Tecnologías de la Información

Universidad de Huelva

PRÁCTICA 1

Ficheros y Punteros

Enunciado

Se plantea el desarrollo de una aplicación para la administración de una estación de autobuses con varios terminales. Solamente se gestionará el control de itinerarios y autobuses asignados, utilizando una serie de ficheros de apoyo. Para ello se pretende diseñar el siguiente menú principal:

```
----- Menú Principal -----

1- Gestión de Autobuses
2- Gestión de Itinerarios
3- Listar itinerarios según franja horaria
4- Salir
```

Opción 1. Permitirá realizar las operaciones que se muestran en el siguiente submenú propuesto:

```
----- Menú Gestión de Autobuses -----

a) Insertar
b) Borrar
c) Modificar
d) Buscar
e) Listar
f) Cargar autobuses desde fichero
g) Guardar autobuses en fichero
v) Volver
```

La idea es poder volcar el contenido del fichero de autobuses a una tabla dinámica en memoria y poder trabajar con ella, volcando finalmente la información en fichero. No obstante, el sistema debe diseñarse para que si no se ha cargado previamente el fichero en memoria podamos trabajar directamente contra el fichero físico.

Así, si no se ha cargado el fichero a través de la opción f, las opciones de la a) a la e) se realizan directamente a fichero. Sin embargo, si se carga el fichero en memoria, dichas opciones trabajan con la tabla dinámica donde se ha cargado, el cual se guarda en fichero al pulsar la opción g). El formato del fichero utilizado es binario.

A) Insertar=> solicita los datos del autobús y lo inserta en la tabla dinámica o fichero, al final. No se permite una matrícula repetida
B) Borrar=> Solicita la matrícula o posición del autobús y lo elimina de la tabla dinámica o fichero. Es decir, se puede actuar por matrícula o posición. Si no existe se muestra mensaje de error
C) Modificar=>Solicita matrícula o posición y si existe el autobús, solicita los nuevos datos para modificar. Si no existe muestra mensaje de error.
D) Buscar => solicita matrícula o posición y si existe el autobús, muestra sus datos por pantalla. Si no existe muestra mensaje de error.
E) Listar=> Lista los autobuses por pantalla.
F) Cargar=> Permite cargar en memoria los datos del fichero de autobuses
G) Guardar=> Vuelca el contenido de la tabla dinámica (si ésta existe) a fichero. En dicho volcado los datos de los autobuses se graban **ordenados** por número de matrícula (puede utilizar el algoritmo de ordenación que desee)

Opción 2. Del mismo modo que la opción anterior, se planteará el siguiente submenú:

```
----- Menú Gestión de Itinerarios -----

a) Insertar
b) Borrar
c) Modificar
d) Listar
e) Buscar
f) Cargar itinerarios desde fichero
g) Guardar itinerarios en fichero
v) Volver
```

En este caso, sólo se trabaja con el fichero una vez se ha volcado en memoria. Eso quiere decir que no se pueden usar las opciones comprendidas entre la a) y la e) si previamente no se ha usado la opción f). El formato del fichero utilizado es binario.

Un itinerario, como se mostrará más adelante, viene identificado por su nombre y por el autobús que lo realiza (A modo de ejemplo el itinerario "Huelva Costa", puede ser desarrollado por varios autobuses)

A) Insertar=> solicita los datos del itinerario y lo inserta en la tabla dinámica, al final. No se permite una "pareja" de nombre de itinerario y matrícula repetida. El autobús se entiende que existe (no debe comprobarlo).
B) Borrar=> Solicita el nombre del itinerario y matrícula o posición del itinerario y lo elimina de la tabla dinámica. Si no existe se muestra mensaje de error.
C) Modificar=>Solicita nombre itinerario y matrícula o posición del itinerario y si existe, solicita los nuevos datos para modificar. Si no existe muestra mensaje de error.
D) Buscar => solicita nombre itinerario y matrícula o posición del itinerario y si existe, muestra sus datos por pantalla. Si no existe muestra mensaje de error.
E) Listar=> Lista los itinerarios por pantalla.
F) Cargar=> Permite cargar en memoria los datos de los itinerarios desde el fichero de itinerarios
G) Guardar=> Vuelca el contenido de la tabla dinámica (si ésta existe) a fichero (tal como está en la tabla dinámica)

Opción 3. Se debe facilitar el listado de itinerarios comprendidos entre una franja horaria solicitada según la siguiente propuesta:

```
Hora inicio: h ¿?  m ¿?
Hora fin: h ¿?  m ¿?
```

Hay que tener cuidado con el formato horario, puesto que se utiliza el de 12H, y tenemos que diferenciar los tramos am y pm.

Opción 4. Antes de dejar la aplicación, debemos volcar toda la información que se disponga en memoria hacia los ficheros de autobuses e itinerarios (siempre que se hayan cargado en memoria).

