Tecnologías de la Información

Universidad de Huelva

PRÁCTICA 2
TADs Lineales Dinámicos

Enunciado

Con esta práctica se pretende afianzar los conceptos relativos al uso de memoria dinámica y de TADs lineales, apoyados en memoria dinámica. Se le irá facilitando distintas versiones del enunciado, conforme los avances en teoría y en el propio desarrollo de la práctica le permita ir avanzando.

Se plantea el desarrollo de una ampliación de la aplicación realizada en la primera práctica, que permita mejorar la gestión de un almacén, en concreto de los pedidos realizados por las tiendas a ese almacén y de los envíos realizados de estos pedidos a las tiendas. La gestión de los pedidos se realizará añadiendo al almacén una cola de pedidos hechos por las tiendas y una lista de envíos pendientes de realizar.

Para ello se modificará el submenú que gestiona los productos de un almacén, añadiendo dos nuevas opciones, Gestión de Pedidos y Gestión de Envíos, que tendrán su correspondiente submenú con las opciones para gestionar la cola de pedidos y la lista de envíos respectivamente, que a continuación se muestran:

```
----- Menú Almacenes -----

1.-  Crear un almacén vacío.
2.-  Abrir un fichero de almacén.
3.-  Cerrar un almacén.
4.-  Listar productos del almacén.
5.-  Añadir un producto.
6.-  Actualizar un producto.
7.-  Consultar un producto.
8.-  Eliminar un producto.
9.-  Gestión de pedidos.
10.- Gestión de envíos.
0.- Salir.
```

Todas las operaciones de este menú deben trabajar directamente con un fichero binario que contiene los productos del almacén. La gestión de los almacenes puede crear y abrir fichero de productos así como añadir, borrar, consultar, y actualizar los productos del fichero en esté abierto.
En todo momento las distintas opciones para la gestión de los almacenes deben controlar que el almacén está creado y/o abierto, si los productos solicitados están o no en el almacén, etc. Además debe mostrar en todo momento cualquier error que impida la realización de cada opción del menú en caso de terminar satisfactoriamente mostrará un mensaje que lo indique.

La gestión de pedidos se basa en las opciones del siguiente submenú, que trabajará utilizando un TAD cola para almacenar los pedidos:

```
----- Gestión de Pedidos -----

1.- Cargar pedidos de fichero.
2.- Añadir pedido.
3.- Atender pedidos.
4.- Listar pedidos completos de todos los productos.
5.- Listar pedidos de un producto.
6.- Listar todas las cantidades pendientes.
7.- Listar cantidades pendientes de un producto.
8.- Guardar pedidos a fichero.
0.- Salir.
```

La gestión de envíos se basa en las opciones del siguiente submenú, que trabajará utilizando un TAD lista para los envíos.

```
----- Gestión de Envíos -----

1.- Cargar envíos de fichero.
2.- Insertar un nuevo envío.
3.- Reparto de envíos a tienda.
4.- Listar todos los envíos.
5.- Listar los envíos a una tienda.
6.- Guardar envíos a fichero.
0.- Salir.
```

Las opciones para la **gestión de los pedidos** del almacén tienen la siguiente funcionalidad:

1.- Cargar pedidos de fichero. Carga la cola de pedidos desde fichero. El fichero tiene una sucesión de elementos de tipo TPedido.

2.- Añadir pedido. Esta opción Añade a la cola de pedidos un nuevo pedido (similar a como debemos ahora hacer cuando no se pueda suministrar un producto a una tienda por no tener suficiente producto en almacén).

3.- Atender pedidos. Con esta opción del menú la idea es que el almacén "compra" productos y atiende con ellos a los pedidos. A esta opción por "abreviar" le hemos llamado atender pedidos.
Atiende los pedidos del producto en cuestión, pendiente de suministrar, incorporándolos a la lista de envíos y eliminando de la cola de pedidos los pedidos atendidos. En esencia es el mecanismo que emplea el almacén en sus compras ordinarias. Cuando compra un producto, con una cantidad X, se atienden los pedidos pendientes que puedan atenderse con dicha cantidad X (si los hubiera). Si algún pedido es atendido parcialmente, por que se acabe el producto comprado, la cola se modificará modificando y dejando pendiente la cantidad correspondiente que no pudo servirse. Se van introduciendo en la lista de envíos los pedidos que puedan atenderse con la cantidad suministrada. Si el producto comprado excede de la cantidad pendiente de servir en la cola de pedidos, o simplemente no hubiera pedidos para ese producto, la cantidad sobrante entra en el almacén (debe actualizarse la cantidad de dicho producto en almacén). Si el producto no existe en el almacén debe emitirse error al usuario.

