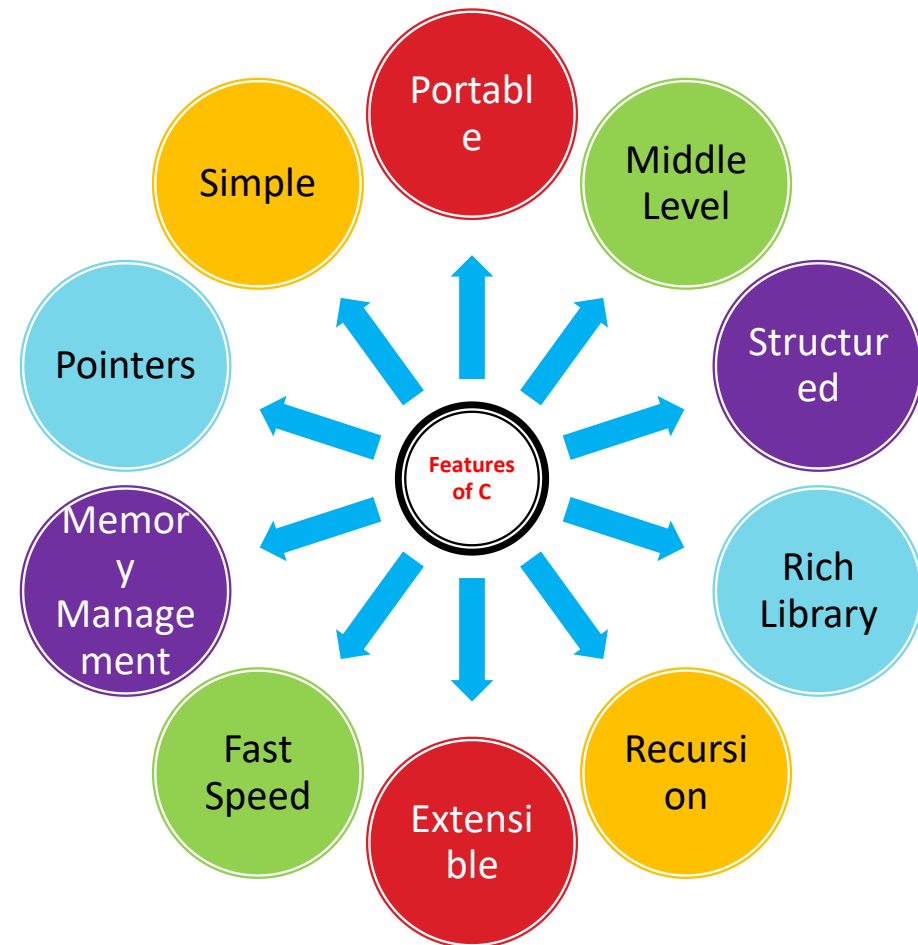


PHƯƠNG PHÁP LẬP TRÌNH

(The C Programming Language)



NỘI DUNG MÔN HỌC

1. Tổng quan về giải thuật
2. Ngôn ngữ lập trình C
3. Cấu trúc điều khiển (*Control structures and statements*)
4. Hàm (*Function*)
5. Mảng 1 chiều (*One array dimension*)
6. Chuỗi ký tự (*String*)
7. Mảng hai chiều (*Two array dimension*)
8. Kiểu dữ liệu cấu trúc (*Struct data type*)
9. Kiểu dữ liệu con trỏ (*Pointer data type*)
10. Tập tin (*File*)

Chương 4

HÀM (Function)



MỤC TIÊU CỦA CHƯƠNG

Sau khi hoàn tất chương này, sinh viên có khả năng hiểu và vận dụng được việc sử dụng hàm trong chương trình.

