TÓM TẮT NỘI DUNG + VÍ DỤ VỀ HÀM VÀ THAM SỐ

1. **Khái niệm hàm**

* Hàm là một đoạn chương trình độc lập:

+ Thực hiện một công việc xác định dựa trên các tham số.

+ Có thể được thực hiện nhiều lần.

* Ví dụ với các hàm được khai báo trong thư viện cmath:
* Hàm sqrt(n) trả về kết quả căn bậc hai của số thực n (công việc: tính căn bậc hai, tham số truyền vào là n).
* Hàm pow(a, b) trả về kết quả (công việc: tính kết quả của a mũ b dựa trên 2 tham số a, b được truyền vào).
* Có thể gọi lại các hàm nhiều lần, bất kì khi nào cần sử dụng. Ví dụ, tính tổng :

#include<iostream>

#include<cmath>

using namespace std;

int main()

{

double ketqua = sqrt(3) + sqrt(7) + sqrt(12);

cout << ketqua;

}

* Mục đích xây dựng hàm:
* Lập trình theo module (tách chương trình thành các hàm, mỗi hàm thực hiện các công việc cụ thể khác nhau, liên kết các hàm với nhau để tạo thành chương trình).
* Tránh lặp lại một thao tác.

Ví dụ: viết chương trình cho phép nhập vào 3 cặp số nguyên a – b, c – d, e – f. Xuất ra giá trị lớn nhất trong mỗi cặp số trên. (chẳng hạn nhập vào a = 5, b = 7, c = 4, d = 12, e = 8, f = 4 thì sẽ xuất ra màn hình các giá trị: b d e vì a < b, c < d, e > f, xuất ra giá trị lớn hơn trong mỗi cặp số).

* Khi không khai báo hàm, chương trình sẽ viết như sau:

#include<iostream>

#include<cmath>

using namespace std;

int main()

{

int a, b, c, d, e, f;

cout << "Nhap a, b, c, d, e, f: ";

cin >> a >> b >> c >> d >> e >> f;

if (a > b)

cout << a;

else

cout << b;

if (c > d)

cout << c;

else

cout << d;

if (e > f)

cout << e;

else

cout << f;

}

* Đoạn chương trình trên lặp đi lặp lại cùng 1 thao tác để tìm được số lớn hơn trong mỗi cặp số, ta viết riêng 1 hàm tìm số lớn hơn trong 2 số nguyên như sau:

#include<iostream>

#include<cmath>

using namespace std;

int TimMax(int a, int b)

{

int max;

if (a > b)

max = a;

else

max = b;

return max;

}

int main()

{

int a, b, c, d, e, f;

cout << "Nhap a, b, c, d, e, f: ";

cin >> a >> b >> c >> d >> e >> f;

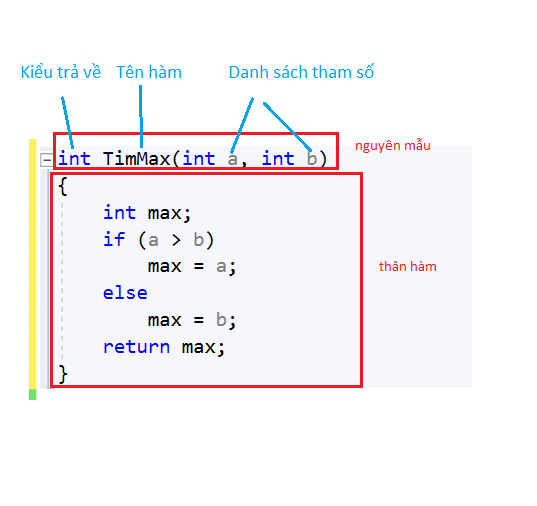
cout << TimMax(a, b) << endl;

cout << TimMax(c, d) << endl;

cout << TimMax(e, f) << endl;

}

1. **Khai báo hàm**



* Các thành phần:
* Nguyên mẫu: khai báo tên hàm, danh sách tham số, kiểu trả về của hàm. Các tham số truyền vào được đặt trong cặp dấu ngoặc tròn, cách nhau bởi dấu phẩy (, ). Cú pháp như sau:

<Kiểu trả về> <Tên hàm> (<Danh sách tham số>)

* Thân hàm: là khối lệnh ngay sau nguyên mẫu hàm, chứa các lệnh để thực hiện các công việc của hàm.

+ Kiểu trả về của hàm thuộc kiểu gì thì trong thân hàm phải có câu lệnh **return** để trả về giá trị thuộc kiểu đó.

+ Nếu hàm không trả về giá trị thì kiểu trả về là **void**. Khi đó thân hàm có thể không có câu lệnh return, nếu có thì chỉ chứa câu lệnh return; và không trả về giá trị gì.

Ví dụ:

* Hàm xuất ra dòng chữ “hello”.

void Xuat()

{

cout << "hello";

}

Ở trên, ta khai báo hàm tên là Xuat, không trả về giá trị (void), không chứa tham số truyền vào (không có tham số nào trong cặp dấu ngoặc tròn). Trong phần thân hàm, khai báo câu lệnh xuất ra màn hình dòng chữ “hello”.

* Hàm tính và trả về giá trị tổng hai số nguyên a, b:

int TinhTong(int a, int b)

{

int ketqua = a + b;

return ketqua;

}

Ở ví dụ này, tên hàm là TinhTong, tham số truyền vào là hai giá trị số nguyên được đặt tên tương ứng là a và b. Hàm trả về kết quả số nguyên tương ứng với giá trị tổng của a và b. Thân hàm thực hiện tính tổng a, b và sử dụng lệnh return để trả về kết quả.

