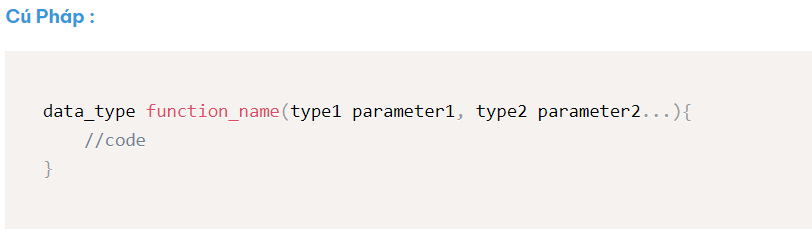
**1.Định Nghĩa Hàm**

Bạn đã từng sử dụng các hàm như sqrt, pow, abs... của các thư viện có sẵn trong C++, tuy nhiên để giải quyết bài toán của bạn thì bạn cần phải tự xây dựng các hàm để giải quyết các chức năng nhỏ cho bài toán của mình.

Hàm (function) là một các khối lệnh có nhiệm vụ thực hiện một chức năng nào đó.

Ở các bài học trước bạn đều viết mã nguồn C++ bên trong hàm main, chương trình sẽ trở nên khó quản lý khi số lượng dòng code trở nên lớn, cùng điểm qua các lợi ích của việc sử dụng hàm

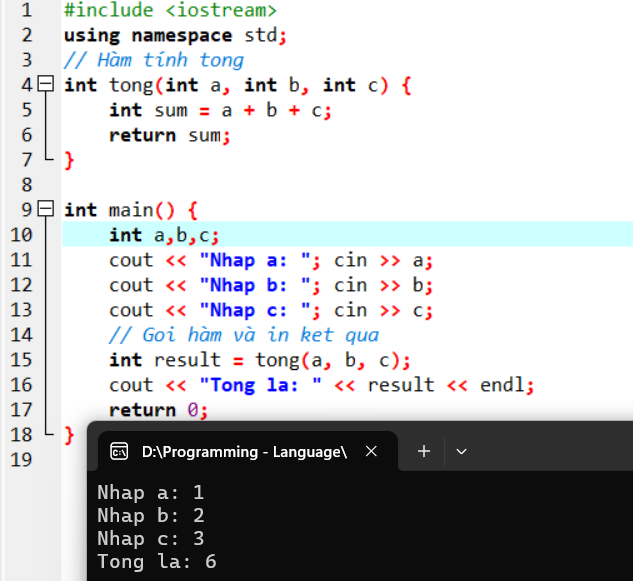




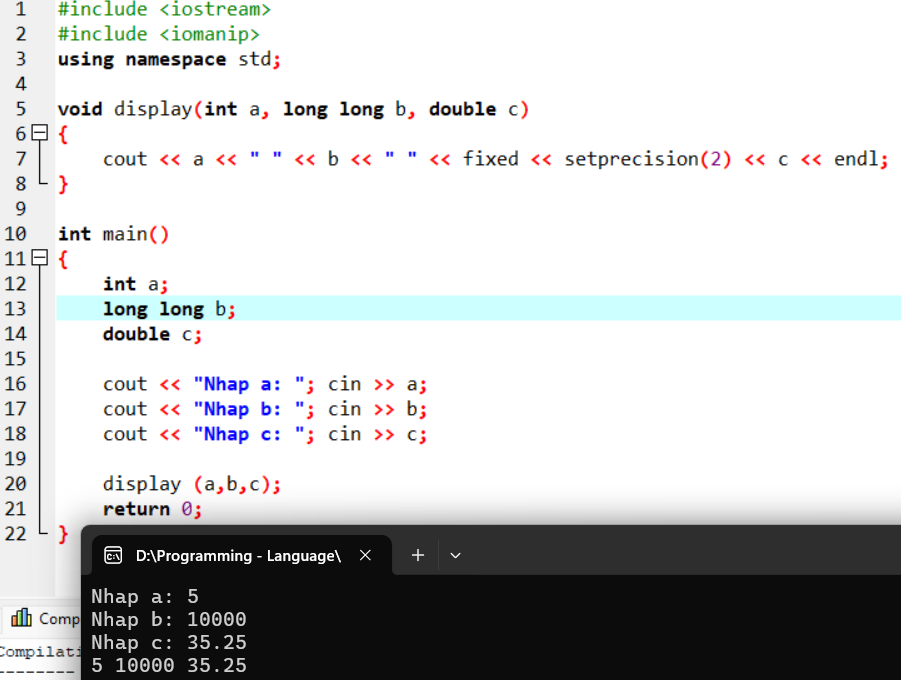
**Các thành phần của hàm :**

* data\_type : Kiểu trả về của hàm, có thể là các kiểu dữ liệu như int, long long, float, char, double, hoặc void (tương ứng với kiểu trả về là rỗng)
* function\_name : Tên của hàm, cần tuân theo quy tắc như đặt tên biến
* parameter : Tham số của hàm, đây được coi như đầu vào của hàm. Bạn có thể xây dựng bao nhiêu tham số tùy ý và lựa chọn kiểu dữ liệu cho từng tham số.
* code : Các câu lệnh bên trong của hàm

