```
//  IMPLEMENTATION FILE (fraction.cpp)
//  This module exports a FracType class
//  with two operator functions,
//  operator* and operator==.

#include <iostream.h>
#include <stdlib.h>     // For abs() function
#include "fraction.h"

int GreatestCommonDivisor( int, int );
// Auxiliary function prototype

// Private members of class:
//     int numer;
//     int denom;
//
// CLASSINV: denom > 0

FracType::FracType( /* in */ int initNumer,
                    /* in */ int initDenom )
// Constructor
// PRE:  Assigned(initNumer)  &&  initDenom > 0
// POST: numer == initNumer  &&  denom == initDenom
{
   numer = initNumer;
   denom = initDenom;
}
```

```
void FracType::Write() const
// POST: Fraction has been displayed as
// numer/denom with no blanks
{
     cout << numer << '/' << denom;
}

float FracType::FloatEquiv() const
// POST: FCTVAL == float equivalent of this fraction
{
   return float(numer) / float(denom);
}

void FracType::Simplify()
// POST: Fraction is reduced to lowest terms. (No integer > 1
//       evenly divides both numer and denom)
{
   int gcd;
   int absNumer = abs(numer);

   if (numer==0 || absNumer==1 || denom==1)
      return;
   gcd = GreatestCommonDivisor(absNumer, denom);
   if (gcd > 1) {
      numer /= gcd;
      denom /= gcd;
   }
}
```

```
FracType FracType::operator*( /* in */ FracType frac2 ) const
// PRE:  This fraction and frac2 are in simplest terms
// POST: FCTVAL == this fraction * frac2
// (fraction multiplication)
//      (WARNING: Overflow is possible)
{
   int resultNumer = numer * frac2.numer;
   int resultDenom = denom * frac2.denom;

   FracType resultFrac(resultNumer, resultDenom);
   // ASSERT: New fraction created

   resultFrac.Simplify();
   return resultFrac;
}

Boolean FracType::operator==( /* in */ FracType frac2 ) const
// PRE:  This fraction and frac2 are in simplest terms
// POST: FCTVAL == TRUE, if this fraction == frac2
//             == FALSE, otherwise
{
   return (numer==frac2.numer) && (denom==frac2.denom);
}
```

```
int GreatestCommonDivisor( /* in */ int a,
                           /* in */ int b )
// PRE:  a >= 0  &&  b > 0
// POST: FCTVAL == Greatest common divisor of a and b
//              (Algorithm: the Euclidean algorithm)
{
   int temp = a % b;
   while (temp > 0) {  // INV (prior to test):
                       //    No integer > b evenly divides
                       //    both a<entry> and b<entry>
      a = b;
      b = temp;
      temp = a % b;
   }
// ASSERT: b == Greatest common divisor of
// a<entry> and b<entry>
   return b;
}
```