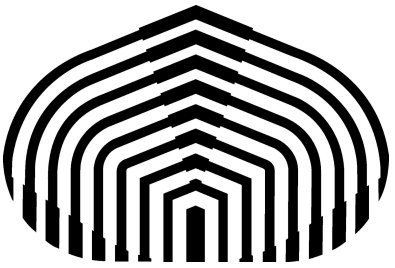


Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Laboratorio de Algoritmos y Estructuras I – CI2691
Prof. Carmen Rosseline Rodríguez

Especificaciones del Proyecto
4 en Línea

Integrantes:
· Manuel Faria. Carnet: 15-10463
· Juan Oropeza. Carnet: 15–11041

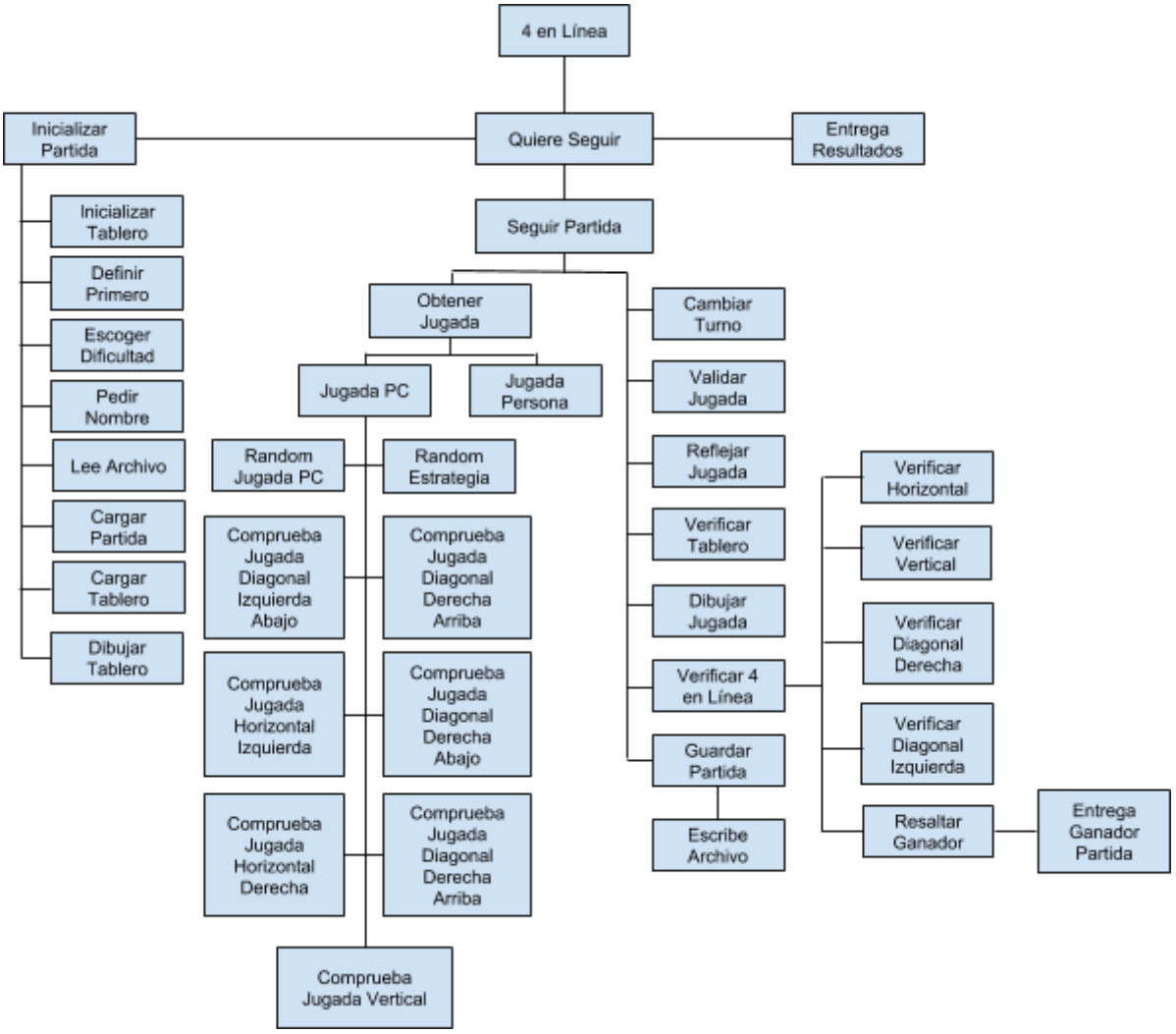
Sartenejas, abril de 2018

Introducción

El juego cuatro en línea consiste en un tablero (generalmente de 6 filas y 7 columnas) en el cual el objetivo es lograr realizar una línea de cuatro fichas en cualquiera de las direcciones posibles (Horizontal, Vertical o Diagonal). Los jugadores se intercalan para colocar en el tablero una ficha a la vez, quien logre realizar la línea antes explicada gana la partida.

Este proyecto consiste en la implementación de dicho juego en los lenguajes de programación GCL y Python. Para ello se utilizaron las técnicas de programación estructurada enseñadas en este curso. Los principios de este trabajo se muestran en la carta estructurada anexada en la siguiente página, sobre ella, se basa todo el proyecto. Se aplica la estrategia de Divide and Conquer para dividir el problema que puede llegar a ser bastante complejo en pequeños problemas que son más fáciles de resolver.

Carta Estructurada



Objetivo de los subprogramas planteados en la carta estructurada:

1. ***Inicializar Partida:*** Función que se encarga de preparar los elementos necesarios para comenzar una nueva partida. Establece los valores de la cantidad de jugadas y la última jugada realizada. Devuelve 4 enteros y un string que representan la dificultad, el primer jugador, el número de jugadas del usuario, el número de jugadas de la computador y el nombre del usuario.
    1.1. ***Inicializar Tablero:*** Procedimiento que rellena la matriz del tablero con ceros para indicar que el tablero está vacío y así poder comenzar una nueva partida.