Ví dụ 1 : Hàm có kiểu trả về là int, có 3 tham số là a, b, c



Ví dụ 2 : Hàm có kiểu trả về là void, có 3 tham số là a kiểu int, b kiểu long long, c kiểu double

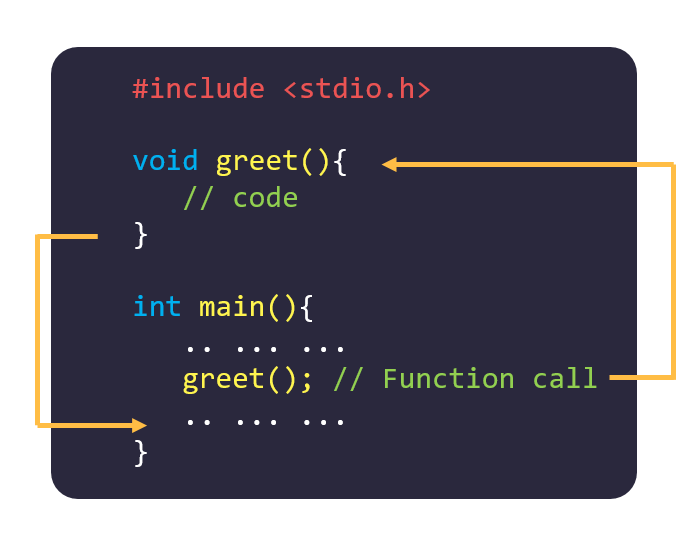


**2. Lời Gọi Hàm**

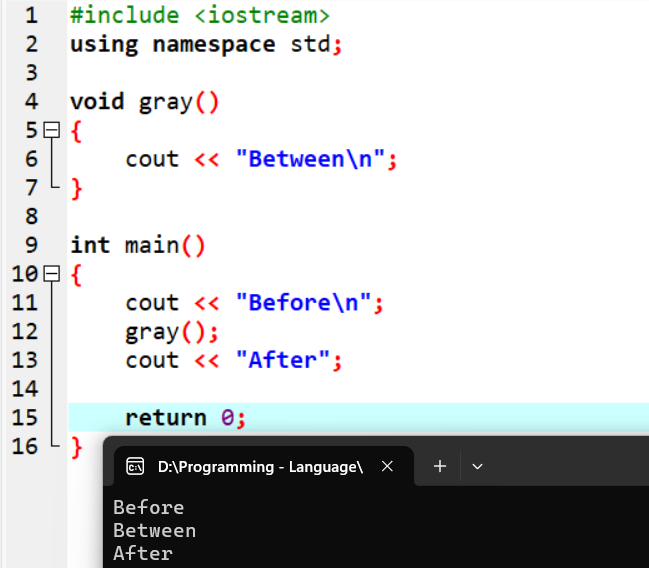
Sau khi xây dựng hàm xong để hàm có thể thực thi bạn cần gọi nó trong hàm main và truyền cho nó tham số nếu cần.

Khi bạn gọi hàm trong hàm main thì các câu lệnh bên trong hàm sẽ được thực thi, sau khi thực thi hết các câu lệnh thì hàm kết thúc và chương trình tiếp tục thực hiện các câu lệnh bên dưới hàm.

Mỗi lần bạn gọi hàm thì các câu lệnh trong hàm sẽ được thực hiện.



Ví dụ 3:



Giải thích :

1. Hàm main thực hiện câu lệnh đầu tiên in ra "Before"
2. Câu lệnh thứ 2 trong main là lời gọi hàm gray(), chương trình tiến hành nhảy vào bên trong hàm gray() và thực hiện lần lượt 2 câu lệnh in ra "Between"
3. Sau khi thực hiện xong 2 câu lệnh thì hàm gray() kết thúc tương đương câu lệnh thứ 2 trong main thực hiện xong
4. Hàm main tiếp tục thực hiện câu lệnh thứ 3 in ra "After" sau đó kết thúc chương trình

**3. Đối Số Và Tham Số**

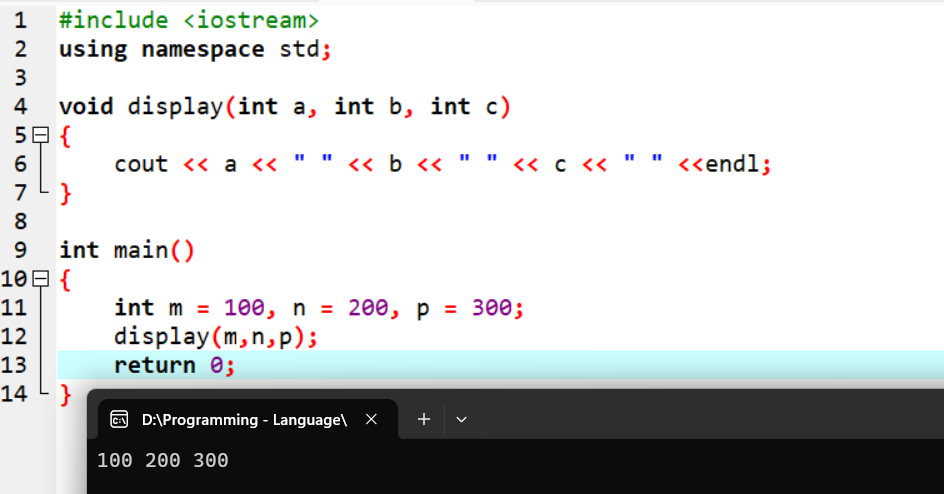
**Tham số** (Parameter) hay tham số hình thức là các thành phần khi bạn xây dựng hàm, xem xét ví dụ dưới đây thì a, b, c sẽ được gọi là tham số

