


Trường ĐH Nguyễn Tất Thành Viện Đào tạo Quốc tế	Môn: Lập trình cơ bản Lab 04: Hàm con - Function	
--	---	---

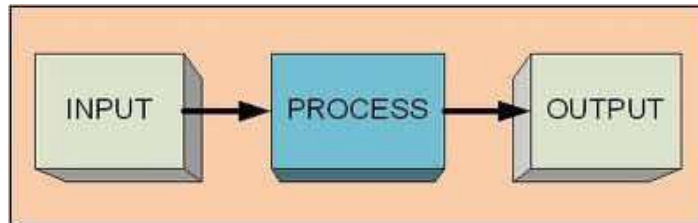
A. MỤC TIÊU:

- Trình bày được các thành phần của một hàm trong C/C++.
- Cài đặt được hàm trong C/C++.
- Gọi được hàm và truyền được tham số trong C/C++.
- Phân biệt tham trị và tham biến.
- Xây dựng được hàm đệ qui trong C/C++.

B. NỘI DUNG THỰC HÀNH

I. Tóm tắt lý thuyết

I. Tóm tắt lý thuyết



1. Các thành phần của hàm:

Hàm là một đoạn chương trình dùng để giải quyết một vấn đề cụ thể nào đó. Một hàm gồm các thành phần sau:

- Tên hàm: do người lập trình tự đặt theo qui tắc đặt tên của C++
- Input: đầu vào của hàm
- Output: đầu ra của hàm
- Processing: quá trình xử lý của hàm (thuật toán)

Ví dụ: Viết hàm tính diện tích hình tròn khi biết bán kính

- Tên hàm: `tinhDienTichHinhTron`
- Input: bán kính
- Output: diện tích hình tròn

□ Processing: $S = \text{bán kính} * \text{bán kính} * \text{PI}$ (Trong đó, PI là hằng số có giá trị là 3.14)

2. Cú pháp khai báo hàm

```

kiểu trả về> <Tên hàm> (<danh sách tham số>)
{
    //Câu lệnh thân hàm
}
  
```

Trong đó

- <kiểu trả về>: kiểu dữ liệu của output, nếu hàm không có output thì kiểu trả về là void
- <danh sách tham số>: là tất cả các input của hàm (bao gồm cả kiểu dữ liệu của input và tên của input, các input cách nhau bởi dấu “,”)

- Thân hàm: thuật toán (các câu lệnh bắt đầu từ input tìm ra output)

Ví dụ: Viết hàm tính diện tích hình tròn khi biết bán kính

```
float tinhDienTichHinhTron(int bankinh)
{
    float s;
    s = bankinh * bankinh * 3.14;
    return s;
}
```

Ví dụ: In ra màn hình n lần từ Hello, mỗi từ trên 1 dòng

```
void inHello (int n)
{
    for(int i=0; i < n; i++)
        printf("Hello\n");
}
```

3. Cách gọi hàm

- Ta có thể gọi hàm ở bất kỳ vị trí nào, trong chương trình chính hoặc trong thân của một hàm khác
- Hàm phải được khai báo ở vị trí sau phần khai báo nguyên mẫu hàm (function prototype)
- Cách gọi hàm:

A. Hàm có kiểu trả về void:

<Tên hàm> (<ds tham số>);

Ví dụ: Muốn in ra màn hình 10 lần từ Hello ta gọi hàm như sau:

```
inHello (10);
```

B. Hàm có kiểu trả về khác void

- Khai báo biến có cùng kiểu trả về với hàm
- Gán giá trị của hàm vào cho biến vừa khai báo <tên biến> = <Tên hàm>(<ds tham số>);
- Xử lý biến theo ý muốn

Ví dụ:

Ta có bán kính hình tròn $r = 6$.

