Bài: Biến tham chiếu trong C++.(Reference variables)

Xem bài học trên website để ủng hộ Kteam: Biến tham chiếu trong C++.(Reference variables)

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 được giới thiệu các khái niệm liên quan đến CON TRỎ & HẰNG (Pointers and const) trong C++.

Hôm nay, chúng ta sẽ cùng tìm hiểu về **Biến tham chiếu (Reference variables)** trong C++.

Nội dung

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

- BIẾN TRONG C++ (Variables in C++)
- HẰNG SỐ TRONG C++ (Constants)
- TRUYỀN THAM CHIẾU CHO HÀM (Passing Arguments by Reference)
- CON TRỎ TRONG C++ (Pointer)

Trong bài ta sẽ cùng tìm hiểu các vấn đề:

- Biến tham chiếu trong C++
- Tham chiếu dưới dạng bí danh
- Tham chiếu dưới dạng tham số hàm
- Tham chiếu và con trỏ
- Lưu ý khi sử dụng tham chiếu

Biến tham chiếu trong C++

Cho đến bài học này, chúng ta đã biết được 2 loại biến cơ bản trong C++:

- Biến giá trị dùng để chứa dữ liệu trực tiếp (số nguyên, số thực, ký tự, ...).
- Biến con trỏ dùng để chứa địa chỉ (hoặc null).

Các biến này đều được cấp phát bộ nhớ và có địa chỉ cụ thể.

Ngoài ra, C++ cho phép sử dụng loại biến thứ ba là **biến tham chiếu (reference variables)**. So với 2 loại biến nói trên, biến tham chiếu có những đặc điểm sau:

- Biến tham chiếu **không được cấp phát bộ nhớ, không có địa chỉ riêng**.
- Nó dùng làm bí danh cho một biến (kiểu giá trị) nào đó và nó sử dụng vùng nhớ của biến này.

Một tham chiếu được khai báo bằng cách sử dụng **toán tử &** giữa kiểu dữ liệu và tên biến:

C++:

```
int value = 10;
int &ref = value; // ref tham chiếu đến biến value
```

> **Chú ý:** Trong ngữ cảnh này, **dấu &** không có nghĩa là "địa chỉ", nó có nghĩa là "tham chiếu đến".

Tham chiếu dưới dạng bí danh

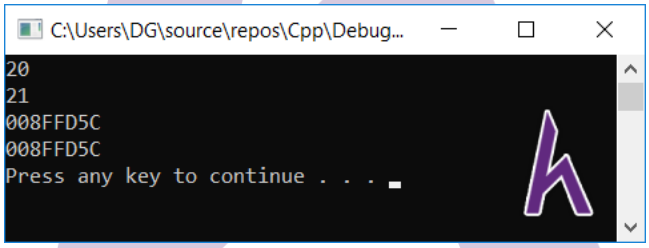
Một tham chiếu hoạt động **như một bí danh** cho đối tượng đang được tham chiếu.

Ví dụ:

C++:

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

int main()
{
	int value = 10;
	int &ref = value; // ref tham chiếu đến biến value

	value = 15; // value = 15
	ref = 20; // value = 20

	cout << value << "\n"; // 20
	++ref;
	cout << value << "\n"; // 21

	// in địa chỉ value và ref
	cout << &value << "\n";
	cout << &ref << "\n";

	system("pause");
	return 0;
}
```

Output:



Trong ví dụ trên, **ref** và **value** được xử lý giống hệt nhau, và chúng có **cùng một địa chỉ** trong bộ nhớ. Điều đó có nghĩa là mọi **thay đổi trên biến ref sẽ làm thay đổi biến value và ngược lại**.

Tham chiếu dưới dạng tham số hàm

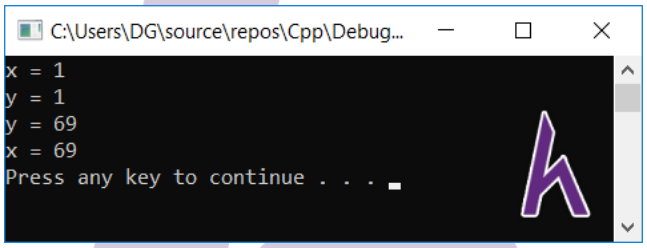
Các tham chiếu thường được sử dụng như các **tham số hàm**.

Tham số tham chiếu hoạt động như một **bí danh cho đối số** và **không có bản sao** của đối số được đưa vào tham số. Điều này làm cho hiệu suất tốt hơn nếu đối số là kiểu dữ liệu có kích thước lớn.

C++:

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

void change(int &y)     // y là biến tham chiếu
{
        cout << "y = " << y << endl;

        y = 69;

        cout << "y = " << y << endl;
}       // y bị hủy

int main()
{
        int x = 1;
        cout << "x = " << x << endl;

        change(x);

        cout << "x = " << x << endl;

        system("pause");
        return 0;
}
```

