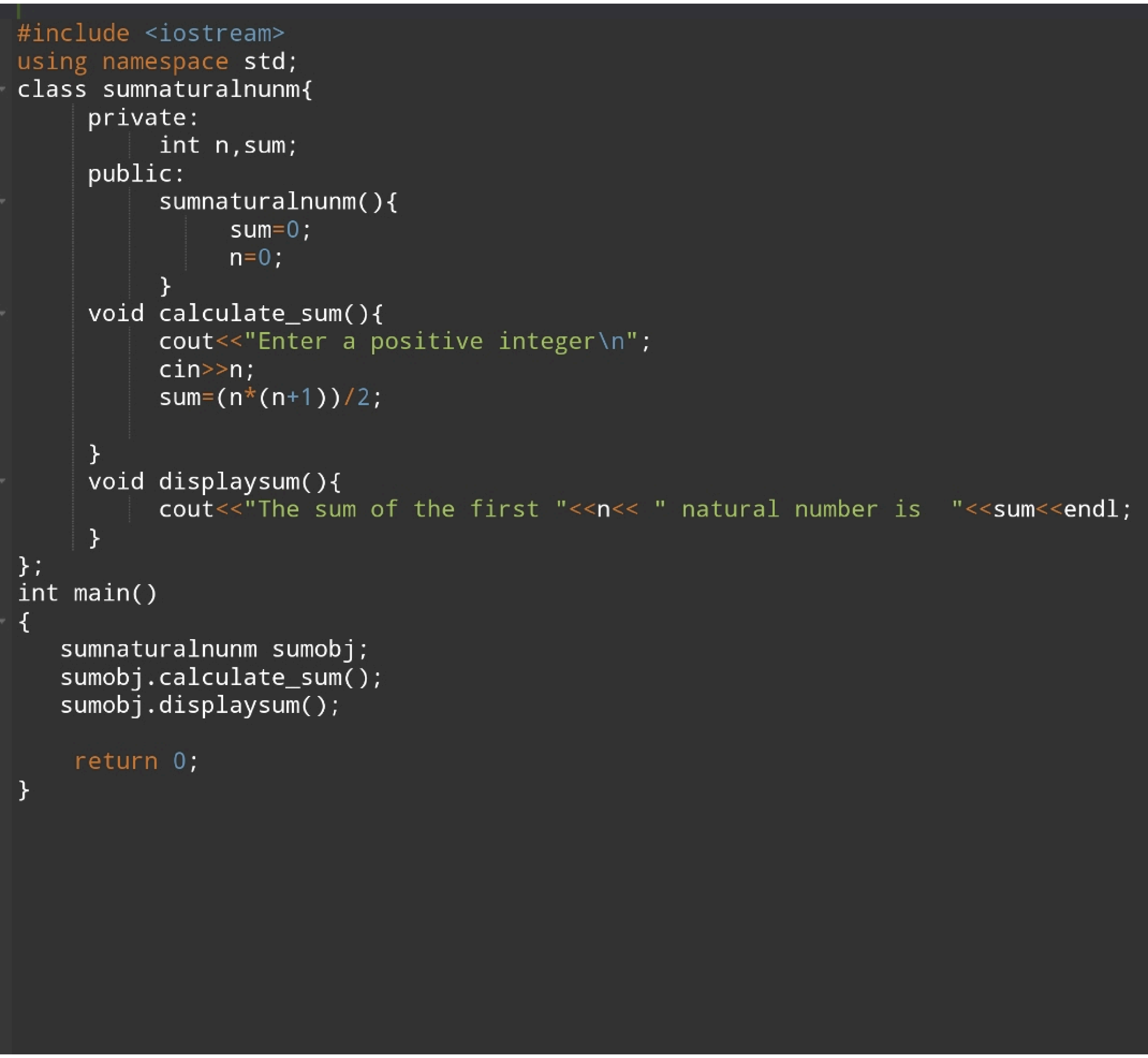
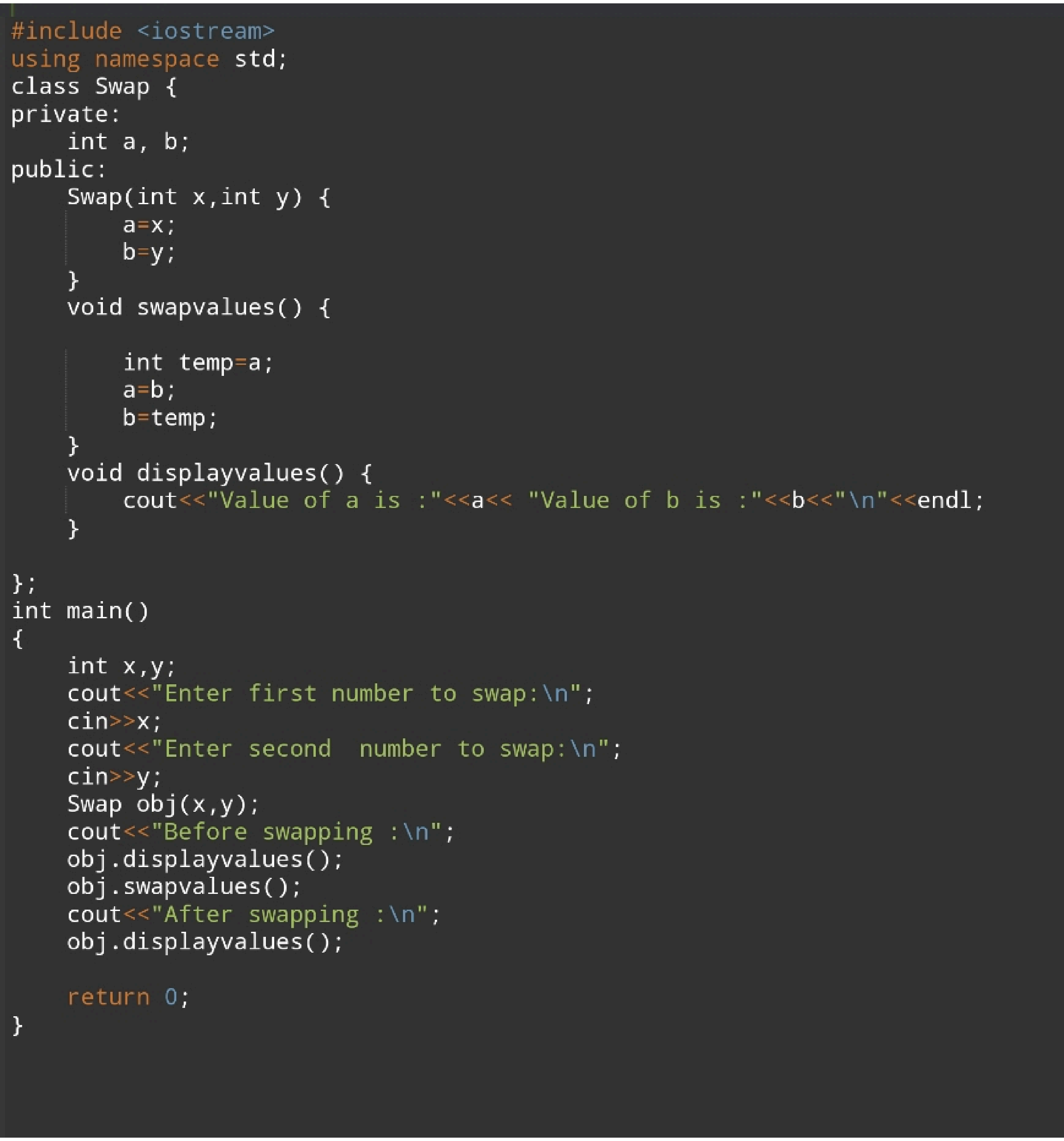
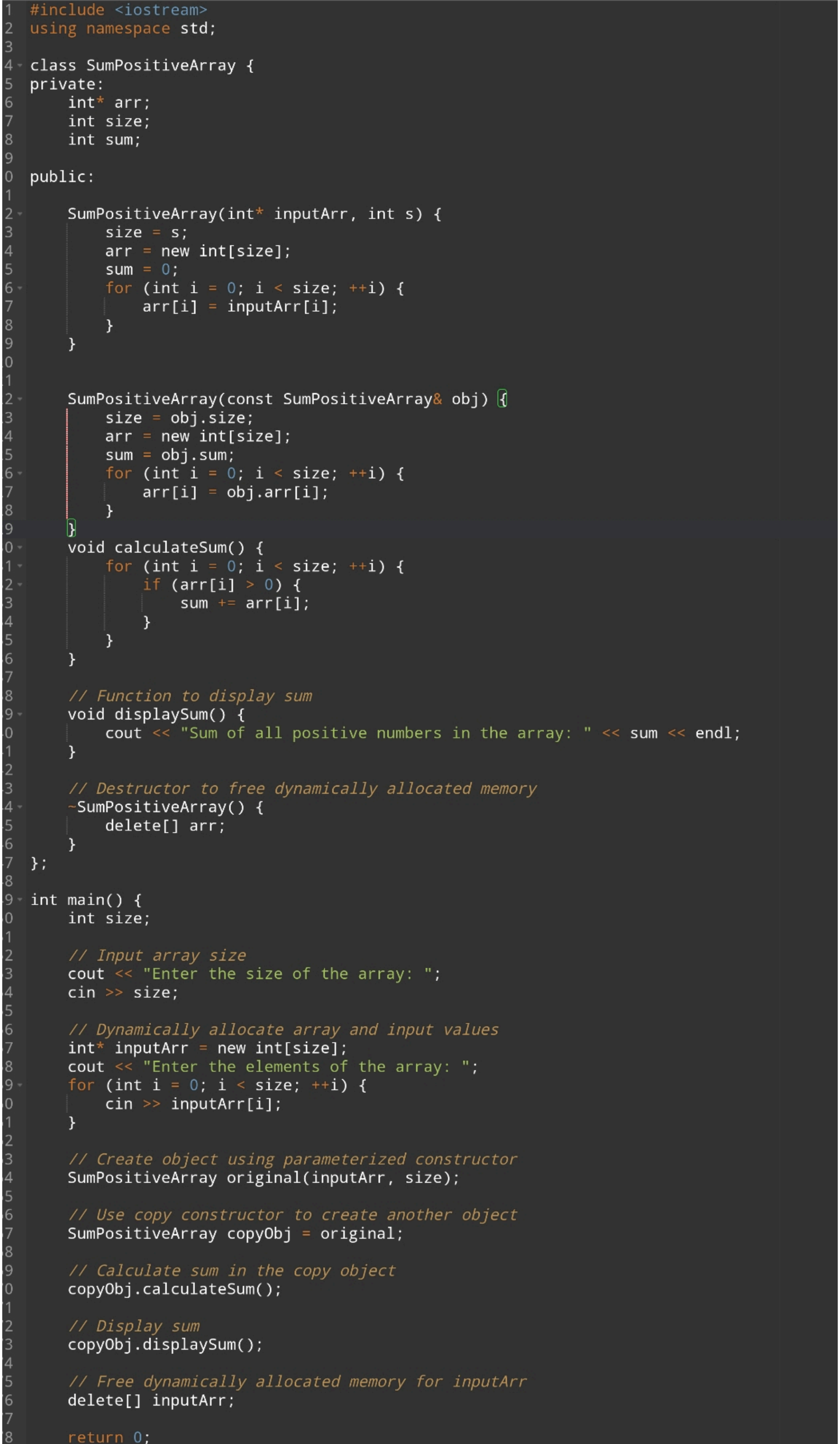


