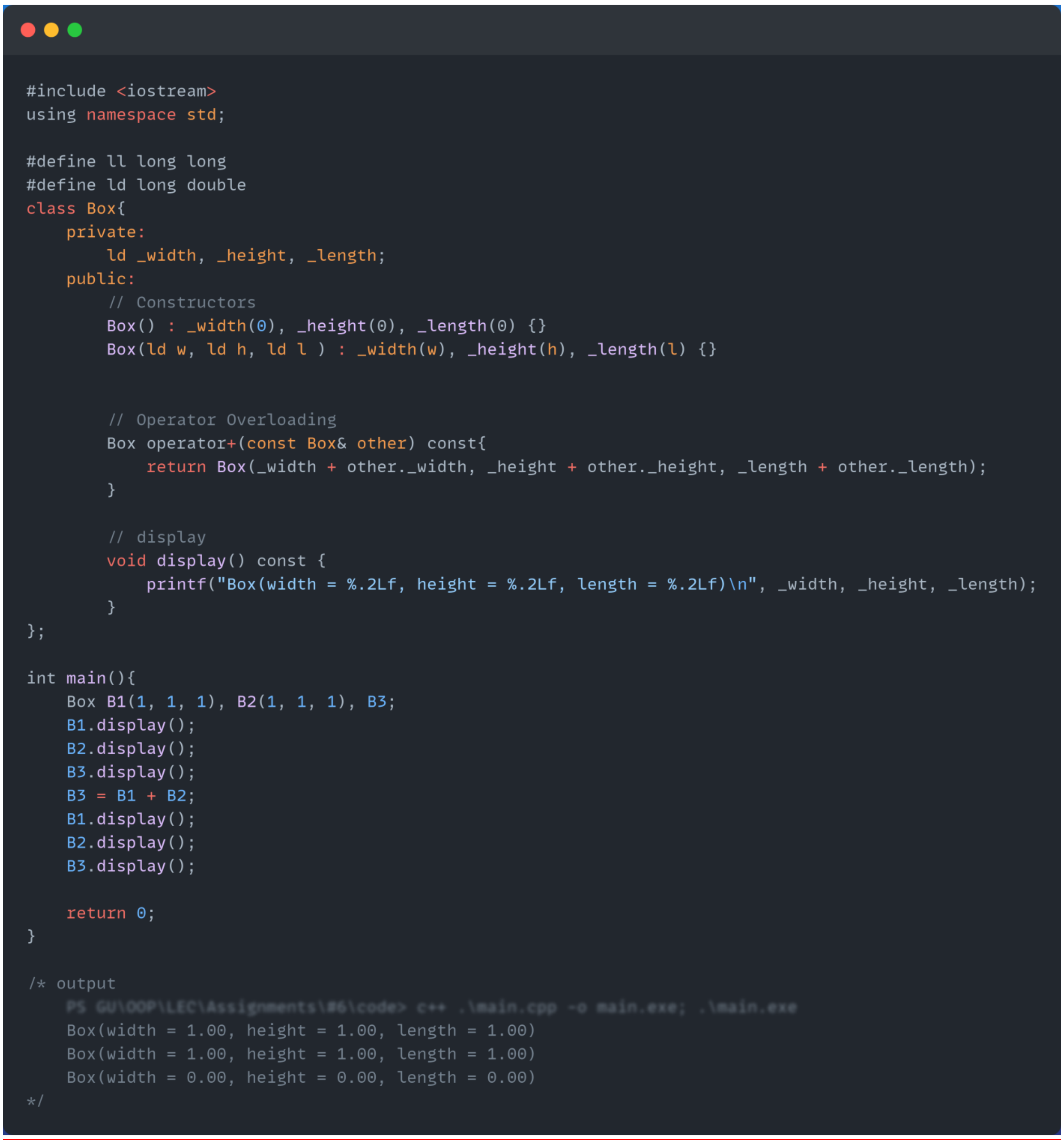
a) A box has three dimensions, length, width, and height. Write a C++ code to create the following: 1- A class Box with private data length, width, and height. 2- A default constructor with zero dimensions. 3- A parameterized constructor. 4- Overloading the arithmetic operator + to add two Box objects such that for objects B1, B2, and B3 of the operation B3 = B1+B2 is carried out as follows: length of Box3 = length of Box1 + length of Box2 width of Box3 = width of Box1 + width of Box2 height of Box3 = height of Box1 + height of Box2 5- A member function Show to display length, width, and height of the Box object. 6- Write a test program to do the following: i) Create two objects of the class Box. ii) Add the two boxes. iii) Display dimensions of the two boxes and the result after addition.

