Chương 3

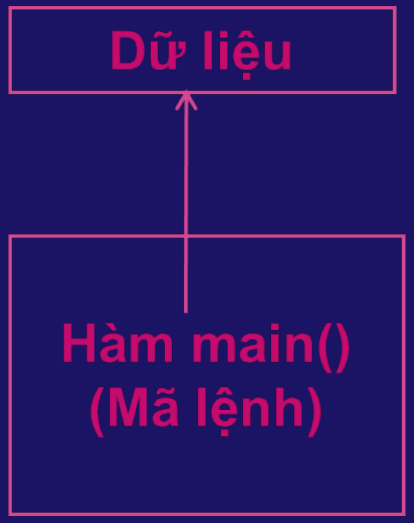
Kỹ thuật lập trình module

Giảng viên: Nguyễn Chiến Thắng
Email: thangnc.haui@gmail.com

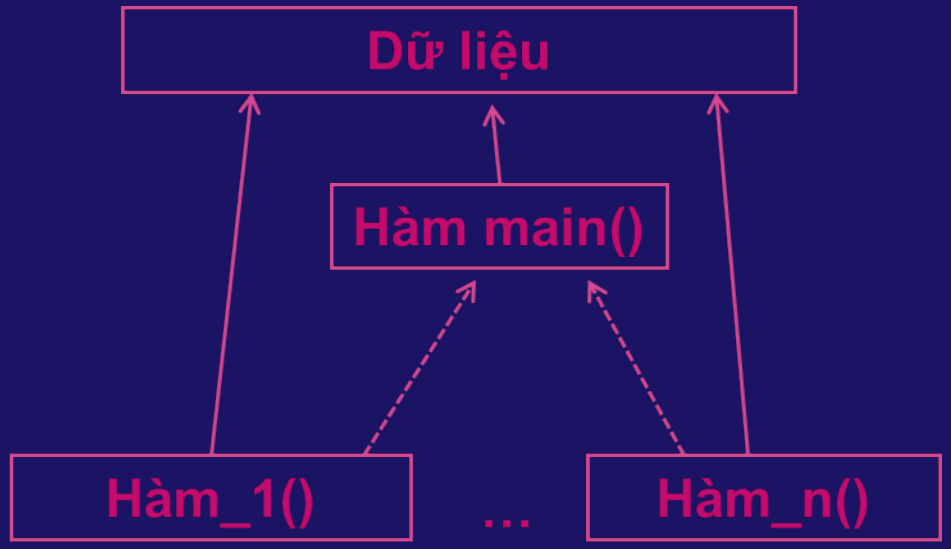
Nội dung

1. Khái niệm module - hàm
2. Cấu trúc của hàm
3. Cách khai báo hàm trong chương trình
4. Phân loại hàm
5. Xây dựng và sử dụng hàm
6. Truyền tham số cho hàm
7. Phạm vi của các loại biến

3.1. Khái niệm hàm



Chương trình không dùng hàm



Chương trình chia thành các hàm

Khái niệm hàm (tt)

1. Hàm là một đoạn chương trình thực hiện một tác vụ được định nghĩa cụ thể.
2. Các hàm được sử dụng để rút gọn cho một chuỗi các chỉ thị được thực hiện nhiều lần.
3. Việc gỡ lỗi chương trình trở nên dễ dàng hơn khi cấu trúc của chương trình rõ ràng với hình thức lập trình theo module.
4. Chương trình cấu tạo từ các hàm cũng dễ dàng bảo trì, bởi vì sự sửa đổi khi có yêu cầu được giới hạn trong từng hàm của chương trình.