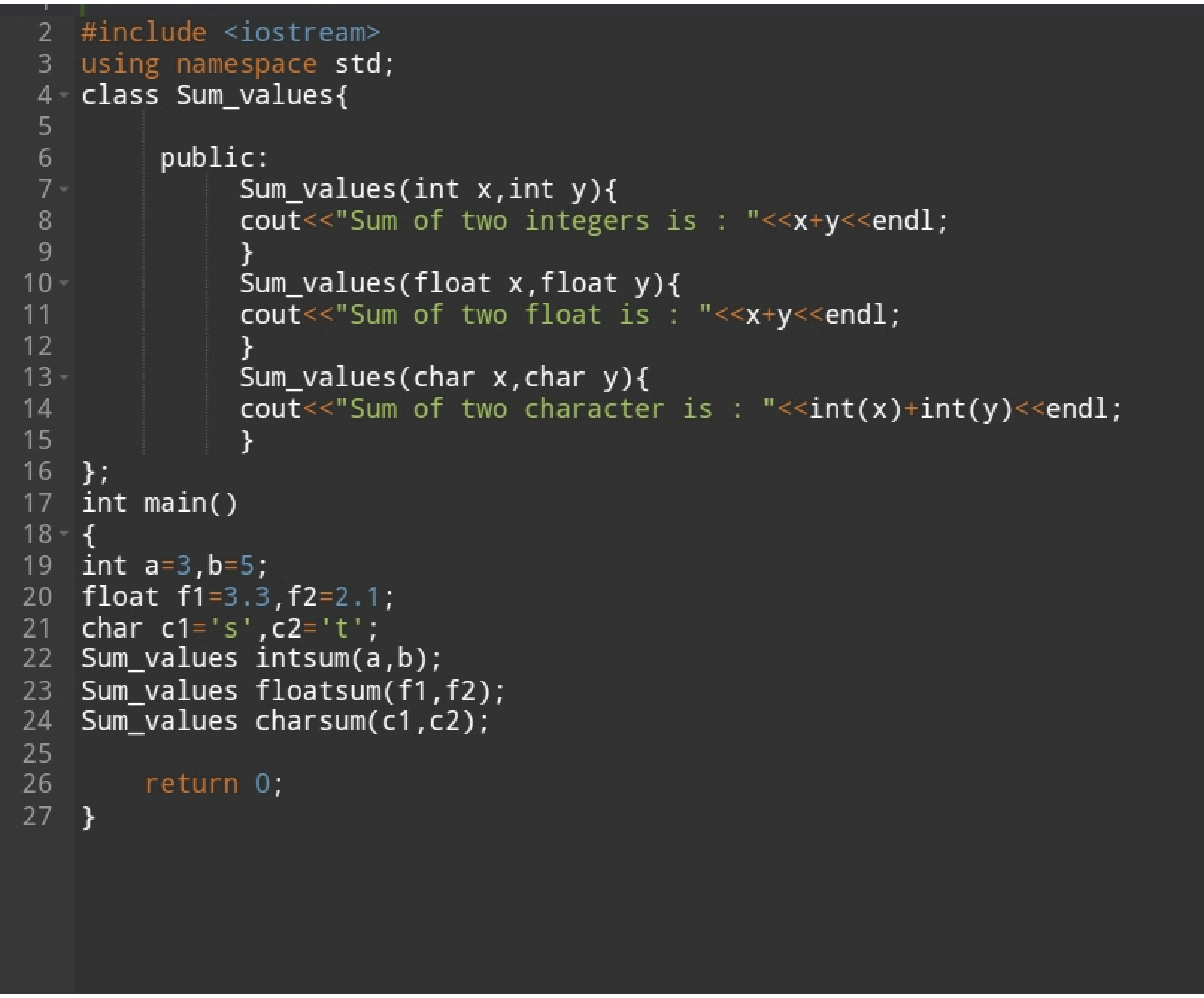
```cpp
#include <iostream>
using namespace std;
class sumnaturalnunm{
     private:
          int n,sum;
     public:
          sumnaturalnunm(){
               sum=0;
               n=0;
          }
     void calculate_sum(){
          cout<<"Enter a positive integer\n";
          cin>>n;
          sum=(n*(n+1))/2;

     }
     void displaysum(){
          cout<<"The sum of the first "<<n<< " natural number is  "<<sum<<endl;
     }
};
int main()
{
   sumnaturalnunm sumobj;
   sumobj.calculate_sum();
   sumobj.displaysum();

    return 0;
}
```



