



THỰC HỌC – THỰC NGHIỆP



Conceive Design Implement Operate

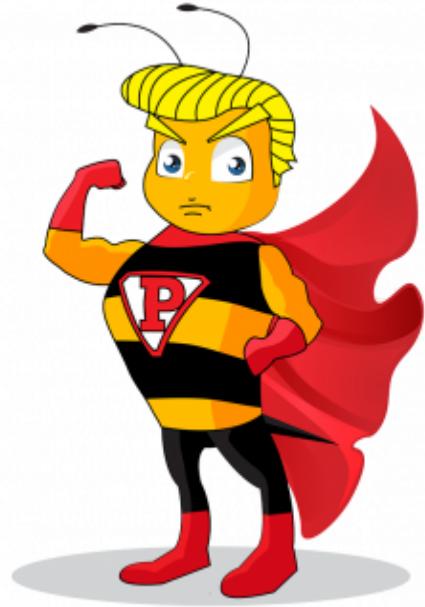
NHẬP MÔN LẬP TRÌNH

ĐẠI CƯƠNG LẬP TRÌNH

BÀI 5: FUNCTION



- Sử dụng thành thạo vòng lặp
- Biết cách khai báo và gọi hàm
- Biết cách sử dụng hàm định nghĩa sẵn
- Biết cách gọi hàm theo giá trị và theo tham chiếu



