Programación 2
Curso 2013/2014

Práctica 2
MiniTunes (parte 2)

En esta práctica se amplían las funcionalidades de la primera parte, añadiendo las opciones restantes. La práctica 3 consistirá en implementar esta misma práctica pero usando programación orientada a objetos.

Normas generales

1. El último día para entregar esta parte de la práctica es el **viernes 4 de abril, hasta las 23:59**. No se admitirán entregas fuera de plazo. Se realizará una entrega parcial el **viernes día 28 de marzo**, en la cual el programa deberá pasar un mínimo de pruebas del autocorrector para sumar 1 punto a la nota de la práctica.

2. Se debe entregar toda la práctica en un único fichero fuente llamado "`minitunes.cc`".

Normas generales comunes a todas las partes de la práctica

1. Lee atentamente estas instrucciones.

2. La práctica se debe entregar exclusivamente a través del servidor de prácticas del DLSI, al que se puede acceder desde la página principal del departamento (`www.dlsi.ua.es`, "Entrega de prácticas") o directamente en `http://pracdlsi.dlsi.ua.es`.

   - No se admitirán entregas por otros medios (correo electrónico, Campus Virtual, etc.).
   - El usuario y contraseña para entregar prácticas es el mismo que se utiliza en el Campus Virtual.
   - La práctica se puede entregar varias veces, pero sólo se corregirá la última entrega.

3. El programa debe poder ser compilado sin errores con el compilador de C++ existente en la distribución de Linux de los laboratorios de prácticas; si la práctica no se puede compilar su calificación será 0. Se recomienda compilar y probar la práctica con el autocorrector inmediatamente antes de entregarla.

4. La práctica debe ser un trabajo original de los alumnos; **en caso de detectarse indicios de copia de una o más prácticas se suspenderá la asignatura a todos los alumnos implicados**.

5. Los ficheros fuente deben estar adecuadamente documentados, con comentarios donde se considere necesario. Además, no se pueden utilizar variables globales de ninguna clase, los únicos símbolos globales permitidos son las funciones, los tipos y las constantes.

6. La corrección de la práctica se hará de forma automática, por lo que es imprescindible respetar estrictamente los formatos de salida que se indican en este enunciado. Además, al principio de todos los ficheros fuente entregados se debe incluir un comentario con el DNI de los alumnos que entregan la práctica, con el siguiente formato:

   `// DNI1` *tuDNI* *Primer nombre de la pareja*
   `// DNI2` *tuDNI* *Segundo nombre de la pareja*

   donde *tuDNI* es el DNI del alumno correspondiente (sin letras), **tal y como aparece en el Campus Virtual**, y *Nombre* es el nombre completo; por ejemplo, el comentario podría ser:

   ```
   // DNI1   12345678  GARCIA GARCIA, JUAN MANUEL
   // DNI2   87654321  PEREZ PEREZ, ANA
   ```

7. El cálculo de la nota de la práctica y su influencia en la nota final de la asignatura se detallan en las transparencias de la presentación de la asignatura.

8. Si quieres implementar más rápido la práctica, **imprime** este enunciado. Es mucho mejor que ir intercalando continuamente en la pantalla el código y el enunciado.

9. En caso de dudas sobre lo que debe hacer el programa, puedes probar el fichero ejecutable que también se proporciona con los materiales para ver cómo debería comportarse.

Funcionamiento de la práctica

El objetivo de esta segunda práctica es implementar las funcionalidades restantes del programa, que incluyen la gestión de playlist, carga y almacenamiento de datos, gestión de argumentos del programa y reproducción de las canciones.

Registros

En esta práctica es necesario añadir un nuevo registro para guardar un playlist, que es una lista de canciones seleccionadas de la colección. Lo implementaremos declarando un vector que contiene los identificadores de las canciones elegidas. Para comenzar añade el siguiente registro a tu código:

```
typedef struct
{
  vector<int> idSong;
} Playlist;
```

...y declara una variable `playlist` en el `main`.

