75.04/95.12 Algoritmos y Programación II
Trabajo práctico 0: programación C++

Universidad de Buenos Aires - FIUBA
Primer cuatrimestre de 2021

1. Objetivos

Ejercitar los conceptos básicos de programación C++ vistos en las primeras clases. Familiarizarse con algunas de las herramientas de software que usamos en el curso y en los siguientes trabajos prácticos, implementando un programa (y su correspondiente documentación) que resuelva el problema que presentaremos más abajo.

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá será entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes, un informe impreso de acuerdo con lo que mencionaremos en la sección 5, y con una copia digital de los archivos fuente necesarios para compilar el trabajo.

4. Descripción

El objetivo fundamental del trabajo es implementar números enteros de precisión fija (pero arbitrariamente grande). Entederemos como *precisión* a la cantidad de dígitos decimales que conforman el número.

En una primera etapa, deseamos representar este concepto utilizando arreglos. Para ello, desarrollaremos una clase C++ `bignum` siguiendo el esquema general que se muestra a continuación:

```
class bignum
{
    private:
        unsigned short *digits;
        // ...
    public:
        // ...
        friend bignum operator+(const bignum&, const bignum&);
        friend bignum operator-(const bignum&, const bignum&);
        friend bignum operator*(const bignum&, const bignum&);
        friend std::ostream& operator<<(std::ostream&, const bignum&);
        friend std::istream& operator>>(std::istream&, bignum&);
};
```

Como se observa en este fragmento, será preciso además sobrecargar los operadores aritméticos de suma, resta y multiplicación y los de entrada y salida con formato. Esto se debe a que, a los fines de introducirse en el lenguaje y familiarizarse con sus construcciones, realizaremos un ejercicio sencillo que consiste en tomar expresiones artiméticas de dos operandos desde *streams* configurables y resolverlas, informando el resultado en otro stream de salida, también configurable.

Las porciones marcadas con puntos suspensivos corresponden a los restantes métodos y/o variables necesarias para el correcto funcionamiento del programa. Su elección queda a criterio de cada grupo.

Especificaremos tanto los nombres de los streams como así también la precisión con la que trabajaremos a través de opciones de línea de comando (ver sección 4.2).

4.1. Consideraciones algorítmicas

Si bien existen diversos algoritmos para resolver el problema de la multiplicación de números de precisión arbitraria, en este primer acercamiento la idea será implementar el algoritmo trivial de multiplicación enseñado en la escuela primaria [1]. En instancias de evaluación posteriores retomaremos esta problemática e implementaremos algoritmos de mayor sofisticación, dejando también espacio para un análisis riguroso que nos permita comparar ambas alternativas.

4.2. Línea de comando

Las opciones `-i` y `-o` permiten seleccionar los *streams* de entrada y salida respectivamente. Por defecto, éstos serán `cin` y `cout`. Lo mismo ocurrirá al recibir "`-`" como argumento.

Por otro lado, la opción `-p` indicará el valor de la precisión con la que llevaremos a cabo el procesamiento. Puede asumirse que los números ingresados en el stream de entrada se ajustan a dicha precisión, aunque podría ocurrir que el resultado de alguna de las expresiones de entrada requiera mayor cantidad de dígitos. Cada grupo deberá definir y documentar en forma adecuada qué hacer en caso de recibir un número no representable con la precisión definida.

Al finalizar, todos nuestros programas retornarán un valor nulo en caso de no detectar

ningún problema; en caso contrario, devolveremos un valor no nulo (por ejemplo `1`).

4.3. Formato de los archivos de entrada y salida

El formato a adoptar para el *stream* de entrada consiste en una secuencia de cero o más expresiones aritméticas binarias, cada una ocupando una línea distinta. A su vez, la separación entre cada operando y el operador puede darse con cero o más espacios, entendiendo por tales a los caracteres `SP`, `\f`, `\r`, `\t`, `\v`.

Por otro lado, el *stream* de salida deberá listar en líneas distintas los números resultantes de evaluar cada expresión, manteniendo el mismo orden de las operaciones de la entrada.

A continuación, presentaremos algunos ejemplos puntuales de estos archivos.

4.4. Ejemplos

Primero, usamos un archivo vacío como entrada del programa:

```
$ ./tp0 -i /dev/null
```

Notar que la salida es también vacía.

En el siguiente ejemplo, pasamos una única operación como entrada. La salida deberá contener sólo una línea con el resultado adecuado:

```
$ echo 1123581321345589*123456789 | ./tp0 -p 20
13871374211370357725372l
```

No olvidemos que los números son *enteros*: también pueden ser negativos. El siguiente ejemplo muestra un correcto comportamiento del programa frente a estos casos:

```
$ echo "-1  - 5" | ./tp0 -p 1
-6
```

Notar además que la precisión puede ser incluso mayor a la cantidad de dígitos del número de mayor valor absoluto contenido en la entrada.

4.5. Portabilidad

A los efectos de este primer trabajo, es deseable (pero no requisito) que la implementación desarrollada provea un grado mínimo de portabilidad. Sugerimos verificar nuestros programas en Windows/MS-DOS y/o alguna versión reciente de UNIX: BSD, o Linux[1].

[1] RedHat, SuSe, Debian, Ubuntu.

5. Informe

El informe deberá incluir:

- Una carátula que incluya los nombres de los integrantes y el listado de todas las entregas realizadas hasta ese momento, con sus respectivas fechas.
- Documentación relevante al diseño e implementación del programa.
- Documentación relevante al proceso de compilación: cómo obtener el ejecutable a partir de los archivos fuente.
- Las corridas de prueba, con los comentarios pertinentes.
- El código fuente, en lenguaje C++ (en dos formatos, digital e impreso).
- Este enunciado.

6. Fechas

La última fecha de entrega y presentación será el **jueves 27 de mayo**.

Referencias

[1] Wikipedia, "Long multiplication." `https://en.wikipedia.org/wiki/Multiplication_algorithm#Long_multiplication`.