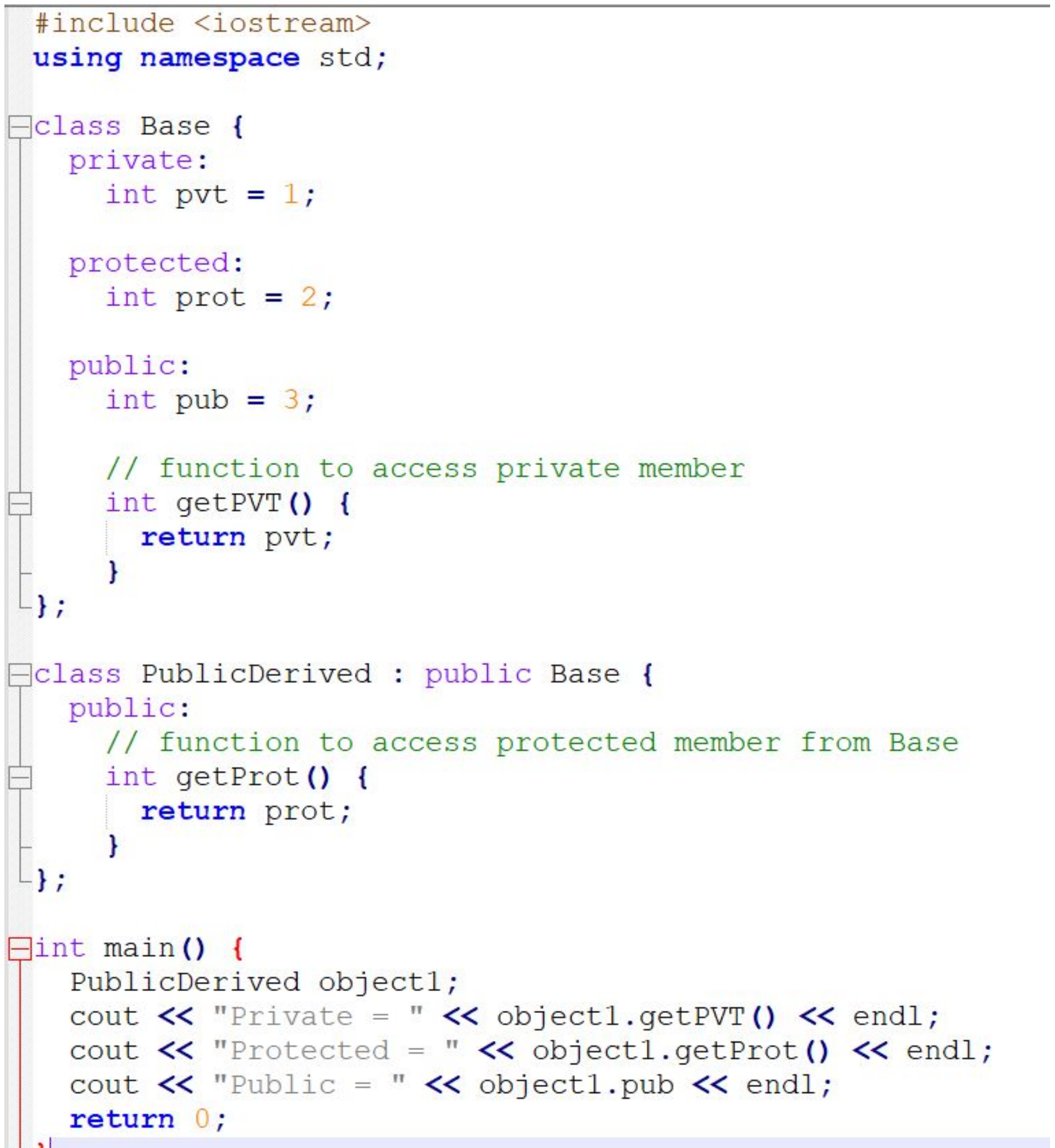
C++ Inheritance and Polymorphism

Public, Protected and Private Inheritance

- **public inheritance** makes public members of the base class public in the derived class, and the protected members of the base class remain protected in the derived class
- **protected inheritance** makes the public and protected members of the base class protected in the derived class
- **private inheritance** makes the public and protected members of the base class private in the derived class
- **private members of the base class are inaccessible to the derived class**

Example 1

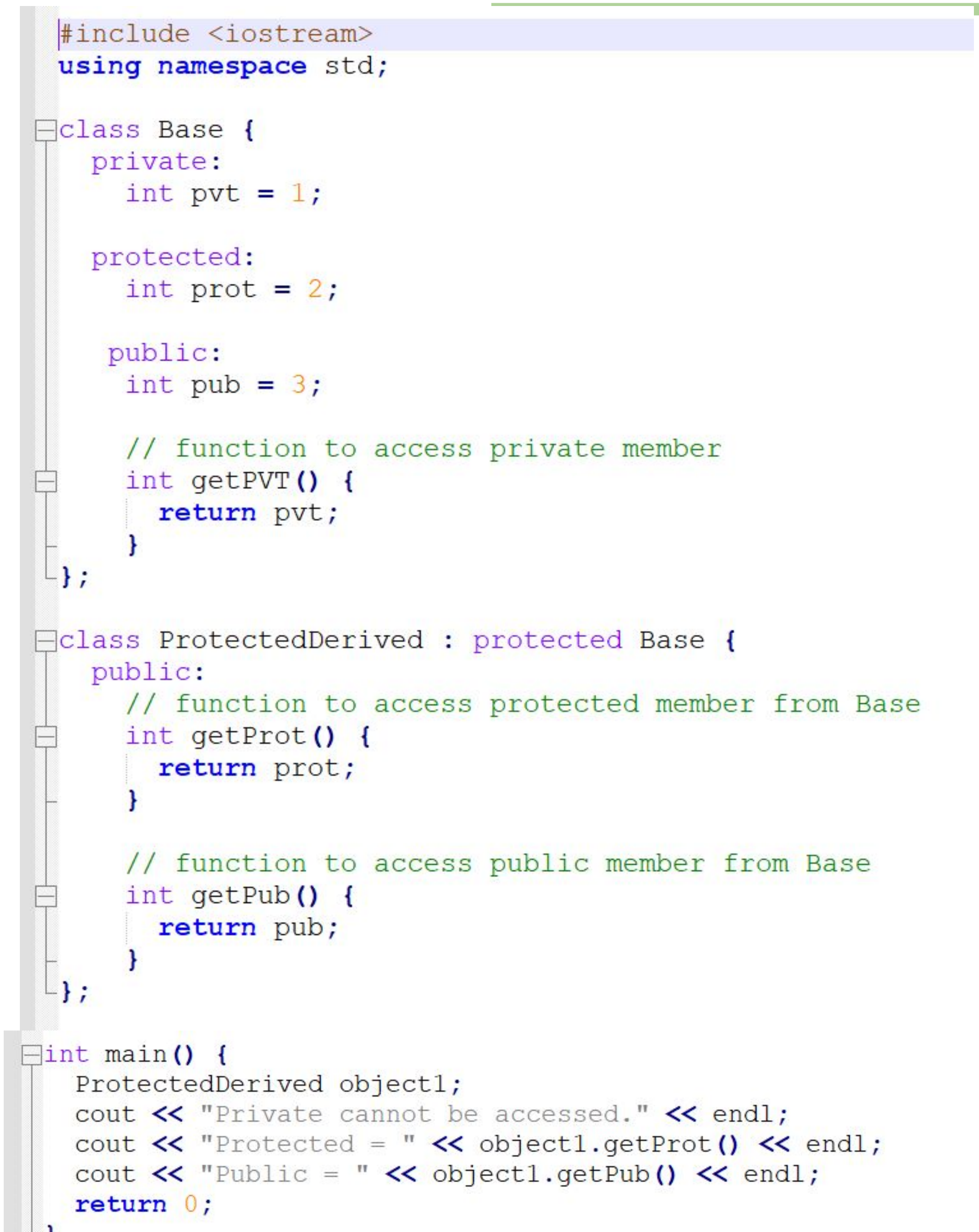


```cpp
#include <iostream>
using namespace std;

class Base {
  private:
    int pvt = 1;

  protected:
    int prot = 2;

  public:
    int pub = 3;

    // function to access private member
    int getPVT() {
      return pvt;
    }
};

class PublicDerived : public Base {
  public:
    // function to access protected member from Base
    int getProt() {
      return prot;
    }
};

int main() {
  PublicDerived object1;
  cout << "Private = " << object1.getPVT() << endl;
  cout << "Protected = " << object1.getProt() << endl;
  cout << "Public = " << object1.pub << endl;
  return 0;
```