Manage playlist

Esta opción nos permitirá añadir o borrar canciones en el playlist. Inicialmente debe mostrar el siguiente menú:

```
1- Add songs
2- Remove songs
3- Clear
q- Back to main menu
Option:
```

Si el usuario no introduce una opción válida[1], debe mostrarse el mensaje `Error: Unknown option` y a continuación de nuevo este menú. Cuando finalice alguna de las opciones volverá a mostrarse este menú, y sólo se volverá al menú principal con `q`. A continuación se indica lo que debe hacer cada opción.

1-Add songs

Esta opción permite añadir una serie de canciones al playlist. Si la colección no contiene canciones, se debe mostrar el mensaje `Empty collection`. En caso contrario, se mostrarán las canciones de la colección mediante `showCollection` y a continuación el mensaje:

```
Select songs (list separated by spaces):
```

El usuario introducirá una serie de identificadores separados por espacios, por ejemplo: `1 10 4 1 53`. Si algún identificador de canción no está en la colección, se mostrará el mensaje:

```
Error: Unknown song x
```

siendo `x` el identificador no encontrado. Las canciones que sí estén en la colección se añadirán al playlist.

2- Remove songs

Esta opción permite eliminar canciones del playlist. Si este no contiene ninguna canción debe mostrarse el mensaje `Empty playlist`. En caso contrario, se mostrarán todas las canciones del playlist y a continuación el mensaje:

[1]Por simplificar, se supondrá que el usuario sólo introducirá un carácter

```
Select songs (list separated by spaces):
```

El usuario introducirá una serie de identificadores separados por espacios, por ejemplo: `1 10`. Si algún identificador no está en el playlist, se mostrará el mensaje:

```
Error: Song x not in playlist
```

siendo `x` el identificador no encontrado. Las canciones que sí estén en el playlist se borrarán de la misma, pero no de la colección. El playlist puede tener identificadores de canciones duplicados, por lo que el borrado puede ser de más de un elemento. Con el ejemplo `1 10 4 1 53`, si borramos la canción `1`, el playlist quedaría con `10 4 53`.

3- Clear

Si el playlist no está vacío, se mostrará el mensaje:

```
Clear playlist? (Y/N):
```

Si el usuario elige la opción `Y`, se vaciará el playlist. En caso de que el playlist ya estuviera vacío se mostrará el mensaje `Empty playlist`.

Import iTunes JSON data

Esta opción debe pedir el nombre de un fichero de canciones con el mensaje `Filename:` y llamar a una función `importJSON` que abrirá y leerá este fichero en formato JSON. Si el fichero no se puede abrir, se mostrará el mensaje `Error opening file x`, siendo `x` el nombre del fichero. En caso contrario, se leerán las canciones de este fichero y se guardarán en la colección.

iTunes tiene un servicio JSON que permite obtener información sobre canciones (autor, álbum, etc). Puedes probarlo escribiendo en el terminal[2]:

```
wget itunes.apple.com/search?term="Beatles" -q -O beatles.txt
```

Este comando guarda en el fichero `beatles.txt` las entradas que se encuentran en la base de datos de iTunes[3] que contienen la palabra "Beatles". iTunes sólo devuelve como máximo 50 resultados de búsqueda.

Abre el fichero `beatles.txt` para ver su contenido. Para importar las canciones de este fichero a nuestro programa, tenemos que buscar las líneas que contienen `"kind":"song"`, y para cada una de ellas extraer los campos que aparecen en la tabla 1.

Por ejemplo, podemos buscar `"artistName":"` y leer lo que hay a continuación hasta el siguiente carácter `"`, almacenando el valor leído (por ejemplo, `The Beatles`) en el campo `artist`.

[2]Para esto debes tener instalado el programa `wget` y conexión a internet.

[3]Más información sobre las opciones de búsqueda en `http://www.apple.com/itunes/affiliates/resources/documentation/itunes-store-web-service-search-api.html`

iTunes	Song
trackName	title
collectionName	album
artistName	artist
primaryGenreName	genre
previewUrl	url

Tabla 1: Correspondencia de los campos JSON con el registro `Song`

Para simplificar el problema supondremos que cada canción está en una sola línea del fichero, aunque en algunos casos excepcionales esto no ocurre (por ejemplo, la del autor G.I en el fichero `beatles.txt`).

