Tarea 5: Diseño de clases

Mario Medina
mariomedina@udec.cl

Programación orientada al objeto — 10 de octubre de 2018

Introducción

En esta quinta tarea, Uds. deben experimentar con el desarrollo e implementación de clases y la sobrecarga de operadores.

Info: De más está decirle que esta tarea es individual: puede comentar posibles métodos de solución con sus compañeros, pero se espera que los códigos entregados por todos los alumnos sean diferentes.

Envíeme su código fuente junto con un informe de a lo más 3 planas, detallando su método de solución y los resultados de las cuatro simulaciones el día viernes 19 de octubre, antes de medianoche a `mariomedina@udec.cl`.

Enteros de precisión arbitraria

Los números enteros en C++ son tradicionalmente representados en tipos de datos `long int`, de 32 bits ó `long long int`, de 64 bits. El tipo de dato `long int` puede representar enteros entre −2.147.483.648 y +2.147.483.647, mientras que el tipo de datos `long long int` puede representar enteros entre −9.223.372.036.854.775.808 y 9.223.372.036.854.775.807.

La representación de números enteros de más de 20 dígitos no es posible utilizando estos tipos de datos. Por ello, Ud. debe escribir la clase `BigInteger` que permite representar enteros en forma arbitraria. En esta clase, un número entero se representará como un objeto `string` conteniendo los dígitos del número, y un `char`, conteniendo el signo del número (1 para los números positivos, y −1 para los números negativos).

Su clase debe incluir constructores que permitan crear un objeto `BigInteger` a partir de datos de tipos `char`, `short int`, `long int`, `int` y `long long int`, en sus formas `signed` y `unsigned`. Además, debe incluir constructores que reciban como argumentos datos de tipo `float` y `double`, donde el objeto `BigInteger` construido contiene sólo la parte entera de los argumentos.

Como ejemplo, el codigo `BigInteger bi(-314)` deberá crear un objeto `BigInteger` conteniendo 3 dígitos: 3, 1 y 4, y cuya variable miembro de signo es −1, mientras que el codigo `BigInteger bi(3.14)` deberá crear un objeto `BigInteger` conteniendo sólo el dígito 3 y cuya variable miembro de signo es 1.

No olvide escribir el constructor para copia, el operador de asignación y un destructor para la clase.

Además, Ud. debe implementar las siguientes funciones miembro:

- `int signo() const` que retorna el signo de `BigInteger` como 1 (positivo) ó −1 (negativo).
- `BigInteger abs() const` que retorna un objeto `BigInteger` correspondiente al valor absoluto del objeto actual.
- `int cmp(BigInteger& q) const` que retorna un valor −1, 0 ó 1 si el objeto actual es menor, igual o mayor, respectivamente, al argumento `q`.
- `bool esPar() const` que retorna `true` si el objeto actual es par.
- `bool esImpar() const` que retorna `true` si el objeto actual es impar.
- `bool esChar() const` que retorna `true` si el valor del objeto cabe en una variable `char`.
- `bool esShortInt() const` que retorna `true` si el valor del objeto cabe en una variable `short int`.
- `bool esLongInt() const` que retorna `true` si el valor del objeto cabe en una variable `long int`.
- `bool esLongLongInt() const` que retorna `true` si el valor del objeto cabe en una variable `long long int`.

Sobrecargue el operador + de manera de poder realizar sumas entre objetos `BigInteger`, y de sumas entre objetos `BigInteger` y tipos de datos enteros primitivos. Para ello, puede usar código semejante a sus constructores anteriores. En otras palabras, para sumar, por ejemplo, un objeto `BigInteger` y un entero de tipo `long int`, puede convertir el entero `long int` en un `BigInteger` y luego realizar una suma entre objetos `BigInteger`.

Sobrecargue también el operador − de manera semejante.

Sobrecargue también el operador ==, que retorne `true` si dos objetos `BigInteger` son iguales.

Adicionalmente, implemente toda otra función miembro que Ud. necesite para resolver las siguientes preguntas.

Objetos `BigInteger` y los números de Fibonacci

En esta sección, Ud. debe utilizar su clase `BigInteger` para calcular números de Fibonacci de gran cantidad de dígitos. En todos los casos siguientes, incluya en su informe los valores de n y $\mathcal{F}_n$ que cumplen la condición solicitada.

1. Calcule el menor número de Fibonacci $\mathcal{F}_n$ en tener más de 1000 dígitos.
2. Calcule el primer número de Fibonacci $\mathcal{F}_n$ para el cual los 9 últimos dígitos conforman un número pandigital 1-a-9.
3. Calcule el primer número de Fibonacci $\mathcal{F}_n$ para el cual los 9 primeros dígitos conforman un número pandigital 1-a-9.
4. Calcule el primer número de Fibonacci $\mathcal{F}_n$ para el cual los 9 últimos dígitos y los 9 primeros dígitos conforman números pandigitales 1-a-9 (no necesariamente el mismo!).