// Error: member "Base::pvt" is inaccessible
cout << "Private = " << object1.pvt;

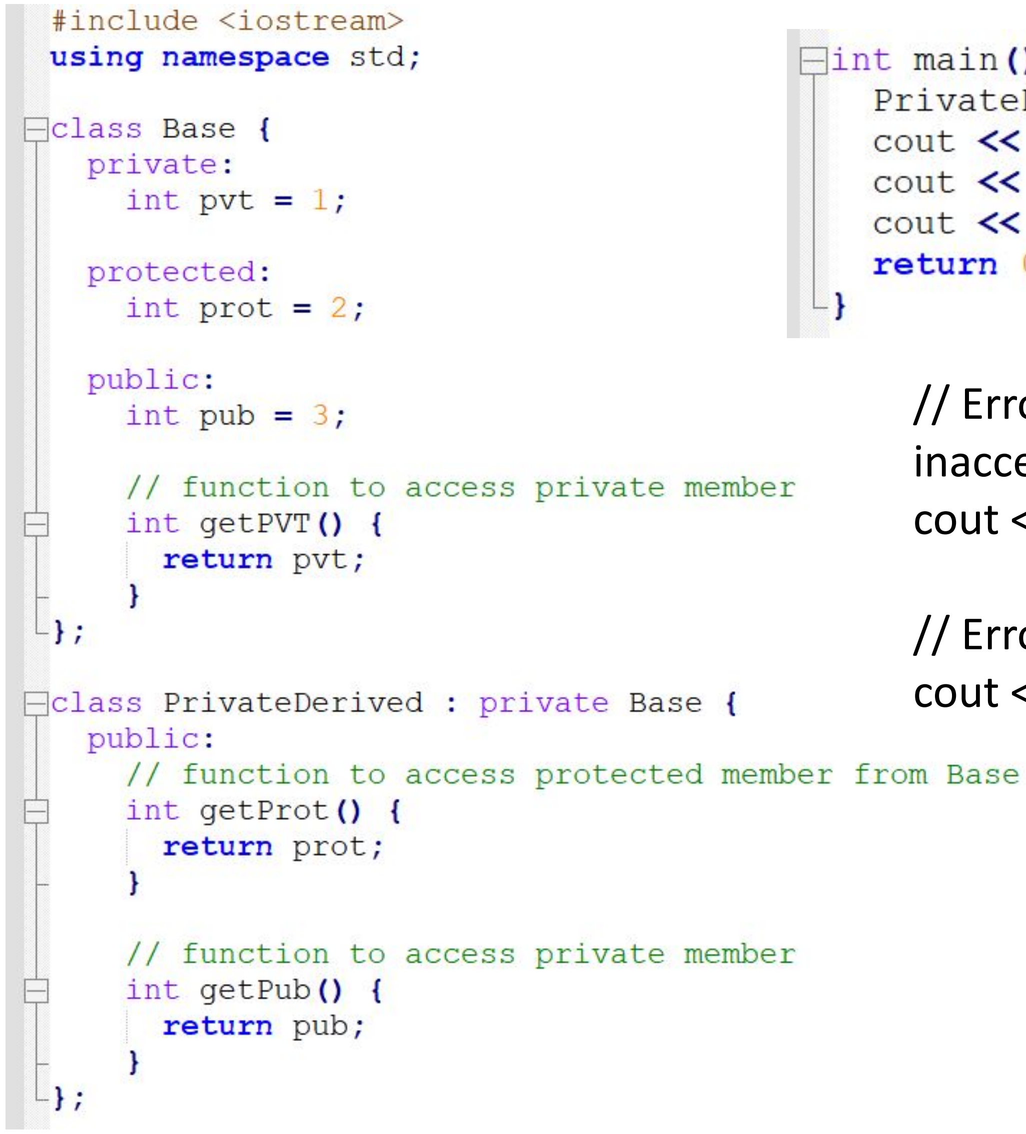
// Error: member "Base::prot" is inaccessible
cout << "Protected = " << object1.prot;

Example 2

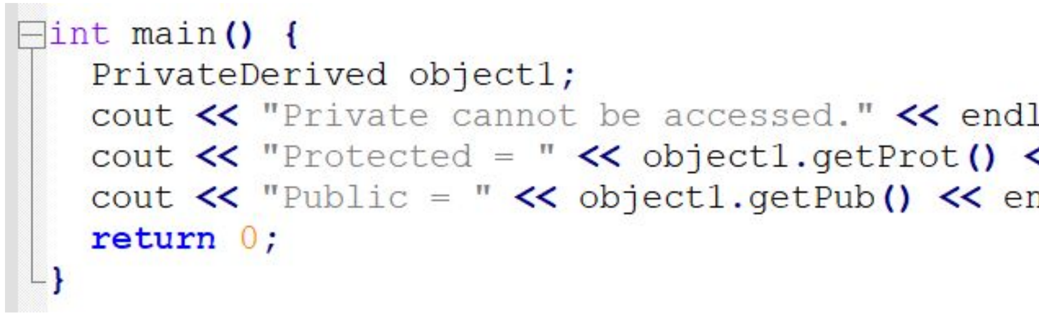


```
#include <iostream>
using namespace std;

class Base {
  private:
    int pvt = 1;

  protected:
    int prot = 2;

   public:
    int pub = 3;

    // function to access private member
    int getPVT() {
      return pvt;
    }
};

class ProtectedDerived : protected Base {
  public:
    // function to access protected member from Base
    int getProt() {
      return prot;
    }

    // function to access public member from Base
    int getPub() {
      return pub;
    }
};

int main() {
  ProtectedDerived object1;
  cout << "Private cannot be accessed." << endl;
  cout << "Protected = " << object1.getProt() << endl;
  cout << "Public = " << object1.getPub() << endl;
  return 0;
```