    1.2. ***Definir Primero:*** Función que establece quien es el primer jugador que podrá colocar una ficha en el tablero. Si es la primera partida por defecto comenzará el usuario, en caso contrario, comenzará el ganador de la última partida. Se devuelve un entero (1 si comienza el usuario y 0 si comienza la computadora.).
    1.3. ***Escoger Dificultad:*** Función que le pregunta qué tipo de dificultad desea para la partida actual. Si el usuario desea un nivel básico deberá ingresar 0 en la consola, en caso que desee un nivel medio deberá ingresar 1. Arroja el valor que ingrese el usuario.
    1.4. ***Pedir Nombre:*** Función que solicita el nombre del usuario para tener una atención personalizada con el usuario.
    1.5. ***Lee Archivo:*** Función que lee el archivo donde está guardada la partida y carga todos los valores. Devuelve todos los valores necesarios para continuar la partida.
    1.6. ***Cargar Partida:*** Función que le pregunta al usuario si desea cargar una partida. Devuelve un booleano con el resultado de la consulta.
    1.7. ***Cargar Tablero:*** Procedimiento que dibuja el tablero y el estado del mismo una vez que se carga la partida.
    1.8. ***Dibujar Tablero:*** Procedimiento que dibuja las líneas del tablero en la interfaz gráfica.
2. ***Seguir Partida***: Función que le pregunta al usuario si desea continuar con la partida actual, para ello el usuario debe ingresar 0 si no desea continuar o 1 si desea hacerlo. Devuelve un valor booleano con la decisión tomada por el jugador.
    2.1. ***Obtener Jugada:*** Función que verifica quién debe jugar y obtiene la jugada dependiendo de quién juega. Devuelve el número de la columna donde desea jugar y la estrategia tomada por la computadora.
        2.1.1. ***Jugada PC:*** Función que se encarga de la obtener una jugada de la computadora. Busca una jugada posible de acuerdo con la estrategia que la computadora haya tomado en turnos anteriores, en caso de no poder hacerlo selecciona una nueva

jugada de manera aleatoria. Devuelve la columna donde el usuario desea jugar y la estrategia adoptada por la computadora.

