

KỸ THUẬT LẬP TRÌNH C/C++

Hàm và cấu trúc chương trình

Thi-Lan Le

Thi-Lan.Le@mica.edu.vn; lan.lethi1@hust.edu.vn

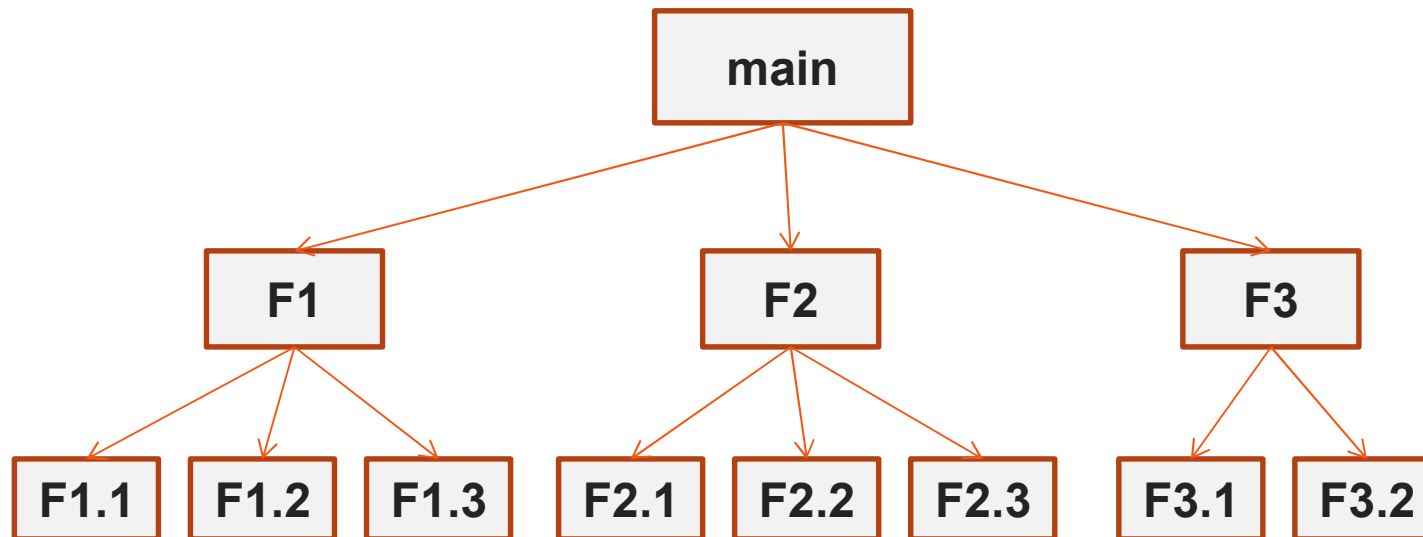
[Webpage: http://www.mica.edu.vn/perso/Le-Thi-Lan](http://www.mica.edu.vn/perso/Le-Thi-Lan)

Các nội dung chính

- ◆ Cấu trúc một chương trình
- ◆ Hàm
- ◆ Cách tổ chức chương trình
- ◆ Một số loại biến đặc biệt trong chương trình
- ◆ Con trỏ hàm
- ◆ Hàm callback