Example 3



```
#include <iostream>
using namespace std;

class Base {
  private:
    int pvt = 1;

  protected:
    int prot = 2;

  public:
    int pub = 3;

    // function to access private member
    int getPVT() {
      return pvt;
    }
};

class PrivateDerived : private Base {
  public:
    // function to access protected member from Base
    int getProt() {
      return prot;
    }

    // function to access private member
    int getPub() {
      return pub;
    }
};
```



```
int main() {
    PrivateDerived object1;
    cout << "Private cannot be accessed." << endl
    cout << "Protected = " << object1.getProt() <
    cout << "Public = " << object1.getPub() << en
    return 0;
}
```

// Error: member "Base::getPVT()" is inaccessible
cout << "Private = " << object1.getPVT();

// Error: member "Base::pub" is inaccessible
cout << "Public = " << object1.pub;

Inheritance

```
class Vehicle {
  public:
    string brand = “HONDA";
    void func() {
      cout << “I am HONDA! \n" ;
    }
};

class Car: public Vehicle {
  public:
    string model = “CIVIC";
};
int main() {
  Car myCar;
  myCar.func();
  cout << myCar.brand + " " + myCar.model;
  return 0;
}
```

Multi Level Inheritance

```
class MyClass {
  public:
    void myFunction() {
      cout << "Some content in parent class." ;
    }
};


class MyChild: public MyClass {
};


class MyGrandChild: public MyChild {
};

int main() {
  MyGrandChild myObj;
  myObj.myFunction();
  return 0;
}
```

Multiple Inheritance

```
class MyClass {
  public:
    void myFunction() {
      cout << "Some content in parent class." ;
    }
};


class MyOtherClass {
  public:
    void myOtherFunction() {
      cout << "Some content in another class." ;
    }
};


class MyChildClass: public MyClass, public MyOtherClass {
};

int main() {
  MyChildClass myObj;
  myObj.myFunction();
  myObj.myOtherFunction();
  return 0;
}
```

Access Specified

Public: members of a class are accessible from outside the class

Private: members can only be accessed within the class)

Protected: is similar to private, but it can also be accessed in the inherited class:

```
class Employee {
  protected:
    int salary;
};

// Derived class
class Programmer: public Employee {
  public:
    int bonus;
    void setSalary(int s) {
      salary = s;
    }
    int getSalary() {
      return salary;
    }
};

int main() {
  Programmer myObj;
  myObj.setSalary(50000);
  myObj.bonus = 15000;
  cout << "Salary: " << myObj.getSalary() << "\n";
  cout << "Bonus: " << myObj.bonus << "\n";
  return 0;
}
```

Polymorphism

```cpp
// Base class
class Animal {
  public:
    void animalSound() {
      cout << "The animal makes a sound \n";
    }
};

// Derived class
class Pig : public Animal {
  public:
    void animalSound() {
      cout << "The pig says: wee wee \n";
    }
};

// Derived class
class Dog : public Animal {
  public:
    void animalSound() {
      cout << "The dog says: bow wow \n";
    }
};
```

```cpp
int main() {
  Animal myAnimal;
  Pig myPig;
  Dog myDog;

  myAnimal.animalSound();
  myPig.animalSound();
  myDog.animalSound();
  return 0;
}
```

C++ Inheritance and Polymorphism MCQs

MCQ

1. What is Inheritance in C++?
a) Wrapping of data into a single class
b) Deriving new classes from existing classes
c) Overloading of classes
d) Classes with same names

MCQ

What is meant by multiple inheritance?

A. Deriving a base class from derived class
B. Deriving a derived class from base class
C. Deriving a derived class from more than one base class
D. None of the mentioned

MCQ

2. How many specifies are used to derive a class?
a) 1
b) 2
c) 3
d) 4

MCQ

3. Which specifier makes all the data members and functions of base class inaccessible by the derived class?
a) private
b) protected
c) public
d) both private and protected

MCQ

When the inheritance is private, the private methods in base class are _________ in the derived class (in C++).

A. Inaccessible
B. Accessible
C. Protected
D. Public

MCQ

4. If a class is derived privately from a base class then
a) no members of the base class is inherited
b) all members are accessible by the derived class
c) all the members are inherited by the class but are hidden and cannot be accessible
d) no derivation of the class gives an error

MCQ

What will be the order of execution of base class constructors in the following method of inheritance.class a: public b, public c {...};

A. b(); c(); a();
B. c(); b(); a();
C. a(); b(); c();
D. b(); a(); c();

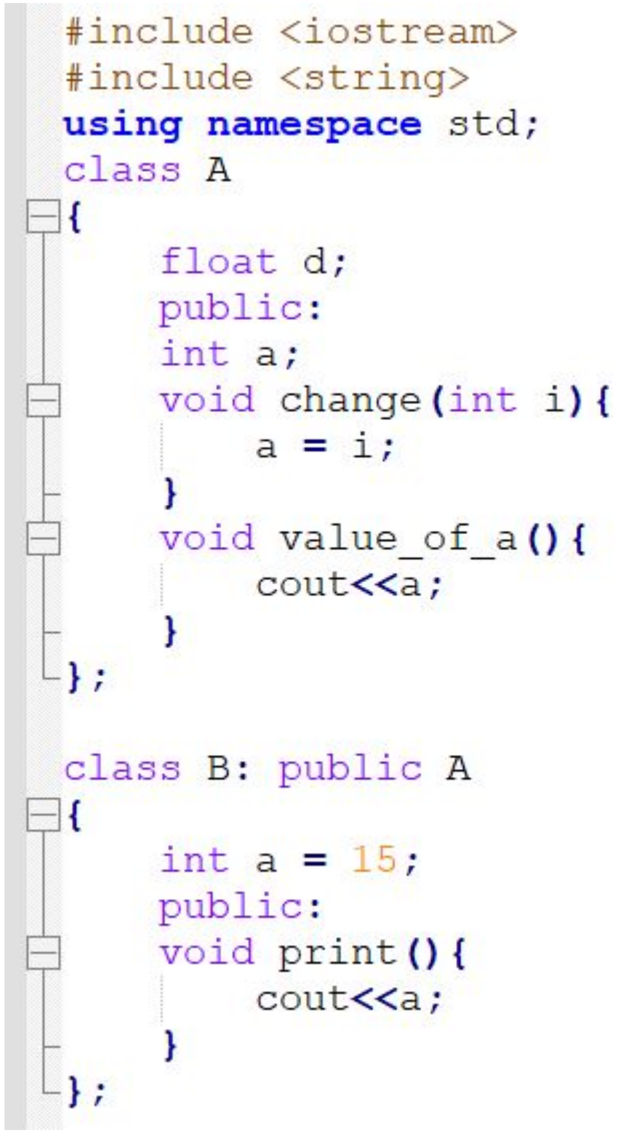
MCQ

Inheritance allow in C++ Program?