3.2. Cấu trúc của hàm

- Cấu trúc của một hàm trong C++

```
[kiểu_trả_về] <tên_hàm> (các đối của hàm)
{
    //Thân hàm
}
```

- [kiểu_trả_về]: xác định kiểu dữ liệu của giá trị mà hàm sẽ trả về
- <tên_hàm>: là một định danh
- (các đối của hàm) xuất hiện trong cặp dấu ngoặc ( ) biểu diễn cho dữ liệu mà hàm xử lý

3.3. Vị trí định nghĩa hàm trong chương trình

- Khai báo nguyên mẫu các hàm trước hàm main().
- Định nghĩa hàm sau hàm main().

```
int f1(int a, int b); //Nguyên mẫu của hàm f1
void f2(float a, char C);

int main() {
      // Thân hàm main
}

//Định nghĩa hàm f1, f2
int  f1(int a, int b) {
      //Thân hàm f1
}
void f2(float a, char C) {
      //Thân hàm f2
}
```

Vị trí định nghĩa hàm trong chương trình (tt)

- Khai báo và định nghĩa các hàm trước hàm main().

```
int  f1(int a, int b) {
      //Thân hàm f1
}
void f2(float a, char C) {
      //Thân hàm f2
}

int main() {
      // Thân hàm main
}
```

3.3. Phân loại hàm

- Theo giá trị trả về
  - Hàm định kiểu: Hàm có giá trị trả về.

```
int binh_phuong(int a) {
        int t;
        t = a * a;
        return t;
}
```

  - Hàm không định kiểu: Hàm không có giá trị trả về.

```
void hien_thi(char ten[]) {
    cout<<"Ten ban la "<<ten<<endndl;
}
```

3.3. Phân loại hàm

- Theo giá đối số
  - Hàm có đối:

```
int tong(int a, int b) {
        int t;
        t = a + b;
        return t;
}
```

  - Hàm không có đối:

```
void thong_bao() {
    cout<<"Xin chao cac ban"<<endl;
}
```

3.5. Xây dựng hàm
3.5.1. Các bước xây dựng hàm

1. Phân tích yêu cầu mà hàm phải thể hiện
   - Xác định dữ liệu vào, dữ liệu ra
   - Xây dựng thuật toán
2. Xác định kiểu trả về của hàm (DL ra)
3. Đặt tên hàm (theo yêu cầu)
4. Xác định các tham số hình thức (DL vào, ra)
5. Xây dựng thân hàm (thuật toán)

Xây dựng hàm – Ví dụ

1. Xây dựng hàm tìm số lớn nhất trong 5 số nguyên.

```
int max(int a,int b,int c,int d,int e){
    int m = a;
    if (m < b) m = b;
    if (m < c) m = c;
    if (m < d) m = d;
    if (m < e) m = e;
    return m;
}
```

Xây dựng hàm – Ví dụ

2. Xây dựng hàm giải phương trình bậc hai.

```
void giaiPtb2(float a,float b,float c){
     if (a == 0){
        cout<<"Khong phai PTB2";
        return;
     }
     float delta = b*b-4*a*c;
     if (delta < 0) cout<<"PT vo nghiem";
     else if (delta == 0){
        float x = -b/2/a;
        cout<<"PT co nghiem kep x1 = x2 = "<<x;
     } else{
        float x1 = (-b+sqrt(delta))/2/a;
        float x2 = (-b-sqrt(delta))/2/a;
        cout<<"PT co 2 nghiem phan biet";
        cout<<"\n\tx1 = "<<x1<<"\n\tx2 = "<<x2;
     }
}
```

3.6. Sử dụng hàm

- Gọi hàm
- Sự trở về từ một hàm
- Phạm vi của biến
- Truyền tham số

3.6.1. Gọi hàm

- Cú pháp:

```
<Tên_hàm> <([DS tham số])>
```

- Chú ý
  - Cặp dấu ngoặc () là bắt buộc theo sau tên hàm.
  - Số lượng tham số phải bằng số lượng đối.
  - Kiểu của các tham số phải tương ứng với các đối.
  - Nếu hàm có giá trị trả về thì lời gọi hàm được sử dụng như một biểu thức.
  - Nếu hàm không có giá trị trả về thì lời gọi hàm là một lệnh.
- Ví dụ:

```
M = max(2, 5, 7, 4, -5);
giaiPTB2(4, -5, 1);
```

