Chương 3

# Kỹ thuật lập trình module

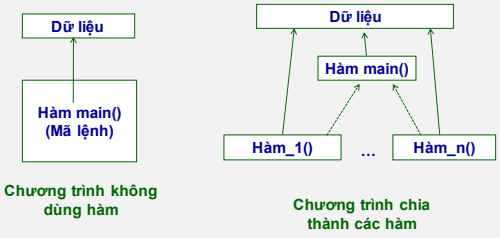
Design by Minh An
Email: anvanminh.haui@gmail.com

## 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)

- Hàm là một đoạn chương trình thực hiện một tác vụ được định nghĩa cụ thể.
- Các hàm được sử dụng để rút gọn cho một chuỗi các chỉ thị được thực hiện nhiều lần.
- 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.
- 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 như sau:

```
[kiểu_trả_về] <tên_hàm>(tham số hình thức)
{
      //Thân hàm
}
```

Nguyên mẫu hàm

- `[kiểu_trả_về]`: xác định kiểu dữ liệu của giá trị mà hàm sẽ trả về.
- `<tên_hàm>`: Là một định danh.
- `(các tham số hình thức)` xuất hiện trong cặp dấu ngoặc () biểu diễn cho dữ liệu vào/ra của hàm.



## 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[]) {
    printf("Ten ban la %s", ten);
}
```



## Phân loại hàm

- Theo đối tượng tạo ra hàm:
  - ✓ Hàm chuẩn: Hàm có sẵn trong thư viện
    getch()-conio.h, sqrt()-math.h, gets()-stdio.h v.v…
  - ✓ Hàm tự định nghĩa: Hàm do người lập trình tự viết
- Ví dụ:

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



## 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.
2. Xây dựng hàm giải phương trình bậc hai.



## 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 thực sự])>
```

- Chú ý
  - ✓ Cặp dấu ngoặc () là bắt buộc theo sau tên hàm.
  - ✓ Số lượng tham số thực sự phải bằng số lượng tham số hình thức.
  - ✓ Kiểu của các tham số phải tương ứng.
  - ✓ 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 như một lệnh.
- Ví dụ:

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



## 2.6.2. Sự trở về từ một hàm

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

- Lệnh return ngay lập tức chuyển điều khiển từ hàm trở về chương trình gọi.
- Giá trị theo sau lệnh return được trả về cho chương trình gọi.



## 2.6.3. 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.
- Tham số hình thức
  - ✓ 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



## 2.6.3. Truyền tham số cho hàm

1. Truyền tham trị
2. Truyền tham chiếu



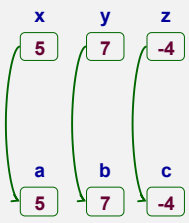
## 2.6.3.1. Truyền tham trị

- Khi các tham số thực sự được truyền đến hàm, các giá trị được truyền thông qua các biến tạm (các tham số hình thức).
- Mọi sự thao tác chỉ được thực hiện trên các biến tạm.
- Tham số thực sự có thể là các hằng giá trị, hoặc các giá trị chứa trong các biến.
- Giá trị chứa trong 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;
}
```



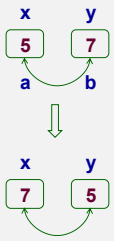
## 2.6.3.2. Truyền tham chiếu

- Các tham số thực sự được truyền đến hàm phải là địa chỉ của các biến.
- Địa chỉ của các biến (chứ không phải giá trị) được truyền đến hàm.
- Mọi sự thao tác 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.

Design by Minh An

## 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;
}
```




Design by Minh An

# Bài tập

Design by Minh An