Facultad de Informática – Universidad Complutense de Madrid

Fundamentos de la Programación

Curso 2018–2019, Grupos D, G, DG-ADE

Examen de Julio

Duración del examen: 3 horas

Una agencia de contratación necesita un programa modular en C++ para gestionar los contratos que ofrecen a estudiantes que están finalizando sus carreras. El programa consistirá en un programa principal y tres módulos: *ListaEmpresas*, *ListaEstudiantes* y *Agencia*.

Módulo *ListaEmpresas* (4 puntos) Máx. 10 empresas

Declara un tipo **`tHistoria`** como matriz de 5 años (filas) por 10 empresas (columnas). La matriz representa el número de estudiantes contratados (entre 1 y 50) por cada empresa en los últimos 5 años. Añade un tipo **`tEmpresa`** como estructura con cuatro campos para representar la informacion acerca de la empresa: código (entero entre 1 y 10), nombre (cadena), sector (cadena) y número de contratos realizables (entero). Define también un tipo **`tListaEmpresas`** para listas de empresas con **array estático de punteros a variables dinámicas** y mantenla **ordenada por código** de menor a mayor.

Implementa, al menos, los siguientes subprogramas:

- ✓ `cargar(...)`. Carga una matriz del tipo **`tHistoria`** con los números de estudiantes contratados por las 10 empresas en los últimos 5 años desde el archivo **`historia.txt`**; devuelve un boolean para decir si el proceso ha tenido éxito.
- ✓ `cargar(...)`. Carga una lista de empresas del tipo **`tListaEmpresas`** del archivo **`empresas.txt`** y devuelve un boolean para decir si el proceso ha tenido éxito. El fichero comienza con el número de empresas que contiene, y a continuación aparece la información de cada una de ellas (código, nombre y sector, cada uno en una línea). El número de contratos realizables por cada empresa no está en el fichero y debe ser obtenido usando la función `mediaContratos()`.
- ✓ `mediaContratos(...)`. Dada una matriz del tipo **`tHistoria`** y una columna (código de empresa – 1) válida, calcula el número de contratos realizables por la empresa según la media de los 5 años en esa empresa. Redondea la media al entero más grande no mayor de x con la función floor(x) de la librería **`cmath`**.
- ✓ `insertar(...)`. Dada una lista de empresas del tipo **`tListaEmpresas`** y una empresa del tipo **`tEmpresa`**, inserta la empresa en la lista manteniendo su orden (no puede usarse ningún algoritmo de ordenación). Devuelve un boolean que indica si la inserción se pudo realizar.

- ✓ `seleccionar(...)`. Dada una lista de empresas del tipo **`tListaEmpresas`**, la visualiza y permite al usuario seleccionar una de las empresas; para ello pregunta por un código de empresa y lo valida y devuelve la posición de la empresa en la lista.
- ✓ `mostrar(...)`. Dada una lista de empresas del tipo **`tListaEmpresas`**, muestra la información de cada empresa en una línea, siguiendo el formato del ejemplo de ejecución.
- ✓ `liberar(...)`. Dada una lista de empresas del tipo **`tListaEmpresas`**, libera la memoria dinámica utilizada por la lista.

Módulo *ListaEstudiantes* (1.5 puntos) Máx. 100 estudiantes

Declara un tipo **`tEstudiante`** para representar la información de un estudiante como estructura con dos campos: nombre (cadena) y NIF (cadena). Añade un tipo **`tListaEstudiantes`** para listas de **`tEstudiante`** (sin orden). Puedes elegir la implementación.

Implementa, al menos, los siguientes subprogramas:

- ✓ `cargar(...)`. Carga una lista de estudiantes del tipo **`tListaEstudiantes`** desde el archivo **`estudiantes.txt`**; devuelve un boolean para decir si el proceso ha tenido éxito. El fichero comienza con el número de estudiantes que contiene, y a continuación aparece la información de cada uno de ellos (nombre y NIF, cada uno en una línea).
- ✓ `buscar(...)`. Dada una lista de estudiantes del tipo **`tListaEstudiantes`** y un NIF devuelve la posición del estudiante en la lista (o -1 si no existe).
- ✓ `seleccionar(...)`. Dada una lista de estudiantes del tipo **`tListaEstudiantes`**, la visualiza y permite al usuario seleccionar un estudiante por posición y valida ésta; devuelve la posición del estudiante en la lista.
- ✓ `mostrar(...)`. Dada una lista de estudiantes del tipo **`tListaEstudiantes`**, muestra la información de cada estudiante en una línea, comenzando con la posición del estudiante en la lista y siguiendo el formato del ejemplo de ejecución.

Módulo *Agencia* (4 puntos)

Define un tipo **`tContrato`**, para representar la información de un contrato entre una empresa y un estudiante, como estructura con tres campos: un puntero a una empresa existente en la lista de empresas, el NIF de un estudiante y un sueldo (número real). Declara también un tipo **`tAgencia`** para listas de **`tContrato`** implementadas con **array dinámico**. Estas listas no guardan ningún orden.

Implementa, al menos, los siguientes subprogramas:

- ✓ `inicializar(...)`. Devuelve una lista de contratos (una agencia) del tipo **`tAgencia`** con espacio para 3 contratos.
- ✓ `insertar(...)`. Dada una agencia del tipo **`tAgencia`**, una lista de empresas del tipo **`tListaEmpresas`** y una lista de estudiantes del tipo **`tListaEstudiantes`**, permite al usuario seleccionar una empresa y un estudiante (a través de las correspondientes

operaciones `seleccionar` antes descritas) y proporcionar un salario. Si el estudiante ya tiene un contrato o la empresa no tiene más contratos realizables no se hace nada; si no se incorpora el nuevo contrato a la agencia. Si no hay espacio en la agencia para un nuevo contrato, el array dinámico de la lista es substituido por otro con 3 elementos más disponibles. Recuerda actualizar el número de contratos realizables en la empresa (disminuyéndolo en 1) si el contrato se pudo realizar. El subprograma devolverá un boolean indicando si se pudo realizar el contrato o no .

