Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Estructuras de Datos
Taller 2: estructuras lineales, **2023-10**

1 Objetivo

Implementar un sistema que use estructuras lineales para el manejo de información estructurada. En especial, se desea evaluar las habilidades del estudiante en el desarrollo y uso de las operaciones de inserción y búsqueda de información en estas estructuras.

2 Recordatorio: compilación con g++

La compilación con g++ (compilador estándar que será usado en este curso para evaluar y calificar las entregas) se realiza con los siguientes pasos:

1. **Compilación**: de todo el código fuente compilable (**ÚNICAMENTE LOS ARCHIVOS CON EXTENSIONES** `*.c`, `*.cpp`, `*.cxx`)
   `g++ -std=c++11 -c *.c *.cxx *.cpp`
2. **Encadenamiento**: de todo el código de bajo nivel en el archivo ejecutable
   `g++ -std=c++11 -o nombre_de_mi_programa *.o`

   Nota: Estos dos pasos (compilación y encadenamiento) pueden abreviarse en un sólo comando:
   `g++ -std=c++11 -o nombre_de_mi_programa *.c *.cxx *.cpp`
3. **Ejecución**: del programa ejecutable anteriormente generado
   `./nombre_de_mi_programa`

ATENCIÓN: Los archivos de encabezados (*.h, *.hpp, *.hxx) **NO SE COMPILAN**, se incluyen en otros archivos (encabezados o código). Así mismo, los archivos de código fuente (*.c, *.cpp, *.cxx) **NO SE INCLUYEN**, se compilan. Si el programa entregado como respuesta a este Taller no atiende estas recomendaciones, automáticamente se calificará la entrega sobre un 25% menos de la calificación máxima.

2.1 Recordatorio: lectura de archivos de texto

La librería `fstream` en C++ permite el uso de operaciones de lectura y escritura de archivos de texto, a través de un flujo que se conecta al archivo. A continuación se presenta un ejemplo de lectura de un archivo por líneas (tomado de `http://www.cplusplus.com/doc/tutorial/files/`):

```
// reading a text file
#include <iostream>
#include <fstream>
#include <string>
int main () {
  std::string line;
  std::ifstream myfile ("example.txt");
  if (myfile.is_open()) {
    while ( getline (myfile,line) ) {
      std::cout << line << '\n';
    }
    myfile.close();
  }
  else std::cout << "Unable to open file";
  return 0;
}
```

3 Descripción del problema

JaveMusic es un programa que permite al usuario manejar y acceder a la música que tiene almacenada, además que le permite adicionar, eliminar y editar las canciones que éste tiene guardadas. Adicionalmente, le permite al usuario realizar búsquedas de sus canciones de múltiples formas, bien sea por su título, artista, género o por el álbum al que está asociada. Finalmente, el sistema permite registrar información adicional para las canciones, como su duración y la calificación que el usuario le asigna.

4 Desarrollo del taller

El objetivo del taller es implementar las funcionalidades de lectura y búsqueda de información de JaveMusic, detalladas a continuación.

- Lectura de información: JaveMusic lee la información sobre las canciones desde un archivo de texto, el cual tiene la siguiente estructura:
  - **Línea 1**: Número (N) de canciones.
  - **Líneas 2 a la N+1**: Nombre de la canción, autor (sólo el primero), género (sólo un género), nombre del álbum, año. Todos estos datos están separados por el caracter *pipe* '|'.
- Búsqueda de información: las siguientes operaciones de búsqueda deben estar disponibles para el usuario:
  - Listar todos los autores presentes, ordenados alfabéticamente.
  - Dado un autor, listar todas sus canciones ordenadas alfabéticamente.
  - Listar todos los álbumes presentes, ordenados cronológicamente.
  - Dado un álbum, listar todas las canciones que componen un álbum, ordenadas alfabéticamente por el nombre de la canción.
  - Listar todas las canciones y su álbum, ordenadas alfabéticamente primero por el álbum y después por el nombre de la canción.

El sistema debe permitir al usuario realizar las operaciones de lectura y las diferentes búsquedas a través de un menú de opciones. Para ejecutar las búsquedas rápidamente, se sugiere que el programa utilice estructuras lineales ordenadas para almacenar los datos al momento de leerlos desde el archivo, de acuerdo a la información requerida por las diferentes búsquedas. Por ejemplo, para listar todas las canciones ordenadas por autor, puede crearse una lista de autores, donde cada elemento de la lista tenga una referencia a las canciones asociadas a dicho autor. Sin embargo, tampoco se espera llegar al extremo de duplicar o triplicar la información para facilitar las búsquedas.

5 Evaluación

La entrega se hará a través de la correspondiente actividad de UVirtual, antes de la medianoche del próximo jueves 25 de febrero. Se debe entregar un único archivo comprimido (únicos formatos aceptados: .zip, .tar, .tar.gz, .tar.bz2, .tgz), nombrado con los apellidos de los integrantes del grupo. Este comprimido debe contener, dentro de un mismo directorio (sin estructura de carpetas interna), el documento de diseño (.pdf) y el código fuente (.h, .hxx, .hpp, .c, .cxx, .cpp). Si la entrega contiene archivos en cualquier otro formato, será descartada y no será evaluada, es decir, la nota definitiva de la entrega será de 0 (cero) sobre 5 (cinco).

La evaluación del taller tendrá la siguiente escala, para cada uno de los comandos (opciones) pedidos:

- **Excelente** (**5.0/5.0**): El estudiante diseñó e implementó una solución completa del comando.
- **No fue un trabajo formal de ingeniería** (**3.5/5.0**): El estudiante implementó una solución completa del comando, pero no la diseñó.
- **Necesita mejoras sustanciales** (**2.5/5.0**): El estudiante diseñó y/o implementó una solución, pero no es completa (no soluciona lo pedido).
- **Malo** (**1.0/5.0**): El código entregado por el estudiante no compila en el compilador g++ (mínimo versión número 4.5).
- **No entregó** (**0.0/5.0**).

IMPORTANTE: Por "diseño de la solución" se entiende la especificación de los TAD y el diagrama de relación entre TADs en el formato visto en clase.