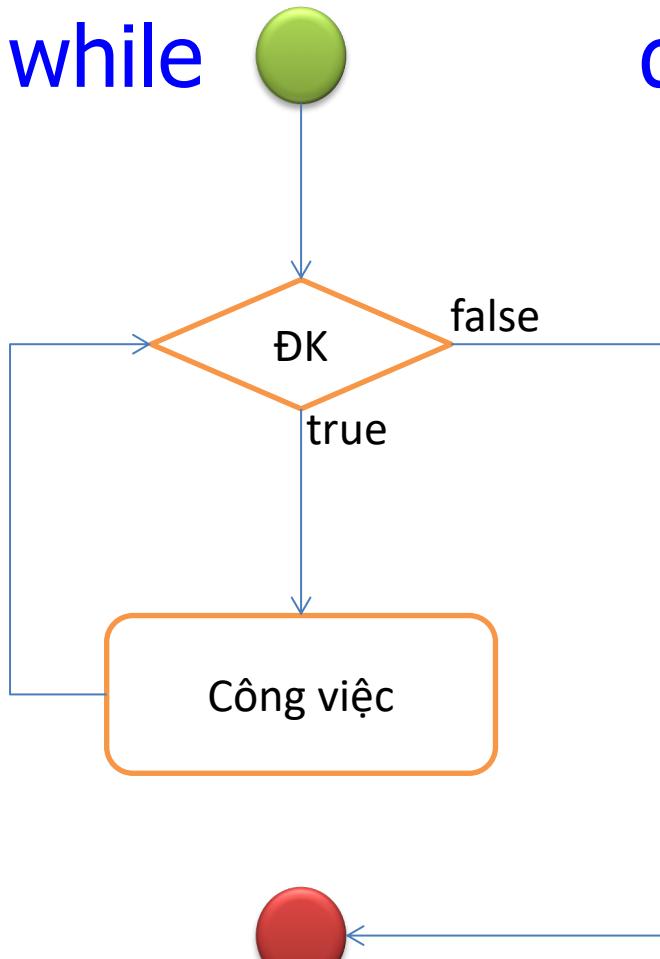
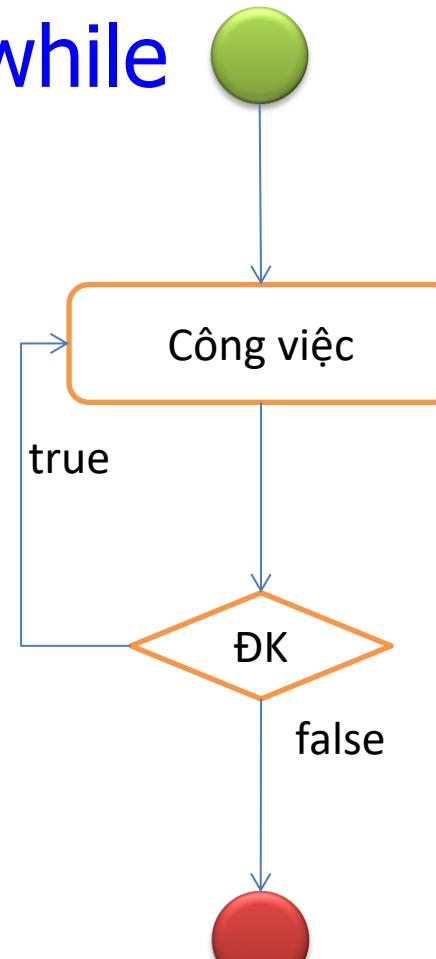
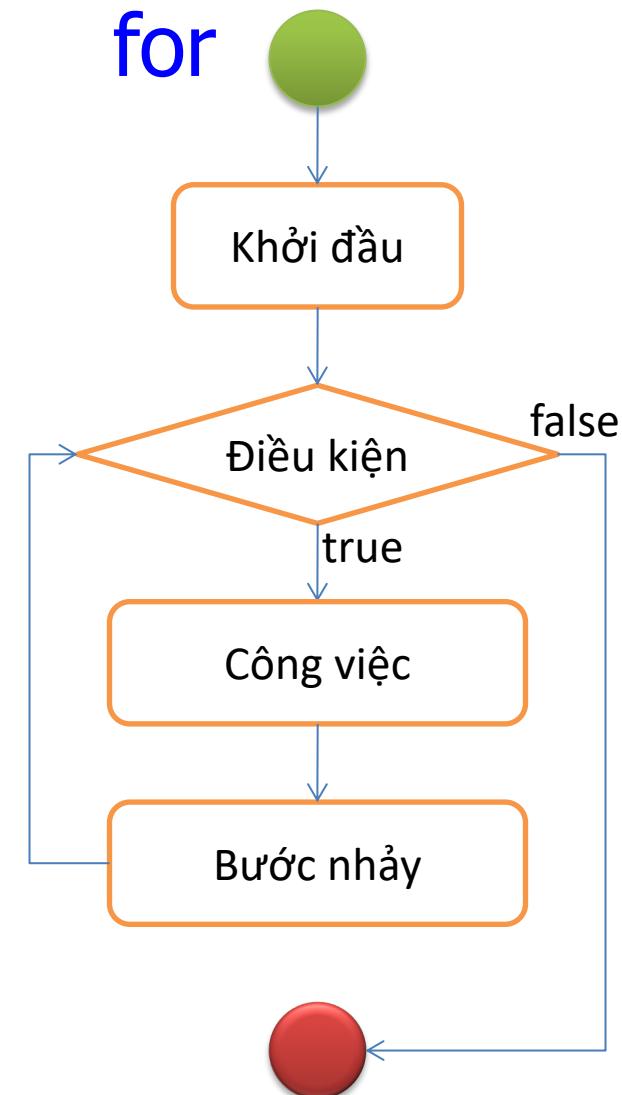
PHẦN 1: ÔN TẬP CÁC VÒNG LẮP & HÀM



