Trabajo Práctico DoCe

# Funcionalidades

## Funcionalidades de Interfaz y Control General

1. **Menú principal**
   * Mostrar opciones: Jugar, Ver ranking y Salir.
   * Validar entrada del usuario.
   * Redirigir según la opción elegida.
2. **Carga del jugador**
   * Solicitar e ingresar nombre del jugador.
   * Validar el nombre del usuario.
3. **Selección de dificultad**
   * Permite elegir entre tres niveles: Fácil, Medio y Difícil.

## Funcionalidades del juego

1. **Inicialización de la partida**
   * Crear y mezclar el mazo inicial.
   * Repartir 3 cartas a cada jugador.
   * Seleccionar aleatoriamente quien inicia.
2. **Lógica de juego por turno**
   * Mostrar cartas disponibles del jugador.
   * Permitir al jugador elegir una carta válida de su mano.
   * Aplicar el efecto de la carta.
   * Robar una carta del mazo.
   * Controlar repetición de turno si corresponde.
   * Registrar jugada.
   * Verificar si hay un ganador.
   * Controlar que ningún jugador tenga puntos negativos.
3. **Reposición del mazo**
   * Si el mazo se agota, mezclar los descartes y continuar el juego.

## Inteligencia Artificial (IA)

1. Fácil.
2. Medio.
3. Difícil.

## Registro e informe de partida

1. **Registro interno de jugadas:** 
   * Guardar en memoria: Jugador que juega, carta jugada, efecto aplicado, puntos actuales de cada jugador, etcétera.
2. **Generar informe final:** Crear archivo con nombre, *informe-juego\_YYYY-MM-DD-HH-mm.txt*, incluyendo todas las jugadas, puntos y ganador.

## Funcionalidades API

1. **Lectura de configuración para la API desde archivo**.
2. **Envío de resultado del juego (POST).**
3. **Consultar ranking (GET).**

# Estructura de datos propuestas

## Mazo de cartas

Estructura de cola, la lógica sería porque es una estructura FIFO y las cartas descartadas se vuelven a mezclar y se agregan al final.

## Mano de cada jugador

Estructura de datos lista simplemente enlazada, dado que se pueden eliminar cartas en cualquier posición, se puede recorrer con facilidad y siempre que se robe una carta del mazo, se inserta al final.

## Cartas descartadas

Se puede utilizar una pila o una lista simplemente enlazada.

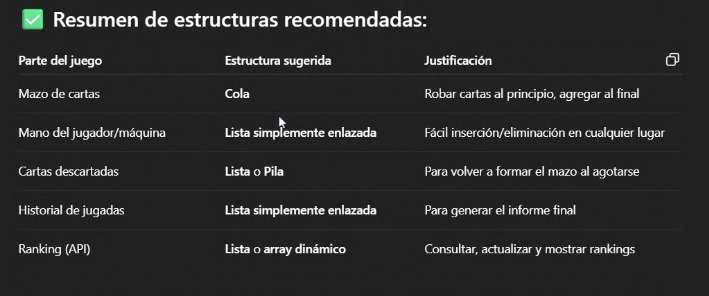
## Historial de jugadas

La estructura de datos a utilizar es una lista simplemente enlazada, dado que se puede ir insertando jugadas al final fácilmente y se pueden recorrer todas para generar el informe.

## Ranking de jugadores

La estructura de datos a utilizar es una lista simplemente enlazada, la información la vamos a leer desde la API y la podemos recorrer partiendo.

## Resumen de estructuras:



# Algoritmo de dificultades

## Nivel Fácil

Totalmente random.

## Nivel Medio

* No usar cartas que restan puntos si el jugador tiene 0.
* Si la máquina tiene 8 o más puntos, priorizar cartas que suman. (priorizar ganar)
* Solo usar repetir turno si tiene otra carta buena en mano.
* Si no se cumple, se realiza una jugada aleatoria.

**En resumen son 3 condiciones a chequear:**

1. **PRIMERA CONDICIÓN:** si el jugador tiene 0 puntos NO tirar negativo(descartarla), y elegir entre las dos restantes.
2. **SEGUNDA CONDICIÓN**: evaluar si la máquina tiene 8 o más, solo tirar cartas que suman.
3. **TERCERA CONDICIÓN:** si le llegó carta “repetir” a la máquina, tirala inmediatamente y que agarrar otra.
4. **FALSO DE TODAS:** tirar aleatorio.

