FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Algoritmos y Programación I

Curso Mendez

TP2 - Buscando la Copa

Fecha Presentación	25/04/2019
Fecha Entrega	30/05/2019

Introducción

El **Torneo de los Tres Magos** es un concurso mágico celebrado entre los tres principales colegios de magia de Europa: el Colegio Hogwarts de Magia y Hechicería, el Instituto Durmstrang y la Academia de Magia Beauxbatons, cada escuela representada por un **campeón**. Los campeones seleccionados compiten en tres pruebas -tradicionalmente juzgados por los directores o directoras de las escuelas participantes- diseñadas para probar la capacidad mágica, la inteligencia y el coraje. Los campeones compiten por el honor y la gloria de ganar el Torneo, por la Copa de los Tres Magos y por un premio monetario.

Objetivo

El presente trabajo práctico tiene como objetivo que el alumno:

- Diseñe y desarrolle las funcionalidades de una biblioteca con un contrato preestablecido.
- Se familiarice con y utilice correctamente los tipos de datos estructurados.
- Desarrolle una interfaz gráfica amigable y entendible para el usuario.

Por supuesto, se requiere que el trabajo cumpla con las buenas prácticas de programación profesadas por la cátedra. Se considerarán críticos la modularización y reutilización de código y la claridad del código.

Enunciado

En esta oportunidad se deberá desarrollar un juego que represente la 3er prueba del Torneo de los 3 Magos a la cual se enfrentaron los campeones.

Dicho juego deberá permitir a un jugador moverse dentro de un laberinto con el objetivo de alcanzar la Copa, hecho que dará por finalizado el juego.

Dentro del mismo laberinto se encontrarán obstáculos que harán más difícil la travesía y ayudas que intentarán allanar el camino.

Por otro lado y con el mismo fin que el jugador, se encontrará otro jugador intentando alcanzar la Copa, este jugador no será controlado por el usuario y tendrá movimientos precalculados que se explicarán en la sección correspondiente.

A continuación se explicarán cada uno de los elementos que forman parte del juego:

Laberinto

Será representado como una matriz. Como todos sabemos, los laberintos están compuestos de pasillos y paredes. Desde la cátedra se les proveerá un procedimiento que, enviándole una matriz, la devolverá cargada con pasillos y paredes. Sirviendo ésta como laberinto para nuestro juego.

Jugador

Será nuestro personaje, deberá moverse por los pasillos del laberinto buscando la Copa, será controlado por el usuario y no puede atravesar las paredes del laberinto. Empieza con 50 puntos de vida. Cada paso le resta 3 puntos de vida. La vida no puede ser superior a 50, es decir si tiene 40 puntos de vida y agarra una poción, llegará a 50 pero no a 55.

Obstáculos

Existen 3 tipos de obstáculos, de los cuales solo habrá **uno de cada uno** dentro del laberinto.

- **Escreguto de cola explosiva**: Parecidos a escorpiones gigantes usan sus explosiones para acercarse al enemigo. **Quita 20 de vida si no tienes impedimenta**.
- **Acromántula**: Es una especie mágica de monstruosas y gigantes arañas que tienen ocho patas, ocho ojos, la capacidad de hablar y tienen también unos colmillos que poseen veneno muy tóxico. **Quita 10 de vida**.

- **Boggart**: Es un no-ser amortal de forma cambiante capaz de transformarse en la imagen de lo que más teme su espectador. **Quita 15 de vida si no tienes riddikulus**.

Ayudas

Existen 2 tipos de ayudas, por un lado los embrujos y encantamientos que pueden aprenderse y por otro las pociones que nos recobran vida, dentro del laberinto hay 3 hechizos, **uno de cada uno**, y 3 pociones.

- **Impedimenta**: Es un embrujo que impide el movimiento hacia adelante, bien ralentizando su movimiento, o inmovilizando por completo durante un corto periodo de tiempo. Si se aprende, el **escreguto de cola explosiva** no nos dañará.
- **Riddikulus**: Es un encantamiento que se utiliza para defenderse contra un Boggart. Hace que la criatura asuma una forma que sea chistosa, contrarrestando así su capacidad para aterrorizar. Si se aprende, el **boggart** no nos dañará.
- **Esfinge**: Es un elemento que permite deshechizar la copa, la cual se encuentra oculta hasta el momento de encontrar la esfinge.
- **Poción**: Cada poción nos recobra 15 puntos de vida.

Rival

Existe otro competidor con el mismo objetivo que nosotros, alcanzar la copa. Este jugador se moverá de manera automática y predecible. Su turno será luego del nuestro. Este competidor no es afectado por las ayudas ni los obstáculos.

