Relación de Ejercicios de Ficheros con Structs (2)

En ocasiones, nos va a interesar usar un fichero de texto en lugar de uno binario. Los ficheros binarios tienen varias ventajas: ocupan menos espacio, son más rápidos, no permiten al usuario ver los datos directamente con el bloc de notas, etc. La principal ventaja de los ficheros de texto es que podemos ver la información fácilmente y así detectar errores. Otra ventaja es la posibilidad de usar el formato de intercambio CSV para importar y exportar datos desde otras aplicaciones, como Excel, Access y otras bases de datos que usen este formato.

Añade estas funciones al proyecto correspondiente de *Alumno* y *Banco*.

1. Escribe la función *EscribeFicheroAlumnosTXT* similar a la función *EscribeFicheroAlumnos* pero usando un fichero de texto. La estructura del fichero será similar, pero ahora guardaremos un valor en cada línea. Ejemplo:

   ```
   2
   Pedro
   23
   7,2
   Juan
   15
   2,1
   ```

2. Escribe la función *LeeFicheroAlumnosTXT* similar a la función *LeeFicheroAlumnos* pero que funciona con los ficheros de texto del ejercicio anterior.

3. Escribe la función *GuardarFicheroBancoTXT* que hace lo mismo que la función *GuardarFicheroBanco* que hicimos anteriormente pero con ficheros de texto. Ejemplo:

   ```
   1
   83749
   Juan Jiménez
   1050,50
   ```

4. Escribe la función *LeerFicheroBancoTXT* que hace lo mismo que la función *LeerFicheroBanco* pero con ficheros de texto.

Los ficheros CSV (Comma Separated Values) (Valores Separados por Comas) son la forma más simple de guardar una tabla de una base de datos. Cada registro va en una línea y los valores de cada registro van separados por comas u otro valor de separación (por ejemplo, punto y coma).

5. Escribe la función *EscribeFicheroAlumnosCSV*. En este caso, no vamos a guardar el número de registros en la primera línea, ya que los ficheros CSV estándar no lo hacen. Como carácter separador usaremos el punto y coma, ya que uno de los datos que usamos ya contiene comas.
   Un ejemplo del fichero sería:

   ```
   Pedro;23;7,2
   Juan;15;2,1
   ```

6. Escribe la función *LeeFicheroAlumnosCSV* que lee los datos del fichero anterior en una lista. En este caso no sabemos cuántas líneas tenemos, así que usaremos EndOfStream. Para separar los datos, lo más fácil es usar la función *Split* de las cadenas, que nos divide la cadena en subcadenas usando como carácter de separación el que nosotros le digamos. Por ejemplo:

   ```
   string s = "Pedro;23;7,2";
   string[] cad = s.Split(';');
   ```

   Con esto conseguiremos un array de cadenas compuesto por tres cadenas: "Pedro", "23", "7,2", que podremos meter en el elemento ficha_alumno y luego en la lista.

7. Escribe la función *GuardarFicheroBancoCSV* que guarda los datos del banco en un fichero CSV. Ejemplo:

   ```
   29348348;Juan Jiménez;3432,39
   23487239;José Pérez;10,5
   ```

8. Escribe la función *LeerFicheroBancoCSV* que lee los datos del banco del fichero CSV.