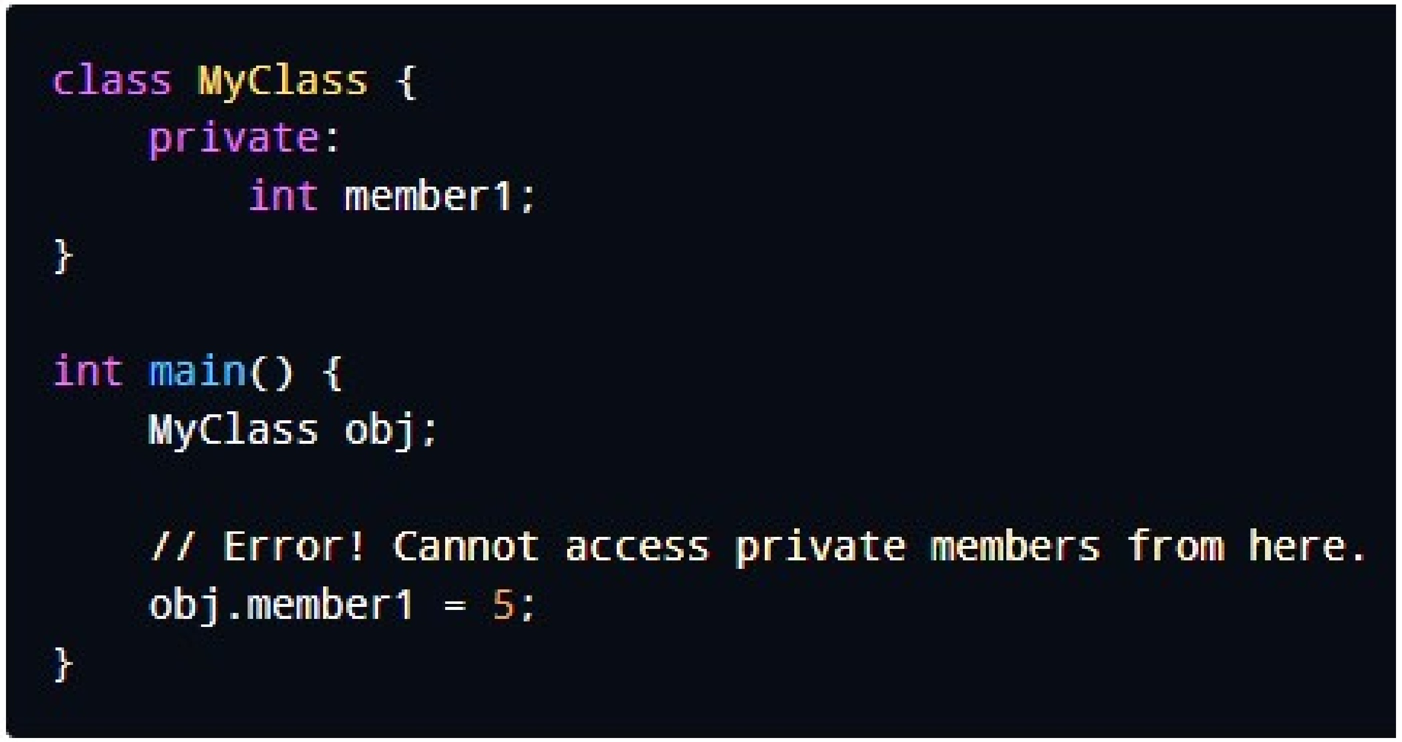
C++ FRIEND FUNCTION AND FRIEND CLASSES

- Data hiding is a fundamental concept of object-oriented programming. It restricts the access of private members from outside of the class.
- Similarly, protected members can only be accessed by derived classes and are inaccessible from outside. For example,