4.- Listar los pedidos de todos los productos. Muestra el contenido completo de la cola de pedidos (con toda la información de los productos obtenida del almacén).

5.- Listar pedidos de un producto. Muestra los pedidos concretos de un determinado producto con toda la información.

6.- Listar todas las cantidades pendientes. Muestra las cantidades pendientes de suministrar de todos los productos, almacenados en la cola.

7.- Listar cantidades pendientes de un producto. Muestra la cantidad pendiente de suministrar de los pedidos almacenados en la cola del producto en cuestión.

8.- Guardar pedidos. Salva en fichero la cola de pedidos.

Las opciones para la **gestión de los envíos** del almacén tienen la siguiente funcionalidad:

1.- Cargar envíos de fichero. Almacena la lista de envíos desde fichero. El fichero tiene una sucesión de elementos de tipo TPedido.

2.- Insertar un nuevo envío. Añade de forma ordenada, por el nombre de la tienda, a la lista de envíos, los pedidos a enviar.

3.- Reparto de envío a tienda. Se encarga de sacar de la lista de envíos aquellos que tienen por destino una determinada tienda, mostrando dichos envíos por pantalla.

4.- Listar todos los envíos. Muestra por pantalla la lista de envíos pendientes de enviar.

5.- Listar los envíos a una tienda. Muestra la lista de envíos pendientes a una determinada tienda. Si la tienda no tuviera envíos pendientes se debe mostrar que no tiene envíos.

6.- Guardar envíos. Salva en fichero la lista de envíos.

La gestión de los productos y estanterías de la tienda seguirá disponiendo de las mismas opciones, aunque algunas de ellas se tendrán que modificar al generar pedidos que habrá que gestionar en la cola de pedidos del almacén, en concreto añadir un estante y actualizar un estante como se mostrará a continuación:

```
----- Menú Tienda -----

1.- Crear una tienda vacía.
2.- Abrir un fichero tienda.
3.- Cerrar la tienda.
4.- Actualizar el fichero tienda.
5.- Listar productos de la tienda.
6.- Añadir un estante.
7.- Actualizar un estante.
8.- Consultar un estante.
9.- Eliminar un estante.
0.- Salir.
```

Todas las operaciones de este menú deben trabajar directamente en memoria y solo se utilizará el fichero binario para cargar y guardar los datos de la tienda. La gestión de la tienda puede crear y abrir fichero de estantes, así como añadir, borrar, consultar, y actualizar estantes pero también pueden consultar y actualizar los productos del fichero almacén que haya sido abierto en la opción 1 del menú principal.

En todo momento las distintas opciones para la gestión de la tienda deben controlar que la información de la tienda esté cargada en memoria y que en caso necesario el fichero de almacén está también abierto. Además, debe mostrar en todo momento cualquier error que impida la realización de cada opción del menú en caso de terminar satisfactoriamente mostrará un mensaje que lo indique.

Las distintas opciones del menú tienen la siguiente funcionalidad:

