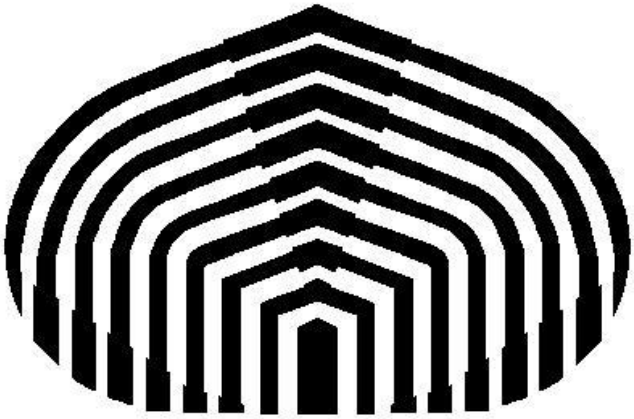


Universidad Simón Bolívar
Departamento de computación y tecnología de la información
Laboratorio de Algoritmos y Estructuras I – CI2691

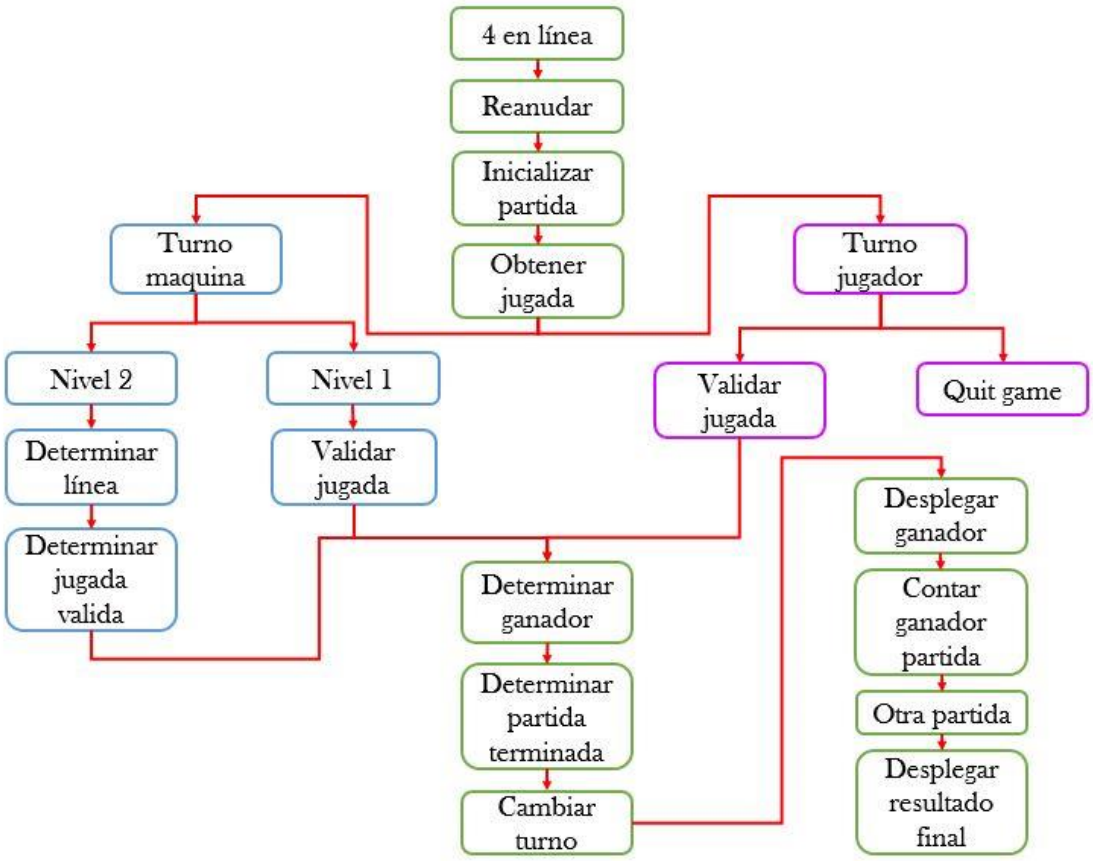
PROYECTO
4 EN LINEA

Integrantes:
Pietro Iaia. Carnet: 15-10718
Antonella Requena. Carnet: 15-11196

INTRODUCCION:

El juego 4 en línea consiste en un juego de dos personas en el cual el objetivo de los dos es hacer una línea vertical, horizontal o diagonal de 4 fichas consecutivas. La ficha que agreguen debe ser puesta en la ultima fila disponible (a diferencia de el juego “la vieja” donde puedes ponerla en la fila que desees). Si se rellenan todas las casillas y nadie tiene 4 fichas consecutivas, el juego termina en empate.