Cần tính diện tích hình tròn rồi in diện tích đó ra màn hình ta có thể gọi như sau:

```
int r = 6;
float dt = tinhDienTichHinhTron(r);
printf("Diện tích hình tròn là: %.2f", dt);
```

4. Tham số của hàm

Hàm có 2 loại tham số:

- **Tham số trị:** giá trị tham số không thay đổi so với trước và sau khi thực hiện hàm, dù trong hàm, giá trị tham số có thay đổi. Nó được sử dụng khi không có nhu cầu thay đổi giá trị của

tham số sau khi thực hiện hàm.

- **Tham số biến (tham chiếu):** giá trị tham số thay đổi so với trước và sau khi thực hiện hàm nếu trong hàm, giá trị tham số có thay đổi. Nó được sử dụng khi có nhu cầu thay đổi hoặc nhận giá trị của tham số sau khi thực hiện hàm.

Ví dụ: xét hàm hoán đổi giá trị 2 biến số nguyên x và y.

Khi dùng tham số trị

```
#include <stdio.h>
#include <conio.h>
//-----
//Khai báo prototype
void swap(int x, int y); // dùng tham số trị

//-----
//Hàm main
void main()
{
    int x=5, y=8;
    printf("\n Gia tri x va y truoc khi gọi ham: %d, %d", x, y);
    swap(x, y);
    printf("\n Gia tri x va y sau khi gọi ham: %d, %d", x, y);
    getch();
}
//
//Các hàm con
void swap(int x, int y)
{
    int t=x;
    x=y;
    y=t;
    printf("\n Gia tri x va y sau khi hoan doi trong ham: %d, %d", x, y);
}
```

Kết quả thực thi:

```
Gia tri x va y truoc khi gọi ham: 5, 8
Gia tri x va y sau khi hoan doi trong ham: 8, 5
Gia tri x va y sau khi gọi ham: 5, 8
```

Khi dùng tham biến

```
#include<stdio.h>
#include<conio.h>
//-----
//Khai báo prototype
void swap(int &x, int &y);

//-----
//Hàm main
void main()
{
    int x=5, y=8;
    printf("\n Gia tri x va y truoc khi goi ham: %d, %d", x,y);
    swap(x,y);
    printf("\n Gia tri x va y sau khi goi ham: %d, %d", x,y);
    getch();
}
//
//Các hàm con
void swap(int &x, int &y)
{
    int t=x;
    x=y;
    y=t;
    printf("\n Gia tri x va y sau khi hoan doi trong ham: %d, %d", x,y);
}
```

Kết quả:

```
Gia tri x va y truoc khi goi ham: 5, 8
Gia tri x va y sau khi hoan doi trong ham: 8, 5
Gia tri x va y sau khi goi ham: 8, 5
```

5. Hàm đệ qui:

Hàm đệ qui là hàm gọi lại chính nó trong thân hàm. Xây dựng hàm đệ qui cần xác định 2 điều kiện sau:

- Điều kiện dừng: là trường hợp mà hàm trả về kết quả trực tiếp
- Điều kiện đệ qui: trường hợp mà hàm gọi lại chính nó ở mức độ nhỏ hơn

<kiểu trả về> <Tên hàm> (<danh sách tham số>)

```
{
    if (<thỏa điều kiện dừng>)
    {
        //xử lý
        return <giá trị>

    else
    {
        //xử lý
```

//gọi lại chính nó <Tên hàm> (<tham số>);

```
}  
}
```

Ví dụ: Viết hàm tính tổng $S=1+2+3+\dots+n$

- Điều kiện dừng: $n=1 \Rightarrow S(1)=1$
- Điều kiện đệ quy: $n>1: S(n)=n+S(n-1)$

```
int tinhTong(int n)  
{  
    if (n==1)  
        return 1;  
    else  
        return n + tinhTong(n-1);  
}
```

6. Một số lưu ý trong cách đặt tên hàm

- Tên hàm phải thể hiện ý nghĩa xử lý của hàm đó. Do đó phải đặt bằng động từ/cụm động từ, thường bắt đầu bằng động từ chính.
- Tên hàm viết thường ký tự đầu tiên và viết hoa các ký tự đầu từ còn lại.

