Facultad de Informática – Universidad Complutense de Madrid

Fundamentos de la Programación

Curso 2016–2017, Grupos E, F y G

Examen de Septiembre

Duración del examen: 3 horas

Se pide construir un programa modular en C++ que permita gestionar grupos de alumnos. La solución constará de un programa principal (**`main.cpp`**) y tres módulos: *ListaAsignaturas*, *ListaMatriculas* y *Grupo*.

Módulo *ListaAsignaturas* (1.5 puntos) Máx. 30 asignaturas

Declara un tipo de estructura **`tAsignatura`** con tres campos: código, nombre y créditos. Define también un tipo **`tListaAsignaturas`** para la lista de asignaturas (hasta 30). Esta lista no está ordenada.

Implementa, al menos, las siguientes funciones:

- ✓ `cargarAsignaturas()`. Carga una lista de asignaturas. Entre los datos que se encuentran en el archivo **`matriculas.txt`** están los datos correspondientes a las distintas listas de asignaturas que corresponden a cada alumno. En estas listas aparece el número de asignaturas y cada asignatura consta de tres líneas: código, nombre y créditos (ver ejemplo de archivo al final del enunciado).
- ✓ `insertarAsignatura()`. Añade una asignatura al final de una lista de asignaturas.

Módulo *ListaMatrículas* (3 puntos) Máx. 50 matriculas

Declara un tipo **`tMatricula`** con seis campos: apellidos, nombre, nif (8 dígitos y una letra en una cadena), tipo de estudios (cadena), asignaturas (una lista del tipo **`tListaAsignaturas`**) y coste (precio total con coste de crédito 26.50 €). Añade un tipo **`tListaMatriculas`** para la lista de matrículas (hasta 50). Implementa esta lista con un **array estático de punteros a variables dinámicas** y mantenla **ordenada** por apellidos de menor a mayor.

Implementa, al menos, las siguientes funciones:

- ✓ `cargarMatriculas()`. Carga la lista de matrículas del archivo **`matriculas.txt`**. El fichero comienza con el número de matrículas que contiene, y a continuación aparece la información de cada una de ellas.
- ✓ `insertarMatricula()`. Inserta una nueva matrícula en una lista ordenada de matrículas. Debe mantenerse el orden.
- ✓ `mostrarMatriculas()`. Muestra una lista de matrículas según el formato dado en el ejemplo.

- ✓ `seleccionarMatricula()`. Permite seleccionar una matrícula de la lista de matrículas. Muestra la lista y el usuario puede elegir por teclado una matrícula según el número de orden. Devuelve dicha posición en la lista (orden).
- ✓ `liberar()`. Libera la memoria dinámica utilizada con una lista de matrículas.

Módulo *Grupo* (4 puntos) Máx. 10 alumnos

Define un tipo **`tAlumno`** con un puntero a una matrícula (de la lista de matrículas) y una cuenta de correo electrónico. Declara también un tipo **`tGrupo`** para listas de **`tAlumno`** implementadas con **array dinámico** que guardan el identificador del grupo y el aula asignada (un entero del 1 al 20) además del array y el contador. Estas listas no guardan ningún orden.

Implementa, al menos, las siguientes funciones:

- ✓ `nuevo()`. Dado un identificador y un aula, devuelve un grupo inicializado.
- ✓ `leer()`. Dada la lista de matrículas, da a elegir al usuario un alumno matriculado para a continuación confirmar que no está en el grupo (utilizando el nif) y asignar el puntero obtenido al nuevo alumno. Solicita también la cuenta de correo electrónico del alumno. La función devuelve el nuevo alumno.
- ✓ `mostrarAlumno()`. Dado un alumno muestra sus datos en una línea según el formato dado.
- ✓ `insertarAlumno()`. Inserta un nuevo alumno al final de un grupo.
- ✓ `buscarAlumno()`. Busca un alumno en un grupo utilizando un nif.
- ✓ `eliminarAlumno()`. Elimina un alumno de un grupo según su nif.
- ✓ `mostrarGrupo()`. Muestra toda la información de un grupo siguiendo el formato dado. Debe implementarse de forma recursiva.
- ✓ `liberar()`. Libera la memoria dinámica utilizada con un grupo.