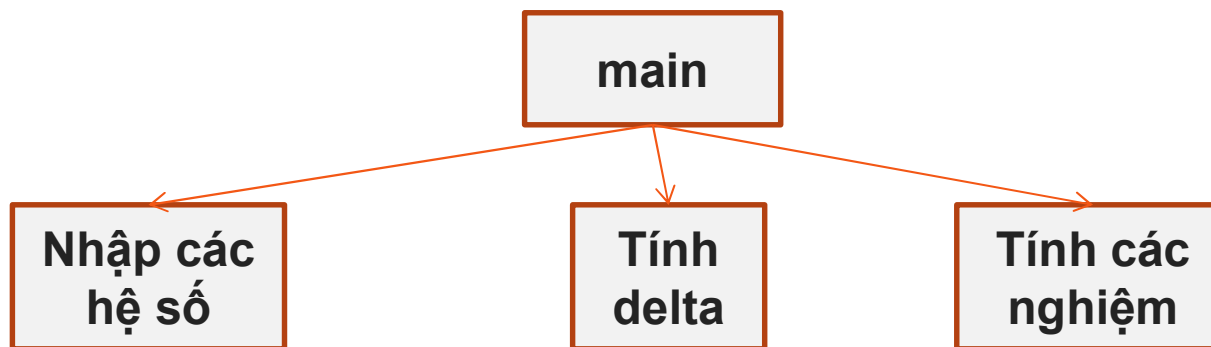
Cấu trúc một chương trình

◆ Mô hình hướng chức năng



Cấu trúc một chương trình

- ◆ VD: các chức năng của chương trình giải PT bậc 2



• Tính $\Delta = b^2 - 4ac$

+) $\Delta > 0$: PT có 2 nghiệm: $x_1 = \frac{-b - \sqrt{\Delta}}{2a}$, $x_2 = \frac{-b + \sqrt{\Delta}}{2a}$

+) $\Delta = 0$: PT có nghiệm kép: $x_1 = x_2 = \frac{-b}{2a}$

+) $\Delta < 0$: PT vô nghiệm.

Hàm

- ◆ Khái niệm
- ◆ Phân loại
- ◆ Cấu trúc một hàm
- ◆ Các thao tác cơ bản trên hàm

Hàm

◆ Khái niệm:

- Là một đơn vị chức năng của chương trình. Mỗi chức năng của chương trình được cài đặt bằng một hoặc nhiều hàm
- Nên hàm còn được gọi là “chương trình con”

◆ Phân loại: trong C phân làm 2 loại hàm:

- Hàm **main**: là hàm chính của chương trình
- Hàm con: là các hàm còn lại

Hàm

- Cấu trúc của một hàm: gồm 2 phần
 - **Phần đầu** (header): lại gồm tên hàm, kiểu giá trị trả về (***void*** hoặc một kiểu DL), và danh sách các tham số (có thể rỗng)
 - **Phần thân** (body): là khối lệnh chứa các lệnh cài đặt cho chức năng của hàm.

```
float Delta(float a, float b, float c) HEADER
```

```
{  
    float d;  
    d = b*b - 4*a*c;  
    return d;  
}
```

BODY

Hàm

◆ Các thao tác cơ bản với hàm:

- Định nghĩa hàm (definition)
- Khai báo hàm (declaration)
- Gọi hàm (call)

Định nghĩa hàm

- Là phần cài đặt chi tiết cho một hàm
- Mỗi hàm cần **có một và chỉ một** định nghĩa
- Định nghĩa này có thể được đặt trước hoặc sau hàm **main**
- Không cho phép đặt định nghĩa hàm này lồng trong định nghĩa của hàm khác, kể cả hàm **main**.
- Khi định nghĩa một hàm cần phải xác định đầy đủ, chi tiết tất cả các thành phần của hàm đó, gồm phần đầu và phần thân của nó.

Định nghĩa hàm

◆ Cú pháp:

T tên_hàm (T1 v1, T2 v2, ...)

{

 Lệnh 1;

 Lệnh 2;

 ...

}

Header

Body

Định nghĩa hàm

◆ Phần đầu hàm: cần xác định

- Tên hàm
- Kiểu dữ liệu trả về cho hàm (kiểu hàm)
- Tên, kiểu dữ liệu cho các tham số, và kiểu tham số (đầu vào, đầu ra, hoặc cả hai)



Định nghĩa hàm

- ◆ Lưu ý về tham số đầu ra: Trong C, tham số đóng vai trò đầu ra (hoặc vừa đầu vào vừa đầu ra, hoặc chỉ đầu ra) ***phải là kiểu con trỏ***.

Định nghĩa hàm

- Phần thân hàm:

- Là khối lệnh chứa các lệnh xử lý cho phần đầu hàm
- Có thể khai báo thêm các kiểu dữ liệu (biến/hằng) có phạm vi sử dụng cục bộ trong khối lệnh thân hàm
- Các tham số trong phần đầu hàm được sử dụng như các dữ liệu cục bộ, nhưng cần chú ý thêm đến vai trò vào/ra của chúng
- Phần này có thể chứa các lệnh **return** (có hoặc không có tham số) để thực hiện kết thúc khối lệnh (và có trả về giá trị cho hàm này nếu có tham số)

