

BỘ CÔNG THƯƠNG
ĐẠI HỌC CÔNG NGHIỆP TP. HỒ CHÍ MINH



Bài giảng

NGÔN NGỮ LẬP TRÌNH C
THE C PROGRAMMING LANGUAGE

Lecturer : Le Ngoc Tran, PhD

Email : lengoctran@iuh.edu.vn

CHƯƠNG 5

HÀM (FUNCTION)

5.1. Đặt vấn đề

❑ 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);
}
```

Trong đoạn chương trình trên phần xử lý **nhập và kiểm tra** số nguyên dương **LẶP LẠI 2 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.

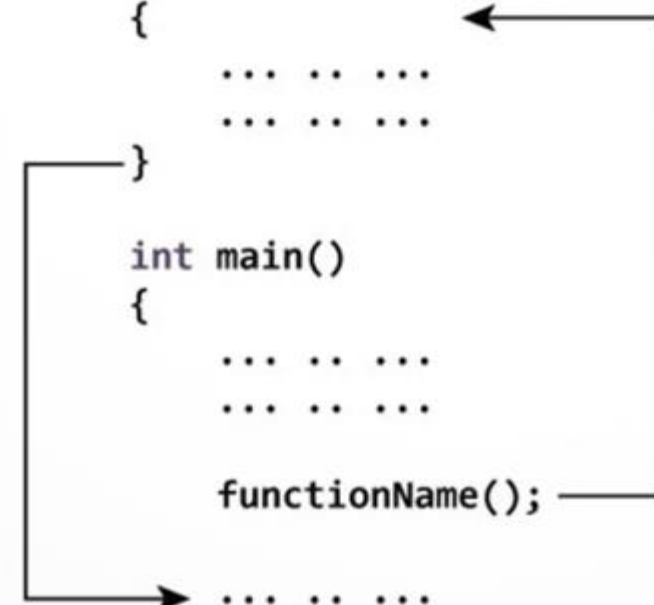
5.2. Khái niệm về Hàm

- ❑ 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.

```
#include <stdio.h>

void functionName()
{
    ... ..
    ... ..
}

int main()
{
    ... ..
    ... ..
    functionName();
    ... ..
    ... ..
}
```



The diagram illustrates the execution flow between the `main` function and a user-defined function `functionName`. An arrow originates from the `functionName();` line within the `main` function's body and points to the opening curly brace of the `functionName` function definition. A second arrow originates from the closing curly brace of the `functionName` function and points back to the line immediately following the `functionName();` call in the `main` function, indicating the return of control.

5.2. Khái niệm về Hàm

❑ Ưu điểm khi sử dụng hàm:

- ❖ Tránh viết lại cùng một mã nhiều lần và có thể tái sử dụng mã
- ❖ Có thể gọi các hàm nhiều lần trong chương trình và từ bất kỳ vị trí nào của chương trình
- ❖ Có thể dễ dàng theo dõi và quản lý khi chương trình C lớn được chia thành nhiều hàm
- ❖ Tuy nhiên, khi gọi hàm thì cần phải tốn thêm chi phí trong chương trình

❑ Có hai loại Hàm:

- ❖ Hàm dựng sẵn
- ❖ Hàm tự định nghĩa (do lập trình viên tạo ra)

5.3. Hàm tự định nghĩa

❑ Cú pháp:

```
<kiểu trả về> <tên hàm> ([danh sách các 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 các 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**.

5.3. Hàm tự định nghĩa

❖ Gọi 1 hàm trong C

❑ Gọi hàm thông qua tên hàm và truyền tham số đầu vào hoặc xử lý kết quả trả về nếu có.

