**INHERITANCE**

1. **KHÁI QUÁT VỀ KẾ THỪA**
2. **Định nghĩa**

Tính kế thừa (Inheritance) là một trong bốn tính chất cơ bản của lập trình hướng đối tượng.

Khả năng tái sử dụng phương thức, thuộc tính của 1 lớp khác gọi là kế thừa. Nói cách khác, nó cho phép chúng ta tạo một lớp mới (lớp dẫn xuất/ lớp con) từ một lớp hiện có (lớp cơ sở/ lớp cha). Lớp con kế thừa những gì lớp cha có và cho phép.

Có nghĩa là:

* Nếu lớp A kế thừa lớp B thì lớp A sẽ có những thuộc tính và phương thức của lớp B.
* Lớp con có thể có các thuộc tính và phương thức riêng của nó.
* Class con chỉ được truy cập các thành viên public và protected của class cha, và không được phép truy cập đến thành viên private của class cha.

***Ví dụ:*** Quản lí trường đại học, với hai lớp Nhân viên và lớp Sinh viên, đều kế thừa từ lớp Người

|  |  |
| --- | --- |
| **Lớp Người** ***Thuộc tính*** Tên Ngày sinh Giới tính ***Phương thức*** Nhập/xem tên Nhập/xem ngày sinh Nhập/xem giới tính | |
| **Lớp Nhân viên kế thừa**  **từ Lớp Người**   |  | | --- | | ***Thuộc tính*** | | Lương | | ***Phương thức*** | | Nhập/xem lương | | |  | | --- | | **Lớp Nhân viên kế thừa**  **từ Lớp Người** | | ***Thuộc tính*** | | Lớp | | ***Phương thức*** | | Nhập/xem lớp | |

1. **Tại sao sử dụng kế thừa**

* Tránh việc trùng lặp dữ liệu
* Tăng tính tái sử dụng của code

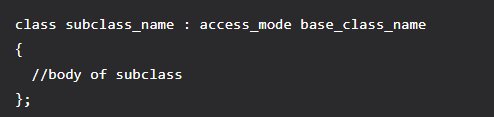
1. **Khi nào sử dụng kế thừa**

Kế thừa là một mối quan hệ is-a. Chúng ta sử dụng kế thừa chỉ khi có một mối quan hệ is-a giữa hai lớp.

***Ví dụ:***

* A car is a vehicle.
* Orange is a fruit.
* A surgeon is a doctor.
* A dog is an animal.

1. **Syntax**

****

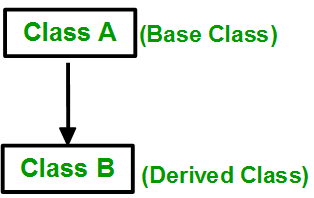
Trong đó:

* subclass\_name: Tên của lớp con
* access\_mode: Chế độ mà bạn muốn kế thừa lớp con này (Phạm vi truy cập)
  + Public: Thành viên public của lớp cơ sở sẽ trở thành công khai trong lớp dẫn xuất và các thành viên protected lớp cơ sở sẽ trở thành được bảo vệ trong lớp dẫn xuất.
  + Protected: Cả thành viên public và thành viên protected của lớp cơ sở sẽ trở thành được bảo vệ trong lớp dẫn xuất.
  + Private: Khi đó cả thành viên public và thành viên protected của lớp cơ sở sẽ trở thành Riêng tư trong lớp dẫn xuất
* Base\_class\_name: Tên lớp cha
* ***Lưu ý:*** Các thành viên private trong lớp cơ sở không thể được truy cập trực tiếp trong lớp dẫn xuất.
* Dưới đây là bảng tóm tắt 3 phương thức kế thừa và hiển thị chỉ định truy cập của các thành viên của lớp cơ sở trong lớp con khi dẫn xuất trong các chế độ public, protected, private.

