PRÁCTICA 1 Estructuras de datos y algoritmos

Título: Programación de TAD genérico lineal en C++ Curso 2018/19

Objetivo:

- Diseñar una implementación lineal y en memoria dinámica en C++ del TAD genérico "*Recopilación*", y usarlo para implementar un TAD no genérico *"Álbum de canciones"*. Desarrollar un programa que gestione un álbum de canciones, de acuerdo a lo que se describe en el enunciado de la práctica.

Fecha límite de entrega: 7-11-2018 (incluido)

Descripción detallada:

Se trata de implementar un TAD genérico "*Recopilación*", particularizarlo y utilizarlo para implementar un TAD no genérico *"Álbum de canciones"* que sirva para gestionar una colección de canciones, y las pistas que ocupan, en un álbum de canciones. Por último, se implementará un programa que gestione un álbum de canciones y permita comprobar la correcta ejecución de las operaciones de los TADs implementados.

Tienes que realizar las siguientes **tareas**:

1. Desarrollar una implementación dinámica (utilizando un lista enlazada) en C++ del TAD genérico *Recopilación*, cuya especificación se incluye a continuación:

```
espec recopilación
  usa cadenas, booleanos, naturales[1], enteros
parámetros formales
  géneros clave, dato
  operaciones {se requiere que estén definidas las siguientes operaciones:}
    iguales: clave c1, clave c2 -> booleano {Devuelve cierto si y sólo si c1 es igual que c2}
    generaCadena: clave o -> cadena {transformación a cadena}
    generaCadena: dato v -> cadena {transformación a cadena}
    tamaño: dato v -> entero {Devuelve el "tamaño" de un dato v}
fpf
género recopilación {Los valores del TAD representan conjuntos de pares (clave,dato), en los que
no se permiten claves repetidas, y que cuentan con algunas operaciones de consulta y modificación
guiadas por clave, pero también con operaciones que permiten consultar, controlar y modificar la
posición relativa (puesto) de los pares dentro de la recopilación, e incluso operaciones para la
consulta y modificación de los pares según su puesto en la recopilación.
El puesto de un par en la recopilación vendrá dado en todo momento por su posición relativa
respecto al resto de los pares, de tal forma que: el par con menor puesto ocupará el puesto número
1; no podrá haber más de un par en el mismo puesto; y en ningún caso podrá haber puestos vacíos.

El TAD ofrece además las operaciones de un iterador que permite visitar los pares de una
recopilación por orden de puesto, empezando por el del puesto número uno, y acabando por el del
último puesto.}

operaciones
  crear: -> recopilación
   {Devuelve una recopilación vacía, sin elementos (pares).}

  cardinal: recopilación r -> natural
   {Devuelve el nº de pares en la recopilación r.}

  tamañoTotal: recopilación r -> entero
   {Devuelve el tamaño total de la recopilación r, es decir, la suma total de los tamaños de los
   datos de todos los pares en la recopilación r.}

  existe?: recopilación r , clave k -> booleano
   {Devuelve verdad si y sólo si en r existe un par con clave k.}

  introducir: recopilación r , clave k , dato v -> recopilación
   {Si en r no existe ningún par con clave k (not existe?(r,k)), devuelve la recopilación
   resultante de añadir en r un par (k,v) ocupando su último puesto. Si en r ya existe un par
   (k,v1) entonces devuelve la recopilación resultante de sustituir dicho par en r por un par
   (k,v), conservando el mismo puesto.}

  parcial obtenerDato: recopilación r , clave k -> dato
   {Si en r existe un par (k,v), devuelve su dato v.
   Parcial: la operación no está definida si not existe?(r,k).}

  recolocarEnPuesto: recopilación r , clave k, entero delta -> recopilación
   {Si not existe?(r,k), devuelve una recopilación igual a r. Si existe?(r,k), devuelve la
   recopilación resultante de modificar el puesto relativo del par (k,v) entre los pares en r,
   alejándolo delta posiciones del primer puesto de r si delta es positivo (o colocándolo el
   último si no hay tantos puestos), o acercándolo –delta posiciones al primer puesto de r si
```

