

Bài 53: TRUYỀN ĐỊA CHỈ CHO HÀM (PASSING ARGUMENTS BY ADDRESS)

Xem bài học trên website để ủng hộ Kteam: [Truyền địa chỉ cho hàm \(Passing arguments by address\)](#)

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 một cách để sử dụng mảng động mà không cần thao tác quá phức tạp bằng con trỏ, đó là [LỚP DỮ LIỆU std::Vector](#).

Hôm nay, sau khi đã nắm được khái niệm con trỏ trong những bài học trước, chúng ta sẽ cùng quay lại phần hàm trong C++, cụ thể là vấn đề **Truyền địa chỉ cho hàm (Passing arguments by address)**.

Nội dung

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

- [CƠ BẢN VỀ HÀM VÀ GIÁ TRỊ TRẢ VỀ \(basics of functions and return values\)](#)
- [TRUYỀN GIÁ TRỊ CHO HÀM \(passing arguments by value\)](#)
- [TRUYỀN THAM CHIẾU CHO HÀM \(passing arguments by reference\)](#)
- [CON TRỎ CƠ BẢN TRONG C++ \(pointers\)](#)

Trong bài ta sẽ cùng tìm hiểu các vấn đề:

- Truyền địa chỉ cho hàm (Passing arguments by address)
- Truyền địa chỉ cho hàm bằng giá trị
- Truyền địa chỉ cho hàm bằng tham chiếu
- Tổng kết về phương pháp truyền địa chỉ cho hàm

Truyền địa chỉ cho hàm (Passing arguments by address)

Hiện tại, mình đã giới thiệu 2 cách truyền đối số cho một hàm trong C++ là: **truyền giá trị (Call by value)** và **truyền tham chiếu (Call by reference)**. Trong bài học này, mình sẽ giới thiệu về **phương pháp truyền địa chỉ cho hàm**.

Truyền địa chỉ cho hàm là truyền địa chỉ của biến đối số chứ không phải giá trị biến đối số. Vì **đối số là một địa chỉ, tham số hàm phải là một con trỏ**. Sau đó, hàm có thể truy cập hoặc thay đổi giá trị được trỏ đến.