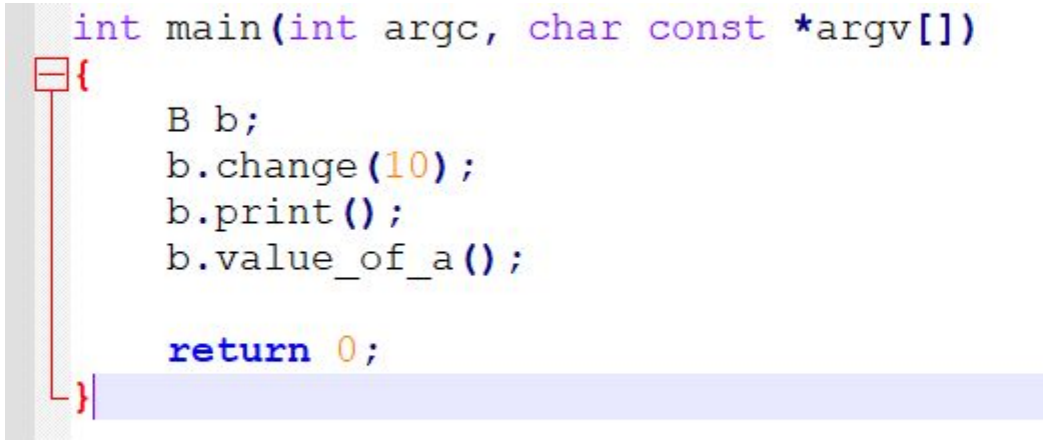
A. Class Re-usability
B. Creating a hierarchy of classes
C. Extendibility
D. All of the above

What is a virtual function in C++?
a) Any member function of a class
b) All functions that are derived from the base class
c) All the members that are accessing base class data members
d) All the functions which are declared in the base class and is re-defined/overridden by the derived class

What will be the output of the following C++ code?



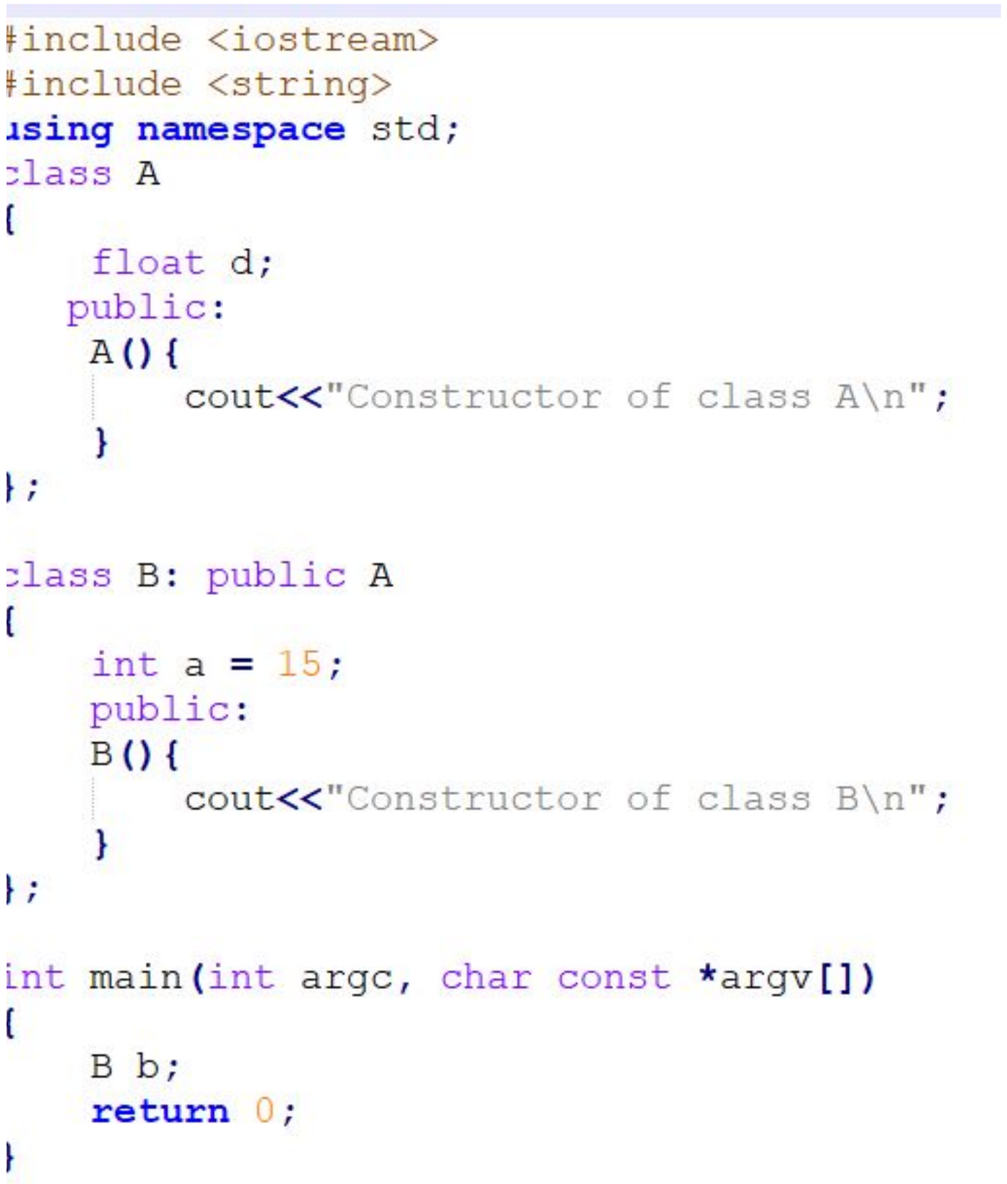
```cpp
#include <iostream>
#include <string>
using namespace std;
class A
{
    float d;
    public:
    int a;
    void change(int i){
        a = i;
    }
    void value_of_a(){
        cout<<a;
    }
};

class B: public A
{
    int a = 15;
    public:
    void print(){
        cout<<a;
    }
};
```



```cpp
int main(int argc, char const *argv[])
{
    B b;
    b.change(10);
    b.print();
    b.value_of_a();

    return 0;
}
```

a) 1010
b) 1510
c) 1515
d) 5110

What will be the output of the following C++ code?



```
#include <iostream>
#include <string>
using namespace std;
class A
{
    float d;
   public:
    A(){
        cout<<"Constructor of class A\n";
    }
};

class B: public A
{
    int a = 15;
    public:
    B(){
        cout<<"Constructor of class B\n";
    }
};

int main(int argc, char const *argv[])
{
    B b;
    return 0;
}
```