[1] Los números naturales incluyendo el 0.

```
        delta es negativo (o colocándolo el primero, si no hay tantos puestos); en cualquier caso no se
        modifican las posiciones relativas entre sí del resto de los pares. Si delta<0, el par que
        ocupaba en r el puesto destino para el par con clave k, en la recopilación resultante quedará
        ocupando el puesto siguiente al que tenía en r. Si delta>0, el par que ocupaba en r el puesto
        destino para el par con clave k, en la recopilación resultante ocupará el puesto anterior al
        que tenía en r.}

      eliminarXPuesto: recopilación r , natural p -> recopilación
       {Si 1≤p≤cardinal(r) existe un par en r en el puesto p, entonces devuelve una recopilación igual
       a la resultante de eliminar de r el par que ocupaba el puesto p, con todos los demás pares
       iguales a los que hay en r, y en las mismas posiciones relativas. En caso contrario, devuelve
       una recopilación igual a r.}

      parcial obtenerXPuesto: recopilación r , entero p -> clave
       {Si 1≤p≤cardinal(r) existe un puesto p en r ocupado por un par (k,v), y devuelve su clave k.
       Parcial: la operación no está definida si no existe un puesto p en r.}

      parcial puestoDeClave: recopilación r , clave k -> natural
       {Si existe?(r,k), devuelve el puesto del par (k,v) en r.
       Parcial: la operación no está definida si not existe?(r,k).}

      iniciarIterador: recopilación r -> recopilación
       {Inicializa el iterador para recorrer los pares de la recopilación r, de forma que el siguiente
       par sea el del primer puesto de la recopilación r (situación de no haber visitado ningún par).}

      existeSiguiente?: recopilación r -> booleano
      {Devuelve verdad si y sólo si queda algún par por visitar con el iterador de la recopilación r.}

      parcial siguienteClave: recopilación r -> clave
       {Devuelve la clave del siguiente par a visitar de r.
       Parcial: la operación no está definida si ya se ha visitado el último par.}

      parcial siguienteDato: recopilación r -> dato
       {Devuelve el dato del siguiente par a visitar de r.
       Parcial: la operación no está definida si ya se ha visitado el último par.}

      parcial avanza: recopilación r -> recopilación
       {Prepara el iterador para visitar el par del siguiente puesto de la recopilación r.
       Parcial: la operación no está definida si ya se ha visitado el último par.}

    fespec
```

2. Diseñar e implementar un TAD *canción* que defina un nuevo tipo *canción* y que deberá poder utilizarse para representar toda la información que corresponda a un concepto de canción que incluya: el nombre de la canción, el nombre del autor (sendos datos de tipo cadena), el año de su composición (de tipo *entero*), y la duración en segundos de la canción (de tipo *entero*). El tipo *canción* se deberá implementar como TAD, de acuerdo a las indicaciones dadas en la asignatura, y ser diseñado con las propiedades y operaciones que se consideren adecuadas. Entre ellas deberá haber una operación que devuelva una cadena conteniendo toda la información de una canción con formato: *nombre --- autor --- año (duración)* que será necesario para cumplir con los formatos de la salida tal como se describen más adelante.

3. Utilizar la implementación dinámica del TAD genérico "*Recopilación*" del punto 1) para implementar en C++ el TAD *Álbum de Canciones* que se especifica a continuación. El objetivo del TAD *Álbum de Canciones* es gestionar toda la información de un álbum de canciones, incluyendo el título del álbum, y especialmente la recopilación de canciones incluidas en el álbum, así como las pistas que ocupan dichas canciones. Cada canción incluida en el álbum se identificará mediante una clave única de tipo *cadena* de caracteres, que permitirá distinguir unas canciones de otras. El tipo *canción* será el descrito en el apartado 2, y que se describió también en el enunciado de la práctica 0.

   La especificación del TAD *Álbum de Canciones* que vamos a considerar es:

```
spec álbumDeCanciones
  usa cadenas, canción[2], recopilación, booleanos, naturales, enteros
género álbum {Los valores del TAD representan álbumes de canciones, caracterizados por la siguiente
información: un título del álbum (cadena), y una recopilación de canciones identificadas en el álbum
por claves (únicas) de tipo cadena. Las operaciones del TAD permiten gestionar las canciones que ocupan
las pistas del álbum. La primera canción del álbum estará en la pista número 1. }

operaciones
  crear: cadena título -> álbum
  {Crea un álbum con el título dado y sin canciones.}

  númeroDeCanciones: álbum b -> natural
  {Devuelve el número de canciones en el álbum b}

  duración: álbum b -> entero
  {Devuelve la duración total del álbum b, es decir la suma de las duraciones de todas sus canciones.}
```

