



Hàm - Function

Hàm - Function

- Một số nguyên tắc
- Cách khai báo và gọi thực hiện
- Prototype của hàm
- Truyền tham số cho hàm
- Biến toàn cục, biến cục bộ, biến static, biến thanh ghi, ...
- Cách thức C thực hiện các lời gọi hàm – stack.

Một số nguyên tắc

- Các hàm trong NNLT C đều ngang cấp với nhau:
 - Hàm không được khai báo lồng nhau.
 - Thứ tự khai báo không quan trọng.
- Hàm có thể nhận và xử lý nhiều tham số hoặc không có tham số nào
- Hàm có thể trả về một giá trị hoặc không.
- Biến khai báo trong hàm F chỉ có giá trị trong F, không sử dụng được biến này trong các hàm khác được.

Ví dụ: hàm tính x^n

kiểu của giá trị trả về

nhận vào 2 tham số khi được gọi

```
double    Power(double x, int n)
{
    double result = 1;

    for(int i=1; i< n; i++)
        result *= x;

    return result;
}
```

giá trị được trả về qua lệnh return

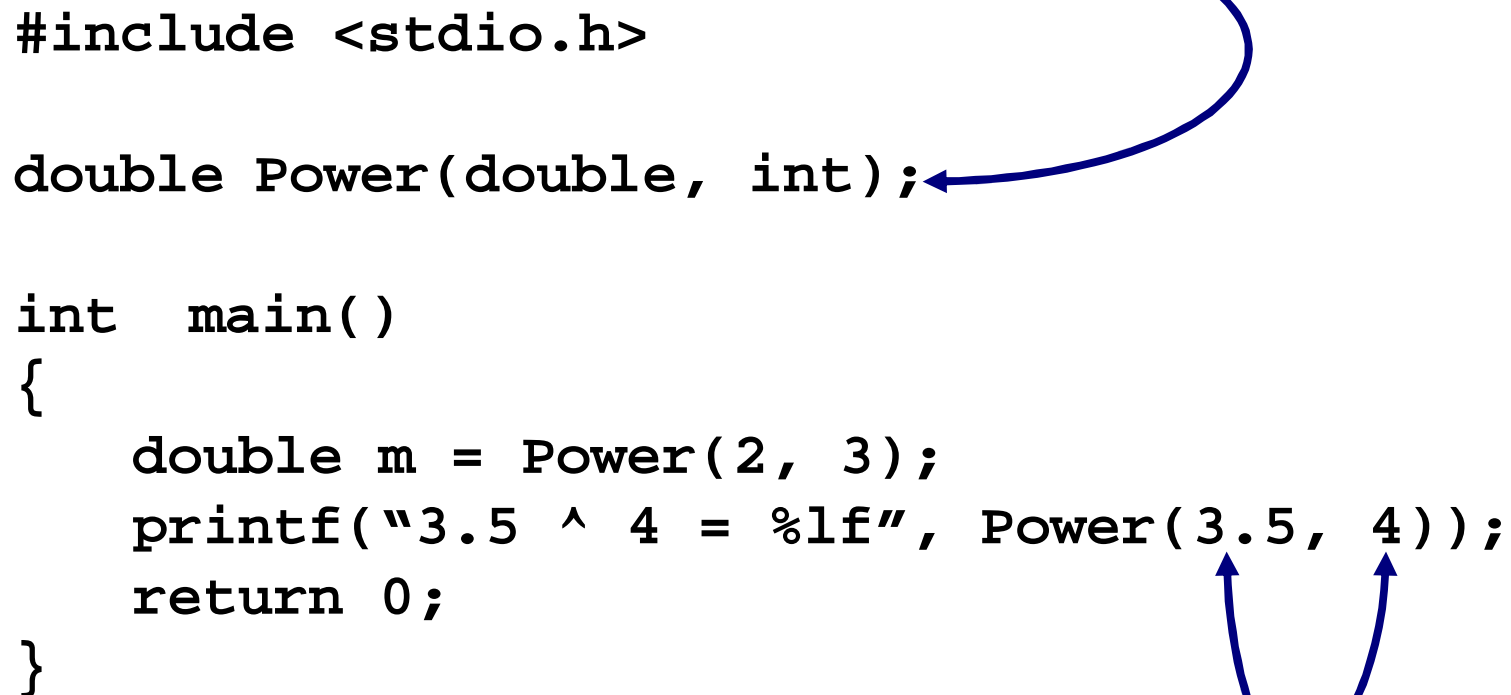
Ví dụ: gọi thực hiện hàm Power

Chỉ thị cho chương trình biết prototype của hàm Power

```
#include <stdio.h>

double Power(double, int);

int main()
{
    double m = Power(2, 3);
    printf("3.5 ^ 4 = %lf", Power(3.5, 4));
    return 0;
}
```



