ALGORITMOS Y ESTRUCTURAS DE DATOS 2023/1
Proyecto Unidad 1

PROFESOR: Manuel Alejandro Moscoso Domínguez
manuel.moscoso.d@gmail.com

El presente documento entrega la descripción del escenario relacionado con el problema a resolver de la Unidad 1.

Objetivos

- Realizar el análisis de un problema y aplicar en el diseño de una solución estructuras de datos lineales, tipos de datos abstractos y las operaciones comunes sobre estas.
- Desarrollar algoritmos que permitan entregar una solución a los problemas entregados.

Requerimientos

El Software desarrollado debe cumplir con los siguientes requerimientos:

- El programa debe estar completamente funcional, con ausencia de errores y alertas (warnings).
- El programa debe estar bien documentado para lo cual se necesita:
    - La inclusión de un archivo README donde se explica el problema y la solución.
    - Presencia de comentarios que expliquen el código (descripción de estructura y funciones).
    - La inclusión de un archivo INSTALL donde se explica el procedimiento para instalar el programa (compilar).
- El programa debe compilar con ausencia de errores.
- El programa debe ser implementado a través del uso del lenguaje de programación C.

The Guardians Tournament

En un mundo devastado por la guerra y la lucha constante, donde la magia es real, existen los Guardianes. Los Guardianes son héroes con habilidades y destrezas únicas que protegen a las aldeas a lo largo de toda la tierra siendo estas los últimos asentamientos de la humanidad.

Todos los años se realiza el torneo de los guardianes donde se busca encontrar al campeón quien gozará del reconocimiento y la gloría eterna.

El juego se basa en la posibilidad de ganar el torneo para lo cual se debe considerar:

- Existen 4 tipos de guardianes; magos, vikingos, nigromantes y bestias.
- Los guardianes tienen un nombre, tipo, cantidad de vida, puntos de ataque y puntos de defensa.
- Los combates se generan por turnos y en cada turno se puede realizar una o dos acciones según la preferencia del usuario y el resultado del lanzamiento de un dado. Los posibles resultados son:
    - Si el usuario quiere atacar y obtiene de los dados los números 1, 3 o 5:
        - factor 0.8 en el caso de obtener el número 2.
        - factor 1 en caso de obtener el 4.
        - factor 1.3 en el caso de obtener 6.
        - Un número par bloquea significa que fue bloqueado el ataque por el rival.
    - Si el usuario quiere defender y obtiene de los dados los números 2, 4 o 6: Se realiza una defensa considerando.
        - factor 0.5 en el caso de obtener el número 2.
        - factor 1 en caso de obtener el 4.
        - factor 1.2 en el caso de obtener 6.
        - Un número impar significa la aplicación del desgaste sobre los puntos de defensa.
    - Para el caso de obtener el número 6 se puede volver a lanzar. Solo se puede acumular una acción adicional.

- Al defender lo que se realiza es aumentar la vida lo cual se realiza a través de la multiplicación del factor obtenido por el dado por los puntos de defensa.
- Al atacar lo que se realiza es restar a la vida del rival considerando para restar la multiplicación del factor obtenido por el dado por los puntos de ataque.
- El torneo consiste en combatir para derrotar a todos los contendientes para lo cual existen tres niveles:
    - Principiante: Derrotar a tres guardianes.
    - Intermedio: Derrotar a cinco guardianes.
    - Experto: Derrotar a siete guardianes.

Para poder realizar la implementación de esta solución se necesitan realizar considerar:

- Dentro de la solución deben existir los tipos de datos que permitan almacenar la información de los guardianes y los torneos.
- La información de los guardianes debe ser leída desde un archivo, se adjunta una muestra de ejemplo.
- El jugador puede crear su propio guardián para lo cual solo puede definir el nombre y el tipo. Los puntos deben ser obtenidos de manera aleatoria para lo cual se deben considerar los mínimos y máximos de los guardianes que se cargan del archivo.
- Un jugador puede seleccionar un guardián. La aparición de un guardián es única por lo que esto se debe considerar al momento de generar el torneo.
- Un jugador debe tener la posibilidad de poder seleccionar el nivel en el cual quiere participar.
- Los guardianes para cada torneo son de manera aleatoria y no se pueden repetir.
- En caso que el jugador esté interesado en poder continuar avanzando en la dificultad, esto debe considerar que no se deben volver a repetir guardianes en los otros torneos.
- Cada guardián tiene un desgaste durante una pelea lo que corresponde a la pérdida de puntos de defensa. El desgaste durante una defensa incompleta para lo cual se resta cada vez que ocupa la defensa en un 5% del puntaje llegando a un mínimo de 30.

