Facultad de Informática – Universidad Complutense

1º curso de los Grados

Fundamentos de la programación – Grupos A y B

Curso 2017–2018

Examen de junio de 2018

Tiempo disponible: 3 horas

Se desea implementar un juego de tablero consistente en colocar aspersores para regar las plantas de un jardín. El jardín está representado por una cuadrícula en la que cada parcela puede estar vacía u ocupada por una planta. Cada planta tiene una necesidad de riego representada por el número de litros de agua que necesita para no secarse (siempre entre 1 y 8 litros).

El jugador ha de colocar aspersores en las parcelas vacías, según desee, para regar las plantas. Un aspersor riega las ocho parcelas que le rodean, 1 litro a cada parcela. Una vez colocados los aspersores indicados por el usuario, se encienden y se riega. Si una parcela es regada por varios aspersores, recibirá tantos litros de agua como aspersores la rieguen.

```
1. Jugar.
2. Mostrar jugadores.
3. Planta condenada?.
0.Salir.
Introduce una opcion: 1
Introduzca el nombre del jugador: Clara
Introduce el nombre del fichero: jardin.txt
5 0 0 0 0 0 0 3
0 0 0 6 0 0 0 0
0 8 0 0 0 1 0 2
0 0 0 0 0 0 0 0
0 3 0 0 0 5 0 0
Introduzca las coordenadas. Para terminar introduzca -1.
Coordenada: 0 6
Coordenada: 1 6
Coordenada: 1 7
Coordenada: -1
```

A continuación se recopilan los puntos obtenidos por el jugador sumando las siguientes puntuaciones para cada planta del jardín:

- 0 puntos por cada planta que haya recibido menos agua de la que necesita,
- tantos puntos como la necesidad de riego de la planta si ha recibido exactamente la cantidad de agua que necesita,
- por cada planta que ha recibido más agua de la necesaria obtiene (agua necesaria-agua recibida) puntos, es decir, regar excesivamente las plantas resta puntos.

Si la suma de los puntos obtenidos por todas las plantas del jardín es negativa, el jugador obtendrá 0 puntos totales.

Los puntos totales obtenidos por el jugador se incorporan a un listado de jugadores ordenado alfabéticamente por el nombre de los jugadores, de forma que si el jugador ya estaba en el listado se actualiza su puntuación sumando la puntuación total de esta partida, y en caso contrario se añade al listado, manteniendo en todo caso la ordenación indicada.

La aplicación cargará el listado de jugadores del fichero jugadores.txt y mostrará un menú con las siguientes opciones:

1. Jugar: se solicita el nombre del fichero del que se cargará un tablero de juego (el jardín) y el nombre del jugador. Se mostrará el tablero. El jugador indicará las casillas donde desea colocar los aspersores y se recopilarán los puntos de dicho jugador, tras haber regado el jardín, actualizando de manera conveniente la lista de jugadores.
2. Mostrar la lista de jugadores: se ofrecerán dos opciones
    a. Por nombre del jugador: se mostrará la lista de jugadores ordenada por nombre con sus puntos.
    b. Por puntos: se generará una lista de jugadores ordenada por puntos, y a igualdad de puntos por orden alfabético, y se mostrará al usuario.
3. Planta condenada: se solicita el nombre de un fichero del que se carga un tablero de juego, y se indica al usuario si en dicho tablero hay alguna planta condenada a morir porque no hay forma alguna de que se pueda regar con el agua que necesita.

0. Salir. Al salir del juego se guardará en jugadores.txt la lista de jugadores.

El programa consta de 5 módulos: *coordenada*, *jardín*, *listaCoordenadas*, *listaJugadores* y módulo principal.

Módulo *coordenada*:

Este módulo **<u>te lo proporciona el profesor y no se debe modificar</u>**. Contiene la declaración de un tipo `tCoordenada`, formado por dos enteros, fila y columna, que indican una coordenada dentro de la matriz que representa el jardín. Las funciones disponibles están explicadas en el fichero `coordenada.h`. Debes usar esas funciones para tratar las coordenadas y no acceder a su representación.

Módulo *jardín* (4 puntos)

Declara un tipo `tParcela` con información de:

- la cantidad de agua necesaria: 0 representará que la parcela está vacía, y un número entre 1 y 8 que la parcela tiene una planta que necesita ese número de litros de agua.
- la cantidad de agua recibida tras el riego

Declara un tipo `tJardin`, que es una matriz de `tParcela` de dimensión `num_filas * num_columnas` con un tamaño máximo de 50 x 50.

Las operaciones sobre el tipo `tJardin` serán al menos:

- `cargarJardin`: carga desde un fichero la información del jardín. El fichero comienza con una línea con dos valores que indican el número `n` de filas y `m` de columnas de la matriz. En las `n` filas siguientes se dan `m` valores correspondientes al agua necesaria de las parcelas de cada fila (números entre 0 y 8). Se muestra un ejemplo más abajo.

- `mostrarJardin`: muestra la cantidad de agua necesaria de las parcelas del jardín, según la imagen mostrada más arriba.
- `regar`: dada una coordenada, riega desde esa parcela de la matriz.
- `calcularPuntuacion`: devuelve la suma de los puntos obtenidos para cada una de las parcelas con planta del jardín.
- `plantaCondenada`: devuelve un booleano que indica si el jardín tiene alguna planta condenada a morir porque la cantidad de parcelas vacías que la rodean, en las que se podrían colocar aspersores, es menor que la cantidad de agua que necesita. En caso de existir devolverá además la primera coordenada (de arriba abajo y de izquierda a derecha) en la que se encuentre una planta condenada.
- `esLibre`: dada una coordenada y un jardín, devuelve un booleano que indica si esa coordenada está dentro de los límites del jardín y se trata de una parcela vacía.

