Programación 2
Curso 2013/2014

Práctica 1
MiniTunes (parte 1)

Esta práctica es la primera parte de una aplicación de gestión de canciones. En posteriores prácticas se ampliarán las funcionalidades del mismo.

Normas generales

1. El último día para entregar esta parte de la práctica es el **viernes 21 de febrero, hasta las 23:59**. No se admitirán entregas fuera de plazo. Se realizará una entrega parcial el **viernes día 14 de febrero**, 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 el Campus Virtual se proporciona un fichero `minitunes.cc` como esqueleto para comenzar la práctica

10. En caso de dudas sobre lo que debe hacer el programa, puedes probar el fichero ejecutable que también se proporciona en el Campus Virtual para ver cómo debería comportarse.

Funcionamiento de la práctica

El objetivo de esta primera práctica es implementar las funcionalidades básicas del programa, que incluyen el menú principal y las opciones para añadir, editar, borrar y buscar canciones.

Menú principal

Al iniciar el programa se mostrará el siguiente menú[1]:

```
--------------------------------
----------- MiniTunes  ---------
--------------------------------
1- Add song
2- Edit song
3- Delete song
4- Show collection
5- Manage playlist
6- Import iTunes JSON data
7- Export playlist to XSPF
8- Save data
9- Load data
0- Play playlist
q- Quit
Option:
```

A continuación el usuario deberá introducir[2] una opción. En esta primera práctica sólo se implementarán las opciones 1, 2, 3 y 4, por lo que si se selecciona alguna de las otras no se hará nada y volverá a mostrarse el menú.

Si la opción seleccionada no está entre las válidas, se mostrará el mensaje `Error: Unknown option` y a continuación de nuevo el menú. El programa sólo finalizará cuando el usuario elija la opción `q`.

Registros

En el fichero `minitunes.cc` proporcionado se declaran dos registros, uno para almacenar una canción y otro para la colección de canciones.

Cada canción (`Song`) tiene un identificador numérico único (`id`), e información sobre su título, autor, album, género y url.

Para almacenar una lista de canciones usaremos un registro `Collection`. Una colección contiene un vector de canciones y un número (`idNextSong`) que indica el identificador numérico que debe tener la próxima canción que se añada a la colección. A continuación se explican las opciones del menú.

[1] No es necesario que el número de guiones de la cabecera coincida con el del ejemplo.

[2] Para simplificar la implementación, se supone que el usuario nunca introducirá más de un carácter.

Add song

Esta opción servirá para añadir una nueva canción a la colección. Para esto hay que solicitar los datos de la canción mediante los mensajes:

```
Title:
Artist:
Album:
Genre:
Url:
```

Tras mostrar cada uno de estos mensajes debe leerse la información introducida por el usuario y guardarla en su campo correspondiente.

A continuación se añadirá la canción a la colección sólo si no hay ya otra igual. Se considerará que dos canciones son iguales si tienen el mismo título y autor, sin consultar el resto de campos. En caso de que haya una canción igual en la colección, se debe mostrar el siguiente mensaje y no añadirla:

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

siendo `x` es el título de la canción. Si la canción no estaba en la colección, se añadirá a la misma.

Show collection

Al elegir esta opción debe mostrarse el mensaje `Search:` y pedir una cadena al usuario. Tras introducirlo, el programa debe mostrar todas las canciones que contienen este dato en los campos `title`, `artist`, `album` o `genre`. La url no se considerará en la búsqueda ni se mostrará en los resultados. Por ejemplo, si introducimos `Beat` y tenemos algunas canciones en la colección, una posible salida sería:

```
35 | Here Comes the Sun | The Beatles | Abbey Road | Rock
36 | Let It Be | The Beatles | Let It Be | Rock
37 | In My Life | The Beatles | Rubber Soul | Rock
```

El primer número es el identificador de la canción, y a continuación se imprimirá el título, autor, album y género. Para esto debe implementarse una función `printSong` que muestre por pantalla los datos de una canción (ver apartado de implementación al final del enunciado), y llamar a esta función por cada canción que cumpla los criterios de búsqueda.

Si la búsqueda no produce ningún resultado, debe imprimirse el mensaje `No results`.

En el caso de que el usuario introduzca un string vacío cuando se le solicite el término a buscar, deberán mostrarse todas las canciones de la colección.

Edit song

Si se selecciona esta opción, el programa debe mostrar primero las canciones como se indica en la opción `Show collection`, y a continuación el mensaje `Select song:` para que el usuario introduzca un número de canción, que no debe estar necesariamente en los resultados de búsqueda.