Especificaciones

De las convenciones

- Se representarán los elementos de la siguiente manera:
    - Jugador: J.
    - Rival: G.
    - Copa: C.
    - Pociones: P.
    - Impedimenta: I.
    - Riddikulus: R.
    - Escreguto: E.
    - Acromántula: A.
    - Boggart: B.
    - Esfinge: F.
- Las teclas para los movimientos del jugador serán:
    - Izquierda: a.
    - Abajo: s.
    - Derecha: d.
    - Arriba: w.
- La dimensión del laberinto será de 15x15.
- La matriz generada por la cátedra tendrá '#' donde haya pared y ' ' donde haya pasillo.
- Se deberá usa **snake_case** para el desarrollo del trabajo.

De las estructuras de datos

- Serán provistas por la cátedra.
- Podrán ser utilizadas todas las estructuras que el alumno considere, siempre que respete la correcta actualización de las que la cátedra provee, ya que las pruebas se realizarán sobre dichas estructuras.
- Se vuelve necesario mantener en estructuras separadas el laberinto original y las posiciones de los elementos, del laberinto mostrado al usuario. Esto significa que a cada turno, el laberinto mostrado se actualizará con la información del laberinto original y los elementos en la posición actual.
- En este sentido, es convención determinada por la cátedra que en el laberinto serán representadas las paredes con una '#' y los pasillos con un ' ', sin embargo, en la visualización pueden mostrarse de otro modo ya que son matrices distintas.

De las ayudas y los obstáculos

- Las ayudas y obstáculos deberán desaparecer luego de ser *pisados*, esto implica que deberán ser eliminados del vector donde están almacenados, si es que no tienen acción inmediata (Impedimenta y Riddikulus) deben ser pasados al vector de ayudas del jugador.

De la visualización del juego

- Debe visualizarse como una matriz.
- En caso de posicionarse el jugador sobre cualquier otro elemento, se mostrará el jugador.
- En caso de posicionarse el rival sobre algún elemento (salvo el jugador) se mostrará el rival.
- El jugador y el rival pueden cruzarse, eso no implica ningún problema.
- La copa no será mostrada hasta que se agarre la esfinge o al jugador le queden 15 puntos de vida o menos.

Del posicionamiento de los elementos

- Las posiciones de la copa, obstáculos, ayuda, jugador y rival serán aleatorios.
- No puede haber más de un elemento inicializado en la misma posición.
- El orden de posicionamiento será: **Copa - Escreguto - Acromántula - Boggart - Impedimenta - Riddikulus - Pociones - Esfinge - Rival - Jugador**.
- El Rival y el Jugador deben ser posicionados a una distancia (ver sección **distancia Manhattan**) mayor a 10 de la Copa.
- Ninguno de los elementos puede ser posicionado sobre una pared.

Del orden de los movimientos

- El primero en moverse será el jugador, luego el rival.
- En caso de que el juego termine, ya sea porque el jugador alcanza la copa o se le terminan los puntos de vida, el rival no se moverá.
- En caso de que el rival alcance la copa, el jugador no podrá moverse ya que el juego finalizará.
- Cada turno resta 3 puntos de vida.
- El orden de manejo de puntos de vida es, primero se resta el movimiento, después se consume la poción, es decir que en caso de estar al límite de la vida y tener al siguiente paso una poción, no se llegará a agarrarla ya que no se tendrá fuerzas.

De la generación del laberinto

- La cátedra brindará un procedimiento que cargará en una matriz recibida por parámetro las paredes y pasillos correspondientes al laberinto.
- Es imprescindible **no** asumir que es el único laberinto en el que debe funcionar correctamente el trabajo ya que las pruebas se realizarán con otro laberinto.
- Se deberá incluir la biblioteca **laberinto.h** provista por la cátedra, la cual cuenta con un procedimiento público llamado **inicializar_paredes_laberinto** que deberá ser invocado antes de posicionar todos los demás elementos ya que sus posiciones dependen de las paredes del laberinto.

De los movimientos del rival

- El rival se moverá siempre después del jugador.
- Su movimiento será automático y seguirá el siguiente ciclo:
    - 4 turnos hacia la derecha.
    - 4 turnos hacia abajo.
    - 4 turnos hacia la izquierda.
    - 4 turnos hacia arriba.
    - Luego vuelve a la derecha y así sucesivamente.
- El rival siempre deberá moverse, en caso de que tenga que ir a la derecha y no pueda, volverá a intentar moverse, es decir, probará 4 veces a la derecha, luego 4 hacia abajo y así hasta que pueda moverse.

