Implementation of Classes in C++

Larry Caretto
Computer Science 106

Computing in Engineering and Science

May 16, 2006

California State University Northridge

Schedule

- Today: Second lecture on classes
- Thursday
  - Project 3 deadline
  - Review for final
- Tuesday, May 23: Final exam 12:45 to 2:45 pm in this room
- Friday, May 26: Last day for late assignments

California State University Northridge 2

Outline

- Quiz results and comments
- Review idea of classes
- Review example of a class to be used for calculations with complex numbers
  - Definition of class
  - Use of class
- Details of class functions

California State University Northridge 3

Quiz Results

- Number of students: 8
- Maximum possible score: 25
- Mean score: 19.1
- Median score: 19.5
- Standard deviation of scores: 4.82
- Grade distribution:
  12 15 15 19 20 22 25 25

California State University Northridge 4

Basic Code for Solution

```
double rmsV = 0;
double rmsI = 0;
double power = 0;
for ( int k = 0; k < n; k++ )    {
       rmsV  += v[k] * v[k];
       rmsI  += i[k] * i[k];
       power += v[k] * i[k];
}
rmsV = sqrt( rmsV / n );
rmsI = sqrt( rmsI / n );
power /= n;
```

California State University Northridge 5

Quiz Comments

- Remember to use use pass-by-reference to return more than one function calculation to the calling function
- In loops, do only the sums
  - Leave final divisions, subtractions, square roots, etc. to be done after loop ends
- Note difference between array size and number of elements actually defined
- Use meaningful variable names

California State University Northridge 6

Review Class Definition and Use

- Class declaration
  - Specify data and functions as public or private
  - Declares data items belonging to class
  - Provides prototypes of class member functions and friend functions
    - May have function body declared as inline functions
    - Only member functions can access private member
- Definition of class member functions
  - Separate from class declaration
  - Usually done in separate file

Review File Structure

- Header file
  - Contains class declaration (with member function prototypes)
  - Used in files that define member functions and files that use member functions
- Definition of class member functions in separate file
- Other files that use classes include header file with class definitions

Review Complex Number Class

- Shows example of class that allows operations on complex numbers
- Usually seen in electrical circuits, aerodynamics, and electromagnetics
- Complex numbers have a real and an imaginary part
- We want to be able to perform mathematical operations with complex numbers
- Also need input/output

Review Complex Number, z

- Two basic representations
  - Rectangular: real (x) and imaginary (y) parts
  - Polar: magnitude (r) and angle (θ)

$$z = x + jy = re^{j\theta} \qquad j = \sqrt{-1}$$

$$x = r\cos(\theta) \qquad y = r\sin(\theta)$$

$$r = \sqrt{x^2 + y^2} \qquad \theta = \tan^{-1}\left(y / x \right)$$

Review Complex Number Operations

$$z_2 = cz_1 \Rightarrow x_2 = cx_1 \qquad y_2 = cy_1$$

$$z_3 = z_1 \pm z_2 \Rightarrow x_3 = x_1 \pm x_2 \qquad y_3 = y_1 \pm y_2$$

$$z_3 = z_1 z_2 \quad \Rightarrow \quad \begin{cases} x_3 = x_1 x_2 - y_1 y_2 \\ y_3 = y_1 x_2 + y_2 x_1 \end{cases}$$

$$z_3 = \frac{z_1}{z_2} \quad \Rightarrow \quad \begin{cases} \left(x_2^2 + y_2^2\right) x_3 = x_1 x_2 + y_1 y_2 \\ \left(x_2^2 + y_2^2\right) y_3 = y_1 x_2 - y_2 x_1 \end{cases}$$

Review Complex class

- Class declaration
  - Two member data components: real and imaginary parts of a complex number
  - Various member functions to get data about the complex number, provide input and output and define operators for complex numbers
- Complex class objects are complex numbers
- Functions specified in class declaration implemented in separate (header) file

Review Code for Complex Class

- Start with class declaration that will show prototypes of available functions/operators
- In file complex.h used by other functions
- Show main function that uses the complex class – show commands and output from the commands
- Show definition of member functions after showing prototypes and result of use
- All files have header, "complex.h"

California State University Northridge 13

```
class complex
{
    private:
       double Re;     // Real part of complex number
       double Im;     // Imaginary part of complex number

  // Member functions (only prototypes required in class
  // definition.  These are public so that they can be used
  // by remainder of code.  { Some are inline. }

    public:
       complex() { Re = 0; Im = 0; }        // constructor
       complex( double inRe, double inIm )  // second constuctor
            { Re = inRe; Im = inIm; }
       double getRe() { return Re; }   // returns value
       double getIm() { return Im; }   // returns value
       double getMagnitude() { return
           sqrt( Re * Re + Im * Im ); }// magnitude of number
       double getPhase()               // phase angle
             { return atan2( Im, Re ); }
```

14

```
 // Various function operators.  Note overloading and "friends."

   friend ostream& operator<< ( ostream& os, const complex& c )
       { os << '(' << c.Re << ", " << c.Im << ')';   return os;
   friend istream& operator>> ( istream& is, complex& c )
       { is >> c.Re >> c.Im;   return is; }
   complex plus( complex c );
   friend complex add( complex c1, complex c2 );
   complex operator+( complex c );
   complex operator*( complex c );
   friend complex operator*( complex c, double d );
   friend complex operator*( double d, complex c );

};       // End of class definition uses a } plus a semicolon!
```

