CSE 143

Introduction to C++ Classes and User-Defined Data Types

[pp 125-141 (skim the example), appendix A]
[Complex class from lecture on CSE143 web]

Complex Numbers in C

- To use Complex numbers in C, we'd might have...

```
typedef struct {  /* rectangular rep */
   double re, im; /* of complex */
} Complex;
...
Complex x,y,z;
x = makeComplex(0,1);
y = makeComplex(17,42);
z = Cadd(x,y);
```

- Very clumsy compared to built-in numeric types.
- C++ solution: classes. A mechanism to define new types, their representations, and operations on them.

C++ Classes

- Class Declaration (in a .h file)

```
class ClassName {
public:
   information available to clients
private:
   private information
};
```

- Private information is only accessible in routines that are given permission in the class declaration
  - Limits region of program with access to implementation details
  - Details may be changed without affecting clients as long as public interface remains the same (but may need to recompile)

Complex Numbers in C++

```
// complex.h -- simple complex number class
class Complex {
public:
    // constructors:
    Complex( );                    // = 0+0i
    Complex(double a, double b);   // = a+bi

    // double->Complex constructor
    Complex(double r);             // = r+0i

    // component access
    friend double real(Complex);
    friend double imag(Complex);
```


Complex Numbers (cont)

```
 // operations
 friend Complex operator+(Complex,Complex);
 friend Complex operator-(Complex,Complex);
 friend Complex operator-(Complex);
 friend Complex operator*(Complex,Complex);

 friend int operator==(Complex,Complex);
 friend int operator!=(Complex,Complex);
  ...
private:
 double re, im;    // representation:
                   // rectangular real and
                   // imaginary parts
};
```


Using Complex

- The Complex class is used like this:

```
#include "complex.h"
...
Complex a = 2.3;     // i.e., 2.3+0i
Complex b = 2*a;
Complex c = Complex(0,1);  // 0+1i
Complex f;                 // 0+0i

f = (2*a+b)*Complex(3,-1);
cout << "the answer is " << f << endl;
```

- The new definitions of operators +, -, *, ==, !=, and << *overload* (not replace) the existing definitions.
- Implicit conversions happen if appropriate constructors are available: `2*a` means `Complex((double)2)*a`.

Friend Functions

- Ordinary functions; definitions placed in .cpp implementation files as usual
- But: have access to private class info because named as friends in the class declaration

```
// complex.cpp -- Complex class impl.
#include "complex.h"
// = real part of c
double real(Complex c) { return c.re; }
// = sum of c and d
Complex operator+(Complex c, Complex d){
   return Complex(c.re+d.re, c.im+d.im);
...
```


Constructors

- Special functions executed every time a variable with a class type is created
- Cannot be executed in any other situation
- Name is same as name of the class
- No explicit return type
- A place to assign sensible initial values to fields of new variable
- Direct access to fields of new variable via field names (the variable is not a parameter)
- Also used to create anonymous values of the class type
- May have several constructors in a class - correct one picked based on argument types

Constructor declaration

- Located in class declaration in header file

```
// complex.h -- simple complex number class
class Complex {
public:
    // constructors:
    Complex( );                    // = 0+0i
    Complex(double a, double b);   // = a+bi

    // double->Complex constructor
    Complex(double r);             // = r+0i
...
};
```


Constructor Implementation

- In .cpp implementation file as usual, but…
- Unusual syntax, because not ordinary function.

```
// in complex.cpp…
// constructors:
Complex::Complex( ) {re=0; im=0;}
Complex::Complex(double r, double i){
   re = r; im = i;
}
Complex::Complex(double r) {
   re = r; im = 0;
}
```

- Details about "::" notation to be revealed later.

Stream Output for Complex

- Operator << can also be overloaded for new types
- Arcane syntax - imitate blindly for now; to be explained later
- Declaration in file Complex.h:

```
// write Complex number as <real,imag>
friend
ostream& operator<<(ostream &s,Complex z);
```

Only part of prototype that varies for new type

Stream Output for Complex

- Implementation in complex.cpp

```
// write Complex number as <real,imag>
ostream& operator<<(ostream &s,Complex z){
   return s << '<' << z.re << ','
            << z.im << '>';
}
```

Only parts of << implementation that depend on the type of the item

- **operator<<** has access to private parts of z because it is a **friend** of class Complex
- Must have **#include <iostream.h>** in.cpp file for this to work.