1.- Crear una tienda vacía =>Solicita los datos de la tienda (nombre, dirección y nombre de fichero) y crea un fichero físico vacío y lo cerrado para posteriormente guardar en él los datos de la tienda.
2.- Abrir un fichero tienda => Solicita un fichero de tienda y lo carga en memoria para su utilización.
3.- Cerrar una tienda => Actualiza el fichero de tienda con los datos de la memoria y elimina toda la información de la tienda.
4.- Actualizar el fichero tienda =>Actualiza el fichero de tienda con los datos de la memoria.
5.- Listar productos de la tienda => Muestra por pantalla la información de cada estante, incluido el nombre del producto, precio, cantidad y valor total de la cantidad de productos que hay en el estante.
6.- Añadir un estante => Pedirá todos los datos de un estante y verificará que éste no existe previamente. Además, verificará que el producto que va en el estante existe en el almacén y retirará del fichero almacén la cantidad de producto que va a ser colocada en el estante. Si la cantidad indicada de producto excede a la que hay en el almacén, solo se pondrá en el estante la cantidad que hay en el almacén **y se generará un pedido por la cantidad restante**.
7.- Actualizar un estante => Debe solicitar el estante a actualizar. Si existe, mostrará por pantalla el número de productos que hay en el estante y preguntará si quiere actualizarlo. En caso afirmativo, solicitará la nueva cantidad de producto para el estante. Si la cantidad es menor a la que había, se ha de devolver la diferencia al almacén y si es mayor habrá que reponer esa cantidad desde el almacén. En el caso que la cantidad a reponer sea superior a la que hay en el almacén se repondrá solo la cantidad existente en el almacén, **y se generará un pedido por la cantidad restante**. En cada caso se indicará por pantalla que cantidad se ha repuesto en el estante, devuelto al almacén **o pedida al almacén**.
8.- Consultar un estante => Solicitará el estante a consultar. Si existe, lo mostrará por pantalla.
9.- Eliminar un estante => Debe solicitar el estante a eliminar. Una vez localizado se procederá a eliminarlo de la memoria.

La opción de reposición de productos en tienda no se ve modificada con esta ampliación de la práctica. Como en toda tienda, una vez terminada la jornada laboral, se repone en los estantes la máxima cantidad de producto que cabe en el estante siempre y cuando hay suficiente producto en el almacén. Para ellos, mostrará por cada estante de la tienda, el código del estante, la capacidad, el nombre del producto, el estado (no repuesto, repuesto parcialmente o repuesto completamente) y el porcentaje de ocupación del estante.

Nota. En todas las opciones del menú que se muestren datos habrá que indicar qué son, es decir, si se ha de mostrar el nombre y la cantidad de un producto aparecerá en pantalla, por ejemplo:
Producto: Naranjas, Cantidad: 10.
En el caso de listados habrá que poner una cabecera con los datos que se va a mostrar y en las siguientes líneas los valores de dichos datos.
En el caso de listados de productos del almacén o de la tienda además de lo mencionado, habrá que poner una cabecera donde se indique el nombre y la dirección del almacén y/o de la tienda.

Clases, tipos de datos y formatos de ficheros

Fichero cabecera TTipos.h

```
#ifndef TTIPOS_H
#define TTIPOS_H
#include <string.h>
#include <iostream>
#include <fstream>


typedef char Cadena[90];
typedef char Linea[500];
#define Incremento 4

struct TFecha //Almacena una fecha
{
    int Dia;
    int Mes;
    int Anyo;
};

struct TProducto
{
    Cadena CodProd;          //Código de producto.
    int Cantidad;            //Cantidad en el almacén.
    Cadena NombreProd;       //Nombre del producto.
    float Precio;            //Precio por unidad.
    Cadena Descripcion;      //Descripción opcional del producto.
    TFecha Caducicidad;      //Caducidad del producto.
};

struct TEstante
{
    int CodEstante;          //Código del estante.
    int Posicion;            //Posición del estante, valores (1-centrado, 2- arriba, 3- abajo).
    int Capacidad;           //Máxima capacidad que admite el estante.
    Cadena CodProd;          //Código del producto que contiene el estante.
    int NoProductos;         //Número de productos que hay en el estante.
};

struct TPedido
{
    Cadena CodProd;          //Nombre del producto pedido
    int CantidadPed;         //Cantidad pedida del producto
    Cadena Nomtienda;        //Nombre del fichero tienda
};
#endif // TTIPOS_H
```

