Facultad de Informática – Universidad Complutense
1º curso de los Grados

Fundamentos de la programación – Grupos D, G y D.G.

Curso 2016–2017

Examen de junio de 2017

Tiempo disponible: 3 horas

Se pide construir un programa modular que permita gestionar las ventas de una lonja de pescado. La lonja funciona de la siguiente forma: cuando un barco llega al puerto desembarca el pescado clasificado en cajas. Las cajas se trasladan a la lonja, donde se etiquetan, se pesan y se les da un precio de salida a subasta, pasando a formar un lote. En el momento de la subasta cada lote comienza con un precio de salida que va disminuyendo hasta que algún comprador detiene la subasta y lo compra.

El programa leerá de un fichero la información relativa a cada lote ya pesado y con precio, y la almacenará en una lista estática de componentes dinámicas. A continuación, se leerá de teclado la identificación del comprador y el precio de compra definitivos, se introducirá esta información en la lista de lotes y se creará una lista dinámica de compradores que resumirá la información de lo que va comprando cada uno.

El programa consta de 4 módulos: *lotes*, *listaLonja*, *listaCompradores* y módulo principal.

Módulo *lotes* (1.5 puntos)

Declara un tipo de estructura *tLotes* con información sobre la identificación del lote (`string`), que será único; el tipo de pescado del lote: sardina, merluza...(`string`); el peso del lote (`float`); su precio (`float`); y la identificación de quién lo compra (long long int).

Implementar como mínimo las siguientes operaciones sobre los lotes:

- Sobrecargar el operador menor para que ordene los lotes por tipo de pescado (orden alfabético) y a igualdad de tipo por la identificación del lote.
- Sobrecargar el operador de igualdad. Dos lotes son iguales cuando lo es su identificación.
- mostrarLote(). Muestra la información de un lote formateada de la siguiente forma:

```
Lote: S030514        Peso del lote: 10.87
Tipo: Sardina        Precio de salida: 8.30
```

El tamaño máximo del identificador del lote y del tipo de pescado es de 10 caracteres. El peso y el precio se mostrarán con 2 decimales. La separación entre las dos columnas es de 10 caracteres.

- mostrarPrecio(). Muestra el comprador y el precio definitivo de compra de un lote:

  ```
  Lote: S030514        Comprador: 554734
  Tipo: Sardina        Precio de compra: 3.40
  ```

- modificarLote(). Dado un lote, la identificación de un comprador y un precio de compra, modifica adecuadamente la información de dicho lote.

Módulo *listaLonja* (4 puntos) Max. 300 lotes.

Declara una lista con información sobre los lotes que hay en la lonja. Esta lista estará implementada con un array estático de punteros a variables dinámicas. Además estará ordenada por el tipo de pescado y a igualdad del tipo de pescado por su identificación de lote.

Se implementarán al menos las siguientes operaciones:

- inicializar(). Crea una lista vacía de lotes.
- numLotes(). Devuelve el número de lotes que hay almacenados en la lista.
- insertar(). Dada una lista de lotes de una lonja inserta un nuevo lote. La lista debe seguir ordenada después de la inserción. En caso de que el lote ya estuviese en la lista, la operación no tiene efecto.
- cargar(). Crea una lista de lotes a partir del fichero de entrada, que no necesariamente estará ordenado.
- buscar(). Dada la identificación de un lote y su tipo de pescado, devuelve la posición de la lista en la que se encuentra el lote, o falso si no se encuentra. La búsqueda se implementará con un algoritmo recursivo eficiente.
- obtenerLote(). Dada una posición válida dentro de la lista, devuelve el lote que se encuentra en dicha posición en la lista.
- mostrar(). Muestra la lista una vez finalizada la subasta. De cada lote se muestra su identificador, comprador, tipo de pescado y precio de compra con el formato mostrado anteriormente.
- liberar(). Libera la memoria dinámica reservada.

Módulo *listaCompradores (3 puntos)*

Declara un tipo de estructura *tComprador* con información sobre: número de comprador (`long long int`), que será único; y el total del importe de la compra de este comprador.