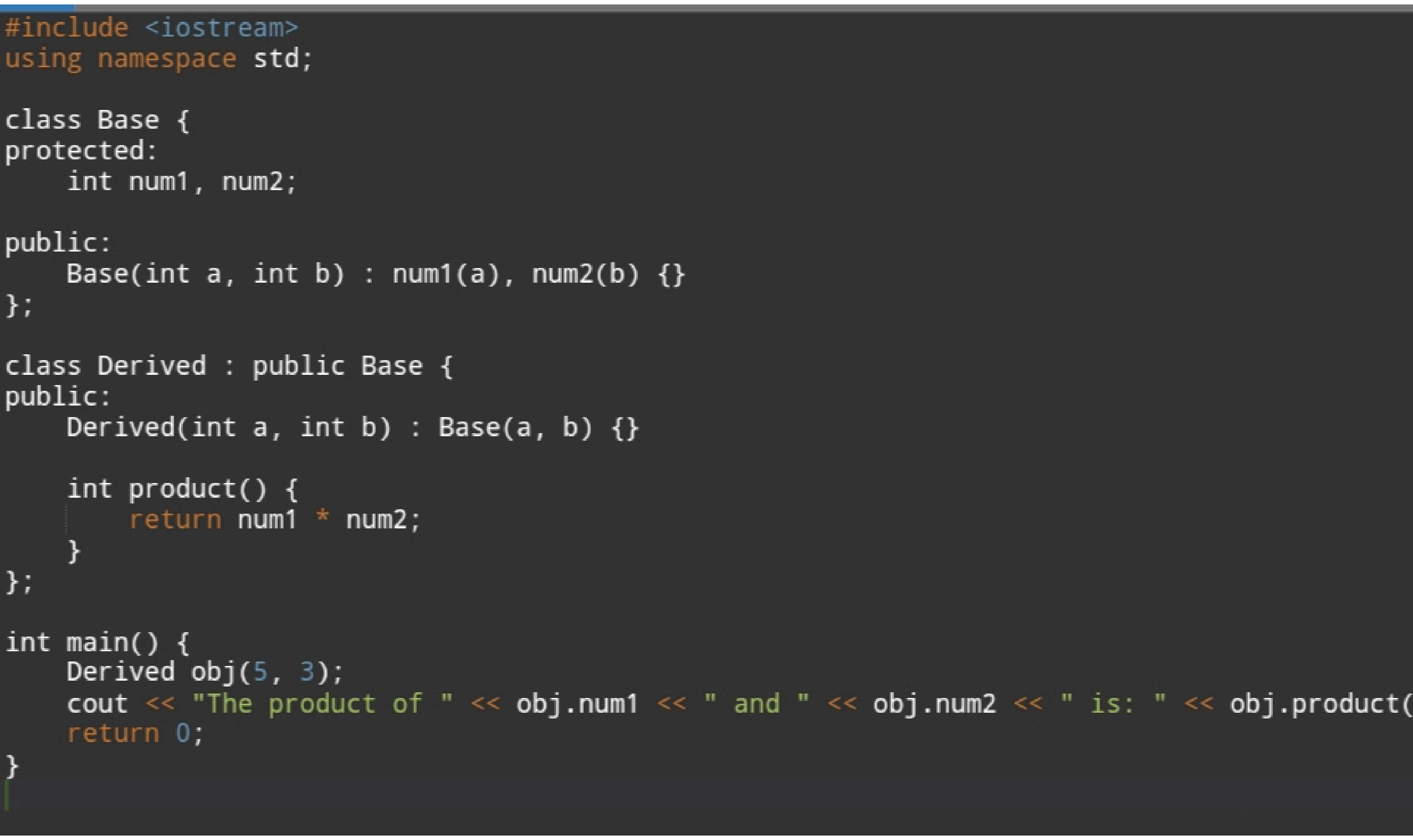
```cpp
#include <iostream>
using namespace std;
class Swap {
private:
    int a, b;
public:
    Swap(int x,int y) {
        a=x;
        b=y;
    }
    void swapvalues() {

        int temp=a;
        a=b;
        b=temp;
    }
    void displayvalues() {
        cout<<"Value of a is :"<<a<< "Value of b is :"<<b<<"\n"<<endl;
    }

};
int main()
{
    int x,y;
    cout<<"Enter first number to swap:\n";
    cin>>x;
    cout<<"Enter second  number to swap:\n";
    cin>>y;
    Swap obj(x,y);
    cout<<"Before swapping :\n";
    obj.displayvalues();
    obj.swapvalues();
    cout<<"After swapping :\n";
    obj.displayvalues();

    return 0;
}
```