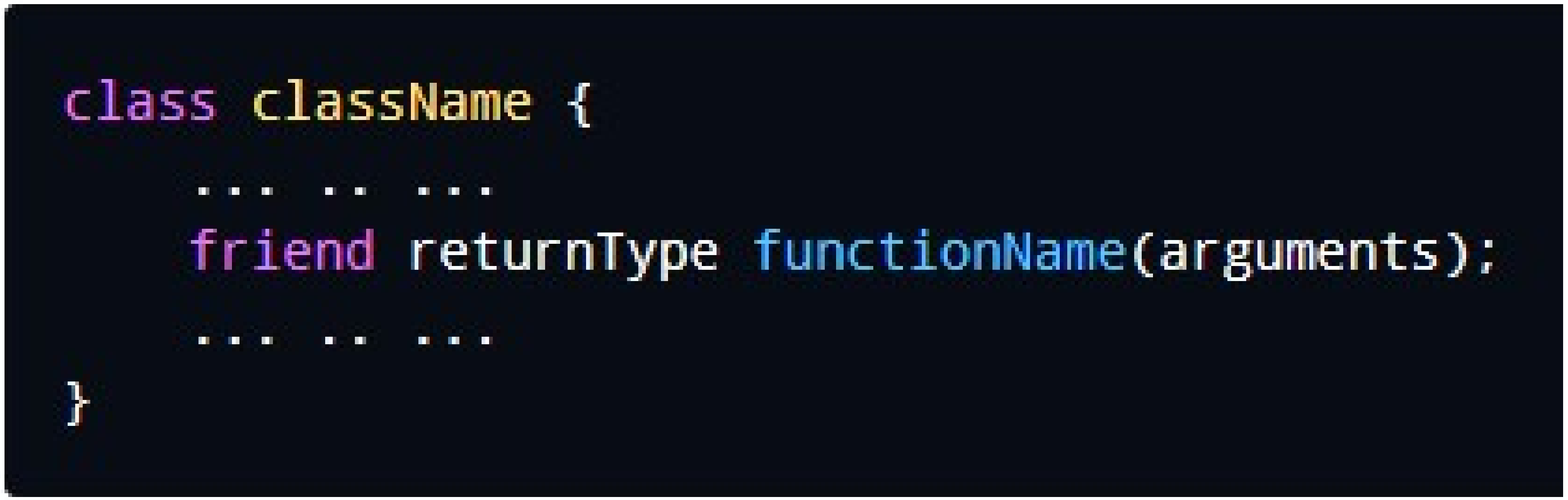


```cpp
class MyClass {
    private:
        int member1;
}

int main() {
    MyClass obj;

    // Error! Cannot access private members from here.
    obj.member1 = 5;
}
```

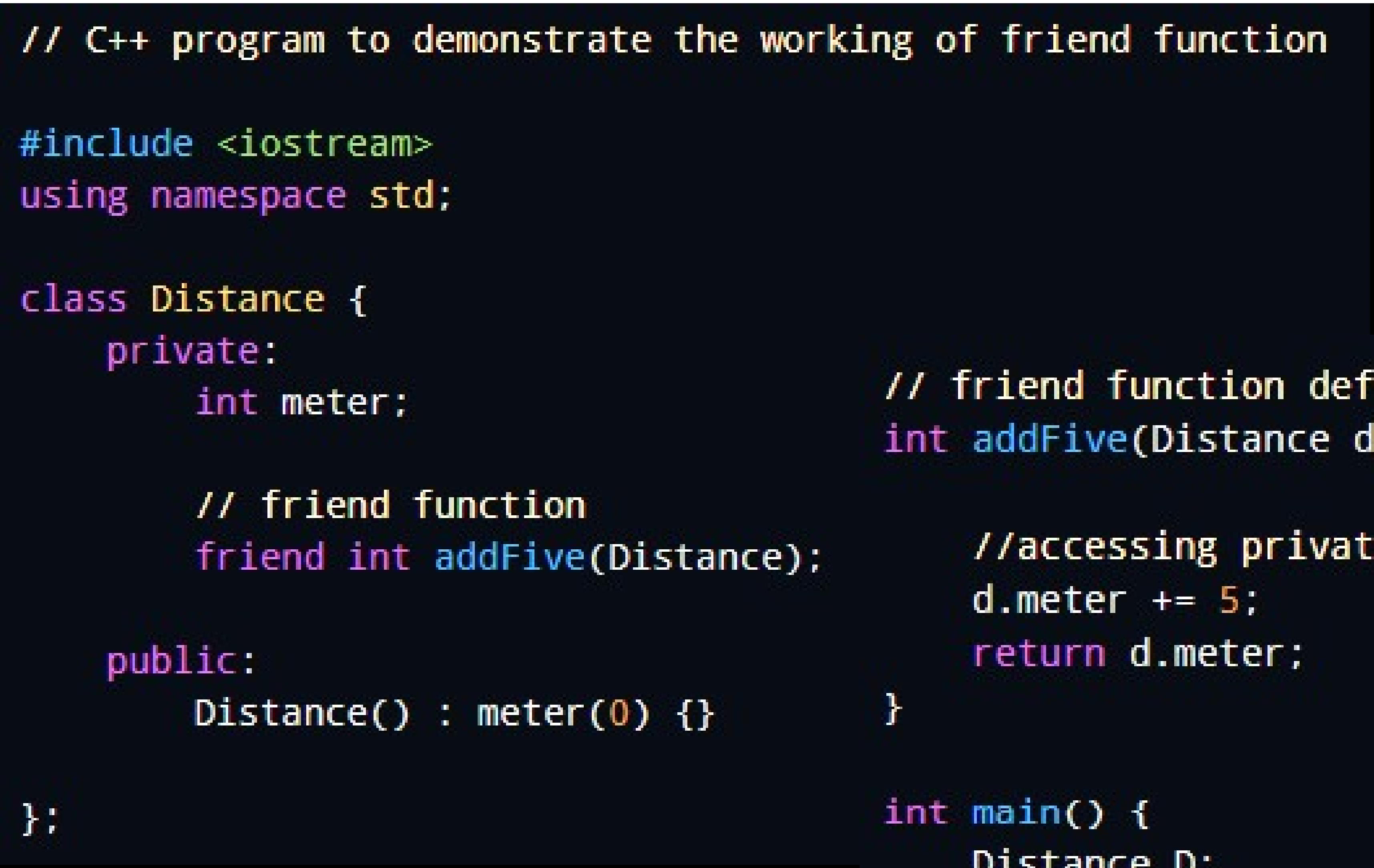
- However, there is a feature in C++ called **friend functions** that break this rule and allow us to access member functions from outside the class.
- Similarly, there is a **friend class** as well, which we will learn later in this tutorial.

