# CHƯƠNG 2 (TT)

## HÀM NÂNG CAO

# Nội dung

1. Các tham số của hàm main
2. Hàm có đối số mặc định
3. Hàm trả về tham chiếu
4. Hàm nội tuyến (inline)

# Các đối số của chương trình

## Các đối số của chương trình

- Hàm main là hàm nên cũng có tham số.
- Chương trình tự động thực hiện hàm main mà không cần lời gọi hàm.

➔ Làm sao truyền đối số?

➔ Khi thực thi tập tin chương trình (.exe), ta truyền kèm đối số. Tất nhiên, hàm main cũng phải định nghĩa các tham số để có thể nhận các đối số này.

# Các tham số của hàm main

## Các tham số của hàm main

```
void main(int argc, char *argv[])
{
        ...
}
```

- Trong đó
  - argc là số lượng đối số (tính luôn tên tập tin chương trình)
  - argv là mảng chứa các đối số (dạng chuỗi)

# Các tham số của hàm main

- Ví dụ
  - Viết chương trình có tên Cong, nhận 2 đối số x và y và xuất ra giá trị x + y.

```
argv = {"Cong.EXE", "2912", "1706"};
```

```
argc = 3
```

# Các tham số của hàm main

- Ví dụ
  - Viết chương trình có tên Cong, nhận 2 đối số x và y và xuất ra giá trị x + y.

```
#include <stdio.h>
#include <stdlib.h>      // atoi
void main(int argc, char *argv[]) {
      if (argc == 3) {
            int x = atoi(argv[1]);
            int y = atoi(argv[2]);
            printf("%d + %d = %d", x, y, x+y);
      }
      else
            printf("Sai! VD: Cong 2912 1706");
}
```

# Các tham số của hàm main

- Ví dụ
  - Viết chương trình có tên test nhận dữ liệu từ tập tin input.txt, xử lý và xuất kết quả ra tập tin output.txt.

```
argv = {"test", "input.txt", "output.txt"};
```

```
argc = 3
```

# Các tham số của hàm main

- Ví dụ
  - Viết chương trình có tên test nhận dữ liệu từ tập tin input.txt, xử lý và xuất kết quả ra tập tin output.txt.

```
#include <stdio.h>
void main(int argc, char *argv[]) {
	if (argc == 3) {
		// Nhập dữ liệu từ tập tin argv[1]
		// Xử lý
		// Xuất kết quả ra tập tin argv[2]
	}
	else
		printf("Sai! VD: test in.txt out.txt");
}
```

# Hàm có đối số mặc định

- Ví dụ
  - Viết hàm Tong để tính tổng 4 số x, y, z, t

```
int Tong(int x, int y, int z, int t)
{
        return x + y + z + t;
}
```

  - Tính tổng 4 số 2912, 1706, 1506, 1904

```
Tong(2912, 1706, 1506, 1904);
```

  - Nếu chỉ muốn tính tổng 2 số 2912, 1706

```
Tong(2912, 1706, 0, 0);     // z = 0, t = 0
```

# Hàm có đối số mặc định

- Khái niệm
  - Hàm có đối số mặc định là hàm có một hay nhiều tham số hình thức được gán giá trị.
  - Tham số này nhận giá trị mặc định đó nếu không có đối số truyền vào cho tham số đó.
  - Phải được dồn về tận cùng bên phải.
- Ví dụ

```
int Tong(int x, int y, int z = 0 , int t = 0)
{
        return x + y + z + t;
}
```

# Hàm có đối số mặc định

- Lưu ý
  - Muốn truyền đối số khác thay cho đối số mặc định, phải truyền đối số thay cho các đối số mặc định trước nó.

```
int Tong(int x, int y = 0, int z = 0);

    Tong(1, 5);

    Tong(1, 0, 5);
```

# Hàm có đối số mặc định

- Ví dụ
  - In thông tin SV trong lớp gồm: họ tên, phái, lớp, năm sinh

```
void XuatThongTin(char *hoten, char phai = 0,
char *lop = "TH07", int namsinh = 1989)
{
        puts(hoten);
        printf(phai == 0? "Nam\n" : "Nu\n");
        puts(lop);
        printf("%d", namsinh);
}
```

# Hàm có đối số mặc định

- Ví dụ
  - In thông tin SV trong lớp gồm: họ tên, phái, lớp, năm sinh

```
void main()
{
        XuatThongTin("Nguyen Van A");
        XuatThongTin("Tran Thi B", 1);
        XuatThongTin("Hoang Van C", 0, "TH00");
        XuatThongTin("Le D", 1, "TH07", 1988);
}
```

# Hàm có đối số mặc định

- Nhận xét
  - x = a thường xuyên xảy ra thì nên chuyển x thành tham số có đối số mặc định là a.

    Ví dụ, hầu hết phai = 0 (nam), lop = "TH07" và namsinh = 1989.
  - x = a và y = b thường xuyên xảy ra nhưng y = b thường xuyên hơn thì nên đặt tham số mặc định x trước y.

    Ví dụ, lop = "TH07" xảy ra nhiều hơn phai = 0 nên đặt lop sau phai.

