Escuela Superior de Ingeniería y Tecnología
Universidad de La Laguna

Grado en Ingeniería Informática
Algoritmos y Estructuras de Datos Avanzadas

Curso 2022-2023

Práctica 2: Calculadora de números grandes en binario

---

1. Objetivo

En esta práctica se trabajarán los siguientes conceptos del lenguaje C++: la especialización de plantillas como mecanismo para expresar situaciones particulares y la sobrecarga del operador cambio de tipo para clases definidas por el usuario.

2. Entrega

Se realizará una sesión de entrega presencial en el laboratorio entre el 27 de febrero y el 3 de marzo de 2023.

3. Enunciado

A partir de la implementación de la plantilla `BigInt<size_t Base>`, elaborada en la práctica 1 para representar números grandes en notación posicional [1], se quiere generar una nueva versión de la plantilla donde se especializa el comportamiento para números grandes en formato binario (`Base=2`).

La representación de números enteros en el sistema binario Complemento a dos [2] permite optimizar el almacenamiento y la implementación de las operaciones. En este formato no se requiere almacenar el signo del número, pero cambia la forma de calcular la aportación de cada dígito al valor total del número respecto a la notación posicional. En complemento a dos, si `N` es un número entero almacenado en `n` dígitos, se verifica:

- El rango de valores que puede tomar está definido en $-2^{n-1} <= N <= 2^{n-1}-1$
- Si $N >= 0$ entonces su bits más significativo debe ser `0`, y el valor que aporta cada dígito depende de su posición 2^i con $0 <= i < n-1$;.
- Si $N < 0$ entonces su bit más significativo debe ser `1`, y el valor del número se corresponde con el valor usando la representación posicional de su complemento a dos, $C_2(N) = 2^n - N$.

La representación en complemento a dos permite calcular la resta de números enteros binarios sumando al minuendo el complemento a dos del sustraendo. Esto es, N - M = N + C_2(M).

Por otro lado, en los números en formato binario es posible almacenar los dígitos del número (bits) en un vector de `bool`, en lugar de un vector de `char` como se ha hecho en la plantilla genérica para cualquier valor de la `Base`. La librería estándar del lenguaje C++ implementa una especialización de su estructura `std::vector<bool>` [3] para optimizar el espacio de almacenamiento.

Se pide desarrollar un programa que implemente una calculadora de expresiones en notación polaca inversa [4] con operandos de tipo `BigInt<2>`. Para manejar los operandos el programa define una tabla `Board` de parejas: etiqueta alfanumérica y valor `BigInt<2>`, que se leerán desde un fichero de entrada que tiene el siguiente formato:

- La primera fila contiene la `Base` utilizada para expresar los números `BigInt<Base>`.

- Las siguientes líneas comienzan con una etiqueta seguida de:
  - Un símbolo `'='` y un número `BigInt<Base>`, o
  - Un símbolo `'?'` y una expresión en notación polaca inversa que utiliza las etiquetas previas como operandos y cualquiera de los operadores aritméticos definidos para el tipo `BigInt<2>`.

Por ejemplo:

Base = 10
N1 = 442142117615672
N2 = 46651367647546
E1 ? N1 N2 +
E2 ? E1 N1 N2 - +

En cada paso el programa lee una línea del fichero. Si la línea contiene un operando, pareja etiqueta y número, lo almacena en la tabla `Board`. Si la línea contiene una expresión en notación polaca inversa, evalúa la expresión sustituyendo cada etiqueta por el número almacenado en la tabla `Board` tras convertirlo a la base binaria, y guarda en la tabla la nueva pareja, etiqueta y número resultado en la base dada. Cuando termina de procesar todas las líneas del fichero de entrada, el programa genera un fichero de salida con los valores contenidos en la tabla `Board`.

Para el fichero de ejemplo anterior:

Base = 10
N1 = 442142117615672
N2 = 46651367647546
E1 = 488793485263218
E2 = 884284235231344

4. Notas de implementación

- Las particularidades en la implementación de la plantilla de números para la base binaria se expresa mediante una especialización en la plantilla `BigInt<Base>`, que se define con la siguiente sintaxis:

```
template<>
class BigInt<2> { //Código de plantilla para Base=2, binario en complemento a dos };
```

- La especialización de una plantilla forma parte de la misma plantilla donde se define el comportamiento genérico. Por tanto, el código de la especialización también se coloca en el mismo fichero cabecera de la plantilla genérica.

- La conversión de un número almacenado en un objeto de tipo `BigInt<Base>` al mismo número almacenado en un objeto de tipo `BigInt<2>`, o la conversión de un número en formato binario al mismo número en cualquier otra Base, requiere la implementación en la plantilla de clases de los correspondientes operadores de cambio de tipo.

```
template<size_t Base>
class BigInt {
  public:
    ...
    operator BigInt<2>() { // código de conversión de Base a binario }
    ...
```

```
};

template<>
class BigInt<2> {
  public:
    ...
    operator BigInt<8>()  { // código de conversión de binario a octal }
    operator BigInt<10>() { // código de conversión de binario a decimal }
    operator BigInt<16>() { // código de conversión de binario a hexadecimal }
    ...
```

5. Referencias

[1] Notación posicional [Wikipedia]: https://es.wikipedia.org/wiki/Notaci%C3%B3n_posicional

[2] Complemento a dos [Wikipedia]: https://es.wikipedia.org/wiki/Complemento_a_dos

[3] `std::vector<bool>` [cpluplus.com]: https://cplusplus.com/reference/vector/vector-bool/

[4] Notación polaca inversa [Wikipedia]:

https://es.wikipedia.org/wiki/Notaci%C3%B3n_polaca_inversa#El_algoritmo_RPN