# Lập trình module - hàm

# Lập trình module - hàm

- Lập trình Module
- Hàm
  - Định nghĩa hàm và gọi hàm.
  - Prototype của hàm.

# Lập trình Module

- Lập trình Module (Modulaar programming): chia nhỏ chương trình giải quyết 1 vấn đề ra thành các chương trình con là những hàm để quản lý các modules.
  - Ví dụ: Một công ty được chia thành nhiều phòng ban, mỗi phòng ban phụ trách một mảng công việc cụ thể, như vậy việc điều hành, quản lý…được dễ dàng, rõ ràng và nhanh chóng hơn.
- Hàm (function): là một tập hợp các lệnh của chương trình để thực hiện 1 công việc nào đó. Hàm là một đơn vị độc lập với chương trình và không được xây dựng hàm bên trong 1 hàm khác.
- Ưu điểm:
  - Tăng tính dễ quản lý của chương trình
  - Giúp đơn giản hóa việc viết chương trình

# Hàm

- Cú pháp khai báo hàm:
  - <Kiểu Dữ Liệu>  TênHàm (<danh sách các tham số>)
  - {
    - <khai báo các biến >;
    - <Khối lệnh>;
    - return <giá trị hoặc kết thúc hàm>;
  - }
- Gọi hàm:
  - TênHàm(<danh sách các tham số>)

# Gọi hàm

- Khi một hàm được gọi chương trình sẽ thực hiện thân hàm bị gọi.
- Sau khi thực hiện hết các lệnh trong hàm được gọi chương trình sẽ quay về lệnh tiếp sau lệnh gọi hàm.
- Hàm main có thể gọi bất kỳ hàm nào đã được định nghĩa.
- Các hàm đã được định nghĩa có thể gọi lẫn nhau.
- Trình biên dịch phải biết các yếu tố sau khi một hàm được gọi:
  - Tên hàm
  - Kiểu dữ liệu trả về của hàm nếu có
  - Số lượng các tham số của hàm
  - Kiểu dữ liệu ứng với từng tham số

# Nguyên mẫu hàm

- Nguyên mẫu (Prototype) của hàm:
  - Cấu trúc khai báo:

    <Kiểu Dữ Liệu>  TênHàm (<danh sách các tham số>);
  - Ví dụ:

    int TinhTong(int a, int b);
  - Dùng để loại trừ việc bắt buộc phải định nghĩa hàm trước khi gọi và thường được đặt trước hàm main, phần đầu chương trình.
  - Sau khi đã sử dụng prototype của hàm thì có thể viết định nghĩa chi tiết hàm ở bất kỳ vị trí nào trong chương trình (thông thường nên đặt sau hàm main).

# Định nghĩa và gọi hàm

```
int Tong(int a, int b);
int Hieu(int a, int b);
int Tich(int a, int b);
float Thuong(int a, int b);
int main(){
    int a, b;   printf("\nNhap a,b:"); scanf("%d %d", &a, &b);
    Menu();
    int tong = Tong(a, b);
    printf("Tong %d+%d=%d", a, b, tong);
    printf("Hieu %d-%d=%d", b, a, Hieu(b, a));
    }
return 0;
}
void Menu(){
    printf("\n======== Menu ==========");
    printf("\n[1. Tinh tong           ]");
    printf("\n[2. Tinh hieu           ]");
    printf("\n[3. Tinh tich           ]");
    printf("\n[4. Tinh thuong         ]");
}
int Tong(int a, int b){ return a+b;}
int Hieu(int a, int b){ return a-b;}
int Tich(int a, int b){ return a*b;}
float Thuong(int a, int b){     return (float)a/ b;}
```

Nguyên mẫu các hàm (định nghĩa trước hàm main()