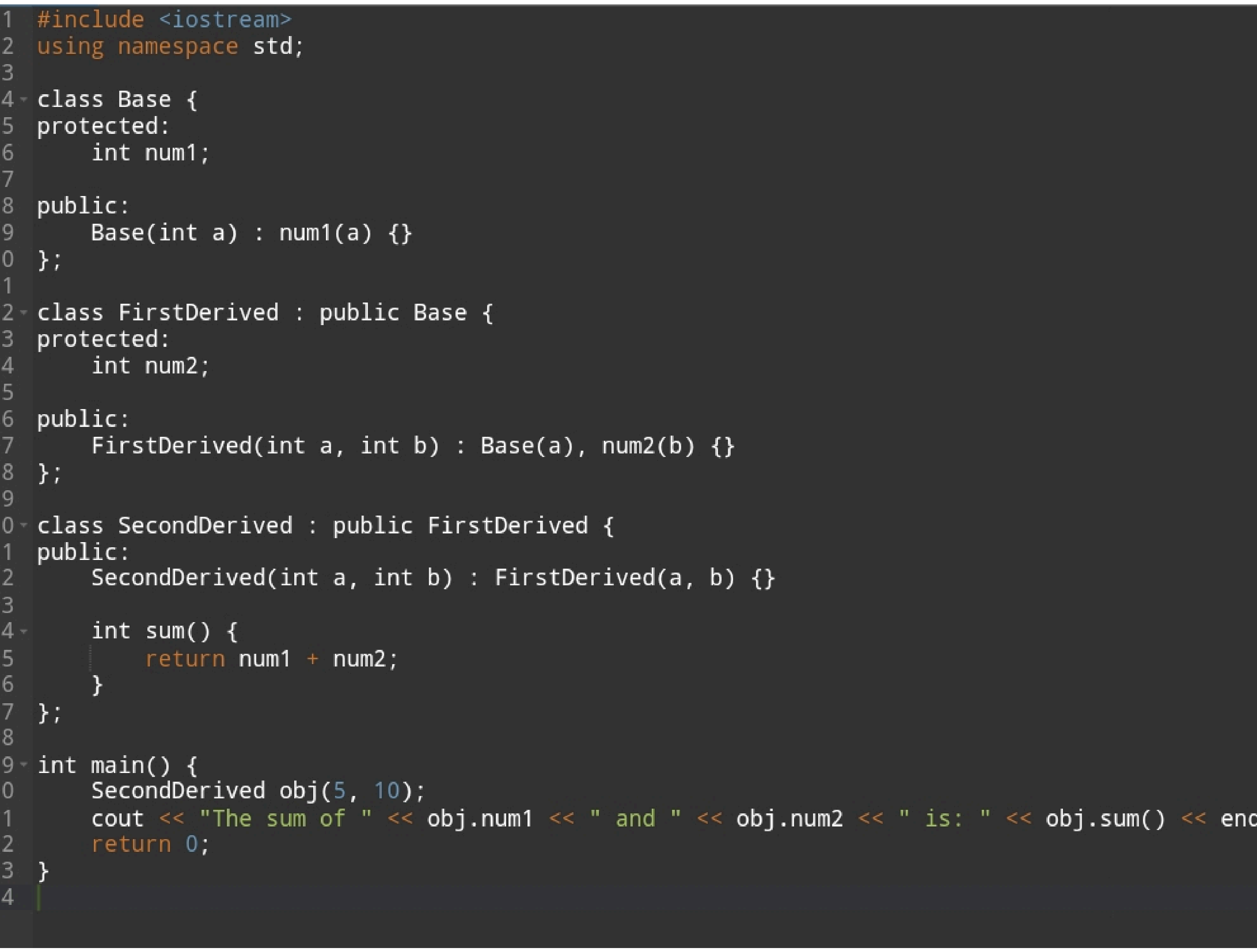
```cpp
#include <iostream>
using namespace std;

class SumPositiveArray {
private:
    int* arr;
    int size;
    int sum;

public:

    SumPositiveArray(int* inputArr, int s) {
        size = s;
        arr = new int[size];
        sum = 0;
        for (int i = 0; i < size; ++i) {
            arr[i] = inputArr[i];
        }
    }


    SumPositiveArray(const SumPositiveArray& obj) {
        size = obj.size;
        arr = new int[size];
        sum = obj.sum;
        for (int i = 0; i < size; ++i) {
            arr[i] = obj.arr[i];
        }
    }
    void calculateSum() {
        for (int i = 0; i < size; ++i) {
            if (arr[i] > 0) {
                sum += arr[i];
            }
        }
    }

    // Function to display sum
    void displaySum() {
        cout << "Sum of all positive numbers in the array: " << sum << endl;
    }

    // Destructor to free dynamically allocated memory
    ~SumPositiveArray() {
        delete[] arr;
    }
};

int main() {
    int size;

    // Input array size
    cout << "Enter the size of the array: ";
    cin >> size;

    // Dynamically allocate array and input values
    int* inputArr = new int[size];
    cout << "Enter the elements of the array: ";
    for (int i = 0; i < size; ++i) {
        cin >> inputArr[i];
    }

    // Create object using parameterized constructor
    SumPositiveArray original(inputArr, size);

    // Use copy constructor to create another object
    SumPositiveArray copyObj = original;

    // Calculate sum in the copy object
    copyObj.calculateSum();

    // Display sum
    copyObj.displaySum();

    // Free dynamically allocated memory for inputArr
    delete[] inputArr;

    return 0;
```



