Facultad de Informática – Universidad Complutense

Fundamentos de la programación

Curso 2012–2013

Hoja de ejercicios del Tema 10

1. Dados dos archivos binarios (`uno.dat` y `dos.dat`) que contienen secuencias ordenadas de enteros pequeños sin signo (`unsigned short int`), desarrolla los siguientes subprogramas:

- ✓ `unir()`: Crea un tercer archivo binario (`tres.dat`) con el contenido del primer archivo seguido del contenido del segundo archivo.
- ✓ `concatenar()`: Añade el contenido del segundo archivo al final del primer archivo.
- ✓ `intercalar()`: Genera un nuevo archivo binario intercalando las secuencias de partida (coge un elemento de cada una cada vez). Por ejemplo, el intercalado de las secuencias 1,2,3,7 y 4,5,6 es 1,4,2,5,3,6,7.
- ✓ `mezclar()`: Genera un nuevo archivo binario mezclando las secuencias iniciales y preservando el orden. Por ejemplo, la mezcla de las secuencias ordenadas 1,2,3,7 y 4,5,6 es 1,2,3,4,5,6,7.

Desarrolla también un programa de prueba que utilice los subprogramas anteriores.

2. Partiendo del ejercicio sobre gestión de listas de empleados (ejercicio 10 del Tema 5 que posteriormente modificaste en el Tema 6 –ejercicio 9– y en el Tema 9 –ejercicio 11–), añádele al programa la capacidad para cargar y guardar la lista de empleados en un archivo binario de empleados.

Nota: Tendrás que redefinir el campo nombre de los empleados y cambiarlo por una cadena de tipo array de char (estilo C).

3. Queremos preparar un programa para el mantenimiento del censo de un pueblo pequeño, con menos de 100 habitantes. El programa almacenará la siguiente información de cada uno de los habitantes censados:

- ✓ Nombre (cadena de estilo C de hasta 120 caracteres)
- ✓ Apellido 1 (cadena de estilo C de hasta 120 caracteres)
- ✓ Apellido 2 (cadena de estilo C de hasta 120 caracteres)
- ✓ Sexo (hombre, mujer)
- ✓ Estado civil (soltero, casado, pareja de hecho, divorciado, viudo)
- ✓ Dirección (cadena de estilo C de hasta 500 caracteres)

- ✓ Fecha de nacimiento (estructura con día, mes y año)
- ✓ DNI (sin letra, entero sin signo)
- ✓ Código postal (entero corto sin signo)

Define las estructuras de datos necesarias para almacenar una lista de hasta 100 habitantes y responde a las siguientes preguntas (escribe un programa para ayudarte):

- ✓ ¿Cuántos bytes ocupa el estado civil?
- ✓ ¿Cuántos bytes ocupa la fecha de nacimiento?
- ✓ ¿Cuántos bytes ocupa un registro completo de un habitante?
- ✓ ¿Cuántos bytes ocupa la lista completa?
- ✓ Si cambias las cadenas de estilo C por variables de tipo string, ¿cuánto ocupa un registro completo de un habitante? ¿por qué?

4. Partiendo de la estructura de datos del ejercicio anterior, desarrolla un programa con las siguientes funcionalidades:

a) Añadir un nuevo habitante al censo manteniendo la lista ordenada por DNI.

b) Eliminar un dato de la lista a partir de un número de DNI (deberás buscar el DNI con una búsqueda binaria y eliminarlo sólo si existe).

c) Guardar la lista del censo en un archivo binario de habitantes.

d) Cargar la lista del censo a partir de un archivo binario de habitantes.

5. Dado el éxito del programa del censo, queremos vendérselo a grandes ciudades para que almacenen sus propios censos. Para ello, vamos a aumentar el tamaño del censo para soportar un máximo de 30 millones de habitantes. ¿Cuántos bytes son necesarios para almacenar la nueva lista de tamaño variable si tuviese un único habitante censado?

Dado que no resulta práctico tener todo el censo cargado en memoria, reimplementa tu programa para que, en lugar de usar una lista de registros, emplee un archivo binario de registros y trabaje directamente el archivo.

6. Partiendo del ejercicio 7 del Tema 8 (campeonatos de atletismo), modifica el programa para que, en lugar de trabajar con una lista de atletas, trabaje directamente sobre un archivo binario de atletas.