KỸ THUẬT LẬP TRÌNH

CHƯƠNG 4. HÀM (FUNCTION)

Nội dung

1. Khái niệm hàm (Function)
2. Định nghĩa hàm
3. Thực thi hàm
4. Truyền tham số
5. Kết quả trả về
6. Prototype của hàm

Nội dung

7. Các hàm chuẩn
8. Thư viện hàm
9. Sự đệ quy
10. Bài tập

Mục tiêu bài học

- Trình bày được khái niệm hàm (function)
- Nêu được ý nghĩa của hàm con trong giải toán lập trình.
- Trình bày được cú pháp của hàm.
- Xác định được các thành phần của hàm trong bài toán thực tế
- Khai báo được tham số cho hàm.
- Khai báo prototype cho hàm.
- Phân loại được các dạng hàm.
- Nêu được một số hàm chuẩn trong NNLT C.
- Trình bày được khái niệm đệ qui và hàm đệ qui
- Vận dụng hàm con trong 1 số bài toán thông dụng

Dẫn nhập

- Bài toán

  Viết chương trình tính $C_n^k = \frac{n!}{k!(n-k)!}$ với n và k là các số nguyên dương k<=n;

- Các bước giải quyết bài toán
  - B1: Nhập k, n
  - B2: Tính n!
  - B3: Tính n! và (n-k)!
  - B4: Tính C_n^k

Dẫn nhập

```
int n, k, int kq;
//Nhập n và k
//Tính n giai thua
int n_giaithua=1;
for(int i=1;i<=n;i++)
    n_giaithua=n_giaithua*i;
//tính k giai thua
int k_giaithua=1;
for(int i=1;i<=k;i++)
    k_giaithua=k_giaithua*i;
```

```
//Tinh h=n-k giai thua
int h=n-k, h_giaithua=1;
for(int i=1;i<=h;i++)
    h_giaithua=h_giaithua*i;
//tính ket qua
int m;
m=(k_giaithua*h_giaithua);
kq=n_giaithua/
printf("ket qua
%5d \n", kq);
```

Dẫn nhập

- Nhận xét
  - **Mục đích**: Cả ba đoạn chương trình đều mục đích là tính giai thừa của một số nguyên: n, k, h=n-k
  - **Câu lệnh**: Cấu trúc câu lệnh giống nhau

⇒ Khó cải tiến, sửa chữa

Cách khắc phục: Sử dụng hàm

4.1 Khái niệm hàm

```
int TinhGT(int x)
{
        int gt=1;
        for(int i=1;i<=x;i++)
                gt=gt*i;
        return gt;
}
```

TinhGT(k)

TinhGT(n)

TinhGT(h)

```
int n, k,m,k, int kq;
//Nhập n và k
//tính ket qua
int m, h=n-k;
m=( k_giaithua *  h_giaithua );
kq=n_giaithua / m;
printf("ket qua can tinh la %5d
\n", kq);
```

4.1 Khái niệm hàm

- ***Khái niệm:*** Hàm là một phần của chương trình giải quyết một công việc nào đó.
- ***Ưu điểm***
  - Chương trình ngắn gọn, dễ hiểu, dễ quản lý
  - Dễ bảo trì, sửa chữa
  - Dễ kiểm tra lỗi
- ***Ưu điểm***
  - Hàm chuẩn
  - Hàm tự định nghĩa

4.2 Định nghĩa hàm

```
<Tên kiểu kết quả><Tên hàm> ([<kiểu t số><tham số>][…])
{
    [<Khai báo biến cục bộ và các câu lệnh thực hiện hàm>]
    [return <Biểu thức>;]
}
```

- **Tên kiểu trả về**: là kiểu dữ liệu của kết quả trả về
- **Tên hàm**: là định danh trong C và không trùng với tên biến và hàm khác
- **Tham số**: là các dữ liệu đầu vào của hàm
- **Kiểu t số**: là kiểu dữ liệu của tham số

4.2 Định nghĩa hàm

- Ví dụ:
  - Viết hàm tính tổng hai số nguyên.
  - Viết hàm tính chu vi hình tròn khi biết bán kính r.
  - Viết hàm tính tổng n số tự nhiên đầu tiên.
  - Viết hàm cho biết số tự nhiên n có phải là số nguyên tố không?
  - Viết hàm tìm ước chung lớn nhất của hai số nguyên dương
  - Viết hàm tính tổng các chữ số trong số tự nhiên n

4.2 Định nghĩa hàm

- Cách xác định hàm
  - **Hàm sẽ thực hiện công việc gì?**
  - **Tên hàm**: do người lập trình tự đặt, nên đặt tên hàm sao cho dễ nhớ và liên quan tới nội dung của hàm:

VD: KtraSNT, chuviHT, tong2songuyen,...

