Tarea 2

Manejo de datos y punteros

Entrega: miércoles 30 de abril a las 21:00 hrs.

1. Introducción

En esta tarea, se le pide que escriba un programa en `C` que gestione una base de datos de canciones y permita al usuario generar listas personalizadas aplicando distintos filtros. El propósito es ofrecer una herramienta rápida y eficiente para crear listas de canciones basadas en criterios como duración, género o año de lanzamiento, y guardar los resultados en archivos de texto.

2. Objetivo

Crear un programa en `C` que:

- Lea un archivo `.csv` con información de canciones.
- Almacene los datos en memoria dinámica usando `struct` y punteros (sin duplicar registros).
- Permita al usuario aplicar filtros interactivos (menú por consola).
- Genere listas de canciones y las escriba en archivos `.txt` con un formato concreto.

Los detalles de estos requisitos se entregan a continuación.

3. Funcionamiento

Su programa deberá maximizar la velocidad de ejecución y la eficiencia en el uso de memoria, según las pautas que se dictan en este enunciado.

i) **Lectura del archivo (0,5 pts.)** El programa recibirá como argumento el nombre del `.csv` (p.ej. `songs_bd.csv`), que puede estar compuesto de decenas de miles de filas con los siguientes atributos:

```
song_id;song_title;artist;album;genre;release_date;duration;popularity;stream;
language;explicit_content;label;composer;producer;collaboration
```

ii) **Almacenar cada canción** en un arreglo de `struct` en memoria dinámica **(1,5 pts.)** Para maximizar la velocidad, los datos deben permanecer en memoria. Para ahorrar memoria, evite copiar datos innecesariamente.

iii) **Filtros (2,5 pts.)** El usuario podrá combinar estos filtros en cualquier orden:

- Para los filtros de Artista, Álbum, Género o Idioma, el sistema listará todas las opciones disponibles sin repetir. Si hay más de 100 valores distintos, se mostrarán de forma arbitraria únicamente 100 de ellos para facilitar la selección.
- Para filtrar por año de lanzamiento, el programa pedirá dos valores numéricos consecutivos:
  ej:

  ```
  Año inicio:
  Año fin:
  ```

  Sólo las canciones cuyo año esté dentro de ese rango deben ser seleccionadas.
- Para filtrar por duración (segundos), el programa pedirá dos valores numéricos consecutivos:
  ej:

  ```
  Duración mínima (s):
  Duración máxima (s):
  ```

  Sólo las canciones cuyo año esté dentro de ese rango deben ser seleccionadas.

iv) **Generación de listas (1,5 pts.)**

Después de cada filtro aplicado, se muestra en consola el número de canciones seleccionadas en la lista, si este es mayor a 0, se le pregunta al usuario si desea:

- Aplicar otro filtro sobre los resultados ya seleccionados.
- Visualizar las primeras 100 canciones en la lista (en consola).
- Exportar la lista a un archivo de texto.
- Salir para terminar la ejecución.

El archivo .txt debe tener por línea el siguiente formato para tener puntaje completo: `SONG_ID: ARTIST - SONG_TITLE (RELEASE_YEAR) -- GENRE`.

Es importante que los datos resultantes en la lista generada sean correctos y consistentes con los filtros aplicados.

4. Recomendaciones

- Recuerde que su tarea será evaluada por funcionalidades. Esto tiene dos implicancias prácticas importantes: 1) más vale una tarea en que funcionan la mitad de las cosas que una que está a punto de funcionar pero no compila, y 2) antes de desesperarse o bloquearse, divida la tarea en funcionalidades e impleméntelas una a una de manera de ir asegurando puntaje.

- Comience el desarrollo con anticipación. Es muy probable que en el camino encuentre problemas que requieran tiempo para pensar o discutir con el profesor o los ayudantes.

5. Consideraciones Generales sobre la Evaluación

En cada ítem se reservará una parte del puntaje para la **eficiencia de la solución**, en términos de cantidad total de memoria utilizada y velocidad de respuesta.

Recuerde que está prohibido usar materia no vista en clases, salvo que se autorice explícitamente lo contrario. En este contexto, está *permitido* usar punteros (incluidos punteros a funciones), `malloc` y sus funciones relacionadas, `struct`, y `typedef`. Por su importancia para la tarea, se subirá a la página del curso en Canvas un video explicativo sobre `struct`.

Su programa debe ser robusto frente a datos que no corresponden. Esto quiere decir que su programa no debe caerse en caso de que el usuario haga operaciones matemáticas no válidas (p.ej., dividir por cero si fuera el caso). Sin embargo, se puede asumir que el usuario será amigable, en el sentido de que cuando le soliciten un número este no ingresará letras, por ejemplo.

6. Desarrollo, compilación y entrega

Esta tarea puede ser resuleta en grupos de máximo 2 personas. El plazo para la entrega de la tarea vence impostergablemente el miércoles 30 de abril a las 21:00 hrs.. Cuide su tarea para que no sea copiada parcial o íntegramente por otros. Todas las tareas entregadas serán comparadas por un sistema automático de detección de plagio. Cualquier sospecha de copia (de otras tareas o de internet) podrá ser denunciada, investigada y eventualmente penalizada de conformidad al Reglamento del Alumno.

Formato de entrega: Subir un único archivo con el código de su programa al módulo de tareas de la página del curso, con un nombre de archivo que incluya ambos apellidos de los autores de la tarea separados por guiones, seguido de "Tarea2.c", de la forma "`Prat Chacon-Carrera Pinto-Tarea2.c`". El incumplimiento de este requisito de nombre de archivo será penalizado con un descuento de un punto en la nota. Los archivos compilados no serán tomados en cuenta, si se llega a subir sólo un archivo compilado, este será ignorado, y será evaluado con nota 1.

En el momento de compilación, se deberá indicar las siguientes *flags* al compilador:

```
-Wall -Wextra -Wundef -Werror -Wuninitialized -Winit-self
```

Aquellas tareas que no compilen de la forma básica (`gcc -o salida entrada`) serán evaluadas con nota 1. Las que compilen, pero se caigan durante la ejecución, serán evaluadas con nota máxima 3.

Su programa podría ser evaluado con múltiples casos de prueba y deberá ser capaz de ejecutarlos todos de manera correcta. De fallar en algún caso de prueba serán descontados los puntos correspondientes a dicho caso.