[2] Sea la especificación de un TAD *canción* de acuerdo a lo descrito en el apartado 2.

existeCanción?: álbum b ,cadena k -> booleano
{Devuelve verdad si y sólo si en el álbum b existe una canción identificada por la clave k.}

parcial obtenerCanción: álbum b ,cadena k -> canción
{Si existeCanción?(b,k), devuelve la canción identificada por la clave k en b.
Parcial: la operación no está definida si not existeCanción?(b,k).}

añadirCanción: álbum b , cadena k, canción c -> álbum
{Si not existeCanción?(b,k), devuelve el álbum resultante de añadir en b una canción c identificada por una clave k, ocupando la última pista entre las canciones del álbum.
Si existeCanción?(b,k), en b existe una canción c1 identificada por la clave k, entonces devuelve el álbum resultante de sustituir en b la canción c1 por la canción c, conservando la misma clave y la misma pista que c1 entre las canciones del álbum b.}

eliminarCanción: álbum b , natural p -> álbum
{Si 1≤p≤númeroDeCanciones(b), entonces existe una canción en b en la pista p, y devuelve un álbum igual al resultante de eliminar de b la canción que ocupaba la pista p, con todas las demás canciones(y sus claves) iguales a las que hay en b, y en las mismas posiciones relativas. En caso contrario, devuelve un álbum igual a b.}

parcial puestoDeCanción: álbum b, cadena k -> natural
{Si existeCanción?(b,k), devuelve la pista de la canción identificada por la clave k en b.
Parcial: la operación no está definida si not existeCanción?(b,k).}

parcial canciónDePuesto: álbum b, entero p -> cadena
{Si 1≤p≤númeroDeCanciones(b), entonces existe una canción en b en la pista p, y devuelve la clave con la que se identifica a la canción que ocupa dicha pista en b.
Parcial: la operación no está definida si no existe una pista p en b.}

intercambiarCanciones: álbum b, cadena k1, cadena k2 -> álbum
{Si existeCanción?(b,k1) AND existeCanción?(b,k2), devuelve el álbum resultante de intercambiar en b las pistas ocupadas por las canciones identificadas respectivamente por las claves k1 y k2, sin cambiar las claves que identifican a cada canción y sin modificar la pista o la información de ninguna otra canción del álbum. Si not existeCanción?(b,k1) OR not existeCanción?(b,k2), devuelve un álbum igual a b.}

listarÁlbum: álbum b -> cadena
{Devuelve una cadena que contiene la representación como cadena de caracteres de toda la información sobre todas las canciones del álbum, por orden de pista en sentido ascendente, y de tal forma que toda la información sobre una canción esté separada de la siguiente canción con el carácter de salto de línea; y para cada canción su información se incluya con el siguiente formato:
la cadena "n) ", siendo n su número de pista en el álbum b, seguida de la cadena que identifica a la canción en b, seguida a continuación de una cadena ":::", y seguida de una cadena con toda la información de la canción[3].
***La implementación de esta operación deberá hacerse obligatoriamente utilizando las operaciones del iterador de la recopilación de canciones del álbum**.}*

fespec

4. Utilizar la implementación del TAD del apartado 3), para implementar un programa de prueba que nos permita probar la implementación del TAD *Álbum de Canciones* y del TAD *Recopilación*.

   El código fuente del programa de prueba (*main*) deberá encontrarse en un fichero llamado *"practica1.cpp"* y cumplir escrupulosamente con el funcionamiento y formatos que se describen a continuación, o la práctica no será evaluada.

   El programa de prueba deberá crear un álbum de canciones vacío con título *"Mi lista de canciones"*, y a continuación leer las instrucciones de las operaciones a realizar con el álbum de canciones desde un fichero de texto denominado "entrada.txt". **Para resolver cada instrucción, el programa deberá utilizar las operaciones ofrecidas por el TAD *Álbum de Canciones*.** El fichero "entrada.txt" tendrá la siguiente estructura o formato:

```
<instrucción1>
<datos para la instrucción1>
...
<datos para la instrucción1>
<instrucción2>
<datos para la instrucción2>
...
<datos para la instrucción2>
...
<instrucciónÚltima>
<datos para la instrucciónÚltima>
...
<datos para la instrucciónÚltima>
```

[3] De acuerdo al formato descrito en el apartado 2.