**Đối số** (Argument) hay tham số chính thức là các giá trị bạn truyền vào cho hàm khi gọi hàm, xem xét ví dụ dưới đây thì m, n, p được gọi là đối số

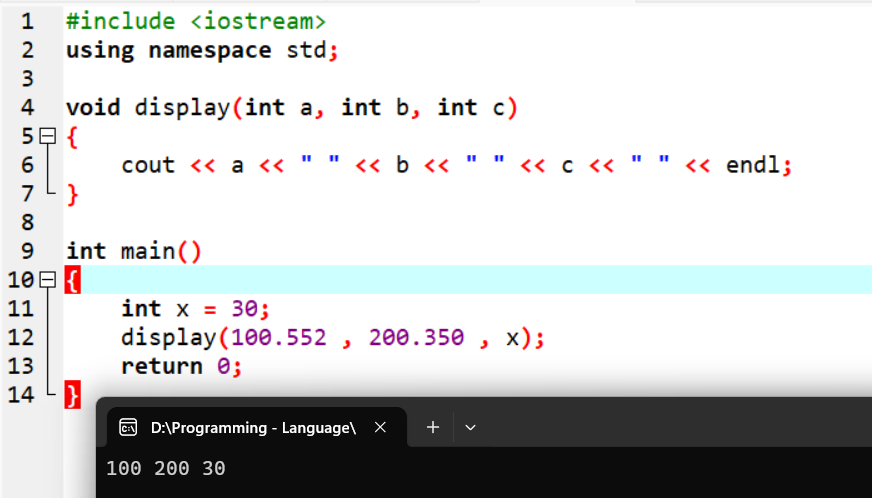
Khi bạn gọi hàm thì lần lượt giá trị của các đối số sẽ được gán cho tham số, trong ví dụ dưới thì m được gán cho a, n được gán cho b, p được gán cho c. Những gì thay đổi trên tham số sẽ không có ảnh hưởng gì tới đối số

**Chú ý :** Kiểu dữ liệu của đối số và tham số nên trùng nhau hoặc của tham số nên là kiểu dữ liệu lớn hơn kiểu dữ liệu của đối số. Ví dụ bạn xây dựng 1 hàm có tham số là long long thì nó có thể áp dụng với 1 số int nhưng ngược lại thì không.

Ví dụ 4:



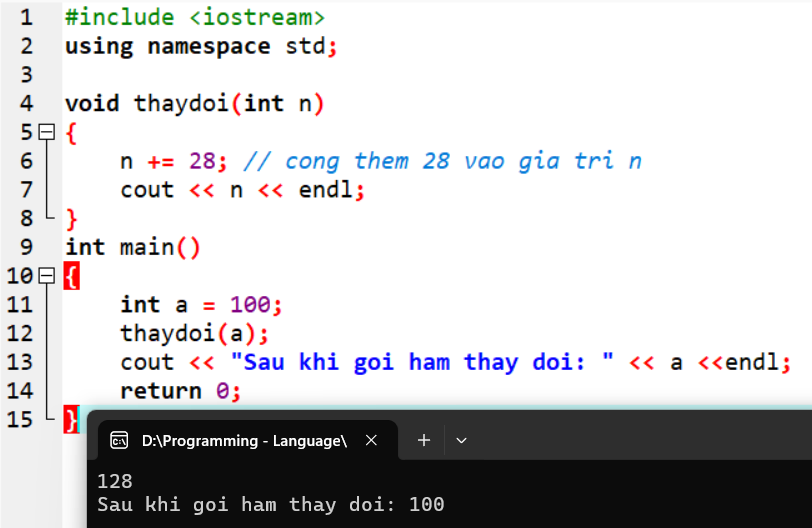
Ví dụ 5:



**Giải thích :**

1. a được gán giá trị là 100.552 nhưng do a có kiểu là int nên chỉ lưu được 100
2. b được gán giá trị là 200.350 nhưng do b có kiểu là int nên chỉ lưu được 200
3. c được gán giá trị của x tương tương với 30
4. Câu lệnh in ra a, b, c lần lượt là 100 200 30

Ví dụ 3 : Thay đổi tham số sẽ không ảnh hưởng gì tới đố số



Giải thích:

1. Tham số n được gán giá trị của đối số a nên n = 100
2. n += 28 => n = 128, câu lệnh printf bên trong hàm thay đổi in ra n sẽ là 128
3. Hàm kết thúc, câu lệnh in ra a thì a vẫn là 100