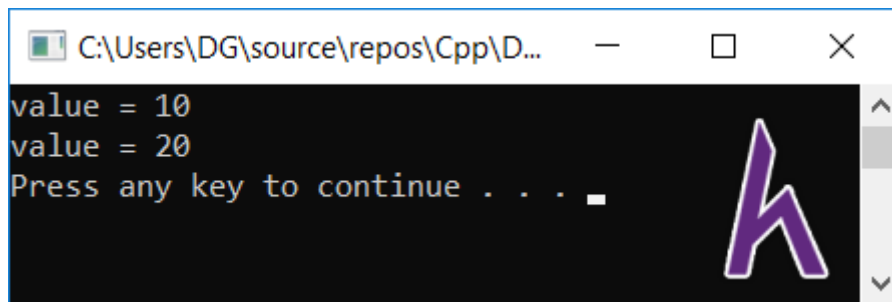
```
#include <iostream>
using namespace std;

void foo(int *ptr)
{
    *ptr = 20;
}

int main()
{
    int value = 10;

    cout << "value = " << value << '\n';
    foo(&value);
    cout << "value = " << value << '\n';

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

Output:

Trong ví dụ trên, hàm **foo()** đã thay đổi giá trị của đối số (biến value) thông qua tham số hàm là con trỏ **ptr** (địa chỉ biến value).

Truyền địa chỉ cho hàm thường được sử dụng với mảng. Ví dụ:

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

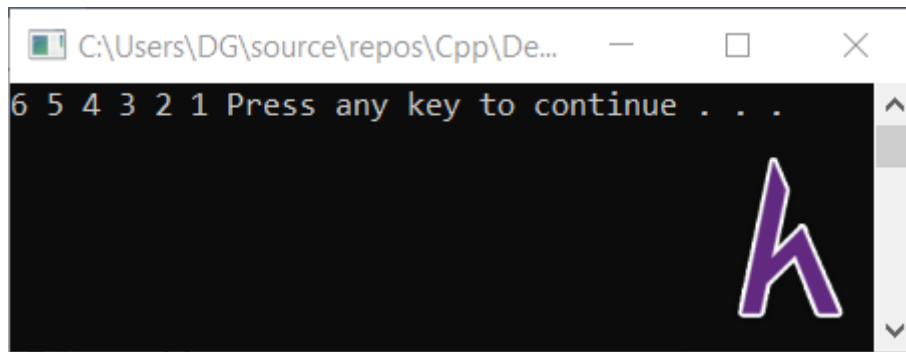
void printArray(const int *array, int length)
{
    // kiểm tra con trỏ null
    if (!array)
        return;

    for (int index = 0; index < length; ++index)
        cout << array[index] << ' ';
}

int main()
{
    int array[6] = { 6, 5, 4, 3, 2, 1 };
    printArray(array, 6);

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

Output:



Như ví dụ trên, bạn có thể sử dụng tham số hằng con trỏ trong trường hợp hàm không có mục đích thay đổi giá trị đối số.

Truyền địa chỉ cho hàm bằng giá trị

Khi truyền một con trỏ tới một hàm theo địa chỉ, **giá trị của con trỏ** (địa chỉ mà nó trỏ tới) sẽ được **sao chép** từ **đối số** sang **tham số hàm**.

Nói cách khác, nó là phương pháp truyền giá trị. Nếu bạn **thay đổi giá trị** của tham số hàm, bạn **chỉ thay đổi một bản sao**. Do đó, đối số con trỏ ban đầu sẽ không bị thay đổi.

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

// tempPtr là bản sao của ptr
void setToNull(int *tempPtr)
{
    // trỏ tempPtr đến một vị trí khác, không phải thay đổi giá trị tempPtr trỏ
    // tới
    tempPtr = nullptr;
}

int main()
{
    int value = 10;
    int *ptr = &value;

    cout << *ptr << "\n"; // 10

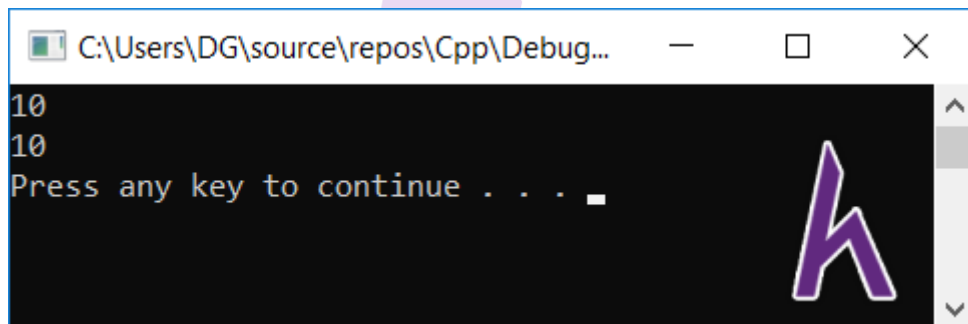
    // tempPtr là bản sao của ptr
```

```
setToNull(ptr);

if (ptr)
    cout << *ptr << "\n"; // 10
else
    cout << " ptr is null";

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

Output:



Chú ý:

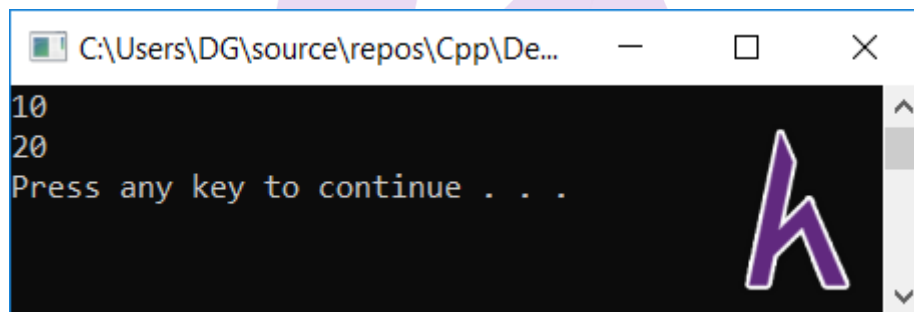
- Khi truyền một đối số theo địa chỉ, tham số hàm nhận được một bản sao của địa chỉ từ đối số. Tại thời điểm này, tham số hàm và đối số đều trỏ đến cùng một giá trị.
- Nếu thay đổi giá trị được tham số hàm trỏ đến, điều đó sẽ thay đổi giá trị mà đối số đang trỏ đến, vì cả tham số hàm và đối số đều **trỏ đến cùng một giá trị**.
- Nếu tham số hàm được gán một địa chỉ khác, điều đó **sẽ không ảnh hưởng** đến đối số, vì tham số hàm chỉ là bản sao của đối số.

Xem ví dụ bên dưới để hiểu rõ hơn:

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

void setValue(int *tempPtr)
{
    *tempPtr = 20; // thay đổi giá trị tempPtr và ptr trỏ tới
```

```
}  
  
int main()  
{  
    int value = 10;  
    int *ptr = &value;  
  
    cout << *ptr << "\n"; // 10  
  
    // tempPtr là bản sao của ptr  
    setValue(ptr);  
  
    if (ptr)  
        cout << *ptr << "\n"; // 20  
    else  
        cout << " ptr is null";  
  
    system("pause");  
    return 0;  
}
```