  - **Kiểu trả về**: là kiểu của kết quả trả về, nếu không có kết quả trả về kiểu trở về là void

VD: Tổng 2 số nguyên: int; Chu vi hình tròn: float

4.2 Định nghĩa hàm

- **Tham số** là các dữ liệu đầu vào của hàm

  Tính tổng 2 số nguyên: 2 số nguyên a, b

  Chu vi hình tròn: bán kính r

  Kiểm số nguyên tố n: số tự nhiên n

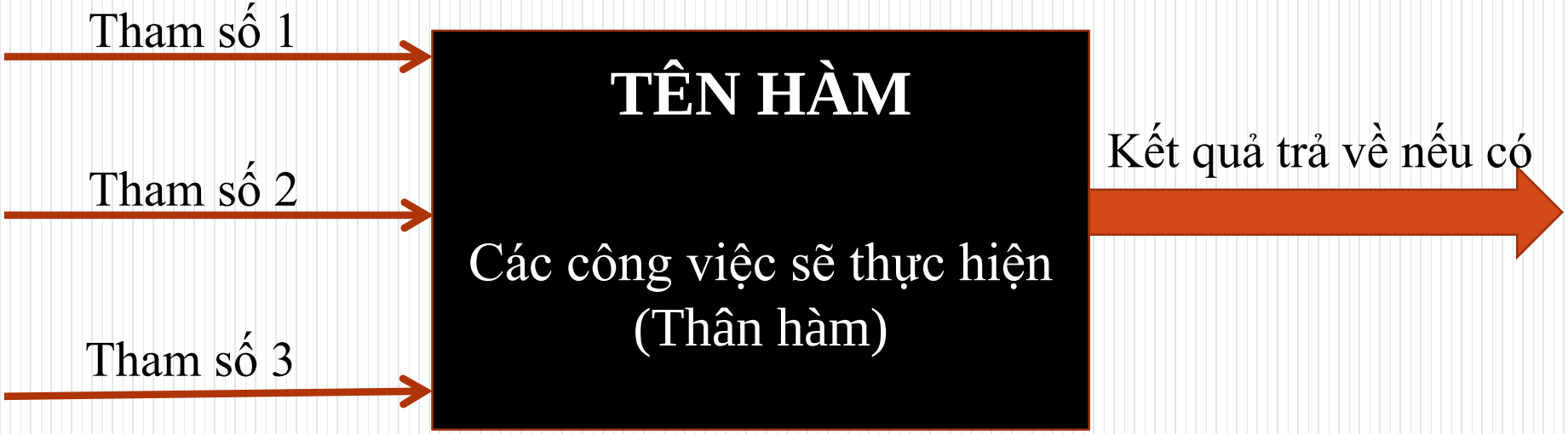
- **Kiểu tham số**: là kiểu dữ liệu của dữ liệu đầu vào

  Tổng 2 số nguyên: int a, int b

  Chu vi hình tròn: float r

- **Thân hàm**: là thuật toán, các câu lệnh để giải quyết công việc của hàm.

4.2 Định nghĩa hàm



4.2 Định nghĩa hàm

Ví dụ: Viết hàm tính tổng 2 số nguyên

- Tên
- Côn
- Tha
- Kiể
- Kiể
  số r
- Thâ

```
int Tong2songuyen(int a, int b)
{
        int tong=0;
        tong=a +b;
        return tong;
}
```

Tên hàm

DS tham số

Kiểu trả về

KQ trả về

4.2 Định nghĩa hàm

Ví dụ: Viết hàm tính chu vi hình tròn khi biết bán kính

- Tên hàm: chuviHT
- bán kính r
- t

```
float chuviHT (float r)
{
        return 2*r*3.14;
}
```

4.2 Định nghĩa hàm

Ví dụ: In ra màn hình n từ xin chao!

```
void Xuatloichao(int n)
{
        for(int i=0;i<n;i++)
                printf("xin chao!\n");
}
```

4.3 Thực thi hàm

- Thực thi làm là sử dụng hàm đã viết. Hàm chỉ được thực thi khi ta có một lời gọi tới hàm đó.