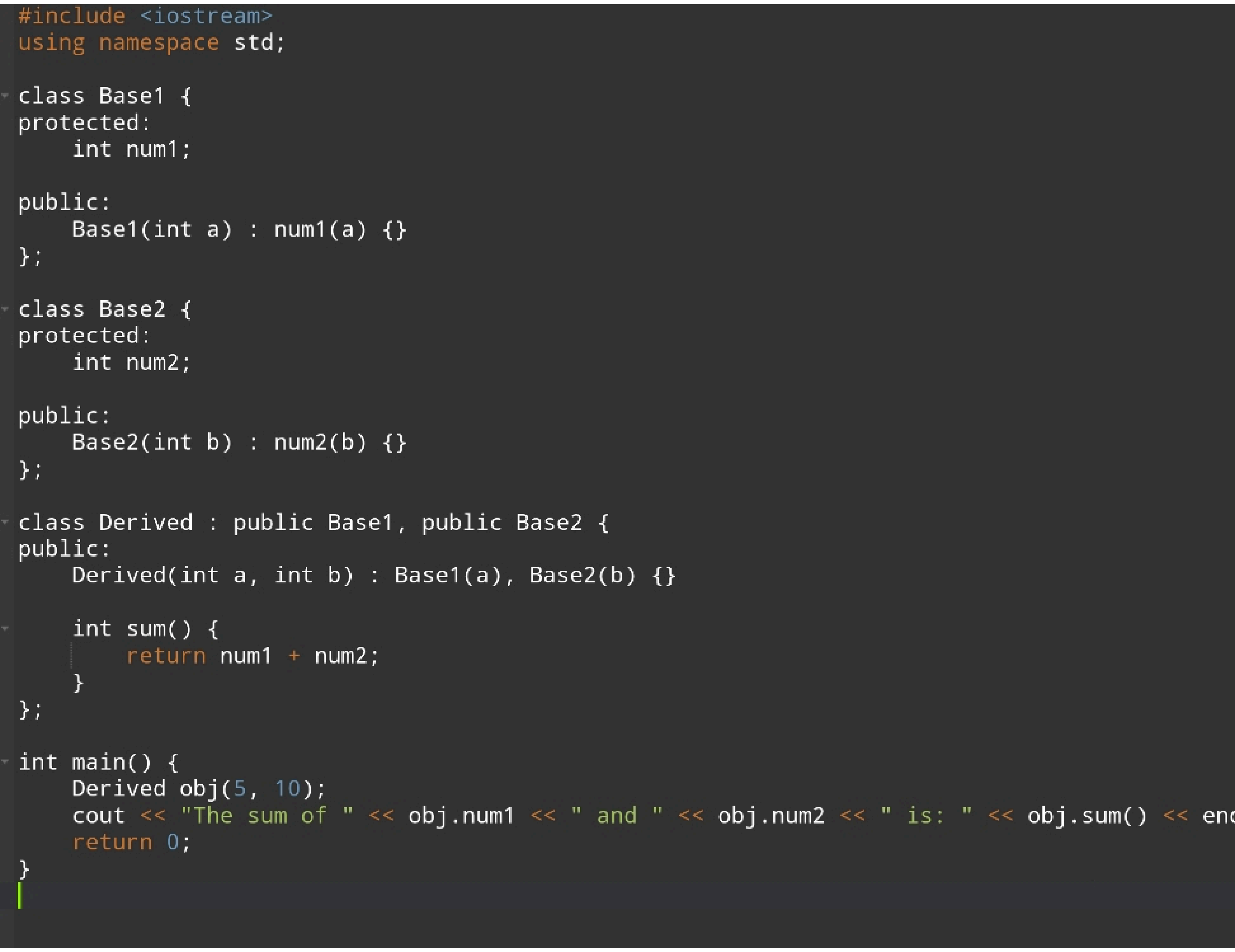
```cpp
#include <iostream>
using namespace std;
class Sum_values{

	public:
		Sum_values(int x,int y){
		cout<<"Sum of two integers is : "<<x+y<<endl;
		}
		Sum_values(float x,float y){
		cout<<"Sum of two float is : "<<x+y<<endl;
		}
		Sum_values(char x,char y){
		cout<<"Sum of two character is : "<<int(x)+int(y)<<endl;
		}
};
int main()
{
int a=3,b=5;
float f1=3.3,f2=2.1;
char c1='s',c2='t';
Sum_values intsum(a,b);
Sum_values floatsum(f1,f2);
Sum_values charsum(c1,c2);

    return 0;
}
```



