là chúng **được tạo tại thời điểm định nghĩa**, và bị **hủy khi ra khỏi khối lệnh** mà biến đó được định nghĩa

Khi sử dụng **từ khóa "static"** với các biến cục bộ, nó sẽ trở thành **biến tĩnh (static variables)**.

> **Biến tĩnh (static variables)** là biến được tạo ra **duy nhất một lần** khi gọi hàm lần đầu tiên và nó sẽ tiếp tục tồn trong suốt vòng đời của chương trình. Đây là sự khác biệt giữa biến tĩnh và biến cục bộ.

Biến tĩnh (static variables) là loại biến **lưỡng tính**, vừa có tính chất của 1 **biến toàn cục (global variables)**, vừa mang tính chất của 1 **biến cục bộ (local variables)**:

- **Tính chất của biến toàn cục**: biến **không mất đi** khi khối lệnh định nghĩa nó kết thúc, nó vẫn nằm trong vùng nhớ của chương trình và được tự động cập nhật khi khối lệnh đó được gọi lại.
- **Tính chất của biến cục bộ**: biến chỉ có thể được **sử dụng trong khối lệnh** mà nó được khai báo.

Ví dụ: sử dụng biến cục bộ (local variables):

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

// Automatic duration
void doSomeThing()
{
	int value(0); // automatic duration by default
	++value;
```

```
	cout << value << endl;
} // value is destroyed here

int main()
{
	// Local variables
	doSomeThing();
	doSomeThing();
	doSomeThing();

	return 0;
}
```

Output:

Trong chương trình trên, bên trong hàm doSomeThing() sử dụng biến cục bộ (local variables) có thời gian tự động. Nên 3 lần gọi lại hàm doSomeThing() là 3 lần khởi tạo và tăng giá trị cho biến **value**. Nên kết quả gọi hàm trong 3 lần là như nhau.

Ví dụ sử dụng biến tĩnh (static variables):

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

// Static duration
void doSomeThing_static()
{
	// static duration via static keyword.  This line is only executed once.
	static int s_value(0);
	++s_value;
```

```
	cout << s_value << endl;
} // s_value is not destroyed here, but becomes inaccessible

int main()
{
	// Static variables
	doSomeThing_static();
	doSomeThing_static();
	doSomeThing_static();

	return 0;
}
```

Output:

Trong chương trình trên, **s_value** là một biến tĩnh (static variables), nó sẽ được khởi tạo 1 lần duy nhất khi hàm doSomeThing_static() được gọi lần đầu. Nó không bị hủy khi kết thúc hàm, nên mỗi lần gọi hàm sau đó sẽ sử dụng giá trị của **s_value** tại thời điểm hiện tại.

> Theo quy ước, nên **đặt tiền tố "s_" trước các biến tĩnh (static variables)** để dễ dàng phân biệt với những biến khác.

Biến tĩnh (static variables) có hai dạng: liên kết bên ngoài (external linkage) và liên kết nội bộ (internal linkage)

- **Liên kết bên ngoài (external linkage):** Khai báo ở ngoài mọi hàm (biến toàn cục)

```
int g_value;
```

```
void doSomeThing()
{
}
```

- **Liên kết nội bộ (internal linkage):**

  - Khai báo trong hàm với từ khóa static

```
void doSomeThing()
{
    static int s_value(0);
}
```

  - Khai báo ngoài hàm với từ khóa static (biến toàn cục)

```
static int g_value;
void doSomeThing()
{
}
```

Khi nào nên sử dụng biến tĩnh

Sử dụng biến tĩnh **khi có nhu cầu giữ giá trị của biến** trong chương trình.

Ví dụ: ứng dụng vào chương trình có chức năng tạo ID không trùng nhau, bạn có thể sử dụng biến tĩnh để làm thực hiện nó:

```
#include <iostream>
#include <string>
using namespace std;
int generateID()
{
        static int s_id(0);
        ++s_id;
        return s_id;
}
```

```
int main()
{
	int nID1 = generateID();
	string strName1("Kteam");

	int nID2 = generateID();
	string strName2("FreeEducation");

	int nID3 = generateID();
	string strName3("HowKteam.com");

	cout << nID1 << " : " + strName1 << endl;
	cout << nID2 << " : " + strName2 << endl;
	cout << nID3 << " : " + strName3 << endl;

	return 0;
}
```

Output:

Trong chương trình trên, sau mỗi lần gọi hàm generateID(), **s_id** sẽ được tăng lên, nên giá trị trả về sẽ duy nhất và không trùng nhau.

Kết luận

Qua bài học này, bạn đã nắm được Biến tĩnh trong C++ (Static variables).

Ở bài tiếp theo, bạn sẽ được học về ÉP KIỂU NGẦM ĐỊNH TRONG C++ (Static variables).

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**".