Một số ví dụ định nghĩa hàm

◆ Định nghĩa hàm tính USCLN(a,b)

```
//Cách 1: hàm có giá trị trả về  
int uscln(int a, int b)  
{  
    while(a!=b)  
        if(a>b) a -= b;  
        else b -= a;  
    return a;  
}
```

Một số ví dụ

◆ Định nghĩa hàm tính USCLN(a,b)

```
//Cách 2: hàm không có giá trị trả về  
void uscln(int a, int b, int* u)  
{  
    while(a!=b)  
        if(a>b) a -= b;  
        else b -= a;  
    *u = a;  
}
```

Một số ví dụ

- ◆ Định nghĩa hàm tính tổng của một dãy a có n số

```
//Cách 1: hàm có giá trị trả về  
float sum(float a[], int N)  
{  
    int i;  
    float sf=0;  
    for (i=0;i<N;i++) sf += a[i];  
    return sf;  
}
```


Một số ví dụ

- ◆ Định nghĩa hàm tính tổng của một dãy a có n số

```
//Cách 2: hàm không có giá trị trả về  
void sum(float a[], int N, float* s)  
{  
    int i;  
    float sf=0;  
    for (i=0;i<N;i++) sf += a[i];  
    *s = sf;  
}
```

Khai báo hàm

- Là thao tác nhằm thông báo cấu trúc của phần đầu hàm trước khi gọi hàm đó
- Cú pháp:

T tên_hàm (T1 v1, T2 v2, ...);

Trong đó: Ti: kiểu tham số

vi: tên tham số

- Khai báo hàm nhằm 2 mục đích chính:
 - Đảm bảo việc gọi đúng hàm cần dùng
 - Giúp cho việc tìm và liên kết hàm dễ dàng hơn

Khai báo hàm

- Một số lưu ý khi khai báo hàm:
 - Vị trí khai báo hàm tương tự như vị trí khai báo dữ liệu, và phạm vi của hàm cũng có hai loại **cục bộ** và **toàn cục**, phụ thuộc vào vị trí khai báo như dữ liệu
 - Thao tác này không bắt buộc phải có, nếu trước khi gọi hàm đã có phần định nghĩa của hàm này. Còn nếu để định nghĩa ở sau khi gọi hàm, hoặc để ở file khác thì cần phải có phần khai báo này
 - Khi khai báo thì tên của các tham số không quan trọng, và có thể bỏ đi, nhưng kiểu dữ liệu của chúng thì nhất định phải đầy đủ
 - Các tham số của hàm dùng khi định nghĩa/khai báo được gọi là *tham số hình thức*. Còn sau này khi gọi hàm, các *tham số thực* sẽ được sử dụng để thế chỗ cho các tham số hình thức này

Một số ví dụ khai báo hàm

- ◆ `int uscln(int a, int b);`
- ◆ `int uscln(int , int);`
- ◆ `int uscln(int aa, int bb);`

- ◆ `float sum(float a[], int n);`
- ◆ `float sum(float[], int);`

Gọi hàm

- ◆ Khi muốn sử dụng một hàm đã được định nghĩa, ta cần gọi (call) hàm đó.
- ◆ Cú pháp:

tên_hàm (v1, v2, ...);

Trong đó: vi: tên các tham số thực

- ◆ Một số lưu ý khi gọi hàm:
 - Các tham số thực phải khớp với các tham số hình thức cả về số lượng và kiểu dữ liệu
 - Với hàm có giá trị trả về, ta có thể gọi ở một trong hai cách:
 - Cách lấy giá trị trả về đó: $T = f();$
 - Cách không cần lấy giá trị đó: $f();$

Ví dụ 1: chương trình tính tổng của dãy số

```
#include <stdio.h>
#include <conio.h>
#define N 5
int main(){
    float sum(float [], int); //Khai báo hàm
    float x[N] = {1.5,2,3.5,4,5.5};
    float s = sum (x,N);      //Gọi hàm
    printf("Tong cua day so =%.2f\n ",s);
    getch();
} //end main

float sum(float a[], int n){ //Định nghĩa hàm
    int i;
    float sf=0;
    for (i=0;i<n;i++) sf += a[i];
    return sf;
}
```

Tham số thực và tham số hình thức

Tham số hình thức

```
float sum(float [], int); // Khai báo hàm
float sum(float a[], int n) { // Định nghĩa hàm
    int i;
    float sf=0;
    for (i=0; i<n; i++) sf += a[i];
    return sf;
}
```

