

DESCRIPCION ENTREGA 2 DEL PROYECTO

1. Diseño del TAD Canción

Para el diseño del TAD Canción, se tomarán la siguiente conceptualización y operaciones primitivas. Usted debe adaptar el diseño de la entrega 1 de su proyecto para que cumpla todas las especificaciones dadas aquí, es decir, nombres, tipos, parámetros, entre otros.

1.1. Conceptualización del TAD

CANCION: (Nombre, Duración, Tamaño, Género, Intérprete, Album)
Donde Nombre es de la forma String.ext
Duración es de la forma MM:SS con $MM, SS \in \mathbb{Z}$
Tamaño $\in \mathfrak{R}$
Género, Intérprete, Album \in String

1.2. Operaciones del TAD Canción :

1.2.1. Primitivas

```
C_Crear: String x
         int x
         int x float x String x String x String → CANCION
/* PRE: Dados el nombre, min, seg, tamaño, género, intérprete
        y  $0 \leq \text{min} < 60$  y  $0 \leq \text{seg} < 60$ 
        POST: Crea un objeto del tipo CANCION */

C_Destruir: CANCION → void
/* PRE: Dado un objeto CANCION
        POST: Libera el espacio ocupado por el objeto */

C_Nombre: CANCION → String
/* PRE: Dado un objeto CANCION
        POST: Devuelve el nombre de la canción sin la extensión */

C_Extension: CANCION → String
/* PRE: Dado un objeto CANCION
        POST: Devuelve la extensión del nombre de la canción */

C_DuracionMin: CANCION → int
/* PRE: Dado un objeto CANCION
        POST: Devuelve la duración (minutos) de la canción */

C_DuracionSeg: CANCION → int
/* PRE: Dado un objeto CANCION
        POST: Devuelve la duración (segundos) de la canción */

C_Genero: CANCION → String
/* PRE: Dado un objeto CANCION
        POST: Devuelve el género de la canción */

C_Inteprete: CANCION → String
/* PRE: Dado un objeto CANCION
        POST: Devuelve el intérprete de la canción */
```

```

C_Album: CANCION → String
/* PRE: Dado un objeto CANCION
   POST: Devuelve el album de la cancion */

C_CambiarNombre: CANCION x String → CANCION
/* PRE: Dado un objeto CANCION y un String
   POST: Devuelve el objeto CANCION modificado con el nuevo
         Nombre */

C_CambiarDuracionMin: CANCION x int → CANCION
/* PRE: Dado un objeto CANCION y un entero m tal que  $0 \leq m < 60$ 
   POST: Devuelve el objeto CANCION modificado con la nueva
         Duración en minutos */

C_CambiarDuracionSeg: CANCION x int → CANCION
/* PRE: Dado un objeto CANCION y un entero s tal que  $0 \leq s < 60$ 
   POST: Devuelve el objeto CANCION modificado con la nueva
         Duración en segundos */

C_CambiarGenero: CANCION x String → CANCION
/* PRE: Dado un objeto CANCION y un String
   POST: Devuelve el objeto CANCION modificado con el nuevo
         Genero */

C_CambiarInterprete: CANCION x String → CANCION
/* PRE: Dado un objeto CANCION y un String
   POST: Devuelve el objeto CANCION modificado con el nuevo
         Interprete */

C_CambiarAlbum: CANCION x String → CANCION
/* PRE: Dado un objeto CANCION y un String
   POST: Devuelve el objeto CANCION modificado con el nuevo
         Album */

C_Reproducir: CANCION → void
/* PRE: Dado un objeto CANCION
   POST: Reproduce la canción en el Simulador */

```

1.2.2. No Primitivas

```

C_Imprimir: CANCION → void
/* PRE: Dado un objeto CANCION
   POST: Imprime en pantallas los datos de la canción */

```

2. Diseño del TAD Lista de Canciones

2.1. Conceptualización del TAD

LC: (c_1, c_2, \dots, c_n) donde $c_i \in \text{CANCION}$, existe un elemento ventana c_k que corresponde a la canción que se está reproduciendo