FRIEND FUNCTION IN C++

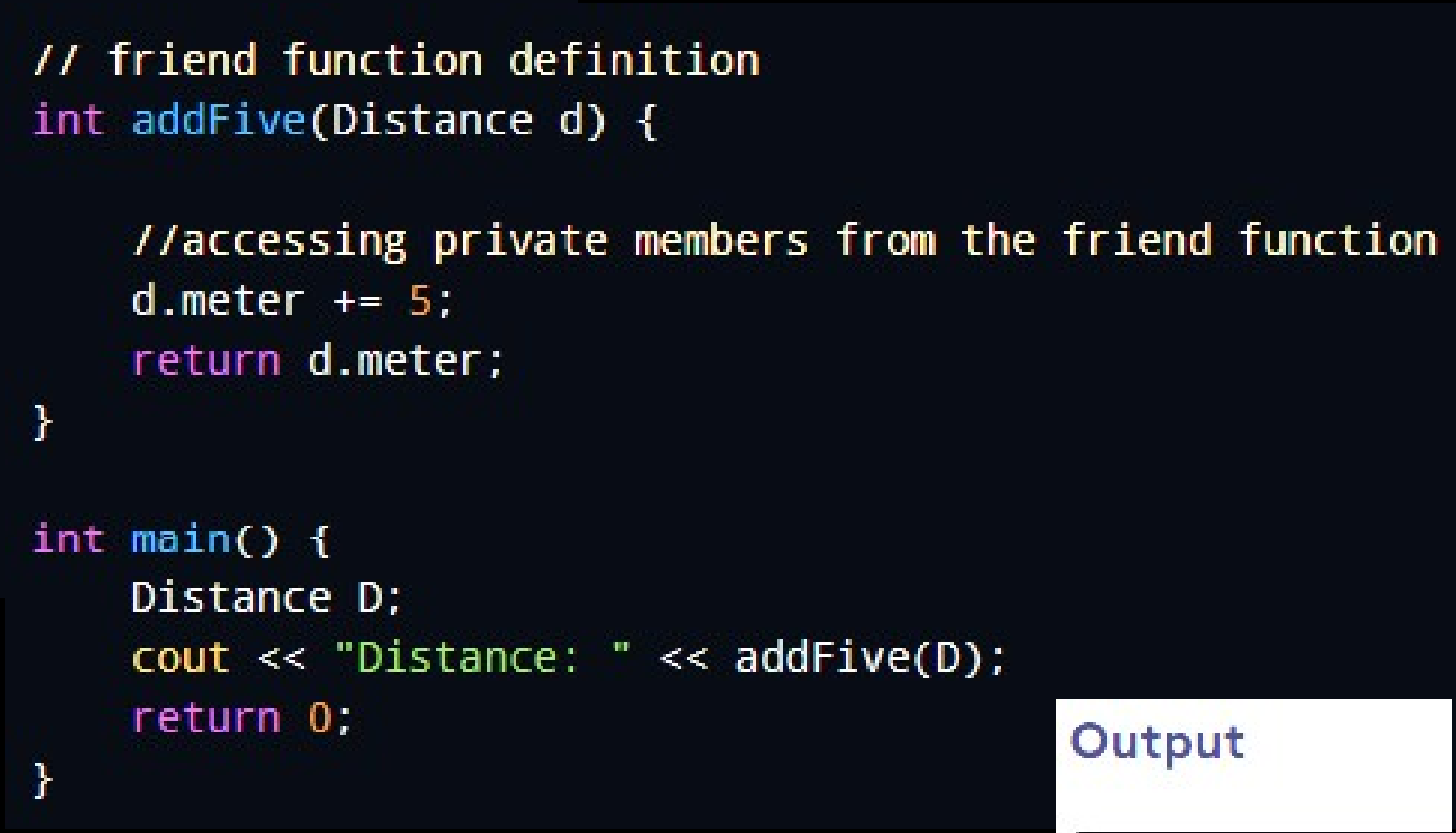
- A friend function can access the private and protected data of a class. We declare a friend function using the friend keyword inside the body of the class.



