HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

HÀM VÀ LẬP TRÌNH CÓ CẤU TRÚC

SOICT-HUST

Hàm

- Là một nhóm các khai báo và câu lệnh mà được đặt tên
- Một hàm tương đương với một chương trình con
- Chương trình chúng ta viết là một hàm được đặt tên là main và có thể gọi đến các chương trình con
- Các chương trình con này lại có thể sử các hàm khác, và cứ tiếp tục như vậy
- Tuy nhiên về mặt cú pháp, tất cả các hàm trong C đều có cùng một cấp. Do vậy một hàm có thể gọi tới một hàm khác bất kì có trong chương trình

Ví dụ

Định nghĩa hàm

Gọi hàm

```
#include <stdio.h>

/* Ham in loi chao mung */
void sayHello ( void )
{
  printf("Hello World!\n");
}

/*
 * Goi ham in loi chao mung
 */
int main(void)
{
  sayHello();
  return 0;
}
```

Tại sao sử dụng hàm?

- Chúng cho phép chia nhỏ vấn đề thành các công việc con
  - Giúp giải quyết dễ dàng hơn những vấn đề phức tạp
- Sử dụng hàm chương trình được viết sẽ sáng sủa hơn nhờ khả năng "trừu tượng hoá"
  - Chúng ta chỉ cần biết một hàm làm gì mà không quan tâm nó làm thế nào
- Chúng cho phép tổng quát hoá một số nhóm lệnh lặp nhiều lần
  - Tránh viết đi viết lại nhiều lần một số nhóm lệnh

Xây dựng hàm

- Với mỗi hàm xây dựng cần phải đặc tả:
  - Tên hàm
  - Tham số truyền vào
  - Loại giá trị mà hàm trả về nếu có
  - Khối lệnh được thực hiện khi hàm được gọi đến
- Khối lệnh thực hiện còn được gọi là thân hàm

Ví dụ hàm giai thừa

Tên hàm

Thân hàm

```
#include <stdio.h>

int giaithua (int a) {
    int i, gt=1;
    for(i=1; i<=a; i++)
       gt = gt * i;
    return gt;
}

int main( void ) {
    int num;

    printf("Nhap so nguyen:");
    scanf("%d",&num);

    printf("%d!=%d\n",
           num, giaithua(num));
}
```

Tham số hàm

- Là các biến địa phương của hàm mà giá trị được xác định cho mỗi lần gọi hàm
  - Vì vậy tham số có giá trị khác nhau cho mỗi lần gọi
  - Tham số chỉ có thể truy nhập bên trong hàm mà thôi
  - Khi gọi hàm giá trị cho tất cả các tham số phải được xác định
- Chú ý phân biệt
  - Biến địa phương bên trong hàm dùng chứa tham số và biến mà giá trị của nó được dùng để gọi hàm
  - Vì thế nếu hàm thay đổi giá trị cho tham số, nó sẽ không làm thay đổi giá trị của biến được sử dụng để gọi hàm
- Một hàm không có tham số có thể được khai báo với từ khoá void cho danh sách tham số

Ví dụ tham số

```
#include <stdio.h>

int addOne ( int i )
{
    i = i + 1;
    return i;
}

int main(void)
{
  int i = 3;

  printf("%d\n", addOne(i) );
  printf("%d\n", i);

 return 0;
}
```

Khai báo tham số là biến địa phương

Thay đổi giá trị biến địa phương

Truyền giá trị của biến **i** trong hàm **main** cho lời gọi hàm

Output:

4

3

Ví dụ (tiếp)

```
void badSwap ( int a, int b )
{ int temp;
  temp = a;
  a = b;
  b = temp;
  printf("Called environment: %d %d\n",a,b);
}

int main(void)
{ int a = 3, b = 5;
  printf("Calling environment: %d %d\n",a,b);
  badSwap ( a, b );
  printf("Calling environment: %d %d\n",a,b);
  return 0;
}
```