NỘI DUNG CHƯƠNG

1. Đặt vấn đề

2. Khái niệm về hàm

3. Định nghĩa hàm

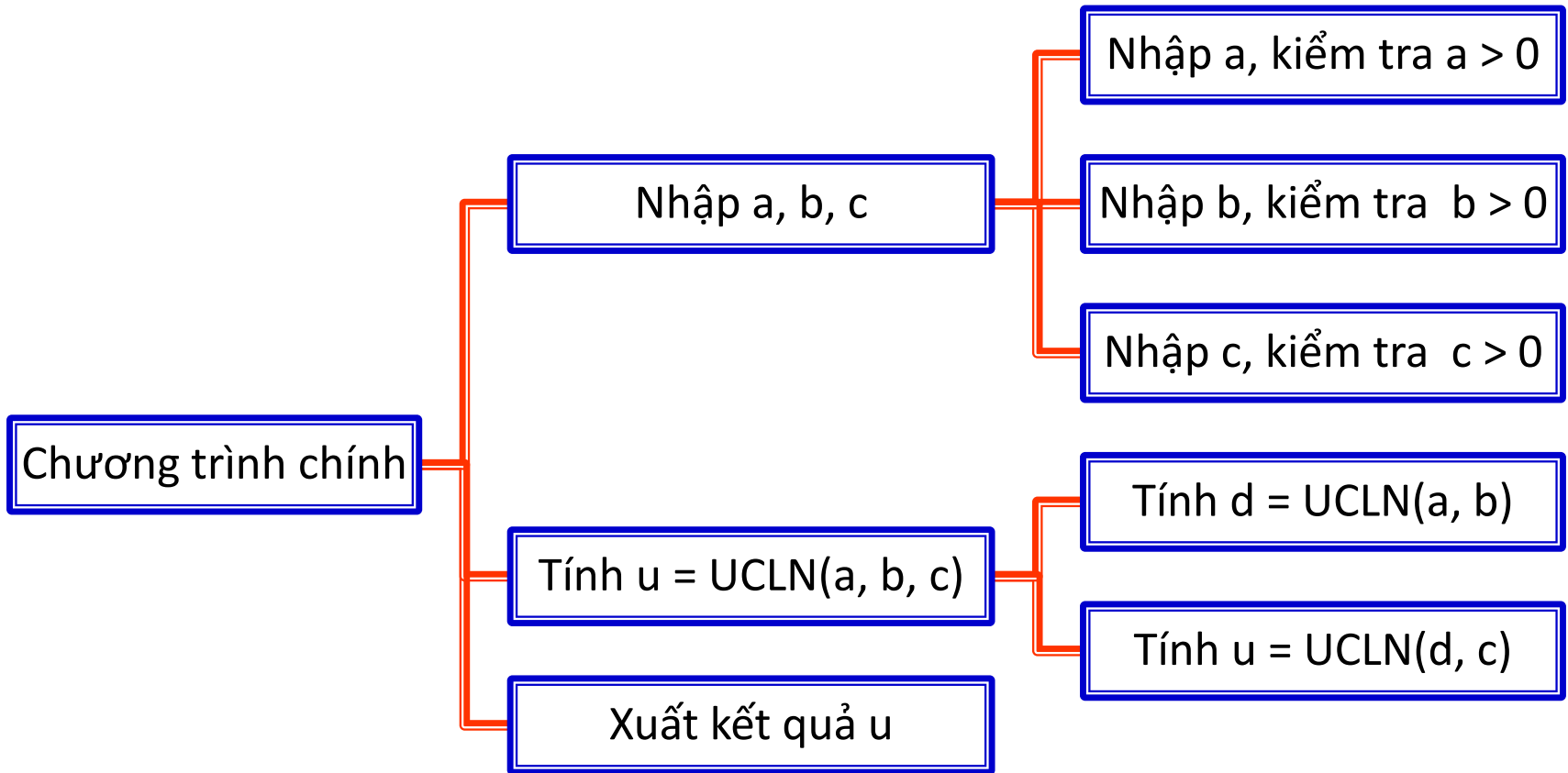
4. Lời gọi hàm

5. Nguyên tắc hoạt động của hàm

6. Biến cục bộ và biến toàn cục

1. ĐẶT VẤN ĐỀ

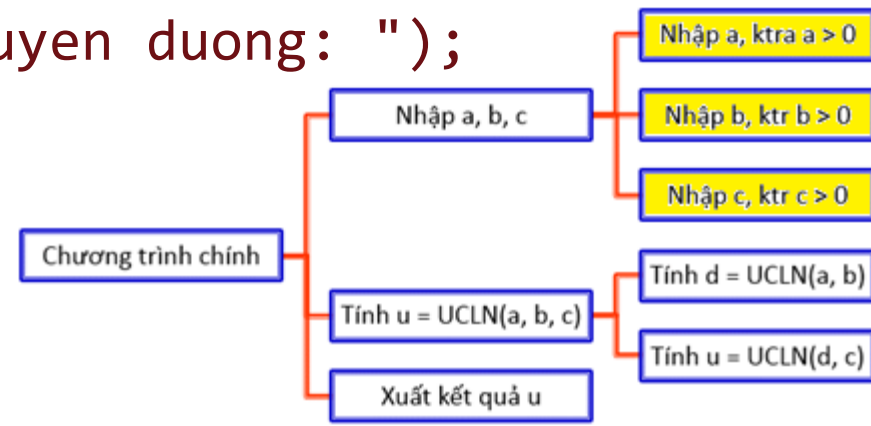
- Yêu cầu của chương trình: Nhập 03 số nguyên dương a , b , c . Tính ước số chung lớn nhất của cả 03 số này.



1. Đặt vấn đề

- *Nhận xét 1*: 3 đoạn lệnh nhập và kiểm tra một số lớn hơn 0 lặp lại 03 lần.

```
int a, b, c;  
do  
{  
    printf("Nhap mot so nguyen duong: ");  
    scanf("%d",&a);  
} while (!(a > 0));  
do  
{  
    printf("Nhap mot so nguyen duong: ");  
    scanf("%d", &b);  
} while (!(b > 0));  
do  
{  
    printf("Nhap mot so nguyen duong: ");  
    scanf("%d", &c);  
} while (!(c > 0));
```

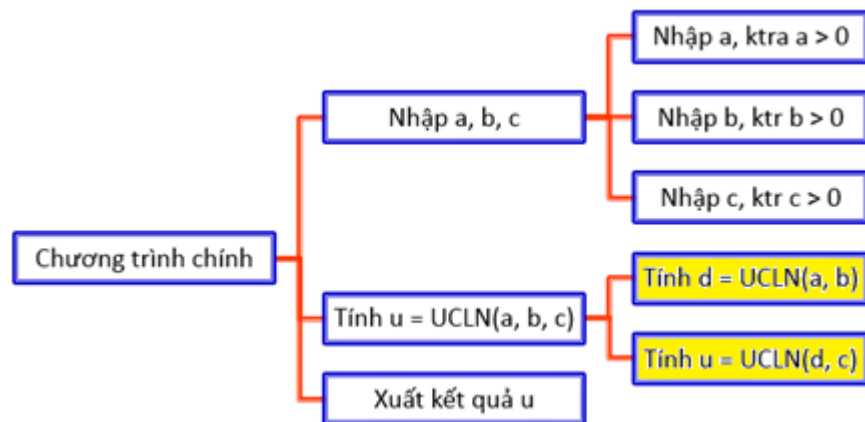


1. Đặt vấn đề

- **Nhận xét 2:** Đoạn lệnh tính ước chung lớn nhất của 02 số **lặp lại 02 lần.**

```
int d, u;  
while (a != b)  
{  
    if (a > b)  
        a -= b;  
    else  
        b -= a;  
}  
d = a;  