① Ôn tập vòng lặp

- ❖ while
- ❖ do...while
- ❖ for
- ❖ break
- ❖ continue
- ❖ goto

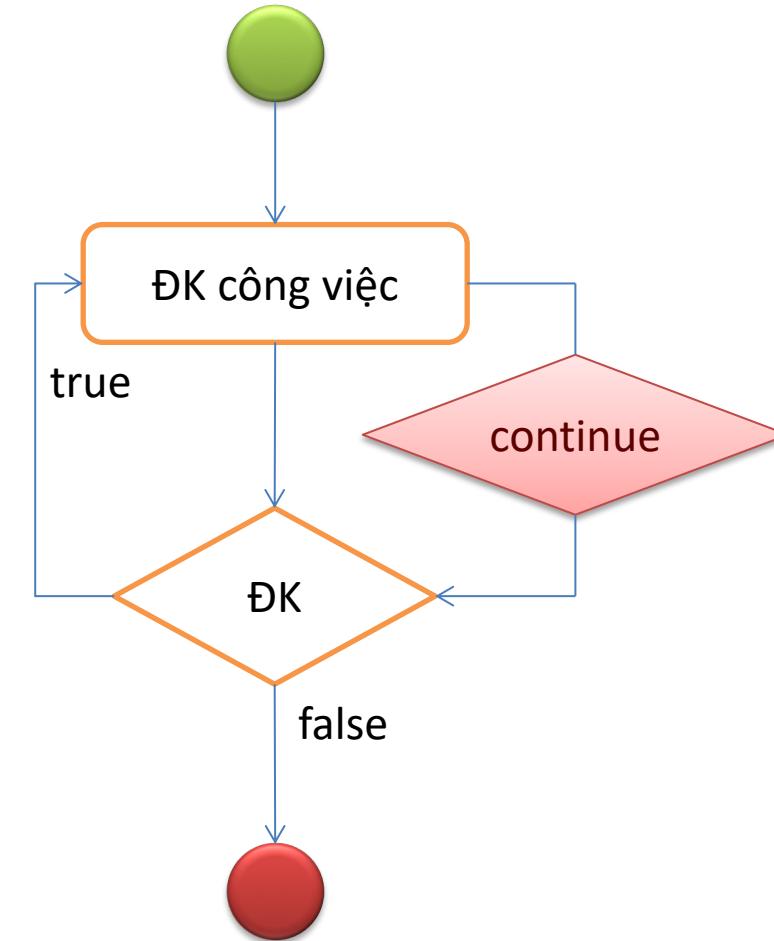
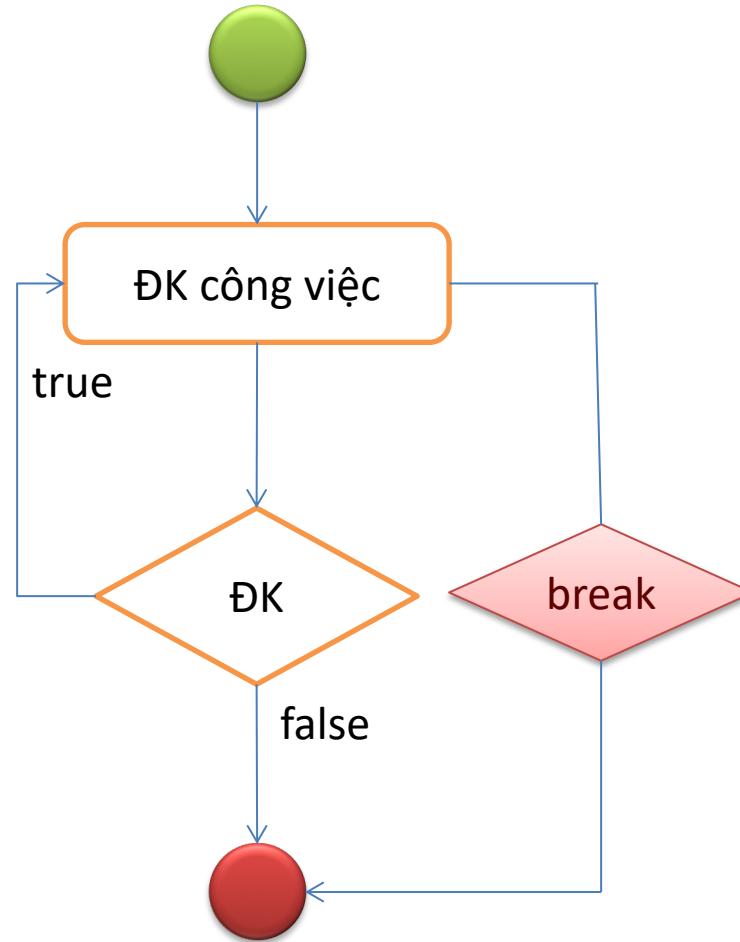
② Hàm