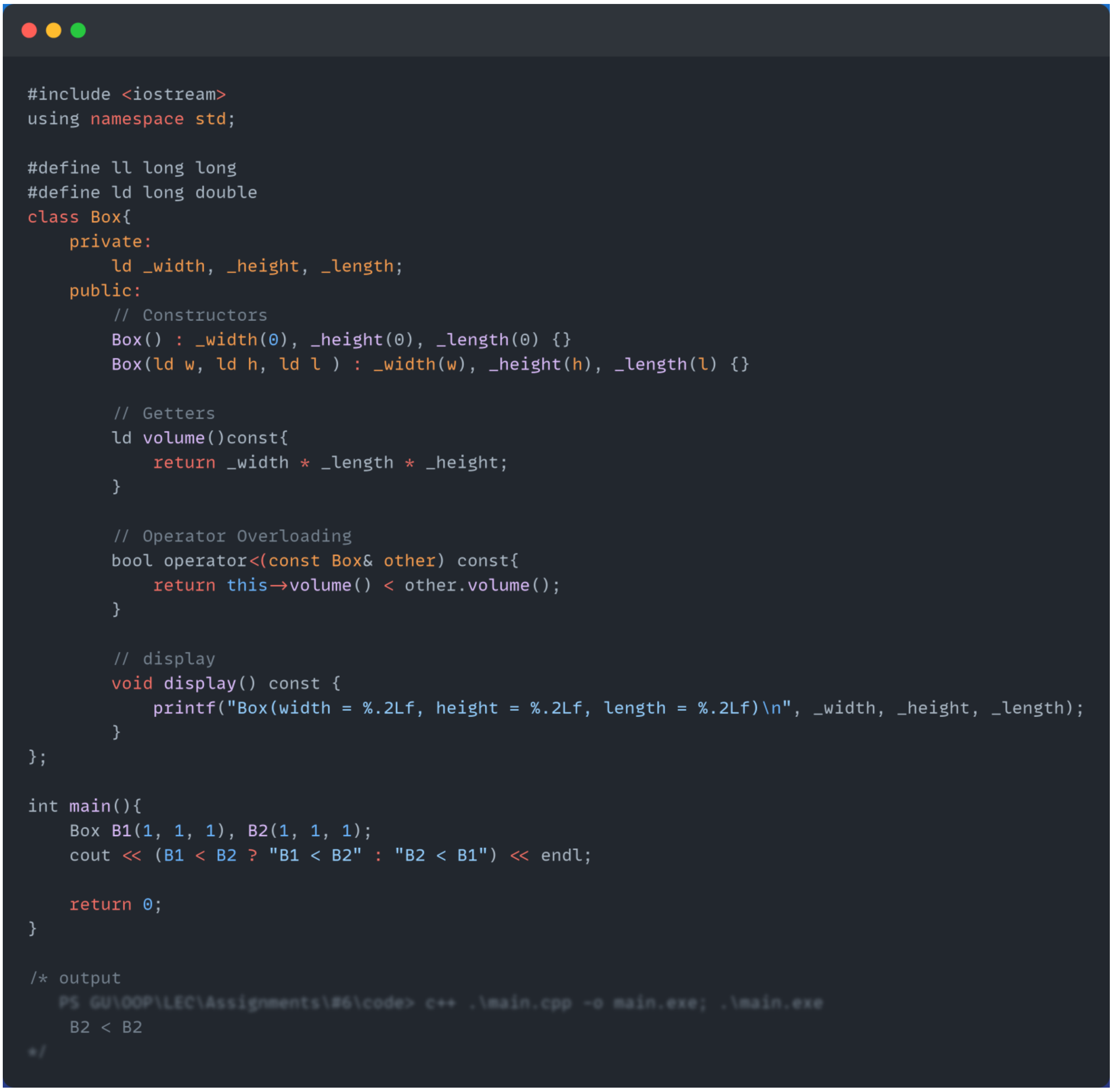
Solution



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

#define ll long long
#define ld long double
class Box{
    private:
        ld _width, _height, _length;
    public:
        // Constructors
        Box() : _width(0), _height(0), _length(0) {}
        Box(ld w, ld h, ld l ) : _width(w), _height(h), _length(l) {}