Fichero cabecera TAlmacen.h

```
#ifndef TALMACEN_H
#define TALMACEN_H
#include "TTipos.h"
#include "TADcola.h"      //Fichero de definición de la cola de pedidos
#include "TADlista.h"     //Fichero de definición de la lista de envios
using namespace std;


class TAlmacen
{
    Cadena Nombre;           //Nombre del almacén.
    Cadena Direccion;        //Dirección del almacén.
    fstream FicheProductos;  //Fichero que almacena los productos del almacén.
    int NProduc;             //Número de productos que hay en el almacén. Si el almacén está cerrado
                             //deberá tener el valor -1.
    cola Pedidos;            //Estructura de tipo cola que almacena los pedidos
    lista Envios;            //Estructura de tipo lista que almacena los pedidos a enviar
```

```
public:
    TAlmacen();       //Constructor que debe inicializar los atributos de la clase.
    ~TAlmacen();      //Destructor que cerrará el almacén en caso de que el usuario no lo haya hecho.

    //Devuelve los atributos nombre y dirección por parámetro.
    void DatosAlmacen(Cadena pNombAlmacen, Cadena pDirAlmacen);

    //Crea un fichero binario vacío con el nombre pasado por parámetro. Crea la cabecera del fichero
    //y lo deja abierto para su utilización. Devuelve true si se ha creado.
    bool CrearAlmacen(Cadena pNomFiche);


    //Igual que el método anterior, pero actualizando los atributos nombre y dirección con los valores
    //pasados por parámetro. Devuelve true si ha podido crear el fichero.
    bool CrearAlmacen(Cadena pNombAlmacen, Cadena pDirAlmacen, Cadena pNomFiche);

    //Abre un fichero y actualiza los atributos de la clase con los datos de cabecera del fichero y lo
    //lo deja abierto. No se puede abrir un fichero si previamente el almacén está abierto. Devuelve true
    //si ha podido abrir el fichero.
    bool AbrirAlmacen(Cadena pNomFiche);

    //Cierra el fichero e inicializa los atributos a valores iniciales. Devuelve true si se ha cerrado el
    //almacén.
    bool CerrarAlmacen();

    bool EstaAbierto();      //Devuelve true si el fichero está abierto.
    int  NProductos();       //Devuelve el número de productos.

    //Dado un código de producto, devuelve la posición dentro del fichero donde se encuentra. Si no
    //lo encuentra devuelve -1.
    int BuscarProducto(Cadena pCodProd);

    //Dado la posición devuelve el producto del fichero situado en dicha posición. Debe verificar
    //previamente que la posición sea correcta. Si la posición no es correcta devolverá un producto cuyo
    //código tendrá el valor "NULO".
    TProducto ObtenerProducto(int pPos);

    //Dado un producto, lo busca en el fichero y si no lo encuentra lo añade al final del fichero.
    //Devuelve true si se ha añadido el producto.
    bool AnadirProducto(TProducto pProduc);

    //Dada la posición de un producto en el fichero lo actualiza con la información del producto pasado
    //por parámetro. Devuelve true si se ha actualizado el producto. Se debe verificar previamente que la
    //posición sea correcta.
    bool ActualizarProducto(int pPos, TProducto pProduc);

    //Dado la posición de un producto en el fichero lo borra. Devuelve true si se ha podido borrar. Se
    //debe verificar que la posición sea correcta.
    bool EliminarProducto(int pPos);

    //Método que carga la lista de envíos a partir del nombre del fichero que se le pasa por parámetro.
    //El fichero tiene una sucesión de elementos de tipo TPedido.
    bool CargarListaEnvios(Cadena Nomf);

    //Método que carga la cola de pedidos a partir del nombre del fichero que se le pasa por parámetro.
    //El fichero tiene una sucesión de elementos de tipo TPedido.
    bool CargarColaPedidos(Cadena Nomf);

    //Añadirá un nuevo pedido a la cola de pedidos.
    void AñadirPedido (TPedido p);

    //Método que atiende los pedidos del producto en cuestión pendientes de suministrar con la cantidad
    //comprada por el almacén, los incorpora a la lista de Envíos, eliminando de la cola de pedidos los
    //pedidos atendidos.
    //Si algún pedido es atendido parcialmente por que se acabe el producto, la cola se modificará
    //modificando y dejando pendiente la cantidad correspondiente, introduciendo en la lista de Envios
    //la cantidad que se puede suministrar.
    //Si el producto comprado excede de la cantidad pendiente de servir en los pedidos, la cantidad
    //sobrante, entra en el Almacén.
    bool AtenderPedidos(Cadena CodProd,int cantidadcomprada);

    //Muestra el contenido completo, con todos los datos de los productos leídos del almacén, de la cola
    //si CodProd es '' o muestra los pedidos del Codprod pasado con todos sus datos del almacén.
    void ListarPedidosCompleto(Cadena CodProd);
```

```
    //Muestra la cantidad pendiente de cada tipo de producto si CodProd es '' o del producto concreto
    //que se pase por parámetro.
    void ListarCantidadesPendientes(Cadena CodProd);

    //Se encarga de meter en la lista de envíos, de forma ordenada, por nombre del fichero de tienda, el
    //pedido a enviar.
    bool InsertarEnvios(TPedido p);

    //Se encarga de sacar de la lista los envíos que tienen por destino la tienda que se le pasa por
    //parámetro mostrando por pantalla los envíos que van en el camión.
    bool SalidaCamionTienda(Cadena NomTienda);

    //Si recibe Nomtienda a '' muestra por pantalla todo el contenido de la lista de envíos.
    //Si se le pasa el nombre de una tienda muestra por pantalla los envíos a dicha tienda.
    void ListarListaEnvios(Cadena Nomtienda);

    //Método que vuelca en el fichero nomf la cola de pedidos.
    bool SalvarColaPedidos(Cadena Nomf);

    //Método que vuelca en el fichero nomf la lista de envíos.
    bool SalvarListaEnvios(Cadena Nomf);

};

#endif // TALMACEN_H
```