while (d != c)  
{  
    if (d > c)  
        d -= c;  
    else  
        c -= d;  
}  
u = d;
```

Cần giải pháp **viết 01 lần** và nhưng có thể **dùng nhiều lần**



1. Đặt vấn đề

– Ví dụ 2

Cho nhập 3 số a,b,c.
Tìm ra số lớn nhất,
số nhỏ nhất và số
lớn nhì?

```
// YÊU CẦU: Cho nhập 3 số a,b,c. Tìm max, mid, min
#include <stdio.h>
#include <conio.h>
void main()
{
    int a, b, c, max, min, mid;
    //B1: Input
    printf("Nhap so thu 1: "); scanf("%d", &a);
    printf("Nhap so thu 2: "); scanf("%d", &b);
    printf("Nhap so thu 3: "); scanf("%d", &c);
    //B2: Process
    //Tìm max
    if ((a >= b) && (a >= c))
        max = a;
    else
        if ((b >= a) && (b >= c))
            max = b;
        else
            max = c;
    //Tìm min
    if ((a <= b) && (a <= c))
        min = a;
    else
        if ((b <= a) && (b <= c))
            min = b;
        else
            min = c;
    //Tìm mid
    mid = (a + b + c) - (max + min);
    //b3: output
    printf("Max= %d\n; Mid= %d; Min=%d", max, mid, min);
}
```