a)

```
Constructor of class A
Constructor of class B
```

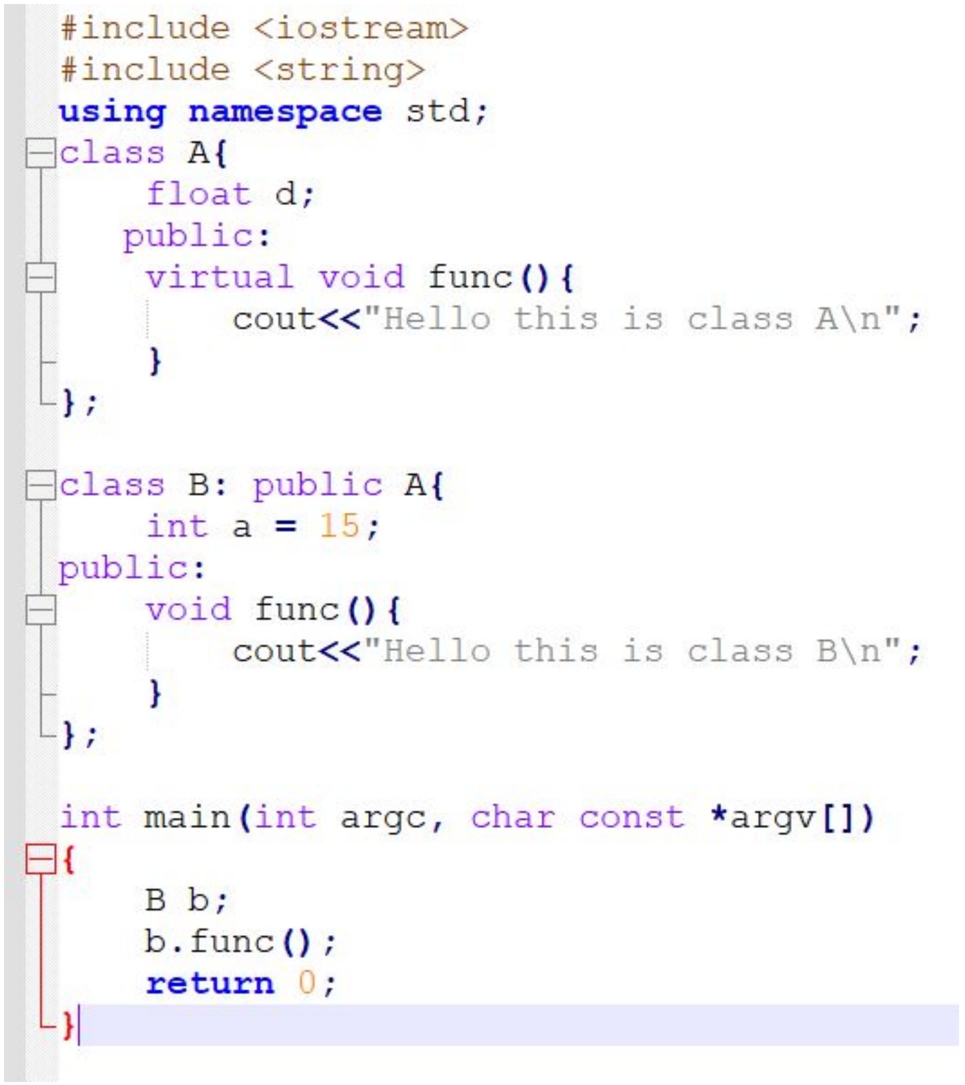
b) Constructor of class A

c) Constructor of class B

d)

```
Constructor of class B
Constructor of class A
```

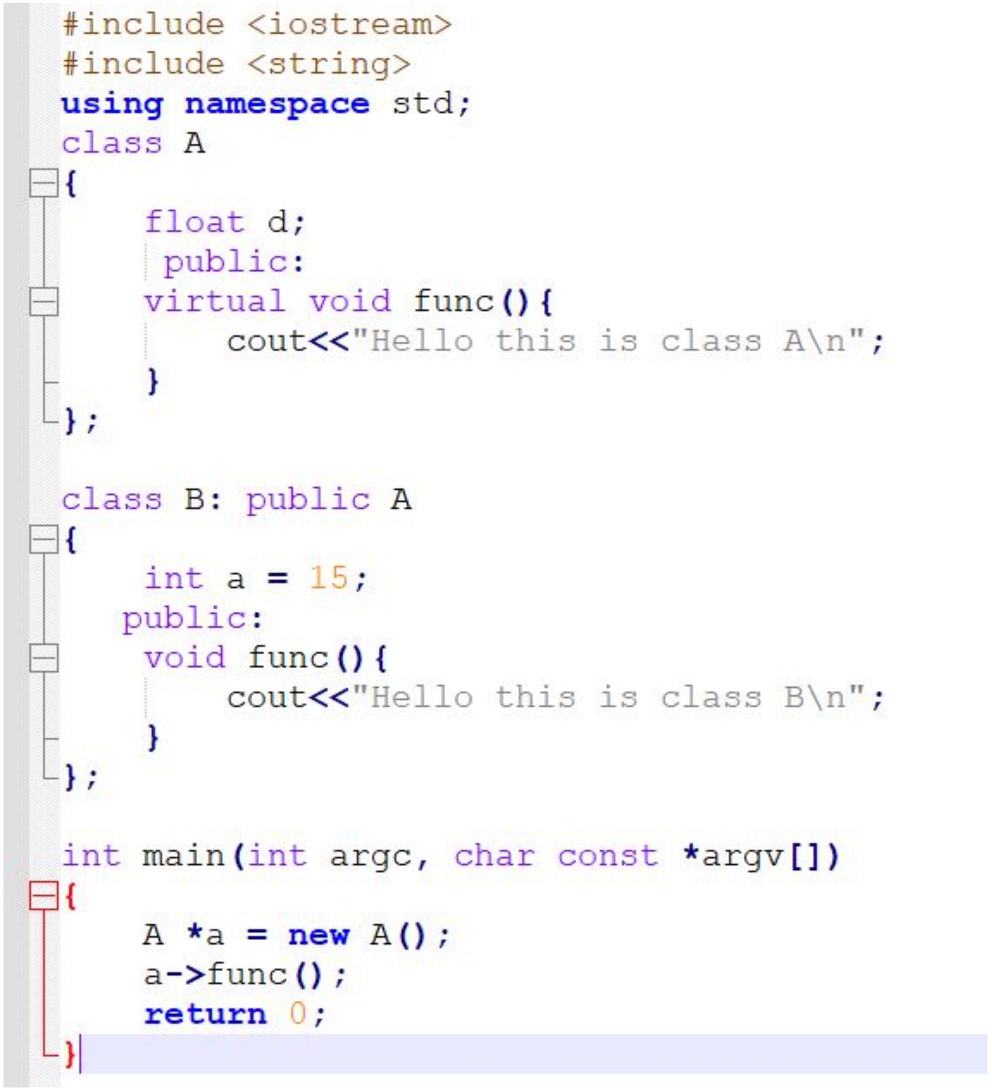
What will be the output of the following C++ code?



```
#include <iostream>
#include <string>
using namespace std;
class A{
    float d;
   public:
    virtual void func(){
        cout<<"Hello this is class A\n";
    }
};

class B: public A{
    int a = 15;
public:
    void func(){
        cout<<"Hello this is class B\n";
    }
};

int main(int argc, char const *argv[])
{
    B b;
    b.func();
    return 0;
}
```