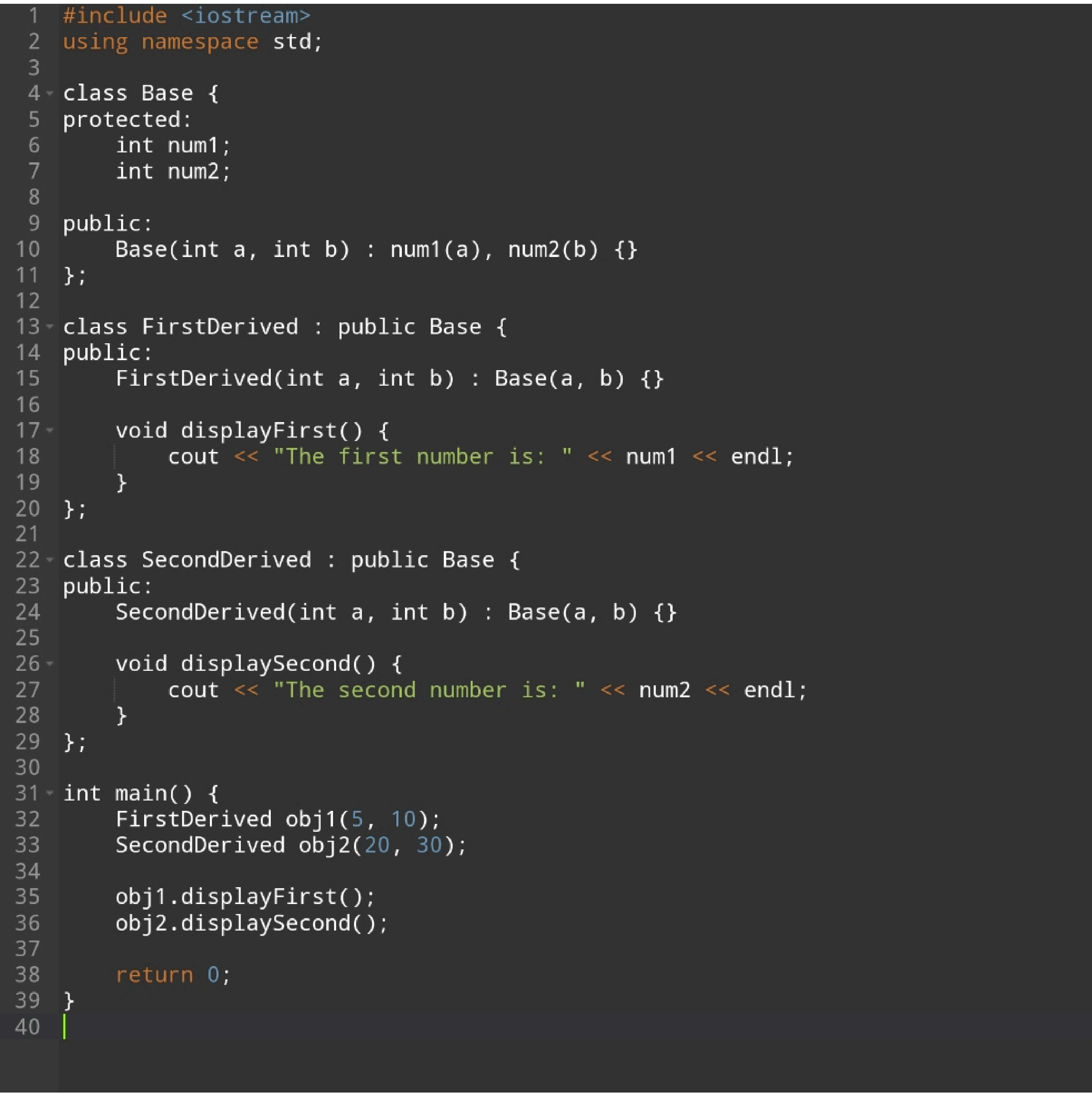
```cpp
#include <iostream>
using namespace std;

class Base {
protected:
    int num1, num2;

public:
    Base(int a, int b) : num1(a), num2(b) {}
};

class Derived : public Base {
public:
    Derived(int a, int b) : Base(a, b) {}

    int product() {
        return num1 * num2;
    }
};

int main() {
    Derived obj(5, 3);
    cout << "The product of " << obj.num1 << " and " << obj.num2 << " is: " << obj.product(
    return 0;
}
```