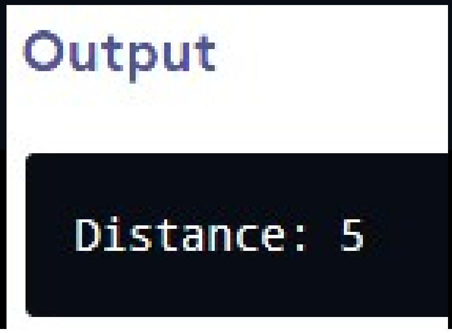
```
class className {
    ... .. ...
    friend returnType functionName(arguments);
    ... .. ...
}
```



```cpp
// C++ program to demonstrate the working of friend function

#include <iostream>
using namespace std;

class Distance {
    private:
        int meter;

        // friend function
        friend int addFive(Distance);

    public:
        Distance() : meter(0) {}

};
```



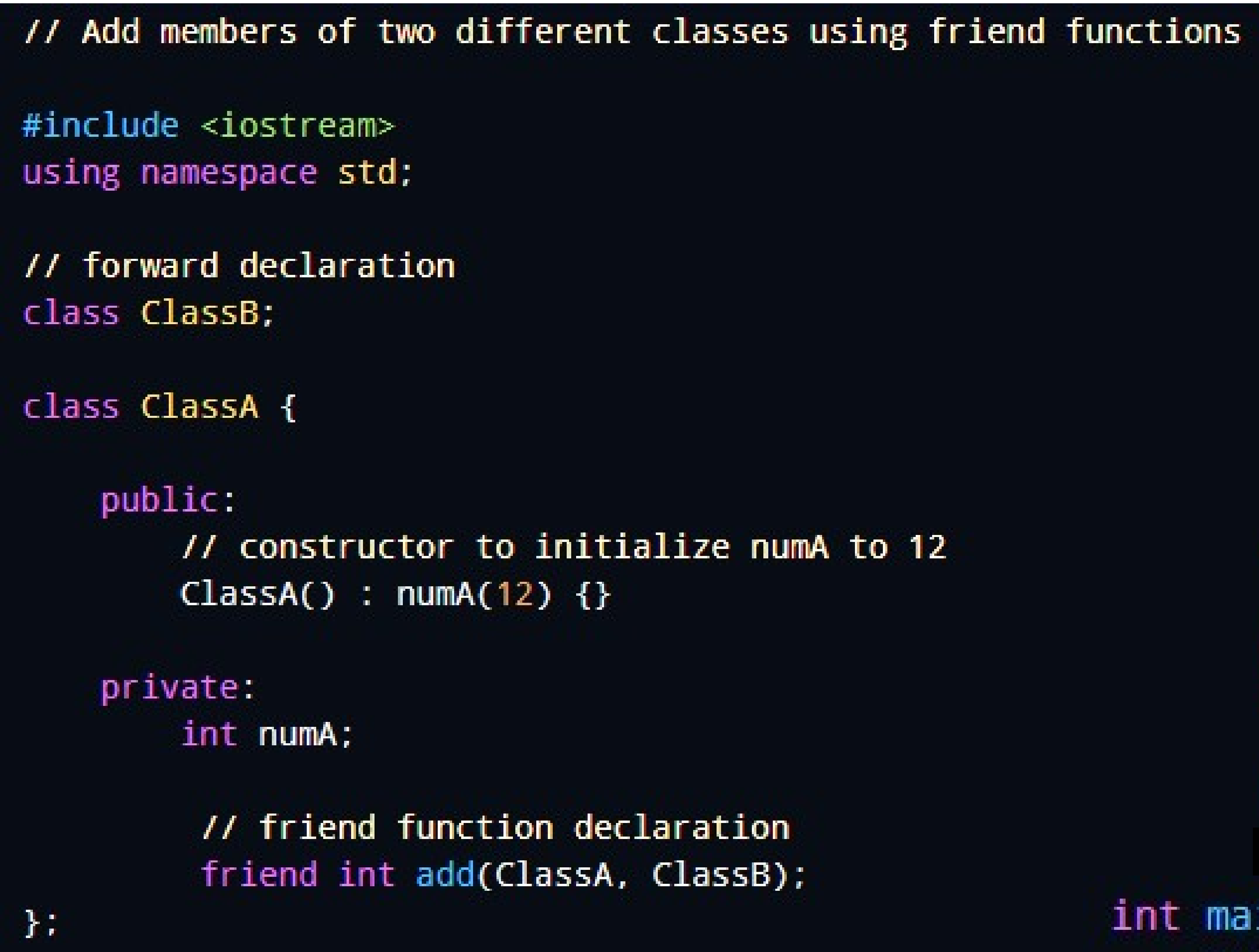
```cpp
// friend function definition
int addFive(Distance d) {

    //accessing private members from the friend function
    d.meter += 5;
    return d.meter;
}

int main() {
    Distance D;
    cout << "Distance: " << addFive(D);
    return 0;
}
```