2.2. Operaciones del TAD Lista de Canciones :

2.2.1. Primitivas

LC_Crear: void \rightarrow LC

/* POST: Crea una lista vacia de canciones */

LC_Destruir: LC \rightarrow void

/* PRE: Dado una lista de canciones creada

POST: Libera el espacio ocupado por la lista */

LC_Agregar: LC x CANCIÓN \rightarrow LC

/* PRE: Dada una canción y una lista de canciones creadas

POST: Agrega la canción al final de la lista de canciones
*/

LC_VerCancion: LC \rightarrow CANCIÓN

/* PRE: Dada una lista de canciones creada

POST: Devuelve la canción señalada por la ventana */

LC_Tamano: LC \rightarrow int

/* PRE: Dada una lista de canciones creada

POST: Devuelve el tamano actual de la lista */

LC_Primerero: LC \rightarrow LC

/* PRE: Dada una lista de canciones creada

POST: Mueve la ventana al primer elemento de la lista*/

LC_Ultimo: LC \rightarrow LC

/* PRE: Dada una lista de canciones creada

POST: Mueve la ventana al ultimo elemento de la lista*/

LC_Siguiente: LC \rightarrow LC

/* PRE: Dada una lista de canciones creada y ventana definida

POST: Mueve la ventana al elemento siguiente de la lista*/

LC_Anterior: LC \rightarrow LC

/* PRE: Dada una lista de canciones creada y ventana definida

POST: Mueve la ventana al elemento anterior de la lista*/

LC_Vacia: LC \rightarrow BOOL

/* PRE: Dada una lista de canciones creada

POST: Dice si la lista está vacía */

LC_Inicio: LC \rightarrow BOOL

/* PRE: Dada una lista de canciones creada y ventana definida

POST: Dice si la ventana esta sobre el primer elemento de
la lista */

LC_Final: LC \rightarrow BOOL

/* PRE: Dada una lista de canciones creada y ventana definida

POST: Dice si la ventana esta sobre el ultimo elemento de
la lista */

2.2.2. No Primitivas

```
LC_Cargar: String → LC
/* PRE: Dada un nombre de archivo con formato valido
   POST: Carga la información del archivo en la lista
         Si el tamaño del archivo es mayor que la capacidad
         del reproductor, se lo indica con un mensaje al
         usuario y sólo carga hasta su capacidad*/

LC_ReproducirTodas: LC → void
/* PRE: Dada una lista de canciones creada
   POST: Reproduce todas las canciones de la lista. Sigue
         reproduciendo desde el principio hasta que el
         usuario cambie el modo de reproducción o desee
         salir. También debe permitir que el usuario cambie
         de dirección la reproducción en cualquier punto donde
         se encuentre la misma.
*/

LC_ReproducirPorGenero: LC x String → void
/* PRE: Dada una lista de canciones creada y un genero
   POST: Reproduce las canciones de la lista que tienen el
         Genero dado. Sigue reproduciendo desde el principio
         hasta que el usuario cambie el modo de reproducción
         o desee salir. También debe permitir que el usuario
         cambie de dirección la reproducción en cualquier
         punto donde se encuentre la misma
*/

LC_ReproducirAleatorio: LC → void
/* PRE: Dada una lista de canciones creada
   POST: Reproduce las canciones de la lista de manera
         aleatoria. Sigue reproduciendo de manera aleatoria
         desde el principio hasta que el usuario cambie el
         modo de reproducción o desee salir. En este caso se
         la reproducción puede ir en cualquier dirección:
         hacia delante o hacia atrás.
*/
```

3. Implementación del TAD Lista de Canciones

Usted debe implementar este TAD usando listas doblemente enlazadas circulares. Note que esto le permitirá moverse en diferentes direcciones y continuar la reproducción de las canciones indefinidamente, mientras que el usuario lo desee.

Note que la operación de “Cargar lista” sólo debe realizarse al iniciar (“prender”) el reproductor. Y antes de apagarlo se debe destruir la lista. No esta permitido cargar listas auxiliares para la reproducción por género. En este caso se debe recorrer la lista buscando el género deseado.

4. Diseño del Archivo

El formato del archivo es el siguiente para cada canción:

```
<nombre de la canción>.xxx  
MM:SS // Corresponde a la duracion  
DDDD.DD // Corresponde al tamaño  
<genero>  
<interprete>  
<album>
```

Cada canción estaría separada por una línea

Ejemplo:

```
Avispas.mp3  
03:07  
135.87  
Merengue  
Juan Luis Guerra  
Para ti
```

```
La Camisa Negra.mp3  
04:76  
123:54  
PopFolk  
Juanes  
Mi Sangre
```

Para facilitar la lectura de una línea completa utilice la función `fgets`, la cual lee hasta el salto de línea (`'\n'`) colocando lo leído en un string. Posteriormente, usted haría el procesamiento del string leído.

5. Programa principal

El programa principal debe proveer opciones para

- ❖ Cargar Lista de Canciones
- ❖ Imprimir Lista
- ❖ Reproducir
- ❖ Salir

En el caso de la opción Reproducir se debe proveer un submenú donde el usuario pueda escoger entre los tres modos de reproducción (secuencial, por género o aleatoria). En todos los casos debe permitirse al usuario interrumpir la reproducción para regresarse al menú de selección de modo de reproducción o salir. En los dos primeros casos también se debe permitir cambiar la dirección de reproducción ascendente o descendente. Tome como defecto la ascendente.