Output:

Truyền địa chỉ cho hàm bằng tham chiếu

Chúng ta có thể **truyền địa chỉ cho hàm bằng tham chiếu** để thay đổi địa chỉ mà đối số trỏ đến từ bên trong hàm.

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

// tempPtr tham chiếu đến con trỏ ptr, mọi thay đổi trên tempPtr sẽ làm thay đổi ptr
void setToNull(int *&tempPtr)
{
    tempPtr = nullptr;
}

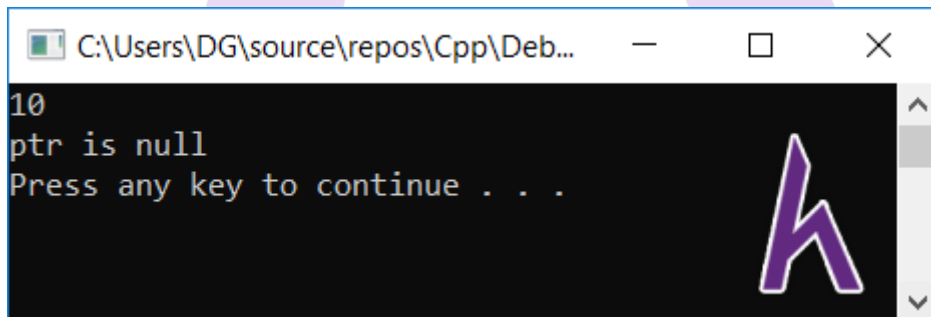
int main()
{
    int value = 10;
    int *ptr = &value;

    cout << *ptr << "\n"; // 10

    // tempPtr là tham chiếu đến ptr
    setToNull(ptr);

    if (ptr)
        cout << *ptr << "\n";
    else
        cout << "ptr is null" << "\n";

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

Output:

Tổng kết về phương pháp truyền địa chỉ cho hàm

Ưu điểm:

- Hàm có thể thay đổi giá trị của các đối số. Ngược lại, bạn có thể sử dụng tham chiếu hằng (const reference) nếu không muốn hàm thay đổi giá trị đối số.
- Không mất thời gian và bộ nhớ để sao chép giá trị của đối số vào tham số của hàm, ngay cả khi được sử dụng với các dữ liệu có cấu trúc hoặc lớp.
- Hàm có thể trả về nhiều giá trị thông qua tham chiếu.

Nhược điểm:

- Đối số phải là biến thông thường (có địa chỉ trong bộ nhớ), vì hằng hay biểu thức không có địa chỉ.
- Cần kiểm tra null trước khi sử dụng tham số.

Khi nào nên sử dụng:

- Khi đối số là kiểu mảng.
- Khi có nhu cầu thay đổi giá trị của đối số sau khi thực hiện hàm.

Khi nào không nên sử dụng:

- Khi đối số là kiểu cấu trúc (struct) hoặc lớp (class) (sử dụng truyền tham chiếu).
- Khi đối số có kiểu dữ liệu cơ bản (sử dụng truyền giá trị).

Chú ý: Vì truyền tham chiếu thường an toàn hơn so với truyền địa chỉ, nên truyền tham chiếu thường được ưu tiên trong hầu hết các trường hợp.

Kết luận

Qua bài học này, bạn đã nắm được phương pháp Truyền địa chỉ cho hàm (Passing arguments by address). Và những ưu điểm, nhược điểm, khi nào nên và không nên sử dụng của phương pháp trên.

Trong bài tiếp theo, chúng ta sẽ cùng tìm hiểu [CÁC KIỂU TRẢ VỀ CỦA HÀM \(Returning values by value, reference, and address\)](#) 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**".

