Introduction to Structures and Classes in C++

Larry Caretto
Computer Science 106

Computing in Engineering and Science

May 11, 2006

California State University Northridge

Schedule

- Today and Tuesday: Lecture on classes
- Thursday (May 18)
  - 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

Outline

- Introduction to C/C++ structured variables (structs)
- Expansion of structs to classes
- Classes *versus* objects of a class
- Example of a class to be used for calculations with complex numbers
  - Definition of class
  - Use of class
  - Details of class functions

Goals for This Topic

- Provide introduction to structs and classes used in object-oriented programming
- Show how functions similar to those we have used for input/output (e.g., .good(), .open(), etc.) are written
- Give background appropriate for
  - students interested in further study of C++
  - Individuals working on programs that use structs and classes
- Topics will not be covered on final

What is a Structure?

- Previously, we have used simple variables and arrays
- Arrays as example of a data structure
  - x is array x[i] is array element
  - all elements must have the same type
- What if we want a structure to represent data for each student at CSUN
- We would want several pieces of information, name, ID, etc. with different data types

Another Example of a Structure

- In exercise eight we had data on x and y for which we wanted to compute the count, mean, standard deviation, max and min
- We could declare a structure to hold the data and the values to be computed.
- We would then declare different variables as having the type named by the struct
- Each struct variable has each component defined in the structure

Definition of a Structure

```
struct dataSet
{
  double data[MAX_DATA]; // all data
                         // values
   int count;    // number of values
   double mean;  // mean value
   double max;   // maximum value
   double min;   // minimum value
   double stdDev;// standard deviation
};               // note ; at end
```

- dataSet is name of struct (like a type)

California State University Northridge 7

Use of a Structure

- When we define a structure like `dataSet`, we have a user defined type (UDT)
- We can declare variables of type `dataSet`, and each variable will have all the fields listed in the structure
- To refer to a field we use the name of the variable, followed by a period, followed by the component name
- See examples on the next chart

California State University Northridge 8

Use of a Structure II

```
struct dataSet  {
   double data[MAX_DATA]; int count;
   double mean;           double max;
   double min;          double stdDev;
};
dataSet x, y, xlow, ylow, xhigh,
        yhigh;   // declare variables
getInput ( x.data, x.count )
getInput ( y.data, y.count )
x.mean = getAverage( x.data, x.count )
```

- Each variable has all the fields defined in the structure used as *<variable>.<field>*

California State University Northridge 9

Use of a Structure III

```
dataSet x, y, xlow, ylow, xhigh,
        yhigh;   // declare variables
getInput ( x.data, x.count )
getInput ( y.data, y.count )
x.mean = getAverage( x.data, x.count )
```

- Can pass entire structures or individual fields to functions
- Note that struct dataSet is abstract concept until variables with that struct are created
- Look at simple example of a struct with count = 3 and data[] = { -1, 0, 1}

California State University Northridge 10

Struct Example

x.data[0]	-1
x.data[1]	0
x.data[2]	1
x.data[3] to x.data[MAX_DATA-1]	undefined
x.count	3
x.mean	0
x.max	1
x.min	-1
x.stdDev	1

California State University Northridge 11

From structs to classes

- Classes are similar to structs
  - Abstract data type that defines a structure
  - Variables are declared that belong to a class
- In addition to having data values, classes can have functions that operate on data
- Can "hide" data from user in classes
  - User cannot change data by accident
  - User can only access data through functions
  - Use can change data in ways allowed by functions for the class

California State University Northridge 12

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

California State University Northridge 13

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

California State University Northridge 14

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

California State University Northridge 15

Complex Number, z

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

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

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

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

California State University Northridge 16

Complex Number Operations

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

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

$$z_3 = z_1 z_2 \Rightarrow \begin{cases} x_3 = x_1x_2 - y_1y_2 \\ y_3 = y_1x_2 + y_2x_1 \end{cases}$$

$$z_3 = \frac{z_1}{z_2} \Rightarrow \begin{cases} \left(x_2^2 + y_2^2\right)x_3 = x_1x_2 + y_1y_2 \\ \left(x_2^2 + y_2^2\right)y_3 = y_1x_2 - y_2x_1 \end{cases}$$

California State University Northridge 17

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

California State University Northridge 18

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

19

```
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 ); }
```

20

```
  // 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!
```

21

Code using 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

22

```
          // main shows use of the complex class
    #include "complex.h"
    DEGREES_PER_RADIAN = 45 / atan(1.0); // 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. */
```

23

```
   // 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)
```

24

```
      // 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.
```

25

```
   //   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
```

26