

Guion de prácticas

MPALABRADOS (tiles-1)

Abril 2020







Metodología de la Programación

DGIM

Curso 2019/2020

Índice

1.	Descripción	5					
2.	Práctica a entregar 2.1. Configuración de la práctica	5 7 7					
	2.3. Entrega de la práctica	7					
3.	Formato de las partidas salvadas	7					
	3.1. Una secuencia de estados a partir de una partida nueva3.2. Una secuencia de estados a partir de una partida existente						



1. Descripción

En esta práctica se va a desarrollar la siguiente capa de la arquitectura, según el plan de trabajo fiijado en el guión de la Práctica 1. En este caso, se va a implementar la primera parte de la clase **Tiles**, según la documentación sobre la misma contenida en el fichero **tiles.h**. Esta nueva capa de la arquitectura nos va a permitir almacenar en memoria, el tablero de letras, pero sin tener en cuenta cruces válidos o salidas fuera de los márgenes del tablero. Este tablero bidimensional se va a almacenar como una matriz bidimensional dinámica, siguiendo cualquiera de los modelos explicados en la clase de teoría. Además de esto, se deberán guardar los datos completos de una partida (puntuación, diccionario, dimensiones,tablero de juego, estado de player y estado de bag) en un fichero, para poder continuar más adelante con ella.

2. Práctica a entregar

Se deberá duplicar el proyecto de Netbeans de la práctica anterior y realizar los siguientes cambios (en todos ellos aparece la marca @warning avisando de las tareas de implementación que están pendientes).

tiles.h

Añadirlo al proyecto recién creado. Añadir también los nuevos métodos de las clases **Move** y **Movelist** que se indican en los ficheros **move.h** y **movelist.h**.

tiles.cpp

Completar la implementación y añadirlo al proyecto. Tengase en cuenta que las coordenadas de la matriz de **Tiles** empiezan en (1,1).

Carpetas tests y data

Eliminar los ficheros de test y data anteriores y susituirlos por los que están en Prado.

main.cpp

Sustituir al anterior y completar el código para realizar el siguiente programa.

- 1. El main() recibe múltiples parámetros de entrada, distinguiendo entre dos modos de funcionamiento.
 - a) Empezar una partida nueva. Para ello los parámetros de llamada serán

```
-l <lang> -w <int> -h <int> -p <pfile> [-r <int> -save <matchfile>]
```

especificando el diccionario, el ancho y alto del tablero de juego, el fichero de movimientos registrados y, opcionalmente, el número aleatorio y la posibilidad de salvar la la jugada en un fichero con extensión .match con la opción -save. En caso de que no se indique esta última opción, entonces deberá mostrar en pantalla el estado final de la partida con el mismo formato (ver Sección 3).



b) Continuar una partida existente. Para ello los parámetros de llamada serán

```
-open <matchfile> -p <playfile> [-save <matchfile>]
```

indicando la apertura de un fichero .match desde el que se restaura el estado anterior de la partida, y un fichero de movimientos. Opcionalmente, se podrá grabar la partida final si se indica el parámetro -save comentado antes.

- 2. Crear una instancia de la clase Language con el ID indicado.
- Crear una instancia de la clase Bag, si es una partida nueva, inicializar la bolsa, en otro caso, cargarla directamente desde el fichero .match
- 4. Crear una instancia de la clase **Player** y inicializarla por completo con caracteres de la bolsa o bien leerla del fichero **.match**.
- 5. Crear una instancia de la clase **Tiles** y dimensionarla según los parámetros leídos, bien desde la línea de comandos, bien desde el fichero .match (atención a la lectura de caracteres multinacionales, según lo comentado en el guión de la Práctica 1).
- 6. Crear una instancia de la clase **Movelist** llamada original y leer todos los movimientos desde el fichero indicado en el parámetro -p usando operador sobrecargado >>
- 7. Crear una instancia de **Movelist** llamada legal que contenga sólo los movimientos de original que están en el diccionario del lenguaje elegido. Usar, para ello, el método zip(...)
- 8. Crear dos instancias adicionales de **Movelist** y llamarlas accepted y rejected
- 9. Recorrer toda la lista de movimientos leída y, por cada uno de ellos
 - a) Si el movimiento está en el diccionario, añadir la palabra a la lista accepted, calcular su puntuación, según el idioma y acumularla. A continuación, se deberá colocar cada movimiento en su posición correspondiente en la instancia de Tiles creada anteriormente, ignorando aquellos caracteres que caigan fuera de las dimensiones permitidas y sobreescribiendo los caracteres que se crucen. En este caso, y a partir de ahora, se deben respetar los datos sobre la fila y la columna de cada movimiento leído y si es horizontal o vertical, a la hora de introducirlos en tiles.
 - b) En otro caso añadirla a la lista rejected.
- 10. Terminar mostrando el estado de la partida en pantalla o guardándolo en disco según la presencia o no de -save.
- 11. Si en cualquier momento se presenta un error en los argumentos, en la apertura de ficheros o en la lectura de datos del fichero, se debe usar la función errorBreak(...) para notificar el error y parar el programa.



2.1. Configuración de la práctica

La misma que en la práctica anterior. Asegurarse de tener actualizadas las scripts de NetBeans.

2.2. Validación de la práctica

Se debe ejecutar la script **doTests.sh** y comprobar que los resultados que aparecen por pantalla están en verde.