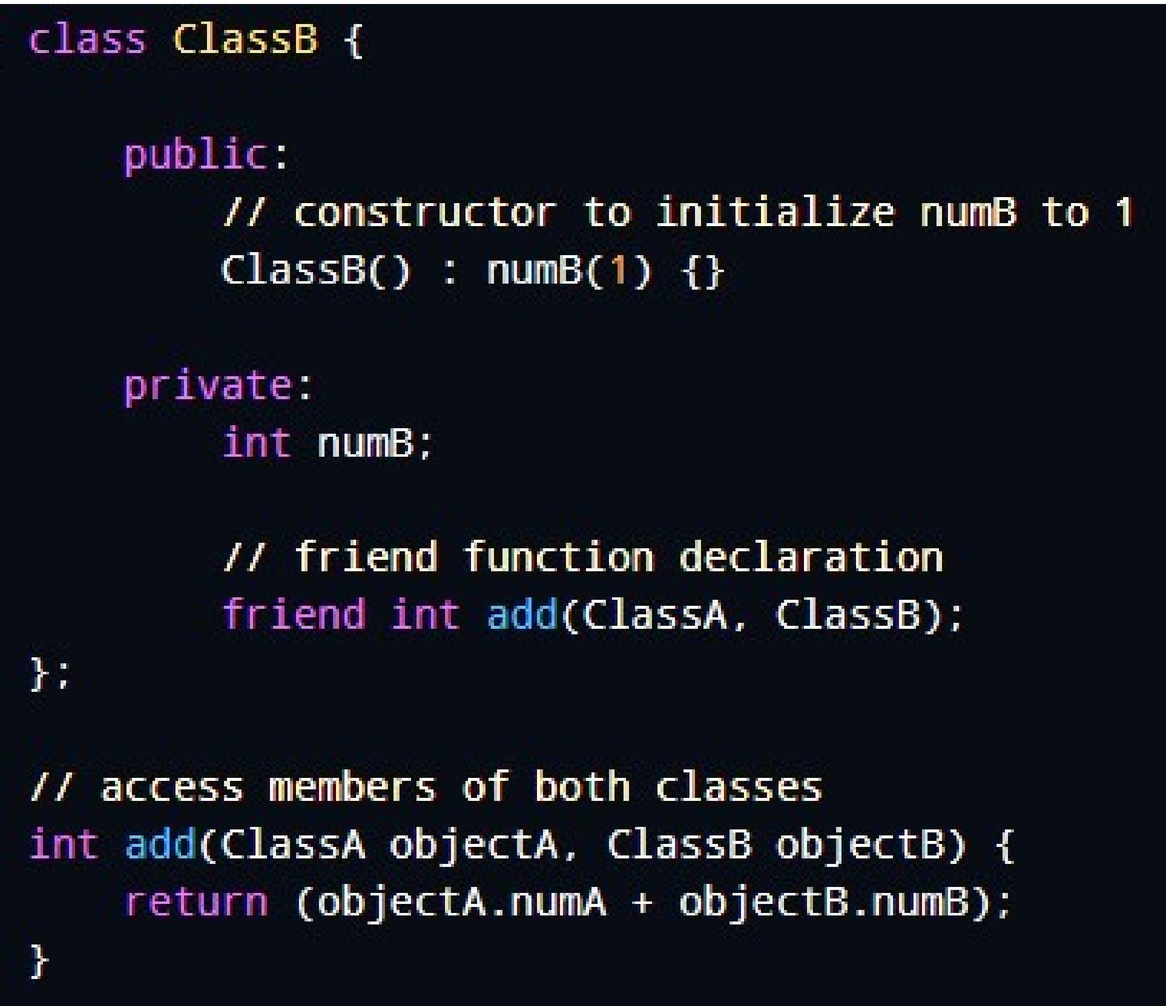
Output

```
Distance: 5
```

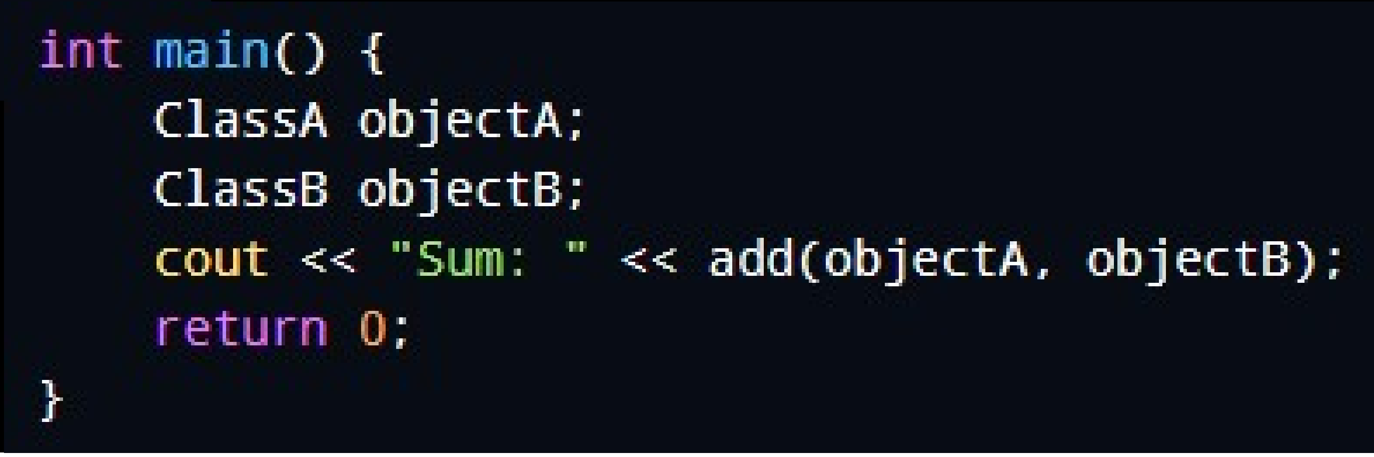
- Here, addFive() is a friend function that can access both private and public data members.
- Though this example gives us an idea about the concept of a friend function, it doesn't show any meaningful use.
- A more meaningful use would be operating on objects of two different classes.
- That's when the friend function can be very helpful.



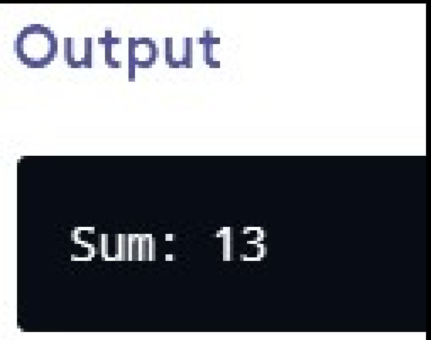
```cpp
// Add members of two different classes using friend functions

#include <iostream>
using namespace std;

// forward declaration
class ClassB;

class ClassA {

    public:
        // constructor to initialize numA to 12
        ClassA() : numA(12) {}

    private:
        int numA;

         // friend function declaration
         friend int add(ClassA, ClassB);
};
```



```cpp
class ClassB {

    public:
        // constructor to initialize numB to 1
        ClassB() : numB(1) {}

    private:
        int numB;

        // friend function declaration
        friend int add(ClassA, ClassB);
};

// access members of both classes
int add(ClassA objectA, ClassB objectB) {
    return (objectA.numA + objectB.numB);
}
```