2.1.1.1. ***Random Jugada Computadora:*** Función que devuelve una jugada para la computadora aleatoriamente.

2.1.1.2. ***Random Estrategia:*** Función que devuelve una estrategia aleatoria.

2.1.1.3. ***Comprueba Jugada Vertical:*** Función que verifica si puede realizar una jugada con la estrategia vertical. Devuelve la columna donde se puede realizar la jugada (Si no se puede realizar la jugada devuelve -1), un booleano que indica si se puede jugar y la estrategia tomada por la computadora (Si no se puedo aplicar la estrategia se escoge una nueva aleatoriamente).

2.1.1.4. ***Comprueba Jugada Horizontal Derecha:*** Función que verifica si puede realizar una jugada con la estrategia horizontal por la derecha. Devuelve la columna donde se puede realizar la jugada (Si no se puede realizar la jugada devuelve -1), un booleano que indica si se puede jugar y la estrategia tomada por la computadora (Si no se puedo aplicar la estrategia se escoge una nueva aleatoriamente).

2.1.1.5. ***Comprueba Jugada Horizontal Izquierda:*** Función que verifica si puede realizar una jugada con la estrategia horizontal por la izquierda. Devuelve la columna donde se puede realizar la jugada (Si no se puede realizar la jugada devuelve -1), un booleano que indica si se puede jugar y la estrategia tomada por la computadora (Si no se puedo aplicar la estrategia se escoge una nueva aleatoriamente).

2.1.1.6. ***Comprueba Jugada Diagonal Derecha Abajo:*** Función que verifica si puede realizar una jugada con la estrategia diagonal por la izquierda hacia abajo. Devuelve la columna donde se puede realizar la jugada (Si no se puede realizar la jugada devuelve -1), un booleano que indica si se puede jugar y la estrategia tomada por la computadora (Si no se puedo aplicar la estrategia se escoge una nueva aleatoriamente).

2.1.1.7. ***Comprueba Jugada Diagonal Derecha Arriba:*** Función que verifica si puede realizar una jugada con la estrategia diagonal por la izquierda hacia arriba. Devuelve la columna donde se puede realizar la jugada (Si no se

puede realizar la jugada devuelve -1), un booleano que indica si se puede jugar y la estrategia tomada por la computadora (Si no se puedo aplicar la estrategia se escoge una nueva aleatoriamente).

2.1.1.8. ***Comprueba Jugada Diagonal Izquierda Abajo:*** Función que verifica si puede realizar una jugada con la estrategia diagonal por la izquierda hacia abajo. Devuelve la columna donde se puede realizar la jugada (Si no se puede realizar la jugada devuelve -1), un booleano que indica si se puede jugar y la estrategia tomada por la computadora (Si no se puedo aplicar la estrategia se escoge una nueva aleatoriamente).

2.1.1.9. ***Comprueba Jugada Diagonal Izquierda Arriba:*** Función que verifica si puede realizar una jugada con la estrategia diagonal por la izquierda hacia arriba. Devuelve la columna donde se puede realizar la jugada (Si no se puede realizar la jugada devuelve -1), un booleano que indica si se puede jugar y la estrategia tomada por la computadora (Si no se puedo aplicar la estrategia se escoge una nueva aleatoriamente).

2.1.2. ***Jugada Persona:*** Función que le pregunta al usuario en cual columna desea jugar. Devuelve el valor de la columna que el usuario ingrese en la consola.

