Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza
1542

Práctica 1

Tecnología de Programación
Grado en Ingeniería Informática

Objetivos

- Recordar los conceptos básicos de la programación imperativa en C++.
- Presentar los aspectos básicos de la sintaxis en programación orientada a objetos mediante la implementación de tipos abstractos de datos sencillos.
- Comparar la implementación de un tipo de datos mediante la programación imperativa pura y la programación orientada a objetos.
- Presentar la definición de operadores en C++.

En esta práctica vas a implementar una sencilla calculadora de fracciones, mediante la implementación de un tipo de datos **`Rational`**.

La implementación se hara de dos formas distintas, mediante un **TAD** con las técnicas que ya conoces, y transformando ese TAD a una **clase** de C++, con su correspondiente implementación siguiendo la notación de la Programación Orientada a Objetos.

1. TAD en C++

Para implementar esta tarea debes partir de los ficheros que te facilitamos en la carpeta `rational-struct` del material de la práctica.

1.1. Tarea 1

Define e implementa mediante un registro (`struct`) un TAD **`Rational`** que represente una fracción con las operaciones:

- Inicialización con un valor por defecto de 0/1.
- Inicialización a partir de dos enteros (numerador y denominador).
- Lectura de un stream.
- Escritura en un stream.
- Operaciones aritmeticas de suma, resta, multiplicación y división entre racionales.
- Operaciones aritmeticas de suma, resta, multiplicación y división entre racionales y números enteros, en cualquier orden.

Despues de la inicialización y de cualquiera de las operaciones el número racional debe quedar simplificado.

Utiliza las técnicas que ya conoces (miembros privados, funciones amigas) para limitar el acceso a la información interna de tu TAD.

1.2. Tarea 2

Diseña un programa principal que pruebe todos los métodos implementados: una calculadora que iterativamente, dado un racional, un operador y otro racional, muestre el resultado de aplicar la operación a los dos operandos:

```
~> main ↵
? 5/6 + 1/4
= 13/12
? 1/3 + 1/2
= 5/6
? .
~>
```

El programa termina cuando se introduce cualquier dato erróneo (por ejemplo un '.'). Puedes detectar esa condición mediante la funcion `cin.fail()` (ver `main.cc`).

2. Clase en C++

Para implementar esta tarea debes partir de los ficheros que te facilitamos en la carpeta `rational-class` del material de la práctica.

2.1. Tarea 1

Convierte la implementación del tipo **`Rational`** de un TAD con un registro y funciones que operan sobre él a una **clase** con sus correspondientes métodos:

- No existe función `init(...)`. Casi la totalidad de los lenguajes orientados a objetos permiten la definición de un constructor, que hace las veces de inicializador, que tiene el mismo nombre que la clase y que se ejecuta automáticamente siempre que se crea en memoria el objeto correspondiente.
- Al contrario que en las funciones (en la que el parámetro que representa al TAD se declara explícitamente) en una clase los métodos tienen un parámetro implícito `this`, que es un puntero que apunta al propio objeto y que permite acceder a sus atributos y métodos. En la mayoría de los casos se puede omitir.
- Existen funciones que se pueden convertir en métodos (aquellas en las que el primer parámetro es el objeto) y otras que deben seguir implementándose como funciones externas a la clase. Para las funciones que implementan los operadores aritméticos, deberás decidir que forma de implementación se adapta mejor a cada una.

2.2. Tarea 2

Modifica el programa principal para que utilice la clase **Rational**, con sus correspondientes métodos.

(continúa en la siguiente página...)

EJERCICIO OPCIONAL

3. Operadores

C++ permite la definición de operadores para clases, de forma que el uso de esas clases sea mucho más parecido al de los tipos de datos predefinidos. Esto es especialmente útil para tipos numéricos, como el `Rational` que acabas de implementar. En lugar de escribir código de este tipo:

```
Rational a,b,c,r;
a.read(cin);
b.read(cin);
c.read(cin);
r = a.add( b.mul(c) );
r.write(cout);
cout << endl;
```

podemos definir los operadores adecuados para escribirlo así:

```
Rational a,b,c,r;
cin >> a >> b >> c;
r = a + b*c;
cout << r << endl;
```

Modifica los métodos y funciones necesarios en la clase `Rational` para implementar los operadores aritméticos de suma, resta multiplicación y división (con todas sus variantes), y los operadores de lectura y escritura en *streams*.

Modifica el programa principal para utilizarlos.

Entrega

Los archivos de código fuente de la práctica deberán estar separados en dos subdirectorios diferentes, con la siguiente estructura (XXXXXX es tu NIP, o los NIPs de la pareja de prácticas, ver más adelante):

```
practica1_XXXXXX
     \---- rational-struct
          \---- rational.h
          \---- rational.cc
          \---- main.cc
          \---- Makefile
     \---- rational-class
          \---- rational.h
          \---- rational.cc
          \---- main.cc
          \---- Makefile
```

Cada subdirectorio no deberá incluir otros archivos de código fuente, y los nombres de los archivos serán los indicados en la práctica.

El programa en C++ deberá ser compilable y ejecutable con los archivos que has entregado en cada subcarpeta mediante los siguientes comandos:

```
make
./main
```

En caso de no compilar siguiendo estas instrucciones, el resultado de la evaluación de la práctica será de 0. No se deben utilizar paquetes ni librerías ni ninguna infraestructura adicional fuera de las librerías estándar del propio lenguaje.
Todos los archivos de código fuente solicitados en este guión deberán ser comprimidos en un único archivo `zip` con el siguiente nombre:

- `practica1_<nip1>_<nip2>.zip` (donde `<nip1>` y `<nip2>` son los NIPs de los estudiantes involucrados) si el trabajo ha sido realizado por parejas. En este caso sólo uno de los dos estudiantes deberá hacer la entrega.

- `practica1_<nip>.zip` (donde `<nip>` es el NIP del estudiante involucrado) si el trabajo ha sido realizado de forma individual.

El archivo comprimido a entregar no debe contener ningún fichero aparte de los fuentes que te pedimos: ningún fichero ejecutable o de objeto, ni ningún otro fichero adicional. La entrega se hará en la tarea correspondiente a través de la plataforma Moodle:
`http:://moodle.unizar.es`