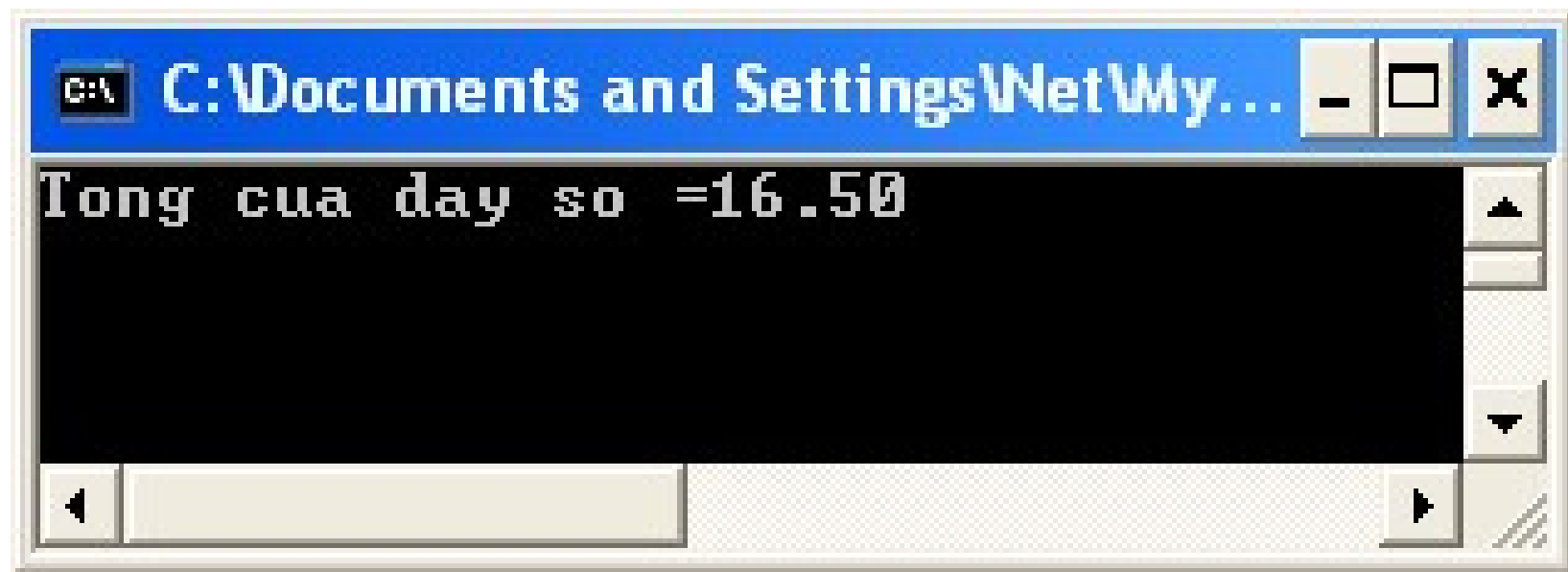
Khi gọi hàm, giá trị của các tham số thực sẽ được copy và thay thế cho các tham số hình thức trong đ/n hàm để thi hành hàm đó.

Tham số thực

```
float s = sum (x, N); // Gọi hàm
```

Ví dụ 1

- ◆ Kết quả chạy chương trình:

