Chủ đề: Con trỏ trong C++

1. **Khái niệm về con trỏ**

* Trong C++, con trỏ là 1 biến đặc biệt, có vai trò lưu trữ địa chỉ của 1 biến khác trong bộ nhớ, cùng kiểu với nó
* Con trỏ và biến mà con trỏ đó lưu trữ địa chỉ phải có cùng kiểu dữ liệu
* Cú pháp khai báo:

<kiểu dữ liệu> \*<tên con trỏ>

VD: int \*a,;

Double \*b;

SINHVIEN \*x; // đây là con trỏ đối tượng, sẽ học ở phần môn lập trình hướng đối tượng

* Mỗi con trỏ chỉ trỏ được tới 1 biến, mỗi biến có thể được trỏ bằng nhiều con trỏ

1. **Một số thao tác cơ bản trên con trỏ**
2. **Lấy địa chỉ của biến đặt vào con trỏ**

Giả sử có biến con trỏ x lưu địa chỉ của biến a (2 biến cùng mang kiểu dữ liệu int). Cú pháp lấy địa chỉ của biến a lưu vào con trỏ x:

x = &a

Khi đó ta nói: Con trỏ x trỏ tới biến a

Toán tử & có vai trò lấy địa chỉ của 1 biến trong bộ nhớ

1. **Gán con trỏ cho con trỏ**

Khi gán con trỏ này và con trỏ kia, 2 con trỏ sẽ lưu trữ cùng 1 địa chỉ, tức là cùng trỏ vào 1 biến

VD: int \*p, \*q;

p = &a;

q = p;

* p và q là 2 con trỏ cùng trỏ vào biến a

1. **Thao tác gián tiếp giữa con trỏ và biến**
2. **Giải tham chiếu**

* Khi khai báo con trỏ, dấu \* sẽ có vai trò xác định xem đó là con trỏ hay biến thông thường. Tuy nhiên, trong quá trình thao tác, dấu \* còn được gọi là giải tham chiếu. Khi sử dụng biến con trỏ có giải tham chiếu, nó sẽ trả về giá trị của biến mà con trỏ đó chỉ tới

VD: int \*p;

Int a = 10;

p = &a;

cout << \*p;

return 0;

* Khi đó ở dòng cout << \*p, dấu \* có vai trò là giải tham chiếu, và sẽ in ra màn hình giá trị của biến mà con trỏ p trỏ tới, là biến a. Mà biến a có giá trị là 10 -> In ra màn hình là 10

1. **Thao tác với dữ liệu thông qua con trỏ:**

Qua việc sử dụng giải tham chiếu, chúng ta có thể thao tác với các dữ liệu thông qua con trỏ

VD: Cho chương trình sau :

#include <iostream>

using namespace std;

int main(){

int \*p;

int a;

p = &a;

cin >> \*p;

cout << \*p;

return 0;

}

Ở chương trình này, chúng ta sử dụng con trỏ p để gián tiếp nhập dữ liệu cho biến a, thông qua sử dụng giải tham chiếu

1. **Kiểu dữ liệu con trỏ**

* Kiểu dữ liệu con trỏ là kiểu dữ liệu đặc biệt. Khi khai báo một biến mang kiểu dữ liệu con trỏ, nó sẽ mặc định là 1 con trỏ mà không cần sử dụng toán tử \*
* Cú pháp khai báo:

<tên kiểu dữ liệu>\* < tên biến>

(Dấu \* được viết cạnh tên của kiểu dữ liệu)

VD: int\* a;

double\* t;

* Khi đó 2 biến a và t sẽ là con trỏ, và khi thao tác chúng ta thao tác tương tự như những con trỏ thường khác

Bài tập áp dụng:

Bài 1: Nhập 2 số nguyên dương a và b từ bàn phím, sử dụng 2 con trỏ p,q lưu địa chỉ lần lượt của a và b để thao tác. Tính tổng 2 số đó và in ra màn hình (có thể sử dụng kiểu dữ liệu con trỏ hoặc không )

Bài làm:

#include <iostream>

using namespace std;

int main(){

int \*p;

int \*q;

int a,b;

p = &a;

q = &b;

cin >> \*p >> \*q;

cout << \*p + \*q;

return 0;

}

Bài 2: Nhập 3 số nguyên dương a, b và c từ bàn phím, sử dụng 3 con trỏ x,y,z lần lượt lưu địa chỉ của a,b,c để thao tác. Tính trung bình cộng của 3 số đó, in ra màn hình số lớn nhất và nhỏ nhất trong 3 số đó ra màn hình

Bài 3: Nhập 3 số nguyên dương a,b và c từ bàn phím, sử dụng 3 con trỏ x,y,z lần lượt lưu địa chỉ của a,b,c để thao tác. Cho biết a,b,c có phải 3 cạnh của 1 tam giác vuông hay không ?. In ra màn hình YES nếu đúng, NO nếu sai

Bài 4: Nhập 3 số nguyên dương a,b,c từ bàn phím là 3 cạnh của 1 tam giác, sử dụng 3 con trỏ x,y,z lần lượt lưu địa chỉ của a,b,c để thao tác. Tính diện tích của tam giác đó theo công thức Hê-rông (kết quả làm tròn đến chữ số thập phân thứ 2).

Lưu ý: Phải xét điều kiện 3 cạnh đó có tạo nên 1 tam giác hay không. Nếu không tạo thành 1 tam giác, yêu cầu người dùng nhập lại.

Bài 5: Nhập 4 số nguyên dương a,b,c từ bàn phím, sử dụng 3 con trỏ x,y,z lần lượt lưu địa chỉ của a,b,c để thao tác.. Sắp xếp 3 biến đó theo thứ tự tăng dần và in ra màn hình kết quả, mỗi số cách nhau bởi 1 dấu cách

1. **Con trỏ và mảng**
2. **Đặt vấn đề**

* Trước đây ở bài mảng, chúng ta cấp phát 1 mảng (hay còn gọi là cấp phát tĩnh), với 1 khối lượng bộ nhớ nhất định trong mảng đó và thực hiện các thao tác và công việc với các phần tử

VD: int a[100];

Việc cấp phát này sẽ dễ dàng hơn, tuy nhiên sẽ gặp phải 1 vấn đề: Trong quá trình thao tác, khối lượng cấp phát cho mảng đó sẽ cố định và không thể thay đổi(ví dụ cấp phát 100 ô nhớ thì trong quá trình thao tác, chúng ta không thể xin cấp phát thêm ô nhớ, mà phải tạo hẳn 1 mảng khác, có kích cỡ lớn hơn để phục vụ lợi ích của mình). Điều này sẽ gây bất tiện và thiếu linh hoạt trong quản lý bộ nhớ và chương trình

* Việc sử dụng mảng và con trỏ(hay còn gọi là cấp phát động) sẽ khiến người lập trình chủ động trong việc cấp phát bộ nhớ trong chương trình và việc quản lý bộ nhớ sẽ tối ưu hơn. Chúng ta có thể cấp phát lại, thêm, bớt “thủ công”, dựa trên nhu cầu của chúng ta

1. **Cấp phát**

* Để cấp phát mảng bằng con trỏ, chúng ta sử dụng từ khóa new

+ Cấp phát 1 ô nhớ (cấp phát biến thông thường, cách này sử dụng tương đối ít)

<Tên con trỏ> = new <Kiểu của con trỏ>;

+ Cấp phát một khối nhớ (nhiều ô nhớ, memory, block)

<Tên con trỏ> = new <Kiểu của con trỏ>[n];