Facultad de Informática – Universidad Complutense

Fundamentos de la programación – Grupos

Curso 2015–2016

Examen final – 6 de septiembre de 2016

Tiempo disponible: 3 horas

Se pide construir un programa modular que permita crear grupos de chat a partir de una lista de contactos que está en un archivo en disco. El programa constará de cuatro módulos: *Contacto*, *ListaContactos*, *GrupoChat* y módulo principal (`main.cpp`).

Módulo *Contacto* (1 punto)

Declara un tipo de estructura `tContacto` con los campos: identificador, nombre completo, email y teléfono (cuatro cadenas de caracteres, el nombre posiblemente con espacios). Implementa, al menos, las siguientes funciones:

- ✓ `cargar()`: Carga un contacto de un flujo. En el archivo cada contacto consta de 4 líneas, una por cada campo (ver ejemplo de archivo al final del enunciado).
- ✓ `mostrar()`: Dado un contacto lo muestra en la salida estándar, como aparece en el ejemplo al final del enunciado.

Módulo *ListaContactos* (2 puntos) Máx. 50 contactos

Declara un tipo de estructura `tListaContactos` para la lista de contactos (hasta 50). Esta lista estará implementada con un **array estático de punteros a variables dinámicas**. Implementa, al menos, las siguientes funciones:

- ✓ `cargar()`: Carga la lista de contactos de un flujo. El archivo comienza con el número de contactos que contiene (en una línea), y a continuación aparece la información de cada uno de ellos.
- ✓ `destruir()`: Libera la memoria dinámica que utiliza una lista de contactos.

Módulo *GrupoChat* (4 puntos)

Define un tipo de estructura `tElemento` con un puntero a un contacto (de la lista de contactos) y el número de mensajes enviados al grupo por ese contacto (un entero). Declara un tipo de estructura `tGrupoChat` para listas de `tElemento` implementadas con un **array dinámico**. Además guardará el nombre del grupo (una cadena de caracteres con posibles espacios). La lista estará ordenada por el identificador.

Implementa, al menos, las siguientes funciones:

- ✓ `nuevo()`: Dado un nombre y una capacidad (`dim`), devuelve un grupo de chat vacío, adecuadamente inicializado para poder contener `dim` elementos.

- ✓ `insertar()`: Dado un grupo de chat y un elemento, si el identificador del contacto ya está en el grupo devuelve falso. En otro caso lo inserta en la posición que le corresponde en el orden (no es necesario redimensionar la lista).
- ✓ `buscar()`: Dado un grupo de chat y el identificador de un contacto, determina si el contacto se encuentra en el grupo y la posición en la que está. Si no está en el grupo, devuelve la posición en el que debería estar. Debe hacerse de forma **recursiva**.
- ✓ `mostrar()`: Dado un grupo de chat, muestra por pantalla su nombre, los contactos y el número total de mensajes intercambiados. Sigue el formato del ejemplo al final del enunciado.
- ✓ `combinar()`: Dados dos grupos de chat con nombres `"A"` y `"B"`, genera y devuelve un tercer grupo de chat (con nombre `"A y B"`) con los elementos de los dos grupos combinados de forma que si un contacto aparece en los dos grupos, en el grupo resultante el número total de mensajes enviados es la suma de los enviados a cada grupo. Ver ejemplo de ejecución al final del enunciado. Debe hacerse sin realizar desplazamientos de los elementos en el array (0,5 puntos).
- ✓ `destruir()`: Dado un grupo de chat, libera la memoria dinámica que utiliza.

Módulo principal (3 puntos)

Carga los contactos del archivo `contactos.txt` en una lista de contactos y a continuación solicita al usuario un nombre de grupo y un número de contactos `numContactos`, crea un nuevo grupo de chat con ese nombre y `numContactos` contactos distintos, elegidos aleatoriamente de la lista de contactos (el número de mensajes de cada contacto también se decide de forma aleatoria, un valor positivo menor a 100). Crea un segundo grupo de chat de la misma forma y combina los dos grupos en un tercer grupo, mostrando en la pantalla el contenido de los tres grupos de chat. Al salir se deberá liberar toda la memoria dinámica utilizada.

Nota: Al inicio del `main`, pon la instrucción `srand(1)` para que al hacer las pruebas siempre se obtenga la misma lista de reproducción aleatoria.

Se valorará la legibilidad, así como el uso adecuado de los esquemas de recorrido y búsqueda, de la comunicación entre subprogramas y de la memoria.

Recuerda: El comando para que se muestre la memoria no liberada es

```
_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF);
```

Entrega el código del programa (solo los ficheros fuente: .cpp y .h). Añade al inicio del .cpp del módulo principal un comentario con tus datos (nombre completo, DNI, y nº de puesto).

Ejemplo de ejecución:

```
Nombre del grupo: Uno
Numero de contactos: 2
Nombre del grupo: Dos
Numero de contactos: 2
--------------------------
Grupo de chat: Uno
--------------------------
Contacto: juanmi - Juan Miguel Garcia
   email: jmgarcia@yahoo.vt - tlf: 600111223
Numero de mensajes: 16
Contacto: marti - Martina Martin
   email: marti@gmail.es - tlf: 600111222
Numero de mensajes: 95
TOTAL MENSAJES: 111
--------------------------
Grupo de chat: Dos
--------------------------
Contacto: dal - Dalia Martin
   email: dmartin@gmail.es - tlf: 600111223
Numero de mensajes: 50
Contacto: juanmi - Juan Miguel Garcia
   email: jmgarcia@yahoo.vt - tlf: 600111223
Numero de mensajes: 31
TOTAL MENSAJES: 81
--------------------------
Grupo de chat: Uno y Dos
--------------------------
Contacto: dal - Dalia Martin
   email: dmartin@gmail.es - tlf: 600111223
Numero de mensajes: 50
Contacto: juanmi - Juan Miguel Garcia
   email: jmgarcia@yahoo.vt - tlf: 600111223
Numero de mensajes: 47
Contacto: marti - Martina Martin
   email: marti@gmail.es - tlf: 600111222
Numero de mensajes: 95
TOTAL MENSAJES: 192
Presione una tecla para continuar . . .
```

Ejemplo de archivo:

contactos.txt

Archivo Edición Formato Ver Ayuda

```
5
juanmi
Juan Miguel Garcia
jmgarcia@yahoo.vt
600111223
rakh
Teresa Cospe Garcia
rcospe@gmail.es
600111223
jonas
Jaime Garcia
jonas@gmail.es
600111222
marti
Martina Martin
marti@gmail.es
600111222
dal
Dalia Martin
dmartin@gmail.es
600111223
```

Archivo checkML.h

```
#ifdef _DEBUG
#define _CRTDBG_MAP_ALLOC
#include <stdlib.h>
#include <crtdbg.h>
#ifndef DBG_NEW
#define DBG_NEW new ( _NORMAL_BLOCK , __FILE__ , __LINE__ )
#define new DBG_NEW
#endif
#endif
```

Puedes copiar los archivos `contactos.txt` y `checkML.h` que están en el CV.