Donde <instrucciónX> podrá ser alguna de las siguientes cadenas de caracteres: AC, LC, EP, LP, OA, LA, que representarán las operaciones de: *"Añadir una Canción al álbum"*, *"Listar todos los datos de una Canción"*, *"Eliminar la canción de una Pista*", *"Listar la canción de una Pista*", *"Organizar las pistas del Álbum*", y *"Listar todas las canciones del Álbum"*, respectivamente. El programa finalizará cuando haya procesado todas las instrucciones del fichero. Supondremos que el fichero "entrada.txt" tendrá siempre una estructura como la descrita, sin errores:

- Si la instrucción es *AC*, las 5 líneas siguientes contendrán los valores de: en la primera línea, la clave con la que se deberá introducir la canción en el álbum; en la segunda línea, el nombre de la canción; en la tercera línea, el autor de la canción; en la cuarta línea, el año en el que fue compuesta la canción; y en la quinta línea, el número de segundos que dura la canción.

- Si la instrucción es *LC*, en la siguiente línea del fichero aparecerá la clave de la canción del álbum que se deberá listar.

- Si la instrucción es *EP* o *LP*, en la siguiente línea del fichero aparecerá el número de pista de la canción del álbum que se deberá eliminar o listar, respectivamente.

- Si la instrucción es *OA*, en las siguientes dos líneas del fichero aparecerán las claves (una clave por línea) de las canciones del álbum que deberán intercambiar sus pistas.

- Si la instrucción es *LA*, la operación no necesita más datos, así que la siguiente línea en el fichero será la del inicio de la siguiente instrucción (o fin de fichero).

Cuando se procese una instrucción *AC*, el programa intentará introducir o modificar en el álbum una canción, utilizando la clave y los datos dados para la instrucción. Cuando se procese una instrucción *EP*, el programa intentará eliminar del álbum la canción que ocupa la pista dada para la instrucción. Cuando se procese una instrucción *LP*, el programa deberá listar en su salida la información que contiene el álbum sobre la canción que ocupa la pista dada para la instrucción. Cuando se procese una instrucción *LC*, el programa deberá listar en su salida la información que contiene el álbum sobre la canción con la clave dada para la instrucción. Cuando se procese una instrucción *OA*, el programa intentará intercambiar las pistas ocupadas por las canciones identificadas por las claves dadas para la instrucción. Cuando se procese una instrucción *LA*, el programa deberá listar en su salida la información sobre todas las canciones que contiene el álbum.

Como resultado de su ejecución, el programa creará un fichero de texto "salida.txt" en el que irá escribiendo el resultado de cada orden ejecutada, y que constará de las siguientes líneas:

a) Por cada instrucción de tipo *AC*: **si es posible introducir los datos de una nueva canción en el álbum**, se escribirá una línea en el fichero de salida que empiece con la cadena "INSERCION: "; **si no es posible introducir los datos de una nueva canción en el álbum pero sí que se actualiza una canción en el álbum**, se escribirá una línea en el fichero que empiece con la cadena "ACTUALIZACION: ". En cualquiera de los dos casos, se seguirá escribiendo en el fichero la concatenación de:

   1) la clave de la canción, seguida de la cadena ":::",

   2) seguida de la información de la canción con el siguiente formato:

      - la cadena "<* ", seguida del nombre de la canción, seguido de la cadena " --- ",
      - seguida del nombre del autor, seguido de la cadena " --- ",
      - seguida del año de composición de la canción, seguido de la cadena " (",
      - seguida del número de segundos de duración que tiene la canción, y
      - seguido de la cadena ")*>" , y finalmente un salto de línea.

   Es decir, una canción con el formato: *Clave:::<* nombre --- autor --- año (duración)*>*

b) Por cada instrucción de tipo *LC*: **si en el álbum se encuentra una canción con la clave dada**, se escribirá una línea en el fichero de salida que empiece con la cadena "ENCONTRADA: ", seguida del número de pista en la que se encuentra la canción, seguida de una cadena " … ", y seguida de la misma concatenación descrita en los puntos a.1–a.2, con la información de la clave, y datos de la canción en el álbum. **Si en el álbum no se encuentra una canción con la clave dada**, se escribirá una línea en el fichero de salida que empiece con la cadena "cancion DESCONOCIDA: ", seguida de la clave utilizada para la instrucción.

