# `fraction` VERSION 2

Overloaded Operators Version

Delroy A. Brinkerhoff

# CHAPTER 9 fraction CLASS

```
class fraction
{
    private:
        int      numerator;
        int      denominator;

    public:
        fraction(int n = 0, int d = 1);
        fraction add(fraction f2) const;
        fraction sub(fraction f2) const;
        fraction mult(fraction f2) const;
        fraction div(fraction f2) const;
        void     print() const;
        void     read();
};
```

# THE fraction CLASS SPECIFICATION

```
class fraction
{
    private:
        int     numerator;
        int     denominator;

    public:
                fraction(int n = 0, int d = 1);
        friend fraction operator+(fraction f1, fraction f2);
        friend fraction operator-(fraction f1, fraction f2);
        friend fraction operator*(fraction f1, fraction f2);
        friend fraction operator/(fraction f1, fraction f2);
        friend ostream& operator<<(ostream& out, fraction& f);
        friend istream& operator>>(istream& in, fraction& f);
    private:
        void    reduce();
        int     gcd(int, int);
};
```

## HELPER FUNCTIONS

### GREATEST COMMON DIVISOR
### gcd

- Finds the greatest common divisor of two integers
  - gcd(8, 12) = 4
  - gcd(8, 16) = 8
- Implemented with iteration or recursion

### reduce

```
void fraction::reduce()
{
    int common = gcd(numerator, denominator);
    numerator /= common;
    denominator /= common;
}
```

## THE fraction CONSTRUCTOR

```
fraction::fraction(int n, int d)
    : numerator(n), denominator(d)
{
    reduce();
}
```

- `fraction f;`
- `fraction f(5);`
- `fraction f(2, 3);`

## ADDITION AND SUBTRACTION

```
fraction operator+(fraction f1, fraction f2)
{
    int    n = f1.numerator * f2.denominator +
                f2.numerator * f1.denominator;
    int    d = f1.denominator * f2.denominator;

    return fraction(n, d);
}
```

## MULTIPLICATION AND DIVISION

```
fraction operator*(fraction f1, fraction f2)
{
    int    n = f1.numerator * f2.numerator;
    int    d = f1.denominator * f2.denominator;

    return fraction(n, d);
}
```

## I/O OPERATORS

```
ostream& operator<<(ostream& out, fraction& f)
{
    cout << endl << f.numerator << "/" << f.denominator << endl;
    return out;
}

istream& operator>>(istream& in, fraction& f)
{
    cin >> f.numerator;
    cin >> f.denominator;
    f.reduce();
    return in;
}
```

# EXCERPTS FROM A FRACTION CALCULATOR

```
input(left, right);

void input(fraction& l, fraction& r)
{
    cout << "Left-hand fraction";
    cin >> l;
    cout << "Right-hand fraction";
    cin >> r;
}
```

```
fraction    left;
fraction    right;
fraction    result;

result = left + right;
result = left - right;
result = left * right;
result = left / right;

cout << result << endl;
```