        // Operator Overloading
        Box operator+(const Box& other) const{
            return Box(_width + other._width, _height + other._height, _length + other._length);
        }

        // display
        void display() const {
            printf("Box(width = %.2Lf, height = %.2Lf, length = %.2Lf)\n", _width, _height, _length);
        }
};

int main(){
    Box B1(1, 1, 1), B2(1, 1, 1), B3;
    B1.display();
    B2.display();
    B3.display();
    B3 = B1 + B2;
    B1.display();
    B2.display();
    B3.display();

    return 0;
}

/* output
    [redacted]
    Box(width = 1.00, height = 1.00, length = 1.00)
    Box(width = 1.00, height = 1.00, length = 1.00)
    Box(width = 0.00, height = 0.00, length = 0.00)
*/
```

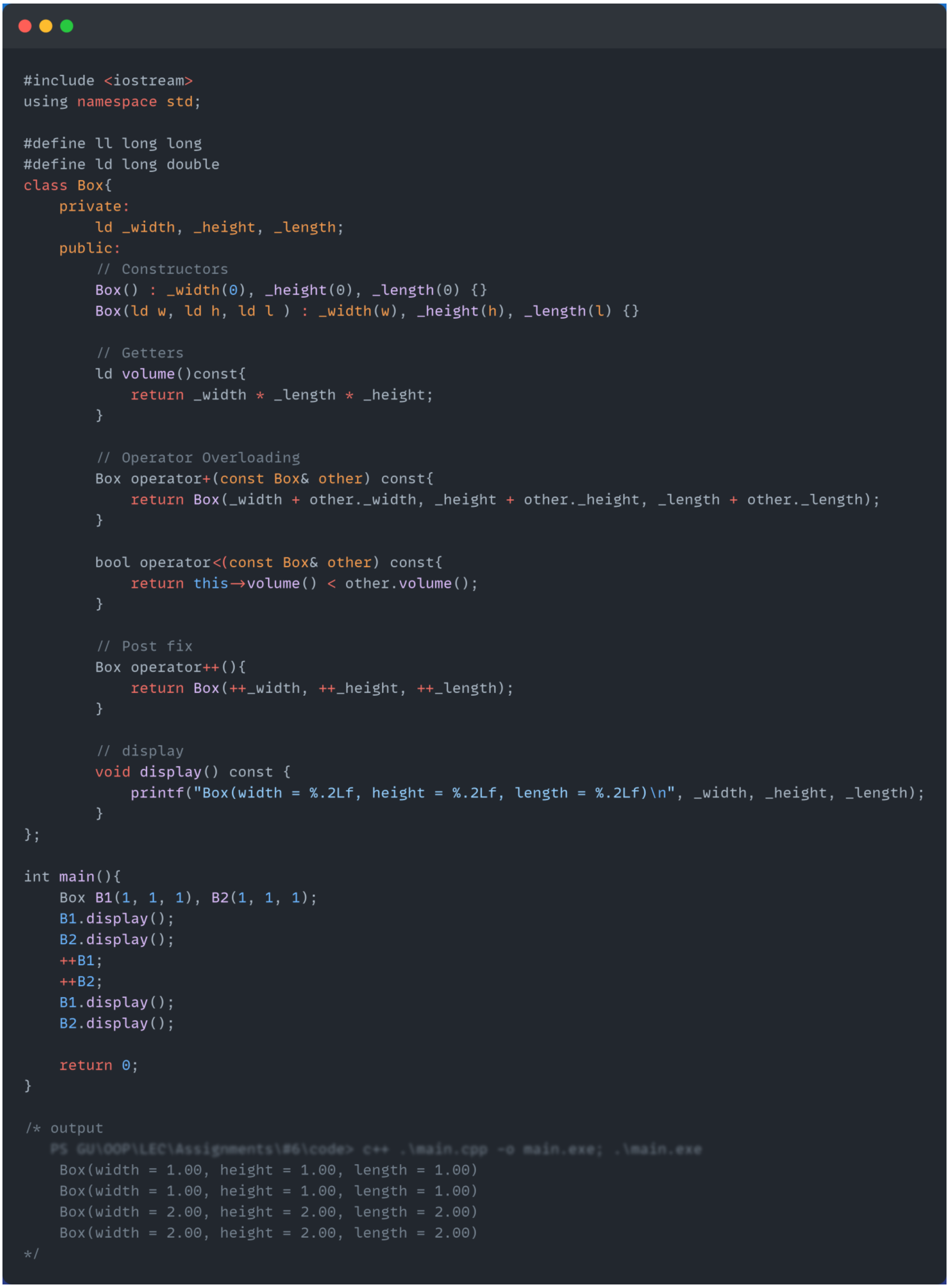
b) Based on the Box class in part 3-a, write a C++ code to create the following: 1- Overloading the less than operator < to compare the volumes of two objects of class Box. 2- A test program to do the following: i) Create two objects of the class Box. ii) Compare the volumes of the two boxes.

Solution



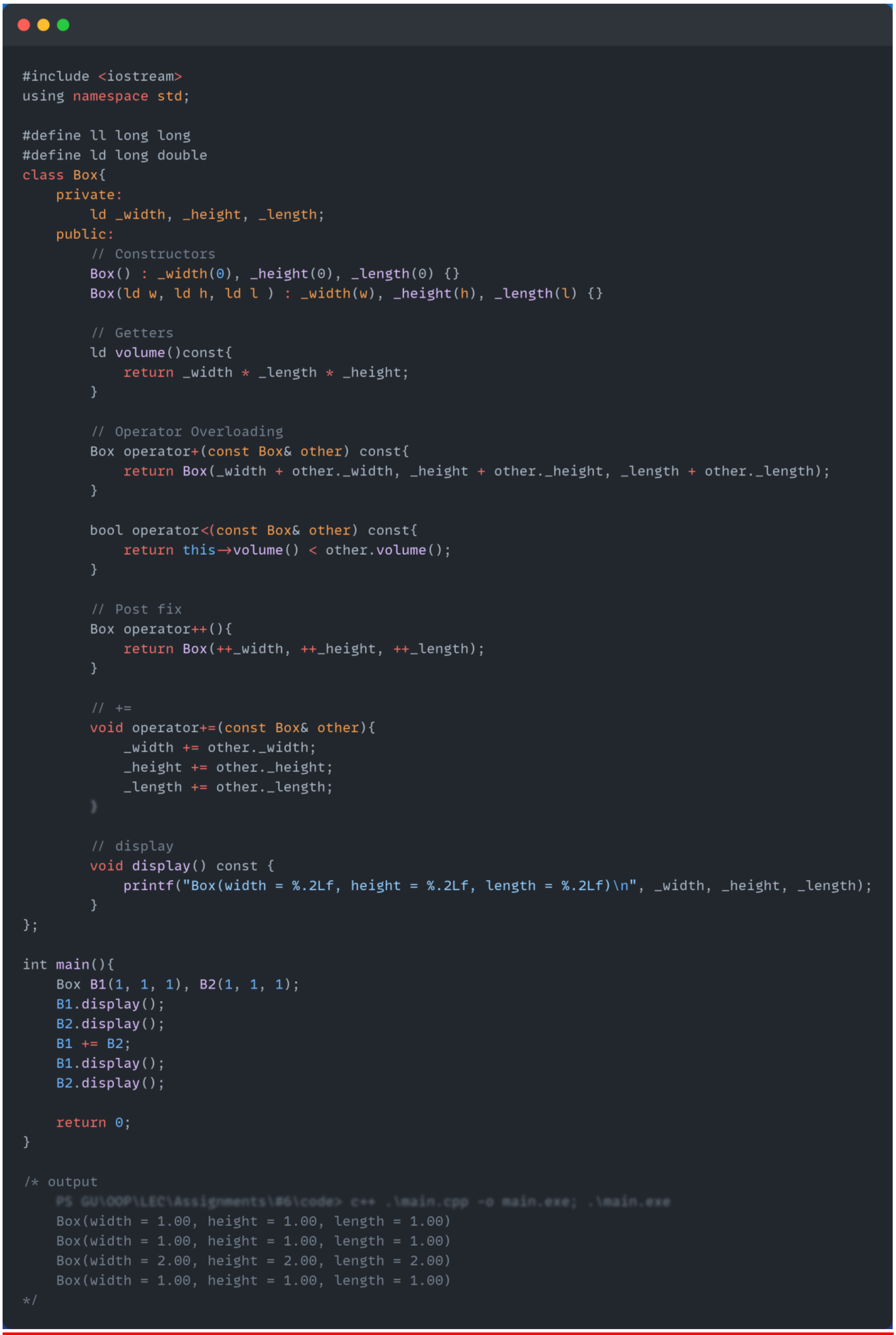
```cpp
#include <iostream>
using namespace std;

