Trình bày lưu ý khi sử dụng hàm.

1. Tham số truyền vào hàm

Có hai loại tham số bao gồm tham chiếu ( hoặc gọi là tham biến) và tham trị

Với tham chiếu, khi các tham số là tham chiếu bị thay đổi trong hàm thì biến từ tên hàm cũng bị thay đổi theo. Khi dùng tham biến, tham số truyền vào tên hàm bắt buộc phải là biến

Với tham trị, các tham số là trị cho dù bị thay đổi trong hàm nhưng tham số truyền vào từ tên hàm không bị thay đổi theo. Khi dùng tham trị, ta có thể truyền giá trị hoặc biến chứa giá trị nào đó

Ví dụ

Void ChangeNumber(int a) {  
a = 4;  
}

ChangeNumber(2);

Int x = 10;

ChangeNumber(x);

Giá trị a bên trong hàm ChangeNumber bị thay đổi nhưng nó không bị ảnh hưởng bởi biến x vì nó chỉ truyền giá trị của biến x vào thôi

Tương tự như ví dụ trên nếu ta chỉnh sửa hàm ChangeNumber 1 chút:

Void ChangeNumber(int \*a) {

a = 10;

}

ChangeNumber(2); => lỗi

Int x = 20;

ChangeNumber(x) => kết quả x = 10;

Đối với tham số là mảng, nó sẽ tự động thay đổi các giá trị bên trong mảng, vì bản chất, biến mảng là một biến con trỏ dùng để quản lí 1 dãy ô nhớ liên tục trên thanh RAM

Ví dụ

void arrayX(int \*a) {

a[0] = 10;

}

Hoặc

void arrayX(int a[]) {

a[0] = 10;

}

// Cả 2 phương pháp viết như này về mặt ý nghĩa là giống nhau

Khi ta có 1 mảng int b[] = {1,2}; và ta truyền mảng b này vào hàm arrayX(b)

Do trong hàm thay đổi giá trị của phần tử a[0] nên b[0] cũng bị thay đổi theo

Đối với tham số là con trỏ hàm

Con trỏ hàm là con trỏ quản lí địa chỉ lưu trữ hàm đó, con trỏ hàm nó giống như 1 tham số bất kì…

Lợi ích của con trỏ hàm là có thể sử dụng hàm thông qua tham số mà k cần đến gọi hàm thông qua tên hàm được tạo. 1 ví dụ cực hay cho bài này là bài toán sắp xếp tuyến tính

Ví dụ

Void Sort(int \*a, int n) {

Int temp;

For(int I = 0; I < n - 1;i++) {

For(int j= i+1;j<n;j++ {

If(a[i] > a[j] {

Temp = a[i]

a[i] = a[j]

a[j] = temp;

}

}

}

}

Đây là hàm sắp xếp sử dụng phương pháp sắp xếp tuyến tính, tuy nhiên ở hàm này ta thấy rằng thuật toán sắp xếp này là tăng dần

Và nếu ta muốn sử dụng hàm sắp xếp giảm giành thì ta lại copy hàm trên và sửa ngược lại với điều kiện từ (a[i] > a[j]) thành (a[i] < a[j])

Việc tạo ra 1 hàm mới như thế và chỉ khác điều kiện khiến cho code rất dài và con trỏ hàm sẽ thực hiện được điều này

Ban đầu ta tạo ra 2 hàm

Bool TangDan(int a, int b) {

Return a > b;

}

Bool GiamDan(int a,int b) {

Return a < b;

}

Sau đó ở trên hàm sort ta cần sửa lại tham số truyền vào

Void Sort(int \*a, int n, bool (\*compare)(int, int)) {

…

If(compare(a[i],a[j) {

// hoán đổi giá trị

}

…

}

Và trong lời gọi hàm ta có thể truyền ham TangDan hoặc GiamDan vào hàm sort để sort theo ý muốn

Sort(a,n,TangDan);

Sort(a,n,GiamDan);

Trình bày lưu ý khi sử dụng getline()

Vấn đề sinh ra từ lệnh getline

Trong C++, chúng ta thường sử dụng cin để thực hiện việc nhập giá trị nào đó từ bàn phím, tuy nhiên nó chỉ lấy các kí tự như số, chữ, hoặc các kí tự đặc biệt và bỏ qua kí tự dấu cách và enter

Chính vì thế việc lấy một chuỗi kí tự gặp khá nhiều khó khăn, từ đó getline được ra đời.

Một số lưu ý khi dùng getline

Để sử dụng hàm này, chúng ta cần phải xóa bộ nhớ đệm bởi trong trường hợp khi ta dùng câu lệnh cin, giá nó chỉ lấy giá trị nhập vào gán cho biến trước đó và trong bộ nhớ đệm còn lại dấu enter. Khi getline đọc ra từ bộ nhớ đệm thấy dấu enter, nó sẽ gán giá trị rỗng cho biến chuỗi. chính vì thế chúng ta cần xóa bộ nhớ đệm trước khi sử dụng getline

Để xóa bộ nhớ đệm ta sử dụng hàm ignore() từ đối tượng cin. => cin.ignore()

Trình bày lưu ý khi sử dụng kdl struct.

Một lưu ý cực kì quan trọng là việc sắp xếp giá trị phù hợp với kiểu struct

Thông thường chúng ta nên sắp xếp các keys bên trong theo kích thước của biến mà ta sử dụng

Vì khi ta sắp xếp lộn xộn, nó sẽ chiếm một vùng nhớ lớn và đáng kể

Trong struct, chúng ta có thể tạo con trỏ của struct đó là 1 thành phần của kiểu dữ liệu struct

Việc tạo con trỏ struct đó chứa bên trong struct sẽ giúp chúng ta tạo ra được danh sách liên kết, đây là một trong những cấu trúc dữ liệu được sử dụng phổ biến.

Bai 1:

#include <iostream>

using namespace std;

int main()

{

int a[] = { -4, 5, 0, 6, - 1, 0, 8, 3, - 11, 15, 2, 1};

int n = 12;

cout << "Nhap so luong phan tu: " << endl;

cin >> n;

for (int i = 0; i < n; i++) {

cout << "Nhap phan tu thu " << i << ": ";

cin >> a[i];

}

int temp;

for (int i = 0; i < n - 1; i++) {

for (int j = i+1; j < n; j++) {

if (a[i] > a[j]) {

temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

}

int i = 0;

int b[100];

while ((a[i] < 0) && i < n) {

b[i++] = a[i];

}

for (int j = i-1; j >=0; j--) {

cout << b[j] << endl;

}

for (; i < n; i++) {

cout << a[i] << endl;

}

return 0;

}

#include <iostream>

using namespace std;

#include <math.h>

int main()

{

int a[] = { 24, 45, 23, 13, 43, - 12 };

int n = 6;

cout << "Nhap so luong phan tu: " << endl;

cin >> n;

for (int i = 0; i < n; i++) {

cout << "Nhap phan tu thu " << i << ": ";

cin >> a[i];

}

cout << "Nhap x: ";

int x = 15;

cin >> x;

int min = a[0];

int max = a[0];

for (int i = 1; i < n; i++) {

if (a[i] < min) {

min = a[i];

}

else if (max < a[i]) {

max = a[i];

}

}

if ((abs(min) - abs(x)) > (abs(max) - abs(min))) {

cout << min;

}

else if ((abs(min) - abs(x)) < (abs(max) - abs(min))) {

cout << max;

}

else {

cout << min << endl;

cout << max << endl;

}

return 0;

}