Práctica 8. Tipos Abstractos de Datos

Como hemos visto en clase de teoría, para definir un tipo abstracto de datos (TAD) debemos dar su especificación (donde se establece el comportamiento del TAD, es decir donde se proporcionan los valores, operaciones y propiedades del TAD) y su implementación (donde se determina la representación de los valores del tipo y se codifican las operaciones en base a la representación dada). En esta práctica veremos como definir TADs en C++.

Para definir un TAD en C++ tenemos que proporcionar dos ficheros:

- un fichero de cabecera que tiene extensión .h y donde se proporciona la especificación del TAD; y
- un fichero de implementación con extensión .cpp donde se proporciona la implementación del TAD.

La razón para tener estos dos ficheros es que el fichero de cabecera va a permitir a otros programas interactuar con nuestro TAD sin necesidad de saber cómo está implementado. Esto permite intercambiar la implementación del TAD sin que afecte a los programas que hacen uso de dicho TAD.

Ejercicio 1. El TAD Racional

El primer TAD que vamos a definir es el *TAD racional* que tiene la siguiente especificación.

```
void crear(int a, int b, racional & r);
{Pre: b distinto de 0}
{Post: crea el racional r con numerador a y denominador b}

int numerador(racional r);
{Pre: }
{Post: devuelve el numerador del racional r}

int denominador(racional r);
{Pre: }
{Post: devuelve el denominador del racional r}

void irreducible(racional r, racional &rr);
{Pre: }
{Post: crea rr como el racional irreducible de r}
```

Vamos a definir este TAD en C++, para ello sigue los siguientes pasos:

1. Crea un nuevo proyecto de consola en Codeblocks.
2. Añade un nuevo fichero de cabecera al proyecto que has creado. Para ello ve a *File → New → File…* En la ventana que aparece selecciona el tipo de fichero *C/C++ header* y pulsa el botón *Go*. En la nueva ventana que aparece debes proporcionar el nombre del fichero (llamalo *racional*) con el path completo y añade el fichero al proyecto marcando las casillas *Debug* y *Release*. **Nota.** Es importante que marques las dos casillas anteriores, de lo contrario aparecerán luego problemas.
3. El paso anterior crea un nuevo fichero llamado *racional.h*. En este fichero debes definir el tipo de los racionales (puedes usar un registro con dos componentes, un vector, etc.) y proporcionar las cabeceras de las operaciones indicadas en la especificación del TAD. **Nota.** En este fichero sólo se proporcionan las cabeceras de las operaciones, no la implementación de las mismas.
4. Para proporcionar la implementación de nuestro TAD, añade un nuevo fichero fuente al proyecto que has creado. Para ello ve a *File → New → File…* En la ventana que aparece selecciona el tipo de fichero *C/C++ source* y pulsa el botón *Go*. En la nueva ventana que aparece debes proporcionar el nombre del fichero (llamalo *racional*) con el path completo y añadir el fichero al proyecto marcando las casillas *Debug* y *Release*. **Nota.** Al igual que antes es necesario que marques las dos casillas anteriores.
5. El paso anterior crea un nuevo fichero llamado *racional.cpp*. En este fichero debes implementar las operaciones del TAD. La primera línea de este fichero debe ser la inclusión del fichero de cabecera, es decir: `#include "racional.h"`. **Nota**. No copies directamente el anterior `include` en tu código ya que C++ no reconoce bien las comillas al pegarlas desde un fichero de texto, así que es mejor que lo escribas directamente.

Una vez que hemos definido el TAD anterior, vamos a ver cómo utilizarlo. Para ello abre el fichero *main.cpp* del proyecto e incluye el fichero de cabecera del TAD racional en l proyecto. Es decir, añade la línea `#include "racional.h"`. Esto te dará acceso a todas las operaciones definidas en el TAD racional. A continuación crea los siguientes subprogramas que utilizan el TAD racional:

1. Construir un subprograma que lea un racional del teclado.
2. Construir un subprograma que muestre por pantalla un racional.
3. Construir un subprograma que sume dos raciones.
4. Construir un subprograma que multiplique dos racionales.
5. Construir un subprograma que decida si dos racionales son o no el mismo.
6. Construir un subprograma que lea del teclado una secuencia de racionales (terminada en el racional 0) y calcule la suma.

Importante. En la implementación de los subprogramas indicados anteriormente no debes usar la implementación concreta del TAD racional, sino que sólo debes utilizar las operaciones. O dicho de otra manera, si se cambia la implementación del TAD racional, los subprogramas anteriores deberían seguir funcionando.

Ejercicio 2. El TAD punto y el TAD circunferencia

En esta segunda parte de la práctica vamos a crear un nuevo proyecto donde utilizaremos el TAD punto y el TAD circunferencia.

Apartado 2.1. Definir un TAD en C++ para representar puntos del plano. Para ello, dar la especificación y una implementación. La especificación debe contener operaciones para:

1. Construir un punto a partir de una abscisa y ordenada.
2. Obtener la abscisa de un punto.
3. Obtener la ordenada de un punto.
4. Calcular la distancia de un punto al origen.

Apartado 2.2. Definir un TAD en C++ para representar circunferencias en el plano. La especificación de este TAD debe contener operaciones para crear una circunferencia y acceder a las componentes de la misma.

Apartado 2.3. Diseñar los siguientes subprogramas que utilizan el TAD punto y el TAD circunferencia.

1. Diseñar un subprograma que decida si dos circunferencias son concentricas.
2. Diseñar un subprograma que decida si un punto está dentro de una circunferencia.