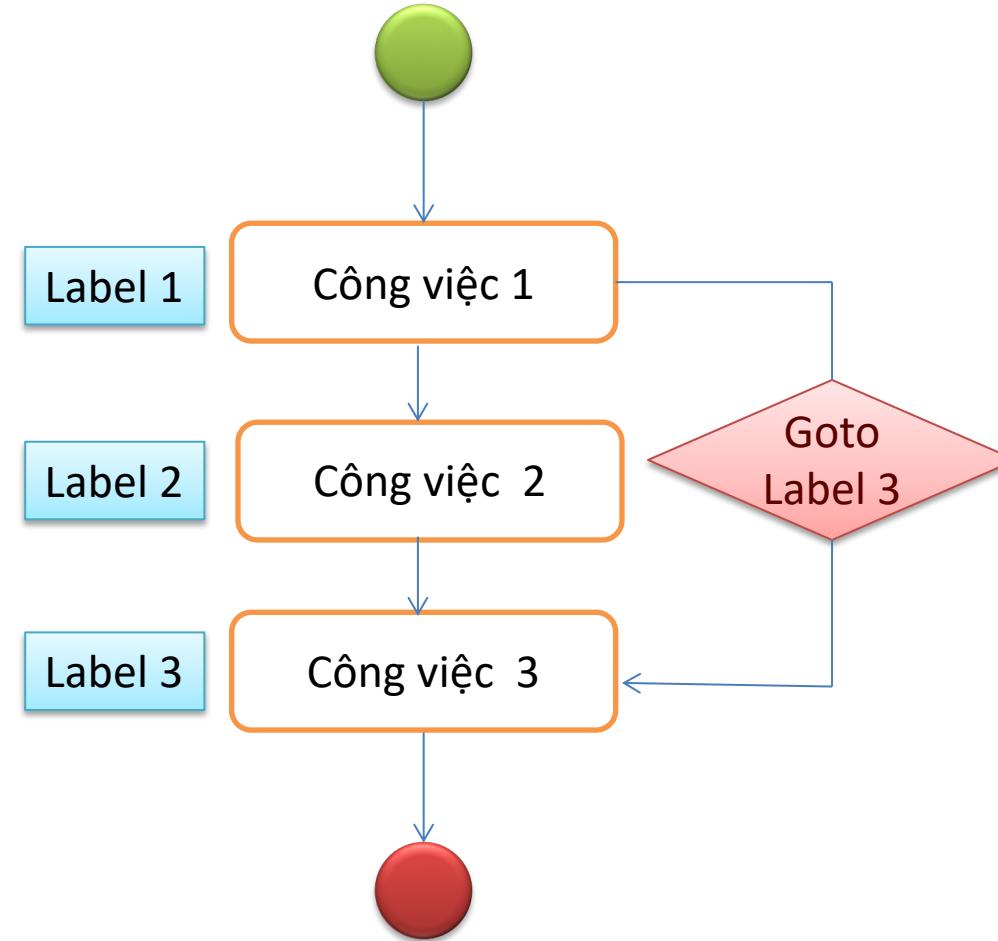
while**do...while****for**

```
int i=1,n=10;           while
    while(i<=n)
    {
        printf("%d\n",i);
        i++;
    }
```

```
int i=1,n=10;           do...while
    do{
        printf("%d\n",i);
        i++;
    }while(i<n);
```

```
int i,n=10;             for
    for(i=1;i<=n;i++)
        printf("%d\n",i);
```







Ôn tập vòng lặp



HÀM

❖ Xét ví dụ sau đây:

```
void main() {  
    int x, y, z;  
    //Nhập số thứ nhất  
    do {  
        printf("x = ");  
        scanf("%d", &x);  
    } while (x<0);  
    //Nhập số thứ hai  
    do {  
        printf("y = ");  
        scanf("%d", &y);  
    } while(y<0);
```

```
//Nhập số thứ ba  
do {  
    printf("z = ");  
    scanf("%d", &z);  
} while (z<0);  
  
int tong = x + y + z;  
printf("Tong 3 so la %d", tong);  
}
```

Trong đoạn chương trình trên phần xử lý **nhập và kiểm tra** số nguyên dương **LẶP LẠI 3 LẦN**, việc này dẫn đến một số vấn đề như: Viết xử lý (code) nhiều lần và khi có sự thay đổi thì phải thay đổi nhiều lần.