c) Por cada instrucción de tipo *EP*: **si en el álbum se encuentra una canción en la pista dada**, se escribirá una línea en el fichero de salida que empiece con la cadena "pista ELIMINADA: ", seguida del número de pista utilizada, seguido de una cadena " … ", y seguida de la información de la canción eliminada de acuerdo al formato descrito en los puntos a.1–a.2. **Si en el álbum no se encuentra una canción en la pista dada**, se escribirá una línea en el

fichero de salida que empiece con la cadena “eliminacion de pista INNECESARIA: ”, seguida del número de pista utilizada para la instrucción.

d) Por cada instrucción de tipo *LP*: **si en el álbum se encuentra una canción en la pista dada**, se escribirá una línea en el fichero de salida que empiece con la cadena “PISTA: ”, seguida del número de pista utilizada, seguido de una cadena “ … ”, y seguida de la concatenación descrita en los puntos a.1–a.2, con la información de la clave, y datos de la canción en dicha pista del álbum. **Si en el álbum no se encuentra una canción en la pista dada**, se escribirá una línea en el fichero de salida que empiece con la cadena “pista INEXISTENTE: ”, seguida del número de pista utilizada para la instrucción.

e) Por cada instrucción de tipo *OA*: **si en el álbum se encuentran las canciones con las claves dadas**, se escribirá una línea en el fichero de salida que empiece con la cadena “INTERCAMBIAR:”, seguida de un salto de línea, seguido de la cadena “pista A) ” seguida del número de pista ocupada (antes del intercambio) por la canción identificada por la primera clave dada para la instrucción, seguido de la cadena “ … ”, seguida de toda la información de dicha canción de acuerdo al formato descrito en los puntos a.1-a.2, finalizada con un salto de línea, seguido de la cadena “pista B) ” seguida del número de pista ocupada (antes del intercambio) por la canción identificada por la segunda clave dada para la instrucción, seguido de la cadena “ … ”, seguida de toda la información de dicha canción de acuerdo al formato descrito en los puntos a.1-a.2, finalizada con salto de línea. **Si en el álbum no se encuentra alguna de las canciones con las claves dadas**, se escribirá una línea en el fichero de salida que empiece con “intercambio IMPOSIBLE: ”, seguida de la primera clave utilizada para la instrucción, seguida de la cadena “ ### ”, seguida de la segunda clave utilizada para la instrucción.

f) Por cada instrucción de tipo *LA*: se escribirá en el fichero de salida una primera línea con la cadena “TITULO: “, seguida del título del álbum, seguido de un salto de línea, seguido de la cadena “DURACION TOTAL: ”, seguida de la duración total del álbum de canciones, seguida de un salto de línea, seguido de la cadena “NUMERO de canciones: ”, seguida del número total de canciones en el álbum, seguido de un salto de línea, y a continuación la información de todas las canciones del álbum, utilizando para cada una de ellas una nueva línea de texto que empiecen con una cadena “I) ”, siendo I el número de la pista ocupada por la canción, y seguida de toda la información de la canción de acuerdo al formato descrito en los puntos a.1–a.2. La información de las canciones se listará por orden de pista, de menor a mayor.

Al final se muestra un ejemplo de fichero “entrada.txt”, y su correspondiente fichero “salida.txt”.

Observaciones.

- **El código fuente entregado será compilado y probado en hendrix, que es donde deberá funcionar correctamente.**
- **El código fuente entregado deberá compilar correctamente con la opción –std=c++11 activada.**
  - Esto significa que si se trabaja con la línea de comandos, deberá compilarse con:
    ```
    g++ -std=c++11 ficheros_compilar...
    ```
  - Si se trabaja con CodeBlocks, el proyecto de la práctica deberá estar configurado con la opción `"Have g++ follow the C++11 ISO C++ language standard [-std=c++11]"` activada (localizable en los menús de CodeBlocks, seleccionando sucesivamente: `"Settings"->"Compiler"->->"Global compiler settings"-> pestaña "Compiler Flags"`)
- Todos los ficheros con código fuente que se presenten como solución de esta práctica deberán estar correctamente documentados.
- En el comentario inicial de cada fichero de código fuente se añadirán los nombres y NIAs de los autores de la práctica.
- **Los TADs deberán implementarse siguiendo las instrucciones dadas en las clases y prácticas de la asignatura, y no se permite utilizar Programación Orientada a Objetos.**
- No se permite usar las clases o componentes de la *Standard Template Library (STL)*, ni similares.
- **Todas las indicaciones que se dan en los enunciados de las prácticas respecto a nombres de ficheros, programas, opciones del programa, formatos de los ficheros de entrada o de los ficheros de salida que deban generarse, etc., deben cumplirse escrupulosamente para que la práctica sea evaluada.**
- El código fuente del programa de prueba (*main*) deberá encontrarse en un fichero llamado *“practica1.cpp”*, y cumplir escrupulosamente con el funcionamiento y formatos que se han descrito en el enunciado.
- La ruta del fichero de entrada deberá ser “entrada.txt” (no “entrada1.txt”, “mientrada.txt”, “datos/entrada.txt”, ni similares). Ídem para el fichero “salida.txt”.
- La salida debe seguir las especificaciones del enunciado. Por ejemplo, cuando escribimos “cancion DESCONOCIDA: ” en el fichero de salida, está escrito sin tilde, la primera palabra en minúsculas, la segunda palabra en mayúsculas, y con un espacio en blanco entre las dos palabras, y otro espacio en blanco tras el carácter ‘:’. De forma similar, será obligatorio cumplir todos los formatos descritos en este enunciado.

