PRÁCTICA VI

Ejercicio 6.1 – Archivos de texto

Realizar un programa que abra un archivo de texto (que contenga un texto en castellano) y que muestre el contenido del mismo en pantalla. Luego de mostrar el contenido, el programa debe indicar la cantidad de caracteres que fueron leídos del archivo.

Ejercicio 6.2 – Archivos de texto

Realizar un programa que guarde todo lo que el usuario tipee en un archivo de texto. Lo primero que realizará el programa es solicitar al usuario que ingrese el nombre del archivo. A continuación el usuario comenzará a tipear el texto que deberá guardarse en dicho archivo. La carga de datos finaliza cuando el usuario tipea dos ENTERs seguidos. Verifique con el Bloc de Notas o con cualquier editor de texto que el archivo haya sido creado correctamente.

Ejercicio 6.3 – Archivos de texto

Realizar un programa que abra un archivo de texto (que contenga un texto en castellano) y que cree otro archivo con un listado (sin repeticiones) de las diferentes palabras que hay en dicho texto.

Ejercicio 6.4 – Archivos binarios

Dada la siguiente declaración:

```
typedef struct
{
     char  dni[9];
     char nombre [100], apellido [100];
     double monto_adeudado;
}
t_datos;
```

Realizar un programa que solicite al usuario que ingrese por teclado dichos datos y los grabe en un archivo binario en modo ***append.*** La carga de datos finalizará cuando el usuario ingrese como DNI el cero. Antes de cargar los datos, el usuario debe indicar por teclado el nombre de archivo.

Ejercicio 6.5 – Archivos binarios

Dado el struct del ejercicio anterior, realizar un programa que solicite al usuario un nombre de archivo y a continuación abra y muestre el contenido de ese archivo, en formato de listado por pantalla, separando los datos en columnas y mostrando en cada renglón un registro distinto.

Ejercicio 6.6 – Archivos binarios

Dado el struct del ejercicio anterior, realizar un programa que solicite al usuario un nombre de archivo y a continuación abra dicho archivo y traspase desde el mismo los datos hacia otro archivo (cuyo nombre tambien debe ingresar el usuario) sin incluir registros con dni repetido. En caso de que haya repeticiones en el archivo original, deben sumarse todos los montos adeudados y grabarse en un único registro para cada DNI.

Ejercicio 6.7 – Archivos binarios

Dada la siguiente declaración:

```
typedef struct
{
     char nombre [100], domicilio [200];
     unsigned int codigo_postal, nro_documento;
     unsigned char edad_sexo;
}
t_persona;
```

se utilizará esta estructura para declarar variables que permitan guardar datos de personas. El campo `edad_sexo` contiene la edad de la persona desde el bit 0 hasta el bit 6 y el sexo en el bit 7 (1 para masculino, 0 para femenino).
Programar las siguientes funciones:

1) Función que reciba como parámetro un nombre de archivo binario (char *) y permita ingresar por teclado los datos de una persona para luego ser agregados en el archivo binario cuyo nombre se recibió como parámetro.
2) Función que reciba como parámetros un nombre archivo de texto csv y un nombre de archivo binario (ambos como char *) y que convierta el contenido del archivo binario en un archivo tipo CSV(*).
3) Función que reciba como parámetros dos nombres de archivo binario (char *) y determine si ambos contienen los mismos registros (no necesariamente en el mismo orden). Es decir que todos los registros de un archivo estén en el otro y viceversa.
4) Función que reciba como parámetros un número de documento y un nombre de archivo y retorne un registro con los datos completos de dicha persona.
5) Función que reciba como parámetros un código postal y un nombre de archivo y retorne un arreglo dinámico de registros con los datos completos de las personas con ese código postal.

Implementar estas funciones en un programa de ejemplo.

(*): Un archivo CSV (Comma Separated Values) es un archivo de texto que contiene registros. Los campos están separados por coma y cada línea contiene un registro diferente, es decir, los registros están separados por '\n'. Dichos archivos pueden ser generados con Microsoft Excel ® (*Archivo → Guardar como → CSV (delimitado por coma))*.

Adicionales

- Elaborar funciones para volcar el contenido de un ABB en un archivo y armar un ABB desde los datos guardados en un archivo. El almacenamiento debe hacerse de forma tal que se mantenga la estructura del árbol. Para esto, cuando se guarde en el archivo deberá hacerse un barrido por niveles y cuando se cargue, deberá hacerse en forma ordenada.
  Dichas funciones deberán aplicarse en un programa que muestre un menú con las opciones: agregar nodos al árbol, guardar el árbol, cargar el árbol (permitiendo seguir agregando nodos), mostrar el árbol y salir del sistema.