

**Informe parcial 2**

Angie Paola Jaramillo Ortega

Emmanuel Garcés Agudelo

Universidad de Antioquia

Facultad de Ingeniería

Informática II

Noviembre de 2023

**Contextualización**

El problema de programación de implementar el juego Othello implica crear un programa que permita a dos jugadores competir en un juego de estrategia por turnos. Othello se juega en un tablero cuadrado con fichas de dos colores, negro y blanco. El objetivo del juego es tener más fichas de tu color en el tablero al final del juego.

Para resolver este problema, es necesario comprender las reglas del juego, la lógica para validar movimientos, determinar el ganador y gestionar los turnos de los jugadores. Además, se deben utilizar conceptos de programación orientada a objetos, punteros, arreglos y gestión dinámica de memoria para implementar las clases y métodos necesarios.

**Análisis**

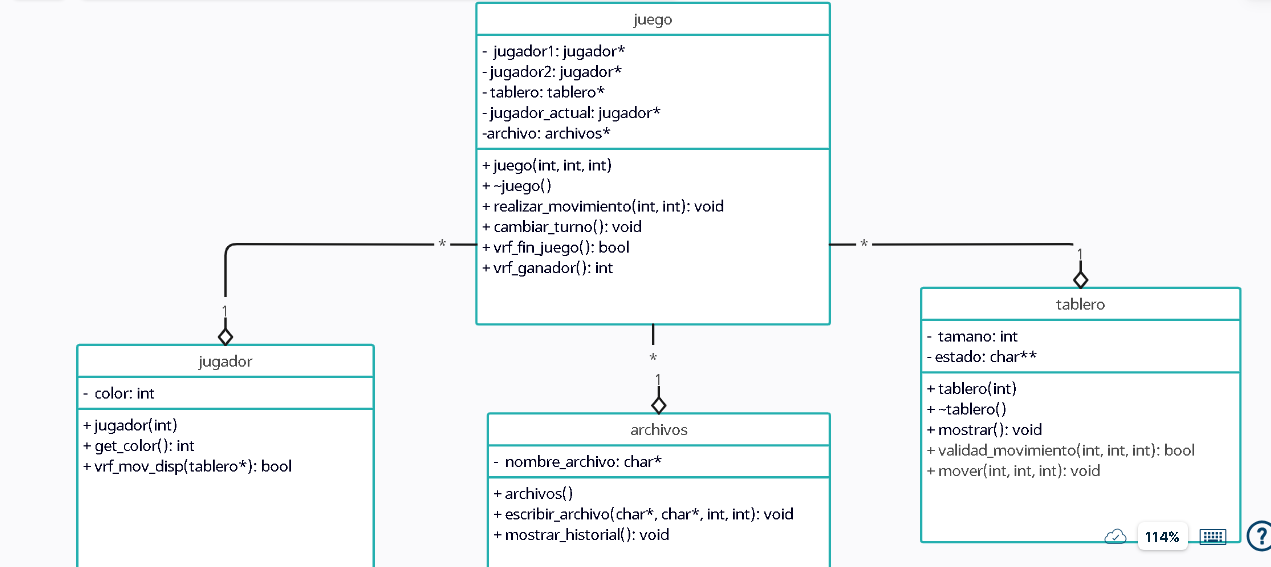
1. Reglas del juego:

* Los jugadores se turnan para colocar una ficha en el tablero.
* Un movimiento es válido si el jugador coloca una ficha en una posición del tablero de tal manera que al menos una ficha del oponente esté atrapada entre la nueva ficha y otra ficha del mismo color en cualquier dirección (horizontal, vertical o diagonal). Las fichas atrapadas se voltean y se convierten en del color del jugador que realiza el movimiento.
* El juego termina cuando no hay más movimientos válidos para realizar. Esto puede deberse a que el tablero está lleno o porque ninguno de los jugadores puede realizar un movimiento que atrape fichas del oponente.
* El jugador con más fichas de su color en el tablero al final del juego es el ganador.

2.Validación de Entrada del Usuario

Se deben implementar mecanismos para validar la entrada del usuario, asegurándose de que las coordenadas proporcionadas estén dentro de los límites del tablero y que los movimientos sean válidos según las reglas del juego.

**Diseño**



**Algoritmos implementados:** <https://github.com/EmmanuelGarces9/Parcial2.git>

**Problemas de desarrollo y solución:**

1. **Problema:** El juego terminaba incorrectamente cuando aún quedaban movimientos válidos para ambos jugadores.

**Solución:** Analizamos la lógica detrás de la condición de finalización del juego y ajustamos el código para verificar correctamente los movimientos disponibles para ambos jugadores antes de declarar el fin del juego.

Identificamos un problema en la lógica de verificación de movimientos disponibles en nuestro juego. La función encargada de esta tarea recorría las casillas del tablero y verificaba si eran movimientos válidos, pero surgía una discrepancia debido a la diferencia entre las coordenadas proporcionadas por los usuarios (comenzando en 1) y las coordenadas internas del programa (comenzando en 0). Como solución, modificamos el código para ajustar estas coordenadas sumando 1 antes de pasarlas a la función de validación.

Esta experiencia resalta la necesidad de coherencia en el manejo de datos y nos enseñó a prestar atención a los detalles para evitar problemas similares en el futuro.

1. **Problema:** Al intentar definir la función de verificación de movimiento disponible en la clase jugador, surgió un error aparentemente relacionado con la inclusión de la clase tablero.

**Solución:** A pesar de que habíamos incluido correctamente "tablero.h" en la clase jugador, aún encontrábamos dificultades para acceder a las variables de la clase tablero. Para arreglar este problema, movimos la función a la clase tablero, donde pudo implementarse sin problemas. Esta experiencia resalta la importancia de entender las dependencias entre clases y cómo las inclusiones pueden afectar la visibilidad de las funciones y variables.