Declara una lista de compradores, *tListaComprador*. Esta lista se implementará con un array dinámico. Dado que el número de compradores de la lonja no es muy elevado, no se mantiene ordenada.

Se implementarán al menos las siguientes operaciones:

- inicializar(). Crea una lista vacía de compradores.
- insertar(). Dado un *tComprador*, si no está en la lista lo inserta; y en caso contrario, modifica la información de la lista incrementando el importe de la compra ese comprador.
- buscar(). Dado un número de comprador, devuelve la posición en la lista de dicho comprador o falso si no se encuentra.
- mostrar(). Muestra un listado de todos los compradores con el precio que debe pagar cada uno. El tamaño máximo del número de comprador es 10 caracteres y la separación entre columnas debe ser de 10 caracteres.
- liberar(). Libera la memoria dinámica reservada.

Módulo principal (1.5 puntos)

El módulo principal carga los datos de los lotes del fichero de entrada en la lista de la lonja y luego simula el proceso de la subasta: muestra la información del siguiente lote a subastar (la lista se recorre de menor a mayor); se introduce por teclado el número de comprador y el precio de compra; y se actualiza convenientemente la lista de la lonja y la de compradores. Una vez finalizada la subasta se muestra la lista de todos los compradores con los productos que han comprado y el precio total a pagar, y a continuación la lista de lotes actualizada.

Se valorará la legibilidad, así como el uso adecuado de los esquemas de recorrido y búsqueda, de la comunicación entre subprogramas y de la memoria.

Recuerda: El comando para que se muestre la memoria no liberada es

_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF);

Entrega del examen:

1- Añade al inicio de tus archivos un comentario con tus datos:

```
/*
Apellidos:
Nombre:
DNI:
Puesto:
*/
```

2- Abre la herramienta de entrega de exámenes por ftp que hay en el escritorio de tu ordenador.

3- Úsala para subir tus archivos (arrastra tus ficheros hacia la ventana derecha).

4- Pasa por el ordenador del profesor, pregúntale si tu archivo se ha recibido correctamente, y firma.

```
// archivo checkML.h
#ifdef _DEBUG
#define _CRTDBG_MAP_ALLOC
#include <stdlib.h>
#include <crtdbg.h>
#ifndef DBG_NEW
#define DBG_NEW new ( _NORMAL_BLOCK , __FILE__ , __LINE__ )
#define new DBG_NEW
#endif
#endif
```

Ejemplo de archivo de entrada:

```
S030514 Sardina 10.875 8.30
B892310 Bacalao 5.750 10.50
T231 Trucha 6.2 18.75
K4356 Sardina 5.00 5.50
GA65455 Pescadilla 15.35 9.30
```

Ejemplo de ejecución

```
Lote:    B892310          Peso del lote: 5.75
Tipo:    Bacalao          Precio de salida: 10.50
Introduzca comprador y precio
1234 8.32
Lote:    GA65455          Peso del lote: 15.35
Tipo: Pescadilla          Precio de salida: 9.30
Introduzca comprador y precio
56789 5.60
Lote:      K4356          Peso del lote: 5.00
Tipo:    Sardina          Precio de salida: 5.50
Introduzca comprador y precio
1234 3.10
Lote:    S030514          Peso del lote: 10.88
Tipo:    Sardina          Precio de salida: 8.30
Introduzca comprador y precio
99 4.68
Lote:       T231          Peso del lote: 6.20
Tipo:     Trucha          Precio de salida: 18.75
Introduzca comprador y precio
99 12.0
Comprador:       1234          Total: 11.42
Comprador:      56789          Total: 5.60
Comprador:         99          Total: 16.68
Lote:    B892310          Comprador: 1234
Tipo:    Bacalao          Precio de compra: 8.32
Lote:    GA65455          Comprador: 56789
Tipo: Pescadilla          Precio de compra: 5.60
Lote:      K4356          Comprador: 1234
Tipo:    Sardina          Precio de compra: 3.10
Lote:    S030514          Comprador: 99
Tipo:    Sardina          Precio de compra: 4.68
Lote:       T231          Comprador: 99
Tipo:     Trucha          Precio de compra: 12.00
Presione una tecla para continuar . . .
```