Desarrollo Parcial 2 – Informática II

Integrantes

Darwin Falcao Betancur Cuartas

Sara Valentina Ochoa Sanchez

Universidad de Antioquia

Facultad de Ingeniería

Departamento de Ingeniería Electrónica y Telecomunicaciones

Semestre 2

2023

Contextualización, análisis y diseño de la solución

El problema de crear el juego Othello en Programación Orientada a Objetos (POO) implica diseñar y desarrollar una aplicación de software que simule este juego de estrategia de tablero para dos jugadores. Othello se juega en un tablero cuadriculado y los jugadores colocan fichas de su color en las casillas, con el objetivo de capturar fichas del oponente y ganar al final del juego al tener más fichas de su color en el tablero.

Para abordar el problema de crear el juego Othello en POO, se pueden definir las siguientes clases y conceptos:

1. Tablero:

Atributos: tamaño.

char\*\* tablero.

Métodos:

* Tablero.
* ~Tablero.
* dibujarTablero.
* tableroInicial.

1. Jugador:

Atributos: nombre, colorFicha.

Métodos: indicarPosicion.

1. administarJuego:

Atributos:

Métodos:

* validarMovimientos (verifica si tiene o no movimientos posibles en el tablero): recibe las fichas de los jugadores y el tamaño del tablero, retorna un bool.
* movimientoValido (verifica si poniendo una ficha en cierta posición, se considera una jugada válida): recibe las fichas de los jugadores, la posición y el tamaño del tablero.
* hacerJugada (hace el movimiento de cambiar las fichas): recibe un bool y la posición.
* cuentaFichas
* guardarInformacion (Guardar Información en un archivo de texto).
* verHistorial (lee el archivo que contiene la información y la escribe por consola).

Explicación:

1. Para clase tablero se definió dos atributos que sería el tamaño de la “matriz” por decirlo de alguna forma, este es el que va a permitir hacer la matriz del tamaño que se desee, al ser un cuadrado, solo con un número se definen filas y columnas y el puntero a punteros de tipo char, con sus respectivos métodos getter para acceder a esa información. Además, tendrá un método llamado tableroInicial que muestra el tablero a los jugados, otro será el dibujarTablero que será el de dibujar el tablero como tal en la consola, con las posiciones de las fichas iniciales y dando pie a que se inicie el juego, este mismo método actualizará el tablero y lo volverá a imprimir por pantalla para mostrar el resultado de cómo va el juego, también tendrá los métodos constructor y destructor que serán los encargados de asignar y liberar la memoria dinámica respectivamente.
2. La clase Jugador tendrá dos atributos que serán el nombre donde quedará guardada la información para el momento de escribir en el documento de texto al finalizar la partida y el color de ficha que le tocó, esta informacion también se utilizará al momento de guardar la información de la partida, estos atributos con sus respectivos setters y getters. También contará con el método señalarPosicion que será el encargado de recibir la información del lugar en el tablero donde el jugador quiere poner su ficha.
3. Por último, tenemos la clase administrarJuego, acá van a estar todas las reglas del juego, que se puede hacer y que no, separadas por métodos, cada respuesta de un método, permitirá (o no) que se pase al siguiente, es decir, el primer método será validarMovimientos, este mirará si el jugador tiene al menos un movimiento válido en el tablero, de no ser así (que sería el caso donde no encierra fichas del otro jugador) tendrá que pasar el turno al otro participante, pero si por el contrario, si tiene jugadas disponibles, pasará al siguiente método que es el de hacerJugada en este se validará que puede poner una ficha en esa posición o si no lo puede hacer (por ejemplo si quisiera colocar una ficha en una posición ocupada) si efectivamente puede poner su ficha en ese lugar para hacer la jugada y luego cambiará las fichas que quedaron encerradas. El método guardarInformacion escribirá en el archivo de texto el resumen de la partida: nombre de los jugadores, fecha, hora, quién ganó y con cuantas fichas. Será invocado cuando ninguno de los jugadores tenga más movimientos disponibles o ya esté lleno el tablero.

