Hoja de ejercicios del Tema 7

1. Utilizando las estructuras de datos definidas en el ejercicio 5 de la hoja de ejercicios del Tema 5 (empleados de una empresa), ordena la lista según el tipo de contratación de los empleados (Fijo, En prácticas, Eventual o Becario) utilizando los tres algoritmos de ordenación vistos en clase:

a) Ordenación por inserción

b) Ordenación por selección directa

c) Método de la burbuja

2. En un archivo de texto `inventario.txt` se guarda información acerca de los productos existentes en un almacén. Para cada producto se guardan los siguientes datos:

- ✓ Código del producto (entero positivo)
- ✓ Nombre del producto
- ✓ Número de unidades (entero)
- ✓ Precio por unidad (real)

El archivo está ordenado por orden creciente del código de producto y termina con un -1 como código.

Se dispone además de otro archivo, `modificaciones.txt`, en el que se guardan las modificaciones en el número de artículos que se han producido a lo largo de todo un día. Cada componente del archivo `modificaciones.txt` consta de los siguientes campos:

- ✓ Código del producto (entero positivo)
- ✓ Código de operación: venta (V), compra (C), o devolución (D)
- ✓ Número de unidades (entero)

El archivo `modificaciones.txt` está organizado secuencialmente y no se encuentra ordenado de ninguna manera especial. Puede haber varias modificaciones relativas al mismo producto. Es decir, el producto de código 2331 puede tener asociadas, por ejemplo, tres componentes del archivo `modificaciones.txt`: la primera puede tratarse de la devolución de 100 unidades, la segunda de la devolución de 50 unidades y la tercera de la venta de 550 unidades. Termina con un -1 como código.

Se pide:

a) Efectuar las declaraciones necesarias.

b) Implementar un subprograma que coloque la información del archivo `inventario.txt` en las estructuras de datos correspondientes.

c) Implementar un subprograma que guarde en un archivo `inventario2.txt` la información contenida en la lista del inventario.

d) Implementar un subprograma que dado un código de producto devuelva su posición en la lista del inventario. Debe realizar una búsqueda binaria.

e) Implementar un subprograma que actualice la lista del inventario procesando los cambios del archivo `modificaciones.txt` (las ventas disminuyen el número de unidades y las compras y devoluciones lo aumentan).

3. Se quiere implementar una aplicación para gestionar la información de las obras pictóricas catalogadas en el museo del Prado de Madrid. El número total de obras del museo no puede superar un cierto valor máximo N, siendo el número de obras existentes en cada momento variable. La información que se quiere almacenar para cada obra consiste en:

- ✓ Número de catálogo (entero positivo)
- ✓ Título de la obra (cadena)
- ✓ Nombre del pintor (cadena)
- ✓ Escuela del pintor (Española, Flamenca, Italiana, Francesa, Holandesa, Alemana o Inglesa).

Las obras se mantendrán ordenadas por el número de catálogo.

Se pide:

a) Redactar las declaraciones adecuadas.

b) Implementar un subprograma que cargue en la lista la información de un archivo de texto `museo.txt` sobre obras pictóricas ordenadas por número de catálogo. Para cada obra, el archivo contiene cuatro líneas, que indican, en este orden, el número de catálogo, el título, el nombre del pintor y la escuela. Termina con -1 como número de catálogo.

c) Implementar un subprograma que, dados una lista de obras y el número de catálogo de una obra, elimine de la lista la obra con dicho número de catálogo, si es que existe, y devuelva un valor que indique si la operación ha tenido éxito. Suponer que la lista de obras está ordenada en orden creciente por el número de catálogo y que no existen números de catálogo repetidos. Si la operación tiene éxito (se suprime la obra): No deben quedar "huecos" en la lista.

