Bài 43: CON TRỎ NULL TRONG C++ (NULL POINTERS)

Xem bài học trên website để ủng hộ Kteam: Con trỏ NULL trong C++ (NULL pointers)

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 sẻ cho các bạn về CON TRỎ CƠ BẢN TRONG C++ (Pointer). Một số ý chính mà bạn cần nắm: Con trỏ là biến chứa địa chỉ bộ nhớ, có thể được truy cập bằng cách sử dụng toán tử dereference (*). Truy cập một con trỏ rác có thể sụp đổ ứng dụng của bạn.

Hôm nay, chúng ta sẽ cùng tìm hiểu về khái niệm **Con trỏ NULL trong C++ (NULL pointers)**.

Nội dung

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

- BIẾN TRONG C++ (Variables)
- CON TRỎ CƠ BẢN TRONG C++ (Pointer)

Trong bài ta sẽ cùng tìm hiểu các vấn đề:

- Giá trị null và con trỏ null (Null values and null pointers)
- Truy cập con trỏ null
- Macro NULL

- nullptr trong C++11

Giá trị null và con trỏ null

Tương tự như các biến thông thường, con trỏ không được khởi tạo khi khai báo. Nếu con trỏ không được khởi tạo một giá trị, chúng sẽ chứa giá trị rác.

Ngoài địa chỉ vùng nhớ, có một giá trị mà con trỏ có thể giữ được, đó là **giá trị null**. **Null** là một giá trị đặc biệt, điều đó có nghĩa là con trỏ đó chưa trỏ đến địa chỉ nào cả. **Con trỏ đang giữ giá trị null được gọi là con trỏ null (null pointer).**

```
float *ptr{ 0 };  // ptr là 1 con trỏ null

float *ptr2; // ptr2 là con trỏ rác
ptr2 = 0; // ptr2 là 1 con trỏ null
Ta có thể sử dụng một điều kiện để kiểm tra xem một con trỏ có null hay không:
double *ptr{ 0 };

// con trỏ sẽ chuyển thành true nếu nó null, và false nếu nó không null
if (ptr)
	cout << "con trỏ trỏ đến địa chỉ.";
else
	cout << "con trỏ null.";
```

Output: con trỏ null.

> **Chú ý:** Nên khởi tạo con trỏ là **null** nếu nó chưa trỏ đến một địa chỉ cụ thể nào khác.

Truy cập con trỏ null

Trong bài học trước, bạn đã biết được việc truy cập một con trỏ rác sẽ dẫn đến kết quả không xác định. Nếu truy xuất giá trị của con trỏ null, chương trình có thể bị đóng bởi hệ điều hành.

Macro NULL

Ngôn ngữ C (không phải C++) định nghĩa một **macro tiền xử lý** đặc biệt được gọi là NULL, nó có giá trị 0.

```
#define NULL 0
```

Mặc dù đây không phải là một phần kỹ thuật của C++, nhưng nó hoạt động phổ biến trong mọi trình biên dịch C++:

```
double *ptr = NULL; // ptr là 1 con trỏ null
```

Tuy nhiên, về mặt kỹ thuật, macro NULL không phải là kỹ thuật trong C++, vì vậy cần tránh sử dụng nó trong C++.

nullptr trong C++11

Lưu ý rằng **giá trị của 0 không phải là một kiểu con trỏ**, do đó gán 0 cho một con trỏ để biểu thị rằng con trỏ là một con trỏ null là **không nhất quán**.

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

void doSomething(double *ptr)
{
	// ptr sẽ chuyển thành true nếu nó null, và false nếu nó không null
	if (ptr)
		std::cout << "You passed in " << *ptr << '\n';
	else
		std::cout << "You passed in a null pointer\n";
}

int main()
{
	doSomething(0); // truyền 0 có thể gây nhầm lẫn rằng tham số hàm là số nguyên
```

```
        return 0;
}
```

Để giải quyết những vấn đề này, C++11 giới thiệu một từ khóa mới có tên **nullptr**. **nullptr** là một **hằng số rvalue**, giống như các từ khóa boolean **true** và **false**.

```
int *ptr = nullptr; // ptr là 1 con trỏ null
doSomething(nullptr); // truyền con trỏ null vào hàm
```

Ngoài ra, C++11 còn định nghĩa một kiểu dữ liệu **nullptr_t**, **nullptr_t chỉ có thể lưu trữ giá trị nullptr**. Nhưng nó thường chỉ được sử dụng trong những trường hợp hiếm hoi, bạn hãy tìm hiểu thêm và chia sẽ trong comment bên dưới nhé.

Kết luận

Qua bài học này, bạn đã nắm được cơ bản về Con trỏ NULL trong C++ (NULL pointers). Lưu ý rằng ta nên khởi tạo con trỏ là null nếu nó chưa trỏ đến một địa chỉ cụ thể nào khác nhé.

Trong bài tiếp theo, mình sẽ giới thiệu cho các bạn về những vấn đề xung quanh CON TRỎ & MẢNG TRONG C++ (Pointers and arrays).

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**".