- ❖ Hàm là một đoạn chương trình có tên, đầu vào và đầu ra.
- ❖ Hàm có chức năng giải quyết một số vấn đề chuyên biệt cho chương trình chính.
- ❖ Hàm được gọi nhiều lần với các tham số khác nhau.
- ❖ Hàm được sử dụng khi có nhu cầu:
 - Tái sử dụng.
 - Sửa lỗi và cải tiến.
- ❖ Hàm có thể được xây dựng bởi lập trình viên hoặc hàm có sẵn trong ngôn ngữ lập trình. Ví dụ trong lập trình C, chúng ta có thể liệt kê các hàm có sẵn như printf, scanf, pow, sqrt,...

❑ Cú pháp:

<kiểu trả về> <tên hàm>([danh sách tham số])

{

<các câu lệnh>

[return <giá trị>;]

}

❖ Trong đó

➢ <kiểu trả về> : kiểu bất kỳ của C (**char, int, long, float,...**). Nếu không trả về thì là **void**.

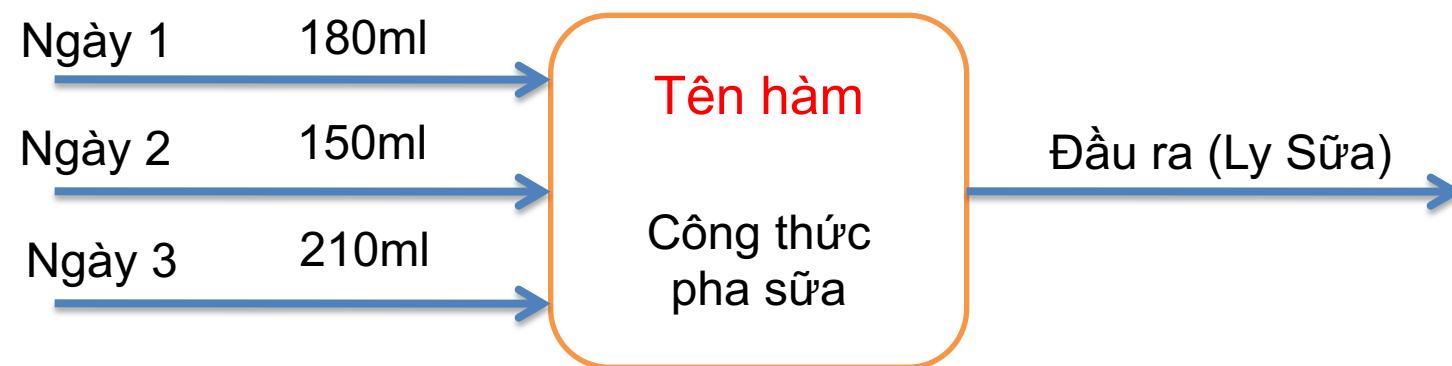
➢ <tên hàm>: theo quy tắc **đặt tên định danh**.

➢ <danh sách tham số> : **tham số hình thức đầu vào** giống khai báo biến, cách nhau bằng dấu ,

➢ <giá trị> : trả về cho hàm qua lệnh **return**.

❑ Ví dụ: Mẹ gọi Tí lại vào bảo: Mẹ viết sẵn công thức pha sữa. Khi mẹ gọi pha sữa, con phải biết pha 1 ly sữa theo dung tích mẹ gọi cho em con uống

- ❖ Pha sữa: hàm
- ❖ Kiểu dữ liệu trả về: ly sữa
- ❖ Dung tích: giá trị truyền vào hàm
- ❖ Mẹ: Developer



☐ Cần xác định các thông tin sau đây:

- ❖ Tên hàm.
- ❖ Hàm sẽ thực hiện công việc gì.
- ❖ Các đầu vào (nếu có).
- ❖ Đầu ra (nếu có).



❑ Hàm tính tổng 2 số nguyên.