4. [Control de abril de 2012] Se pide construir un programa que procese el stock de una tienda. El stock es una lista de longitud variable de productos (máximo 100). Para cada producto se mantiene la siguiente información:

- ✓ Código (entero positivo)
- ✓ Nombre (una sola palabra)
- ✓ Precio (real)
- ✓ Unidades (entero positivo)

El programa comenzará leyendo la información del stock de un archivo `stock.txt`. El archivo contiene cuatro líneas por producto (código, nombre, precio y unidades). Termina con un -1 como código. Hay que comprobar que exista y si hay más del máximo de productos, se ignorarán los adicionales.

A continuación, el programa mostrará la lista de productos y calculará y mostrará el activo total de la tienda (el activo de un producto es su precio por su número de unidades; el activo total es la suma de los activos de todos los productos). Seguirá ordenando la lista por nombres de productos y finalmente volverá a mostrar la lista.

La lista se mostrará con un formato como el siguiente:

```
   123 TV_PLASMA_32"                    457.95 €   4 unidades
  4265 TV_LED_38"                       379.95 €  12 unidades
 43256 DVD_GRABADOR_250_Gb              255.00 €   8 unidades
   672 TABLET_10"_HD_WiFi_3G            428.50 €   5 unidades
```

(Anchura de 6 y ajuste a derecha para el código, anchura de 30 y ajuste a izquierda para el nombre, anchura de 8 y dos decimales para el precio, anchura de 3 y ajuste a derecha para el número de unidades.)

Se desarrollarán, al menos, los siguientes subprogramas: leer stock, activo (activo total de la tienda), ordenar, mostrar producto (línea con la información del producto proporcionado) y mostrar stock.

5. [Control de abril de 2013] Se pide construir un programa que muestre por orden de fecha los alquileres de una agencia de alquiler de coches. Se dispone de dos archivos, uno (`coches.txt`) con la información de códigos (enteros) y nombres (cadenas) de los coches de los que se dispone. La información en el archivo está ordenada por códigos y termina con -1 como código:

```
1325 Seat León
1548 Volkswagen Golf
...
-1
```

El otro archivo (`rent.txt`) contiene (sin ningún orden) la relación de alquileres que se han contratado (código del coche, fecha en formato AA/MM/DD y días que se ha alquilado). Termina con un -1 como código:

```
7377 13/03/19 1
2176 13/02/11 7
6664 13/03/17 3
3349 13/01/21 1
...
-1
```

El programa deberá empezar cargando la información de cada archivo en sendas listas: una lista de coches (máximo 20) y una lista de alquileres (máximo 100). La lista de coches quedará ordenada por orden de códigos (como en el archivo).

Una vez leídas las listas ordenará la lista de alquileres por fechas. Terminará mostrando la información sobre los alquileres (fecha, modelo y días alquilado):

```
13/01/17 Volkswagen Golf          3 día(s)
13/01/21 Opel Zafira              1 día(s)
13/01/31 Opel Vivaro              3 día(s)
...
13/02/26 Volkswagen Passat        3 día(s)
13/02/27 ERROR: ¡¡¡Modelo inexistente!!!
...
```

El programa deberá hacer uso de los siguientes subprogramas:

- ✓ `leerModelos()`: carga la información del archivo `coches.txt` en la lista de coches; devuelve `true` si se ha podido abrir el archivo y `false` en caso contrario. La lista de coches sólo contendrá la información de este archivo.
- ✓ `leerAlquileres()`: carga la información del archivo `rent.txt` en la lista de alquileres; devuelve `true` si se ha podido abrir el archivo y `false` en caso contrario. La lista de alquileres sólo contendrá la información de este archivo.
- ✓ `ordenar()`: ordena la lista de alquileres por orden de fecha (menor a mayor).
- ✓ `buscar()`: dada la lista de coches y un código, devuelve la posición (índice) del elemento de la lista de coches con ese código; si no se encuentra el código devuelve -1. Debe implementarse como búsqueda binaria.
- ✓ `mostrar()`: dadas ambas listas, muestra la relación de alquileres con el formato mostrado arriba; si no se encuentra un código de coche se notificará el error.