<Tên hàm>([Danh sách các tham số])

- Lưu ý khi sử dùng hàm
  - Phải viết đúng tên hàm (giống phần khai báo)
  - Phải truyền tham số chính xác về số lượng và kiểu tham số (tham số thực)
  - Giá trị của hàm có thể được gán vào biến, hay sử dụng trong biểu thức tính toán, trong câu lệnh.

4.3 Thực thi hàm

- VD: Ta có định nghĩa hàm tính hiệu 2 số nguyên như sau:

```
int hieu2so(int a, int b)
{
        return a -b;
}
```

Tham số hình thức

Khi có nhu cầu tính hiệu 2 số nguyên x và y (x-y) ta có thể gọi hàm như sau:

Tham số thực

```
int hieu=hieu2so(x,y);
```

4.3 Thực thi hàm

- VD: Gọi hàm sai

```
int hieu=hieu2so();
```

```
int hieu=hieu2so(x);
```

```
int hieu=hieu2so x,y ;
```

```
int hieu=hieu2so(x, y, z);
```

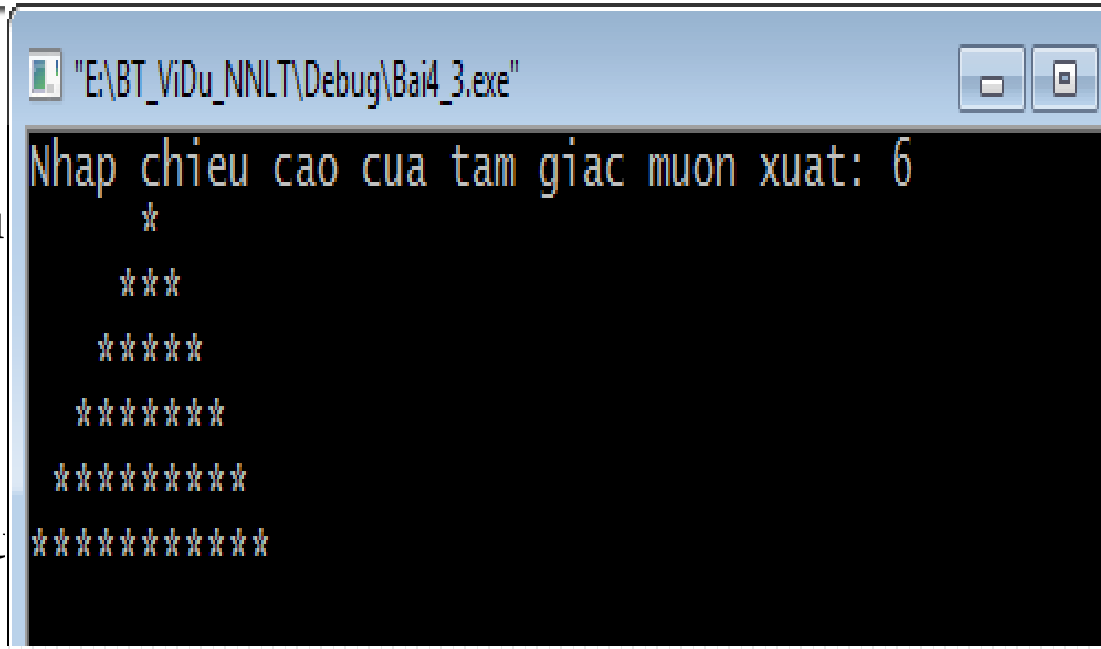
```
int hieu=hieu2so(x, y); với y là một mảng
```

4.3 Sử dụng hàm

- Nguyên lý hoạt động khi gọi hàm
  - Gán giá trị của tham số thực vào tham số hình thức
  - Thực thi lần lượt từng câu lệnh trong hàm
  - Kết thúc khi gặp câu lệnh return hoặc dấu } đóng hàm
  - Tiếp túc trở về thực hiện chương trình đã gọi hàm