Ví dụ: *tinhDienTichHinhTron*, *tinhTong*, *xuatCacSoChan*, *tinhThanhTienCuaHoaDon*, *isEmptyList*, *addNodeAtHead*, *get*, *set*, *findElement*,...

7. Cấu trúc chương trình

```
/*  
- Thông tin tác giả  
- Thông tin chương trình  
*/  
//-----  
//Khai báo thư viện  
#include<stdio.h>  
#include<conio.h>  
  
//-----  
//Khai báo hằng số, cấu trúc (struct) nếu có  
#define PI 3.14  
  
//-----  
//Khai báo prototype của hàm con  
void swap(int &x, int &y);  
  
//-----  
void main()  
{  
    //xử lý chương trình  
    getch();  
}  
//-----  
void swap(int &x, int &y)  
{  
    int t=x;  
    x=y;  
    y=t;  
}
```

```
}  
//...
```

II. Bài tập mẫu hướng dẫn thực hành

Các bước thực hiện

- Bước 1: Xác định tên hàm, input, output, thuật toán
- Bước 2: Cài đặt chương trình
- Bước 3: Gọi hàm trong chương trình chính.

Bài 1. Viết hàm tìm ước chung lớn nhất của 2 số nguyên dương a và b. Viết hàm main tính ước chung lớn nhất cho 2 số nguyên nhập vào từ bàn phím, sử dụng hàm đã viết ở trên.

Bước 1: Xác định tên hàm, input, output, thuật toán

- Tên hàm: UCLN (ước chung lớn nhất)
- Input: 2 số nguyên a, b (kiểu int)
- Output: ước chung lớn nhất của a,b là 1 số nguyên
- Sử dụng thuật toán euclid => số lần lặp của bài toán không xác định => sử dụng vòng lặp while
- Thuật toán euclid tìm UCLN của a và b

Nếu $a=b$ thì $UCLN=a$;

Nếu $a>b$ thì $UCLN(a,b)=UCLN(a-b,b)$

Nếu $a<b$ thì $UCLN(a,b)=UCLN(a,b-a)$

Bước 2: Cài đặt chương trình

- Cài đặt hàm

```
int UCLN(int a, int b)  
{  
    while (a!=b)  
    {  
        if (a>b)  
            a = a - b;  
        else  
            b = b - a;  
    }  
    return a;  
}
```

- Gọi hàm

```
int x,y;  
printf("\n nhap vao 2 so nguyen duong:");  
scanf("%d%d",&x,&y);  
int uc = UCLN(x,y);  
printf("Uoc chung lon nhat can tim la:%d", uc);
```

Bài 2. Viết hàm kiểm tra số nguyên dương n có phải là số nguyên tố hay không? Viết hàm main minh họa kiểm tra số nguyên nhập vào từ bàn phím có phải là số nguyên tố hay không?

Bước 1: Xác định tên hàm, input, output, thuật toán

- Tên hàm: **KT_SNT** (kiểm tra số nguyên tố)
- Input: 1 số nguyên dương n (kiểu int)
- Output: true hoặc false (hoặc kiểu trả về là int (1: đúng, 0: sai))
- Thuật toán:

Đếm số ước của n.

- So sánh số ước của n với 2. Nếu số ước bằng 2 thì đó là số nguyên tố, ngược lại không phải số nguyên tố

Bước 2: Cài đặt chương trình

Cài đặt hàm:

```
int KT_SNT(int n)
{
    //dem so uoc cua n;
    int demuoc=0;
    for(int i=1; i<=n; i++)
        if(n%i==0)
            demuoc++;
    //so sanh so uoc voi 2
    if(demuoc==2)
        return 1; //la so nguyen to
    return 0; //khong la so nguyen to
}
```

- Gọi hàm:

```
int so;
printf("\n nhap vao so nguyen muon kiem tra:");
scanf("%d",&so);
int kq = KT_SNT(so);
if(kq==1)
    printf("\n Day la so nguyen to");
else
    printf("\n Day khong phai la so nguyen to");
```

Bài 3. Viết hàm tính x^n bằng hàm sử dụng đệ quy.

