ALGORITMOS Y ESTRUCTURAS DE DATOS
2023/2

Proyecto 1

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 una solución considerando estructuras de datos lineales y las operaciones comunes sobre ellas.
- Realizar la solución utilizando el lenguaje de **programación C.**
- Realizar una correcta declaración de variables según corresponda.
- Realizar una correcta implementación de algoritmos que involucren decisiones y bucles.
- Realizar una correcta implementación de entrega de resultados por la salida estándar.

Requerimientos de entrega

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 la solución desarrollada considerando el formato recomendado para este tipo de archivos. [1]
    - Presencia de comentarios que expliquen el código, considerando la descripción de estructura, clases, métodos y funciones.
- La inclusión de un archivo INSTALL donde se explica el procedimiento para instalar el programa (compilar).
- El programa debe encontrarse en un repositorio en su cuenta Github.

El programa debe ser implementado a través del uso del lenguaje de programación C la utilización de listas, pilas y/o colas además de la entrada/salida de archivos y menús interactivos en un contexto práctico relacionado con el problema entregado.

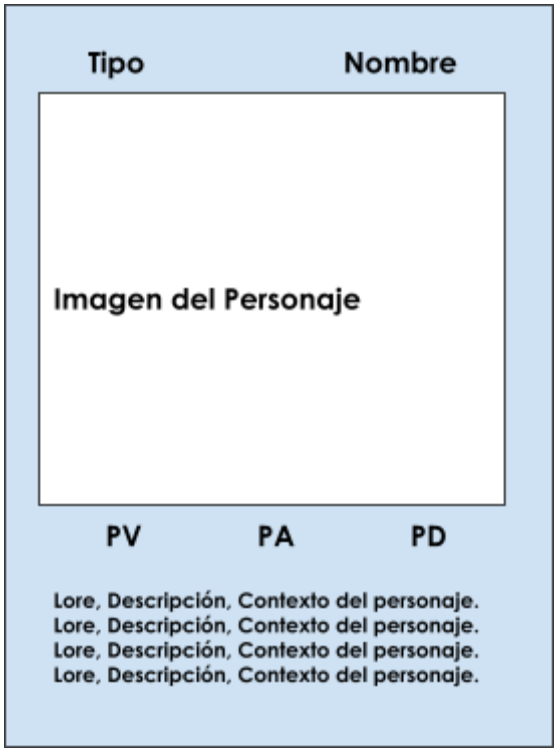
[1] Formato README

The clash of the guardians

Lircay Hub está interesado en desarrollar un piloto de un juego para lo cual es importante realizar la validación de algunos conceptos asociados a este. El juego se llama “The clash of the guardians” que corresponde a un juego de cartas donde 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 un torneo en los cuales por equipo los guardianes pelean en búsqueda de los campeones quienes gozarán del reconocimiento y la gloría eterna.

Las cartas del juego tiene las siguientes características:



- Existen 60 o más cartas distintas.
- Existen 4 tipos de guardianes; magos, vikingos, nigromantes y bestias.
- Todos los guardianes tienen un nombre que solo es una palabra (sin nombres compuestos).
- Todos los guardianes tienen:
    - Puntos de vida (PV): Representa la salud o la vida.
    - Puntos de Ataque (PA): Representa el daño que puede realizar en combate.
    - Puntos de defensa (PD): Representa la capacidad de reducir el daño recibido.

Respecto a las instrucciones del juego, se puede comentar que:

- Cada jugador recibe un lote de 15 cartas, entregadas luego de revolver, de las cuales debe seleccionar tres cartas para comenzar el juego las cuales quedan en su mano.
- Las 12 cartas restantes corresponden a un monto, que debe devolver y dejarlo al costado.
- Cada jugador parte con 5 puntos de vida. Los puntos de vida solo se restan cuando uno de los guardianes (las cartas) es derrotado.
- En cada turno los jugadores sacan una carta y además pueden:
    - Seleccionar una carta y dejarla en el campo de batalla.
    - Atacar a un oponente con una de las cartas que se encuentra en el campo de batalla.

- Cuando se produce una batalla, en otros palabras un ataque, se debe considerar:
    - Calcular el daño. El atacante realiza el daño definido por sus puntos de ataque.
    - Se restan directamente los puntos de daño de los puntos de defensa.
    - Si la diferencia es 0 o menos, es derrotado el guardián correspondiente a un punto menos de vida del jugador.
- El juego continúa hasta que uno de los dos jugadores alcanza 0 puntos. siendo el ganador el otro jugador.

El objetivo es crear un **programa en C** que permita realizar la simulación de una partida del juego señalado en la sección anterior. El programa debe contener:

- Dentro de la solución deben existir los tipos de datos que permitan almacenar la información propia del juego.
- Al iniciar el juego se debe realizar la carga de información desde un archivo, se adjunta una muestra de ejemplo.
- Al iniciar se debe presentar un menú que permita crear una nueva carta, comenzar el juego, historial de la partida, salir.
- Se pueden crear nuevos datos para las pruebas. Aquí se deben entregar al usuario la posibilidad de definir todos los atributos propios del dominio y validar que estos no excedan los provenientes desde el archivo. Dentro de la validación se deben considerar los mínimos y máximos que se cargan del archivo.
- El programa debe actuar como jugador para lo cual se debe crear la función que de manera aleatoria selecciones las acciones durante su turno. En este caso con una mayor probabilidad de atacar más que solo agregar cartas a la mesa.
- El programa, al momento de atacar, debe intentar ganar por lo que se debe considerar que debe atacar al guardián con mayor probabilidad de ser vencido.
- Al finalizar la partida se debe entregar el ganador con el resumen de los puntos de vida que le quedan.
- Una vez entregada esta información se vuelve al menú inicial y al seleccionar la opción “historial de la partida” mostrará dicha información.