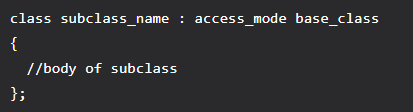
|  |  |  |  |
| --- | --- | --- | --- |
| Access modifier của các thành viên của lớp cha | access\_mode | | |
| Public | Protected | Private |
| Public | Public | Protected | Private |
| Protected | Protected | Protected | Private |
| Private | Không thể  truy cập (Bị ẩn) | Không thể  truy cập (Bị ẩn) | Không thể  truy cập (Bị ẩn) |

1. **ĐƠN KẾ THỪA, ĐA KẾ THỪA**
   1. **Đơn kế thừa (Single Inheritance)**

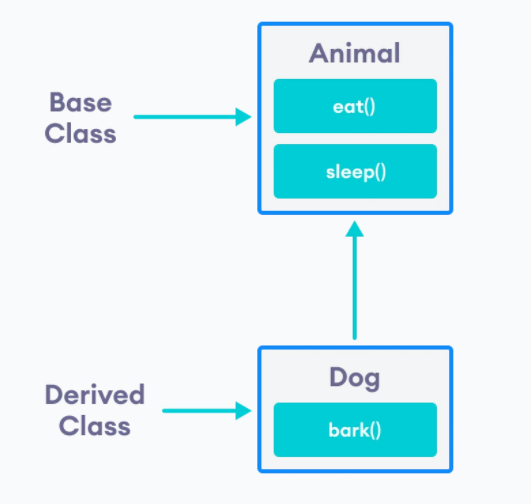
Đơn kế thừa: nghĩa là một lớp chỉ được kế thừa từ đúng một lớp khác. Hay nói cách khác, lớp con chỉ có duy nhất một lớp cha.