a) Hello this is class B
b) Hello this is class A
c) Error
d) Segmentation fault

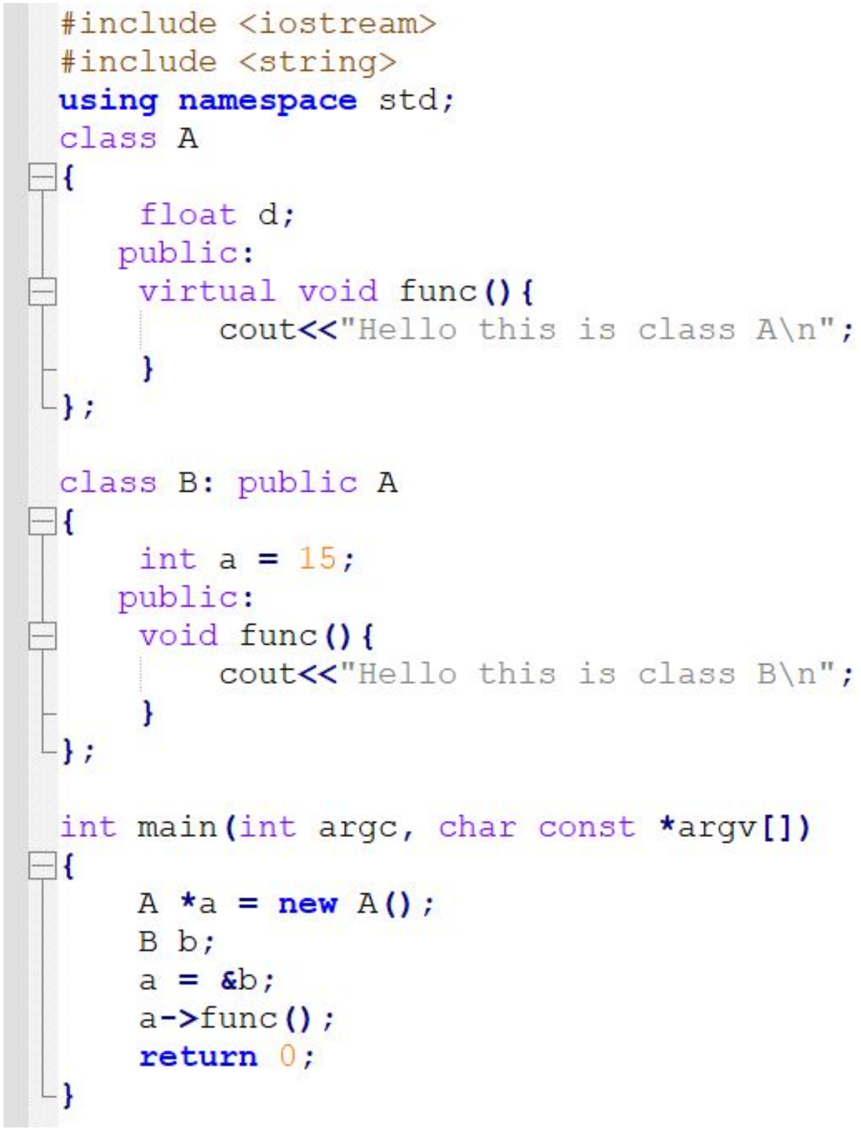
What will be the output of the following C++ code?



```
#include <iostream>
#include <string>
using namespace std;
class A
{
    float d;
     public:
    virtual void func(){
        cout<<"Hello this is class A\n";
    }
};

class B: public A
{
    int a = 15;
   public:
    void func(){
        cout<<"Hello this is class B\n";
    }
};

int main(int argc, char const *argv[])
{
    A *a = new A();
    a->func();
    return 0;
}
```

a) Hello this is class A
b) Hello this is class B
c) Error
d) Segmentation Fault

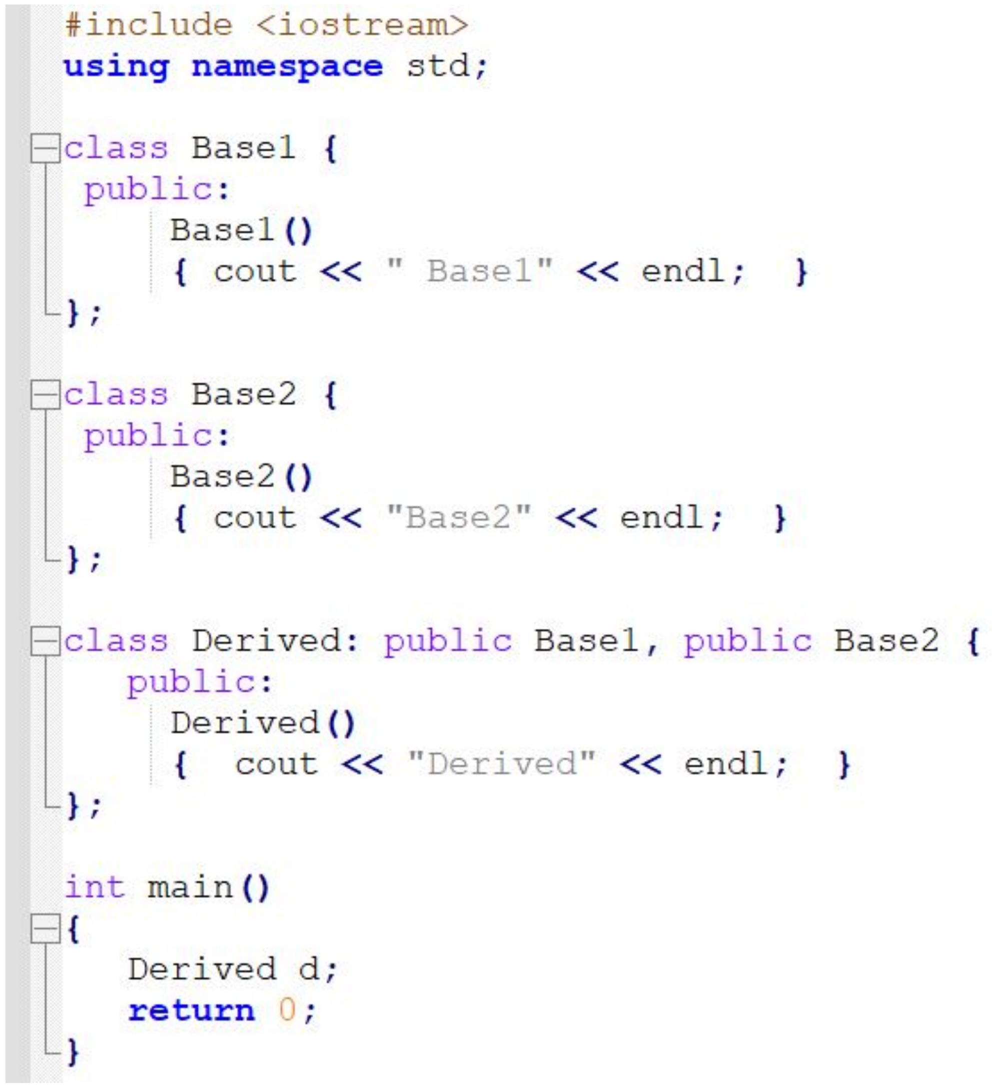
What will be the output of the following C++ code?



```cpp
#include <iostream>
#include <string>
using namespace std;
class A
{
    float d;
   public:
    virtual void func(){
        cout<<"Hello this is class A\n";
    }
};

class B: public A
{
    int a = 15;
   public:
    void func(){
        cout<<"Hello this is class B\n";
    }
};

int main(int argc, char const *argv[])
{
    A *a = new A();
    B b;
    a = &b;
    a->func();
    return 0;
}
```