Gọi hàm Tong

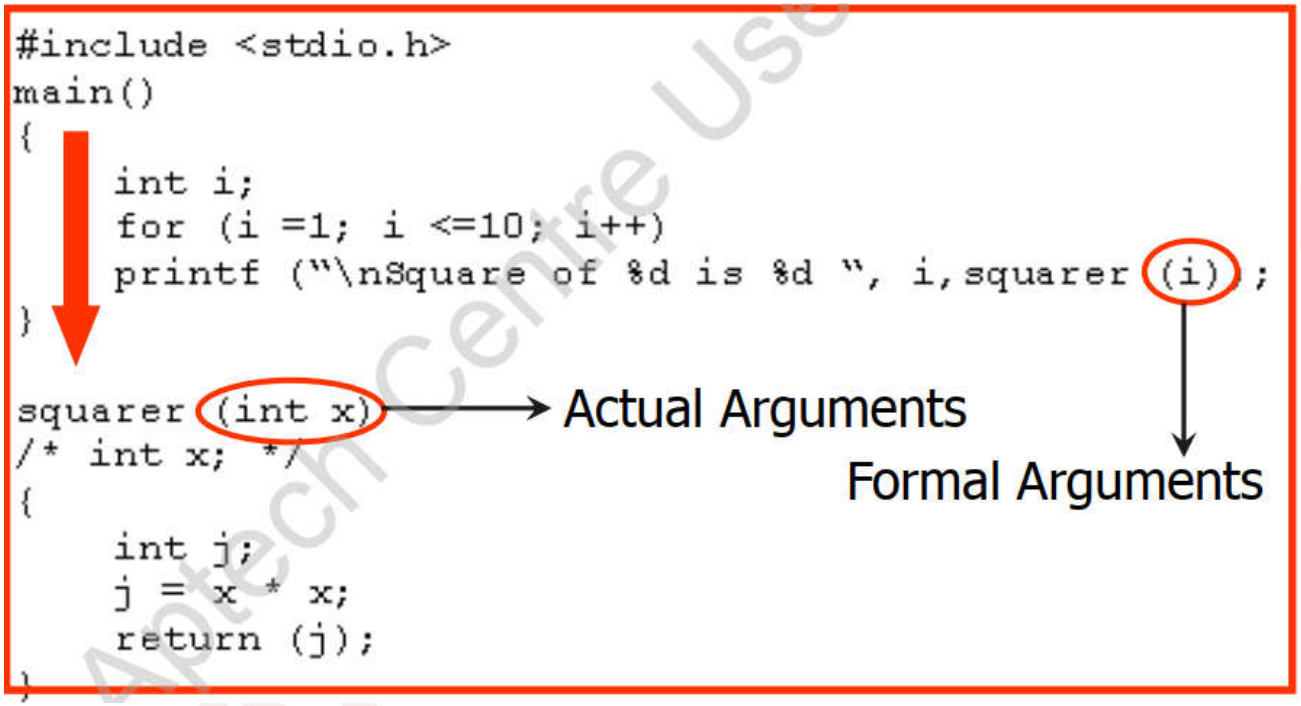
Gọi hàm Hieu

Định nghĩa 5 hàm: Tong, Hieu, Tich, Thuong, Menu

# Lệnh return

- Được sử dụng để kết thúc sự thực thi của chương trình con ngay lập tức.
- Có thể đặt bất kỳ vị trí nào trong thân hàm.
- Một hàm có 1 giá trị trả về trong lệnh return
- Có thể được sử dụng để tránh các kết thúc khác thường của chương trình.
- Nếu không có return hàm sẽ tự động kết thúc khi gặp dấu } kết thúc khối hàm.
- Hàm có kiểu dữ liệu trả về (đinh nghĩa hàm không phải bắt đầu bằng void ) phải có return trả về 1 giá trị hay một biến có cùng kiểu dữ liệu với định nghĩa hàm
- Hàm có kiểu dữ liệu trả vê là void: có thể không cần sử dụng return hoặc sử dụng lệnh return ; (không trả vê giá trị nào)

# Tham số trong hàm




- Chương trinhg tính bình phương các số nguyên từ 1 tới 10.
- Dữ liệu được truyền từ hàm main() tới hàm squarer().
- Hàm làm việc dựa trên dữ liệu sử dụng các tham số

# Biến toàn cục

- Biến toàn cục (global variable): được định nghĩa ngoài tất cả các hàm trong chương trình. Tất cả các khối chương trình, hàm đều có thể sử dụng biến toàn cục trong phạm vi tầm vực của nó.
- Biến toàn cục có thể giúp cho việc chia sẻ một khối lượng lớn dữ liệu giữa các hàm trong chương trình với nhau.
- Tầm vực của biến toàn cục chính là toàn bộ chương trình bắt đầu từ vị trí định nghĩa (khai báo) biến cho đến cuối chương trình.
- Tất cả các biến toàn cục sẽ được tự động khi khai báo (nếu người lập trình không khởi động) với giá trị 0 (biến số) và giá trị NULL (biến ký tự).