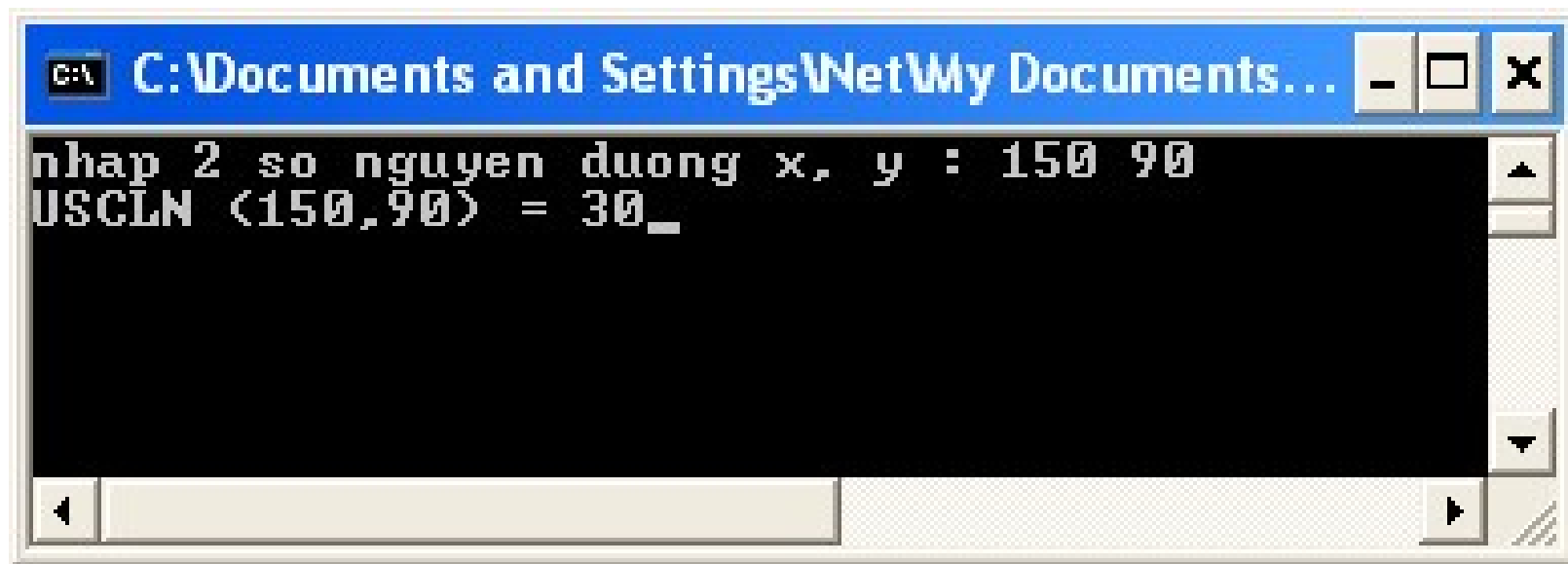


Ví dụ 2: chương trình tính USCLN của 2 số

```
#include <stdio.h>
#include <conio.h>
int uscln(int a, int b);    //Khai báo hàm
//void uscln(int a, int b, int* u);    //Khai báo hàm
void main()    {
    unsigned int x,y,u;
    printf("nhap 2 so nguyen duong x, y : ");
    scanf("%u%u", &x,&y);
    u = uscln(x,y);        //Gọi hàm
    //uscln(x,y,&u);        //Gọi hàm
    printf("USCLN (%d,%d) = %u",x,y,u);
    getch();
} //end main
int uscln(int a, int b){    //Định nghĩa hàm
    while(a!=b)
        if(a>b) a -= b;
        else b -= a;
    return a;
}
```

Ví dụ 2

- ◆ Kết quả chạy chương trình:



A screenshot of a Windows command prompt window. The title bar is blue and contains the text "C:\Documents and Settings\Net\My Documents..." followed by standard window control buttons (minimize, maximize, close). The command prompt area is black with white text. The first line of text is "nhap 2 so nguyen duong x, y : 150 90". The second line is "USCLN (150,90) = 30_". The window has a scroll bar on the right and a status bar at the bottom.

```
C:\Documents and Settings\Net\My Documents...  
nhap 2 so nguyen duong x, y : 150 90  
USCLN (150,90) = 30_
```

Tổ chức chương trình

- ◆ Trong C ta có thể tổ chức một chương trình theo 2 cách:
 - | Tất cả các phần của chương trình nằm trên 1 tệp
 - | Chia các phần của chương trình trên nhiều tệp khác nhau. Khi có nhiều tệp chương trình, thì chúng thường được tổ chức trong một **project**.