1. Đặt vấn đề

– Ví dụ 2

// YÊU CẦU: Cho nhập 3 số a,b,c. Tìm max, mid, min

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

```
#include <conio.h>
```

```
void main()
```

```
{
```

```
    int a, b, c, max, min, mid;
```

```
    //B1: Input
```

```
    printf("Nhap so thu 1: "); scanf("%d", &a);
```

```
    printf("Nhap so thu 2: "); scanf("%d", &b);
```

```
    printf("Nhap so thu 3: "); scanf("%d", &c);
```

```
    //B2: Process
```

```
    //Tìm max
```

```
    if ((a >= b) && (a >= c))
```

```
        max = a;
```

```
    else
```

```
        if ((b >= a) && (b >= c))
```

```
            max = b;
```

```
        else
```

```
            max = c;
```

```
    //Tìm min
```

```
    if ((a <= b) && (a <= c))
```

```
        min = a;
```

```
    else
```

```
        if ((b >= a) && (b >= c))
```

```
            min = b;
```

```
        else
```

```
            min = c;
```

```
    //Tìm mid
```

```
    mid = (a + b + c) - (max + min);
```

```
    //b3: output
```

```
    printf("Max= %d\n; Mid= %d; Min=%d", max, mid, min);
```

```
}
```

```
// YÊU CẦU: Cho nhập 3 số a,b,c. Tìm max, mid, min
```

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

```
#include <conio.h>
```

```
//Khai báo nguyên mẫu hàm
```

```
int TimMax(int a, int b, int c);
```

```
int TimMin(int a, int b, int c);
```

```
void main()
```

```
{    int a, b, c, max, min, mid;
```

```
    //B1: Input
```

```
    printf("Nhap so thu 1: "); scanf("%d", &a);
```

```
    printf("Nhap so thu 2: "); scanf("%d", &b);
```

```
    printf("Nhap so thu 3: "); scanf("%d", &c);
```

```
    //B2: Process
```

```
    max = TimMax(a, b, c); //Tìm max
```

```
    min = TimMin(a, b, c); //Tìm min
```

```
    mid = (a + b + c) - (max + min); //Tìm mid
```

```
    //B3: output
```

```
    printf("Max= %d\n; Mid= %d; Min=%d", max, mid, min);
```

```
}
```

```
int TimMax(int a, int b, int c)
```

```
{    int max;
```

```
    if ((a >= b) && (a >= c))
```

```
        max = a;
```

```
    else
```

```
        if ((b >= a) && (b >= c))
```

```
            max = b;
```

```
        else
```

```
            max = c;
```

```
}
```

```
int TimMin(int a, int b, int c)
```

```
{    int min;
```

```
    if ((a <= b) && (a <= c))
```

```
        min = a;
```

```
    else
```

```
        if ((b >= a) && (b >= c))
```

```
            min = b;
```

```
        else
```

```
            min = c;
```

```
}
```

```
// YÊU CAU: Cho nhập 3 số a,b,c. Tìm max, mid, min
#include <stdio.h>
#include <conio.h>
//Khởi tạo nguyên mẫu hàm
int TimMax(int a, int b, int c);
int TimMin(int a, int b, int c);
void main()
{
    int a, b, c, max, min, mid;
    //B1: Input
    printf("Nhập số thứ 1: "); scanf("%d", &a);
    printf("Nhập số thứ 2: "); scanf("%d", &b);
    printf("Nhập số thứ 3: "); scanf("%d", &c);
    //B2: Process
    max = TimMax(a, b, c); //Tìm max
    min = TimMin(a, b, c); //Tìm min
    mid = (a + b + c) - (max + min); //Tìm mid
    //B3: output
    printf("Max= %d\n; Mid= %d; Min=%d", max, mid, min);
}
```

```
int TimMax(int a, int b, int c)
{
    int max;
    if ((a >= b) && (a >= c))
        max = a;
    else
        if ((b >= a) && (b >= c))
            max = b;
        else
            max = c;
    return max;
}
```

```
int TimMin(int a, int b, int c)
{
    int min;
    if ((a <= b) && (a <= c))
        min = a;
    else
        if ((b >= a) && (b >= c))
            min = b;
        else
            min = c;
    return min;
}
```

```
#include <stdio.h>
#include <conio.h>
//Khởi tạo nguyên mẫu hàm
void Nhap(int &x);
int TimMax(int a, int b, int c);
int TimMin(int a, int b, int c);
void main()
{
    int a, b, c, max, min, mid;
    //B1: Input
    Nhap(a);
    Nhap(b);
    Nhap(c);
    //B2: Process
    max = TimMax(a, b, c); //Tìm max
    min = TimMin(a, b, c); //Tìm min
    mid = (a + b + c) - (max + min); //Tìm mid
    //B3: output
    printf("Max= %d\n; Mid= %d; Min=%d", max, mid, min);
}
```

```
void Nhap(int &x)
{
    printf("Nhập số thứ 3: "); scanf("%d", &x);
}
```

```
int TimMax(int a, int b, int c)
{
    int max;
    if ((a >= b) && (a >= c))
        max = a;
    else
        if ((b >= a) && (b >= c))
            max = b;
        else
            max = c;
    return max;
}
int TimMin(int a, int b, int c)
{
    int min;
    if ((a <= b) && (a <= c))
        min = a;
    else
        if ((b >= a) && (b >= c))
            min = b;
        else
            min = c;
    return min;
}
```

