Escuela Superior de Ingeniería y Tecnología
Universidad de La Laguna

Grado en Ingeniería Informática
Algoritmos y Estructuras de Datos Avanzadas

Curso 2018-2019

Práctica 3: Representación de números en notación posicional (3)

1. Objetivo

En esta práctica se trabajarán los siguientes conceptos del lenguaje C++: la herencia, el polimorfismo, la sobrecarga de operadores, uso de plantillas y control de excepciones:

2. Entrega

Esta práctica se realizará en dos sesiones de laboratorio en las siguientes fechas:

- Sesión tutorada: del 12 al 14 de marzo de 2019.
- Sesión de entrega: del 19 al 21 de marzo de 2019.

3. Enunciado

A partir de la plantilla `Number` elaborada en las prácticas 1 y 2 para representar números en notación posicional, se quiere generar una nueva versión de la plantilla que heredará de la clase abstracta `NumberBase`.

La clase `NumberBase` contiene dos atributos privados donde almacena el valor de la base (`B`) y el número de dígitos (`N`), respectivamente, para un objeto del tipo `Number<N,B,T>`. El constructor de la clase `NumberBase` comprueba que ambos parámetros tienen valores factibles, esto es, positivos y mayores que 1; y en caso contrario lanzan una excepción `wrong_number_exception` indicando este error en el uso de la plantilla `Number`. Esta excepción deriva de la clase `NumberException` definida en la práctica 2. Además, en la clase `NumberBase` se definen los siguientes métodos nulos:

- Protegido, `void to_base(int)` que convierte el parámetro entero a la representación como secuencia de `N` dígitos en la base `B`. Este método nulo se corresponde con el método de la misma signatura ya implementado en la plantilla `Number`.

- Protegido, `ostream& write(ostream&)const` que inserta en el flujo `ostream` los `N` dígitos en base `B` del número representado. Al igual que con el método anterior, este método indica la generalización del método del mismo nombre implementado en la plantilla `Number`. También se sobrecarga como función amiga al operador de inserción en un flujo para que invoque al método `write`.

- Protegido, `NumberBase* duplicate() const` que genera un duplicado del objeto que lo invoca.

- Público, `NumberBase* operator+(const NumberBase*)const` que calcula y retorna un objeto con el número cuyo valor es el resultado de sumar el número actual y el número pasado por parámetro. Este método utiliza la operación Suma que se implementó en la plantilla `Number`.

- Público, `NumberBase* operator-(const NumberBase*)const` que calcula y retorna un objeto con el número cuyo valor es el resultado de restar el número actual y el número pasado por parámetro. Este método utiliza la operación Resta que se implementó en la plantilla `Number`.

A partir de la clase `Number<N,B,char>` se derivarán las clases `BinaryNumber`, `OctalNumber`, `DecimalNumber` y `HexadecimalNumber` que representan con 8 dígitos a los números en base 2, 8, 10 y 16, respectivamente.

En el programa principal de la práctica se solicita al usuario que especifique una base (2, 8, 10 o 16) con la que desea trabajar y el valor para dos números enteros. Se declaran dos variables de tipo puntero a la clase base `NumberBase`, con los que se apunta a los objetos instanciados en memoria dinámica e inicializados con los valores dados, de la clase que corresponda: `BinaryNumber`, `OctalNumber`, `DecimalNumber` o `HexadecimalNumber`. Se realizan las operaciones implementadas entre ambos números y se muestran los resultados por pantalla. El programa principal capturará cualquier excepción que se produzca e informará de la misma.