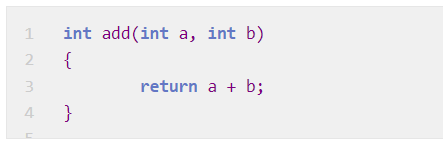
NẠP CHỒNG HÀM TRONG C++ (Funcition overloading)

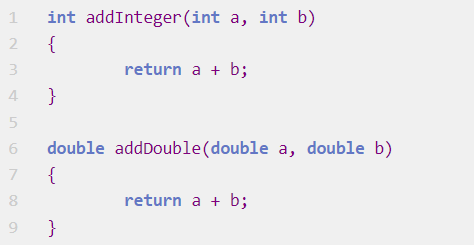
**1. Đặt vấn đề**

Chúng ta có hàm tính tổng 2 số nguyên:



Vì hàm trên có tham số kiểu số nguyên (int), nếu chúng ta có nhu cầu tính tổng 2 số chấm động, hàm này sẽ không thực hiện được.

Để giải quyết vấn đề này, chúng ta có thể tạo ra 2 hàm có tên khác nhau.

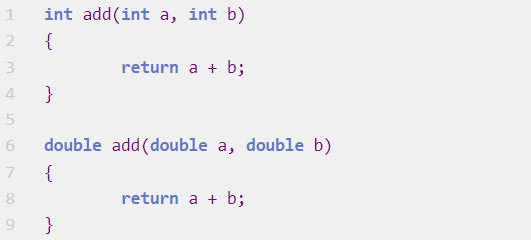


Như vậy, ta sẽ có nhiều hàm với các tên gọi khác nhau. Tuy nhiên, việc sử dụng tên như vậy sẽ gây bất lợi cho người lập trình khi gọi hàm. Nạp chồng hàm (function overloading) ra đời để giải quyết vấn đề trên.

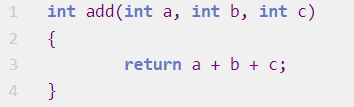
**2. Nạp chồng hàm trong C++ (Function overloading)**

Nạp chồng hàm (function overloading) là tính năng của ngôn ngữ C++ (không có trong C). Kỹ thuật này cho phép sử dụng cùng một tên gọi cho nhiều hàm (có cùng mục đích). Nhưng khác nhau về kiểu dữ liệu tham số hoặc số lượng tham số.

Giải quyết vấn đề trên:



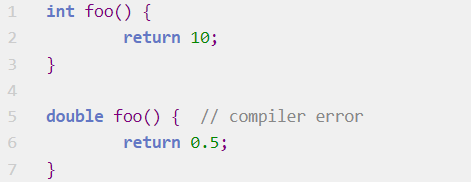
Chúng ta cũng có thể định nghĩa các hàm add() với số lượng tham số khác nhau.



**3. Một số hàm không thể nạp chồng trong C++**

3.1 Hàm chỉ khác nhau kiểu trả về

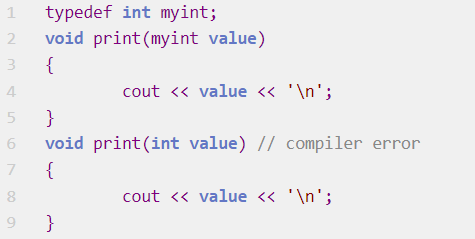
Chú ý: Không thể nạp chồng hàm nếu chúng chỉ khác nhau kiểu dữ liệu trả về.



Hai hàm trên giống nhau về tham số (không có), vì vậy trình biên dịch sẽ báo lỗi.

3.2 Tham số hàm kiểu typedef

Khai báo typedef chỉ là một bí danh (không phải kiểu dữ liệu mới), vì vậy chương trình bên dưới sẽ gặp lỗi:



3.3 Tham số hàm kiểu con trỏ\* và mảng []

Tham số hàm con trỏ \* và mảng [] là tương đương. Lúc này, khai báo mảng [] được chuyển đổi ngầm định thành một con trỏ.



Hai hàm trên giống nhau về tham số (int\*), vì vậy trình biên dịch sẽ báo lỗi.

3.4 Nạp chồng hàm và từ khóa const

Trước tiên chúng ta hãy xem hai ví dụ sau. Chương trình 1 biên dịch thất bại, nhưng chương trình 2 biên dịch và chạy tốt.





C++ cho phép nạp chồng hàm với tham số là const chỉ khi tham số const là **tham chiếu hoặc con trỏ.** Đó là lý do tại sao chương trình 1 gặp lỗi biên dịch, nhưng chương trình 2 hoạt động.

Trong chương trình 1, tham số ‘i’ được truyền theo giá trị, vì vậy ‘i', vì vậy ‘i' trong fun() là bản sao của ‘i' trong main(). Do đó, fun() không thể sửa đổi ‘i' của hàm main(). Vì vậy, không quan trọng việc ‘i' được nhận dưới dạng tham số const hay tham số bình thường.

Khi tham số là tham chiếu hoặc con trỏ, chúng ta có thể sửa đổi giá trị **được tham chiếu hoặc được trỏ tới**, do đó chúng ta có thể có **hai phiên bản của hàm, một phiên bản có thể sửa đổi giá trị được tham chiếu hoặc trỏ tới, một phiên bản không thể thay đổi.**