Si la colección no contiene una canción con ese identificador, se mostrará el mensaje `Error: Unknown song x`, siendo `x` el número introducido, y se saldrá de la función sin hacer nada más.

Si la canción sí está en la colección, se pedirá el campo a editar con el mensaje:

```
Edit (1-Title, 2-Artist, 3-Album, 4-Genre, 5-Url):
```

y el usuario podrá introducir una opción. Si dicha opción es incorrecta, deberá salir `Error: Unknown option`, volviendo a mostrarse el mensaje anterior hasta que el usuario introduzca un valor válido.

En caso de que el valor esté entre 1 y 5, se pedirá el dato elegido con el mensaje `Title:` , `Artist:` , `Album:` , `Genre:` o `Url:` , según corresponda, y se modificará el valor indicado en la canción de la colección. Por simplificar el problema, no deberá comprobarse si ya había en la colección una canción con los mismos datos.

Delete song

Al igual que en la opción anterior, si se selecciona esta opción el programa debe mostrar primero las canciones como se indica en la opción `Show collection`, y a continuación el mensaje `Select song:` para que el usuario introduzca un número de canción , que no debe estar necesariamente en los resultados de búsqueda.

Si la colección no contiene una canción con ese identificador, se mostrará el mensaje `Error: Unknown song x`, siendo `x` el número introducido, y se saldrá de la función sin hacer nada más.

En caso contrario, se pedirá al usuario la confirmación del borrado mediante el mensaje `Delete x? (Y/N):` , siendo `x` los datos de la canción tal como los muestra `printSong`.

Si el usuario introduce la opción `Y` (en mayúscula), se debe eliminar la canción de la colección. En cualquier otro caso, debe mostrarse el mensaje `Song not deleted` y no hacer nada más.

Funciones a implementar

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

- `bool isSongInCollection(const Collection &collection, Song song)`

  Función que busca la canción que se le pasa por parámetro en la colección. Si ya existe una con el mismo título y autor, debe devolver `true`, y en caso contrario `false`.

- `bool addSong(`Collection &collection)

  Función para añadir una canción a la colección. Si ya existe (para ello se debe llamar a la función anterior), debe devolver `false` y no añadirla. En caso contrario la canción debe añadirse a la colección con el identificador `id=idNextSong`, e incrementar `idNextSong` para la siguiente canción que vaya a insertarse en la colección.

  Para añadir un elemento `e` a un vector `v`, puedes usar: `v.push_back(e);`. Esta función debe llamar a `demandSong` para pedir los datos de la canción.

- Song demandSong()

  Función que pide por teclado todos los datos de la canción y la devuelve.

- `void printSong(Song song)`

  Función para imprimir una canción con su identificador, título, autor, album y género. Por ejemplo:

  `35 | Here Comes the Sun | The Beatles | Abbey Road | Rock`

- `int findIdSong(const Collection &collection, int id)`

  Función para buscar un identificador de canción (`id`) en la colección. Devuelve la posición del vector de canciones (`songs`) donde se ha encontrado, o `-1` si la canción no está en la colección.

- `bool showCollection(const Collection &collection)`

  Esta función mostrará por pantalla las canciones que cumplan el criterio de búsqueda solicitado al usuario. Si la búsqueda no produce ningún resultado, debe devolver `false`, y en caso contrario `true`. Para buscar una cadena en otra puedes usar el método `find`, por ejemplo:

  ```
  string s1="Hola";
  string s2="ol";
  if (s1.find(s2)!=string::npos)
     cout << "Found";
  else
     cout << "Not found";
  ```

- `void editSong(Collection &collection)`

  Función que muestra las canciones mediante `showCollection`. Si ésta devuelve `true`, a continuación pedirá el número de canción a editar, que se buscará en la colección mediante `findIdSong`. Si devuelve `false` no se hará nada. Tras localizar la canción se le pedirá al usuario que indique el campo a editar y se modificará esta canción en la colección.

- `void deleteSong(Collection &collection)`

  Función que muestra las canciones mediante `showCollection`. Si ésta devuelve `true`, a continuación pedirá el número de canción a borrar, que se buscará en la colección mediante `findIdSong`. Si devuelve `false` no se hará nada. Tras localizar la canción se le pedirá al usuario un mensaje de confirmación de borrado, y si su respuesta es afirmativa se eliminará la canción de la colección.

  Para borrar de un vector `v` el elemento de la posición `i`, puedes usar la siguiente función: `v.erase(v.begin()+i);`