Operaciones con fracciones

El objetivo de este control es implementar un TAD sencillo para el manejo de fracciones.

1) Conocimiento del dominio

Como es conocido desde la escuela primaria, las fracciones constituyen una forma de representar números racionales mediante ratios de números enteros. La notación más habitual seguida para escribir una fracción es $\frac{a}{b}$. En esta notación, *a* (el *numerador* de la fracción) y *b* (el *denominador*) son números enteros, $\boldsymbol{b \neq 0}$, y la fracción representa el número racional que resulta de dividir *a* entre *b*.

Una fracción $\frac{c}{d}$ se dice *irreducible* cuando: (i) *d* es positivo, y (ii) el máximo común divisor de *c* y *d* es 1. Nótese, entonces, que, para comprobar la igualdad entre fracciones irreducibles, basta comprobar la igualdad entre numeradores y entre denominadores. En términos más formales:

$$\frac{a}{b}\ irreducible \wedge \frac{c}{d}\ irreducible \Rightarrow (\frac{a}{b} = \frac{c}{d} \iff a=c \wedge b=d\ ).$$

Por último, la suma, resta, multiplicación y división de fracciones se definen como sigue:

- $\frac{a}{b} + \frac{c}{d} = \frac{a\frac{mcm(b,d)}{b} + c\frac{mcm(b,d)}{d}}{mcm(b,d)}$, con *mcm*(*b,d*) el *mínimo común múltiplo* de *b* y *d*.
- $\frac{a}{b} - \frac{c}{d} = \frac{a\frac{mcm(b,d)}{b} - c\frac{mcm(b,d)}{d}}{mcm(b,d)}$
- $\frac{a}{b} \times \frac{c}{d} = \frac{ac}{bd}$
- $\frac{a}{b} : \frac{c}{d} = \frac{ad}{bc}$

2) Detalles de la implementación

El TAD se implementará mediante una clase `Racional`. Las fracciones se representarán mediante un par de enteros de tipo **long**, uno correspondiente al *numerador* (lo llamaremos *a*) y el otro al *denominador* (lo llamaremos *b*). Así mismo:

- El invariante de la representación exige que: (i) $b \neq 0$, y (ii) $\frac{a}{b}$ es irreducible.
- La implementación debe incluir las siguientes constructoras:
  - Una constructora por defecto `Racional()`, que construye la fracción $\frac{0}{1}$
  - Una constructora `Racional(`n`,`d`)`, que construye la fracción irreducible equivalente a $\frac{n}{d}$. Se exige, como precondición, que $d \neq 0$. En caso de que se viole dicha precondición, la constructora debe levantar una excepción de tipo `EDenominadorCero`.
- Debe incluir, así mismo, las siguientes operaciones (en la descripción de cada operación, se supone que $\frac{a}{b}$ es la fracción sobre la que actúa la operación):
  - *Suma*. Toma como argumento una fracción $\frac{c}{d}$ y devuelve la fracción que resulta al reducir $\frac{a}{b} + \frac{c}{d}$. Debe implementarse mediante un método `suma`.
  - *Resta*. Toma como argumento una fracción $\frac{c}{d}$ y devuelve la fracción que resulta al reducir $\frac{a}{b} - \frac{c}{d}$. Debe implementarse sobrecargando el operador binario `-`.
  - *Multiplicación y actualización*. Toma como argumento una fracción $\frac{c}{d}$ y actualiza $\frac{a}{b}$ a la fracción que resulta al reducir $\frac{a}{b} \times \frac{c}{d}$. Esta mutadora se deberá implementar sobrecargando el operador `*=`.
  - *División y actualización*. Toma como argumento una fracción $\frac{c}{d}$ y actualiza $\frac{a}{b}$ a la fracción que resulta al reducir $\frac{a}{b} : \frac{c}{d}$. Se exige como precondición que $\frac{c}{d} \neq \frac{0}{1}$. En caso de que se viole dicha condición, deberá deberá levantar una excepción de tipo `EDivisionPorCero`. Esta mutadora deberá implementarse mediante un método `divideYActualiza`.

- *Igualdad*. Esta operación, que implementa la función de equivalencia del TAD, toma como argumento una fracción $\frac{c}{d}$ y comprueba si $\frac{a}{b} = \frac{c}{d}$. Debe implementarse sobrecargando el operador ==.

3) **Trabajo a realizar**

Para realizar el control se proporcionan los siguientes archivos:

- `main.cpp`. Este archivo contiene la función `main` que realiza las pruebas de la implementación del TAD. Para ello procesa un archivo que contiene, en cada línea, la directiva para realizar una determinada operación con fracciones. Las posibles directivas tienen alguna de las siguientes formas:

  `+` a/b c/d
  `-` a/b c/d
  `*` a/b c/d
  `/` a/b c/d
  `==` a/b c/d

  donde *a*, *b*, *c* y *d* son enteros en el rango de **long**, cada a/b representa la fracción $\frac{a}{b}$ y cada c/d la fracción $\frac{c}{d}$. No se asegura que dichas fracciones estén reducidas. Tampoco se asegura que sean correctas (*b* o *d* pueden ser 0).
  El programa procesa las directivas en orden, e imprime el resultado de realizar la operación indicada (o bien un código de error, en caso de que se haya producido un error durante el procesamiento). A continuación, se muestra un ejemplo de entrada procesable por este programa, y de la salida producida:

Entrada	Salida
`+ 5/2 1/2`	`3/1`
`- 5/3 1/5`	`22/15`
`* 2/4 18/6`	`3/2`
`/ 2/4 0/1`	`DIVISION_POR_CERO`
`* 3/2 1/0`	`ERROR_EN_ARGUMENTO`

  Importante: **No se permite realizar ningún tipo de modificación en el archivo** `main.cpp` (en éste se implementa ya completamente el proceso descrito). No obstante, el uso que se hace de las operaciones del TAD en dicho archivo puede clarificar la manera de implementar dichas operaciones (como método convencional, como operador sobrecargado, etc.). Por tanto, se recomienda leer su contenido antes de realizar el resto de actividades.
- `Racional.h`. Este archivo define la clase `Racional`. Se proporciona una definición parcial de la misma. **Deben declararse los métodos correspondientes a las operaciones del TAD, siguiendo las indicaciones dadas** (el código en `main.cpp` depende de que se sobrecarguen adecuadamente los operadores indicados, y de que se implementen las otras operaciones como métodos convencionales).
- `Racional.cpp`. Este archivo contiene la implementación de los métodos de `Racional`. **Deben implementarse los métodos públicos correspondientes a las operaciones**. Para facilitar esta tarea se proporcionan, ya implementados, diferentes métodos privados (entre ellos, el método `reduce`, que convierte la fracción a una irreducible equivalente, o el método `mcm`, que calcula el mínimo común múltiplo de dos enteros).