Tổ chức chương trình trên nhiều tệp

- Mục đích của việc tổ chức chương trình trên nhiều tệp:
 - Hỗ trợ việc phân chia chương trình thành các modul nhỏ hơn, và mỗi modul đó sẽ được cài đặt trên một tệp
 - Hỗ trợ việc phát triển chương trình theo nhóm gồm nhiều người lập trình, khi đó cần phải chia chương trình ra làm nhiều modul, và mỗi người cần viết một hoặc một số modul trong đó; sau đó đến cuối cùng cần phải lắp ghép tất cả các modul đó lại với nhau để thành một chương trình hoàn chỉnh
 - Hỗ trợ việc tái sử dụng các thành phần của chương trình một cách thuận tiện, qua việc xây dựng các tệp thư viện

Tổ chức chương trình trên nhiều tệp

◆ Có 2 loại tệp chủ yếu trong C:

- Tệp chương trình nguồn (**source file**): thường có phần mở rộng là “.c”: là tệp chủ yếu chứa định nghĩa của các thành phần dữ liệu và hàm
- Tệp phần đầu (**header file**): thường có phần mở rộng là “.h”, là tệp thường chứa các khai báo dữ liệu hay các hàm con

Ví dụ

- ◆ Chương trình tính tổng của 2 dãy số, rồi tìm USCLN của 2 tổng đó. Chương trình này được tổ chức trên 3 tệp:
 - **main.c**: chứa hàm main(), trong đó chứa lời gọi đến các hàm tính tổng 1 dãy số và tính USCLN
 - **myLib.c**: chứa định nghĩa các hàm tính tổng 1 dãy số và tính USCLN
 - **myLib.h**: chứa khai báo cho các hàm tính tổng 1 dãy số và tính USCLN

Tập main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "myLib.h"
int main(int argc, char *argv[])
{
    //float sum(float [], int);
    float x[N] = {1,3,5,7,9,11};
    float y[N] = {2,4,6,8,10,12};
    float s1 = sum (x,N);
    float s2 = sum (y,N);
    printf("Tong cua day so 1 =%.0f\n",s1);
    printf("Tong cua day so 2 =%.0f\n",s2);
    printf("USCLN cua tong 2 day = %d\n",
           uscln((int)s1, (int)s2));
    system("PAUSE");
}
```

Tệp myLib.c

```
//#include "myLib.h"
//extern const int N = 6;
int uscln(int a, int b){
    while(a!=b)
        if(a>b) a -= b;
        else b -= a;
    return a;
}
float sum(float a[], int n){
    int i;
    float sf=0;
    for (i=0;i<n;i++) sf += a[i];
    return sf;
}
```


Tệp myLib.h

```
#define N 6  
//const int N = 6;  
int uscln(int a, int b);  
float sum(float [], int );
```

Các loại biến đặc biệt

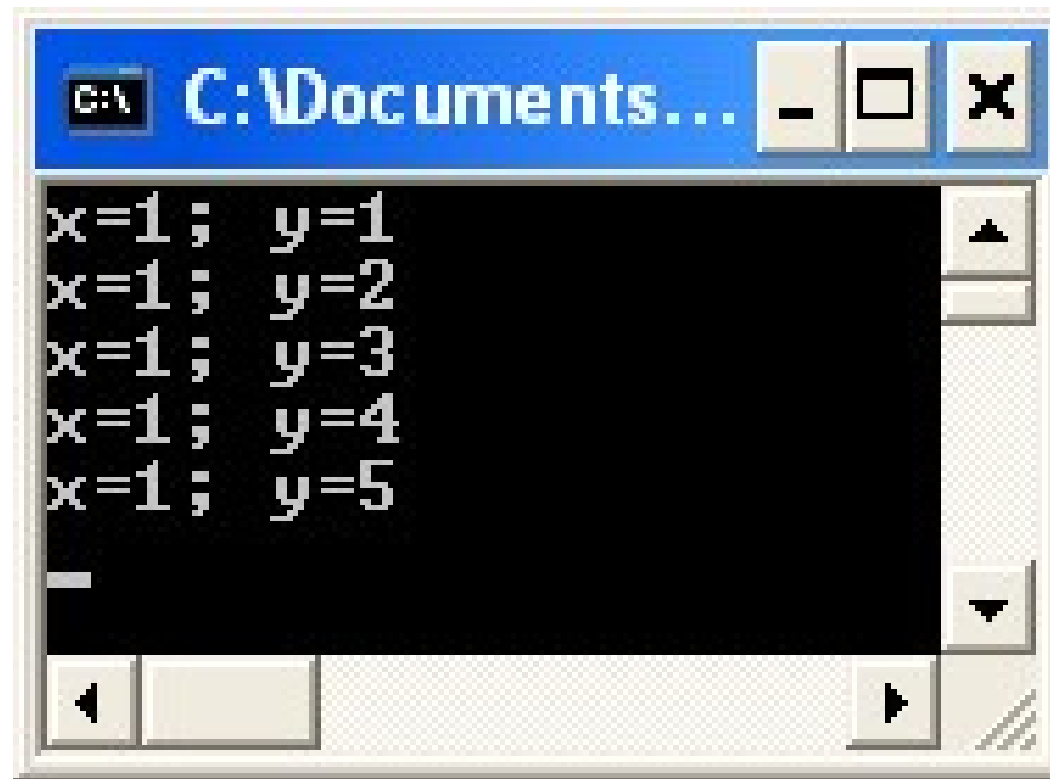
- Biến kiểu **static**: là loại biến có phạm vi sử dụng giống như biến non-static thông thường (có thể cục bộ hoặc toàn cục), nhưng lại có vòng đời trong suốt vòng đời của cả chương trình
- Biến kiểu **extern**: là biến ngoài (external), tức là khi có một biến toàn cục mà phạm vi sử dụng của nó vượt ra ngoài tệp chính chứa nó, thì ở tệp khác muốn sử dụng biến này thì phải khai báo biến đó với từ khóa này.

Ví dụ về biến **static**

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int i;
    for (i=0; i<5; i++) {
        int x=0;
        static int y=0;
        x++;
        y++;
        printf("x=%d; y=%d\n", x, y) ;
    }
    system("PAUSE");
}
```