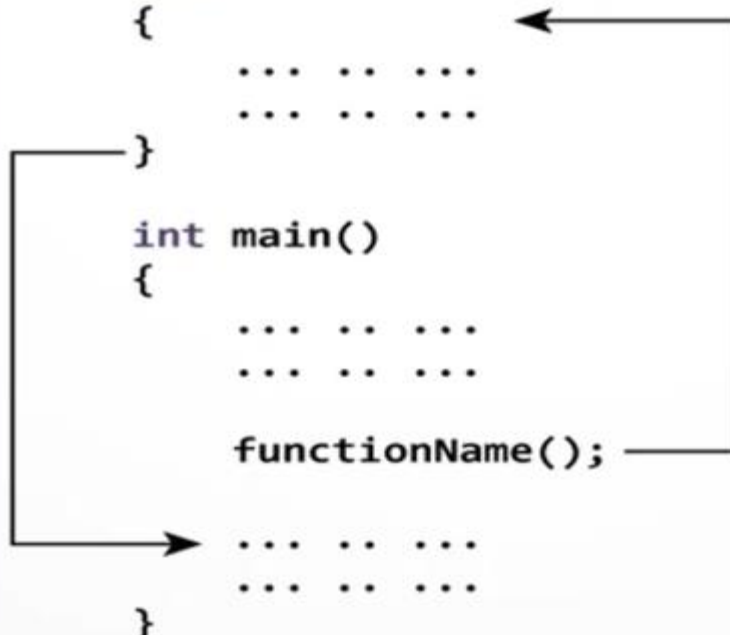
❑ Hàm trong lập trình C hoạt động như thế nào? Hình ảnh sau đây mô tả cách gọi một hàm do người dùng định nghĩa bên trong hàm **main()**

```
#include <stdio.h>

void functionName()
{
    ... ..
    ... ..
}

int main()
{
    ... ..
    ... ..
    functionName();
}

... ..
... ..
```

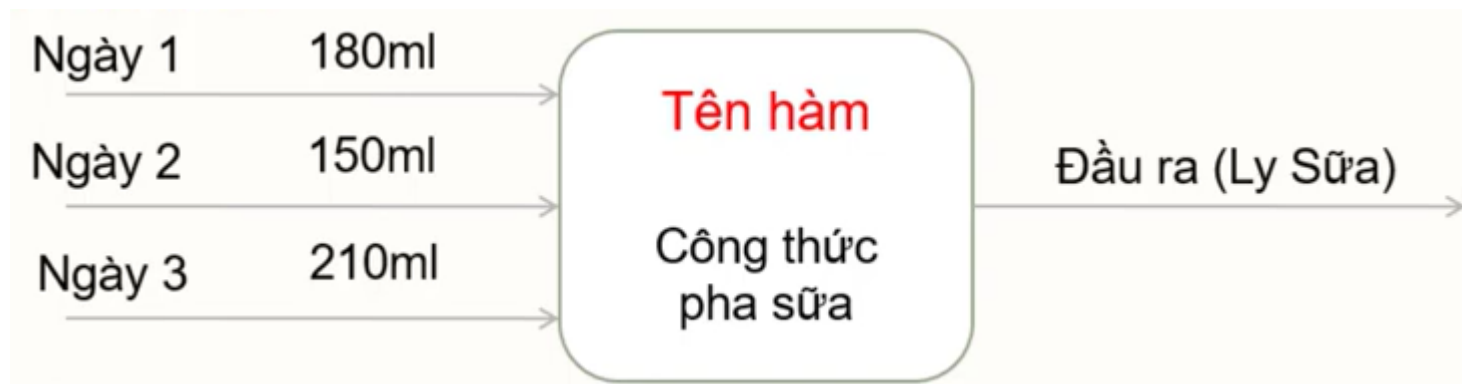


The diagram illustrates the execution flow of a C program. It shows a function definition for `void functionName()` and a `main()` function. Inside `main()`, there is a call to `functionName();`. An arrow originates from this call, points to the opening curly brace of `functionName()`, and then returns to the line following the call in `main()`. This visualizes the process of jumping to the function's code and then returning to the caller.

