UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA

Programación II – Curso 23/24

Práctica 2: Enunciado

1. El problema

La segunda práctica consistirá en añadir un conjunto de nuevas funcionalidades al programa MUSFIC desarrollado en la primera práctica. En concreto, se incluirá una lista de reproducción de canciones a cada usuario.

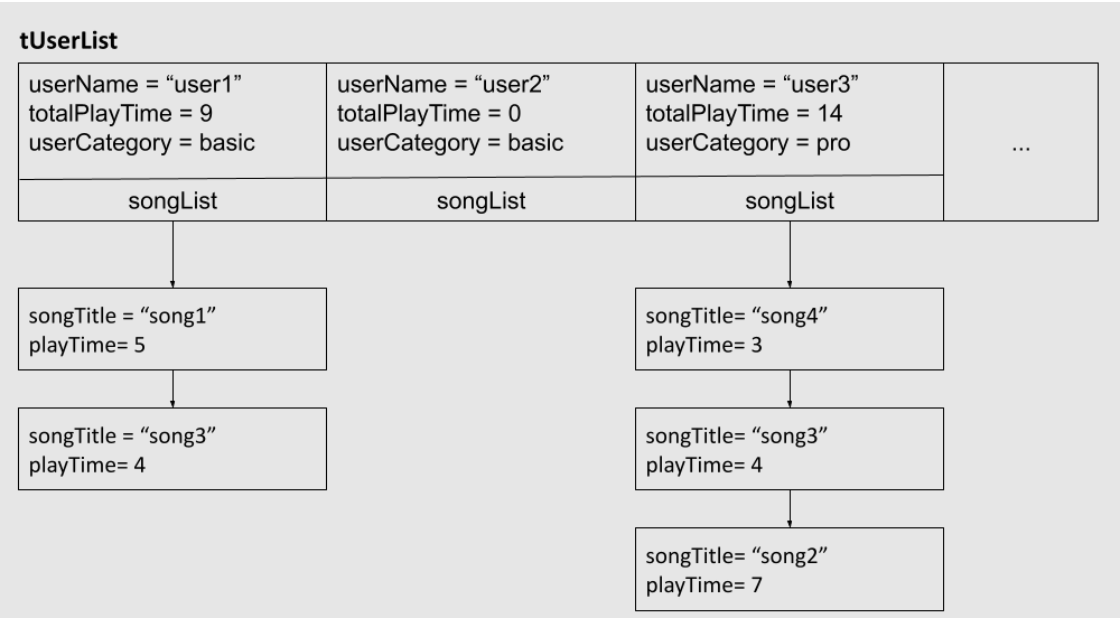
El objetivo de esta práctica es comprender el funcionamiento e implementar varios tipos abstractos de datos (TADs) así como manejar interdependencias entre ellos.

2. Fase 1

Para resolver este problema se utilizarán 2 tipos abstractos de datos (TADs):

- Una Lista Ordenada (TAD UserList) para almacenar la lista de usuarios.
- Una Lista NO Ordenada (TAD SongList) para almacenar las canciones de cada usuario.

Cada usuario de MUSFIC, esto es, cada nodo de la lista ordenada tendrá su propia lista de reproducción, tal y como representa la siguiente figura:



2.1 Librería Types

Algunos tipos de datos comunes se definirán en el fichero `types.h`, ya que se utilizarán en varios TADs.

`NAME_LENGTH_LIMIT`	Longitud máxima de un nombre de usuario y de un título de canción (constante).
`tUserName`	Nombre de un usuario (`string`).
`tUserCategory`	Categoría de usuario (tipo enumerado: `{basic, pro}`)
`tPlayTime`	Tiempo de reproducción, en minutos (`int`).
`tSongTitle`	Título de una canción (`string`).
`tSong`	Datos de una canción. Compuesto por: • `songTitle` de tipo `tSongTitle` • `playTime` de tipo `tPlayTime`

2.2 TAD UserList

Para mantener la lista de usuarios y su información asociada el sistema utilizará un TAD Lista Ordenada con implementación **dinámica, simplemente enlazada** (`user_list.c` y `user_list.h`).

2.2.1. Tipos de datos incluidos en el TAD UserList

`tListU`	Representa una lista de usuarios **ordenada por sus nombres**.
`tItemU`	Datos de un elemento de la lista (un usuario). Compuesto por: • `userName`: de tipo `tUserName` • `totalPlayTime`: de tipo `tPlayTime` • `userCategory`: de tipo `tUserCategory` • `songList`: de tipo `tListS`
`tPosU`	Posición de un elemento de la lista de usuarios.
`NULLU`	Constante usada para indicar posiciones nulas de la lista de usuarios.