Ví dụ về biến **static**

- ◆ Kết quả chạy:



```
C:\Documents...  
x=1; y=1  
x=1; y=2  
x=1; y=3  
x=1; y=4  
x=1; y=5
```

Con trỏ hàm

- ◆ Ý nghĩa: là dữ liệu kiểu con trỏ mà trỏ vào hàm. Bản thân tên hàm cũng là một con trỏ.
- ◆ Các thao tác cơ bản:
 - ***Khai báo***
 - ***Gán***
 - ***Gọi hàm qua con trỏ***

Con trỏ hàm - Các thao tác

- ◆ **Khai báo:** giả sử muốn khai báo con trỏ hàm pf mà sẽ trỏ đến hàm f có dạng:

$$\mathbf{T} \quad \mathbf{f} (\mathbf{T}_1 \quad \mathbf{a}_1, \quad \dots, \quad \mathbf{T}_m \quad \mathbf{a}_m) ;$$

Khi đó, pf sẽ được khai báo như sau:

$$\mathbf{T} \quad (*pf) \quad (\mathbf{T}_1, \quad \dots, \quad \mathbf{T}_m) ;$$

Con trỏ hàm - Các thao tác

◆ Gán:

$pf = f;$ //Gán hàm f cho con trỏ pf

Con trỏ hàm - Các thao tác

- ◆ **Gọi hàm qua con trỏ:** sau khi con trỏ hàm pf đã được gán giá trị là hàm cần gọi f , thì hàm f có thể được gọi qua pf như sau:

$pf(a_1, \dots, a_m)$; hoặc
 $(*pf)(a_1, \dots, a_m)$;

Ví dụ

```
#include<stdio.h>
int addFunc(int, int);
int mulFunc(int, int);
main(void)
{
    int (*pf) (int,int); //Khai báo
    int c;
    do
    {
        printf("Chon ham (1:ham +, 2:
ham *, 0: thoat):");
        scanf("%d", &c);
        if (c==0) break;
        else if (c==1)
            pf=addFunc; //Gán
        else pf = mulFunc;
```

```
int result1,result2;
/* Goi ham qua con tro */
result1 = pf(15,20); //Goi hàm

/* Goi ham theo cach dung tham
chieu nguoc ve ham goc*/
result2 = (*pf) (15,20);

printf("result1 = %d result2 =
%d\n",result1,result2);
    }
    while (c);
}
```

Ví dụ (tiếp)

```
int addFunc(int x, int y)
{
    return x+y;
}
```

```
int mulFunc(int x, int y)
{
    return x*y;
}
```

Hàm callback

- ◆ *Khái niệm:* hàm *callback* là hàm được truyền đến hàm khác để được gọi, thông qua tham số là con trỏ hàm.
- ◆ *Ý nghĩa:* nhằm tham số hóa hàm sẽ được gọi.

```
//callback function
void cbf() {...}

//calling function contains
function pointer as
argument

void cf(void (*fp)()) {
    fp();
}

void main() {
    void (*f)() = cbf;
    cf(f);
}
```