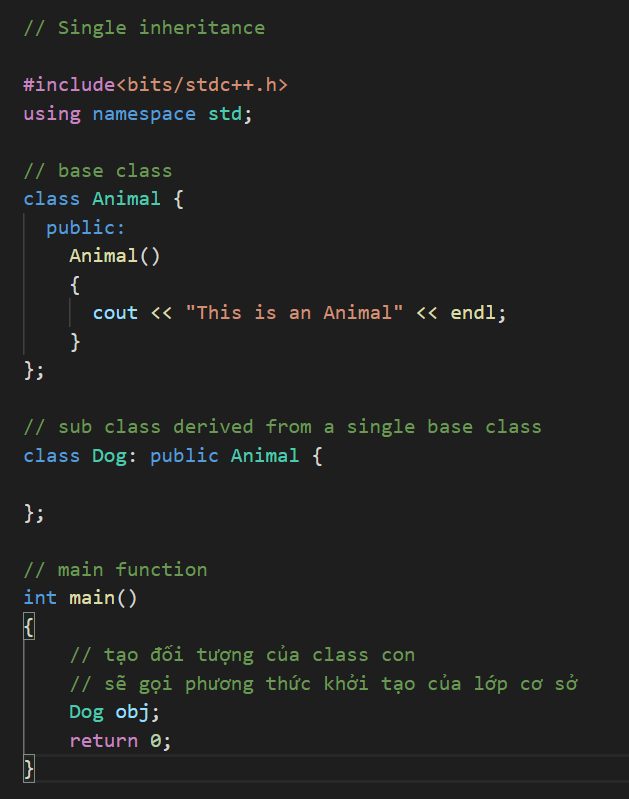
* ***Syntax:***



* ***Ví dụ:***

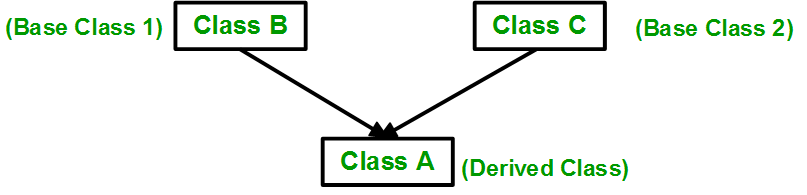


***// Code:***

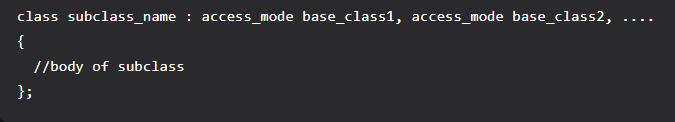


* 1. **Đa kế thừa**

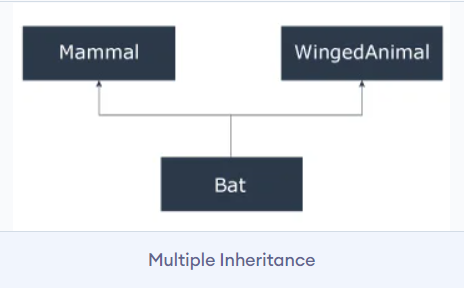
Đa kế thừa là một tính năng của ngôn ngữ C++. Trong đó một lớp có thể kế thừa từ nhiều hơn một lớp khác. Nghĩa là một lớp con được kế thừa từ nhiều hơn một lớp cơ sở.



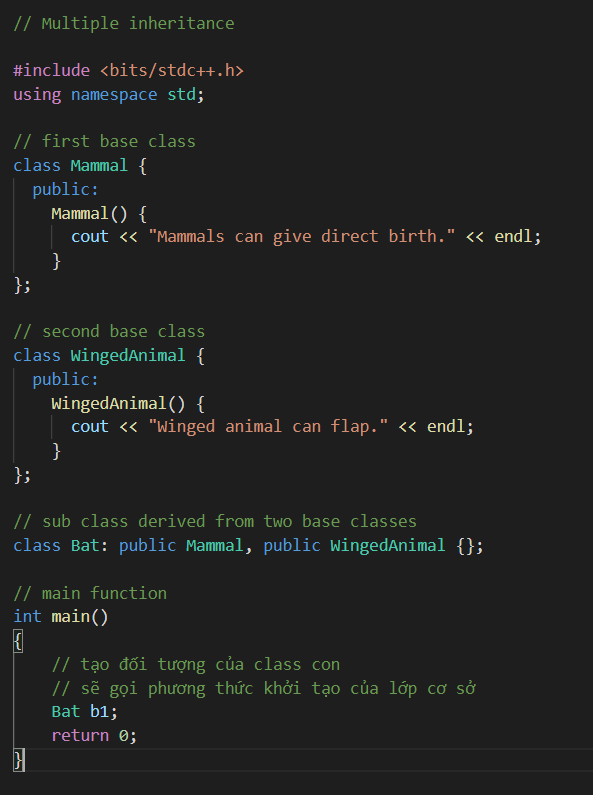
* ***Syntax:***



* Ở đây, số lượng các lớp cơ sở sẽ được phân tách bằng dấu phẩy (‘, ‘) và chế độ truy cập cho mọi lớp cơ sở phải được chỉ định.
* ***Ví dụ:*** Một lớp Bat có nguồn gốc từ các lớp cơ sở Mammal và WingedAnimal. Nó có ý nghĩa vì dơi là động vật có vú cũng như động vật có cánh.

******

***// Code:***

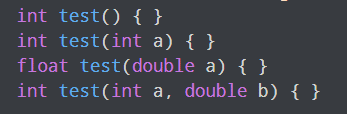
******

1. **FUNCTION OVERLOADING & OPERATOR OVERLOADING**

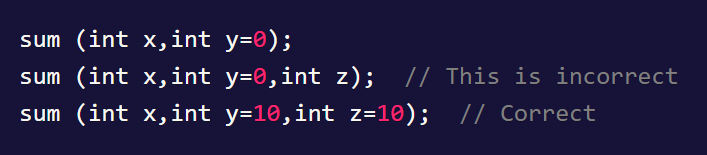
* Khái niệm Overloading thường được sử dụng khi một khối chương trình thực thi cùng một tác vụ về mặt khái niệm nhưng có sự khác biệt nhỏ trong một tập các tham số.
* Nạp chồng là một khái niệm được sử dụng để tránh mã dư thừa trong đó tên phương thức hoặc toán tử giống nhau được sử dụng nhiều lần nhưng với một tập tham số hoặc số toán hạng khác nhau.