2.2.2. Operaciones incluidas en el TAD UserList

Las operaciones son las mismas que las del TAD Lista de la Práctica 1 renombrando los tipos de datos y las operaciones para evitar conflictos con el TAD SongList. Para ello, todos los identificadores del TAD UserList pasarán a terminar en U. Cambia también la especificación de las siguientes operaciones:

- `insertItemU (tItemU, tListU) → tListU, bool`
  Inserta en la lista de forma **ordenada**, en función del campo `userName`, un elemento con los datos indicados. Devuelve un valor `true` si el elemento fue insertado; `false` en caso contrario.
  PostCD: Las posiciones de los elementos de la lista posteriores al insertado pueden cambiar de valor.

- `deleteAtPositionU (tPosU, tListU) → tListU`
  Elimina de la lista el elemento que ocupa la posición indicada.
  PreCD: La posición indicada es una posición válida en la lista **y el usuario en dicha posición tiene una lista de canciones vacía.**
  PostCD: Las posiciones de los elementos de la lista posteriores a la de la posición eliminada pueden haber variado.

Asimismo, la especificación de la operación `findItemU` no cambia (salvo el nombre de los tipos) pero su implementación sí debe tener en cuenta que ahora la lista está **ordenada**:

- `findItemU (tUserName, tListU) → tPosU`

2.3 TAD SongList

Este TAD implementará una lista **estática** basándose en el TAD Lista (`song_list.c` y `song_list.h`) con un tamaño máximo de **25 canciones**.

2.3.1. Tipos de datos incluidos en el TAD SongList

`tListS`	Representa una lista de canciones (**no ordenada**).
`tItemS`	Datos de un elemento de la lista de canciones; es decir, una canción (`tSong`).
`tPosS`	Posición de un elemento de la lista de canciones.
`NULLS`	Constante usada para indicar posiciones nulas.

2.3.2. Operaciones incluidas en el TAD SongList

Las operaciones del TAD Lista son idénticas a las indicadas en la Práctica 1 (consulte el enunciado de dicha práctica) **cambiando los nombres de los tipos adecuadamente y añadiendo el sufijo S al nombre de las operaciones.**

- `createEmptyListS (tListS)→ tListS`
- `isEmptyListS (tListS) → bool`
- `firstS (tListS) → tPosS`
- `lastS (tListS) → tPosS`
- `nextS (tPosS, tListS)→ tPosS`

- `previousS (tPosS, tListS) → tPosS`
- `insertItemS (tItemS, tPosS, tListS) → tListS, bool`
- `deleteAtPositionS (tPosS, tListS) → tListS`
- `getItemS (tPosS, tListS) → tItemS`
- `updateItemS (tItemS, tPosS, tListS) → tListS`
- `findItemS (tSongTitle, tListS) → tPosS`

3. Fase 2

Una vez implementados los TADs, nos centraremos en el programa principal. La tarea consiste en implementar un único programa principal (`main.c`) que **haga uso de la UserList y de la SongList**, y que procese las peticiones de los usuarios de MUSFIC con el siguiente formato:

`N userName userCategory`	***[N]ew***: Alta de un usuario de categoría `basic` o `pro`.
`D userName`	***[D]elete***: Baja de un usuario.
`A userName songTitle`	***[A]dd***: El usuario añade una canción a su lista de reproducción.
`U userName`	***[U]pgrade:*** Sube la categoría de un usuario de `basic` a `pro`.
`P userName songTitle playTime`	***[P]lay:*** Reproducción de `playTime` (minutos) de una canción por parte de un usuario.
`S`	***[S]tats:*** Listar los usuarios actuales de MUSFIC y sus datos.
`R maxTime`	***[R]emove:*** Elimina todos los usuarios `basic` cuyo contador de tiempo de reproducción exceda `maxTime` minutos.

En el programa principal se implementará un bucle que procese una a una las peticiones de los usuarios. Para simplificar tanto el desarrollo como las pruebas, el programa no necesitará introducir ningún dato por teclado, sino que leerá y procesará las peticiones de usuarios contenidas en un fichero (**ver documento** `EjecucionScript_P2.pdf`) En cada iteración del bucle, el programa leerá del fichero una nueva petición y la procesará. Para facilitar la corrección de la práctica todas las peticiones del fichero van numeradas correlativamente.