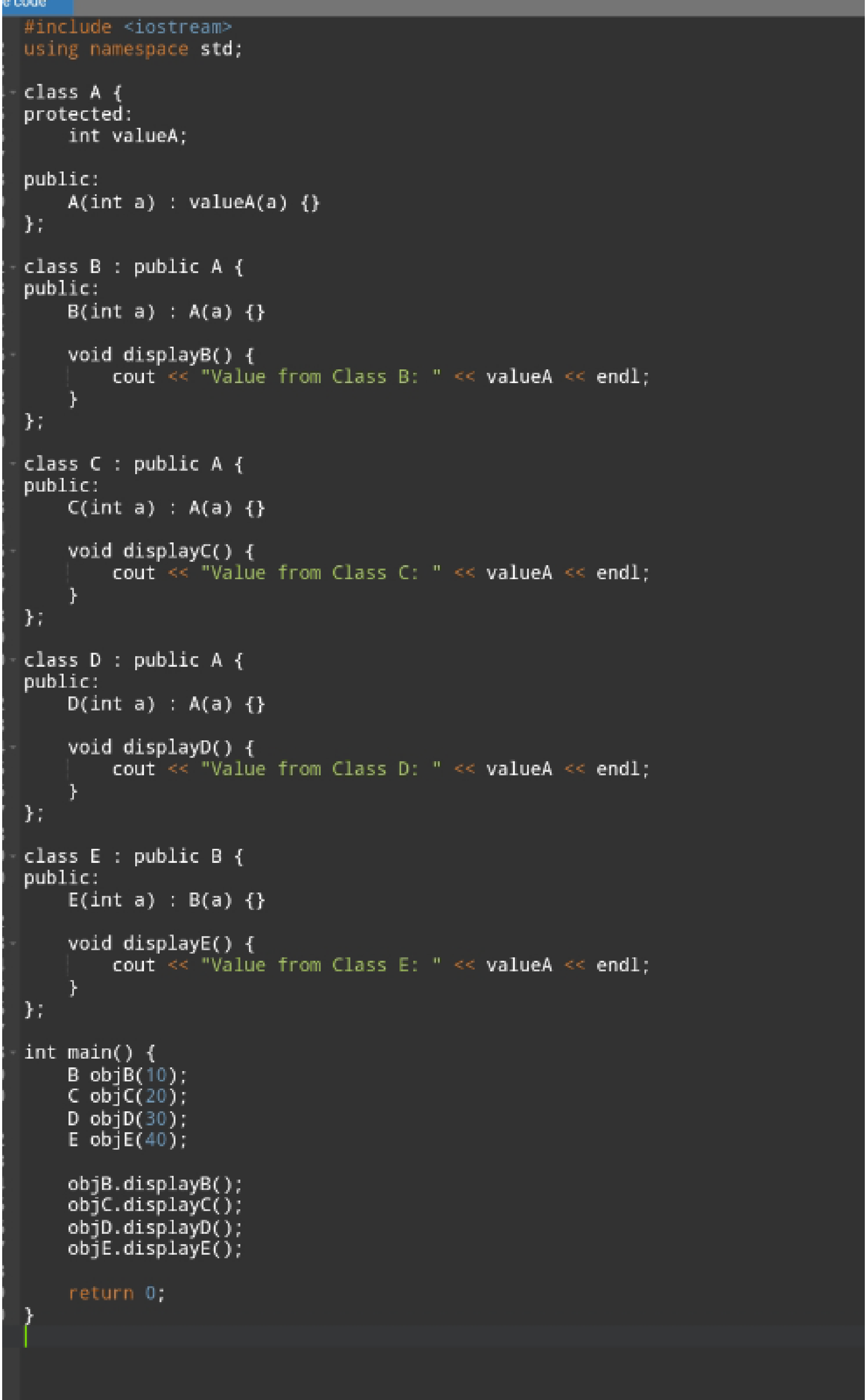


```cpp
#include <iostream>
using namespace std;

class Base {
protected:
    int num1;

public:
    Base(int a) : num1(a) {}
};

class FirstDerived : public Base {
protected:
    int num2;

public:
    FirstDerived(int a, int b) : Base(a), num2(b) {}
};

class SecondDerived : public FirstDerived {
public:
    SecondDerived(int a, int b) : FirstDerived(a, b) {}

    int sum() {
        return num1 + num2;
    }
};

int main() {
    SecondDerived obj(5, 10);
    cout << "The sum of " << obj.num1 << " and " << obj.num2 << " is: " << obj.sum() << enc
    return 0;
}