Biblioteca copa.h

Se deberán utilizar las estructuras propuestas y no podrán ser modificadas:

```
typedef struct coordenada {
	int fil;
	int col;
} coordenada_t;

typedef struct obstaculo {
	char codigo;
	coordenada_t posicion;
	int danio;
} obstaculo_t;

typedef struct ayuda {
	char codigo;
	coordenada_t posicion;
	int vida_a_recuperar;
} ayuda_t;

typedef struct jugador {
	int vida;
	coordenada_t posicion;
	int tope_ayudas;
	ayuda_t ayudas[TOTAL_AYUDAS];
} jugador_t;

typedef struct rival {
	coordenada_t posicion;
	char direccion;
	int cantidad_pasos;
} rival_t;

typedef struct copa {
	char codigo;
	coordenada_t posicion;
} copa_t;

typedef struct juego{
	char laberinto_original[TAMANIO][TAMANIO];
	jugador_t jugador;
	rival_t rival;
	copa_t copa;
	obstaculo_t obstaculos[TOTAL_OBSTACULOS];
	int tope_obstaculos;
	ayuda_t ayudas[TOTAL_AYUDAS];
	int tope_ayudas;
} juego_t;
```

Se deberán implementar las siguientes funciones, utilizando las estructuras propuestas y cumpliendo con las pre y post condiciones enunciadas:

```
/*
 * Inicializará todas las estructuras con los valores correspondientes,
 * creará el laberinto, posicionará todos los elementos, etc.
 */
void inicializar_laberinto(juego_t* juego);

/*
 * Determinará si el caracter ingresado es válido, esto es, es el caracter 'a' o
 * 's' o 'd' o 'w' y además el jugador puede moverse en esa dirección, o sea,
 * hay pasillo.
 */
bool es_movimiento_valido(juego_t* juego, char tecla);

/*
 * Moverá el jugador hacia la dirección especificada.
 * Dicho movimiento es válido.
 */
void mover_jugador(jugador_t* jugador, char direccion);

/*
 * Moverá el rival a la próxima posición.
 */
void mover_rival(juego_t* juego);

/*
 * Actualizará el juego. Restará vida si el jugador está sobre un obstáculo
 * o lo eliminará si cuenta con el hechizo, aprenderá hechizos y todo lo
 * que pueda suceder luego de un turno.
 */
void actualizar_juego(juego_t* juego);

/*
 * Devolverá el estado del juego, 1 ganado, 0 en curso, -1 perdido.
 */
int estado_juego(juego_t juego);

/*
 * Devolverá una coordenada aleatoria dentro del rango TAMANIOxTAMANIO.
 * No valida que dicha coordenada coincida con un pasillo ni que exista
 * otro objeto en esa posición.
 */
coordenada_t posicion_aleatoria();
```

```c
/*
 * Actualizará la matriz mostrada al usuario, con los elementos presentes
 * en el juego.
 */
void actualizar_laberinto(juego_t juego, char laberinto[TAMANIO][TAMANIO]);

/*
 * Mostrará el laberinto por pantalla.
 */
void mostrar_laberinto(char laberinto[TAMANIO][TAMANIO]);
```

Resultado Esperado

El trabajo creado debe:

- Interactuar con el usuario.
- Mostrarle al usuario la información del juego a cada momento.
- Informarle al usuario correctamente cualquier dato que haya sido ingresado incorrectamente.
- Informarle al usuario si ganó o perdió.
- Cumplir con las buenas prácticas de programación.
- Mantener las estructuras propuestas actualizadas a cada momento.

Compilación y Entrega

El trabajo práctico debe ser realizado en un archivo llamado **juego.c**, y la biblioteca de funciones para jugarlo **copa.c** y **copa.h**, y debe poder ser compilado sin errores con el comando:

```
gcc juego.c copa.c laberinto.o -o juego -std=c99 -Wall -Wconversion -Werror -lm
```

laberinto.o es el archivo generado por la compilación del generador del laberinto. De esta manera, confiando en el contrato de **laberinto.h** pueden afirmar que todas las funciones allí presentes funcionarán como dicen sin necesidad de ver el código (caja negra).

Para la generación del laberinto deberán invocar a la función:

```c
inicializar_paredes_laberinto(laberinto);
```

Siendo **laberinto** del tipo:

```c
char laberinto[TAMANIO][TAMANIO]
```

Por último debe ser entregado en la plataforma de corrección de trabajos prácticos Kwyjibo en la cual deberá aparecer con la etiqueta successful.
Para la entrega en Kwyjibo, recuerde que deberá subir un archivo zip conteniendo únicamente los archivos antes mencionados, sin carpetas internas ni otros archivos. De lo contrario, la entrega no será validada por la plataforma.

Anexos

Distancia Manhattan

Para obtener la distancia entre 2 puntos mediante este método, se debe conocer a priori las coordenadas de dichos puntos.
Luego, la distancia entre ellos es la suma de los valores absolutos de las diferencias de las coordenadas.
Se ve claramente en los siguientes ejemplos:

- La distancia entre los puntos (0,0) y (1,1) es 2 ya que:
  |0 - 1| + |0 - 1| = 1 + 1 = 2
- La distancia entre los puntos (10,5) y (2,12) es 15 ya que:
  |10 - 2| + |5 - 12| = 8 + 7 = 15
- La distancia entre los puntos (7,8) y (9,8) es 2 ya que:
  |7 - 9| + |8 - 8| = 2 + 0 = 2

Bibliografía

http://es.albusspotter.wikia.com/wiki/Torneo_de_los_Tres_Magos