5.3. Hàm tự định nghĩa

❑ 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



5.3. Hàm tự định nghĩa

❖ Các bước xây dựng hàm

❑ Cần xác định các thông tin sau đây:

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



5.3. Hàm tự định nghĩa

❖ Ví dụ 1:

❑ 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);  
}
```

5.3. Hàm tự định nghĩa

❖ Ví dụ 1:

- ❑ Gọi hàm **tinhTong1** - Hàm không trả về giá trị và không nhận thông 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);  
}
```

Chương trình Demo thực hiện hàm: **tinh-tong.c**

5.3. Hàm tự định nghĩa

❖ Ví dụ 2:

❑ Hàm nhận thông 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);  
}
```

5.3. Hàm tự định nghĩa

❖ Ví dụ 2:

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

```
void main()  
{  
    tinhTong2(10,20);  
}
```



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

Chương trình Demo thực hiện hàm: **tinh-tong2.c**

5.3. Hàm tự định nghĩa

❖ Ví dụ 3:

❑ 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 $a + b$

```
int tinhTong3() {  
    int a,b, tong;  
    printf("Nhap so thu nhat a = ");  
    scanf("%d", &a);  
    printf("Nhap so thu hai b = ");  
    scanf("%d", &b); tong = a + b;  
    //Kết thúc hàm và trả về kết quả  
    return tong;  
}
```

5.3. Hàm tự định nghĩa

❖ Ví dụ 3:

❑ 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("Nhap so thu nhat a = ");  
    scanf("%d", &a);  
    printf("Nhap so thu hai b = ");  
    scanf("%d", &b); tong = a + b;  
    //Kết thúc hàm và trả về kết quả  
    return tong;  
}
```

Chương trình Demo thực hiện hàm: **tinh-tong3.c**

5.3. Hàm tự định nghĩa

❖ Ví dụ 4:

❑ 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;  
}
```


5.3. Hàm tự định nghĩa

❖ Ví dụ 4:

❑ Gọi hàm **tinhTong4** - Hàm trả về giá trị và không 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;  
}
```

Chương trình Demo thực hiện hàm: **tinh-tong4.c**

5.4. Nguyên mẫu hàm (**Function Prototype**)

- ❑ Nguyên mẫu hàm là cung cấp cho trình biên dịch (compiler) biết tên của hàm, kiểu dữ liệu mà hàm trả về.
- ❑ Giúp cho trình biên dịch xác nhận các lời gọi hàm mà chưa cần định nghĩa hàm đó.

```
void tinhTong(int a, int b); // prototype

void main()
{
    ...
}

void tinhTong(int a, int b)
{
    printf("%d cong %d bang %d", x, y, x + y);
}
```

5.4. Nguyên mẫu hàm (**Function Prototype**)

- ❑ Một nguyên mẫu hàm chỉ đơn giản là khai báo một hàm xác định tên, tham số và kiểu trả về của hàm. Nó không chứa thân hàm.
- ❑ Thông thường người ta thường đặt phần tiêu đề hàm/nguyên mẫu hàm (**prototype**) trên hàm main và phần định nghĩa hàm dưới hàm main.

```
void tinhTong(int a, int b); // prototype

void main()
{
    ...
}

void tinhTong(int a, int b)
{
    printf("%d cong %d bang %d", x, y, x + y);
}
```

- ❑ Cú pháp:

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

CT Demo thực hiện nguyên mẫu hàm: **tinh-tongp.c**

- ❖ 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.

- ❑ Tập tin header là tập tin với phần mở rộng (*.h) chứa các khai báo hàm C, định nghĩa kiểu dữ liệu và định nghĩa macro được chia sẻ giữa các tập tin nguồn (*.c) của C.
- ❑ Có hai kiểu header:
 - ❖ Các header do lập trình viên viết
 - ❖ Các header do C cung cấp cũng với trình biên dịch (Compiler)
- ❑ Sử dụng chỉ thị tiền xử lý `#include` để khai báo sử dụng header trong chương trình
 - ❖ Ví dụ: `#include<stdio.h>`

5.5 Lỗi thường gặp với hàm và các tập tin **Header (*.h)**

- ❑ Cú pháp của include:

```
#include <filename.h>  
  
or  
  
#include "filename.h"
```

- ❑ Thực hành đơn giản với các chương trình C/C++ là duy trì tất cả các hằng (constant), macros, biến toàn cục, và nguyên mẫu hàm trong tập tin header và include vào các tập tin nguồn khi cần.

CT Demo thực hiện header-SLIDE4-PRO : [tinh-tong.c](#)