2.2. ***Cambiar Turno:*** Función que verifica quién es el próximo que debe jugar, en caso de ser la computadora devuelve 2, caso contrario devuelve 1.

2.3. ***Validar Jugada:*** Función que verifica si una jugada es válida o no. Esta función es necesaria ya que a pesar de que en la función Obtener Jugada se restringen los valores para que la columna esté definida dentro de la matriz, en algunas ocasione puede ocurrir que la columna esté llena, en ese caso se debe requerir la jugada del usuario o computadora nuevamente. Devuelve un booleano que representa si la jugada es válida.

2.4. ***Reflejar Jugada:*** Función que realiza los cambios en la matriz del tablero. A su vez modifica los valores de la última jugada y la cantidad de jugadas. Devuelve los valores de la última jugada y la cantidad de jugadas.

2.5. ***Guardar Partida:*** Función que le pregunta al usuario si desea guardar la partida. Devuelve un booleano que indica si el usuario quiere guardar la partida o no.

2.5.1. ***Escribe Archivo:*** Procedimiento que escribe los valores actuales en un archivo para guardar la partida y poder ser cargada luego.

2.6. ***Dibujar Jugada:*** Procedimiento que dibuja una ficha en la posición indicada.

2.7. ***Verificar Tablero:*** Función que verifica si el tablero está lleno o no. Devuelve un valor booleano que indica si está o no.

2.8. ***Verificar 4 en Línea:*** Función que verifica si existe alguna línea de 4 fichas en todo el tablero. Devuelve un booleano y un entero que representan la verificación hecha y el usuario que tiene la línea de 4 fichas.

2.8.1. ***Verificar Horizontal:*** Función que verifica si existe una línea horizontal en el tablero, en caso de haberla devuelve 1 si la línea es del jugador y 2 si es de la computadora. En caso de no haber línea horizontal devuelve 0.

2.8.2. ***Verificar Vertical:*** Función que verifica si existe una línea vertical en el tablero, en caso de haberla devuelve 1 si la línea es del jugador y 2 si es de la computadora. En caso de no haber línea vertical devuelve 0.

2.8.3. ***Verificar Diagonal Derecha:*** Función que verifica si existe una línea diagonal derecha en el tablero, en caso de haberla devuelve 1 si la línea es del jugador y 2 si es de la computadora. En caso de no haber línea diagonal derecha devuelve 0.

2.8.4. ***Verificar Diagonal Izquierda:*** Función que verifica si existe una línea diagonal derecha en el tablero, en caso de haberla devuelve 1 si la línea es del jugador y 2 si es de la computadora. En caso de no haber línea diagonal derecha devuelve 0.

2.8.5. ***Resaltar Ganador:*** Procedimiento que resalta el color de las fichas que han generado una fila.

2.8.5.1. ***Entrega Ganador Partida:*** Procedimiento que imprime en pantalla el ganador de la partida que acaba de finalizar o si es necesario indica que hubo un empate.

3. ***Quiere Seguir:*** Función que le pregunta al usuario si desea jugar una nueva partida, en caso de querer hacerlo el usuario debe ingresar 1 en la consola, caso contrario debe ingresar 0. Devuelve un valor booleano que representa la decisión tomada por el jugador.

4. ***Entrega Resultados:*** Procedimiento que imprime en consola los resultados generales de un juego, es decir, número de partidas ganadas por el jugador, por la computadora y número de empates.