El problema a resolver consiste en lograr desarrollar en el lenguaje de programación Python una adaptación del juego 4 en línea. Para hallar la solución al problema implementaremos el código utilizando la estructura de análisis descendente, basándonos en una carta estructurada.



Objetivos de los sub-programas establecidos en la carta estructurada:

- **Reanudar:** Este subprograma revisa si existe el archivo ‘save.txt’, si lo encuentra se le pregunta al usuario si quiere cargar la partida anterior, si selecciona que si se le cargaran los resultados finales de la partida guardada. Si no se encuentra el ‘save.txt’ o se selecciona que no quiere cargar la partida anterior, el programa no hace efecto y el código sigue.

- **Inicializar Partida:** Este subprograma se encarga de inicializar el Tablero, colocando todas las casillas en blanco e inicializar algunas variables necesarias para cada partida. También es donde se le pedirá al usuario ingresar su nombre y el nivel de dificultad que desee jugar. Si es la primera vez jugando, se le asigna el turno al computador, si no es así se le asigna el turno al ganador de la partida anterior.

- **Seguir Jugando:** Siempre que el jugador haga una jugada este procedimiento le preguntara si desea seguir jugando, si se le responde que ‘si’ el programa le dejara jugar. Si responde que ‘no’ se le pedirá si desea guardar su partida y luego de esto el programa se cerrará.

- **Obtener Jugada:**
    - **Turno del Jugador:** El jugador elegirá la columna que desee jugar con el click del mouse y una vez escogida, se presiona la tecla ‘Espacio’ para lanzar la ficha a la ultima fila libre de esa columna.
        - **Validar Jugada:** Este es el subprograma necesario para verificar la ultima fila disponible en la columna deseada. Si se encuentra una, se elige esta, si no se le avisa al jugador que no

puede jugar en esa columna y lo deja escoger otra.

- **Quit Game:** Si el jugador se cansa de jugar y desea cerrar el juego, presionara la ‘x’ en la esquina de arriba de la pantalla. El programa le preguntará si desea guardar la partida o no. Una vez hecha la decisión, el programa se cerrará.

- **Turno Computadora:**
- **Nivel 1:**
  - **Validar Jugada:** Este es el subprograma necesario para verificar la última fila disponible en la columna deseada. Si se encuentra una, se elige esta, si no se encuentra, en el caso de el turno de la computadora, se saldría del sub-programa.

- **Nivel 2:**
  - **Determinar Línea:** En este sub-programa se verifica cualquier posible jugada por encima o a los lados de la ultima ficha jugada por el computador. Una vez verificado, revisa si puede seguir jugando por su Línea seleccionada anteriormente. Si puede escoge esa, si no escoge cualquier otra que sea Válida.

  - **Determinar Jugada Valida:** Este Sub-programa trabaja igual a Validar Jugada solo que mas centrado en el nivel 2 de dificultad ya que verifica no solo si tiene una ficha por debajo, también verifica donde puede estar colocada la siguiente ficha, esto ayuda a seleccionar el tipo de Línea que se quiere jugar.

- **Determinar Ganador:** Este Sub-programa verifica cada tipo de línea en toda la matriz (Horizontal, vertical, diagonal derecha y diagonal izquierda). Si encuentra 4 fichas consecutivas del mismo color (del mismo jugador) Asigna la variable Ganador a el jugador que lo haya logrado. Además, la resalta.

- **Determinar Partida Terminada:** Primero verifica si hay espacios libres en el tablero, si no los hay, hace que el juego termine ya que no podría haber mas jugadas. Si aun encuentra espacios libres, verifica si hay un ganador. Si lo hay, termina el juego, si no lo continua.

- **Cambiar Turno:** Cuando llegue a este Sub-programa es porque habrá terminado una corrida del ciclo y cambia el turno al siguiente jugador.