a) Hello this is class A
b) Hello this is class B
c) Error
d) Segmentation Fault

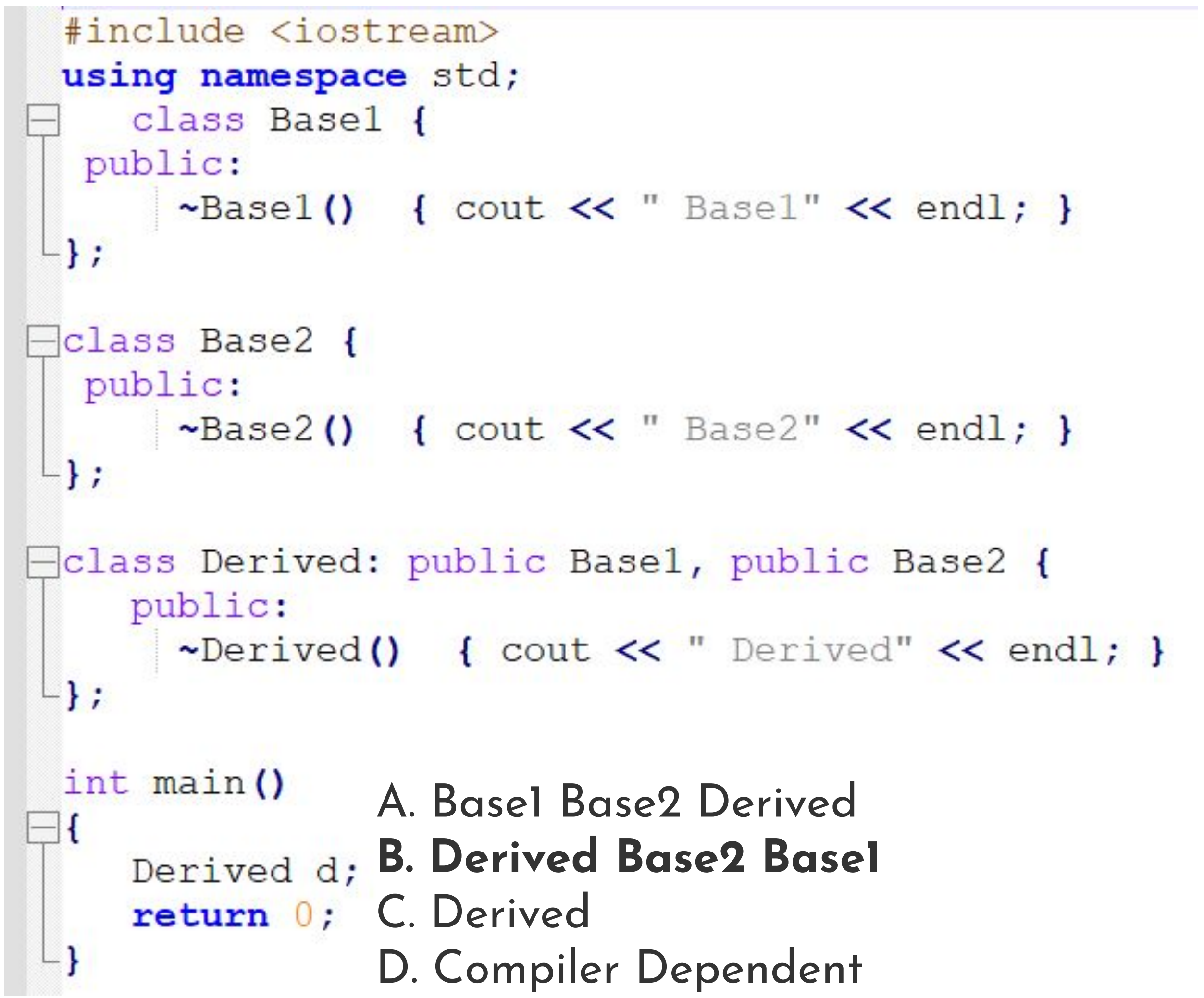
What will be the output of the following C++ code?



```
#include <iostream>
using namespace std;

class Base1 {
 public:
     Base1()
     { cout << " Base1" << endl;  }
};

class Base2 {
 public:
     Base2()
     { cout << "Base2" << endl;  }
};

class Derived: public Base1, public Base2 {
   public:
     Derived()
     {  cout << "Derived" << endl;  }
};

int main()
{
   Derived d;
   return 0;
}
```

A. Compiler Dependent
B. Base1 Base2 Derived
C. Base2 Base1 Derived
D. Compiler Error

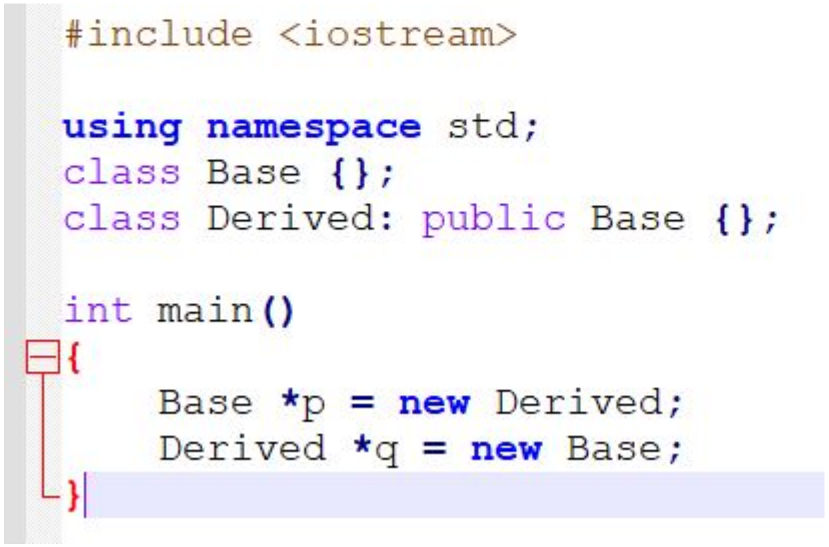
What will be the output of the following C++ code?



```
#include <iostream>
using namespace std;
   class Base1 {
 public:
     ~Base1()  { cout << " Base1" << endl; }
};

class Base2 {
 public:
     ~Base2()  { cout << " Base2" << endl; }
};

class Derived: public Base1, public Base2 {
   public:
     ~Derived()  { cout << " Derived" << endl; }
};

int main()
{
   Derived d;
   return 0;
}
```