Ejecución del programa:

Se creará un menú que le permita al usuario:

1) Jugar

Si elige la opción 1, le pedirá los nombres de ambos jugadores e inmediatamente después les mostrará el tablero indicando quién será las fichas negras (-) y quien las fichas blancas (\*), le pedirá al jugador de ese momento que ingrese la posición en la que quiere colocar la ficha (La indicará con la intersección entre letra y el número, ejemplo: A7) evaluará lo anteriormente explicado, luego de hacer todo el proceso, pasa el turno a la otra persona y así sucesivamente hasta que termine el juego (por alguna de las 2 razones), escribirá la información en el archivo de texto y saldrá al menú principal.

2) visualizar historial

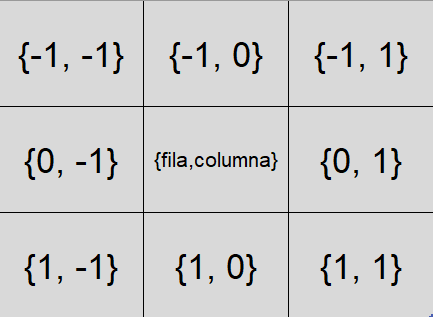
Si eligió la opción número dos y es el primer uso del programa (por lo que no hay datos guardados), simplemente saldrá un mensaje por pantalla diciéndole que no hay datos para mostrar, pero en caso de haberlos, los mostrará por consola.

Está claro que la parte “más compleja” es la parte de implementar el algoritmo como tal del juego, que se reparte entre los métodos:

validarMovimientos: Es el que permite mirar si un jugador tiene posibilidad de jugar esa ronda y ese método se ayuda del método movimientoValido

movimientoValido; Dice si al colocar una ficha en esa posición si logra encerrar fichas del oponente

hacerJugada: Materializa la jugada, pone la ficha en esa posición y cambia las necesarias.

En el método movimientoValido se encontrará un arreglo “direcciones” que es una matriz 2D que se utiliza para representar las ocho direcciones posibles en las que se pueden voltear fichas del oponente. Cada dirección está definida por un par de valores (r, c) que indican cómo se debe mover a lo largo del tablero para explorar esa dirección específica.

Cada par (f, c) representa una dirección:

{1, 0} representa el movimiento hacia abajo en el tablero.

{-1, 0} representa el movimiento hacia arriba en el tablero.

{0, 1} representa el movimiento hacia la derecha en el tablero.

{0, -1} representa el movimiento hacia la izquierda en el tablero.

{-1, -1} representa el movimiento en diagonal hacia arriba y hacia la izquierda.

{-1, 1} representa el movimiento en diagonal hacia arriba y hacia la derecha.

{1, -1} representa el movimiento en diagonal hacia abajo y hacia la izquierda.

{1, 1} representa el movimiento en diagonal hacia abajo y hacia la derecha.

Estas direcciones son utilizadas en el bucle for dentro del método validarMovimiento para explorar cada dirección desde la posición (fila, columna) y verificar si se pueden voltear fichas del oponente en esa dirección.

Al verificar cada dirección, se mueve a lo largo de esa dirección en el tablero para buscar fichas del oponente que puedan ser volteadas por el movimiento válido.

La combinación de r y c te permite avanzar a través del tablero en la dirección específica que estás explorando en el bucle for. r controla el movimiento horizontal, y c controla el movimiento vertical en esa dirección.

Problemas de desarrollo que afrontó

Al trabajar con el paradigma de la Programación Orientada a Objetos, la parte que supone un poco de trabajo es la parte de la abstracción, saber cual es el comportamiento de los objetos que integran el juego Othello, saber cuáles serán sus atributos y los métodos, saber exactamente lo que se necesita porque por ejemplo en nuestro caso como quedó la estructura al final del programa no fue la misma que definimos en un inicio, si bien varias de las componentes si quedaron, hubo otras que se eliminaron.

Otro problema (el más notable) fue la parte de la integración de todo el programa como tal, como nos repartimos las tareas, cada uno fue trabajando por métodos separados y funcionaban, pero cuando nos sentamos a integrarlos, surgieron muchos errores, saber que colocar y en donde colocarlo, etc. Fue un poco complejo, pero al final se pudo lograr el funcionamiento del programa.

Sentimos que aprendimos mucho en cuanto a este paradigma realizando este programa, saber cómo interactúan los objetos con sus atributos y métodos junto con otros objetos, que se puede hacer y que no, etc.