Bước 1: Xác định tên hàm, input, output, thuật toán

- Tên hàm: LuyThua
- Input: 1 số thực x, 1 số nguyên dương n
- Output: x^n là một số thực (float)
- Thuật toán đệ quy.
 - Điều kiện dừng: $n=0 \Rightarrow$ Lũy thừa =1.
 - Điều kiện đệ quy $n>0$: $LuyThua(x,n)=x*LuyThua(x,n-1)$;

Bước 2: Cài đặt chương trình

– Cài đặt hàm

```
float LuyThua(float x, int n)
{
    if (n==0)
        return 1;
    return x*LuyThua(x, n-1);
}
```

– Cách gọi hàm giống hàm bình thường

III. Bài thực hành trên lớp

Bài 4. Viết chương trình tính diện tích hình tam giác khi biết ba cạnh của tam giác (yêu cầu viết hàm tính diện tích tam giác).

Bài 5. Viết chương trình cho phép người dùng thực hiện một trong các yêu cầu sau:

1. Tính tổng các ước của số nguyên n.
2. Tìm số ngày của một tháng khi biết tháng và năm.
3. Đổi chỗ 2 số thực x, y.
4. Tìm BCNN của hai số nguyên dương a và b
5. Kiểm tra số nguyên dương n có các chữ số toàn là *chữ số lẻ* hay không?

(Yêu cầu: Mỗi chức năng viết hàm thực hiện, hàm main gọi hàm vừa viết thực thi).

Bài 6. Viết hàm tính tổng các số nguyên tố nhỏ hơn n, trong đó n là số nguyên dương nhập từ bàn phím (yêu cầu viết hàm kiểm tra số nguyên tố, hàm tính tổng các số nguyên tố nhỏ hơn n).

Bài 7. Viết hàm đếm các số chính phương nhỏ hơn n, trong đó n là số nguyên dương nhập từ bàn phím (yêu cầu viết hàm kiểm tra số chính phương và hàm đếm các số chính phương nhỏ hơn n).

Bài 8. Viết hàm tính $S(n) = \sqrt{2 + \sqrt{2 + \dots + \sqrt{2}}}$ (n dấu căn). Viết hàm main thực thi.

Bài 9. Viết hàm tính n! bằng cách sử dụng hàm đệ qui. Viết hàm main thực thi: $n! = 1*2*3*\dots*n$

Bài 10. Viết 2 hàm hiển thị menu để hiển thị danh sách các hàm đã có (mỗi hàm là 1 số).

Hàm thực thi thực hiện lặp lại việc:

- + hiển thị menu danh sách hàm.
- + cho người dùng nhập số tương ứng của bài muốn thực thi.
- + thực thi hàm đã được chọn. Nếu người dùng nhập 0 thì kết thúc chương trình.

Bài 11. Cài đặt các bài tập tuần 2 dạng hàm con và gọi thực thi.

IV. Bài tập về nhà:

Bài 12. Cài đặt tất cả các hàm ở bài thực hành tuần 2 và tuần 3 về dạng hàm.

Bài 13. Viết hàm kiểm tra một số nguyên x có là số hoàn thiện (còn gọi là số hoàn chỉnh, số hoàn hảo) không? Biết rằng số hoàn thiện là số có tổng các ước của nó (trừ chính nó) bằng chính nó. Ví dụ: 6 có các ước 1, 2, 3 và $1+2+3 = 6$ nên 6 là số hoàn thiện.

Bài 14. Xuất n số nguyên tố đầu tiên với $n>0$ nhập từ bàn phím.

Ví dụ: $n=5$. 5 số nguyên tố đầu tiên là: 2 3 5 7 11

Bài 15. Xuất n số chính phương đầu tiên với $n>0$ nhập từ bàn phím. Ví

dụ: $n = 3$. 3 số chính phương đầu tiên là: 1 4, 9