Output:



Khi đối số **x** được truyền cho hàm, tham số **y** được đặt làm **tham chiếu đến đối số x**. Điều này cho phép hàm thay đổi giá trị của **x** thông qua **y**.

Để hiểu rõ hơn về phần này, bạn có thể xem lại bài TRUYỀN THAM CHIẾU CHO HÀM (Passing Arguments by Reference).

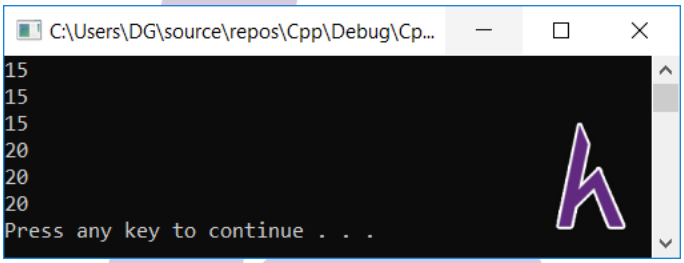
Tham chiếu và con trỏ

Tham chiếu tới một biến giống như một con trỏ được ngầm tham chiếu khi truy cập. Ví dụ:

C++:

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

int main()
{
	int value = 10;
	int *const ptr = &value; // hằng con trỏ
	int &ref = value;

	*ptr = 15;

	cout << *ptr << "\n";
	cout << ref << "\n";
	cout << value << "\n";

	ref = 20;

	cout << *ptr << "\n";
	cout << ref << "\n";
	cout << value << "\n";

	system("pause");
	return 0;
}
```

Output:



Mặc dù tham chiếu và con trỏ có điểm tương đồng, tuy nhiên các tham chiếu nói chung an toàn hơn nhiều so với con trỏ khi thao tác (con trỏ có thể null hoặc ở trạng thái "lơ lửng").

> **Chú ý:** Nếu một vấn đề có thể được giải quyết bằng một tham chiếu hoặc một con trỏ, thì tham chiếu thường được ưu tiên. Con trỏ chỉ nên được sử dụng trong các tình huống mà tham chiếu không hỗ trợ (ví dụ như cấp phát bộ nhớ động).

Lưu ý khi sử dụng tham chiếu

Tham chiếu phải được khởi tạo

Không giống như con trỏ, tham chiếu không **không thể giữ giá trị null**. Vì vậy, tham chiếu phải được khởi tạo khi khai báo.

C++:

```
int value = 10;
int &ref = value; // ok

int &invalidRef; // lỗi: invalid, needs to reference something
```

Tham chiếu và hằng

Tham chiếu đến các giá trị không phải hằng chỉ có thể được khởi tạo với các giá trị không phải là hằng.

C++:

```
int x = 10;
int &ref1 = x; // ok

const int y = 7;
int &ref2 = y; // lỗi

int &ref3 = 6; // lỗi
```

Để tham chiếu đến một đối tượng hằng, ta sử dụng **tham chiếu hằng** bằng cách thêm từ khóa **const**:

C++:

```
const int y = 7;
const int &ref = y; // ok
```

Tham chiếu hằng có thể tham chiếu đến một đối tượng không phải hằng:

C++:

```
int y = 7;
const int &ref = y; // ok
```

Không thể tham chiếu đến một đối tượng khác sau khi khởi tạo

Sau khi được khởi tạo, tham chiếu không thể thay đổi để tham chiếu đến biến khác.

C++:

```
int value1 = 5;
int value2 = 6;

int &ref = value1; // ok
ref = value2; // ref = value1 = 6
// ref luôn tham chiếu đến value1, và ko thể thay đổi
```

Kết luận

Qua bài học này, bạn đã nắm được Biến tham chiếu (Reference variables), một khái niệm được sử dụng rất nhiều trong C++.

Trong bài tiếp theo, mình sẽ giới thiệu cho các bạn khái niệm CON TRỎ VOID 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**".