LÝ THUYẾT CON TRỎ

1. Giới thiệu về con trỏ:

Con trỏ trong ngôn ngữ lập trình C/C++ được coi là một phần mạnh mẽ và tương đối khó khi sử dụng.

Con trỏ cho phép xây dựng hàm truyền tham chiếu (pass-by-reference) giúp thay đổi giá trị của một đối số sau khi hàm kết thúc. Ngoài ra các cấu trúc dữ liệu cấp phát động như danh sách liên kết, cây nhị phân được dựa trên con trỏ.

2. Biến kiểu con trỏ:

Con trỏ cũng là một biến, nhưng khác với các biến thông thường thì giá trị mà biến con trỏ lưu là một địa chỉ của biến khác. Con trỏ và biến mà con trỏ quản lý phải có cùng kiểu dữ liệu.

Khi khai báo biến con trỏ, để phân biệt nó với biến thông thường ta bổ sung thêm dấu *. Dấu * này có thể đặt ở phía tên biến hoặc đặt gần ở kiểu dữ liệu.

Ví dụ:

```
#include <bits/sdtc++.h>
using namespace std;
int main(){
    int x; // biến int
    int *a; // biến con trỏ kiểu int
    double* b; // biến con trỏ kiểu double
    char* c; // biến con trỏ kiểu char
}
```

3. Gán giá trị cho con trỏ, toán tử địa chỉ và toán tử giải tham chiếu:

Toán tử địa chỉ & : Trả về địa chỉ của một biến trong bộ nhớ.

Toán tử giải tham chiếu * : Trả về giá trị của biến mà con trỏ đang tham chiếu tới.

3. Gán giá trị cho con trỏ, toán tử địa chỉ và toán tử giải tham chiếu:

Ví giá trị của con trỏ là một địa chỉ của biến khác, nên khi không khởi tạo giá trị cho con trỏ thì nó được gọi là con trỏ NULL, còn khi gán địa chỉ của 1 biến nào đó cho con trỏ thì ta nói con trỏ đang quản lý hoặc trỏ tới biến đó.

```
#include <bits/sdtc++.h>
using namespace std;
int main(){
    int a = 100;
    int *ptr; // đang là con trỏ NULL
    cout << &a << endl; // địa chỉ của biến a
    ptr = &a; // cho ptr trỏ tới a
    cout << *ptr << endl; // giải tham chiếu để truy cập a thông qua ptr
}
```

OUTPUT

```
0x6ffe1c
100
```

3. Gán giá trị cho con trỏ, toán tử địa chỉ và toán tử giải tham chiếu:

Sau khi cho con trỏ quản lý 1 biến thì ta có thể thông qua toán tử giải tham chiếu để thay đổi giá trị của biến mà con trỏ đang quản lý. Trong ví dụ dưới việc bạn sử dụng a hay *ptr có ý nghĩa như nhau.

```
#include <bits/sdtc++.h>
using namespace std;
int main(){
    int a = 100;
    int *ptr; // đang là con trỏ NULL
    ptr = &a; // cho ptr trỏ tới a
    cout << "Gia tri cua bien ma ptr quan ly :" << *ptr << endl;
    *ptr = 500; // thay đổi luôn giá trị của a
    cout << "Gia tri cua bien a :"<< a << endl;
}
```

OUTPUT

```
Gia tri cua bien ma ptr quan ly :100
Gia tri cua bien a :500
```

3. Gán giá trị cho con trỏ, toán tử địa chỉ và toán tử giải tham chiếu:

Cần phân biệt rõ 2 dấu *

Khi khai báo kiểu con trỏ, ta thêm dấu * trước tên biến để chỉ ra rằng đây là biến con trỏ.

```
int *ptr;
```

Dấu * trước tên con trỏ ở những câu lệnh sau câu lệnh khai báo là toán tử giải tham chiếu.

```
*ptr = ...;
```

4. Truyền tham chiếu (Pass-by-reference):

Trong bài này chúng ta học cách truyền tham chiếu thông qua con trỏ. Khi gọi hàm có tham số là con trỏ, các bạn phải truyền địa chỉ của biến vào làm đối số cho hàm.

Ví dụ:

```
#include <bits/sdtc++.h>
using namespace std;
void change(int *n){
    *n += 100;
}

int main(){
    int a = 100;
    change(&a);
    cout << a << endl;
}
```

OUTPUT

200

Hoán vị giá trị của 2 biến

```
#include <bits/sdtc++.h>
using namespace std;
void hoanvi(int *a, int *b){
    int tmp = *a;
    *a = *b;
    *b = tmp;
}

int main(){
    int a = 100, b = 200;
    hoanvi(&a, &b);
    cout << a << " " << b << endl;
}
```

OUTPUT

200 100

5. Mảng và con trỏ:

Thực chất thì mảng là một hằng con trỏ, bạn có thể in ra giá trị của mảng, khi đó giá trị của mảng chính là địa chỉ của phần tử đầu tiên trong mảng.

```
int a = {1, 2, 3, 4, 5);
```

a	1	2	3	4	5
Con trỏ	a	a + 1	a + 2	a + 3	a + 4

Phần tử có chỉ số X trong mảng được quản lý bởi con trỏ a + X

```
#include <bits/sdtc++.h>
using namespace std;
int main(){
    int a[] = {1, 2, 3, 4, 5};
    cout << a << endl;
    for(int i = 0; i < 5; i++){
        cout << &a[i] << " ";
    }
    cout << endl;
    for(int i = 0; i < 5; i++){
        cout << (a + i) << " ";
    }
}
```

OUTPUT

```
0x6ffde0
0x6ffde0 0x6ffde4 0x6ffde8 0x6ffdec 0x6ffdf0
0x6ffde0 0x6ffde4 0x6ffde8 0x6ffdec 0x6ffdf0
```

5. Mảng và con trỏ:

Có thể thao tác với mảng thông qua con trỏ.

Ví dụ:

```
#include <bits/sdtc++.h>
using namespace std;
int main(){
    int n; cin >> n;
    int a[n];
    for(int i = 0; i < n; i++){
        cin >> *(a + i);
    }
    for(int i = 0; i < n; i++){
        cout << *(a + i) << " ";
    }
}
```

5. Mảng và con trỏ:

Sau khi cho một con trỏ trỏ vào phần tử đầu tiên trong mảng, bạn có thể sử dụng con trỏ này như mảng.

Ví dụ:

```
#include <bits/sdtc++.h>
using namespace std;
int main(){
    int n; cin >> n;
    int a[n];
    int *b = a;
    for(int i = 0; i < n; i++){
        cin >> b[i];
    }
    for(int i = 0; i < n; i++){
        cout << b[i] << " ";
    }
}
```

5. Mảng và con trỏ:

Con trỏ và toán tử:

Khi con trỏ quản lý một mảng, các bạn có thể dùng các toán tử như ++, --, +, - để di chuyển con trỏ tới vị trí mong muốn trong mảng.

```
#include <bits/sdtc++.h>
using namespace std;
int main(){
    int a[] = {1, 2, 4, 5, 6, 9, 8, 7};
    int *ptr = a;
    ++ptr;
    cout << *ptr << endl;
    for(int *x = a; x != (a + 8); ++x){
        cout << *x << " ";
    }
}
```

OUTPUT

```
2
1 2 3 4 5 6 7 8 9
```