# Biến cục bộ

- Biến cục bộ (local variable): được định nghĩa trong hàm hoặc trong một khối chương trình. Chỉ có thể truy cập (sử dụng) trong vùng khai báo đó (hàm hoặc khối chương trình đã khai báo biến).
- Các hàm và khối chương trình khác nhau vẫn có thể khai báo cùng một tên biến.
- Khi hàm A gọi hàm B chỉ có các biến trong hàm B mới có tác dụng trong quá trình thực thi hàm B. Các biến cùng tên trong hàm A không có tác dụng trong khi hàm B đang chạy.

# Ví dụ biến toàn cục-biến cục bộ

```
#include<stdio.h>
int n;//bien toan cuc n
void F1(int a)
        a=10;
        int n=a;//bien n va a la cac bien cuc bo
}
void F2(int a)
        n = a;//thay doi noi dung bien toan cuc n

void main()
        int x=2;//bien x cuc bo trong main
        F1(x);
        printf("\nx=%d, n=%d", x, n);//in noi dung x cuc bo va n toan cuc
        F2(x);
        printf("\nx=%d, n=%d", x, n);//in noi dung x cuc bo va n toan cuc
}
```

# Truyền tham số: Truyền giá trị – truyền tham chiếu

- Truyền giá trị
  - Mặc định, tất cả các tham số hàm truyền cho hàm khi gọi là bởi giá trị. Các giá trị truyền vào trong hàm thông qua biến tạm cục bộ của hàm.
  - Tất cả thao tác với biến tham số trong hàm đều chỉ thao tác trên biến cục bộ
  - Bất cứ của sự thay đổi trên biến tham số của hàm không ảnh hưởng tới giá trị biến được truyền cho hàm.

# Truyền tham trị

```
#include<stdio.h>
void swap(int a, int b) //2 tham số a, b truyền tham trị
        int temp = a; //temp: biến cục bộ trong swap
        a = b;
        b = temp;

void main()

        int x=1, y=2;
        swap(x, y);//Mọi thay đổi trong swap không ảnh hưởng tới x, y
        printf("x=%d, y=%d", x, y);//x=1, y=2
```

# Truyền tham biến

- Hàm được phép truy xuất tới địa chỉ của các đối số.
- Hàm có thể thay đổi giá trị các đối số khi thực thi.
- Khi định nghĩa hàm, các tham số sử dụng truyền tham chiếu, được sử dụng biến con tro

```
void swap2(int *a, int *b){
        int temp= *a;
        *a = *b;
        *b = temp;
```

- Khi gọi hàm: Cần truyền vào địa chỉ các đối số (thay vì giá trị)

```
        swap2(&x, &y);
```

# Ví dụ

```
#include<stdio.h>
void swap(int a, int b)
        int temp = a;
        a=b;     b=temp;

void swap2(int *a, int *b){
        int temp= *a;
        *a = *b;
        *b = temp;

void main()
        int x=1, y=2;
        swap(x, y);
        printf("x=%d, y=%d", x, y);//x=1, y=2
        x=1; y=2;
        swap2(&x, &y);
        printf("\nx=%d, y=%d", x, y);//x=1, y=2
}
```

# Truyền tham số là mảng

- Tên mảng là địa chỉ, khi gọi hàm, không cần toán tử & trước biến khi gọi hàm

```
#include<stdio.h>
void nhapMang(int M[])   // void nhapMang(int *M)
{           for(int i=0; i<5; i++)
            {           printf("M[%d]=", i);
                        scanf("%d", &M[i]);
            }
}
void xuatMang(int M[])  // xuatMang(int *M)
{           for(int i=0; i<5; i++)
                        printf("%d ", M[i]);
}
void main()
{
            int M[5];
            nhapMang(M);
            xuatMang(M);
}
```