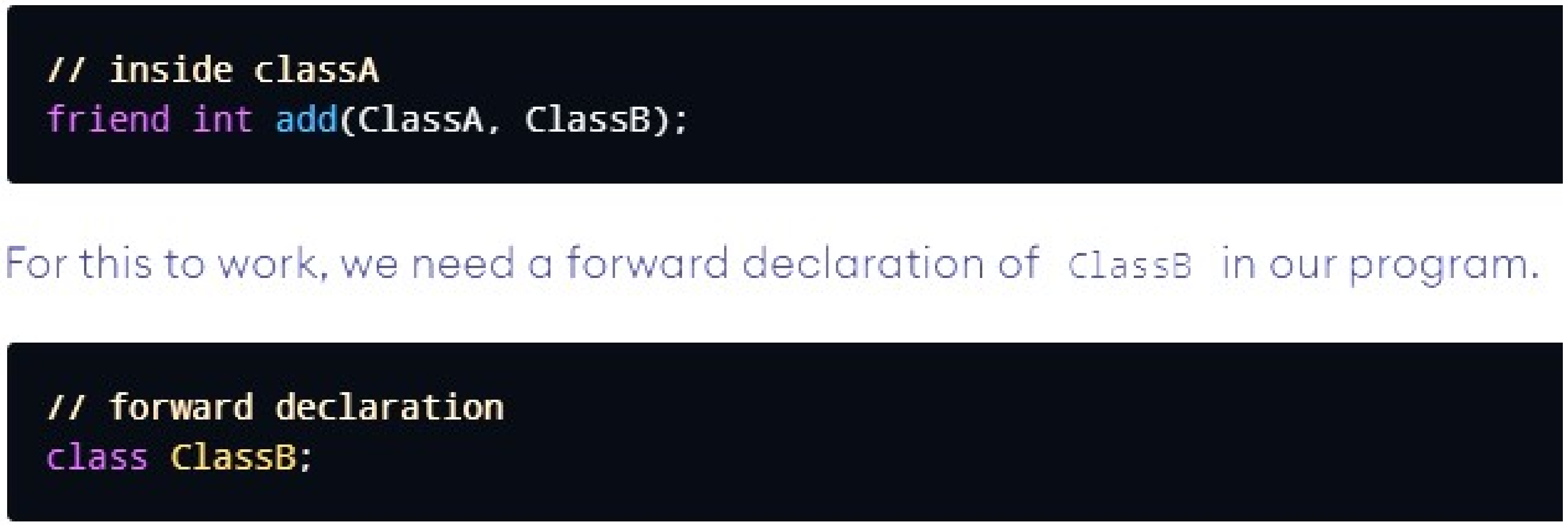
```cpp
int main() {
    ClassA objectA;
    ClassB objectB;
    cout << "Sum: " << add(objectA, objectB);
    return 0;
}
```



Output

```
Sum: 13
```

- In this program, ClassA and ClassB have declared add() as a friend function. Thus, this function can access private data of both classes.
- One thing to notice here is the friend function inside ClassA is using the ClassB. However, we haven't defined ClassB at this point.



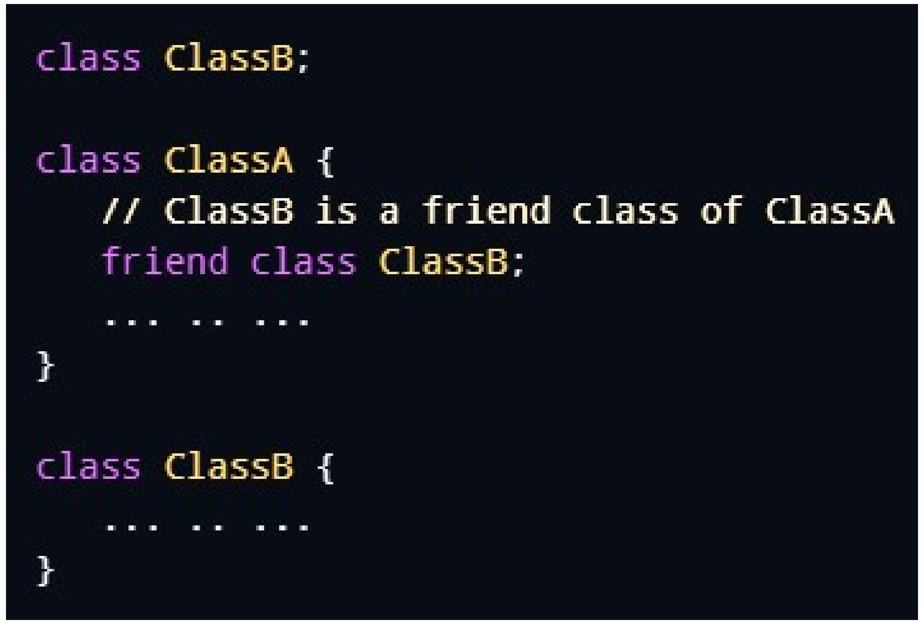
```cpp
// inside classA
friend int add(ClassA, ClassB);
```

For this to work, we need a forward declaration of `ClassB` in our program.

```cpp
// forward declaration
class ClassB;
```

