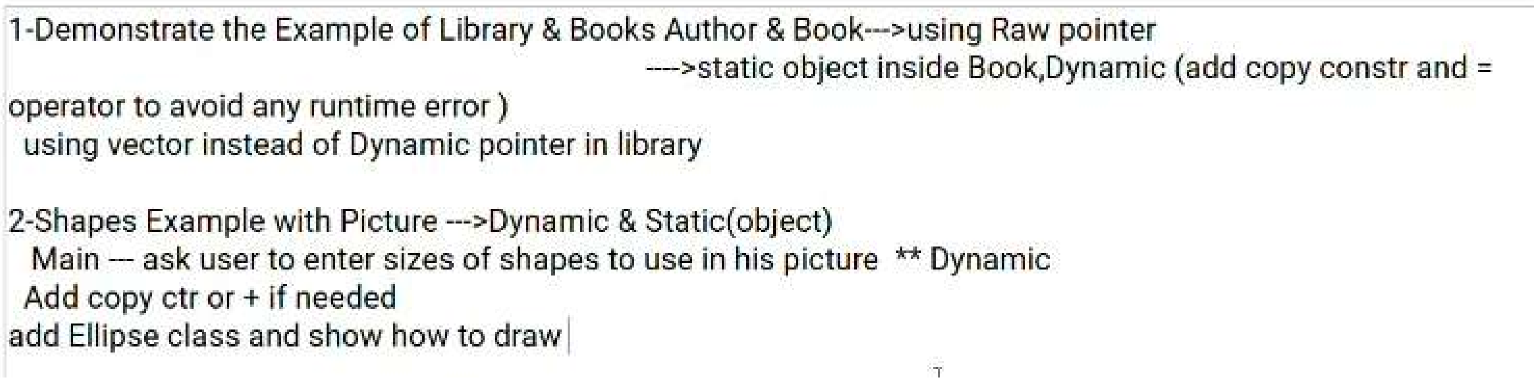
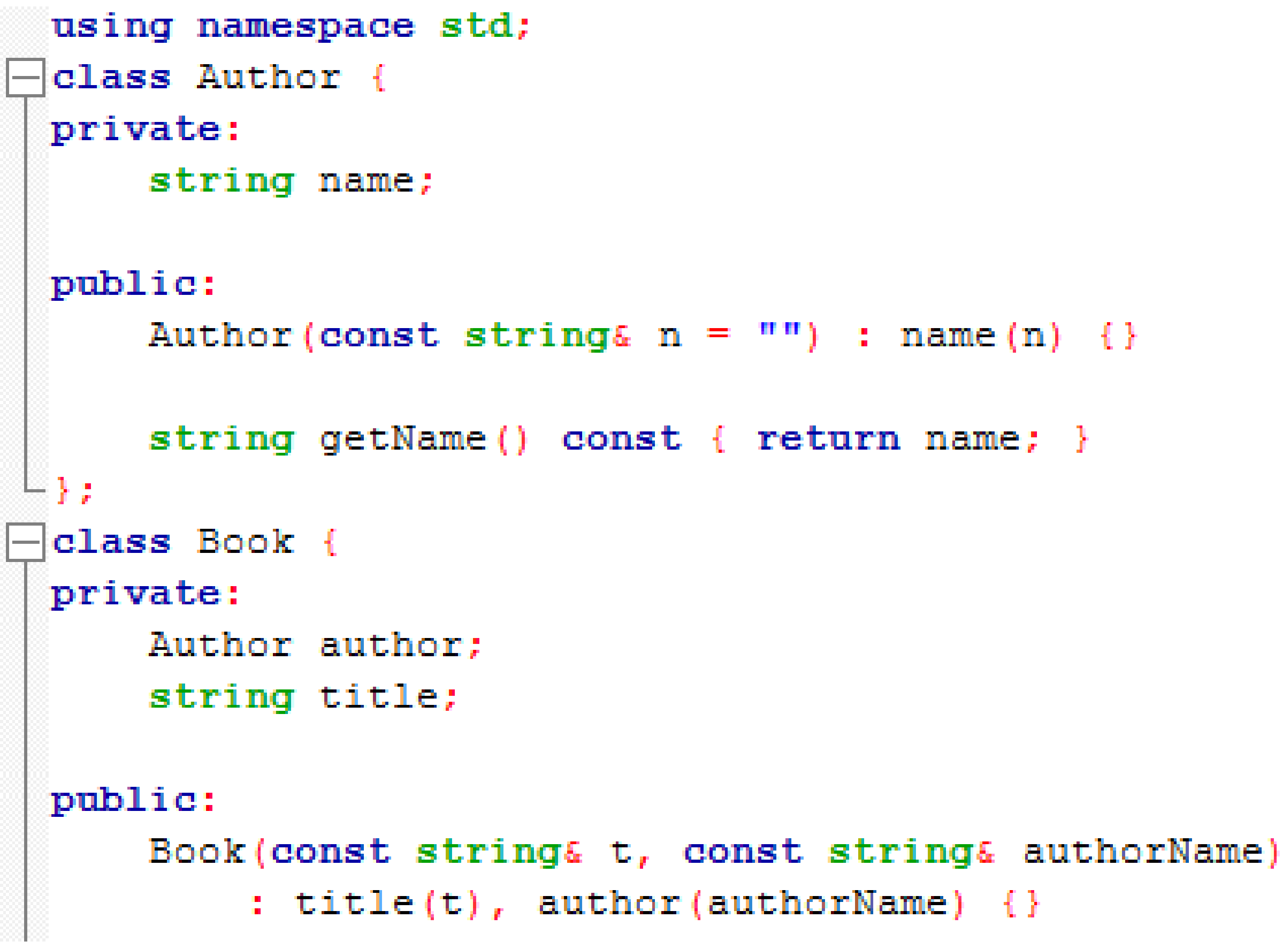
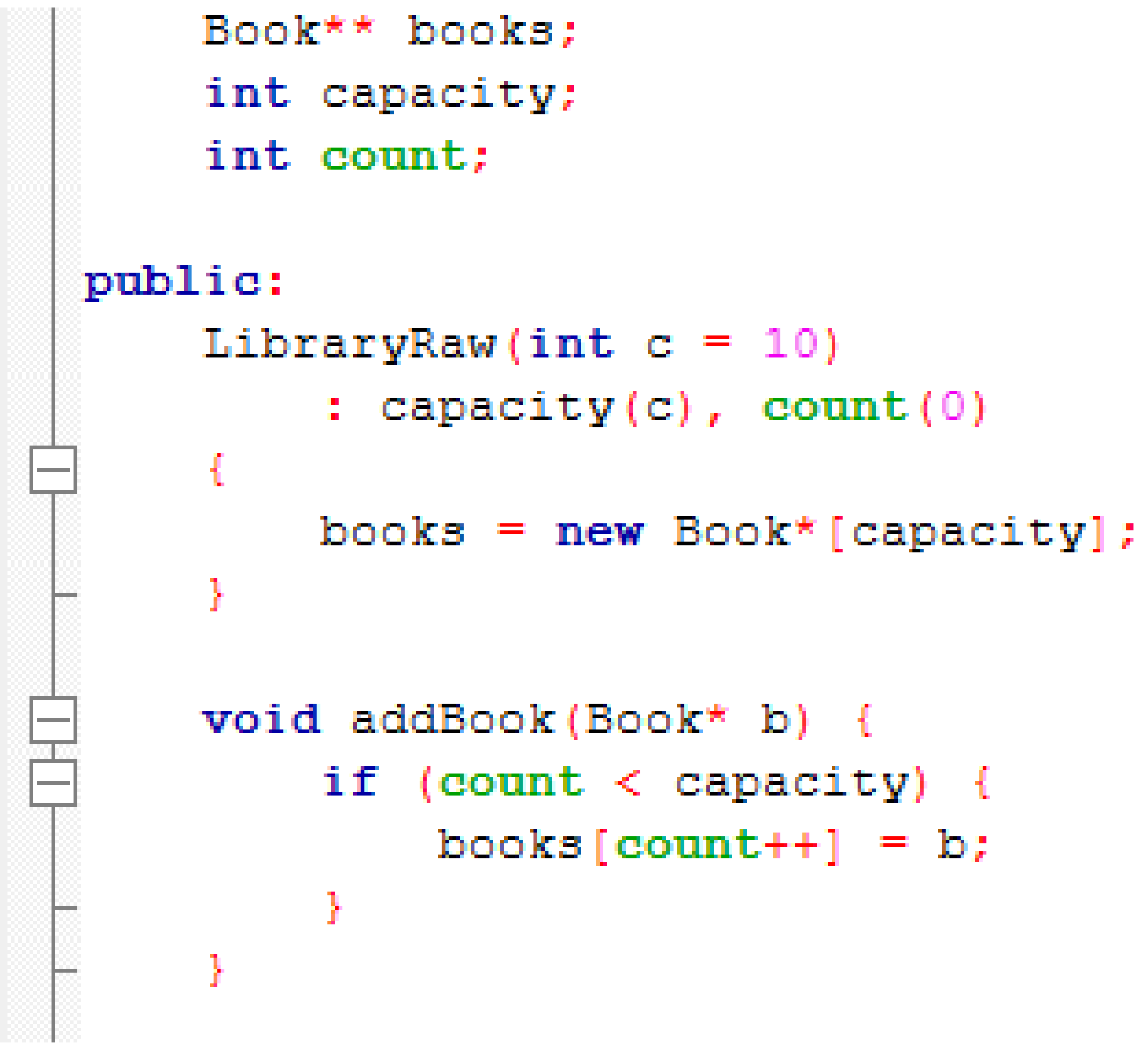


1-Demonstrate the Example of Library & Books Author & Book--->using Raw pointer
                                                                                ---->static object inside Book,Dynamic (add copy constr and = operator to avoid any runtime error)
  using vector instead of Dynamic pointer in library

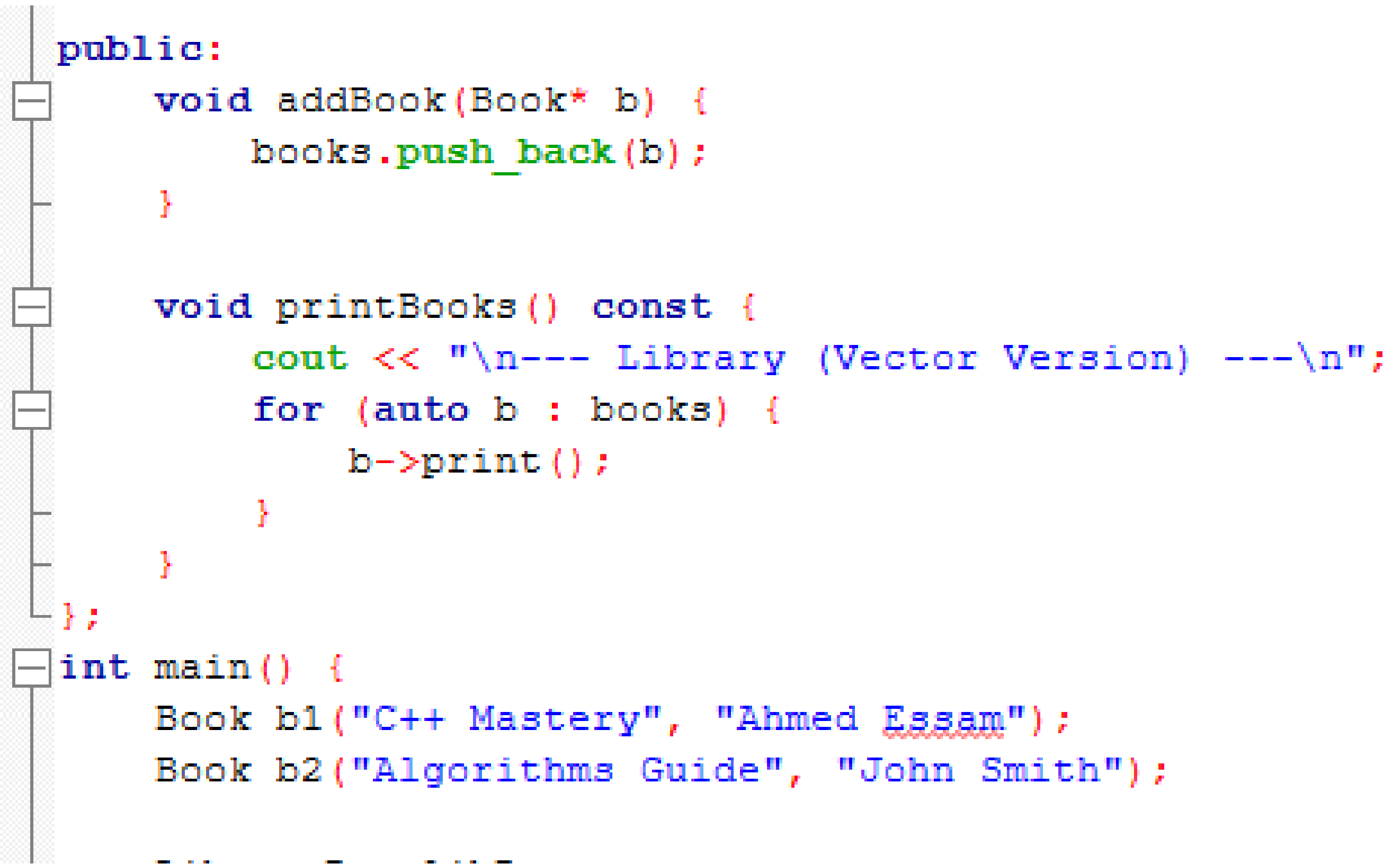
2-Shapes Example with Picture --->Dynamic & Static(object)