A continuación se detallan algunas funciones (puede considerar la inclusión de más funciones según corresponda) que deben estar dentro de la solución:

- **menuOptions**: Función que entrega la posibilidad de seleccionar la opción para el o los jugadores.
- **selectCharacter**: Función que entrega la posibilidad de seleccionar un personaje por un jugador.
- **createCharacter**: Función que entrega la posibilidad de crear un personaje para el jugador.
- **selectTournament**: Función que entrega la posibilidad de seleccionar el nivel de dificultad del torneo.
- **printCharacterStatus**: Función que permiten saber el estado de cada personaje o guardian.
- **startFight**: Función que inicia la pelea entre el personaje del jugador y el guardián.
- **getRollResult**: Función que permite saber el resultado de un lanzamiento de los dados.
- **getResult**: Función que permite conocer el historial del jugador una vez finalizada su participación en un torneo.

Para completar y resolver la actividad se requiere la utilización de listas, pilas/colas además de la entrada/salida de archivos y menús interactivos en un contexto práctico relacionado con el desarrollo de videojuegos.

Adicionales a poder implementar

Se considerará como característica adicionales a ser entregadas dentro del proyecto lo siguiente:

- El jugador debe tener la posibilidad de poder guardar y cargar las partidas. Al guardar una partida se debe considerar el nivel, posición de avance e historial de encuentros sostenidos con otros guardianes.

Con esto las funciones de:

- saveGame: Función que permite guardar el estado del juego para volver a comenzar.
- loadGame: Función que permite cargar el estado de un juego para continuar jugando.

Detalle de ponderación de la solución

Ejercicio	Indicadores	Ponderación
1	Desarrolla la declaración de estructura(s) para trabajar según la problemática.	10%
2	Desarrolla la función que entrega la posibilidad de realizar la carga inicial de personajes	5%
3	Desarrolla la función que entrega la posibilidad de seleccionar la opción para el o los jugadores.	3%
4	Desarrolla la función que entrega la posibilidad de seleccionar un personaje por un jugador.	5%
5	Desarrolla la función que entrega la posibilidad de crear un personaje para el jugador.	10%
6	Desarrolla la función que entrega la posibilidad de seleccionar el nivel de dificultad del torneo.	10%
7	Desarrolla la función que permite saber el estado de cada personaje o guardián.	5%
8	Desarrolla la función que inicia la pelea entre el personaje del jugador y el guardián.	10%
9	Desarrolla la función que permite saber el resultado de un lanzamiento de los dados.	5%
10	Desarrolla la implementación de listas, cola o pila para almacenar la información de la problemática.	10%
11	Desarrolla la implementación del algoritmo para poder competir y avanzar en el torneo.	10%
12	Desarrolla la función que permite conocer el historial del jugador una vez finalizada su participación en un torneo.	5%
13	Desarrolla la implementación de la gestión correcta de liberación de memoria	2%
14	Desarrolla una solución con ausencia de errores y alertas.	3%
15	Desarrolla de manera correcta comentarios en la estructura(s) y funciones.	3%
16	Desarrolla de manera adecuada el contenido del README	2%
17	Desarrolla de manera adecuada el contenido del INSTALL	2%
	Total	100%
	Desarrolla la funcionalidad para guardar una partida.	5%
	Desarrolla la funcionalidad para cargar una partida.	5%
	Desarrolla la actualización de los algoritmos para soportar guardar y cargar las partidas.	5%

Contenido de prueba para los Guardianes

```
Name,Type,Health Points,Attack Points,Defense Points
Aria,mage,480,180,50
Axl,beast,560,135,90
Bjorn,viking,530,110,75
Calantha,mage,440,160,35
Cassius,nigromante,430,180,60
Dalia,beast,510,120,80
Einar,viking,580,140,85
Eris,mage,500,190,40
Gideon,nigromante,480,200,65
Gunnar,viking,570,130,70
Jorgen,beast,590,140,95
Kael,mage,470,170,45
Kai,nigromante,410,190,55
Lena,beast,540,125,85
Loki,viking,510,115,60
Mara,mage,450,150,50
Nyx,nigromante,420,180,70
Odin,viking,550,120,80
Orrin,beast,520,130,90
Raven,mage,490,160,55
Rune,viking,590,150,95
Seth,nigromante,450,200,75
Skadi,beast,570,135,80
Theodora,mage,430,170,30
Valdis,viking,560,125,90
Zephyr,nigromante,400,200,50
```