La entrega del proyecto debe ser acompañada por un video, de no más de 5 minutos en los que se debe:

- Desarrollar un material de apoyo (presentación) que sirva como guía para el video. Debe incorporar la portal además de:
    - Mencionar el problema a resolver.
    - Exponer los aspectos claves de su solución, en este caso las estructuras, listas y funciones más importantes.
- Mostrar parte de la jugabilidad. Aquí puede realizar cortes en la grabación para que pueda presentar todo en los 5 minutos señalados.

Entregable

El entregable corresponde al link de acceso público al repositorio de la cuenta de github. **En el archivo README del repositorio debe incorporar el LINK para el video.**

Porcentajes y ponderación

#	Indicadores	Ponderación
1	Desarrolla la declaración de estructura (s) para trabajar según la problemática.	3%
2	Desarrolla la función que entrega la posibilidad de realizar la carga inicial de información.	2%
3	Desarrolla la función que entrega la posibilidad de creación de nuevas cartas.	5%
4	Desarrolla la funcionalidad que entrega la posibilidad de seleccionar las opciones a través de un menú.	2%
5	Desarrolla la funcionalidad que entrega la posibilidad de entregar cartas de manera aleatoria a los jugadores.	5%
6	Desarrolla la funcionalidad que entrega la posibilidad de gestionar los mazos de cada jugador.	10%
7	Desarrolla la funcionalidad que entrega la posibilidad de gestionar los puntos de vida de los jugadores durante la partida.	10%
8	Desarrolla la implementación del algoritmo para poder competir a través de la simulación por parte del programa (opciones en cada turno).	10%
9	Desarrolla la implementación del algoritmo para poder competir y simular la jugabilidad por parte del programa.	10%
10	Desarrolla la función que permite conocer el historial del jugador una vez finalizada la partida.	10%
11	Desarrolla la implementación de la gestión correcta de liberación de memoria	3%
12	Desarrolla una solución con ausencia de errores y alertas.	5%
13	Desarrolla de manera correcta comentarios en la estructura(s) y funciones.	5%
14	Realiza la gestión del proyecto a través de un repositorio y presenta a lo menos cuatro registros de commits realizados durante el desarrollo.	2%
15	Realiza una correcta descripción del Software junto con la utilización del formato recomendado para la elaboración del archivo README.	3%
16	Realizar una correcta descripción de las instrucciones a realizar para una correcta instalación y/o compilación del producto de Software.	2%

17	Realizar la construcción de material multimedia (video) para ser entregado según los requerimientos solicitados.	3%
18	Desarrolla material de apoyo para ser expuesto en el video.	5%
19	Desarrolla una exposición exposición de la solución desarrollada utilizando un lenguaje técnico y formal	5%
	TOTAL	**100%**

Archivo de Ejemplo

```
Name,Type,LP,AP,DF
Gandalf,Mage,40,20,10
Dracula,Necromancer,30,25,15
Thor,Viking,50,30,20
Medusa,Beast,35,35,25
Merlin,Mage,45,25,15
Frankenstein,Necromancer,25,30,30
Lagertha,Viking,40,35,20
Kraken,Beast,55,40,30
Athena,Mage,35,25,20
Zombie,Necromancer,20,30,35
Odin,Viking,60,35,25
Dragon,Beast,70,50,40
The conqueror Jack,Mage,55,45,25
Queen of Shadows,Necromancer,40,45,30
Bjorn the Brave,Viking,70,40,30
Cerberus,Beast,80,60,50
Sorceress Morgana,Mage,50,35,20
Mad Scientist Igor,Necromancer,30,40,45
Ragnar the Fierce,Viking,75,50,35
Hydra,Beast,90,70,55
Valeria the Enchantress,Mage,45,40,25
Captain Hook,Necromancer,35,45,40
Erik the Red,Viking,80,45,40
Phoenix,Beast,100,80,60
The mysterious Masked Magician,Mage,60,50,30
Mistress of Shadows and Deception,Necromancer,50,60,40
Ivar the Boneless, Viking, 90, 55, 45
Chimera, Beast, 110, 90, 70
Luna the Moonlit Mage, Mage, 65, 55, 35
Baron von Bloodsucker, Necromancer, 40, 55, 60
```

```
Leif the Lucky, Viking, 85, 50, 45
Thunderbird, Beast, 120, 100, 80
The Enigmatic Enchantress, Mage, 70, 60, 40
Blackbeard the Cursed, Necromancer, 45, 60, 55
Astrid the Fearless, Viking, 95, 55, 50
Krampus, Beast, 130, 110, 90
Midnight Mysteria, Mage, 75, 65, 45
Countess Vampira, Necromancer, 50, 65, 70
Rollo the Mighty, Viking, 100, 60, 55
Manticore, Beast, 140, 120, 100
Mystic Magellan, Mage, 80, 70, 50
Lord Lich, Necromancer, 55, 70, 75
Sigrid the Swift, Viking, 105, 65, 60
Serpent King, Beast, 150, 130, 110
```