EJERCICIOS RESUELTOS EN CLASE DE TEORÍA

1. **(P1) (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:
   a. el nombre de la pieza (pueden ser muchas palabras)
   b. unidades disponibles (es un número entero)
   c. 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. **(P2) (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.

EJERCICIOS PARA RESOLVER

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. **(P3) (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
   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.

   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
```

5. (Op1) **(FicheroMontanyaVersion2.cpp)** Amplia el programa "FicheroMontanya.cpp" del fichero de montañas, de manera que se permita desde 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

   (Nota: las opciones nuevas son la 3 y la 4)

6. (Op2) **(FicheroMontanyaVersion3.cpp)** Amplia el programa "FicheroMontanyaVersion2.cpp" del ejercicio 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.

7. (P4) **(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 número 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: 4 -34 4 2 0 7	Ejemplo de matriz1.dat: 3 2 4 -34 4 2 0 7

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

8. (P5) **(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.

9. (puntos.cpp) Con un editor de texto cualquiera crear un fichero llamado "puntos.dat" que contenga la siguiente información y con el siguiente formato:

- ✓ primera línea: un entero que representa el número de puntos que contiene el fichero
- ✓ siguientes líneas: en cada línea las coordenadas del punto

Posteriormente hacer un programa que primero lea el contenido de la matriz y posteriormente escriba los vectores así: (2,2,7) (5,9,8) (11,6,30)	Ejemplo de puntos.dat: 3 2 2 7 5 9 8 11 6 30

Nota: El número máximo de puntos a almacenar será de 100

10. (reemplazar.cpp) Escribe una función que simule la función "reemplazar" que disponen los editores de texto. A dicha función le pasamos el nombre del a fichero donde se encuentra el texto, la cadena que queremos buscar, y la cadena que utilizamos para reemplazar. La función nos debe devolver, además, el número total de reemplazos realizados en el texto.

Usar esta función desde el programa principal, que es donde se le pedirá al usuario que introduzca el nombre del fichero, la palabra a reemplazar y la nueva. Se le mostrará por pantalla el número de reemplazos.

11. (notas.cpp) Escribe un programa que lea desde teclado la información correspondiente a cada alumno de la asignatura de "Informática" y escriba dicha información en un fichero de texto. Esta información es el identificador del alumno, su nombre completo, la nota de prácticas y la nota del examen.

12. (palabras.cpp) Según un estudio de una universidad inglesa, no importa el orden en el que las letras estén escritas, lo único importante es que la primera y la última letra estén escritas en las posiciones correctas. El resto pueden estar totalmente mal y todavía se pueden leer sin problemas. Esto es porque no leemos cada letra por si misma sino como un todo.

Ejemplo:

*Sgeun un etsduio de una uivenrsdiad ignlsea, no ipmotra el odren en el que las ltears etsan ersciats, la uicna csoa ipormtnate es que la pmrirea y la utlima ltera esten ecsritas en la psiocion cocrrtea. El rsteo peuden estar ttaolmntee mal y aun pordas lerelo sin pobrleams. Etso es pquore no lemeos cada ltera por si msima preo la paalbra es un tdoo. Pesornamelnte me preace icrneilb*e...

Siguiendo esta idea, hacer un programa que lea un fichero texto, modifique cada palabra según lo indicado anteriormente, es decir que mantenga la primera y la última letra de cada palabra y mezcle aleatoriamente el resto de posiciones y se cree un nuevo fichero con el resultado de tal modificación.

Modula el programa adecuadamente.

13. **(plásticos.cpp)** Una empresa de plásticos piensa que la empresa competidora está espiándole interceptando los ficheros de información que se transmiten mediando la red. Esta empresa nos pide que le desarrollemos un programa de encriptación para poder enviar los ficheros con seguridad por la red, y poder descifrarlos una vez recibidos. Así un fichero de texto antes de ser enviado se deberá encriptar con una clave que solo sea conocida por los trabajadores de la empresa.

El sistema de encriptación que nos proponen consiste al sumar el código ASCII de cada carácter al código ASCII de la clave. El carácter correspondiente al código ASCII de la suma es el carácter encriptado. Queremos que los código encriptados que generemos sean imprimibles por lo tanto las caracteres encriptados deben estar entre 32 y 127. Aquellos caracteres que correspondan a un código ASCII superior a 127 simplemente no lo codificaremos (por ejemplo la ñ, ç, o, etc.)

Ejemplo:

La clave está en Rebeca	Frase original
rebecarebecarebecarebe	Clave repetida
lb"fpfvf"hwyá!gq$wecgfe	Frase encriptada

14. (Op3) **(productos.cpp)** Supongamos que tenemos dos ficheros de texto productos1.dat y productos2.dat que contienen un vector de registros de productos ordenadas por el número de referencia del producto. Cada registro tiene asociada una estructura como esta:

```
struct producto
{
   int referencia;
   float cantidad;
};
```

La máxima cantidad de productos que existirán en catálogo serán 1000.

Escribe una función para realizar cada una de las siguientes tareas:

- Una función que a partir de los dos ficheros originales escriba un nuevo fichero que contenga una única lista de productos también ordenada por el número de referencia del producto.
- Una función que escriba en un nuevo fichero un listado de aquellos 10 productos del catálogo que más existencias de producto tienen.
- Una función que escriba en un nuevo fichero un listado de aquellos elementos que están por arriba de la media de existencias del catálogo de productos.

Modula el programa adecuadamente.