3.6.2. Sự trở về từ một hàm

```
int x, m;
int binhPhuong(int a) {
     int t = a * a;
     return t;
}
int main() {
     cout<<"Nhap x: "; cin>>x;
     m = binhPhuong(x);
     cout<<"m = "<<m;
}
```

1. Lệnh `return t;` ngay lập tức chuyển điều khiển từ hàm `binhPhuong(…)` trở về hàm `main()`.
2. Giá trị của `t` theo sau `return` được trả về cho hàm `main()`.

3.6.3. Phạm vi của biến

- Biến cục bộ
  - Được khai báo bên trong một hàm
  - Được tạo ra tại điểm vào của một hàm và bị hủy tại điểm ra khỏi hàm đó.
- Các đối của hàm
  - Được khai báo trong định nghĩa hàm như là các biến
  - Hoạt động như một biến cục bộ bên trong một hàm
- Biến toàn cục
  - Được khai báo bên ngoài tất cả các hàm
  - Lưu các giá trị tồn tại suốt thời gian thực thi của chương trình

3.6.3. Phạm vi của biến (tt)

```
int tong;
void nhap(int &a, int &b, int &c) {
    cout << "Nhap 3 so nguyen:";
    cin >> a >> b >> c;
}
void tinh_tong(int a, int b, int c) {
    tong = a + b + c;
}
float tinh_tbc(int a, int b, int c) {
    float t = (a + b + c) / 3.0;
    return t;
}
int main() {
    int a, b, c;
    nhap(a, b, c);
    tinh_tong(a, b, c);
    cout << "Tong = " << tong << endl;
    cout << "TBC = " << tinh_tbc(a, b, c) << endl;
    return 0;
}
```

3.6.4. Truyền tham số cho hàm

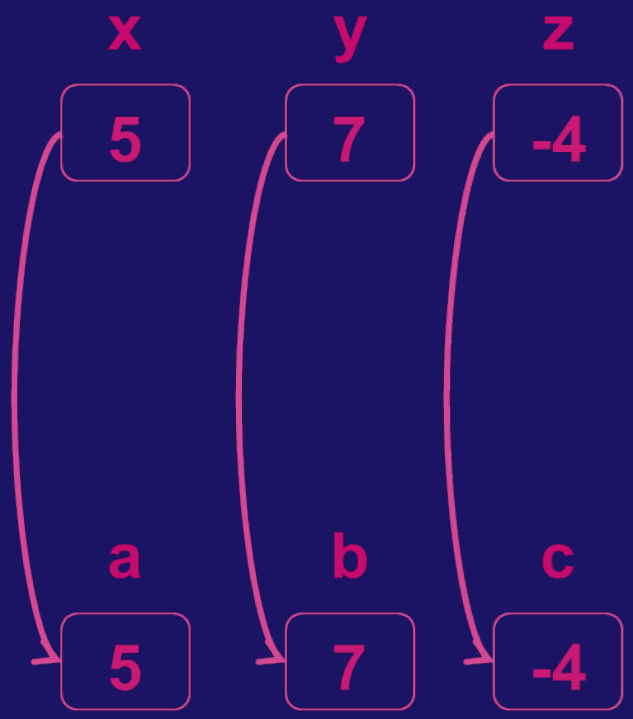
1. Truyền tham trị
2. Truyền tham chiếu

Truyền tham trị

1. Các tham số được truyền đến hàm thông qua các đối, giá trị của tham số được gán cho các đối tương ứng.
2. Các thao tác bên trong hàm chỉ được thực hiện trên các đối.
3. Giá trị chứa trong các tham số (các biến) được truyền đến hàm không bị thay đổi.