# Chỉ thị tiền xử lý #define

- Định nghĩa hằng ký hiệu
  - Chỉ thị #define <name> <value>
  - Mọi chỗ xuất hiện <name> trong chương trình nguồn được thay thế bằng <value> để tạo ra chương trình tiền xử lý.
  - Ví dụ
    - #define MAX 1000
    - #define PI 3.14
    - #define message "Hello World\n"

# Chỉ thị tiền xử lý #define

- Định nghĩa các macro (lệnh gộp - lệnh tắt)
  - #define <name>(<param-list>) <expression>
  - Mọi chỗ xuất hiện của <name> với lượng tham số đưa vào phù hợp sẽ được thay thế bởi <expression> (tham số được thay thế tương ứng)
  - Ví dụ
    - #define showmsg(msg) printf(msg)
    - ➔showmsg("Hello"); ⇔ printf("Hello");

# Hàm nội tuyến (inline)

- Ví dụ
  - Xét 2 cách sau

```
#define PI 3.14159
float addPi(float s)
{
	return s + PI;
}

void main()
{
	float s = 0;
	for (int i = 1; i<=100000; i++)
		s = s + PI;		// Cách 1 (0.7s)
		s = addPi(s);		// Cách 2 (1.4s)
}
```

# Hàm nội tuyến (inline)

- Nhận xét
  - Sử dụng hàm giúp chương trình dễ hiểu nhưng lại tốn chi phí cho lời gọi hàm.
- Khắc phục
  - Sử dụng hàm nội tuyến (inline) bằng cách thêm từ khóa inline trước prototype của hàm.

    ➔inline float addPi(float s) {return s + PI;}
- Khái niệm
  - Sao chép thân hàm đến bất cứ nào nào hàm được gọi ➔ kết quả giống hệt cách 1.

# Hàm nội tuyến (inline)

- Lưu ý
  - Giảm thời gian thực hiện hàm (gọi và kết thúc).
  - Giảm không gian bộ nhớ do các hàm con chiếm dụng khi hàm được gọi.
  - Không cho phép các hàm nội tuyến đệ quy.
  - Phần lớn không cho phép thực hiện nội tuyến các hàm sử dụng vòng lặp while.
  - Chỉ inline các hàm nhỏ, inline các hàm lớn sẽ gây phản tác dụng (bộ nhớ cho hàm inline chiếm giữ sẽ lâu giải phóng hơn).

# Hàm trả về tham chiếu

- Ví dụ
  - Hàm chỉ trả về giá trị. Ví dụ, x = f();
  - Vậy, g() = x hợp lệ hay không?
  - ➔ Hợp lệ khi g(x) trả về tham chiếu đến một biến (C++)
- Cú pháp

```
<kiểu trả về> &<tên hàm>([<ds tham số>])
{
        return <biến>;
}
```

# Hàm trả về tham chiếu

- Ví dụ

```
#include <stdio.h>

int x;

int &getx()
{
        return x;
}

void main()
{
        getx() = 5; // ⇔ x = 5
}
```

# Hàm trả về tham chiếu

- Ứng dụng
  - Chỉ số của mảng trong C/C++ bắt từ 0

    ➔Không quen thuộc lắm.
  - Viết hàm để khi muốn truy cập đến phần tử thứ i của mảng a ta sử dụng V(i) thay vì a[i-1]

```
int a[100];
int &V(int i)
{
      return a[i-1];
}
…
V(1) = 2912; // ⇔ a[0] = 2912;
```

# Hàm trả về tham chiếu

- Chú ý
  - Trong trường hợp sau, biến x phải là biến toàn cục ➔ không nên sử dụng!

```
int x; // biến toàn cục

int &getx()
{
	return x;
}

void main()
{
	getx() = 2912;
}
```

# Hàm trả về tham chiếu

- Chú ý
  - Nếu không muốn sử dụng biến toàn cục, phải truyền x ở dạng tham chiếu.

```
int &getx(int x) {  // SAI! x là tham trị ➔ bản sao
      return x;
}

int &getx() {
      int x;        // SAI! x là biến cục bộ
      return x;
}

int &getx(int &x) { // ĐÚNG! x là tham chiếu
      return x;
}
```

# Hàm trả về tham chiếu

- Ví dụ

```
#include <stdio.h>

int &V(int a[], int i)
{
	return a[i-1];
}

void main()
{
	int a[100];
	for (int i = 1; i <= 100; i++)
		V(a, i) = 0;
}
```

# Bài tập

- Bài 1: Viết chương trình có tên TinhToan sao cho khi gõ: TinhToan 2912 – 1706 sẽ xuất ra màn hình 1206 (có thể thay bằng +, *, /)
- Bài 2: Viết chương trình quản lý thông tin sinh viên (sử dụng hàm có đối số mặc định), bao gồm nhập, sắp xếp tăng dần theo tên và xuất danh sách sinh viên.
- Bài 3: Chuyển các hàm nhỏ hàm nội tuyến.
- Bài 4: Nhập mảng, sắp xếp mảng tăng dần và xuất mảng sử dụng hàm trả về tham chiếu.