**Function:**

1. Công dụng: Hàm có khả năng tái sử dụng, làm cho chương trình gọn gàng, khó nhầm lẫn.
2. Khi khai báo các kiểu dữ liệu giá trị (short, int, long, float, double, char…) thì ta cần sử dụng lệnh return ở cuối dòng, dòng return sẽ trả lại “giá trị” cho hàm, vì thế các hàm có sử dụng lệnh return ở cuối đều có thể xem như là “giá trị”(số).
3. Hàm Void không được sử dụng return vì mục đích hàm này không nằm ở việc trả lại giá trị (có thể sử dụng để setter, cout…).
4. Mọi dòng lệnh bên dưới return sẽ bị bỏ qua.
5. Không thể trả về nhiều giá trị cho hàm return.(có thể trả về nhiều giá trị khi học tham chiếu & con trỏ).

**Syntax:**

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

{

<Các câu lệnh>

<Return <giá trị>>;

}

Trong đó:

+) <Kiểu trả về> : Kiểu bất kỳ của C++(bool, char, int, double…..). Nếu không trả về là void (không có lệnh return).

+) <Tên hàm>: Theo quy tắc đặt tên định danh.

+) <Danh sách tham số>: Tham số có hình thức đầu vào giống khai báo biến, cách nhau dấu “,”, (có thể không có).

+) <giá trị>: Trả về hàm qua lệnh return.(không có với void).

**Pass value:**

Parameter(Tham số): Là biến được khai báo trên lời gọi hàm.

Argument(Đối số): Là giá trị được truyền vào hàm qua lời gọi hàm.

Notes:

- Đối số có thể là biến, hằng, biểu thức, structs, classes hoặt enumerators.

- Đối số sẽ không bị thay đổi qua lời gọi hàm.

Nhược điểm:

* Gây tốn vùng nhớ vì truyền giá trị có bản chất là copy giá trị vào tham số.
* Giảm hiệu suất với các kiểu dữ liệu phức tạp. Vd: Structs, classes, array..
* Hàm chỉ có thể trả lại 1 câu lệnh duy nhất với lệnh return.

Nên sử dụng khi:

* Đối số là các kiểu dữ liệu cơ bản : short, int, float, double..
* Không có nhu cầu thay đổi dữ liệu của đối số.

Không nên sử dụng khi:

* Các đối số là mảng (array), Kiểu cấu trúc(Structs), Lớp(classes), nói chung là kiểu phức tạp.

**Syntax:**

Parameters(tham số): int example(int x, int y, int z)

X,y,z là các tham số được khai báo cách dấu “,”.

Argument(Đối số): example(69,11,12);

Khi gọi hàm example thì thêm giá trị 69,11,12 (ngang thứ tự của tham số và bằng số lượng tham số), các giá trị đấy gọi là đối số

**Pass address**

1. Syntax:

Void function(int \*ptr);

Function(address);

* Tham số hàm phải là con trỏ.
* Giá trị truyền vào tham số là address.

1. Notes:

* Ta có thể làm cho giá trị ngoài hàm không thay đổi bằng cách sử dụng const.
* Khi ta truyền địa chỉ cho tham số, tham số sẽ sao chép địa chỉ đấy, lấy địa chỉ sao chép đấy để truy cập vùng nhớ. Nhưng vì địa chỉ ấy là địa chỉ sao chép nên khi ta thay thế địa chỉ ấy, thì không thể thay thế địa chỉ ở bên ngoài được.
* Muốn thay đổi địa chỉ bên ngoài thì ta có thể sử dụng tham chiếu. function(int \*&ptr)
* Nên dùng với mảng, còn mấy kiểu khác ưu tiên tham chiếu hơn.

**Default arguments**

1. Syntax:

Function(int a=0);

* Khai báo trong tham số hàm.
* Khai báo như thường nhưng được gán thêm 1 value mặc định.
* Chỉ có thể sử dụng ở các tham số cuối hàm liên tiếp:

VD: function(int a=2, int b=3, int c=1) (true)

Function(int a=1, b, c=4) (false) tham số a không liên tiếp với tham số cuối là c.

1. Function:

* Khi dùng thì nếu gọi hàm mà không truyền đối số cho đối số mặc định thì nó sẽ tự lấy giá trị khởi tạo để sử dụng.

1. Notes:

* Đối số mặc định không thể dùng để phân biệt trong nạp chồng hàm:

VD: int add(int a, int b=4);

Int add(int a);

Add(5); => Dễ Lỗi bởi ctr sẽ không phân biệt được cái nào có đối số mặt định khi không truyền đối số.

**Function pointer**

1. Syntax:

<datatype>(\*tencontro)(<danh sach tham so>);

1. Gán địa chỉ cho con trỏ hàm:

<controham>=<tenham>; (không (), không tham số)

* Cùng kiểu.
* Nhớ là gắn với địa chỉ.
* Cùng cấu trúc (kiểu, tham số)

1. Gọi hàm bằng con trỏ:

C1: (\*ptrfunction)(tham số);

C2: ptrfunction(tham số); (giống gọi hàm bth)

* Không sử dụng tham số mặt định ở con trỏ hàm.

1. Truyền hàm dưới dạng đối số:

Bool true(int a, int b)

{ return a<b}

Bool false(int a, int b)

{ return a>b}

Void test(int a, int b, bool(\*testcontroham)(int , int ));

Gọi: test(a,b, true(or false));

* Có thể sử dụng đối số mặc định để khai báo cho tham số con trỏ hàm.

Vd: bool(\*testcontroham)(int, int)=true

1. Std::function (#include<function>)

Function<int(tham số)> fncptrA= funcA;

* Việc sử dụng std::function cũng giống như con trỏ hàm, khác ở chỗ khai báo thôi.

1. Có thể sử dụng auto để xác định con trỏ hàm (tự xác định được thì k cần).