Clases, tipos de datos y formatos de ficheros

```
typedef char cadena[50];
#define SALTO 4 //valor de constante usado para adaptar el tamaño de las tablas
                // dinámicas, si éstas necesitan "crecer"

struct estado {
    bool activo;     //en servicio si/no
    float deposito;  //litros del depósito
    int plazas;   //capacidad
    bool itv;        //itv pasada si/no
};

struct autobus
{
    cadena matricula;    //formato NNNNLLL
    estado e;
    int conductor;      //identificador conductor
};

class autobuses
{
 autobus *elementos;
 fstream fichero;
 int tam; //tamaño de la tabla dinámica
 int n; //número de autobuses
 bool cargado; //indica si el fichero está cargado en memoria

 public:
   autobuses(); //constructor de la clase. Inicializa las variables de la clase. En
                //particular destacar que se "enlaza" ya con el fichero físico de
                //"autobuses.dat" para permitir poder trabajar con él mientras no se
                //cargue en memoria
   void Insertar(autobus a);
   void Borrar (cadena mat,int pos); //borra el autobús (se le pasa matrícula o
        //posición)
   void Modificar(autobus a, cadena mat,int pos); //modifica el autobús cuya
        // matrícula sea mat o esté en la posición pos, con los datos del autobus a)
   int Buscar(cadena mat); //devuelve la posición del autobús mat. – 1 si no lo
        // encuentra
   void Mostrar(cadena mat,int pos); //muestra por pantalla el autobús (se le pasa
        // matrícula o posición)
   void Listar();
   bool Cargar(); //permite cargar en memoria el fichero de autobuses denominado
        // "autobuses.dat". Devuelve true si correcto, false si error.
   bool Guardar();//Vuelca el contenido de la tabla dinámica (si ésta existe) al
        // fichero "autobuses.dat"
};
```

```
struct Hora {
    int h;
    int m;
    bool am;
 };

struct itinerario
{
    cadena nombre;
    cadena origen;
    cadena destino;
    Hora inicio;
    Hora fin;
    cadena matricula;
};

class itinerarios
{
 itinerario *elementos;
 fstream fichero;
 int tam; //tamaño de la tabla dinámica
 int n; //número de autobuses
 bool cargado; //indica si el fichero está cargado en memoria

 public:
   itinerarios(); //constructor de la clase;
   void Insertar(itinerario i);
   void Borrar (cadena nombrei,cadena mat,int pos); //borra el itinerario (se le
        // pasa nombre itinerario y matricula o bien posición)
   void Modificar(itinerario a, cadena nombrei, cadena mat,int pos); //modifica el
        // itinerario cuyo nombre de itinerario sea nombrei y matrícula sea mat o
        // bien esté en la posición pos, con los datos del itinerario i)
   int Buscar(cadena nombrei,cadena mat); //devuelve la posición del itinerario de
        //nombre nombrei y matricula mat. – 1 si no lo encuentra
   void Mostrar(cadena nombrei, cadena mat,int pos); //muestra por pantalla el
        // itinerario de nombre nombrei y matricula mat o bien de posición pos)
   void Listar();
   bool Cargar(); //permite cargar en memoria el fichero de itinerarios denominado
        // "itinerarios.dat". Devuelve true si correcto, false si error.
   bool Guardar();//Vuelca el contenido de la tabla dinámica (si ésta existe) al
        // fichero "itinerarios.dat"
};
```

Ficheros de la aplicación

El fichero de autobuses tiene la siguiente estructura

N Elementos	Autobus 1	Autobus 2	…	Autobus N

Dispone al principio de cuantos autobuses tiene en el fichero, seguido de la secuencia de autobuses almacenados en el mismo

El fichero de itinerarios tiene la siguiente estructura

Itinerario 1	Itinerario 2	Itinerario 3	…	Itinerario M

A modo de ejemplo, se propone el siguiente el contenido de los ficheros iniciales:

Autobuses.dat

```
Matrícula: 1111BBB
Capacidad: 55
Activo: 1
Depósito: 200
Ocupación: 10
Itv: 1
Conductor: 6
Matrícula: 2222BCC
Capacidad: 50
Activo: 1
Depósito: 150
Ocupación: 20
Itv: 1
Conductor: 3
Matrícula: 3333CBB
Capacidad: 60
Activo: 1
Depósito: 220
Ocupación: 15
Itv: 1
Conductor: 1
```

Itinerarios.dat

```
Itinerario: itinerario1
Origen: origen1
Destino: destino1
Inicio: 10:30 1
Fin: 2:30 0
matricula: 1111BBB
Itinerario: itinerario2
Origen: origen2
Destino: destino2
Inicio: 2:15 0
Fin: 5:0 0
Matricula: 2222BCC
```

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.