Estructuras de Datos y de la Información

Práctica 1

Calculadora de Números Racionales

OBJETIVO: El objetivo de la práctica es implementar un tipo de datos definido por el usuario y utilizarlo en un programa en lenguaje C++.

FECHA: Está práctica se entregará los días 16, 17 y 18 de noviembre en el horario asignada a cada grupo.

ENUNCIADO: Partiendo de la clase `Racional` dada en los apuntes, se quiere programar una calculadora de números racionales que implemente las operaciones: suma, resta, producto y cociente.

El programa empieza solicitando un número racional y lo visualiza en la siguiente línea.

```
$> calculadora_racional
Inicio
Introduzca primer operando:  1 / 2
Resultado = 1 / 2
```

A continuación el programa solicita un operador.

```
Introduzca un operador (+ - * /): +
```

Y luego solicita otro número racional y visualiza el resultado.

```
Introduzca segundo operando:  1 / 3
Resultado = 5 / 6
```

El programa continuará solicitando un operador y el segundo operando, realiza la operación con el resultado acumulado y visualiza el nuevo resultado, hasta que se introduce un operador no reconocido.

```
Introduzca un operador (+ - * /): *
Introduzca segundo operando:  2 / 3
Resultado = 10 / 18
Introduzca un operador (+ - * /): a
Fin
```

NOTAS DE IMPLEMENTACION:

1. Generar los ficheros `Racional.h` y `Racional.C` con la declaración de la clase `Racional`.
2. Implementar los métodos `resta` y `cociente` de números racionales.
3. Generar el fichero `calculadora.C` con la implementación de la práctica.
4. Utilizar el `makefile` dado en la práctica 0 para compilar y generar el ejecutable.

A continuación se adjuntan los códigos vistos en clase.

```
//*************************************************************************
//
// Copyright (c) 1997-1999.
//      Richard D. Irwin, Inc.
//
// This software may not be distributed further without permission from
// Richard D. Irwin, Inc.
//
// This software is distributed WITHOUT ANY WARRANTY. No claims are made
// as to its functionality or purpose.
//
// Authors: James P. Cohoon and Jack W. Davidson
// Date: 7/1/98
// $Revision: 1.3 $
// $Name: E2 $
//
//*************************************************************************

// racional.h: Declaración del ADT Racional
#ifndef RACIONAL_H
#define RACIONAL_H

#include <iostream>
#include <string>

using namespace std;

// ADT Racional: descripción de la clase
class Racional {
      public: // funciones miembro
            // constructor por defecto
            Racional();     // Los constructores no indican valor retornado
            // un segundo constructor
            Racional(int numer, int denom = 1);
            // algunas utilidades aritméticas y de stream
            Racional Sumar(const Racional &r) const;
            Racional Multiplicar(const Racional &r) const;
            // Se podría añadir Restar, Dividir y operaciones relacionales
            void Insertar(ostream &sout) const;
            void Extraer(istream &sin);
      protected:
            // inspectors
            int ObtenerNumerador() const;
            int ObtenerDenominador() const;
            // mutators
            void FijarNumerador(int numer);
            void FijarDenominador(int denom);
      private:
            // miembros dato
            int ValorNumerador;
            int ValorDenominador;
};

// ADT Racional: descripción de operador auxiliares
Racional operator+(const Racional &r, const Racional &s);
Racional operator*(const Racional &r, const Racional &s);

ostream& operator<<(ostream &sout, const Racional &s);
istream& operator>>(istream &sin, Racional &r);

#endif
```

```
//*************************************************************************
//
// Copyright (c) 1997-1999.
//      Richard D. Irwin, Inc.
//
// This software may not be distributed further without permission from
// Richard D. Irwin, Inc.
//
// This software is distributed WITHOUT ANY WARRANTY. No claims are made
// as to its functionality or purpose.
//
// Authors: James P. Cohoon and Jack W. Davidson
// Date: 7/1/98
// $Revision: 1.3 $
// $Name: E2 $
//
//*************************************************************************
// racional.C: Declaración del ADT Racional
#include <iostream>
#include <string>
#include "racional.h"

using namespace std;

// Constructor por defecto
Racional::Racional() {
      FijarNumerador(0);
      FijarDenominador(1);
}

// constructor (numerador, denominador)
Racional::Racional(int numerador, int denominador) {
      FijarNumerador(numerador);
      FijarDenominador(denominador);
}

// Obtener El numerador
int Racional::ObtenerNumerador() const {
      return ValorNumerador;
}

// Obtener el denominador
int Racional::ObtenerDenominador() const {
      return ValorDenominador;
}

// Fijar el  numerador
void Racional::FijarNumerador(int numerador) {
      ValorNumerador = numerador;
}

// Fijar el  denominador
void Racional::FijarDenominador(int denominador) {
      if (denominador != 0) {
            ValorDenominador = denominador;
      }
      else {
            cerr << "Denominador ilegal: " << denominador
            << "se usa  1" << endl;
            ValorDenominador = 1;
      }
}
```

```
// Suma de números racionales
Racional Racional::Sumar(const Racional &r) const {
      int a = ObtenerNumerador();
      int b = ObtenerDenominador();
      int c = r.ObtenerNumerador();
      int d = r.ObtenerDenominador();
      return Racional(a*d + b*c, b*d);
}

// Producto de números racionales
Racional Racional::Multiplicar(const Racional &r) const {
      int a = ObtenerNumerador();
      int b = ObtenerDenominador();
      int c = r.ObtenerNumerador();
      int d = r.ObtenerDenominador();
      return Racional(a*c, b*d);
}

// Inserción de un número racional
void Racional::Insertar(ostream &sout) const {
      sout << ObtenerNumerador() << '/' << ObtenerDenominador();
      return;
}

// Extracción de un número racional
void Racional::Extraer(istream &sin) {
      int numerador;
      int denominador;
      char slash;
      sin >> numerador >> slash >> denominador;
      FijarNumerador(numerador);
      FijarDenominador(denominador);
      return;
}

// Suma de números racionales
Racional operator+(const Racional &r, const Racional &s) {
      return r.Sumar(s);
}

// Producto de números racionales
Racional operator*(const Racional &r, const Racional &s) {
      return r.Multiplicar(s);
}

// Inserción de un número racional
ostream& operator<<(ostream &sout, const Racional &r) {
      r.Insertar(sout);
      return sout;
}

// Extracción de un número racional
istream& operator>>(istream &sin, Racional &r) {
      r.Extraer(sin);
      return sin;
}
```