HƯỚNG DẪN THỰC HÀNH

HÀM TRONG C/C++

1 Ý nghĩa

- Hàm gồm tập các dòng lệnh thực hiện một ***công việc độc lập***
- Hàm cho phép cấu trúc chương trình, sử dụng lại source code.

2 Cú pháp

```
Kieu_tra_ve  Ten_ham(kieu_1 tham_so_1,kieu_2 tham_so_2,…){
     <Lệnh 1>;
     <Lệnh 2>;
}
```

Trong đó,

- ***Kieu_tra_ve***: tên của kiểu giá trị trả về, có thể là `int, float, double, char`.
  - Trong trường hợp hàm không trả về giá trị thì `Kieu_tra_ve` là `void`
  - Để trả về giá trị của hàm, dùng lệnh return.

- ***Ten_ham***: tên của hàm, dùng để gọi hàm về sau. Sinh viên có thể đặt tên tùy ý. Lưu ý là:
  - Tên hàm nên thể hiện được tác dụng của hàm.
  - Tên hàm có thể chứa chữ cái, chữ số và dấu gạch dưới "_" và không bắt đầu bằng chữ số

- ***Tham_so_1, tham_so_2***: các giá trị đầu vào của hàm.
- ***Kieu_1, kieu_2***: kiểu dữ liệu của tham số tương ứng.

Chú ý rằng, các hàm có thể có tên trùng nhau, nhưng phải khác kiểu tham số truyền vào.

Ví dụ 1: hàm không trả về giá trị, không có tham số truyền vào

```
#include <stdio.h>

// Hàm in 5 câu "xin chao"
void InXinChao(){
      for (int i = 0; i < 5; i++)
      {
            printf("xin chao!\n");
```

```
	}
}

void main()
{
	// Gọi hàm
	InXinChao();
}
```

Ví dụ 2: Hàm trả về giá trị, không có tham số truyền vào.

```
#include <stdio.h>

// Hàm tính tổng của các số tự nhiên từ 1 đến 5
int TinhTong(){

	int sum = 0;
	for (int i = 1; i <= 5; i++)
	{
		sum += i;
	}
	return sum; // trả về biến có kiểu là int
}

void main()
{
	// Gọi hàm
	int s = TinhTong();
	printf("Tong cua 5 so tu nhien dau tien la %d", s);
}
```

Ví dụ 3: Hàm trả về giá trị, có một tham số truyền vào

```
#include <stdio.h>

// Hàm tính tổng của n số tự nhiên đầu tiên 1 -> n
int TinhTong(int n){

	int sum = 0;
	for (int i = 1; i <= n; i++)
	{
		sum += i;
	}
	return sum; // trả về biến có kiểu là int
}

void main()
{
	int n = 0;
	printf("Nhap gia tri cua n");
	scanf("%d", &n);
```

```
	// Gọi hàm theo tên và truyền tham số. Lưu ý không ghi kiểu dữ liệu nữa
	int s = TinhTong(n);
	printf("Tong cua %d so tu nhien dau tien la %d", n, s);
}
```

Ví dụ 4: Hàm có giá trị trả về, có nhiều hơn một tham số truyền vào

```
#include <stdio.h>

// Hàm tính tổng 2 số thực a và b
float TinhTong(float a, float b){
	int c = a + b;
	return c;
}

void main()
{
	float a = 0;
	float b = 0;
	printf("Nhap gia tri cua a");
	scanf("%f", &a);

	printf("Nhap gia tri cua b");
	scanf("%f", &b);

	// Gọi hàm theo tên và truyền tham số
	float s = TinhTong(a, b);
	printf("%f + %f = %f", a, b, s);
}
```

3 Truyền tham trị, tham chiếu

- Trong các ví dụ ở phần 2 thì các biến sẽ truyền giá trị của mình cho các tham số hàm khi hàm tương ứng được gọi. Trường hợp này gọi là ***truyền tham trị***.

- ***Truyền tham trị*** sẽ không làm thay đổi giá trị của biến truyền vào dù cho thay đổi giá trị của tham số tương ứng trong hàm.

- ***Ví dụ truyền tham trị***: Khi gọi hàm `NhanDoi(a)` thì giá trị của biến `a`, là 4, sẽ được truyền vào trong hàm. Kết thúc hàm, giá trị biến a vẫn là 4 (mặc dù trong hàm thực hiện nhân đôi `a`)

```
#include <stdio.h>
// Nhân đôi giá trị của a
void NhanDoi(int a){

	a = a * 2;
}

void main()
{
	int a = 4;

	// Gọi hàm theo tên và truyền tham số
	NhanDoi(a);
	printf("Gia tri cua a sau khi goi ham %d", a);
}
```

- C/C++ còn hỗ trợ ***truyền tham biến***, cho phép thay đổi giá trị của tham số truyền vào bên trong hàm.
- Để ***truyền tham biến***, thêm dấu ***"&"*** vào trước tên của tham số hàm.

- ***Ví dụ truyền tham biến***: Hàm `NhanDoi` hỗ trợ truyền tham biến (***dấu &)*** nên những thay đổi lên `a` vẫn còn hiệu lực sau khi hàm `NhanDoi(a)` được gọi. Nghĩa là, `a` sẽ bị thay đổi giá trị thành 8.

```
#include <stdio.h>

// Nhân đôi giá trị của a
float NhanDoi(int &a){
	a = a * 2;
}

void main()
{
	int a = 4;

	// Gọi hàm theo tên và truyền tham số
	NhanDoi(a);
	printf("Gia tri cua a sau khi goi ham %d", a);
}
```

4 Khai báo hàm

Trong C/C++, chỉ có thể gọi hàm khi hàm đó đã được khai báo. Có 2 cách khai báo hàm:

Cách 1: Hàm cần được viết (định nghĩa) trước khi được gọi.

Chẳng hạn, trong ví dụ phần 3 thì hàm `void NhanDoi(int a)` được viết ở đầu tập tin, trước khi nó được gọi trong hàm `void main()`

Cách 2: khai báo prototype của hàm trước, còn phần định nghĩa hàm có thể viết sau khi được gọi

Dưới đây là một ví dụ khai báo hàm.

```
#include <stdio.h>

// Khai báo hàm, chỉ cần ghi tên hàm, kiểu trả về và tham số,
// Khai báo kết thúc bằng dấu ";"
void NhanDoi(int a);
int TinhTong(int a, int b);

void main()
{
	int a = 4;
	int b = 5;

	// Gọi hàm theo tên và truyền tham số
	NhanDoi(a);
	printf("Gia tri cua a sau khi goi ham %d", a);

	int x = TinhTong(a, b);
	printf("a + b = %d", x);
}


// Định nghĩa các hàm đã khai báo,
// và có thể đặt sau khi gọi ở hàm main

// Nhân đôi giá trị của a
void NhanDoi(int a){

	a = a * 2;
}

// Hàm tính tổng 2 số nguyên a và b
int TinhTong(int a, int b){
	int c = a + b;
	return c;
}
```