Programa Principal (1.5 puntos)

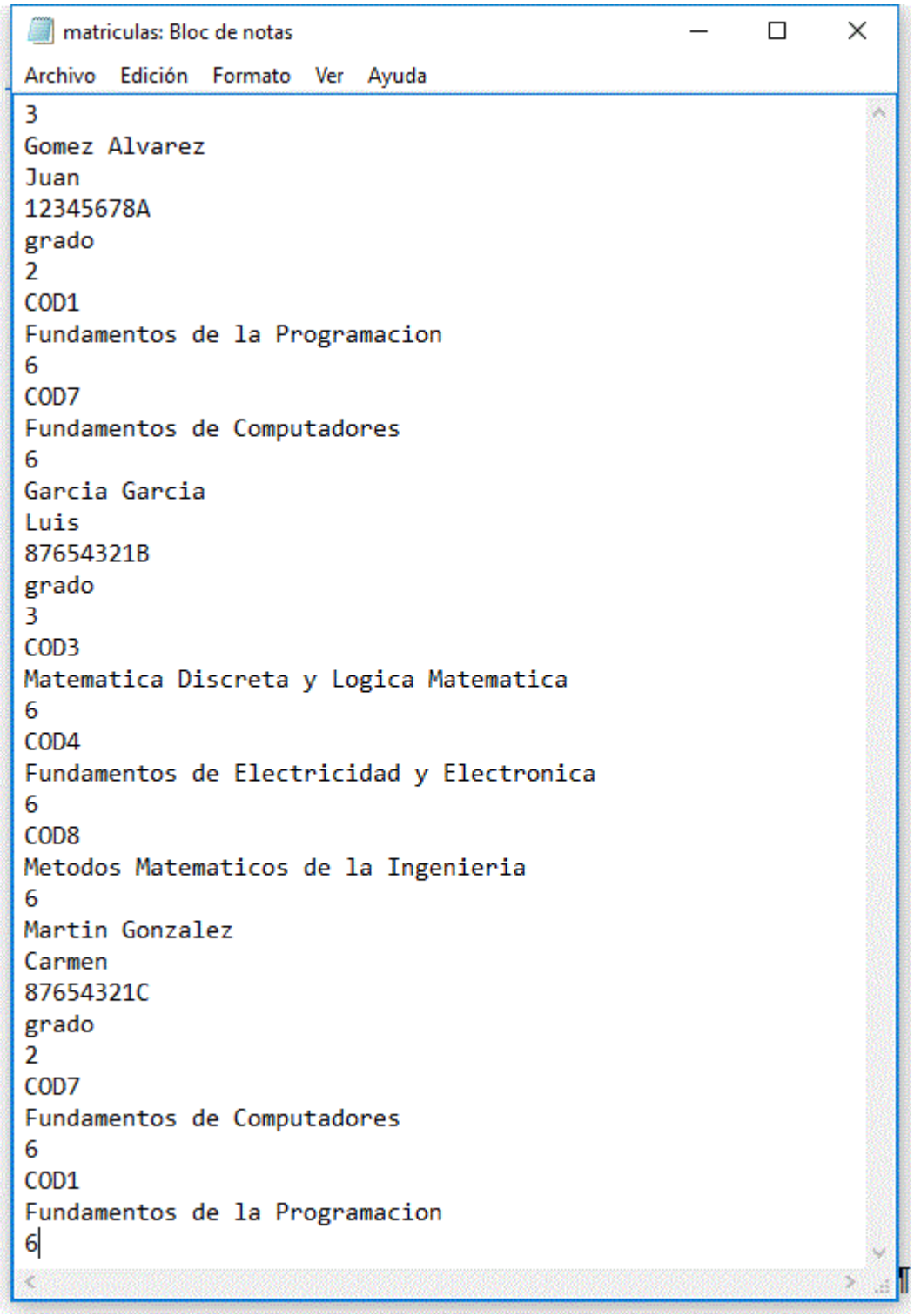
Carga el contenido del archivo **`matriculas.txt`** en la lista maestra de matrículas, crea un nuevo grupo vacío (solicitando el identificador y el aula) y muestra al usuario un menú con tres opciones más la de salida (opción 0): añadir un alumno al grupo, eliminar un alumno del grupo y mostrar el estado actual del grupo. Al eliminar un alumno de un grupo debe mantenerse la información de la matrícula.

