Hàm trong C++

Hàm là một khối lệnh được thực hiện khi nó được gọi từ một điểm khác của chương trình. Dạng thức của nó như sau:

```
type name ( argument1, argument2, ...) {
  statement
}
```

trong đó:

- `type` là kiểu dữ liệu được trả về của hàm
- `name` là tên gọi của hàm.
- `arguments` là các tham số (có nhiều bao nhiêu cũng được tuỳ theo nhu cầu). Một tham số bao gồm tên kiểu dữ liệu sau đó là tên của tham số giống như khi khai báo biến (ví dụ int x) và đóng vai trò bên trong hàm như bất kì biến nào khác. Chúng dùng để truyền tham số cho hàm khi nó được gọi. Các tham số khác nhau được ngăn cách bởi các dấu phẩy.
- `statement` là thân của hàm. Nó có thể là một lệnh đơn hay một khối lệnh.

Các nội dung cơ bản về Hàm trong ngôn ngữ lập trình C++:

- Hàm không kiểu - Void trong C++
- Hàm - Tham số giá trị và tham số biến trong C++
- Hàm - Giá trị mặc định của tham số trong C++
- Hàm nạp chồng trong C++
- Hàm inline trong C++
- Hàm nguyên mẫu trong C++

Dưới đây là ví dụ đầu tiên về hàm:

```
// function example
#include <iostream.h>

int addition (int a, int b)
{
  int r;
  r=a+b;
  return (r);
```

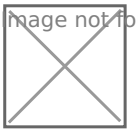
```
}

int main ()
{
  int z;
  z = addition (5,3);
  cout << "The result is " << z;
  return 0;
}
```

Kết quả:

```
The result is 8
```

Để có thể hiểu được đoạn mã này, trước hết hãy nhớ lại những điều đã nói ở bài đầu tiên: một chương trình C++ luôn bắt đầu thực hiện từ hàm **`main`**. Vì vậy chúng ta bắt đầu từ đây.

Chúng ta có thể thấy hàm **`main`** bắt đầu bằng việc khai báo biến z kiểu **`int`**. Ngay sau đó là một lời gọi tới hàm **`addition`**. Nếu để ý chúng ta sẽ thấy sự tương tự giữa cấu trúc của lời gọi hàm với khai báo của hàm:



Các tham số có vai trò thật rõ ràng. Bên trong hàm **main** chúng ta gọi hàm **addition** và truyền hai giá trị: **5** và **3** tương ứng với hai tham số **int a** và **int b** được khai báo cho hàm **addition**.

Vào thời điểm hàm được gọi từ **main**, quyền điều khiển được chuyển sang cho hàm **addition**. Giá trị của c hai tham số (**5** và **3**) được copy sang hai biến cục bộ **int a** và **int b** bên trong hàm.

Dòng lệnh sau:

```
return (r);
```

kết thúc hàm **addition**, và trả lại quyền điều khiển cho hàm nào đã gọi nó (**main**) và tiếp tục chương trình ở cái điểm mà nó bị ngắt bởi lời gọi đến **addition**. Nhưng thêm vào đó, giá trị được dùng với lệnh **return** (**r**) chính là giá trị được trả về của hàm.

Giá trị trả về bởi một hàm chính là giá trị của hàm khi nó được tính toán. Vì vậy biến **z** sẽ có có giá trị được trả về bởi **addition (5, 3)**, đó là **8**.

Đây là một ví dụ khác về hàm:

```
// function example
#include <iostream.h>

int subtraction (int a, int b)
{
  int r;
  r=a-b;
  return (r);
}

int main ()
{
  int x=5, y=3, z;
  z = subtraction (7,2);
  cout << "The first result is " << z << '\n';
  cout << "The second result is " << subtraction (7,2) << '\n';
  cout << "The third result is " << subtraction (x,y) << '\n';
  z= 4 + subtraction (x,y);
  cout << "The fourth result is " << z << '\n';
  return 0;
}
```

Trong trường hợp này chúng ta tạo ra hàm **`subtraction`**. Chức năng của hàm này là lấy hiệu của hai tham số rồi trả về kết quả.

Tuy nhiên, nếu phân tích hàm **main** các bạn sẽ thấy chương trình đã vài lần gọi đến hàm **`subtraction`**. Tôi đã sử dụng vài cách gọi khác nhau để các bạn thấy các cách khác nhau mà một hàm có thể được gọi.

Để có hiểu cặn kẽ ví dụ này bạn cần nhớ rằng một lời gọi đến một hàm có thể hoàn toàn được thay thế bởi giá trị của nó. Ví dụ trong lệnh gọi hàm đầu tiên :

```
z = subtraction (7,2);
cout << "The first result is " << z;
```

Nếu chúng ta thay lời gọi hàm bằng giá trị của nó (đó là **5**), chúng ta sẽ có:

```
z = 5;
cout << "The first result is " << z;
```

Tương tự như vậy

```
cout << "The second result is " << subtraction (7,2);
```

cũng cho kết quả giống như hai dòng lệnh trên nhưng trong trường hợp này chúng ta gọi hàm subtraction trực tiếp như là một tham số của cout. Chúng ta cũng có thể viết:

```
cout << "The second result is " << 5;
```

vì **5** là kết quả của **subtraction (7,2)**.

Còn với lệnh:

```
cout << "The third result is " << subtraction (x,y);
```

Điều mới mẻ duy nhất ở đây là các tham số của **subtraction** là các biến thay vì các hằng. Điều này là hoàn toàn hợp lệ. Trong trường hợp này giá trị được truyền cho hàm **subtraction** là *giá trị* của **x** and **y**.

Trường hợp thứ tư cũng hoàn toàn tương tự. Thay vì viết

```
z = 4 + subtraction (x,y);
```

chúng ta có thể viết:

```
z = subtraction (x,y) + 4;
```

cũng hoàn toàn cho kết quả tương đương. Chú ý rằng dấu chấm phẩy được đặt ở cuối biểu thức chứ không cần thiết phải đặt ngay sau lời gọi hàm.

Revision #3
Created 5 October 2019 05:27:43 by Laptrinh.vn
Updated 5 October 2019 16:11:06 by Laptrinh.vn