1. **Function overloading (Nạp chồng hàm)**

Nạp chồng hàm là một tính năng của lập trình hướng đối tượng trong đó hai hoặc nhiều hàm có thể có cùng tên nhưng số lượng và / hoặc kiểu đối số được truyền khác nhau. Nạp chồng hàm có thể được coi là một ví dụ về tính năng đa hình trong C ++.

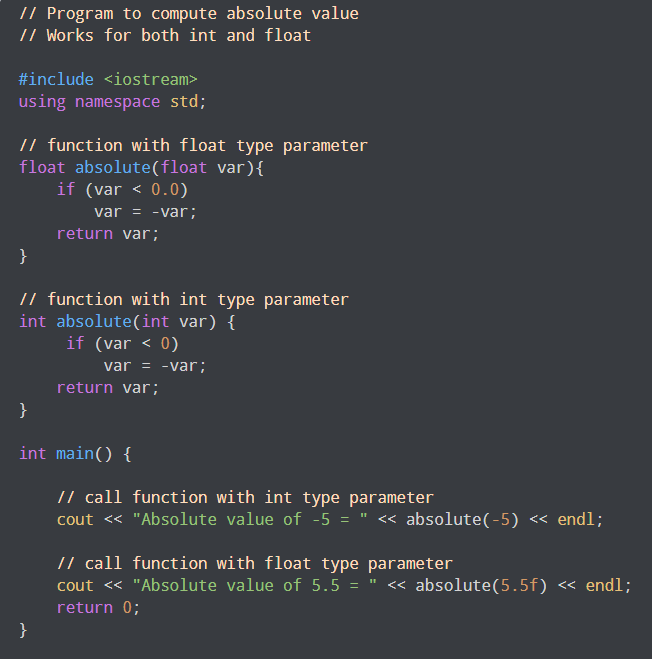


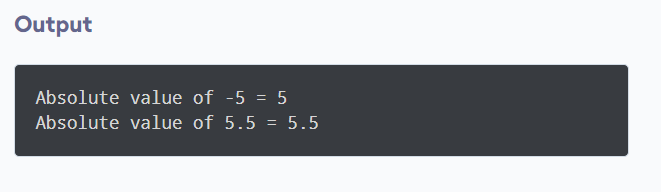
* Ở đây, cả 4 hàm đều là hàm nạp chồng.
* Chú ý rằng kiểu trả về của cả 4 hàm này không giống nhau. Các hàm nạp chồng có thể có hoặc không có các kiểu trả về khác nhau nhưng chúng phải có các đối số khác nhau. Ví dụ:

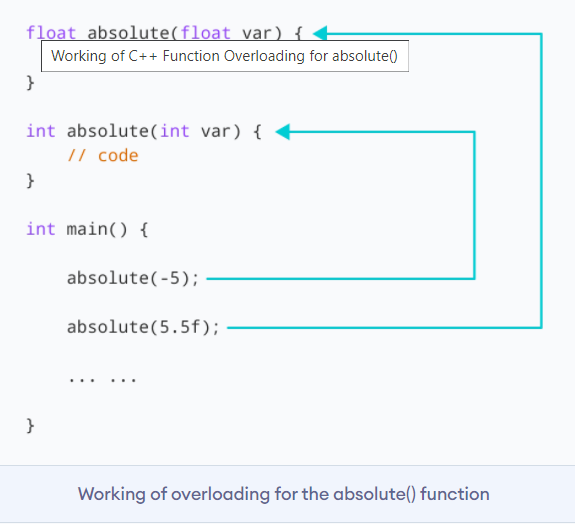


* Ở đây, cả hai hàm đều có cùng tên, cùng kiểu và cùng số lượng đối số. Do đó, trình biên dịch sẽ xuất hiện lỗi.