   Main --- ask user to enter sizes of shapes to use in his picture ** Dynamic
  Add copy ctr or + if needed
add Ellipse class and show how to draw



```cpp
using namespace std;
class Author {
private:
    string name;

public:
    Author(const string& n = "") : name(n) {}

    string getName() const { return name; }
};
class Book {
private:
    Author author;
    string title;

public:
    Book(const string& t, const string& authorName)
        : title(t), author(authorName) {}
```

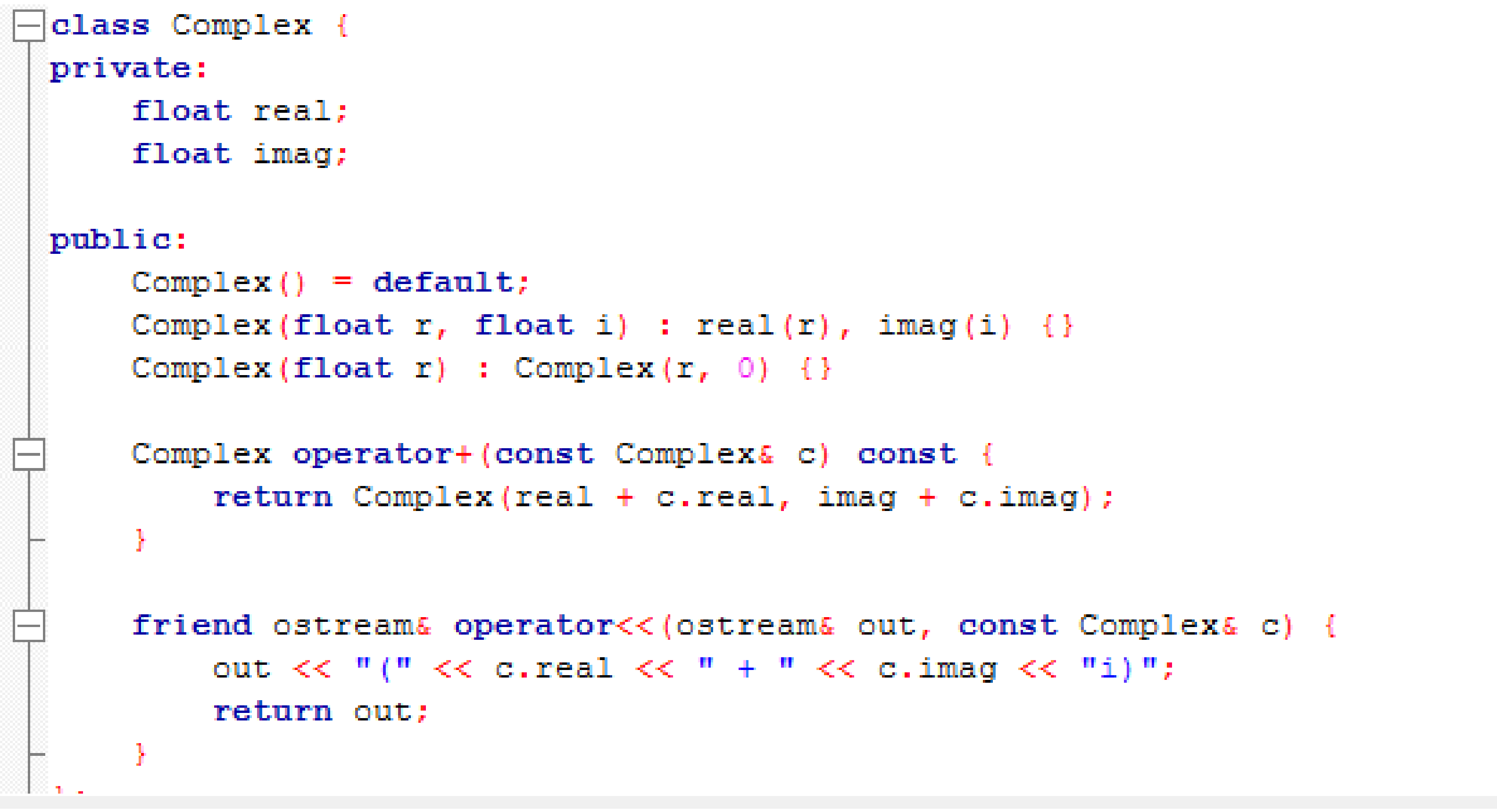