Para cada línea del fichero de entrada, el programa hará lo siguiente:

1. **Muestra una cabecera con la operación a realizar.** Esta cabecera está formada por una primera línea con 20 asteriscos y una segunda línea que indica la operación tal y como se muestra a continuación:

```
********************
CC␣T:␣user/maxtime␣XX␣category/song␣YY␣minutes␣ZZ
```

donde `CC` es el número de petición, `T` es el tipo de operación (`N`, `D`, `A`, `U`, `P`, `S`, `R`), `XX` es el nombre del usuario (`userName`) o el tiempo máximo de reproducción (`maxTime`) (cuando y según corresponda), `YY` es la categoría del usuario (`userCategory`) o el título de la canción (`songTitle`) (cuando y según corresponda), `ZZ` es el tiempo de reproducción para la operación `[P]lay` (`playTime`), y ␣ indica un espacio en blanco. Sólo se imprimirán los parámetros necesarios; es decir, para una petición `[S]tats` se mostraría únicamente "`01 S`", mientras que para `[P]lay` se mostraría "`02 P: user User1 song Song1 minutes 4`".

2. Procesa la petición correspondiente:

- Si la operación es **`[N]ew`**, se incorporará el usuario a la lista de usuarios con la categoría indicada y su contador de tiempo de reproducción (`totalPlayTime`) inicializado a 0. Además, se imprimirá el mensaje:

  `*␣New:␣user␣XX␣category␣YY`

  donde, de nuevo, `XX` es el `userName`, `YY` es la `userCategory`, y ␣ representa un espacio en blanco. El resto de los mensajes siguen el mismo formato.

  Si ya existiese un usuario con ese nombre o no se hubiese podido realizar la inserción, se imprimirá el siguiente mensaje:

  `+␣Error:␣New␣not␣possible`

  Hay que tener en cuenta que cada usuario de la lista tiene una lista de reproducción de canciones que debe gestionarse adecuadamente a la hora de insertarlo.

- Si la operación es **`[D]elete`**, se buscará al usuario en la lista, se borrará y se imprimirá el siguiente mensaje:

  `*␣Delete:␣user␣XX␣category␣YY␣totalplaytime␣ZZ`

  Si no existiese ningún usuario con ese nombre o la lista de usuarios estuviese vacía, se imprimirá el siguiente mensaje:

  `+␣Error:␣Delete␣not␣possible`

  Al igual que la operación anterior, hay que recordar que cada usuario tiene una lista de reproducción de canciones que debe gestionarse adecuadamente al eliminarlo.

- Si la operación es **`[U]pgrade`**, se buscará al usuario en la lista, se actualizará su categoría a `pro` y se mostrará el siguiente mensaje:

  `*␣Upgrade:␣user␣XX␣category␣YY`

Si no existiese ningún usuario con ese nombre, si el usuario ya fuese `pro` o si la lista de usuarios estuviese vacía, se imprimirá el mensaje:

```
+ Error: Upgrade not possible
```

- Si la operación es **[A]dd**, se buscará el usuario y, si no hay ninguna canción con ese título (`songTitle`) en su lista de reproducción, la canción se añadirá **al final** de ésta. El valor inicial de su campo **playTime** será 0. A continuación, se mostrará el mensaje:

  ```
  * Add: user XX adds song YY
  ```

  Si la canción ya estuviese en la lista de canciones, no existiese ningún usuario con ese nombre o la lista de usuarios estuviese vacía, se imprimirá el mensaje:

  ```
  + Error: Add not possible
  ```

- Si la operación es **[P]lay**, se buscará el usuario para, a continuación, buscar la canción en su lista de reproducción. Si la canción está en la lista, se incrementarán en los minutos indicados (`minutes`) tanto el contador de reproducción de la propia canción (`playTime`) como el contador de reproducción total del usuario (`totalPlayTime`). Tras realizar este incremento, se mostrará el siguiente mensaje:

  ```
  * Play: user XX plays song YY playtime ZZ totalplaytime TT
  ```

  Si la canción no estuviese en lista de canciones del usuario, no existiese ningún usuario con ese nombre o si la lista de usuarios o la lista de canciones del usuario estuviesen vacías, se imprimirá el mensaje:

  ```
  + Error: Play not possible
  ```