- ✓ `buscar(...)`. Dada una agencia del tipo **`tAgencia`** y un NIF de estudiante, devuelve la posición del contrato del estudiante en la lista o -1 si no lo encuentra. **Debe implementarse de forma recursiva**.
- ✓ `mostrar(...)`. Dado un contrato del tipo **`tContrato`** y una lista de estudiantes del tipo **`tListaEstudiantes`**, muestra la información del contrato siguiendo el formato del ejemplo de ejecución.
- ✓ `mostrar(...)`. Dada una agencia del tipo **`tAgencia`** y una lista de estudiantes del tipo **`tListaEstudiantes`**, muestra la información de la agencia siguiendo el formato del ejemplo de ejecución.
- ✓ `liberar(...)`. Dada una agencia del tipo **`tAgencia`** libera la memoria dinámica utilizada.

Programa Principal (0.5 puntos)

El programa principal es proporcionado pero faltan algunas instrucciones que debes introducir. Este programa comienza cargando la matriz, la lista de empresas y la lista de estudiantes. A continuación crea una nueva agencia y permite al usuario elegir entre insertar un nuevo contrato, localizar un contrato o mostrar la lista de contratos (o salir).

Ficheros de Datos

`historia.txt`

```
2 5 10 23 45 23 20 19 20 17
7 9 19 24 34 33 50 18 19 25
8 10 20 28 26 33 34 35 34 20
9 11 21 28 27 30 35 37 39 40
7 10 22 29 28 33 37 39 40 41
```

`empresas.txt`

```
7
3
Apple
Tecnologia
.  .  .
10
Telefonica
Comunicaciones
```

`estudiantes.txt`

```
12
Juan Castellano
12345678A
Ana Fuentes
87654321B
.  .  .
Harry Potter
00000000H
```

Ejemplo de Ejecución

```
1 - Insertar un nuevo contrato
2 - Buscar un contrato
```

```
3 - Mostrar los contratos
0 - Salir
Elige opcion: 1

CodigoEmpresa        Sector         Contratos realizables
-----------------------------------------------------
  1   Amazon         Nube                    6
  2   IBM            Servicios               9
  3   Apple          Tecnologia             18
  4   Microsoft      Software               26
  6   Google         Tecnologia             30
  7   Facebook       Tecnologia             35
 10   Telefonica     Comunicaciones         28
Por favor, introduzca el codigo: 3

Lista de estudiantes...
   1  12345678A  Juan Castellano
   2  87654321B  Ana Fuentes
   3  55667788C  Pedro Gonzalez
   4  22334455D  Maria Gomez
   5  12121212F  Carolina Suarez
   6  33443344G  Gregorio Hernandez
   7  23123425Y  Rosa Martin
   8  98765432M  Ana Hernandez
   9  66644666P  Gandalf the Grey
  10  09090909B  Bilbo Baggins
  11  44466444D  Darth Vader
  12  00000000H  Harry Potter
Por favor, introduzca la posicion en la lista: 3
Sueldo: 1659.4

1 - Insertar un nuevo contrato
2 - Buscar un contrato
3 - Mostrar los contratos
0 - Salir
Elige opcion: 3

Lista de contratos...
Apple          Pedro Gonzalez         1659.40

1 - Insertar un nuevo contrato
2 - Buscar un contrato
3 - Mostrar los contratos
0 - Salir
Elige opcion: 1

CodigoEmpresa        Sector         Contratos realizables
-----------------------------------------------------
  1   Amazon         Nube                    6
  2   IBM            Servicios               9
  3   Apple          Tecnologia             17
  4   Microsoft      Software               26
```

```
  6   Google         Tecnologia            30
  7   Facebook       Tecnologia            35
 10   Telefonica     Comunicaciones        28
Por favor, introduzca el codigo: 1

Lista de estudiantes...
   1  12345678A  Juan Castellano
   2  87654321B  Ana Fuentes
   3  55667788C  Pedro Gonzalez
   4  22334455D  Maria Gomez
   5  12121212F  Carolina Suarez
   6  33443344G  Gregorio Hernandez
   7  23123425Y  Rosa Martin
   8  98765432M  Ana Hernandez
   9  66644666P  Gandalf the Grey
  10  09090909B  Bilbo Baggins
  11  44466444D  Darth Vader
  12  00000000H  Harry Potter
Por favor, introduzca la posicion en la lista: 1
Sueldo: 1954

1 - Insertar un nuevo contrato
2 - Buscar un contrato
3 - Mostrar los contratos
0 - Salir
Elige opcion: 3

Lista de contratos...
Apple          Pedro Gonzalez         1659.40
Amazon         Juan Castellano        1954.00

1 - Insertar un nuevo contrato
2 - Buscar un contrato
3 - Mostrar los contratos
0 - Salir
Elige opcion: 2
NIF del estudiante: 12345678A
Empresa: Amazon - Estudiante: Juan Castellano - Sueldo: 1954.00

1 - Insertar un nuevo contrato
2 - Buscar un contrato
3 - Mostrar los contratos
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
```

Entrega del Examen

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.

Sigue las siguientes instrucciones para la entrega:

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

```
/*
Apellidos y 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.