2.KHÁI NIỆM VỀ HÀM

2.1. Khái niệm

Hàm là một đoạn chương trình độc lập thực hiện trọn vẹn một công việc nhất định sau khi thực hiện xong, hàm có thể sẽ trả về giá trị cho chương trình nơi gọi hàm thực hiện.

Hay nói cách khác, hàm là:

- Một đoạn chương trình có tên, đầu vào và đầu ra.
- Có chức năng giải quyết một số vấn đề chuyên biệt cho chương trình chính.
- Được gọi nhiều lần với các tham số khác nhau.
- Được sử dụng khi có nhu cầu:
 - Tái sử dụng.
 - Sửa lỗi và cải tiến.

2. Khái niệm về hàm

2.2. Cấu trúc chương trình khi tổ chức hàm

**Khối
khai báo**

Bao gồm các khai báo:

- ▶ Thư viện sử dụng.
- ▶ Hằng số sử dụng trong chương trình.
- ▶ Hàm con (các nguyên mẫu hàm - prototype).
- ▶ Các biến toàn cục.
- ▶ Các kiểu dữ liệu tự định nghĩa.

Hàm main()

Chứa các biến, các lệnh và các lời gọi hàm cần thiết trong chương trình.

**Khối chứa các
hàm con**

- ▶ Các hàm con được sắp xếp riêng rẽ, mỗi hàm nằm trên 1 đoạn riêng, rời nhau (không đặt nội dung của hàm này chứa trong hàm khác, hoặc nội dung của 2 hàm có phần giao nhau).
- ▶ Không cần quan tâm thứ tự sắp xếp trước/sau của các hàm.
- ▶ Có thể tạo ra 1 file riêng để chứa các hàm

3 ĐỊNH NGHĨA HÀM

3.1. Định nghĩa hàm

- Định nghĩa hàm là viết nội dung của hàm đó. Mỗi hàm sẽ được định nghĩa một lần trong chương trình và có thể được gọi thực hiện bởi một hàm khác có trong chương trình.
- Giống như hàm *main* mỗi hàm trong C gồm hai phần: phần tiêu đề của hàm (*Function Header*) và phần thân hàm (*Function Body*).

```
Data_type function_name(parameter list)  ⇔  Funtion header  
                                         (không có dấu ; ở cuối)  
{  
    Statements;                           ⇔  Funtion body  
}
```

3 Định nghĩa hàm

3.2. Quy tắc đặt tên hàm

- Chỉ được dùng chữ cái, chữ số hoặc dấu _ để đặt tên hàm.
- Ký tự đầu tiên phải là một **chữ cái** hoặc dấu _.
- Tên hàm không được trùng tên với từ khóa.
- Có phân biệt chữ hoa, chữ thường. Nên đặt rõ nghĩa với ký tự đầu là chữ hoa.