- Si la operación es **[S]tats** se mostrará la lista completa de usuarios actuales incluyendo todas las canciones de sus correspondientes listas de reproducción, tal y como se muestra en el siguiente ejemplo (nótese que si la lista de canciones del usuario estuviese vacía, se indicará como "`No songs`"):

  ```
  User XX1 category pro totalplaytime ZZ1
  Song SS11 playtime TT11
  Song SS12 playtime TT12
  Song SS1m playtime TT1m

  User XX2 category pro totalplaytime ZZ2
  No songs

  User XXn category basic totalplaytime ZZn
  Song SSn1 playtime TTn1
  Song SSn2 playtime TTn2
  Song SSnm playtime TTnm
  ```

Debajo de esta lista se mostrará una tabla donde, para cada categoría de usuario, se indica el número de usuarios, el número total de minutos reproducidos, y la media de los minutos reproducidos (con dos decimales). El formato de dicha tabla es el siguiente:

```
Category  Users  TimePlay  Average
Basic     %5d  %9d  %8.2f
Pro       %5d  %9d  %8.2f
```

Si la lista de usuarios estuviese vacía, se imprimirá el mensaje:

```
+ Error: Stats not possible
```

- Si la operación es [**R]emove**, se eliminarán de la lista aquellos usuarios `basic` que hayan excedido el tiempo máximo de reproducción (`maxTime`). Se mostrará el siguiente mensaje:

```
Removing user XX1 totalplaytime ZZ1
Removing user XX3 totalplaytime ZZ3
...
Removing user XXn totalplaytime ZZm
```

Si no existiese ningún usuario de tipo `basic` que hubiese excedido el tiempo máximo o si la lista de usuarios estuviese vacía, se imprimirá el mensaje:

```
+ Error: Remove not possible
```

Hay que recordar que cada usuario tiene una lista de reproducción de canciones que debe gestionarse adecuadamente al ser eliminado.

4. Ejecución de la Práctica

Para facilitar el desarrollo de la práctica se proporciona el siguiente material de especial interés: (1) Un directorio `CLion` que incluye un proyecto plantilla (`P2.zip`); y (2) un directorio `script` donde se proporciona un fichero (`script.sh`) que permite probar de manera conjunta los distintos archivos proporcionados. Además, se facilita un documento de ayuda para su ejecución (`Ejecucion_Script_P2.pdf`). Nótese que, para evitar problemas al ejecutar el script, se recomienda **NO hacer copia-pega directo en el código de texto de este documento**, ya que el formato PDF puede incluir caracteres invisibles que darían por incorrectas salidas (aparentemente) válidas.

5. Documentación del código

El código deberá estar convenientemente **comentado**, incluyendo las variables empleadas. Los comentarios han de ser concisos pero explicativos. Asimismo, después de la cabecera de cada procedimiento o función del programa y/o librería, se incluirá la siguiente

información correspondiente a su **especificación**, tal y como se explicó en el TGR correspondiente:

- *Objetivo* del procedimiento/función.
- *Entradas* (identificador y breve descripción, una por línea).
- *Salidas* (identificador y breve descripción, una por línea).
- *Precondiciones* (condiciones que han de cumplir las entradas para el correcto funcionamiento de la subrutina).
- *Postcondiciones* (otras consecuencias de la ejecución de la subrutina que no quedan reflejadas en la descripción del objetivo o de las salidas).

6. Información importante

El documento `NormasEntrega_CriteriosEvaluacion.pdf`, disponible en la página web de la asignatura detalla claramente las normas de entrega.

Para un mejor seguimiento de la práctica se realizarán **dos entregas parciales obligatorias** antes de las fechas límite y con los contenidos que se indican a continuación:

1. **Entrega parcial #1: viernes 12 de abril a las 22:00 horas:** Implementación y prueba del TAD UserList y TAD SongList (entrega de los ficheros `types.h`, `user_list.h`, `user_list.c`, `song_list.h` y `song_list.c`). Para comprobar el correcto funcionamiento de los TAD se facilitarán los ficheros de prueba `test_user_list.c` y `test_song_list.c`.

2. **Entrega parcial #2: viernes 19 de abril a las 22:00 horas:** Implementación y prueba de las siguientes funcionalidades del programa principal: `[N]ew`, `[A]dd` y `[S]tats` (entrega de los ficheros `user_list.h`, `user_list.c`, `song_list.h`, `song_list.c`, `types.h` y `main.c`). Para comprobar el correcto funcionamiento de las operaciones se facilitarán los ficheros de prueba `new.txt` y `add.txt`.

Se realizará una corrección automática usando el script proporcionado para ver si se supera o no el seguimiento (véase el documento `NormasEntrega_CriteriosEvaluacion.pdf`).

Fecha límite para **entrega final**: **viernes 26 de abril de 2024 a las 22:00 horas.**