Fichero cabecera TTienda.h

```
#ifndef TTIENDA_H
#define TTIENDA_H

#include "TTipos.h"
using namespace std;


class TTienda
{
    Cadena Nombre;            //Nombre de la tienda.
    Cadena Direccion;         //Dirección de la tienda.

    Cadena NomFiche;          //Nombre del fichero que contiene los datos de la tienda.
    TEstante *Estantes;       //Vector dinámico de estantes.
    int NEstan;               //Número de estantes ocupados en el vector dinámico.
    int Tamano;               //Tamaño de estantes del vector.

    //Método privado para ordenar los estantes del vector. La ordenación se realizará por el código de
    //producto en orden ascendente. En caso de tener el mismo código se ordenadará por la posición del
    //producto en el estante en orden ascendente.
    void OrdenarProductos();

public:
    TTienda();        //Constructor que debe inicializar los atributos de la clase.
    ~TTienda();       //Destructor que cerrará la tienda en caso de que el usuario no lo haya hecho.

    //Devuelve los atributos nombre y dirección por parámetro.
    void DatosTienda(Cadena pNombTienda, Cadena pDirTienda);

    //Devuelve el atributo del nombre del fichero de la tienda
    void NombreFicheroTienda(Cadena NomF);


    //Crea un fichero binario vacío con el nombre pasado por parámetro e inicializa los atributos nombre y
    //dirección mediante los parámetros y a continuación lo cerrará. Devolverá true si ha podido crear el
    //fichero.
    bool CrearTienda(Cadena pNombTienda, Cadena pDirTienda, Cadena pNomFiche);

    //Abre un fichero y lo carga a memoria. Si ya había un fichero previamente cargado, guardará los datos
    //de la tienda y procederá a cargar el nuevo fichero. Si el fichero es el mismo que el que está en
    //memoria, eliminará los datos y procederá a cargar nuevamente los datos del fichero. Devolverá true
    //si se ha podido cargar el fichero.
    bool AbrirTienda(Cadena pNomFiche);

    //Vuelca los datos de la memoria al fichero. Devolverá true si se han guardado los datos.
    bool GuardarTienda();
```

```
     //Guarda los datos de la memoria en el fichero y borra todos los atributos del objeto. Devuelve true
    //si se ha podido guardar los datos.
    bool CerrarTienda();

    bool EstaAbierta();       //Devuelve true si la tienda está abierta.
    int NoEstantes();         //Devuelve el número de estantes de la tienda.

    //Dado un código de estante, devuelve la posición dentro del vector donde se encuentra. Si no lo
    //encuentra devuelve -1.
    int BuscarEstante(int pCodEstante);

    //Dado la posición el estante que está en la posición indicada por parámetro.
    TEstante ObtenerEstante(int pPos);

    //Añade un estante nuevo al vector siempre que el estante no esté previamente almacenado en memoria.
    //El vector de estantes debe siempre estar ordenado. Devolverá true si se ha añadido el estante.
    bool AnadirEstante(TEstante pEstante);

    //Dado la posición de un estante lo borra desplazando el resto de estantes una posición hacia abajo.
    //Se debe verificar previamente que la posición sea correcta. Devuelve true si se ha eliminado el
    //estante.
    bool EliminarEstante(int pPos);

    //Dada la posición de un estante, lo actualiza con los datos pasados por parámetros. Se debe verificar
    //previamente que la posición sea correcta Devuelve true si se actualiza el estante.
    bool ActualizarEstante(int pPos, TEstante pEstante);

    //Dada la posición de un estante y un producto del almacén, actualizará el número de productos del
    //estante a su máxima capacidad si hay suficientes unidades en el producto, en caso contrario se
    //añadirán al estante la totalidad de unidades que estén en el producto del almacén. El mismo número
    //de unidades añadidas en el estante deben reducirse del producto. Se debe verificar previamente que
    //la posición sea correcta. El método devuelve:
    //  0 si la posición es incorrecta.
    //  1 si se ha repuesto unidades hasta llegar a la capacidad máxima del estante.
    //  2 si no se ha completado el estante al completo.
    int ReponerEstante(int pPos, TProducto &pProduc);

};

#endif // TTIENDA_H
```

