Operator Overloading

Your Turn

```
class Rational
{
public:
   Rational( int = 0, int = 1 );
   Rational addition( const Rational & );
   Rational subtraction( const Rational & );
   Rational multiplication( const Rational & );
   Rational division( const Rational & );
   void printRational ();
private:
   int numerator;
   int denominator;
   void reduction();
};
```

- Replace the printRational() function with operator<<

Overloading Binary Operators

- Examples of binary operators that can be overloaded are +, -, *, and /

- Unlike the insertion and extraction operators that are overloaded as friend functions, the binary operators are overloaded as regular member functions of the class

Example

- Let us add functionality to the Rational class to support the following:

```
Rational a( 3, 4 );

Rational b( 2, 9 );

Rational c;

c = a + b;
```

Member Function Prototype

- In the class interface, let us add the function prototype for the overloaded operator

```
RationalNumber operator+( const
                            RationalNumber & );
```

Member Function Definition

```
RationalNumber operator+( const
                            RationalNumber & r)
{
  RationalNumber add;
  add.numerator = numerator * r.denominator +
                  denominator * r.nominator;
  add.denominator = denominator * r.denominator;
  add.reduction();
  return add;
}
```

Your Turn

- Overload the multiplication operator in the rational class

Topics for Test

- Constant pointers and pointers to constant data
- Basics of classes; data members, member functions, private, and public
- Access and utility functions (get and set)
- Constructors, destructors, default arguments in constructors, constructor initializer list

Topics for Test

- Returning a reference to a private data member (p. 445)
- const objects and member functions
- const data members
- Composition: objects as members of a class
- Friend functions
- this pointer
- Static

Topics for Test

- Dynamic memory allocation (new, delete)
- Basic operator overloading