- ❖ **Tên hàm:** tinhTong1
- ❖ **Công việc:** tính và xuất tổng 2 số nguyên
- ❖ **Đầu vào:** không có
- ❖ **Đầu ra:** không có

```
void tinhTong1() {  
    int a, b, tong;  
  
    printf("a = ");  
    scanf("%d", &a);  
  
    printf("b = ");  
    scanf("%d", &b);  
  
    tong = a + b;  
    printf("Tong hai so la %d", tong);  
}
```

❑ Hàm nhận tham số đầu vào và không trả về giá trị.

- ❖ **Tên hàm:** tinhTong2
- ❖ **Công việc:** tính và xuất tổng 2 số nguyên
- ❖ **Đầu vào:** hai số nguyên a và b
- ❖ **Đầu ra:** không có

```
void tinhTong2(int a, int b) {  
    int tong;  
    tong = a + b;  
    printf("Tong hai so la %d", tong);  
}
```

☐ Hàm trả về giá trị và không nhận tham số đầu vào.

- ❖ **Tên hàm:** tinhTong3
- ❖ **Công việc:** tính và xuất tổng 2 số nguyên
- ❖ **Đầu vào:** không có
- ❖ **Đầu ra:** một số nguyên có giá trị $a+b$

```
int tinhTong3() {  
    int a,b, tong;  
    printf("Nhập số thứ nhất a = ");  
    scanf("%d", &a);  
    printf("Nhập số thứ hai b = ");  
    scanf("%d", &b); tong = a + b;  
    //Kết thúc hàm và trả về kết quả  
    return tong;  
}
```

❑ Hàm trả về giá trị và nhận tham số đầu vào.

❖ **Tên hàm:** tinhTong4

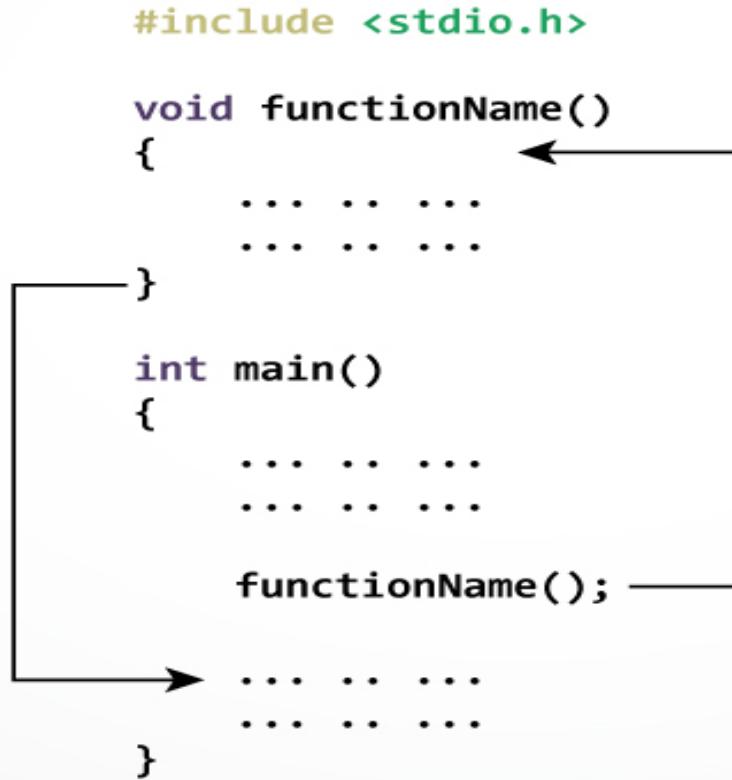
❖ **Công việc:** tính và xuất tổng 2 số nguyên

❖ **Đầu vào:** hai số nguyên a và b

❖ **Đầu ra:** một số nguyên có giá trị a+b

```
int tinhTong4(int a, int b) {  
    int tong;  
    tong = a + b; //Kết thúc hàm và trả về kết quả  
    return tong;  
}
```

- ❑ Gọi hàm thông qua tên hàm và truyền số đầu vào hoặc xử lý kết quả trả về nếu có.
- ❑ Hàm trong lập trình C hoạt động như thế nào? Hình ảnh sau đây mô tả gọi một hàm do người dùng định nghĩa bên trong hàm **main()**.



