CS 271
October 19, 2017

```
class Complex {

   friend ostream & operator<< ( ostream&, const Complex& );

   friend istream & operator>> ( istream&, Complex& );

   private:

      float real;
      float imag;

   public:

      Complex();
      Complex( float, float );
      float getReal();
      float getImag();
      void setReal( float );
      void setImag( float );
      Complex operator+ ( const Complex& ) const;
      //
      // 1) left operand = calling object
      // 2) right operand = parameter
      // 3) return value is a new Complex object

      Complex operator- ( const Complex& ) const;
      Complex operator* ( const Complex& ) const;
      Complex operator/ ( const Complex& ) const;
};
```

Save in a file Complex.h

Let's write the + operator. This should be in a file Complex.cpp

```
Complex Complex::operator+ (const Complex& x ) const {

    float realPart = real + x.getReal();
    float imagPart = imag + x.getImag();
    Complex answer( realPart, imagPart );
    return answer;
}
```

Let's write the << operator.

```
ostream& operator<< ( ostream& out, const Complex& y ) {

   out << y.getReal();

   if (y.getImag() < 0)
      out << " - " << y.getImag() * -1 << "i";
    else
      out << " + " << y.getImag() << "i";

    return out;

}
```

In the driver program (test program):

```
  #include…

  using namespace std;

  int main() {

     Complex num1 ( 0, 3.4 );
     Complex num2 ( 3.1415, 2.7 );

     cout << "The sum is " << (num1 + num2) << endl;

   }
```

Let's write the operator>> function (Complex.cpp)

```
istream& operator>> ( istream& in, Complex& number ) {

    float realPart;
    float imagPart;

    in >> realPart >> imagPart;
    number.setReal( realPart );
    number.setImag( imagPart );

    return in;
}
```

Operators that cannot be overloaded

```
.   .*    ::    ?:
```