



## PROBLEMAS

1. (**FicheroAutos.cpp**) Tenemos un fichero donde se guarda información acerca de los componentes mecánicos de automóvil. La ficha de cada componente mecánico tiene la información sobre:

- el nombre de la pieza (pueden ser muchas palabras)
- unidades disponibles (es un número entero)
- precio de la pieza (puede tener decimales)

La estructura del fichero (ver fichero “autos.txt” adjunto) es:

En la primera línea del fichero nos dice cuántas piezas hay en el fichero. En la siguiente línea, el nombre de la pieza y luego, en otra línea nos encontramos las unidades disponibles y su precio.

Se trata de hacer un programa que lea la información del fichero y nos la almacene en memoria, mostrándonos a continuación esa información. A continuación se da el esquema del programa.

En el programa debe ser capaz de almacenar un **máximo** de 500 piezas. (El fichero nunca tendrá más de ese número y estará escrito correctamente).

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
#define TAM 500 //numero maximo de piezas que tendremos
struct pieza
{
    string nombre;
    int unidades;
    float precio;
};
/* Definicion de prototipos */
void LeerFichero (ifstream &f, pieza vector[TAM] , int & num );
void MostrarMemoria (pieza vector[TAM] , int num);

/* Implementacion de las funciones */
...
int main ( void )
{
    pieza lista[TAM];
    int cuantos;
    string nombre;
    ifstream fich;

    cout << "Este programa lee el fichero de piezas mecanicas.\n" ;
    cout << "Dame el nombre del fichero que contiene la información\n" ;
    cin >> nombre;

    fich.open( nombre.c_str());

    if(!fich)
        cout << "Error abriendo el fichero\n";
    else
    {
        LeerFichero ( fich, lista , cuantos );
        fich.close();
        MostrarMemoria (lista , cuantos );
    }

    system("pause");
    return 0;
}
```



Ejemplo de “fichero autos.txt”:

```
3
correa de distribucion de ford mondeo 1.4
24 123.85
disco de freno de audi
12 12.4
disco de freno de opel corsa
30 3.70
```

2. **(FicheroAutosVersion2.cpp)** Haz una versión nueva del programa anterior donde le añadas una función que nos permita recorrer toda la información que hay en memoria y la guardes en el fichero (es decir, sobrescribas el fichero anterior). Ten en cuenta que debes escribir el fichero según el formato marcado anteriormente.
3. **(FicheroAutosVersion3.cpp)** Usa el programa anterior y añádele una función que nos permita introducir los datos de una nueva pieza (esto se hará en memoria, comprobando previamente que todavía podemos añadir piezas y si no, nos lo debe avisar desde el programa principal). Esta función deberá usar otra función donde se soliciten los datos de la pieza.

Modifica a su vez el programa principal de manera que aparezca un menú con las siguientes opciones (y se repita hasta que se pulse la opción de salir):

- a- Añadir una pieza nueva
- b- Guardar el fichero
- c- Mostrar todas las piezas que tenemos
- d- Salir

Ten en cuenta que, si el fichero que contenía los datos no ha sido cargado, deberás avisarlo con un mensaje pero, igualmente, mostrar el menú anterior.

4. **(FicheroMontanya.cpp)** Hacer un programa que trabaje con información sobre montañas. Estos datos deberán guardarse en un fichero, que contendrá la siguiente información:

Nombre de la montaña (puede ser más de una palabra) y la altitud (que será en metros).

(Ver fichero “montanyas.txt” adjunto)

El programa deberá permitir hacer, desde un menú, las siguientes opciones:

- 1.- Ver todo el fichero (nombre de la montaña y su altitud)
- 2.- Añadir una montaña
- 3.- Mostrar la información de todas las montañas que contengan la palabra... (que el usuario diga)
- 4.- Guardar el fichero
- 0.- Salir

Debes definir la estructura de datos adecuada para este caso y modular bien el problema. Ten en cuenta que no sabemos cuanta información sobre montañas hay en el fichero pero nunca habrá más de 1000.



### EJERCICIOS OPCIONALES:

5. (**Matriz.cpp**) Con un editor de texto cualquiera crear un fichero llamado “matriz1.dat” que contenga la siguiente información y con el siguiente formato:

- ✓ primera línea: un entero que representa el numero de filas de la matriz
- ✓ segunda línea: un entero que representa el número de columnas de la matriz
- ✓ siguientes líneas: una fila de la matriz en cada línea, donde cada elemento está separado del siguiente por un espacio

Posteriormente hacer un programa que primero lea el contenido de la matriz y posteriormente la dibuje en forma de matriz:	Ejemplo de matriz1.dat:
	3
	2
4    -34	4 -34
4    2	4 2
0    7	0 7

**Nota:** El tamaño máximo de las matrices será de 10 x 10

6. (**MultiplicarMatriz.cpp**) Hacer un programa que lea de 2 ficheros “matriz1.dat” y “matriz2.dat” (2 matrices con el formato del ejercicio anterior) y después de multiplicarlas (si son multiplicables), muestre el resultado y lo deje en otro fichero “matriz3.dat”, con el mismo formato que las anteriores matrices.

7. (**FicheroMontanyaVersion2.cpp**) Amplia el programa 4 del fichero de montañas, de manera que se permita desde menú, las siguientes opciones:

- 1.- Ver todo el fichero
- 2.- Añadir una montaña
- 3.- Mostrar la información de todas las montañas que contengan la palabra.... (que el usuario diga)
- 4.- Guardar el fichero
- 5.- Ver el nombre de las montañas y su altitud que superan una altitud introducida por el usuario.
- 6.- Calcular la altitud media
- 0.- Salir

(**Nota:** las opciones nuevas son la 5 y la 6)

Modula el programa adecuadamente.

Ejemplo de “fichero de montañas”:

```
Mulhacén - Sierra Nevada - Granada
3482
Aneto - Pirineo central - Huesca
3404
Moncayo - Sierra de Moncayo - Zaragoza
2313
Javalambre - Sierra de Javalambre - Teruel
2020
Pico de las Nieves - Las Palmas - Gran Canaria
1949
Teide - Tenerife - Tenerife
3718
Penyagolosa - Macizo de Penyagolosa - Castellón
1814
Calderón - Sierra de Javalambre - Valencia
1837
Aitana - Sierra de Aitana - Alicante
1558
```