3.5 và 4: 2 tham số thực sự

Một số lỗi thường gặp

Compiler không hiểu được hàm Power

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int m = Power(2, 3);
```

```
    printf("3.5 ^ 4 = %lf", Power(4));
```

```
    return 1.0;
```

```
}
```

hàm Power thiếu tham số

giá trị trả về không khớp kiểu

Prototypes

- Dòng khai báo

double Power(double, int);

được hiểu là khai báo *prototype* của hàm Power

- Được dùng khi chương trình sử dụng một hàm trước khi khai báo.
- Khai báo prototype thông báo cho trình biên dịch biết kiểu của giá trị trả về và mô tả chi tiết về các tham số của hàm.
- Các hàm thư viện chuẩn được khai báo prototype trong các tập tin header (stdio.h, conio.h, ...).
- Các hàm do lập trình viên tự xây dựng phải tự khai báo prototype.

Hàm: dạng tổng quát

header của hàm

kiểu trả về tên hàm(danh sách tham số hình thức)

{

//khai báo các biến của hàm

//các lệnh thực thi

return giá trị trả về; *//hàm void không có giá trị trả về*

}

thân (body) hàm

Tầm tác dụng của biến

- Biến toàn cục: Không thuộc khối nào, có tác dụng trong toàn chương trình kể từ khi khai báo
- Biến cục bộ: khai báo trong một khối, chỉ có tác dụng trong khối này

“f” của hàm F, không phải của main

```
float g=6.5;
void main()
{
    int i = 5, j, k = 2;
    float f = 2.8F;
    d = 3.7;
}
void F(int v)
{
    double d, e = 0.0, f;
    i++; g--;
    f = 0.0;
}
```

compiler không chấp nhận “d”, “i”

Truyền tham số cho hàm

C hỗ trợ 2 cách truyền tham số:

- ☐ Truyền tham số bởi giá trị (truyền giá trị - call by value)
- ☐ Truyền tham số bởi địa chỉ (truyền địa chỉ - call by address)

Mở rộng với C++

- ☐ Truyền tham chiếu (call by reference)

Truyền giá trị

- Hàm sẽ xử lý trên bản sao của tham số
 - Hàm không thể thay đổi giá trị của tham số được.
- Được dùng trong các trường hợp cần chuyển dữ liệu vào bên trong hàm để xử lý, tính toán
- Các ví dụ trên đều dùng kiểu truyền tham số bởi giá trị
- Ví dụ hàm có sẵn của C truyền giá trị:
 - `float sqrt(float);`
 - `double pow(double, double);`

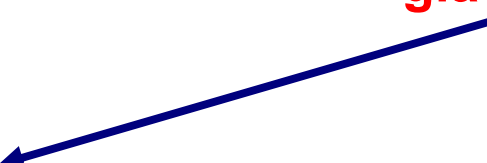
Truyền giá trị - ví dụ

```
#include <stdio.h>
void change(int v);

int main()
{
    int var = 5;
    change(var);
    printf("main: var = %i\n", var);
    return 0;
}

void change(int v)
{
    v *= 100;
    printf("change: v = %i\n", v);
}
```

**hàm change
không thay đổi
giá trị của "var"**



change: v = 500
main: var = 5

Truyền địa chỉ

- Hàm sẽ xử lý trên chính tham số nhờ vào địa chỉ của chúng
- Hàm có thể thay đổi giá trị của tham số.
- Được dùng trong các trường hợp cần chuyển dữ liệu là kết quả xử lý được bên trong hàm ra “*ngoài*” cho các hàm khác sử dụng.

Ví dụ hàm có sẵn của C truyền địa chỉ:

```
int scanf(const char *format, adr1, adr2, ...);
```



Truyền địa chỉ - ví dụ

```
#include <stdio.h>
void change(int *v);

int main()
{
    int var = 5;
    change(&var);
    printf("main: var = %i\n", var);
    return 0;
}

void change(int *v)
{
    (*v) *= 100;
    printf("change: *v = %i\n", (*v));
}
```

v: tham số địa chỉ
của số int, khai
báo với dấu *

truyền địa chỉ của "var"
vào hàm change

change: *v = 500
main: var = 500

Phương thức trao đổi dữ liệu

- C dùng 1 stack để lưu trữ các biến cục bộ và các chuyển các tham số cho hàm với mỗi lần gọi hàm thực hiện
- ① Hàm gọi (O) cất các tham số vào stack.
- ② Gọi thực hiện hàm được gọi (F).
- ③ F nhận lấy các tham số từ stack
- ④ F tạo các biến cục bộ ứng với các tham số trên stack
- ⑤ Khi kết thúc, F cập nhật giá trị các tham số (ref) và trả điều khiển cho O
- ⑥ O nhận lấy các giá trị mới của tham số cũng như giá trị trả về

Phương thức trao đổi dữ liệu

```
#include <stdio.h>
double power(int, int);
int main(void)
{
    int    x = 2;
    double d;
    d = power(x, 5);
    printf("%lf\n", d);
    return 0;
}
double power(int n, int p)
{
    double result = n;
    while(--p > 0)
        result *= n;
    return result;
}
```

32.0	power: result
2	power: n
5	power: p
<hr/>	
32.0	main: d
2	main: x

Tổng kết

- Khai báo và gọi thực hiện hàm
- Khai báo prototypes
- Tầm tác dụng của biến
- Truyền tham số cho hàm
- Vấn đề tổ chức dữ liệu trong chương trình