* Lưu ý:
* Lệnh return dùng để trả về kết quả và thoát khỏi hàm ngay sau đó. Ví dụ:

#include<iostream>

using namespace std;

int TimMax(int a, int b)

{

if (a > b)

return a;

else

return b;

}

int main()

{

int a, b;

cout << "Nhap a, b: ";

cin >> a >> b;

cout << TimMax(a, b) << endl;

}

Hàm TimMax trong đoạn chương trình trên có thể viết gọn hơn thành:

int TimMax(int a, int b)

{

if (a > b)

return a;

return b;

}

Do nếu a > b thỏa mãn thì sẽ trả về a và kết thúc hàm, tất cả các câu lệnh sau đó sẽ không được thực thi nữa. Nếu điều kiện a > b không thỏa mãn thì mới đi đến được lệnh return b.

1. **Thực thi một hàm**

* Để gọi một hàm đã khai báo, cú pháp như sau:

Tên\_hàm([các tham số])

Ví dụ: khai báo hàm tính tổng 2 số kiểu nguyên. Trong hàm main cho phép nhập 2 số nguyên a, b. Gọi hàm đã khai báo để xuất ra tổng của a và b.

#include<iostream>

using namespace std;

int TinhTong(int a, int b)

{

return a + b;

}

int main()

{

int a, b;

cout << "Nhap a, b: ";

cin >> a >> b;

cout << TinhTong(a, b) << endl;

}

Lưu ý:

* Khi gọi một hàm, phải truyền đủ tham số cho hàm và kiểu của tham số truyền vào phải tương ứng với kiểu của đối số (tham số) trong nguyên mẫu của hàm. Ví dụ, đoạn chương trình sau mắc 2 lỗi:

#include<iostream>

using namespace std;

int TinhTong(int a, int b)

{

return a + b;

}

int main()

{

cout << TinhTong(5); // Truyen thieu tham so

cout << TinhTong(1, "2"); // sai kieu du lieu tham so

}

* Chương trình C++ sẽ được thực hiện từ trên xuống, nếu gọi một hàm chưa được khai báo sẽ bị lỗi. Ví dụ:

#include<iostream>

using namespace std;

int main()

{

Xuat();

}

void Xuat()

{

cout << "Hello";

}

Đoạn chương trình trên bị lỗi do hàm Xuat được khai báo sau hàm main, chương trình sẽ chạy hàm main trước tiên nhưng hàm Xuat chưa được khai báo trước đó, trình biên dịch chưa thấy hàm này nên sẽ trả về lỗi. Để khắc phục vấn đề này, ta khai báo hàm Xuat trước hàm main hoặc khai báo trước phần nguyên mẫu ở phía trên hàm main và kết thúc nguyên mẫu bằng dấu chấm phẩy để thông báo cho trình biên dịch biết là ta sẽ có hàm Xuat:

#include<iostream>

using namespace std;

void Xuat();

int main()

{

Xuat();

}

void Xuat()

{

cout << "Hello";

}

1. Tham trị và tham chiếu

* Xét ví dụ sau:

#include<iostream>

using namespace std;

void ThamTri(int n)

{

n = 10;

}

void ThamChieu(int &n)

{

n = 5;

}

int main()

{

int n = 1;

ThamTri(n);

cout << "n = " << n << endl;

ThamChieu(n);

cout << "n = " << n << endl;

}

Kết quả chạy chương trình:



* Ta thấy sau khi thực hiện hàm ThamTri, giá trị của biến n không đổi, còn sau khi thực hiện hàm ThamChieu, giá trị biến n bị thay đổi. Đây là sự khác biệt giữa tham trị và tham chiếu.
* Để khai báo 1 tham số được truyền vào theo kiểu tham chiếu, ta sử đặt dấu (&) trước tên biến như trong ví dụ, nếu không có dấu này thì đó là tham trị.
* Ví dụ hàm hoán đổi giá trị 2 số nguyên a, b cho nhau:

#include<iostream>

using namespace std;

void HoanDoi(int& a, int& b)

{

int c = a;

a = b;

b = c;

}

int main()

{

int a = 1, b = 5;

HoanDoi(a, b);

cout << "a = " << a << ", b = " << b;

}

Kết quả chạy chương trình:



Hãy thử bỏ các dấu và (&) trước mỗi tham số a, b để thấy sự khác biệt.

1. Biến toàn cục và biến cục bộ

|  |  |
| --- | --- |
| Biến toàn cục (global) | Biến cục bộ (local) |
| * Là biến mà lệnh khai báo của nó không nằm trong hàm hay khối lệnh nào. | * Được khai báo bên trong một khối lệnh hoặc hàm. |
| * Có thể được sử dụng trong tất cả các câu lệnh viết bên dưới nó. | * Chỉ có tác dụng bên trong hàm / khối lệnh chứa nó. |
| * Giá trị mặc định được khởi tạo là 0. | * Nếu không khởi tạo thì giá trị của biến cục bộ là không xác định. |
|  |  |

Ví dụ:

#include<iostream>

using namespace std;

int x; // bien toan cuc

int Tong(int n)

{

// n co pham vi trong ham Tong

int S = 0;

// S la bien cuc bo cua ham Tong

// i la bien cuc bo trong vong lap for

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

S += i;

return S;

}

int main()

{

cout << "x = " << x << endl;

int n; // n la bien cuc bo trong ham main

cout << "Nhap n: ";

cin >> n;

cout << Tong(n);

}

* Lưu ý: tên biến toàn cục và cục bộ trong cùng 1 khối lệnh không được trùng nhau. Ở ví dụ trên, có thể khai báo 2 biến n do chúng đều là biến cục bộ và được khai báo ở 2 hàm khác nhau.
* Tham khảo thêm các ví dụ trong tài liệu chương 5.