A. Base1 Base2 Derived
B. Derived Base2 Base1
C. Derived
D. Compiler Dependent

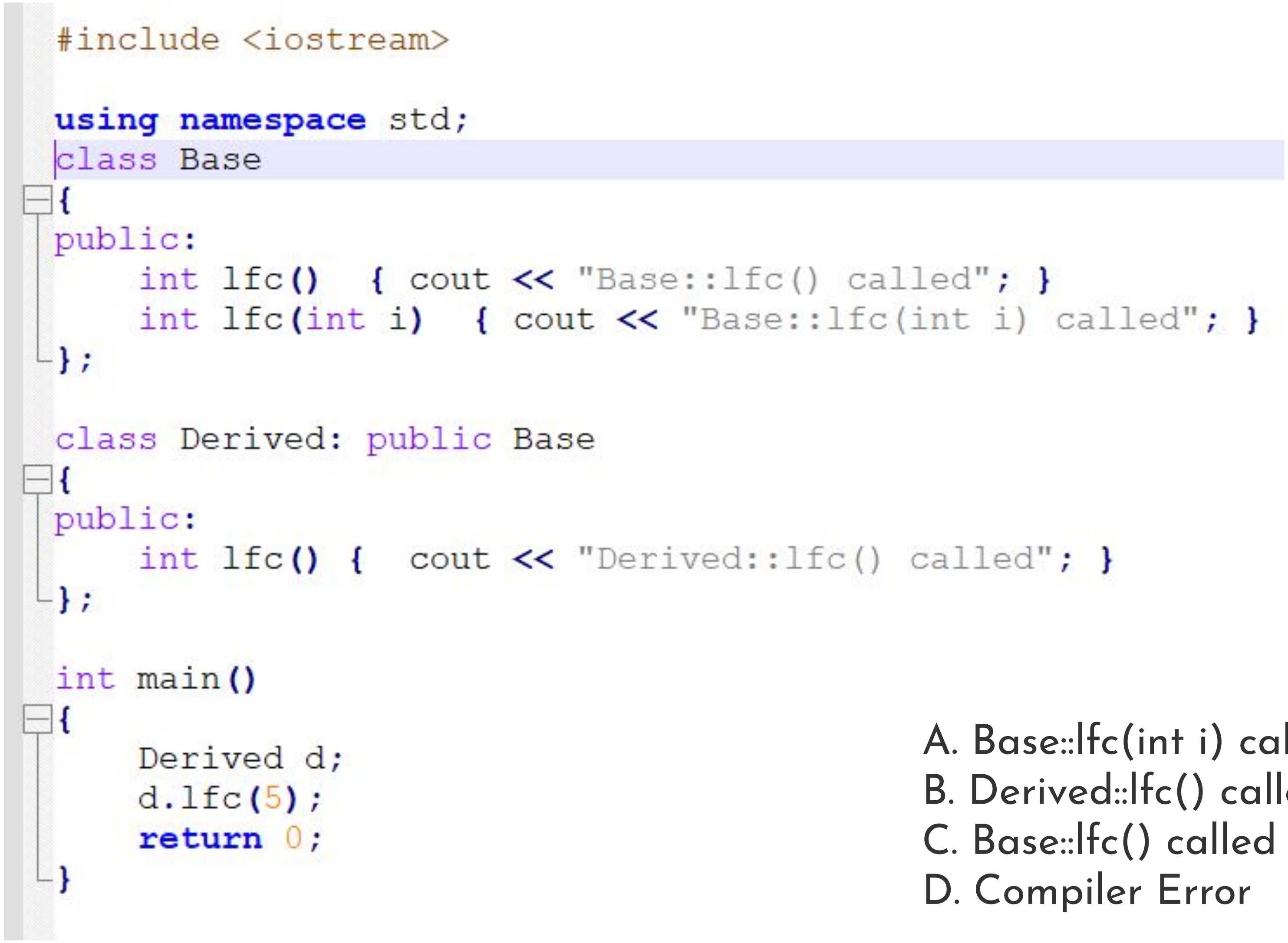
What will be the output of the following C++ code?



```
#include <iostream>

using namespace std;
class Base {};
class Derived: public Base {};

int main()
{
    Base *p = new Derived;
    Derived *q = new Base;
}
```

A. error: invalid conversion from "Derived*" to "Base*"
B. No Compiler Error
C. error: invalid conversion from "Base*" to "Derived*"
D. Runtime Error

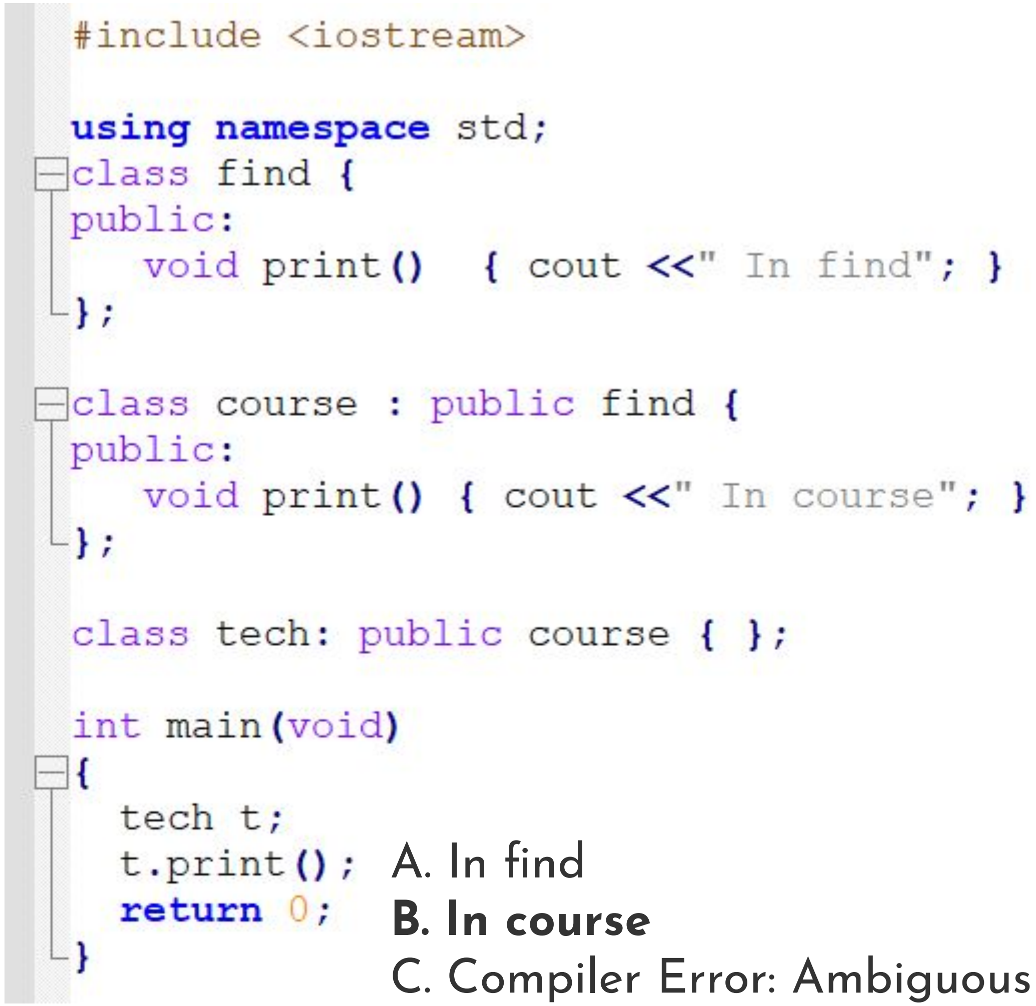
What will be the output of the following C++ code?



```
#include <iostream>

using namespace std;
class Base
{
public:
    int lfc()  { cout << "Base::lfc() called"; }
    int lfc(int i)  { cout << "Base::lfc(int i) called"; }
};

class Derived: public Base
{
public:
    int lfc() {  cout << "Derived::lfc() called"; }
};

int main()
{
    Derived d;
    d.lfc(5);
    return 0;
}
```

A. Base::lfc(int i) called
B. Derived::lfc() called
C. Base::lfc() called
D. Compiler Error

What will be the output of the following C++ code?



```
#include <iostream>

using namespace std;
class find {
public:
    void print()   { cout <<" In find"; }
};

class course : public find {
public:
    void print() { cout <<" In course"; }
};

class tech: public course { };

int main(void)
{
  tech t;
  t.print();
  return 0;
}
```

A. In find
B. In course
C. Compiler Error: Ambiguous call to print()
D. None of the above