Variables del programa principal

- *Tablero (**list**):* Matriz donde se almacenan los datos del tablero.
- *Dificultad (**int**):* Variable donde se almacena la dificultad escogida por el usuario.
- *ultimoGanador (**int**):* Variable que almacena el último jugador en ganar una partida.
- *jugador (**int**):* Variable que almacena el jugador actual.
- *numPartidas (**int**):* Variable que almacena la cantidad de partidas jugadas.
- *jugada (**int**):* Variable que almacena la columna donde se desea jugar.
- *ultimaJugada (**list**):* Arreglo donde se almacenan las coordenadas de la última jugada en el tablero.
- *continuar (**bool**):* Variable que almacena si una partida ha terminado o no.
- *numJugadas (**int**):* Variable que almacena la cantidad de jugadas de la partida.
- *numJugadasPC (**int**):* Variable que almacena la cantidad de jugadas de la partida.
- *numEmpates (**int**):* Variable que almacena la cantidad de empates.
- *partidasGanadasPersona (**int**):* Variable que almacena la cantidad de partidas de ganadas por la persona.
- *partidasGanadasPC (**int**):* Variable que almacena la cantidad de partidas ganadas por la computadora.
- *ganador (**int**):* Variable que almacena el ganador de la partida.
- *validacion (**bool**):* Variable que almacena si una jugada es válida o no.
- *quiereSeguirJugando (**bool**):* Variable que determina si el jugador quiere continuar con la partida.
- *ingresarJugada (**bool**):* Variable utilizada en el ciclo y almacena si una jugada es válida o no.
- *nombre (**str**):* Variable que almacena el nombre del usuario.
- *estrategia (**int**):* Variable que almacena la estrategia tomada por la computadora.

Implementación del Programa:

- Decidimos que debería haber varios subprogramas que verificaran individualmente que existe una línea de 4 fichas en la matriz para determinar si hay algún ganador.
- De igual manera implementamos varios subprogramas que verifican si la estrategia tomada por la computadora es viable, y en caso de no serlo busca una jugada aleatoria en la que se basará la nueva estrategía. Prueba con todas las estrategias en orden aleatorio hasta un máximo definido.

Manera en que la máquina hace la toma de decisiones:

- La computadora selecciona una jugada aleatoriamente y en base a esa jugada estructura su juego.
- Toma una estrategia de manera aleatoria y intenta hacer una línea en función de esa estrategia.
- Si después del máximo de intentos no logra encontrar una jugada que sea válida, busca una jugada aleatoria y la realiza.
- Luego, intenta nuevamente realizar una jugada en función de la anterior y la estrategia seleccionada.

Conclusiones

Este proyecto fue un pilar fundamental para poner en práctica los conocimientos adquiridos en las materias Algoritmos y Estructuras I y Laboratorio de Algoritmos y Estructuras I. La realización de un proyecto de cierto tamaño conlleva tener unas buenas bases en las cuales sustentar el trabajo a realizar. La planificación del trabajo fue fundamental para obtener los resultados que se obtuvieron.

Nos percatamos que a pesar de que los proyectos de computación sufren muchos cambios en el momento de la implementación, siempre es bueno tener una estructura inicial en la cual basar el proyecto, ya que provee una guía para el mismo y ayuda a definir los objetivos y alcances del mismo.

Las soluciones algorítmicas que planteamos siempre fueron orientadas a conseguir la mayor eficiencia posible, utilizamos muchos de los conceptos aprendidos en las clases teóricas. De igual manera, se implementaron las estructuras de datos aprendidas en el laboratorio con el fin de realizar una programación de calidad, en la que el producto final sea mantenible.

La documentación del código es bastante amplia para que la lectura del mismo sea fácil y rápida. Con solo leer el programa principal se tiene una idea de lo que hace el mismo y pueden resolverse los problemas que se presenten de manera eficaz.

Referencias

- https://moodle.asignaturas.usb.ve/pluginfile.php/55944/mod_page/content/5/PRIMEROS%20PASOS%20CON%20PYGAME.pdf
- https://ldc.usb.ve/~meza/ci-2611/e-m2017/IntroduccionALaProgramacion.pdf
- https://ldc.usb.ve/~meza/ci-2611/e-m2017/ProgrammingTheDerivationOfAlgorithms%20Kaldewaij.pdf
- https://docs.python.org/3/