Ví dụ:



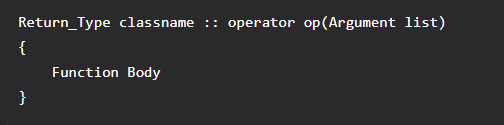




1. **Operator overloading (Nạp chồng toán tử)**

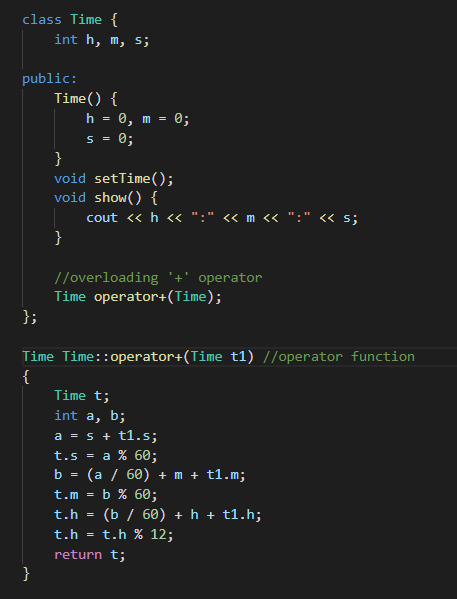
C ++ cung cấp một hàm đặc biệt để thay đổi chức năng hiện tại của một số toán tử trong lớp của nó, thường được gọi là nạp chồng toán tử. Nạp chồng toán tử là phương pháp mà chúng ta có thể thay đổi chức năng của một số toán tử cụ thể để thực hiện một số tác vụ khác nhau.

* Syntax:



Trong đó:

* return\_type: kiểu giá trị được trả về cho 1 đối tượng khác
* classname: tên lớp
* operator: từ khoá
* op: toán tử được nạp chồng
* argument list: danh sách các đối số được truyền cho hàm
* Ví dụ: Nạp chồng toán tử + để cộng hai đối tượng lớp Time



* ***Nhận xét:*** Trong khi phép cộng hai số bình thường trả về kết quả tổng. Trong trường hợp trên, chúng ta đã nạp chồng toán tử +, để thực hiện phép cộng hai đối tượng lớp Time. Chúng ta thêm các giá trị giây, phút và giờ một cách riêng biệt để trả về giá trị mới của thời gian.

1. **FUNCTION OVERRIDING (Ghi đè hàm)**

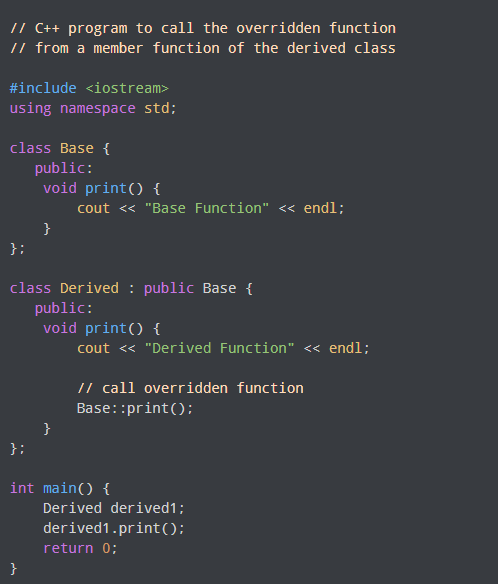
Trong kế thừa, khi lớp con khai báo phương thức có tên trùng với phương thức ở lớp cha thì phương thức của lớp cha sẽ bị thay thế bởi phương thức ở lớp con. Điều này gọi là ghi đè

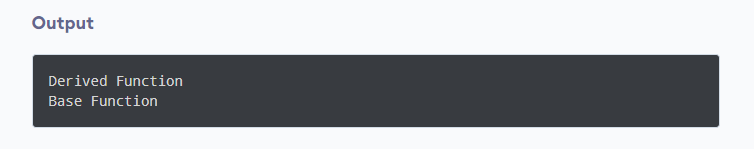
* Ví dụ:

Hàm print () được định nghĩa trong cả hai lớp lớp cha và lớp con. Vì vậy, khi chúng ta gọi print () từ đối tượng derived1 trong lớp Derived, thì print () từ Derived được thực thi bằng cách ghi đè hàm trong Base.