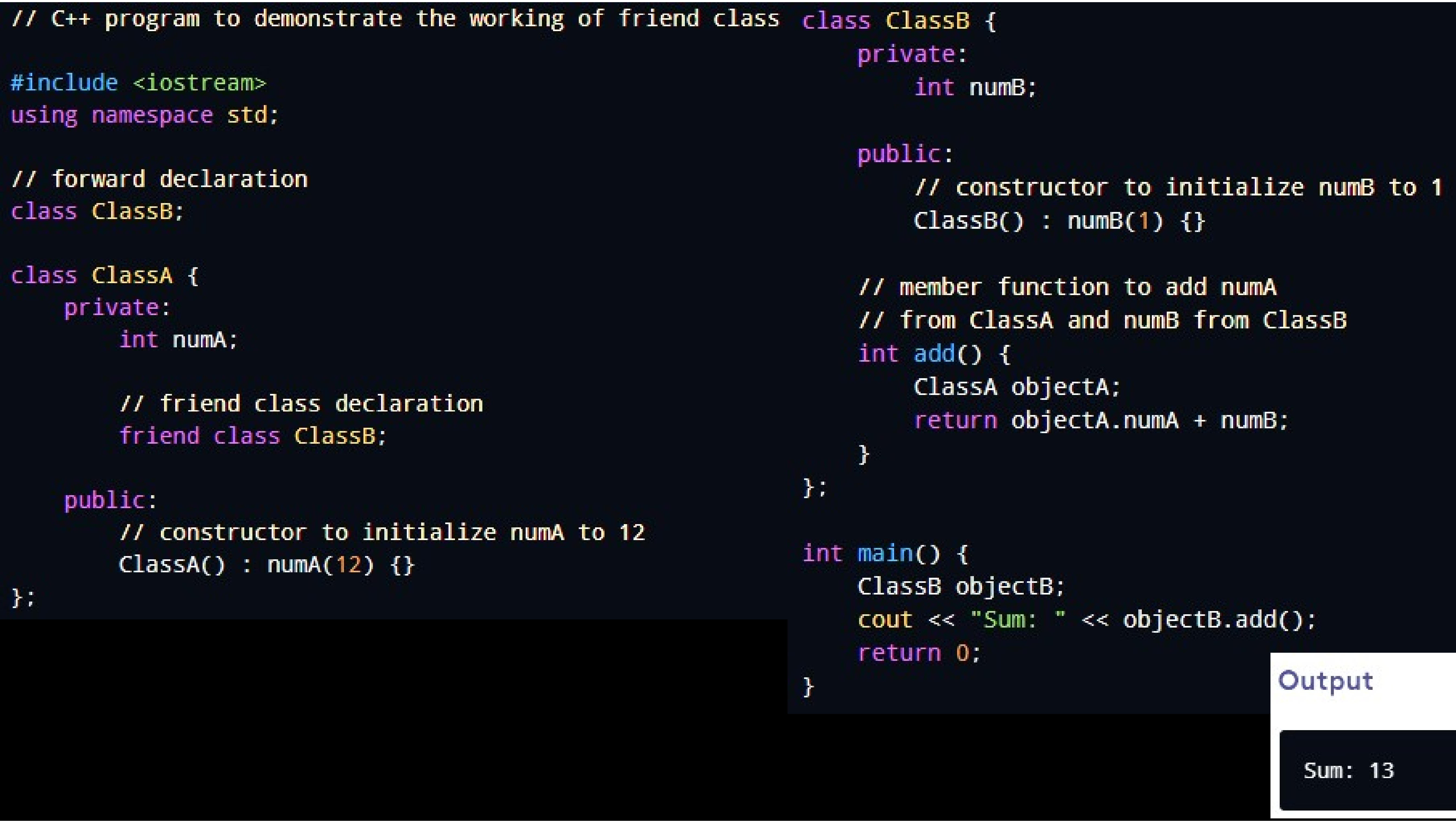
FRIEND CLASS IN C++

- We can also use a friend Class in C++ using the friend keyword. For example,



```
class ClassB;

class ClassA {
   // ClassB is a friend class of ClassA
   friend class ClassB;
   ... .. ...
}

class ClassB {
   ... .. ...
}
```

- When a class is declared a friend class, all the member functions of the friend class become friend functions.
- Since ClassB is a friend class, we can access all members of ClassA from inside ClassB.
- However, we cannot access members of ClassB from inside ClassA.
- It is because friend relation in C++ is only granted, not taken.



```cpp
// C++ program to demonstrate the working of friend class

#include <iostream>
using namespace std;

// forward declaration
class ClassB;

class ClassA {
    private:
        int numA;

        // friend class declaration
        friend class ClassB;

    public:
        // constructor to initialize numA to 12
        ClassA() : numA(12) {}
};
```

```cpp
class ClassB {
    private:
        int numB;

    public:
        // constructor to initialize numB to 1
        ClassB() : numB(1) {}

    // member function to add numA
    // from ClassA and numB from ClassB
    int add() {
        ClassA objectA;
        return objectA.numA + numB;
    }
};

int main() {
    ClassB objectB;
    cout << "Sum: " << objectB.add();
    return 0;
}
```

Output

```
Sum: 13
```

- Here, ClassB is a friend class of ClassA.
- So, ClassB has access to the members of classA.
- In ClassB, we have created a function add() that returns the sum of numA and numB.
- Since ClassB is a friend class,
- we can create objects of ClassA inside of ClassB.