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

Universidad de Buenos Aires - FIUBA
Segundo cuatrimestre de 2019

1. Objetivos

Ejercitar conceptos básicos de programación C++, implementando un programa y su correspondiente documentación que resuelva el problema descripto 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 6, y con una copia digital de los archivos fuente necesarios para compilar el trabajo.

4. Descripción

En esta sección describiremos los algoritmos de transformación que deben implementarse en este TP.

4.1. DFT: discrete fourier transform (ida)

Este algoritmo permite calcular la transformación de una secuencia de N puntos $x(0), \ldots, x(N-1)$ dada por la fórmula:

$$y(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}; k = 0, \ldots, N-1 \quad (1)$$
$$; W_N = e^{-j\frac{2\pi}{N}}$$

Es decir, ante una entrada de N valores complejos, producirá una secuencia de salida de igual tipo y longitud.

4.2. iDFT: inverse discrete fourier transform

En este caso queremos hacer la operación inversa: obtener el vector original, x, a partir del vector transformado y,

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} y(k) W_N^{-nk}; n = 0, \ldots, N-1 \quad (2)$$

Podemos ver entonces, que se trata de un cálculo muy similar al descripto en la sección anterior: las únicas diferencias están en un factor $1/N$, y en el signo negativo en el exponente de W_N.

5. Implementación

Para facilitar el proceso de desarrollo y debug, el programa deberá implementar tanto el algoritmo de transformación directa (DFT), como su inversa (iDFT).

Para configurar el funcionamiento del programa, usaremos parámetros pasados por la línea de comando: por defecto, el programa se limitará a leer con formato la secuencia de números complejos por el stream de texto de entrada `std::cin`, e imprimirá (también en forma de texto con formato) el resultado de la DFT por `std::cout`.

El formato de los streams de entrada y salida consiste en una secuencia de números complejos en la cual los elementos están expresados en forma de par ordenado (i.e., (a, b), siendo a la parte real, y b la parte imaginaria), o como números reales sueltos.

5.1. Interfaz

Las opciones de línea de comando a implementar en este TP son:

- `-o`, o `--output`, permite colocar la secuencia de números complejos de salida en el archivo pasado como argumento; o por salida estándar -`std::cout`- si el argumento es "`-`".
- `-i`, o `--input`, para controlar el stream de entrada, de forma similar a la opción anterior; es decir, el programa leerá de `std::cin` cuando el valor pasado sea "`-`"..
- `-m`, o `--method` permite especificar el algoritmo de cómputo: `dft` (sección 4.1), `idft` (4.2). En este TP, el valor por defecto a utilizar es `dft`.

En todos los casos, el formato de los streams de entrada/salida es el de números complejos con formato, expresados como pares ordenados de la forma `($re, $im)`, siendo `$re` la parte real, y `$im` la parte imaginaria. Asimismo, el programa deberá aceptar números reales sueltos en su entrada.

5.2. Ejemplos

Comencemos invocando al programa con las opciones por defecto, es decir, transformando con DFT los datos de `std::cin` y enviando la salida a `std::cout`:

```
$ cat entrada.txt
1 1 1 1
$ tp1 < entrada.txt
(4, 0) (0, 0) (0, 0) (0, 0)
$ tp1 -m dft < entrada.txt
(4, 0) (0, 0) (0, 0) (0, 0)
```

Observamos que la entrada también puede ser compleja:

```
$ cat entrada2.txt
(1, 0) (0, 0) 0 (0, 0) 0 0 (0, 0) (0, 0)
$ tp1 < entrada2.txt
(1, 0) (1, 0) (1, 0) (1, 0) (1, 0) (1, 0) (1, 0) (1, 0)
```

Además, podemos invocar al programa para que realice los cálculos con DFT, y así validar los resultados anteriores:

```
$ tp1 -m dft < entrada2.txt
(1, 0) (1, 0) (1, 0) (1, 0) (1, 0) (1, 0) (1, 0) (1, 0)
```

Similarmente, para anti-transformar:

```
$ cat entrada4.txt
(0, 0) (0, 0) (4, 0) (0, 0)
$ tp1 -m idft < entrada4.txt
(1, 0) (-1, 0) (1, 0) (-1, 0)
$ tp1 -m idft -o salida4.txt < entrada4.txt
$ cat salida4.txt
(1, 0) (-1, 0) (1, 0) (-1, 0)
```

5.3. Portabilidad

Es deseable que la implementación desarrollada provea un grado mínimo de portabilidad. Sugerimos verificar nuestros programas en alguna versión reciente de UNIX: BSD o Linux.

6. Informe

El informe deberá incluir:

- Documentación relevante al diseño e implementación del programa.
- Documentación relevante a los algoritmos involucrados en la solución del trabajo.
- 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.

7. Fechas

La última fecha de entrega es el jueves 3 de octubre.

Referencias

[1] Alan V. Oppenheim, Roland W. Schafer. Discrete-time signal processing, second edition. Sección 9.3.1: In-Place Computations. Ed. Prentice Hall.