Material a entregar. Instrucciones.

- La práctica solo deberá someterla **uno** de los miembros del equipo de prácticas desde su cuenta de hendrix, y preferiblemente siempre el mismo para todas las prácticas.
- Conectarse a hendrix-ssh.cps.unizar.es según se explica en el documento "Realización y entrega de prácticas en los laboratorios del DIIS" disponible en moodle.
- Crear un directorio, llamado J_p1, si tu profesor tutor es Jorge Bernad, o Y_p1, si tu profesora tutora es Yolanda Villate, donde se guardará un directorio *practica1* que contenga todos los ficheros desarrollados para resolver la práctica (este directorio, *practica1*, deberá contener todos los ficheros con código fuente C++ necesarios para resolver la práctica y dos ficheros de texto, ***entrada.txt*** y ***salida.txt,*** con los formatos explicados pero que sean significativamente diferentes a los proporcionados como ejemplo, y con los que habréis probado la implementación realizada en vuestra práctica). A la hora de evaluar la práctica se utilizará tanto el fichero de prueba que se entregue, como ficheros de prueba entregados por otros compañeros, o ficheros propios de los profesores.
- Crear el fichero X_p1.tar, con X igual a J o Y, dependiendo de quién sea tu profesor tutor, con el contenido del directorio X_p1 ejecutando el comando

```
tar –cvf X_p1.tar X_p1
```

- Enviar el fichero X _p1.tar, con X igual a J o Y, dependiendo de quién sea tu profesor tutor, mediante la orden

```
someter –v eda_18 X_p1.tar
```

ADVERTENCIA: la orden `someter` no permite someter un fichero si el mismo usuario ha sometido antes otro fichero con el mismo nombre y para la misma asignatura, por lo tanto, **antes de someter vuestra práctica, aseguraos de que se trata de la versión definitiva que queréis presentar para su evaluación.**

entrada.txt

```
LC
C1
AC
CM_GM_Sinf3
Symphony N.3
Gustav Mahler
1896
6300
AC
BD_ult_cd2-13
Hurricane
Bob Dylan
1975
512
AC
LoTR_FR-cd3-1
Khazad-dum
Howard Shore
2001
480
LA
LC
BD_ult_cd2-13
EP
9
AC
BD_ult_cd2-13
Hurricane
Bob Dylan y J. Levy
1975
512
LA
AC
BD_ult_cd1-7
Boots Of Spanish
Leather
Bob Dylan
1964
278
AC
BD_ult_cd3-14
Dark Eyes
Bob Dylan
1985
309
LA
OA
BD_ult_cd1-7
BD_ult_cd2-13
LA
LP
9
LP
3
EP
3
LA
OA
C1
CM_GM_Sinf3
```