Trả về giá trị

- Lệnh return dùng để trả về giá trị cho hàm
- Lệnh này phải chứa giá trị mà có thể chuyển về kiểu giá trị trả về được đặc tả cho hàm
- Một hàm có thể có nhiều lệnh return, nhưng lệnh nào được gặp đầu tiên sẽ làm kết thúc hàm tại điểm đó với giá trị trả về tương ứng
- Một hàm không trả về giá trị nào phải được khai báo với kiểu trả về void
  - Khi đó có thể không cần sử dụng lệnh return trong hàm

Khai báo và định nghĩa hàm

- Một định nghĩa hàm đặc tả đầy đủ tất cả các thành phần của hàm bao gồm cả thân hàm
- Một khai báo hàm chỉ cần đặc tả:
  - Tên hàm
  - Kiểu của từng tham số
  - Kiểu của giá trị trả về
- Tạo khai báo hàm bằng các nguyên mẫu (prototype). Ví dụ:

```
int addOne (int);
void sayHello(void);
```

Vai trò của nguyên mẫu

- Một hàm có thể định nghĩa sau khi sử dụng nhưng phải được khai báo trước khi dùng
- Nó cho phép bạn gọi một hàm mà không cần biết đến định nghĩa
  - Ví dụ khi sử dụng printf(), chúng ta không biết hàm làm việc như thế nào nhưng chương trình dịch biết đây là một hàm với kiểu tham số của nó
  - Tệp stdio.h chứa khai báo (nguyên mẫu) của printf()
- Một định nghĩa hàm bao hàm luôn một khai báo

Nguyên mẫu hàm giai thừa

Nguyên mẫu

Định nghĩa hàm

```
#include <stdio.h>

int giaithua (int);

int main( void ) {
    int num;

    printf("Nhap so nguyen:");
    scanf("%d",&num);

    printf("%d!=%d\n",
            num, giaithua(num));
}

int giaithua (int a) {
    int i, gt=1;
    for(i=1; i<=a; i++)
       gt = gt * i;
    return gt;
}
```

Biến tổng thể

- Biến được khai báo trong hàm là một biến cục bộ. Biến này chỉ được truy cập trong hàm của nó mà thôi
- Biến tổng thể là biến được khai báo bên ngoài phạm vi hàm và nó có thể được truy cập bởi nhiều hàm khác nhau.
- Ví dụ:

```
int global;
void f(void) {  global = 0; }
void f(void) {  global = 1; }
```

Biến trùng tên

- Khi có sự trùng tên giữa biến tổng thể và biến cục bộ trong một hàm, thì biến cục bộ được ưu tiên dùng trong hàm đó.
- Ví dụ:

```
int i;
void f() {
  int i;
  i++; // chỉ làm thay đổi giá trị biến i cục bộ
}
void g() {
  i++; // làm thay đổi giá trị của biến i tổng thể
}
```

Các hàm thư viện

- Ngôn ngữ C cung cấp một số hàm thư viện như vào ra, toán học, quản lí bộ nhớ, xử lí xâu chuỗi, v.v.
- Để có thể sử dụng, ta cần khai báo nguyên mẫu của chúng trong chương trình
- Các khai báo nguyên mẫu như thế này đã được viết trong các tệp tiêu đề (.h), ta chỉ cần #include chúng vào chương trình

Thư viện toán học math.h

- Bao gồm một tập các hàm toán học với khai báo nguyên mẫu như sau:

```
double sin(double x);
double cos(double x);
double tan(double x);
…
double log(double x);
double sqrt(double x);
double pow(double x, double y);
int ceil(double x);
int floor(double x);
…
```

Bài tập

- Cho hai hàm nhập giá trị một số nguyên và tìm giá trị lớn nhất của hai số có khai báo nguyên mẫu như sau

```
int nhapso();
int max(int a, int b);
```

- Viết định nghĩa hàm và chương trình chính sử dụng các hàm đã cho để tìm giá trị lớn nhất của 3 số bất kì được nhập vào

HUST

THANK YOU !