

# COMSC-200

## Lab 4

Ryan Jacoby

20 September 2020

### 1 Complex

```
1 // Fig. 10.14: Complex.h
2 // Complex class definition.
3 #ifndef COMPLEX_H
4 #define COMPLEX_H
5
6 #include <iostream>
7 using namespace std;
8
9 class Complex {
10 public:
11     explicit Complex( double = 0.0, double = 0.0 ); // constructor
12     Complex operator+( const Complex & ) const; // addition
13     Complex operator-( const Complex & ) const; // subtraction
14     Complex operator*( const Complex & ) const; // multiplication
15     bool operator==( const Complex & ) const; // equals
16     bool operator!=( const Complex & ) const; // inequality
17     // TODO: Implement << and >>.
18
19     int getReal() { return real; }
20     int getImaginary() { return imaginary; }
21 private:
22     double real; // real part
23     double imaginary; // imaginary part
24 }; // end class Complex
25
26 ostream& operator<<(ostream& out, Complex &c);
27 istream& operator>>(istream& in, Complex &c);
28
29 #endif
```

Listing 1: Complex.h

```
1 // Fig. 10.15: Complex.cpp
2 // Complex class member-function definitions.
3 #include <iostream>
4 #include "Complex.h" // Complex class definition
5 using namespace std;
6
7 // Constructor
8 Complex::Complex( double realPart, double imaginaryPart ) : real( realPart ),
    imaginary( imaginaryPart ) {
```

```

9  }
10
11 // addition operator
12 Complex Complex::operator+( const Complex &operand2 ) const {
13     return Complex( real + operand2.real, imaginary + operand2.imaginary );
14 }
15
16 // subtraction operator
17 Complex Complex::operator-( const Complex &operand2 ) const {
18     return Complex( real - operand2.real, imaginary - operand2.imaginary );
19 }
20
21 // multiplication operator
22 Complex Complex::operator*( const Complex &operand2 ) const {
23     return Complex( (real * operand2.real) + (imaginary * operand2.imaginary), (
24         real * operand2.imaginary) + (imaginary * operand2.real));
25 }
26 // equals operator
27 bool Complex::operator==( const Complex &operand2 ) const {
28     return (real - operand2.real == 0) && (imaginary - operand2.imaginary == 0);
29 }
30
31 //inequals operator
32 bool Complex::operator!=( const Complex &operand2 ) const {
33     return (real - operand2.real != 0) || (imaginary - operand2.imaginary != 0);
34 }
35
36 ostream& operator<<(ostream& out, Complex &c) {
37     out << '(' << c.getReal() << ", " << c.getImaginary() << ')';
38     return out;
39 }
40
41 istream& operator>>(istream& in, Complex &c) {
42     int real, imaginary;
43     char seperator;
44
45     in >> real;
46     in.get(seperator);
47     in >> imaginary;
48
49     c = Complex(real, imaginary);
50
51     return in;
52 }

```

Listing 2: Complex.cpp

```

1 // Fig. 10.16: fig10_16.cpp
2 // Complex class test program.
3 #include <iostream>
4 #include "Complex.h"
5 using namespace std;
6
7 int main() {
8     Complex x;
9     Complex y( 4.3, 8.2 );
10    Complex z( 3.3, 1.1 );
11

```

```

12     cout << "x: " << x;
13     cout << "\ny: " << y;
14     cout << "\nz: " << z;
15
16     x = y + z;
17     cout << "\n\nx = y + z:\n";
18     cout << x << " = " << y << " + " << z;
19
20     x = y - z;
21     cout << "\n\nx = y - z:\n";
22     cout << x << " = " << y << " - " << z;
23
24     x = y * z;
25     cout << "\n\nx = y * z:\n";
26     cout << x << " = " << y << " * " << z;
27
28     if(y == z)
29         cout << "\n\ny == z: True\n";
30     else
31         cout << "\n\ny == z: False\n";
32
33     if(y != z)
34         cout << "\ny != z: True\n";
35     else
36         cout << "\ny != z: False\n";
37
38     return 0;
39 }

```

Listing 3: fig10\_16.cpp (modified)

## 2 Car

```

1 // Ryan Jacoby
2
3 #ifndef Car_H
4 #define Car_H
5
6 class Car {
7 private:
8     int speed, tank;
9 public:
10     explicit Car(int = 0, int = 0);
11     int operator+( const Car & ) const;
12     int operator-( const Car & ) const;
13     bool operator<( const Car & ) const;
14     bool operator>( const Car & ) const;
15     bool operator==( const Car & ) const;
16     bool operator!=( const Car & ) const;
17
18     int operator++();
19     int operator++( int );
20     int operator--();
21     int operator--( int );
22
23     int getSpeed() { return speed; }
24     int getTank() { return tank; }

```

```

25 };
26
27 int operator+( const Car &, int );
28 int operator+( int, const Car & );
29
30 ostream& operator<<(ostream& out, Complex &c);
31 istream& operator>>(istream& in, Complex &c);
32
33 #endif

```

Listing 4: Car.h

```

1 // Ryan Jacoby
2
3 #include <cmath>
4 #include <sstream>
5 #include "Car.h"
6
7 using namespace std;
8
9 Car::Car(int tank, int speed) {
10     this->tank = tank;
11     this->speed = speed;
12 }
13
14 int Car::operator+( const Car c& ) {
15     return tank + c.tank;
16 }
17
18 int Car::operator-( const Car c& ) {
19     return abs(tank - c.tank);
20 }
21
22 bool Car::operator<( const Car c& ) {
23     return tank - c.tank < 0;
24 }
25
26 bool Car::operator>( const Car c& ) {
27     return tank - c.tank > 0;
28 }
29
30 bool Car::operator==( const Car & ) {
31     return tank - c.tank == 0;
32 }
33
34 bool Car::operator!=( const Car & ) {
35     return tank - c.tank != 0;
36 }
37
38 int Car::operator++() {
39     speed++;
40     return speed;
41 }
42
43 int Car::operator++( int ) {
44     speed++;
45     return speed - 1;
46 }
47

```

```

48 int Car::operator--() {
49     if(speed > 0) {
50         speed--;
51         return speed;
52     }
53
54     return speed;
55 }
56 int Car::operator--( int ) {
57     if(speed > 0) {
58         speed--;
59         return speed + 1;
60     }
61
62     return speed;
63 }
64
65
66 int operator+( const Car c&, int n ) {
67     return c.getTank() + n;
68 }
69 int operator+( int n, const Car c& ) {
70     return c.getTank() + n;
71 }
72
73 ostream& operator<<(ostream& out, Complex &c) {
74     out << "Tank: " << c.getTank() << "\tSpeed: " << c.getSpeed();
75     return out;
76 }
77 istream& operator>>(istream& in, Complex &c) {
78     int tank, speed;
79     char seperator;
80
81     in >> tank;
82     in.get(seperator);
83     in >> speed;
84
85     c = Car(tank, speed);
86
87     return in;
88 }

```

Listing 5: Car.cpp