```



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

class Base1 {
protected:
    int num1;

public:
    Base1(int a) : num1(a) {}
};

class Base2 {
protected:
    int num2;

public:
    Base2(int b) : num2(b) {}
};

class Derived : public Base1, public Base2 {
public:
    Derived(int a, int b) : Base1(a), Base2(b) {}

    int sum() {
        return num1 + num2;
    }
};

int main() {
    Derived obj(5, 10);
    cout << "The sum of " << obj.num1 << " and " << obj.num2 << " is: " << obj.sum() << enc
    return 0;
}
```


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

class Base {
protected:
    int num1;
    int num2;

public:
    Base(int a, int b) : num1(a), num2(b) {}
};

class FirstDerived : public Base {
public:
    FirstDerived(int a, int b) : Base(a, b) {}

    void displayFirst() {
        cout << "The first number is: " << num1 << endl;
    }
};

class SecondDerived : public Base {
public:
    SecondDerived(int a, int b) : Base(a, b) {}

    void displaySecond() {
        cout << "The second number is: " << num2 << endl;
    }
};

int main() {
    FirstDerived obj1(5, 10);
    SecondDerived obj2(20, 30);

    obj1.displayFirst();
    obj2.displaySecond();

    return 0;
}
```



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

class A {
protected:
    int valueA;

public:
    A(int a) : valueA(a) {}
};

class B : public A {
public:
    B(int a) : A(a) {}

    void displayB() {
        cout << "Value from Class B: " << valueA << endl;
    }
};

class C : public A {
public:
    C(int a) : A(a) {}

    void displayC() {
        cout << "Value from Class C: " << valueA << endl;
    }
};

class D : public A {
public:
    D(int a) : A(a) {}

    void displayD() {
        cout << "Value from Class D: " << valueA << endl;
    }
};

class E : public B {
public:
    E(int a) : B(a) {}

    void displayE() {
        cout << "Value from Class E: " << valueA << endl;
    }
};

int main() {
    B objB(10);
    C objC(20);
    D objD(30);
    E objE(40);

    objB.displayB();
    objC.displayC();
    objD.displayD();
    objE.displayE();

    return 0;
}
```