4.3 Sử dụng hàm

```
void XuatC(char x, int n)
{       int i;
        for(i = 1; i <= n; i++) cou
}
void main(){
        int h, i;
        printf("Nhap chieu cao cu
        for(i = 1; i <= h; i++){
                XuatC(' ', h-i); //gọi hàm thực thi
                XuatC('*',2*i-1); //gọi hàm thực thi
                XuatC('\n',1); //gọi hàm thực thi
        }
        getch();
```



4.4 Truyền tham số

- Có hai loại tham số:
  - Tham số dạng tham trị: không thay đổi giá trị của tham số khi truyền vào hàm
  - Tham số dạng tham biến: cho phép thay đổi giá trị của tham số trong quá trình sử dụng hàm. Để khai báo một tham số dạng tham biến ta thêm dấu & vào trước tên biến trong khai báo hàm

4.4 Truyền tham số

Tham trị

```
void swap(int x, int y)
{
    int t=x; x=y; y=t;
}
```

Tham biến

```
void swap(int &x, int &y)
{
    int t=x; x=y; y=t;
}
```

```
int m=5, n=9;
swap(m,n);
printf("m=%d, n=%d", m,n);
```

m'=9

n'=5

m=5, n=9

m=9, n=5

4.5 Kết quả trả về

Có hai trường hợp

- Hàm có kết quả trả về*: trong thân hàm bắt buộc phải có lệnh return. Giá trị hàm muốn trả về là giá trị sau từ return*

```
double mean(int num1,int num2)
{
        return ( num1 + num2)/2;
}
```

4.5 Kết quả trả về

- Hàm không có kết quả trả về: *không có return hoặc có return nhưng không có giá trị phía sau*

```
void inHCNsao(int a,int b)
{
        for(int i=0;i<a;i++)
        {
                for(int j=0;j<b;j++)
                        printf(“* ”);
                scanf(“\n”);
        }
}
```

4.6 Prototype của hàm

- Là khai báo nguyên mẫu của hàm: giống như định nghĩa hàm nhưng không có phần thân hàm.

```
<Tên Kiểu><Tên Hàm> ([Danh sách các đối số]);
```

- VD:

```
double atof(char s[]);
void func(int i,int j);
double luythua(double n, int so_mu);
```

4.7 Các hàm chuẩn

- Làm các hàm có sẵn trong ngôn ngữ lập trình C, có sẵn trong các thư viện
- VD:

  Hàm scanf()

  Hàm printf()

  Hàm avg()

  Hàm sqrt()

4.8 Thư viện hàm

- Thư viện hàm là tập hợp các hàm đã được xây dựng trước.
- Mỗi thư viện hàm chứa các hàm theo một công dụng riêng.
- Ví dụ:

  math.h.

  string.h

  ctype.h

4.9 Sự đệ qui

- Đệ qui là việc định nghĩa một vấn đề dựa vào chính vấn đề đó nhưng ở mức độ khác
- VD:

n!= n*(n-1)!

Tổng của mảng n phần tử = tổng mảng có n-1 phần tử cộng với giá trị của phần từ ở vị trí n-1

4.9 Sự đệ qui

- Đệ qui là việc định nghĩa một vấn đề dựa vào chính vấn đề đó nhưng ở mức độ khác
- VD:

n!

T có n-1 phần tử

5! = 5 * 4!

4! = 4 * 3!

3! = 3 * 2!

2! = 2 * 1!

1

=> 5! = 120

4.9 Sự đệ qui

- Hàm đệ qui là hàm được định nghĩa dựa vào chính nó.
- Hàm đệ qui gồm 2 phần
  - Phần cơ sở: là phần không cần không cần đệ qui, dễ dàng tính được giá trị
  - Phần đệ qui: là phần thực hiện đệ qui
- VD: trong công thức tính n!
  - Phần cơ sở: n=1 thì n!=1
  - Phần đệ qui: n!=n*(n-1)!

4.9 Sự đệ qui

- Xác định các phần cơ sở và phần đệ qui của bài toán sau
  - Tính các số tự nhiên từ 1 tới n
  - Tính $\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2+\cdots+\sqrt{2}}}}}$ (n dấu căn)
  - Tìm UCLN của 2 số nguyên a và b

4.9 Sự đệ qui

- Viết hàm đệ qui: ta cũng viết thành 2 phần
- VD: Viết hàm tính n!

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

Phần cơ sở

Phần đệ qui

Bài tập

Làm bài tập 1, 2, …,15 trang 62


STOP
BLUEBUDDIES