VD: LaSoNgyuenTo, HoanVi, SapXep, ...

- Số ký tự tối đa của tên hàm là 31.

3.3. Kiểu dữ liệu trả về của hàm (đầu ra)

Được xác định dựa vào kết quả của bài toán (output). Gồm 2 loại:

- **void**: Hàm không trả về giá trị. Thường dùng cho *nhóm chức năng: Nhập / xuất dữ liệu, thống kê, sắp xếp, liệt kê.*

```
void Tên_hàm ([danh sách các tham số])  
{  
    Khai báo các biến cục bộ  
    Các câu lệnh/khối lệnh hay lời gọi đến hàm khác.  
}
```

- **Kiểu dữ liệu cơ bản (rời rạc/ liên tục) hay kiểu dữ liệu có cấu trúc:**
thường được sử dụng trong các trường hợp: *Đếm, kiểm tra, tìm kiếm, tính trung bình, tổng, tích, ...*

```
<Kiểu dữ liệu> Tên_hàm ([danh sách các tham số])  
{  
    <Kiểu dữ liệu> kq;  
    Khai báo các biến cục bộ  
    Các câu lệnh/khối lệnh hay lời gọi đến hàm khác giúp  
    xác định giá trị biến kq  
    return kq;  
}
```


3.4. Tham số của hàm (đầu vào) - *parameter*

– Một hàm có thể có nhiều hoặc không có tham số đầu vào.

– Ví dụ

- Hàm không có tham số đầu vào:

```
int NhapSoDuong ()  
{   int a;  
    do  
    {       printf("Nhap mot so nguyen duong: ");  
            scanf("%d",&a);  
    } while (!(a > 0));  
    return a;  
}
```

- Hàm có tham số đầu vào:

```
void NhapSoDuong (int &a)  
{  
    do  
    {       printf("Nhap mot so nguyen duong: ");  
            scanf("%d",&a);  
    } while (!(a > 0));  
}
```

3.4. Tham số của hàm (đầu vào) - *parameter*

- Mọi tham số đầu vào/đầu ra của hàm đều có thể được truyền thông qua tham số của hàm.
- **Tham số hình thức trị (tham trị - *pass-by-value*)**
 - Là chỉ truyền giá trị của biến này cho một biến đóng vai trò làm tham số của hàm (*tham số hình thức*).
 - Được dùng khi ta muốn hàm bị gọi không làm thay đổi giá trị của biến (*tham số thực*) mà nơi gọi hàm truyền đến
 - Ví dụ:

```
int USCLN(int a, int b);  
void main()  
{   int x = 5, y = 7, us;  
    printf("TRUOC khi goi ham, x = %d,  
           y= %d\n", x, y);  
    us = USCLN(x, y);  
    printf("SAU khi goi ham, USCLN cua %d  
           va %d la %d", x, y, us);  
}
```

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

3.4. Tham số của hàm (đầu vào) - *parameter*

- Tham số hình thức biến (tham biến/tham trở - *pass-by-references*)
 - Để truyền theo dạng này, chương trình cần truyền địa chỉ của tham số thực cho hàm được gọi.
 - Được dùng khi ta muốn hàm bị gọi làm thay đổi giá trị của biến (*tham số thực*) mà nơi gọi hàm truyền đến
 - Ví dụ: dạng 1, dùng tham chiếu (&)

```
void HoanVi(int &a, int &b);  
void main()  
{   int x = 5, y = 7;  
    printf("TRUOC khi goi ham, x = %d,  
           y= %d\n", x, y);  
  
    HoanVi(x, y);  
    printf("SAU khi goi ham, x = %d,  
           y= %d\n", x, y);  
}
```

```
void HoanVi(int &a, int &b)  
{  
    int temp = a;  
    a = b;  
    b = temp;  
}
```