- ❑ Gọi hàm **tinhTong1** – Hàm **không trả về giá trị** và **không nhận tham số** đầu vào.

```
void main()
{
    tinhTong1();
}
```



```
void tinhTong1() {
    int a, b, tong;

    printf("a = ");
    scanf("%d", &a);

    printf("b = ");
    scanf("%d", &b);

    tong = a + b;
    printf("Tong hai so la %d", tong);
}
```

- ❑ Gọi hàm **tinhTong2** – Hàm **nhận tham số** đầu vào và **không trả về giá trị**.

```
void main()
{
    tinhTong1(10,20);
```



```
void tinhTong2(int a, int b) {
    int tong;
    tong = a + b;
    printf("Tong hai so la %d", tong);
}
```

- ❑ Gọi hàm **tinhTong3** – Hàm trả **về giá trị** và **không nhận tham số** đầu vào.

```
void main()
{
    int ketqua;
    ketqua = tinhTong3();
}
```



```
int tinhTong3() {
    int a,b, tong;
    printf("Nhập số thứ nhất a = ");
    scanf("%d", &a);
    printf("Nhập số thứ hai b = ");
    scanf("%d", &b); tong = a + b;
    //Kết thúc hàm và trả về kết quả
    return tong;
}
```

- ❑ Gọi hàm **tinhTong4** – Hàm trả về giá trị và nhận tham số đầu vào.

```
void main()
{
    int ketqua;
    ketqua = tinhTong4(10,20);
}
```



```
int tinhTong4(int a, int b) {
    int tong;
    tong = a + b; //Kết thúc hàm và
    trả về kết quả
    return tong;
}
```



Xây dựng hàm

Ôn tập vòng lặp

- ❖ while
- ❖ do...while
- ❖ for
- ❖ break
- ❖ continue
- ❖ goto

Hàm





PHẦN 2: HÀM(TT)

Một số lỗi thường gặp khi làm việc với hàm

- ❖ Gọi sai tên hàm
- ❖ Gọi hàm không phù hợp với định nghĩa
- ❖ Hàm có nhận tham số nhưng khi gọi hàm thì không truyền tham số đầu vào
- ❖ Truyền không đúng số lượng tham số
- ❖ Truyền tham số không phù hợp với kiểu dữ liệu lúc định nghĩa hàm

- ❑ Trong C, có rất nhiều hàm đã được định nghĩa sẵn.
- ❑ Có nghĩa là chúng ta không cần phải xây dựng hàm mà chỉ cần biết và gọi hàm ra sử dụng khi cần thiết

Hàm	Diễn giải	Ví dụ
Math.min(a, b)	Lấy số nhỏ nhất của 2 số a và b	x=Math.min(5, 3.5) => x=3.5
Math.max(a, b)	Lấy số lớn nhất của 2 số a và b	x=Math.max(5, 3.5) => x=5
Math.pow(a, n)	Tính a^n (a lũy thừa n)	x=Math.pow(5, 3) => x=75
Math.sqrt(a)	Tính \sqrt{a} (căn bậc 2 của a)	x=Math.sqrt(16) => x=4
Math.abs(a)	Lấy giá trị tuyệt đối của a	x=Math.abs(-5) => x=5
Math.ceil(a)	Lấy số nguyên trên của a	x=Math.ceil(3.5) => x=4
Math.floor(a)	Lấy số nguyên dưới của a	x=Math.floor(3.5) => x=3

☐ Khai báo thư viện math.h

Hàm	Điễn giải	Ví dụ
abs(x)	Lấy giá trị tuyệt đối của x	a = abx(-5) → x = 5
pow(x, y)	Tính x^y (x lũy thừa y)	a=pow(3,2) → a = $3^2 = 9$
Sqrt(x)	Khai căn bậc 2	a= sqrt(16) → a = 4
Ceil(x)	Lấy số nguyên cận trên	a = ceil(14,001) → a = 15
floor(x)	Lấy số nguyên cận dưới	a = ceil(14,991) → a = 14
Sin(x)	Các hàm lượng giác	
Cos(x)	Các hàm lượng giác	

- ❑ Khai báo thư viện graphics.h
- ❑ Khai báo thư viện io.h
- ❑ Khai báo thư viện conio.h

Thư viện graphic.h
initgraph()
line()
putpixel()
getpixel()
setcolor()
...