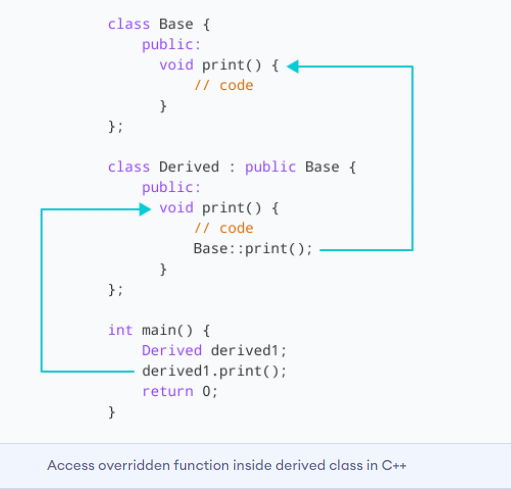
Nói cách khác, phương thức print () của lớp cha đã bị thay thế bởi phương thức print () của lớp con.

* + ***Lưu ý:*** Để truy cập hàm bị ghi đè của lớp cơ sở, chúng ta sử dụng toán tử phân giải phạm vi **: :**





Ở đây, dòng lệnh **Base :: print () ;**truy cập vào hàm print () của lớp cha

****

1. **HÀM FRIEND**

Ẩn dữ liệu là một khái niệm cơ bản của lập trình hướng đối tượng. Nó hạn chế quyền truy cập của các thành viên private từ bên ngoài của class.

Tương tự, các thành viên protected chỉ có thể được truy cập bởi các lớp dẫn xuất và không thể truy cập từ bên ngoài.

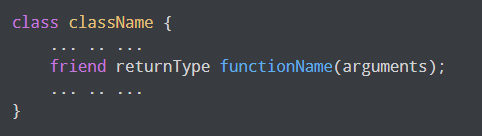
Tuy nhiên, có một tính năng trong C ++ được gọi là hàm bạn phá vỡ quy tắc này và cho phép chúng ta truy cập các hàm thành viên từ bên ngoài lớp.

Tương tự, một lớp bạn có thể truy cập các thành viên private và protected của lớp khác mà lớp đó được khai báo là bạn.

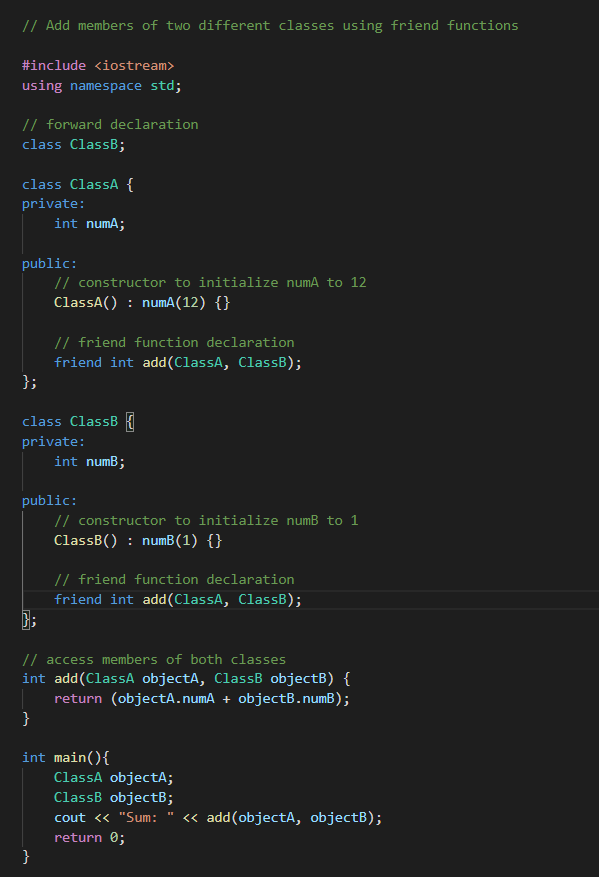
* 1. **Friend Function**

Hàm bạn có quyền truy cập vào tất cả các thành viên private và protected của lớp.