3 Định nghĩa hàm

3.4. Tham số của hàm (đầu vào) - *parameter*

- Tham số hình thức biến (tham biến/tham trỏ - *pass-by-references*)
 - Ví dụ: dạng 2, dùng con trỏ (*)

```
void HoanVi(int *a, int *b);  
void main()  
{   int x = 5, y = 7;  
    printf("TRUOC khi goi ham, x = %d,  
           y= %d\n", x, y);  
    HoanVi(&x, &y);  
    printf("SAU khi goi ham, x = %d,  
           y= %d\n", x, y);  
}
```

```
void HoanVi(int *a, int *b)  
{  
    int temp = *a;  
    *a = *b;  
    *b = temp;  
}
```

3.5. Nguyên mẫu hàm

- Về nguyên tắc khi gọi một hàm thì hàm đó phải được khai báo trước khi sử dụng, nếu không chương trình sẽ báo lỗi.
- Nguyên mẫu hàm thực chất là dòng tiêu đề của hàm có thêm dấu chấm phẩy (;) ở cuối.
- Khai báo nguyên mẫu hàm thường được đặt sau các khai báo include, khai báo hằng số, khai báo kiểu dữ liệu người dùng tự định nghĩa, ... và nằm ngay trên hàm **main**.
- Ví dụ:

```
void HoanVi(int *a, int *b);  
void main()  
{   int x = 5, y = 7;  
    printf("TRUOC khi goi ham, x = %d,  
           y= %d\n", x, y);  
  
    HoanVi(&x, &y);  
    printf("SAU khi goi ham, x = %d,  
           y= %d\n", x, y);  
}
```

```
void HoanVi(int *a, int *b)  
{  
    int temp = *a;  
    *a = *b;  
    *b = temp;  
}
```

4. LỜI GỌI HÀM (*funtion call*)

- Lời gọi hàm giống như một lệnh.
- Lời gọi hàm xuất hiện trong chương trình khi có yêu cầu gọi một hàm nào đó thực hiện.
- Lời gọi hàm thường có dạng:

<i>Kiểu trả về</i>	<i>Cú pháp</i>	<i>Ví dụ</i>
void	Tên_Hàm ([danh sách tham số nếu có (i)]);	HoanVi(a,b);
Các kiểu dữ liệu khác	Tên biến (ii) = Tên_Hàm ([danh sách tham số (i)]);	int us=USCLN(a,b);

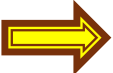
(i): Số lượng và kiểu dữ liệu của các đối số (*Argement*) truyền cho hàm phải tương ứng với số lượng và kiểu dữ liệu của các tham số hình thức trong định nghĩa hàm.

(ii): có kiểu trùng với kiểu dữ liệu trả về của hàm.

5. NGUYÊN TẮC HOẠT ĐỘNG CỦA HÀM

Khi gặp 1 lời gọi hàm trong chương trình, hệ điều hành sẽ thực hiện các bước sau:

- **B1**: Lưu địa chỉ lệnh đang thực hiện vào stack.
- **B2**: chuyển con trỏ lệnh đến địa chỉ chứa hàm.
- **B3**: cấp phát bộ nhớ cho các tham số hình thức và các biến cục bộ (nếu có).
- **B4**: Truyền giá trị hoặc địa chỉ của các tham số thực cho các tham số tương ứng.
- **B5**: Thực hiện các câu lệnh trong thân hàm.
- **B6**: khi gặp lệnh **return** hoặc dấu **}** cuối cùng của thân hàm, hệ điều hành sẽ giải phóng các vùng nhớ đã cấp ở B3 và thoát hàm.
- **B7**: Đưa con trỏ lệnh về địa chỉ đầu tiên có trong stack để thực hiện tiếp chương trình.



```
1 void HoanVi(int &a, int &b);  
2 void main()  
3 {   int x = 5, y = 7;  
4     printf("TRUOC khi goi ham, x = %d,  
5         y= %d\n", x, y);  
6     HoanVi(x, y);  
7     printf("SAU khi goi ham, x = %d,  
8         y= %d\n", x, y);  
9 }
```


```
7 void HoanVi(int &a, int &b)  
8 {  
9     int temp = a;  
10    a = b;  
11    b = temp;  
12 }
```