Hàm callback – Ý nghĩa

Callback functions

```
cbf1(){...}  
cbf2(){...} ...
```

```
f = cbfk;
```

Calling function cf (f)

```
...  
f();  
...
```

Normal function

```
cbf(){...}
```

Calling function cf ()

```
...  
cbf();  
...
```

Ví dụ

- ◆ Xây dựng một hàm callback:

```
sort(int a[], int N, void (*f))
```

để sắp xếp dãy a gồm N phần tử, theo giải thuật sắp xếp được chọn bởi tham số f (là con trỏ hàm sẽ trỏ đến hàm cài đặt cho giải thuật sắp xếp được chọn).

Ví dụ

```
void insertSort(int a[], int N){
    printf("Insert Sort\n");
}
void quickSort(int a[], int N){
    printf("Quick Sort\n");
}
```

//Callback function

```
void sort(int a[], int N, void
(*f)(int [], int)){
    f(a,N);
}
```

```
int main()
{
    void (*f)(int[], int);
    int a[] = {3, 2, 4, 5, 1};
    //Chọn InsertSort
    f = insertSort;
    sort(a, 5, f);

    //Chọn QuickSort
    f = quickSort;
    sort(a, 5, f);

    printf("Done!");
}
```

Tóm tắt nội dung đã học

- ◆ Cấu trúc các chức năng của một chương trình
- ◆ Hàm con và các thao tác cơ bản
- ◆ Các cách tổ chức chương trình trên 1 tệp và trên nhiều tệp
- ◆ Một số loại biến đặc biệt trong chương trình như biến **static** và **extern**

Hàm

TT	Yêu cầu	Mẫu
1	Viết hàm tìm x^n	<code>double Pow(double, int)</code>
2	Viết hàm tìm $N!$	<code>int Factorial(int)</code>
3	Viết hàm tìm số Fibonacci tại t	<code>int Fibonacci(int)</code>
4	Viết hàm tìm số nghiệm và giá trị các nghiệm của phương trình bậc 2	<code>int Root(double, double, double, double *, double *)</code> // hoặc <code>int Root(double, double, double, double *)</code>
5	Viết hàm tính °K và °F từ °C	<code>void Temperature(double, double *, double *)</code>

Mảng

TT	Yêu cầu	Mẫu	Ví dụ
1	Viết hàm in mảng số nguyên có n phần tử	<code>void Print(const int *, int)</code>	
2	Viết hàm tính tổng của mảng các số nguyên có n phần tử	<code>int Sum(const int *, int)</code>	
3	Viết hàm tính trung bình cộng của mảng các số nguyên có n phần tử	<code>double AVERAGE(const int *, int)</code>	
4	Viết hàm tìm giá trị của đa thức bậc n tại x (hệ số của đa thức là một mảng số thực)	<code>double Polynom(const double *, int n, double)</code>	
5	Viết hàm đổi chuỗi ký tự sang số	<code>double Number(const char *)</code>	<code>cout << Number("12.34.56") + 1;</code> Cho kết quả trên màn hình: 13.34
6	Viết hàm chuẩn hóa họ tên tiếng Việt	<code>char * Viet(char *)</code>	<code>cout << Viet("vu hai minh");</code> Cho kết quả trên màn hình: Vu Hai Minh
7	Viết lại các hàm 1 ... 5 theo phương pháp duyệt mảng bằng con trỏ		

Cấp phát động

TT	Yêu cầu	Mẫu	Ví dụ
1	Viết hàm cho dãy n số Fibonacci đầu tiên	<code>int * FS(int)</code>	<code>int * A = FS(5);</code> Cho A = {1, 2, 3, 5, 8}
2	Viết hàm tạo mảng n phần tử từ mảng số nguyên	<code>int * Clone(const int *)</code>	
3	Viết hàm tạo chuỗi mới chuyển các chữ cái thường thành chữ cái viết hoa từ chuỗi đầu vào	<code>char * ToUpper(const char *)</code>	
4	Viết hàm chuyển đổi số nguyên thành chuỗi ký tự nhị phân	<code>char * Dec2Bin(int)</code>	
5	Viết hàm tìm tần suất của các ký tự trong chuỗi	<code>int * Freq(const char *)</code>	