Thư viện io.h
open()
read()
close()
creat()
filelength()
...

Thư viện conio.h
clrscr()
getch()
cputs()
putch()
clreol()
...



Sử dụng các hàm có sẵn trong C

- ❑ Toàn cục: khai báo trong ngoài tất cả các hàm (kể cả hàm main) và có tác dụng lên toàn bộ chương trình.
- ❑ Cục bộ: khai báo trong hàm hoặc khối {} và chỉ có tác dụng trong bản thân hàm hoặc khối đó (kể cả khối con nó). Biến cục bộ sẽ bị xóa khỏi bộ nhớ khi kết thúc khối khai báo nó.

```
int a;
```

```
int Ham1()
```

```
{  
    int a1;  
}
```

```
int Ham2()
```

```
{  
    int a2;  
    {  
        int a21;  
    }  
}
```

```
void main()
```

```
{  
    int a3;  
}
```

```
int n; //biến toàn cục
int main()
{
    int x, y; //biến cục bộ trong hàm main
    x = 5;
    y = 6;
    {
        int x = 4;
        printf("x (in sub block) = %d\n", x);
        y += 1;
    }
    printf("n = %d\n", n);
    printf("x (int main block) = %d\n", x);
    printf("y = %d\n", y);
}
```



```
x (in sub block) = 4
n = 0
x (in main block) = 5
y = 7
```



Phạm vi của biến trong lập trình

- ❑ Call by value – gọi hàm theo giá trị
- ❑ Call by reference – gọi hàm theo tham chiếu

- ❑ Truyền đối số cho hàm **ở dạng giá trị**.
- ❑ Có thể truyền hằng, biến, biểu thức nhưng **hàm chỉ sẽ nhận giá trị**.
- ❑ Được sử dụng khi **không có nhu cầu thay đổi giá trị của tham số** sau khi thực hiện hàm.
- ❑ Ví dụ:

```
void main() {  
    int a = 100;  
    int b = 200;  
  
    printf("Truoc hoan doi, gia tri cua a : %d\n", a );  
    printf("Truoc hoan doi, gia tri cua b : %d\n", b );  
  
    // gọi hàm hoandoi() để hoán đổi các giá trị  
    hoandoi(a, b);  
  
    printf("Sau hoan doi, gia tri cua a : %d\n", a );  
    printf("Sau hoan doi, gia tri cua b : %d\n", b );  
}
```

```
void hoandoi(int x, int y) {  
    int temp;  
  
    temp = x;  
    x = y;  
    y = temp;  
}
```

```
Truoc hoan doi, gia tri cua a : 100  
Truoc hoan doi, gia tri cua b : 200  
Sau hoan doi, gia tri cua a : 100  
Sau hoan doi, gia tri cua b : 200
```

- ❑ Truyền đối số cho hàm **ở dạng địa chỉ** (con trỏ).
- ❑ Không được truyền giá trị cho tham số này.
- ❑ Được sử dụng khi có **nhu cầu thay đổi giá trị của tham số** sau khi thực hiện hàm.
- ❑ Ví dụ:

```

void main() {
    int a = 100;
    int b = 200;

    printf("Truoc hoan doi, gia tri cua a : %d\n", a );
    printf("Truoc hoan doi, gia tri cua b : %d\n", b );

    // gọi hàm hoandoi() để hoán đổi các giá trị
    hoandoi(&a, &b);

    printf("Sau hoan doi, gia tri cua a : %d\n", a );
    printf("Sau hoan doi, gia tri cua b : %d\n", b );
}

```

```

void hoandoi(int *x, int *y)
{
    int temp;

    temp = *x;
    *x = *y;
    *y = temp;
}

```

Truoc hoan doi, gia tri cua a : 100
 Truoc hoan doi, gia tri cua b : 200
 Sau hoan doi, gia tri cua a : 200
 Sau hoan doi, gia tri cua b : 100



Call by value & Call by reference



CHÈN QUIZ

Ôn tập hàm

Hàm định nghĩa sẵn

Gọi hàm

❖ Value

❖ Reference