Evaluamos si se cumple 1era condición:

* true: descartar cartas negativas. Usar las restantes.
* false: nada

**1.2) Validación si el mazo tiene todas cartas negativas, tirar random.**

Evaluamos si se cumple 2da condicion:

* true: tirar carta que sume si o si(primera que encuentra)
* false: nada

Evaluamos si se cumple 3 condición:

true: tirarla inmediatamente y agarrar otra

false: sería el falso de todas -> tirar cualquier carta

## Nivel Difícil

Hasta los 7 puntos de 8 puntos para arriba está cerca de ganar (priorizar ganar???)

1 condición: Si el usuario le tira una carta para restar puntos a la máquina   
   
true ->

* 1.2 condicion: Yo como maquina, tengo una carta espejo?

true-> tiro la carta -> termina el turno   
false -> me voy a la condición 2

false->

* 2 condición: Yo como maquina, tengo una carta de repetir turno & & tengo dos cartas buenas?

True -> tira la carta de repetir -> vuelvo a preguntar la condición 2   
False-> - 3 condición: Tengo cartas para sumar?

True-> tiro esa carta (tiró la mayor)-> termina el turno

False -> 4 condición: Yo como máquina, tengo cartas para restar?

true-> tiro carta de restar (debe ser inteligente) ->fin turno

False -> tira random (serían 3 espejos?)

**Ejemplo de informe:**   
   
Jugador   
Carta: Sumar 2 puntos   
Puntaje 2   
máquina   
Carta: Sumar 1 punto   
Puntaje: 1   
…   
Jugador   
Carta: Sumar 2 puntos   
Puntaje 8   
…   
Jugador   
Carta: Restar 1 puntos   
Puntaje 8   
…   
Máquina   
Carta: espejo (string de tCarta que es la descripción)   
Puntaje: 1   
Puntaje de Usuario: 7

2 repetir y 1 espejo   
   
1 repetir y 2 espejos   
   
2 espejos o un sumar o un restar o a lo sumo otro repetir (que en este caso lo vuelvo a tirar)   
   
caso super límite es que tire el repetir y levanté otro espejo   
entonces tengo 3 espejos   
tiro cualquiera

Si el jugador tiene 8 puntos debería de priorizar el siguiente orden   
1 condición -> 2 condición -> 4 condición -> 3 condición

# Tareas por desarrollar

* Escribir este documento en el [README.MD](http://readme.md).
* Configurar el proyecto con las librerías a utilizar (SDL2, libcurl, etcétera)
* Desarrollar las estructuras de datos.
* Desarrollar la función del menú principal.
* Función para ingresar nombre y validar el usuario.
* Desarrollar función de dificultad.
* Función de crear mazo.
* Función de mezclar mazo.
* Función repartir cartas a jugador y máquina
* Función para sortear quien inicia
* Función mostrar cartas disponibles del jugador
* Función seleccionar carta
* Función calcular puntos (en la partida)
* Función agarrar carta del mazo
* Función validar “repetir” para tirar devuelta
* Registrar las jugadas (máquina y usuario)
* Validar si hay ganador
* Función reponer mazo (se agotaron las 40 cartas, mezclar y reponer)
* Desarrollar nivel facil, medio, dificil
* Cuando se finaliza la partida, generar txt con el registro de jugadas
* Enviar resultado a la API
* Función consultar ranking a la API
* Función salir.

DESCARTAMOS -> ver que queda de todos los puntos  
En general ya tenemos el proyecto bastante avanzado

Nuevas Tareas

* Reorganizar el código, agregar nuevos .c y distribuir un poco el proyecto para que no este todo junto
* Ver todo el tema de la API   
  -> cargar las jugadas con el POST, una vez terminada la partida se tiene que mandar a la API el resultado el juego.  
  -> ver el ranking, que se encarga de hacer el GET y traerse todos los puntajes que tienen las personas en la API y debería de mostrar todos esos resultados

Casos

Si tengo 2 espejos y 1 repetir debería de repetir  
  
Buenas : espejo sumar y repetir  
Malas: restar  
  
  
Texto

El contenido generado por IA puede ser incorrecto.  
Aca le tiramos el -1 punto y debería de haber tirado el espejo, pero eligió repetir el turno  
  
Texto

El contenido generado por IA puede ser incorrecto.  
Aca le tiramos el -2 puntos y debería de haber tirado el espejo, pero volvió a repetir

Texto

El contenido generado por IA puede ser incorrecto.  
Le tire el -2 y la maquina ahora si tiro el espejo, capaz rompe cuando tiene espejo y repetir