Tecnologías de la Información

Universidad de Huelva

PRÁCTICA 1

Ficheros y Tablas Dinámicas

Enunciado

Se plantea el desarrollo de una aplicación para la gestión de un almacén y una tienda de productos. La gestión de la información relativa al almacén y a la tienda se realizará mediante ficheros y memoria dinámica. Para ello se pretende diseñar el siguiente menú principal:

```
----- Menú Principal ----- Nombre Almacén, Nombre Tienda

1.- Gestión del Almacenes.
2.- Gestión de la Tienda.
3.- Reposición de Productos en Tienda.
0- Salir
```

En el título del menú principal, debe aparecer el nombre del almacén y/o el nombre de la tienda cuyos datos se están gestionando.

Opción 1. Permitirá realizar la gestión de productos de un almacén. Esta gestión se basa en las opciones del siguiente submenú:

```
----- Menú Almacenes ----- Nombre Almacén

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.
0.- Salir.
```

HECHO

En el título del menú Almacenes, debe aparece el nombre del almacén que se está gestionando.

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 que 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 y en caso de terminar satisfactoriamente mostrará un mensaje que lo indique.

Las distintas opciones del menú tienen la siguiente funcionalidad:

1.- Crear un almacén vacío =>Solicita los datos del almacén (nombre, dirección y nombre de fichero) y crea un fichero físico vacío y deja el almacén abierto para su utilización.
2.- Abrir un fichero almacén => Solicita un fichero de almacén creado y lo abre para su utilización.
3.- Cerrar un almacén => Cierra el fichero almacén para que no pueda ser modificado.
4.- Listar productos del almacén => Muestra todos los datos de cada producto almacenado en el fichero almacén. Cada producto debe ser mostrado en una sola línea.
5.- Añadir un producto => Solicita los datos de un producto y la añade al fichero.
6.- Actualizar un producto =>Debe solicitar el producto a actualizar. Una vez localizado, por cada dato del producto, debe mostrarlo y preguntar si se desea modificar, si se responde afirmativamente se solicita un nuevo valor. Una vez terminado la entrada de datos se actualiza el producto con los nuevos datos en el fichero.
7.- Consultar un producto => Debe solicitar el producto a mostrar. Una vez localizado debe mostrar todos sus datos por pantalla.
8.- Eliminar un producto => Debe solicitar el producto a eliminar. Una vez localizado se procederá a eliminarlo del fichero.

Opción 2. Del mismo modo que la opción anterior, esta opción se encarga de la gestión de los productos y estanterías de la tienda. Esta gestión se basa en las opciones del siguiente submenú:

```
----- Menú Tienda ----- Nombre 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.
```

En el título del menú Tienda, debe aparece el nombre de la tienda que se está gestionando.

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 tiendas, 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 y 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), crea un fichero físico vacío y lo cierra. Posteriormente se guardarán 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 la memoria dinámica asociada.
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.
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. En cada caso se indicará por pantalla que cantidad se ha repuesto en el estante o devuelto 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. Los productos del estante deben ser devueltos al almacén.

Opción 3. 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 haya suficiente producto en el almacén. Para ello, 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.

Notas.

- En todas las opciones del menú que se muestren datos habrá que indicar qué son, por ejemplo, si se ha de mostrar el nombre y la cantidad de un producto aparecerá en pantalla:

```
Producto: Naranjas, Cantidad: 10
```

- En el caso de listados habrá que poner una cabecera con los datos que se van a mostrar y en las siguientes líneas los valores de dichos datos. Por ejemplo:

```
CODIGO  POSICION  CAPACIDAD   PRODUCTO    NoPRODUCTOS
1271    2         16          CAPc10      14
```

- 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.

```
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
```

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];
#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 Caducidad;         //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.
};

#endif // TTIPOS_H
```

Fichero cabecera TAlmacen.h

```
#ifndef TALMACEN_H
#define TALMACEN_H
#include "TTipos.h"
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.

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);

};

#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 ordenará 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);

    //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);

    //Devuelve el estante cuya posición es 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
```

Ficheros de la aplicación

Para esta prácticas, se proporciona dos ficheros de datos de ejemplo para que la codificación y la comprobación del código sea más rápida.

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 del almacén.

Un ejemplo del contenido del fichero Almacen.dat es:

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
```

El fichero de tienda contiene un número indeterminado de estantes con la siguiente estructura:

Cabecera del Fichero		Productos de la Tienda				
Nombre	Dirección	Estante 1	Estante 2	Estante 3	⋯	Estante M

Dispone una cabecera formada por el nombre y la dirección de la tienda. A continuación de la cabecera estarán los estantes de la tienda.

Un ejemplo del contenido del fichero Tienda.dat es:

Tienda.dat

```
Listado de la tienda "MercaDia" localizado en C/Madrid_SN
************************************************************
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             CAP112         11
1436    1             10             CAP168         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 primera prueba de modificación de prácticas.