Módulo *listaCoordenadas* (2 puntos)

Declara una lista `tListaCoordenadas` con información sobre las coordenadas donde se colocan los aspersores. Esta lista estará implementada con un array estático que contiene registros de tipo `tCoordenada`, no necesariamente ordenada y sin repeticiones.

Se implementarán al menos las siguientes operaciones:

- `crearVacia`: Crea una lista vacía de coordenadas.
- `buscar`: Dada una coordenada, devuelve un booleano que indica si dicha coordenada aparece en la lista.
- `insertar`: Dada una lista de coordenadas inserta una nueva coordenada. En caso de que la coordenada ya estuviese en la lista, la operación no tiene efecto y devuelve falso.
- `sacar`: si la lista no es vacía, devuelve la última coordenada de la lista y la elimina; adicionalmente devuelve un booleano que indica si se ha podido ejecutar la operación.

Módulo *listaJugadores* (4 puntos)

Declara un tipo `tJugador` con información sobre su nombre y los puntos que tiene. Declara una lista de jugadores, `tListaJugadores` implementada con un array dinámico de punteros a `tJugador`. Esta lista se mantendrá **ordenada alfabéticamente por nombre del jugador y no puede tener nombres repetidos**.

Se implementarán al menos las siguientes operaciones:

- `cargarJugadores`: carga los jugadores de fichero. En cada línea aparece el nombre del jugador (un `string` sin espacios) y a continuación, separada por un blanco, su puntuación. En el fichero los jugadores están ordenados por nombre. Se muestra un ejemplo más abajo.
- `guardarJugadores:` guarda los jugadores en el fichero en el mismo formato.
- `mostrarJugadores`: muestra una lista de jugadores dada, según se muestra en las imágenes más abajo.
- `buscar`: dado el nombre de un jugador, devuelve un booleano que indica si dicho jugador aparece en la lista. Además, en caso afirmativo devuelve la posición de la lista donde se encuentra; y en caso negativo, la posición donde le tocaría insertarse para mantener el orden indicado. **Implementar esta función de manera recursiva.**

- `actualizarPuntuacion`: dado el nombre de un jugador y unos puntos, si el jugador ya está en la lista actualiza su puntuación sumando los nuevos puntos; en caso contrario inserta el jugador con esos puntos. La lista ha de mantenerse ordenada por nombre de jugador.
- `mostrarPorPuntos`: dada una lista de jugadores ordenada por nombre, genera una nueva lista temporal, ordenada por puntos, y a igualdad de puntos por orden alfabético, que comparte los jugadores con la lista original y la muestra por pantalla. Indicar el método de ordenación utilizado y justificar la elección.
- `liberar`: Libera la memoria dinámica reservada.

Módulo principal

El módulo principal carga los datos de los jugadores en la lista de jugadores, y muestra el menú indicado más arriba. Al finalizar guarda la lista de jugadores. **El profesor te proporcionará un fichero Main.cpp que no has de modificar**.

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.

Ejemplo de archivo de entrada de jugadores: jugadores.txt

```
Alberto 8
Clara 5
Susana 17
Tadeo 8
Victor 2
```

Ejemplo de archivo de entrada de tablero: jardin.txt

```
5 8
5 0 0 0 0 0 0 3
0 0 0 6 0 0 0 0
0 8 0 0 0 1 0 2
0 0 0 0 0 0 0 0
0 3 0 0 0 5 0 0
```

Ejemplos de ejecución

OPCION 1: JUGAR

```
1. Jugar.
2. Mostrar jugadores.
3. Planta condenada?.
0.Salir.
Introduce una opcion: 1
Introduzca el nombre del jugador: Clara
Introduce el nombre del fichero: jardin.txt
5 0 0 0 0 0 0 3
0 0 0 6 0 0 0 0
0 8 0 0 0 1 0 2
0 0 0 0 0 0 0 0
0 3 0 0 0 5 0 0
Introduzca las coordenadas. Para terminar introduzca -1.
Coordenada: 1 0
Coordenada: 1 1
Coordenada: 1 2
Coordenada: 2 0
Coordenada: 2 2
Coordenada: 3 0
Coordenada: 3 1
Coordenada: 3 2
Coordenada: 0 6
Coordenada: 1 6
Coordenada: 1 7
Coordenada: -1
Clara, has conseguido 17 puntos!.
Presione una tecla para continuar . . .
```

OPCION 2-1: MOSTRAR JUGADORES ORDENADOS POR NOMBRE

```
1. Mostrar jugadores ordenados por nombre.
2. Mostrar jugadores ordenados por puntuacion.
0.Volver.
Introduce una opcion: 1
   Alberto   8
     Clara   5
    Susana  17
     Tadeo   8
    Victor   2
Presione una tecla para continuar . . .
```

OPCION 2-2: MOSTRAR JUGADORES ORDENADOS POR PUNTUACION (Y EN CASO DE EMPATE POR ORDEN ALFABETICO)

```
1. Mostrar jugadores ordenados por nombre.
2. Mostrar jugadores ordenados por puntuacion.
0.Volver.
Introduce una opcion: 2
    Susana  17
   Alberto   8
     Tadeo   8
     Clara   5
    Victor   2
Presione una tecla para continuar . . .
```

OPCION 3: DETERMINAR SI EN UN JARDIN HAY UNA PLANTA CONDENADA

```
1. Jugar.
2. Mostrar jugadores.
3. Planta condenada?.
0.Salir.
Introduce una opcion: 3
Introduce el nombre del fichero: jardin.txt
Tiene al menos una planta condenada, en la coordenada: 0 0
Presione una tecla para continuar . . .
```

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.