15

Review Code for Complex Class

- Members and prototypes provided in class declaration tell us what is available
- If we understand what the functions are supposed to do, we can use them without knowing their details
- The following charts show a main function that declares and uses complex objects
- Output shown as comments in the code to see results of class functions
- Note: $3^2 + 4^2 = 5^2$ and tan(4/5) = 53.1301°

California State University Northridge 16

```
        // main shows use of the complex class
   #include "complex.h"
   DEGREES_PER_RADIAN = 45 / atan(1); // global constant
   int main()
   {           // Declare and output complex data types.

      complex a;             // same as complex a( 0, 0 )
      complex b( 3, 4 );     // Re = 3 and Im = 4
      cout << "a = " << a << " and b = " << b << endl;
      cout << "The magnitude of b is: "
           <<  b.getMagnitude() << endl;
      cout << "The phase angle of b is: " << ( b.getPhase()
           * DEGREES_PER_RADIAN ) << " degrees.\n";


   /*    Program Output
   a = (0, 0) and b = (3, 4)
   The magnitude of b is: 5
   The phase angle of b is: 53.1301 degrees. */
```

17

```
    // Show output operator and addition operations

    complex c( 7, 10 );
    cout << "\nc = " << c << endl;
    cout << "Use of add function, add( b, c ) = " << add( b, c );
    cout << "Use of + operator, b + c = " << ( b + c ) << endl;
    cout << "Before b.plus( c ), b = " << b;
    cout << " and b.plus( c ) = " << b.plus( c ) << endl;
    cout << "After b.plus( c ), b = " << b << endl;

       // Show multiplication operations

 /*    Program Output

 c = (7, 10)
 Use of add function, add( b, c ) = (10, 14)
 Use of + operator, b + c = (10, 14)
 Before b.plus( c ), b = (3, 4) and b.plus( c ) = (10, 14)
 After b.plus( c ), b = (10, 14)
```

18

```
    // Show multiplication operations

  cout << "\n2 * b = " << ( 2 * b ) << " and b * 2 = "
       << ( b * 2 ) << endl;
  complex d = c * b;
  cout << "The real part of d is: " << d.getRe() << endl;
  cout << "The imaginary part of d is: " << d.getIm() << endl;
  cout << "The magnitude of d is: " << d.getMagnitude() << endl;
  cout << "The phase angle of d is: " << d.getPhase()
       << " radians.\n";

  //   Test overloaded extraction (>>) operator for input

/*    Program Output

2 * b = (20, 28) and b * 2 = (20, 28)
The real part of d is: 74
The imaginary part of d is: 94
The magnitude of d is: 119.633
The phase angle of d is: 0.903888 radians.
```

19

```
  //   Test overloaded extraction (>>) operator for input

  cout << "\nEnter real and imaginary parts of your number: ";
  cin >> a;
  cout << "The complex number you entered is: " << a << endl;
  return EXIT_SUCCESS;
}

/*    Program Output

Enter the real and imaginary parts of your number: 15 20
The complex number you entered is: (15, 20)
Press any key to continue
```

20

Complex Class Code Details

- Now examine details of class functions
- Function code similar to normal functions
  - Have special code for operators
  - Use complex:: before names of functions
- Use header complex.h with class definitions in implementation code file
- Use of different meanings for operators in classes is called operator overloading

California State University Northridge

21

```
      // Actual function definitions go here
      // All functions declared in class (but not written as
      // inline functions) are defined below.
      // See http://mathworld.wolfram.com/ComplexNumber.html
      // for background on formulae used in these functions

complex complex::plus( complex c )
{
      // The command b.plus( c ) increases the value
      // of the complex number, b, by the input value c.
   Re = Re + c.Re;    // Re used without an object refers
   Im = Im + c.Im;    // to object used in call b.plus(C)
   return *this;      // Method for returning operations on
                      // base complex number, b
}
complex add( complex c1, complex c2 )
{
   complex c3;
   c3.Re = c1.Re + c2.Re;    // Add two complex numbers and
   c3.Im = c1.Im + c2.Im;    // return the result.  Use:
   return c3;                // complex z = add( a, b );
}
```

22

```
complex complex::operator+( complex c )
{
          // Add two complex numbers by usual addition
          // operator.  Use in code: complex u, v, w;
          // w = u + v;  // Adds u and v; result in w
   complex c2;
   c2.Re = Re + c.Re;
   c2.Im = Im + c.Im;
   return c2;
}
complex complex::operator*( complex c )
{
          // Multiply two complex numbers by usual
          // operator.  Use in code: complex u, v, w;
          // w = u * v;  // Multiply u and v; result in w
   complex c2;
   c2.Re = Re * c.Re - Im + c.Im;
   c2.Im = Re * c.Re + Im + c.Im;
   return c2;
}
```

23

```
complex operator*( complex c, double d )
{
              // Multiply a complex number by scalar.
              // Used when scalar follows the = sign

   complex c2;
   c2.Re = d * c.Re;
   c2.Im = d * c.Im;
   return c2;
}
complex operator*( double d, complex c )
{
             // Multiply a complex number by scalar.
             // Used when scalar follows the = sign

   complex c2;
   c2.Re = d * c.Re;
   c2.Im = d * c.Im;
   return c2;
}
```

24

Conclusions

- structs and classes provide powerful tools for implementing problems
- Can simplify code and provide natural mathematical language for user
- C++ has a standard template library (STL) for many common structures, e.g. vectors, complex numbers, and strings
- Can use library functions without being concerned about their internal structure

California State University Northridge 25