```cpp
    Book** books;
    int capacity;
    int count;

public:
    LibraryRaw(int c = 10)
        : capacity(c), count(0)
    {
        books = new Book*[capacity];
    }

    void addBook(Book* b) {
        if (count < capacity) {
            books[count++] = b;
        }
    }
    void printBooks() const {
        cout << "\n--- Library (Raw Pointer Version) ---\
        for (int i = 0; i < count; i++) {
            books[i]->print();
        }
    }

    ~LibraryRaw() {
        delete[] books;
    }
};
class LibraryVector {
private:
    vector<Book*> books;
```

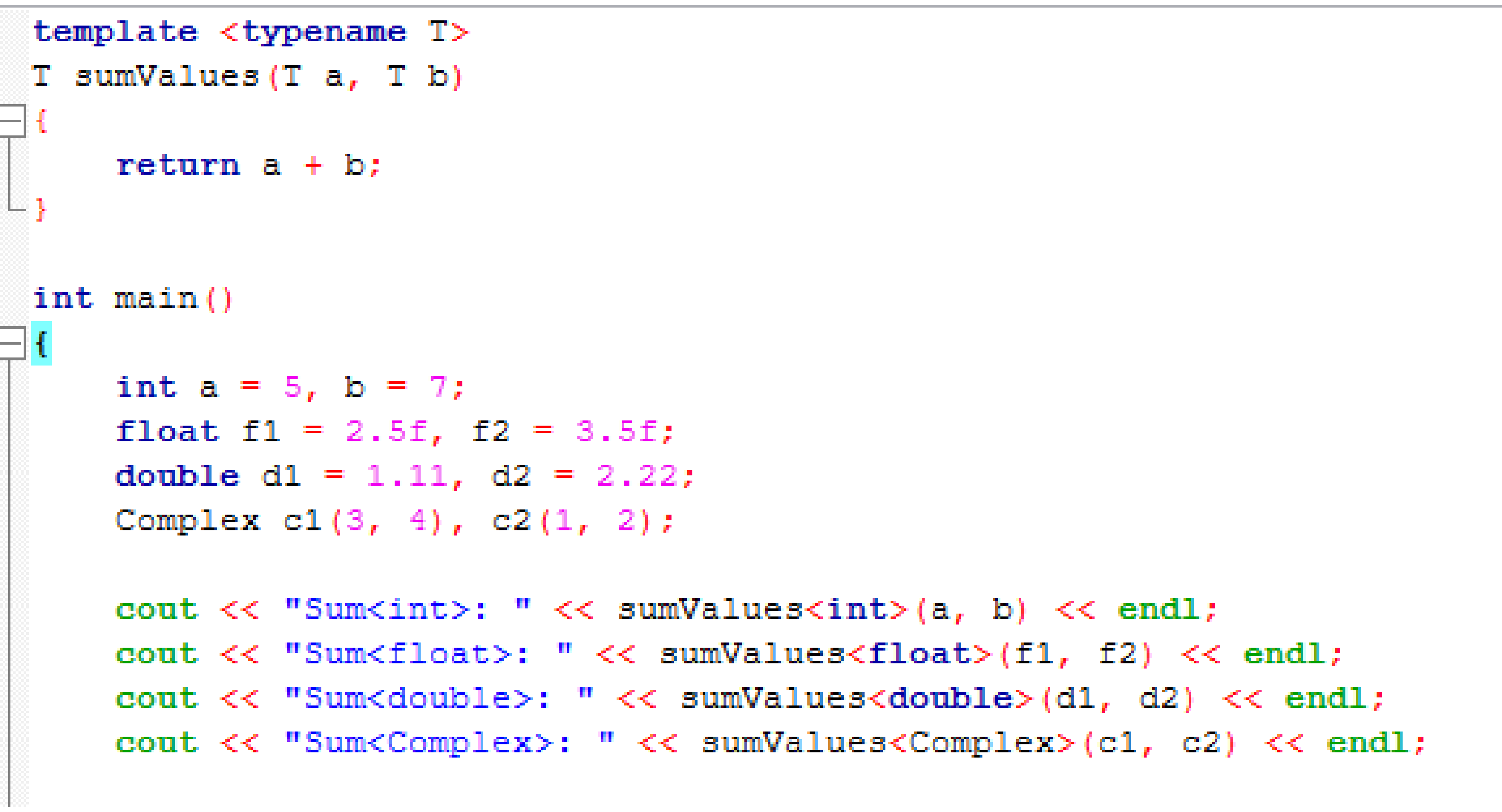


```
public:
    void addBook(Book* b) {
        books.push_back(b);
    }

    void printBooks() const {
        cout << "\n--- Library (Vector Version) ---\n";
        for (auto b : books) {
            b->print();
        }
    }
};
int main() {
    Book b1("C++ Mastery", "Ahmed Essam");
    Book b2("Algorithms Guide", "John Smith");
```

```
LibraryRaw libRaw;
libRaw.addBook(&b1);
libRaw.addBook(&b2);

LibraryVector libVec;
libVec.addBook(&b1);
libVec.addBook(&b2);

// Print results
libRaw.printBooks();
libVec.printBooks();

return 0;
```



```cpp
class Complex {
private:
    float real;
    float imag;

public:
    Complex() = default;
    Complex(float r, float i) : real(r), imag(i) {}
    Complex(float r) : Complex(r, 0) {}

    Complex operator+(const Complex& c) const {
        return Complex(real + c.real, imag + c.imag);
    }

    friend ostream& operator<<(ostream& out, const Complex& c) {
        out << "(" << c.real << " + " << c.imag << "i)";
        return out;
    }
```



```cpp
template <typename T>
T sumValues(T a, T b)
{
    return a + b;
}

int main()
{
    int a = 5, b = 7;
    float f1 = 2.5f, f2 = 3.5f;
    double d1 = 1.11, d2 = 2.22;
    Complex c1(3, 4), c2(1, 2);

    cout << "Sum<int>: " << sumValues<int>(a, b) << endl;
    cout << "Sum<float>: " << sumValues<float>(f1, f2) << endl;
    cout << "Sum<double>: " << sumValues<double>(d1, d2) << endl;
    cout << "Sum<Complex>: " << sumValues<Complex>(c1, c2) << endl;
```