2.3. Entrega de la práctica

Se deberá ejecutar la script **doZipProject.sh** y subir a Prado, en las fechas que se indican en la temporización de la asignatura, el zip resultante, que está almacenado en la carpeta **.zip**/ del proyecto de Netbeans y siempre se llama **MPPractica.zip**.

3. Formato de las partidas salvadas

A la hora de mostrar el resultado final de la partida, se deberá seguir la siguiente estructura:

- La primera línea, que no se muestra en las figuras de esta sección, deberá ser MPALABRADOS-V1 cuando se escriba en disco, o bien %%0UTPUT cuando se escriba en pantalla.
- 2. Un número entero que representa la puntuación acumulada hasta el momento.
- 3. La descripción del diccionario según las normas estándar (ES, EN, FR).
- 4. Número de filas y columnas, en ese orden, del tablero de juego.



- 5. Estado del tablero de juego, mostrando las letras que se han colocado en cada tirada o el carácter '.' para indicar que esa posición está vacía.
- 6. Número de componentes de Player y su estado.
- 7. Número de componentes de **Bag** y su estado.

Una secuencia de estados a partir de una partida 3.1. nueva

Fichero de movimientos

v 1 1 ABANA v 1 2 EXISTE v 1 5 AYUNAS H 3 1 SR H 4 1 TUTE h 9 1 @

Llamada al programa

 $\verb|mp1920| \verb|practica5 -l ES -r 2020 -w 15 -h 15 -p data/ES_2020_mini.data -save data/mini.match| \\$

Secuencia de estados



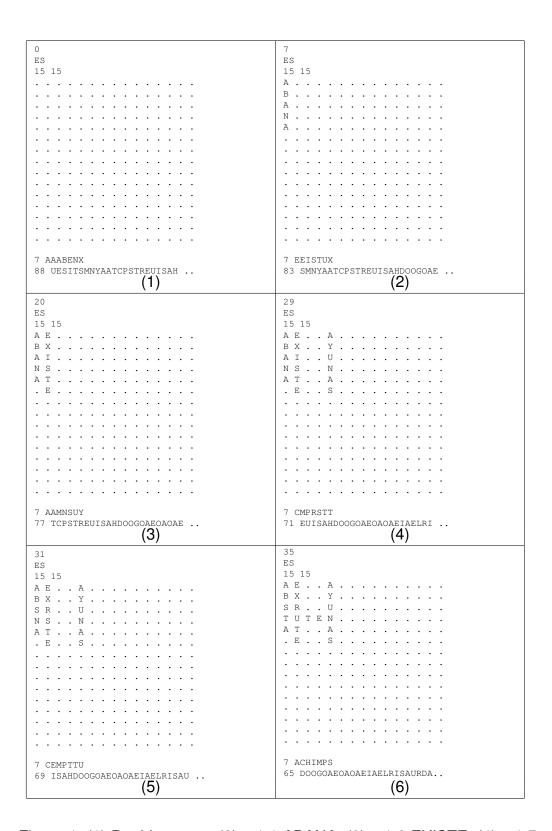


Figura 1: (1) Partida nueva, (2) v 1 1 ABANA, (3) v 1 2 EXISTE, (4) v 1 5 AYUNAS, (5) H 3 1 SR (6) H 4 1 TUTE



Una secuencia de estados a partir de una partida existente

Fichero de movimientos

H 10 1 CHISPA v 1 7 DOGMA H 8 2 EA H 9 1 OIA h 1 1 OLEO h 9 1 @

Llamada al programa, cargando la partida anterior

mp1920practica5 -open data/mini.match -p data/ES_2020_mini2.data

Secuencia de estados



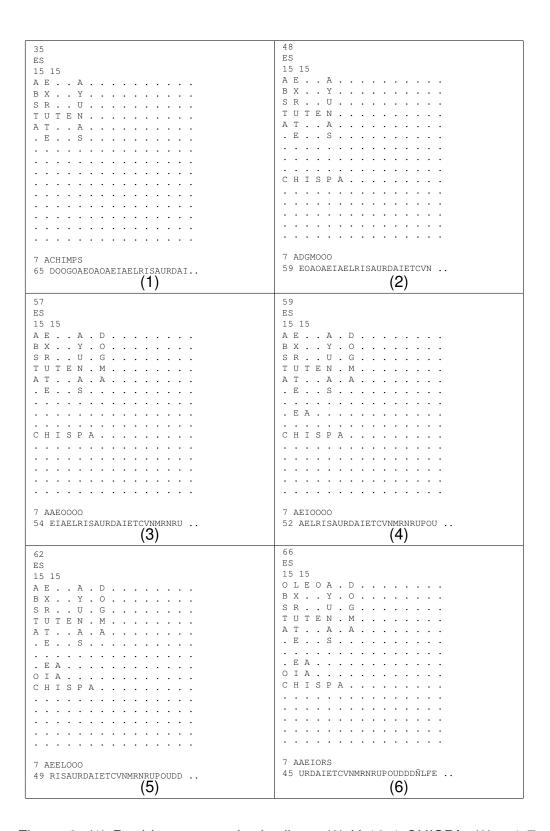


Figura 2: (1) Partida recuperada de disco, (2) H 10 1 CHISPA, (3) v 1 7 DOGMA, (4) H 8 2 EA, (5) H 9 1 OIA, (6) h 1 1 OLEO