Al salir debe liberarse toda la memoria dinámica utilizada.

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

Entrega el código del programa (sólo .cpp y .h comprimidos en un ZIP) utilizando la herramienta de FTP del escritorio. Asegúrate de entregar una versión sin errores de compilación.

Ejemplo de archivo `matriculas.txt`



Ejemplos de ejecución: opciones 1 y 3

```
Introduce el dentificador de grupo: E
Introduce el aula (1 a 20): 2

1. Aniadir un alumno al grupo
2. Eliminar un alumno al grupo
3. Mostrar el estado actual del grupo
0. Salir
Elige opcion: 1

La lista de matriculas cargadas es:
  1. Garcia Garcia       87654321B  (3)  477.00
  2. Gomez Alvarez       12345678A  (2)  318.00
  3. Martin Gonzalez     87654321C  (2)  318.00

Introduce el numero de linea del alumno que se quiere aniadir al grupo: 1
Cuenta de correo: garcia@ucm.es

1. Aniadir un alumno al grupo
2. Eliminar un alumno al grupo
3. Mostrar el estado actual del grupo
0. Salir
Elige opcion: 3

1.  Garcia Garcia       garcia@ucm.es

1. Aniadir un alumno al grupo
2. Eliminar un alumno al grupo
3. Mostrar el estado actual del grupo
0. Salir
Elige opcion: 1

La lista de matriculas cargadas es:
  1. Garcia Garcia       87654321B  (3)  477.00
  2. Gomez Alvarez       12345678A  (2)  318.00
  3. Martin Gonzalez     87654321C  (2)  318.00

Introduce el numero de linea del alumno que se quiere aniadir al grupo: 2
Cuenta de correo: gomez@ucm.es

1. Aniadir un alumno al grupo
2. Eliminar un alumno al grupo
3. Mostrar el estado actual del grupo
0. Salir
Elige opcion: 3

1.  Garcia Garcia       garcia@ucm.es
2.  Gomez Alvarez       gomez@ucm.es
```

Ejemplos de ejecución: opciones 2 y 3

```
1. Aniadir un alumno al grupo
2. Eliminar un alumno al grupo
3. Mostrar el estado actual del grupo
0. Salir
Elige opcion: 2

Nif del alumno que se desea eliminar: 87654321B

1. Aniadir un alumno al grupo
2. Eliminar un alumno al grupo
3. Mostrar el estado actual del grupo
0. Salir
Elige opcion: 3

1.  Gomez Alvarez       gomez@ucm.es

1. Aniadir un alumno al grupo
2. Eliminar un alumno al grupo
3. Mostrar el estado actual del grupo
0. Salir
Elige opcion:0
```

Memoria Dinámica

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

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

// 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
```

Esquema de Código

```
int main()
{
    _CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF);
    tGrupo grupo;
    tListaMatriculas lista;
    tAlumno alumno;
    string id, nif;
    int opcion, aula;
    bool ok = false;
```

```
    cargar(lista, ok);
    if (ok)
    {
        // Tu código
        do
        {
            opcion = menu();
            switch (opcion)
            {
                case 1:
                    // Tu código
                case 2:
                    // Tu código
                case 3:
                    // Tu código
            }
        } while (opcion != 0);
        // Tu código
    }
    else
    {
        cout << "Error al cargar la lista de matrículas." << endl;
    }
    return 0;
} // main
```

Entrega del Examen

1. Añade al inicio de tus archivos un comentario con tus datos:

```
/*
Apellidos:
Nombre:
DNI:
Puesto:
*/
```

2. Abre la herramienta de entrega de exámenes por FTP que hay en el escritorio de tu ordenador.
3. Úsala para subir tus archivos (arrastra tus ficheros hacia la ventana derecha).
4. Pasa por el ordenador del profesor, pregúntale si tu archivo se ha recibido correctamente, y firma.