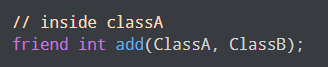
* **Declaration (Khai báo):** Chúng ta khai báo một hàm friend bằng cách sử dụng từ khóa friend bên trong phần thân của class.
  + ***Syntax:***



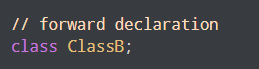
* **Đặc điểm**
  + Hàm không thuộc phạm vi lớp mà nó đã được khai báo là bạn.
  + Nó không thể được gọi bằng cách sử dụng đối tượng vì nó không thuộc phạm vi của lớp đó.
  + Nó có thể được gọi giống như một hàm bình thường mà không cần sử dụng đối tượng.
  + Nó không thể truy cập trực tiếp vào tên thành viên và phải sử dụng tên đối tượng và toán tử. với tên thành viên
  + Nó có thể được khai báo trong phần private hoặc phần public
* ***Ví dụ:***



Trong chương trình này, ClassA và ClassB đã khai báo add () là một hàm bạn. Do đó, chức năng này có thể truy cập dữ liệu private của cả hai lớp. Một điều cần lưu ý ở đây là hàm bạn bên trong ClassA đang sử dụng ClassB. Tuy nhiên, chúng ta chưa xác định ClassB tại thời điểm này.

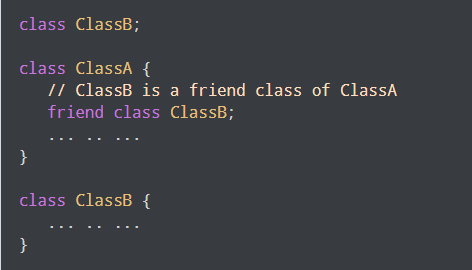


Để nó hoạt động, chúng ta cần một khai báo trước ClassB trong chương trình.

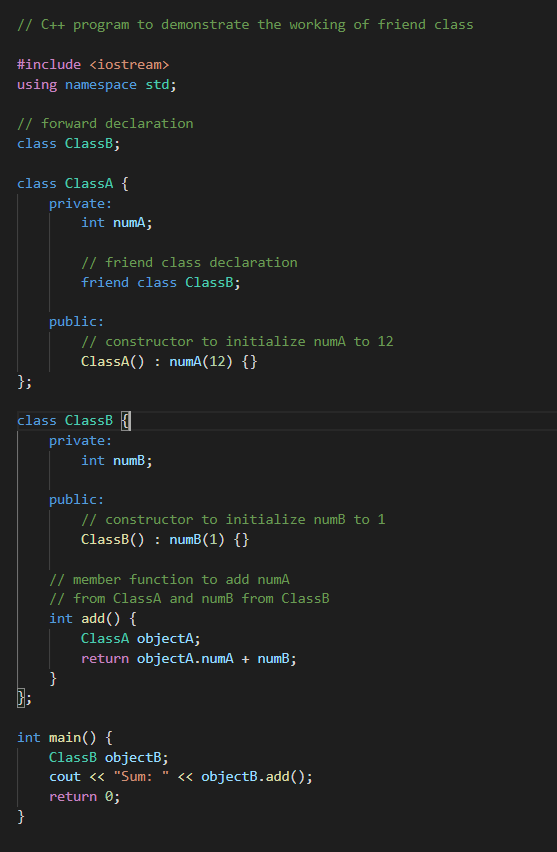


* 1. **Friend Classes**

Chúng ta cũng có thể sử dụng friend Class trong C ++ bằng cách sử dụng từ khóa friend.



* Tính chất và mối quan hệ của lớp bạn
  + Khi một lớp được khai báo là lớp bạn, tất cả các hàm thành viên của lớp bạn sẽ trở thành hàm bạn.
  + Vì ClassB là một lớp bạn, chúng ta có thể truy cập tất cả các thành viên của ClassA từ bên trong ClassB. Tuy nhiên, chúng ta không thể truy cập các thành viên của ClassB từ bên trong ClassA.
* ***Ví dụ:***

******

* Nhận xét:
  + Ở đây, ClassB là một lớp bạn của ClassA. Vì vậy, ClassB có quyền truy cập vào các thành viên của classA.
  + Trong ClassB, chúng ta đã tạo một hàm add () trả về tổng của numA và numB. Vì ClassB là một lớp bạn, chúng ta có thể tạo các đối tượng của ClassA bên trong ClassB.