Habrá que añadir al proyecto los ficheros de definición e implementación del TAD cola y del TAD lista e incluir las cabeceras allí donde se necesite utilizar los tipos cola y lista y sus operaciones.

Ficheros de la aplicación

El fichero de almacén tiene la siguiente estructura:

Cabecera del Fichero			Productos del Almacén			
Número de Productos	Nombre Almacén	Dirección Almacén	Producto 1	Producto 2	…	Producto N

Dispone una cabecera formada por el número de productos almacenados, el nombre y la dirección del almacén. A continuación de la cabecera estarán el conjunto de productos almacenados en el almacén.

Los ficheros de pedidos y envíos contienen un número indeterminado de pedidos con la siguiente estructura:

Pedido 1	**Pedido 2**	**Pedido 3**	**…**	**Pedido M**

A continuación, se muestra parcialmente el contenido de los ficheros de datos Almacen.dat y Tienda.dat.

Almacen.dat

```
Listado del almacén "Productos el Paso" localizado en Avenida Tres Cuartos SN
************************************************************
CODIGO   NOMBRE                                  PRECIO      CANTIDAD    FECHA CADUCIDAD
CAPc10   Aceite de oliva                         8.21429     61          12/12/2018
CAPc99   Aceite de oliva virgen extra            11.7        118         29/12/2018
CAPc40   Aceite de coco                          11.8        124  4/10/2018
CMPo91   Mostaza                                8.76923     81          10/2/2020
CMPo50   Mostaza en grano                       13.0909     64          8/6/2021
CCPu56   Curry amarillo                         18.5        131  29/6/2019
CCPu84   Curry rojo en pasta                    6.88889     103  23/12/2021
CGPu29   Guacamole                              19.1429     73          6/8/2018
CTPo88   Tomate frito                           7.57143     62          16/11/2020
CTPo94   Tomate natural                         8.69231     90          2/8/2018
CTPo40   Tomate rallado                         11.5455     85          9/10/2021
```

Tienda.dat

```
CODIGO   POSICION      CAPACIDAD      PRODUCTO       NoPRODUCTOS
1271     2             16             CAPc10         14
1301     3             12             CAPc21         11
1185     3             18             CAPc40         16
1549     3             18             CAPc70         12
1668     3             17             CAPc99         9
1334     2             11             CAPl12         11
1436     1             10             CAPl68         5
```

Para la elaboración de la práctica el alumno deberá utilizar dichos ficheros como datos iniciales, siendo obligatorio el uso de diseño modular.

Fecha de finalización

La práctica deberá estar finalizada antes de la realización de la segunda prueba de modificación de prácticas.