



Guion de prácticas

MPALABRADOS (tiles-1)

Abril 2020



Metodología de la Programación

DGIM

Curso 2019/2020

Índice

1. Descripción	5
2. Práctica a entregar	5
2.1. Configuración de la práctica	7
2.2. Validación de la práctica	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 nueva . .	8
3.2. Una secuencia de estados a partir de una partida existente	10

1. Descripción

En esta práctica se va a desarrollar la siguiente capa de la arquitectura, según el plan de trabajo fijado 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 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.
3. 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.

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.

3.1. Una secuencia de estados a partir de una partida 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

```
mp1920practica5 -l ES -r 2020 -w 15 -h 15 -p data/ES_2020_mini.data -save data/mini.match
```

Secuencia de estados

<p>0 ES 15 15</p> <p>7 AAABENX 88 UESITSMNYAATCPSTREUISAH ..</p> <p>(1)</p>	<p>7 ES 15 15</p> <p>7 EEISTUX 83 SMNYAATCPSTREUISAHDOOGOAE ..</p> <p>(2)</p>
<p>20 ES 15 15</p> <p>A E B X A I N S A T E</p> <p>7 AAMNSUY 77 TCPSTREUISAHDOOGOAEEOAOAE ..</p> <p>(3)</p>	<p>29 ES 15 15</p> <p>A E . . A B X . . Y A I . . U N S . . N A T . . A E . . S</p> <p>7 CMPRSTT 71 EUISAHDOOGOAEIOAOEIAELRI ..</p> <p>(4)</p>
<p>31 ES 15 15</p> <p>A E . . A B X . . Y S R . . U N S . . N A T . . A E . . S</p> <p>7 CEMPTTU 69 ISAHDOOGOAEIOAOEIAELRISAU ..</p> <p>(5)</p>	<p>35 ES 15 15</p> <p>A E . . A B X . . Y S R . . U T U T E N A T . . A E . . S</p> <p>7 ACHIMPS 65 DOOGOAEIOAOEIAELRISAURDA..</p> <p>(6)</p>

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

3.2. 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

```
mpl920practica5 -open data/mini.match -p data/ES_2020_mini2.data
```

Secuencia de estados

<div>35 ES 15 15 A E . . A B X . . Y S R . . U T U T E N A T . . A E . . S .</div> <div>7 ACHIMPS 65 DOOGOAEIOAOEIAELRISAURDAI..</div>	<div>48 ES 15 15 A E . . A B X . . Y S R . . U T U T E N A T . . A E . . S . C H I S P A .</div> <div>7 ADGMOOO 59 EOAOEIAELRISAURDAIETCVN ..</div>
(1)	(2)
<div>57 ES 15 15 A E . . A . D B X . . Y . O S R . . U . G T U T E N . M A T . . A . A E . . S . C H I S P A .</div> <div>7 AAEOOOO 54 EIAELRISAURDAIETCVNMNRNU ..</div>	<div>59 ES 15 15 A E . . A . D B X . . Y . O S R . . U . G T U T E N . M A T . . A . A E . . S E A C H I S P A .</div> <div>7 AEIOOOO 52 AELRISAURDAIETCVNMNRNUPOU ..</div>
(3)	(4)
<div>62 ES 15 15 A E . . A . D B X . . Y . O S R . . U . G T U T E N . M A T . . A . A E . . S E A O I A C H I S P A .</div> <div>7 AEELOOO 49 RISAUARDAIETCVNMNRNUPODDD ..</div>	<div>66 ES 15 15 O L E O A . D B X . . Y . O S R . . U . G T U T E N . M A T . . A . A E . . S E A O I A C H I S P A .</div> <div>7 AAEIORS 45 URDAIETCVNMNRNUPODDDÑLFE ..</div>
(5)	(6)

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