- **Desplegar Ganador:** Una vez terminada la partida, se checkea el ganador de la partida y lo despliega en la pantalla.

- **Contar Ganador Partida:** Este sub-programa cuenta el numero de veces que gana el Jugador, la computadora y cuantas veces queda en empate.

- **Otra Partida:** Este subprograma le avisa al jugador que la partida termino y le pregunta si quiere jugar otra. Si selecciona que sí, se reinicia el ciclo de nuevo empezando por el subprograma Inicializar Partida. Si selecciona que no,

se le pregunta si quiere guardar la partida y luego sale del ciclo del juego.

- **Desplegar Resultado Final:** Este subprograma es llamado una vez que se vaya a terminar de jugar el juego y muestra en pantalla la cantidad de partidas ganadas por cada uno de los jugadores y las empatadas.

Variables del programa principal:

- **N:**int >> Constante que representa las filas de la matriz (Para este programa, N será 6)
- **M:**int >> Constante que representa las columnas de la matriz (Para este programa, M será 7)
- **NombreJugador:**str >> guarda el nombre del jugador
- **Nivel:**int >> guarda el nivel que desea jugar el usuario
- **Tablero:**array [0,N) of int >> Variable usada para contener la matriz del tablero (posiciones jugadas)
- **Turno:**int >> Variable del turno de los jugadores, vale 1 si el turno es del usuario y 2 si es del CPU. Cambia una vez que pasa por el ciclo (que se obtiene la jugada)
- **JugadaPrimeraVez:**bool >> Variable usada para ver si el CPU en el nivel medio ya jugo por primera vez
- **Linea:**int >> Variable que contiene la Línea que el CPU desea jugar en el nivel medio
- **Ganador:**int >> Variable que contiene el Ganador de la partida (Siempre será 0 hasta que se decida si se queda así o cambia a 1 o 2)
- **siguePartida:**bool >> Variable que determina si la partida sigue o ya termino (Cuando se le pregunta al usuario si desea seguir jugando y dice que sí, vale True)
- **JugarOtra:**bool >> Variable para determinar si se desea jugar otra partida o no

- **P:**int >> Variable que contendrá la última fila jugada por el CPU en el nivel medio
- **Q:**int >> Variable que contendrá la última columna jugada por el CPU en el nivel medio
- **Ganador0:**int >> Contador de veces que ha quedado en empate la partida
- **Ganador1:**int >> Contador de veces que ha ganado el Jugador
- **Ganador2:**int >> Contador de veces que ha ganado el computador
- **PrimeraPartida:** bool >> Esta variable verifica si es la primera partida que se juega entre el computador y el jugador

Manera en que la maquina toma decisiones:

- **Nivel Principiante o 1:**
  Para el nivel 1 de este juego se nos pidió que la computadora escogiera una casilla valida cualquiera y eso es lo que se hizo. El computador escoge una de las 7 columnas y coloca la ficha en la última fila vacía.
- **Nivel Intermedio o 2:**
  Para el Nivel 2 de este juego se nos pidió que la maquina tratara de construir una línea de 4 fichas. La estrategia que implementamos fue que el computador escogiera entre 5 tipos de líneas. Línea horizontal derecha, Línea horizontal izquierda, Línea vertical, Línea diagonal derecha o Línea diagonal izquierda. Y la escoge dependiendo de si es valido jugar en esa dirección. Si el computador ya escogió una Línea en alguna jugada anterior y es posible seguir jugando en esa dirección, el computador lo hará hasta que se encuentre con un obstáculo y ahí cambiará de Línea o, si no puede moverse en ninguna de las 5 direcciones, escoge cualquier casilla vacía valida.

Conclusiones:

Gracias a la realización de este proyecto nos dimos cuenta la facilidad de crear un programa si se divide en tareas mas pequeñas y aisladas. Y aun mas si el programa necesita de muchas funciones.

La utilización de análisis descendiente también fue de ayuda para la lectura y “Debuggin” del código, ya que cuando se presentaba un problema, se sabía rápidamente que sub-programa lo generaba y, al ser este aislado de los demás (es decir, todo lo que ocurría en el no era afectado por otros subprogramas) se podía arreglar el problema más fácilmente.

La realización de este proyecto también fue de gran ayuda en aprender el esfuerzo que lleva construir un videojuego y como trabajar en equipo dentro de un mismo código.