Si alguna canción del fichero ya estaba en la colección, debe mostrarse el siguiente mensaje sin añadirla:

```
The song x is already in the collection
```

siendo `x` el título de la canción. En caso contrario, la canción se añadirá a la colección.

Export playlist to XSPF

Esta opción permite guardar los datos del playlist en un fichero usando el formato XSPF[4]. Para ello se debe llamar a la función `saveXSPFPlaylist`.

Si el playlist está vacío, esta función mostrará el mensaje `Empty playlist` y finalizará devolviendo `false`. En caso contrario, pedirá un nombre de fichero con el mensaje `Filename:`

Si el fichero no se puede abrir, se mostrará `Error opening file x`, siendo `x` el nombre del fichero. En caso contrario, se guardará en el fichero el playlist siguiendo el formato del siguiente ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<playlist version="1" xmlns="http://xspf.org/ns/0/">
  <title>MiniTunes playlist</title>
  <tracklist>
    <track>
        <title>Sultans of Swing</title>
        <creator>Dire Straits</creator>
        <album>Dire Straits</album>
        <location>http://a369.phobos.apple.com/us/r1000/062/Music/0e/15/b3/mzm.cbhfpcxv.aac.p.m4a</location>
        <meta rel="genre">Pop</meta>
    </track>
    <track>
        <title>The Pretender</title>
        <creator>Foo Fighters</creator>
        <album>Echoes, Silence, Patience & Grace</album>
        <location>http://a348.phobos.apple.com/us/r1000/063/Music/df/a5/c0/mzm.khobgtnb.aac.p.m4a</location>
        <meta rel="genre">Alternative</meta>
    </track>
  </trackList>
</playlist>
```

[4] `http://xspf.org/`

Las primeras cuatro líneas serán siempre iguales, y a continuación se escribirá la información de todas las canciones del playlist en el formato del ejemplo.

Save data

Mediante esta opción se podrá guardar los datos de la colección y del playlist en un fichero binario llamado `minitunes.dat`.

Si la colección está vacía, debe mostrarse el mensaje `Empty collection` y salir sin hacer nada más. En caso contrario, se abrirá el fichero para escribir la información del playlist y de la colección.

Si el fichero no se puede abrir se mostrará el mensaje `Error opening file minitunes.dat`. Si se ha podido abrir se guardará el playlist escribiendo en el fichero su número de canciones y a continuación todos sus identificadores usando `write`.

Una vez guardado el playlist, se almacenará también la colección. Para ello, deberá escribirse el valor de la variable `idNextSong` y a continuación todas las canciones del vector en formato binario. Es necesario convertir cada canción a un registro `SongBin`:

```
const int kTITLE=30;
const int kARTIST=60;
const int kALBUM=60;
const int kGENRE=30;
const int kURL=255;

typedef struct
{
  int id;
  char title[kTITLE];
  char artist[kARTIST];
  char album[kALBUM];
  char genre[kGENRE];
  char url[kURL];
} SongBin;
```

Load data

Mediante esta opción se podrán cargar los datos de la colección y del playlist desde el fichero binario `minitunes.dat`. Si la colección estaba vacía, se cargarán los datos sin pedir confirmación de borrado. En caso contrario, se mostrará el mensaje

```
Delete previous collection? (Y/N):
```

Si el usuario introduce `Y`, se llamará a la función `loadData`, que debe leer el fichero binario. Si no se ha podido abrir se mostrará el mensaje `Error opening file minitunes.dat`. En caso contrario la función llamará a `readPlaylist` y `readCollection` para reemplazar los datos de ambos por los del fichero.

Play playlist

Para esta parte es necesario usar los ficheros `Player.cc`, `Player.h` y `Makefile` que se proporcionan con los materiales. Descárgalos en la carpeta donde tengas tu código y añade al principio de tu fichero `minitunes.cc` la siguiente línea:

```
#include "Player.h"
```

A partir de ahora, para compilar el programa deberás escribir `make` desde el terminal. Para poder reproducir las canciones, también necesitarás tener instalado el programa `mplayer`, que puedes descargar desde el repositorio de tu distribución Linux.

Cuando el usuario escoja esta opción, se mostrarán todas las canciones del playlist en formato `id: title (artist)`, por ejemplo:

```
>35: Here Comes the Sun (The Beatles)
 1: Money for Nothing (Dire Straits)
 76: Everlong (Foo Fighters)
 47: Eleanor Rigby (The Beatles)
Command:
```

y comenzará a reproducirse la primera canción del playlist llamando a la función `playSong` que se proporciona con los materiales. La canción que se esté reproduciendo actualmente debe aparecer con un símbolo `>` delante.

Mientras suena una canción el programa quedará a la espera de que el usuario introduzca un comando[5], que puede ser `p`, `s`, `a`, `q`. En caso de que no se introduzca uno de estos valores, el programa debe mostrar el siguiente mensaje de ayuda:

```
Options:
p: Pause/resume
s: Next song
a: Previous song
q: Quit
```

La reproducción sólo finalizará cuando el usuario introduzca `q`. Si la opción es `s`, se pasará a la siguiente canción, y si la canción actual ya es la última, comenzará a sonar la primera del playlist (la reproducción es circular). Si se introduce `a`, se pasará a la canción anterior, y si la actual es la primera, se reproducirá la última.

La opción `p` debe llamar a la función `pauseResumeSong` proporcionada para pausar la canción o reanudar la reproducción. Esta opción no debe volver a mostrar el playlist, a diferencia de las opciones anteriores. Cuando una canción termine de reproducirse no hay que hacer nada (no se reproduce automáticamente la siguiente).

Si el playlist estaba vacío cuando se ha elegido esta opción, se mostrará el mensaje `Empty playlist` y se volverá al menú principal.

[5]Por simplificar, se supondrá que el usuario sólo introducirá un carácter.

Argumentos del programa

El programa debe poder ejecutarse con las siguientes opciones desde línea de comandos:

- `-d`: Si se lanza el programa con esta opción, se cargarán los datos del fichero `minitunes.dat` y después se mostrará el menú principal.
- `-i fichero`: Con esta opción se cargarán en la colección los datos de un fichero de texto en formato iTunes JSON y después se mostrará el menú principal.

Ambas opciones son excluyentes (no pueden ponerse a la vez). Si el usuario introduce algún argumento incorrecto, el programa debe mostrar el siguiente mensaje y terminar su ejecución:

```
Syntax: ./minitunes [-d] [-i jsonfilename]
```

Si con las opciones `-d` o `-i` hay algún error cargando el archivo, el programa debe continuar mostrando el menú aunque no se hayan leído nuevos datos.

Funciones a implementar

El programa debe tener todas aquellas funciones que sea necesario y conveniente para su funcionamiento y legibilidad. Además de las funciones de la práctica anterior, es **obligatorio** que al menos tenga las siguientes con los mismos nombres y parámetros (aunque se pueden añadir más):

- `void managePlaylist(Playlist &playlist, const Collection &collection)`

  Función para mostrar el menú de la opción `Manage Playlist` y gestionarlo. Debe llamar a las funciones `addSongsPlaylist`, `removeSongsPlaylist` o `clearSongsPlaylist` dependiendo de la opción elegida por el usuario. Por simplificar, se supondrá que nunca se va a borrar una canción de la colección que también esté en el playlist. En teoría habría que propagar el borrado al playlist, pero en la práctica no vamos a hacerlo.

- `void addSongsPlaylist(Playlist &playlist, const Collection &collection)`

  Función para añadir las canciones seleccionadas por el usuario al playlist. Para mostrar la colección debe llamar a la función `showCollection`.

- `void removeSongsPlaylist(Playlist &playlist, const Collection &collection)`

  Función para eliminar las canciones seleccionadas por el usuario del playlist. Para mostrar el playlist debe llamar a la función `printPlaylist`.

- `void clearSongsPlaylist(Playlist &playlist)`

  Función para eliminar todas las canciones del playlist. Para borrar todos los elementos de un vector `v` puedes usar `v.clear();`

- `void printPlaylist(const Playlist &playlist, const Collection &collection, int position)`

  Función para mostrar el playlist. Para cada canción se llamará a `printPlaylistSong`, y la canción que se está reproduciendo actualmente (que está en la posición `position` del vector) se mostrará con un símbolo `>`. Si el valor de `position` es `-1`, no se mostrará el símbolo `>` delante de ninguna canción.

- `void printPlaylistSong(Song song)`

  Función para mostrar una canción en formato `id: title (artist)`

- `bool importJSON(Collection &collection, string filename)`

  Función para importar la colección de un fichero en formato JSON. Para cada línea del fichero que contenga `"kind":"song"` se llamará a la función `getSongFromJSONline` para extraer los datos. Si la canción ya estaba en la colección (`isSongInCollection` devuelve `true`), no debe insertarse en el vector. Si el fichero no se ha podido abrir, la función debe devolver `false`.

- `Song getSongFromJSONline(string JSONline)`

  Función para extraer los datos de una canción a partir de una línea en formato JSON.

- `bool saveXSPFPlaylist(const Playlist &playlist, const Collection &collection)`

  Función que pide el nombre de un fichero para guardar en él las canciones del playlist. Para cada canción, llamará a la función `printXSPFSong`. Si el playlist está vacío o el fichero no se ha podido abrir, devolverá `false`.

- `void printXSPFSong(Song song, ofstream &f)`

  Función para imprimir una canción en formato XSPF. Por ejemplo:

```
<track>
    <title>Sultans of Swing</title>
    <creator>Dire Straits</creator>
    <album>Dire Straits</album>
    <location>http://a369.phobos.apple.com/us/r1000/062/Music/0e/15/...</location>
    <meta rel="genre">Pop</meta>
</track>
```

- `bool saveData(const Collection &collection, const Playlist &playlist)`

  Función que abre el fichero binario `minitunes.dat` para escribir los datos del playlist y la colección. Para ello debe llamar a las funciones `writePlaylist` y `writeCollection`.

- `bool writePlaylist(const Playlist &playlist, ofstream &f)`

  Función para escribir el playlist en el fichero binario que se pasa por referencia. Se almacenará el número (`int`) de elementos del vector `idSong` y a continuación sus valores. Los métodos `readPlaylist`, `readCollection`, `writePlaylist` y `writeCollection` deben devolver `false` si el fichero que se pasa por parámetro no está abierto.

- `bool writeCollection(const Collection &collection, ofstream &f)`

  Función para escribir la colección en el fichero binario que se pasa por referencia. Se almacenará el valor de `idNextSong` y a continuación todas las canciones pasadas a formato binario mediante la función `songToBinary`.

- `SongBin songToBinary(const Song &song)`

  Función para pasar una canción (`Song`) a binario (`SongBin`).

- `bool loadData(Collection &collection, Playlist &playlist)`

  Función que abre el fichero binario `minitunes.dat` para leer los datos del playlist y la colección. Para ello debe llamar a las funciones `readPlaylist` y `readCollection`. Si el fichero no se ha podido abrir, no debe hacerse la llamada a las funciones anteriores y `loadData` debe devolver `false`.

- `bool readPlaylist(Playlist &playlist, ifstream &f)`

  Función para leer el playlist desde el fichero binario que se le pasa por referencia. Deben eliminarse los datos actuales del playlist antes de introducir los nuevos.

- `bool readCollection(Collection &collection, ifstream &f)`

  Función para leer la colección desde el fichero binario que se le pasa por referencia. Cada canción leída debe convertirse con `binaryToSong` para poder almacenarla en el vector de canciones (`songs`). Deben eliminarse los datos actuales de la colección antes de introducir los nuevos.

- `Song binaryToSong(const SongBin &songbin)`

  Función para convertir un registro `SongBin` en el registro `Song`.

- `void playPlaylist(const Playlist &playlist, const Collection &collection)`

  Función para gestionar la reproducción de canciones del playlist (opción `Play playlist`). Para ello debe declararse una variable de tipo `Player` al principio de esta función.

- `bool manageArguments(int argc, char *argv[], Collection &collection, Playlist &playlist)`

  Función para gestionar los argumentos del programa (`-d`, `-i`). Si los parámetros son correctos la función debe devolver `true`. Con la opción `-d` se debe llamar a la función `loadData`, y con `-i` a `importJSON`. Esta función debe devolver `false` si los argumentos son incorrectos.