#define ll long long
#define ld long double
class Box{
    private:
        ld _width, _height, _length;
    public:
        // Constructors
        Box() : _width(0), _height(0), _length(0) {}
        Box(ld w, ld h, ld l ) : _width(w), _height(h), _length(l) {}

        // Getters
        ld volume()const{
            return _width * _length * _height;
        }

        // Operator Overloading
        bool operator<(const Box& other) const{
            return this->volume() < other.volume();
        }

        // display
        void display() const {
            printf("Box(width = %.2Lf, height = %.2Lf, length = %.2Lf)\n", _width, _height, _length);
        }
};

int main(){
    Box B1(1, 1, 1), B2(1, 1, 1);
    cout << (B1 < B2 ? "B1 < B2" : "B2 < B1") << endl;

    return 0;
}

/* output
    PS GU\OOP\LEC\Assignments\#6\code> c++ .\main.cpp -o main.exe; .\main.exe
    B2 < B2
*/
```

) Write a C++ program that overloads and tests the postfix increment operator ++ to work with the Rectangle class in part 2-a.



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

#define ll long long
#define ld long double
class Box{
    private:
        ld _width, _height, _length;
    public:
        // Constructors
        Box() : _width(0), _height(0), _length(0) {}
        Box(ld w, ld h, ld l ) : _width(w), _height(h), _length(l) {}

        // Getters
        ld volume()const{
            return _width * _length * _height;
        }

        // Operator Overloading
        Box operator+(const Box& other) const{
            return Box(_width + other._width, _height + other._height, _length + other._length);
        }

        bool operator<(const Box& other) const{
            return this->volume() < other.volume();
        }

        // Post fix
        Box operator++(){
            return Box(++_width, ++_height, ++_length);
        }

        // display
        void display() const {
            printf("Box(width = %.2Lf, height = %.2Lf, length = %.2Lf)\n", _width, _height, _length);
        }
};

int main(){
    Box B1(1, 1, 1), B2(1, 1, 1);
    B1.display();
    B2.display();
    ++B1;
    ++B2;
    B1.display();
    B2.display();

    return 0;
}

/* output
    [redacted]
    Box(width = 1.00, height = 1.00, length = 1.00)
    Box(width = 1.00, height = 1.00, length = 1.00)
    Box(width = 2.00, height = 2.00, length = 2.00)
    Box(width = 2.00, height = 2.00, length = 2.00)
*/
```

d) Write a C++ program that overloads and tests the arithmetic assignment operator += to work with the Box class in part 3-a.



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

#define ll long long
#define ld long double
class Box{
    private:
        ld _width, _height, _length;
    public:
        // Constructors
        Box() : _width(0), _height(0), _length(0) {}
        Box(ld w, ld h, ld l ) : _width(w), _height(h), _length(l) {}

        // Getters
        ld volume()const{
            return _width * _length * _height;
        }

        // Operator Overloading
        Box operator+(const Box& other) const{
            return Box(_width + other._width, _height + other._height, _length + other._length);
        }

        bool operator<(const Box& other) const{
            return this->volume() < other.volume();
        }

        // Post fix
        Box operator++(){
            return Box(++_width, ++_height, ++_length);
        }

        // +=
        void operator+=(const Box& other){
            _width += other._width;
            _height += other._height;
            _length += other._length;
        }

        // display
        void display() const {
            printf("Box(width = %.2Lf, height = %.2Lf, length = %.2Lf)\n", _width, _height, _length);
        }
};

int main(){
    Box B1(1, 1, 1), B2(1, 1, 1);
    B1.display();
    B2.display();
    B1 += B2;
    B1.display();
    B2.display();

    return 0;
}

/* output
    [redacted]
    Box(width = 1.00, height = 1.00, length = 1.00)
    Box(width = 1.00, height = 1.00, length = 1.00)
    Box(width = 2.00, height = 2.00, length = 2.00)
    Box(width = 1.00, height = 1.00, length = 1.00)
*/
```