6. BIẾN CỤC BỘ VÀ BIẾN TOÀN CỤC

6.1. Biến cục bộ

- Là các biến được khai báo trong thân của một hàm.
- Các biến này chỉ có giá trị trong phạm vi từ khi được khai báo trong hàm cho đến khi kết thúc khối lệnh hoặc hàm.
- Ví dụ:


```
int TinhTong(int n);  
int tong = 0;
```



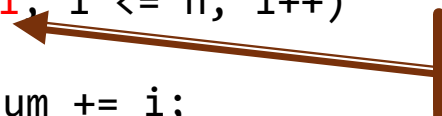
Biến **tong** là biến toàn cục

```
void main()  
{  
    int n = 10;  
    tong = TinhTong(n);  
    printf("Tong cac so tu 1 den %d=%d", n, tong);  
}
```

```
int TinhTong(int n)  
{  
    int sum=0;  
    for (int i; i <= n; i++)  
    {  
        sum += i;  
    }  
    return sum;  
}
```



Biến **sum** là biến cục bộ của hàm



Biến **i** là biến cục bộ của lệnh for

6. Biến cục bộ và biến toàn cục

6.2. Biến toàn cục

- Là các biến được khai báo bên ngoài các hàm.
- Những biến này có giá trị với tất cả các hàm được khai báo sau lệnh khai báo biến.
- Ví dụ:

```
int TinhTong(int n);  
int tong;  
void main()  
{  
    int n = 10;  
    TinhTong(n);  
    printf("Tong cac so tu 1 den %d=%d", n, tong);  
}  
void TinhTong(int n)  
{  
    tong=0;  
    for (int i; i <= n; i++)  
    {  
        tong += i;  
    }  
}
```

6. Biến cục bộ và biến toàn cục

6.3. Trùng tên giữa biến toàn cục và biến cục bộ

- Khi biến toàn cục và biến cục bộ trùng tên thì các **biến cục bộ sẽ ưu tiên được hiểu** trong hàm khai báo nó.
- Ví dụ: xét lại ví dụ trước

```
void Test();
int n;
void main()
{
    int m = 20; //gán giá trị cho biến cục bộ m của hàm main
    n = 10; // gán giá trị cho biến toàn cục
    printf("\nTRUOC khi goi ham Test n = %d, m= %d", n, m);
    Test();
    printf("\nSAU khi goi ham Test n = %d, m= %d", n, m);
}
void Test()
{
    int n = 30, m = 40; //biến cục bộ n, m của hàm Test
    printf("\nTRONG ham Test, n = %d, m= %d", n, m);
}
```

Kết quả: TRUOC khi goi ham Test n=10, m=20
 TRONG ham Test n=30, m=40
 SAU khi goi ham Test n=10, m=20

7. THỰC HÀNH

1. Khai báo tiêu đề cho hàm (prototype) thực hiện các chức năng như sau:
 - a) Hàm thực hiện chức năng rút tiền trên máy ATM.
 - b) Hàm được cài đặt dùng cho máy bán hàng tự động và thanh toán qua thẻ ATM.

7. THỰC HÀNH

2. Lần lượt viết hàm thực hiện các yêu cầu sau:
 - a) Viết hàm đổi một ký tự hoa sang ký tự thường.
 - b) Viết hàm trả về giá trị nhỏ nhất của 4 số nguyên.
 - c) Viết hàm hoán vị hai số nguyên.
 - d) Viết hàm giải phương trình bậc nhất và xuất kết quả ra màn hình
 - e) Viết hàm giải phương trình bậc hai và xuất kết quả ra màn hình
 - f) Viết hàm sắp xếp 4 số nguyên tăng dần.

7. THỰC HÀNH

3. Viết hàm nhận vào số nguyên dương n và thực hiện:
 - a) Đếm số lượng chữ số của số đó
 - b) Tính tổng các chữ số của số đó
 - c) Tính tổng các chữ số lẻ.
 - d) Tính tổng các chữ số chẵn của số đó.
 - e) Tìm số đảo của số n
4. Viết hàm kiểm tra một số có phải là số nguyên tố hay không?
5. Viết hàm kiểm tra một số có phải là số hoàn hảo hay không?
6. Viết hàm đếm số lượng số nguyên tố nhỏ hơn số n