Truyền tham trị (tt)

```
int min(int a, int b, int c)
{
	int m = a;
	if (m > b) m = b;
	if (m > c) m = c;
	return m;
}
int main()
{
	int x = 5, y = 7, z = -4;
	int m = min(x, y, z);
	cout<<"min = "<<m;
}
```

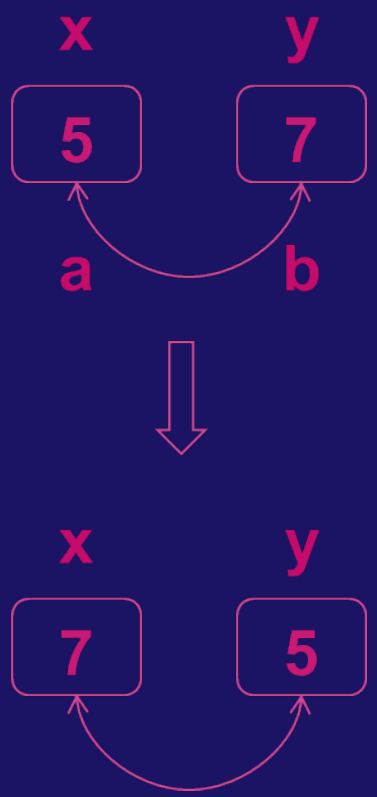


Truyền tham chiếu

- Các đối của hàm tham chiếu đến địa chỉ của các tham số (biến) truyền vào hàm.
- Các thao tác bên trong hàm được thực hiện trực tiếp trên các biến được truyền đến hàm.
- Giá trị của các biến được truyền đến hàm có thể bị thay đổi (thường sẽ thay đổi).

Truyền tham chiếu (tt)

```
void daoGiaTri(int &a, int &b)
{
	int tg = a;
	a = b;
	b = tg;
}
int main()
{
	int x = 5, y = 7;
	daoGiaTri(x, y);
	cout<<x<<" va "<<y;
}
```



Hàm đệ quy

- Hàm mà trong thân hàm có chứa lời gọi đến chính nó gọi là hàm đệ quy.
- Hàm đệ quy được sử dụng để cài đặt các thuật toán được thiết kế dạng đệ quy.
- Đệ quy, thực chất là một kỹ thuật thiết kế thuật toán thay cho kỹ thuật lặp.
- Thuật toán đệ quy rất hữu dụng khi việc thiết kế kế giải thuật lặp thực sự gặp khó khăn.

Hàm đệ quy (tt)

1. Bài toán n!
2. Bài toán số fibonaci

Hàm đệ quy (tt)

1. Bài toán tính n!

- Một số **n!** được định nghĩa như sau:

n! = n*(n-1)*…*3*2*1

- Trường hợp đặc biệt ta quy ước:

0! =1

- Công thức truy hồi:

n! = n*(n-1)!

Hàm đệ quy (tt)

1. Bài toán tính n!

```
int giaiThua(int n)
{
    if (n == 0)
        return 1;
    return n * giaiThua(n - 1);
}

int main()
{
    int n;
    cin >> n;
    cout << n << "! =" << giaiThua(n);
    return 0;
}
```

Hàm đệ quy (tt)

1. Bài toán số fibonaci

- Dãy Fibonacci là dãy số mà số tiếp theo là tổng của hai số liền trước

  Ví dụ: 1, 1, 2, 3, 5, 8, 13

- Trường hợp đặc biệt ta quy ước:

  f(1) = 1; f(2) =1

- Công thức truy hồi:

  f(n) = f(n-1) + f(n-2)

Hàm đệ quy (tt)

1. Bài toán số fibonaci

```
int Fibonacci(int n)
{
    if (n == 1 || n == 2)
        return 1;
    return Fibonacci(n - 1) + Fibonacci(n - 2);
}

int main()
{
    int n;
    cout << "Nhap n: ";
    cin >> n;
    cout << "Fibonaci(" << n << ") = " << Fibonacci(n);
    return 0;
}
```

Thank you!