salida.txt

```
cancion DESCONOCIDA: C1
INSERCION: CM_GM_Sinf3:::<* Symphony N.3 --- Gustav Mahler --- 1896 ( 6300 )*>
INSERCION: BD_ult_cd2-13:::<* Hurricane --- Bob Dylan --- 1975 ( 512 )*>
INSERCION: LoTR_FR-cd3-1:::<* Khazad-dum --- Howard Shore --- 2001 ( 480 )*>
TITULO: Mi lista de canciones
DURACION TOTAL: 7292
NUMERO de canciones: 3
1) CM_GM_Sinf3:::<* Symphony N.3 --- Gustav Mahler --- 1896 ( 6300 )*>
2) BD_ult_cd2-13:::<* Hurricane --- Bob Dylan --- 1975 ( 512 )*>
3) LoTR_FR-cd3-1:::<* Khazad-dum --- Howard Shore --- 2001 ( 480 )*>
ENCONTRADA: 2 ... BD_ult_cd2-13:::<* Hurricane --- Bob Dylan --- 1975 ( 512 )*>
eliminacion de pista INNECESARIA: 9
ACTUALIZACION: BD_ult_cd2-13:::<* Hurricane --- Bob Dylan y J. Levy --- 1975 ( 512 )*>
TITULO: Mi lista de canciones
DURACION TOTAL: 7292
NUMERO de canciones: 3
1) CM_GM_Sinf3:::<* Symphony N.3 --- Gustav Mahler --- 1896 ( 6300 )*>
2) BD_ult_cd2-13:::<* Hurricane --- Bob Dylan y J. Levy --- 1975 ( 512 )*>
3) LoTR_FR-cd3-1:::<* Khazad-dum --- Howard Shore --- 2001 ( 480 )*>
INSERCION: BD_ult_cd1-7:::<* Boots Of Spanish Leather --- Bob Dylan --- 1964 ( 278 )*>
INSERCION: BD_ult_cd3-14:::<* Dark Eyes --- Bob Dylan --- 1985 ( 309 )*>
TITULO: Mi lista de canciones
DURACION TOTAL: 7879
NUMERO de canciones: 5
1) CM_GM_Sinf3:::<* Symphony N.3 --- Gustav Mahler --- 1896 ( 6300 )*>
2) BD_ult_cd2-13:::<* Hurricane --- Bob Dylan y J. Levy --- 1975 ( 512 )*>
3) LoTR_FR-cd3-1:::<* Khazad-dum --- Howard Shore --- 2001 ( 480 )*>
4) BD_ult_cd1-7:::<* Boots Of Spanish Leather --- Bob Dylan --- 1964 ( 278 )*>
5) BD_ult_cd3-14:::<* Dark Eyes --- Bob Dylan --- 1985 ( 309 )*>
INTERCAMBIAR:
pista A) 4 ... BD_ult_cd1-7:::<* Boots Of Spanish Leather --- Bob Dylan --- 1964 ( 278 )*>
pista B) 2 ... BD_ult_cd2-13:::<* Hurricane --- Bob Dylan y J. Levy --- 1975 ( 512 )*>
TITULO: Mi lista de canciones
DURACION TOTAL: 7879
NUMERO de canciones: 5
1) CM_GM_Sinf3:::<* Symphony N.3 --- Gustav Mahler --- 1896 ( 6300 )*>
2) BD_ult_cd1-7:::<* Boots Of Spanish Leather --- Bob Dylan --- 1964 ( 278 )*>
3) LoTR_FR-cd3-1:::<* Khazad-dum --- Howard Shore --- 2001 ( 480 )*>
4) BD_ult_cd2-13:::<* Hurricane --- Bob Dylan y J. Levy --- 1975 ( 512 )*>
5) BD_ult_cd3-14:::<* Dark Eyes --- Bob Dylan --- 1985 ( 309 )*>
pista INEXISTENTE: 9
PISTA: 3 ... LoTR_FR-cd3-1:::<* Khazad-dum --- Howard Shore --- 2001 ( 480 )*>
pista ELIMINADA: 3 ... LoTR_FR-cd3-1:::<* Khazad-dum --- Howard Shore --- 2001 ( 480 )*>
TITULO: Mi lista de canciones
DURACION TOTAL: 7399
NUMERO de canciones: 4
1) CM_GM_Sinf3:::<* Symphony N.3 --- Gustav Mahler --- 1896 ( 6300 )*>
2) BD_ult_cd1-7:::<* Boots Of Spanish Leather --- Bob Dylan --- 1964 ( 278 )*>
3) BD_ult_cd2-13:::<* Hurricane --- Bob Dylan y J. Levy --- 1975 ( 512 )*>
4) BD_ult_cd3-14:::<* Dark Eyes --- Bob Dylan --- 1985 ( 309 )*>
intercambio IMPOSIBLE: C1 ### CM_GM_Sinf3
```