Parcial 02

Entrega Final

JUEGO DE OTHELLO:
ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE LA SOLUCIÓN

Sergio Giraldo
Julio Benavides

Departamento de Ingeniería Electrónica y Telecomunicaciones

Universidad de Antioquia Medellín

Octubre de 2023

Índice

1. Sinopsis del Juego

Othelloi, es un juego de mesa estratégico para dos jugadores que se juega en un tablero de 8x8 casillas. Los jugadores se turnan para colocar fichas de su color con el objetivo de capturar las del oponente al rodearlas en cualquier dirección. El juego combina reglas sencillas con profundas estrategias, lo que lo hace atractivo para todos los niveles. La victoria se determina por la cantidad de fichas propias en el tablero al final de la partida. Othello es un desafío intelectual que ha perdurado como un juego de estrategia clásico.

2. Contextualización

El problema que aborda este proyecto es crear una aplicación de software que permita a dos jugadores competir en el juego de Othello en una computadora. El software debe proporcionar la mecánica del juego, incluida la colocación de fichas, la captura de fichas del oponente y la determinación del ganador.

Consideraciones para el abordaje del problema y desarrollo tanto de propuesta como de la estrategia de solución:

Diseño y Mecánica del Juego: El primer paso es comprender en profundidad las reglas y mecánicas del juego de Othello. Esto incluye cómo se coloca y captura fichas, las reglas para ganar y perder, y los aspectos estratégicos clave. La comprensión de las reglas es fundamental para el desarrollo preciso de la aplicación.

Representación del Tablero: Una de las consideraciones más críticas es cómo representar el tablero de 8x8 en la memoria de la computadora. La elección de una estructura de datos adecuada es esencial para facilitar la colocación y captura de fichas, así como para determinar el estado del juego.

Interfaz de Usuario: Para que los jugadores interactúen con la aplicación, es necesario crear una interfaz de usuario. Esto incluye la visualización del tablero y la capacidad de ingresar las jugadas de los jugadores.

Algoritmos de Juego: Para implementar las reglas de Othello, se deben desarrollar algoritmos que permitan verificar la validez de las jugadas, determinar las fichas capturadas y evaluar el estado del juego en busca de un ganador.

Turnos de Jugadores: El juego debe alternar los turnos de los jugadores, permitiéndoles realizar jugadas válidas. La aplicación debe asegurarse de que cada jugador juegue con su color asignado y que las jugadas sean legales.

Finalización del Juego: La aplicación debe detectar cuándo se ha alcanzado el final del juego, ya sea porque el tablero está lleno o porque ningún jugador puede realizar una jugada válida. En este punto, se deben contar las fichas y determinar el ganador.

Manejo de Errores: Se deben incorporar mecanismos para manejar situaciones de error, como jugadas inválidas o entradas incorrectas por parte de los jugadores.

Optimización y Eficiencia: A medida que el juego progresa, es importante optimizar la eficiencia de los algoritmos, especialmente en juegos avanzados donde el número de posibles jugadas puede ser considerable.

Pruebas y Depuración: La aplicación debe someterse a pruebas exhaustivas para garantizar que funcione según lo previsto y que cumpla con las reglas del juego en todo momento. Además, se deben abordar y corregir posibles errores o problemas.

Documentación y Comentarios: Es fundamental documentar el código de manera clara y proporcionar comentarios adecuados para que otros desarrolladores puedan comprender y mantener la aplicación en el futuro.

3. Análisis

- **Análisis para Abordar el Juego de Othello en C++ (Ejecución por Consola):**

- Estructura de Datos: Representar el tablero como una matriz 8x8 para mantener el estado del juego. Usar variables para rastrear el jugador actual y el número de fichas capturadas por cada jugador.

- Entrada de Usuario: Permitir que los jugadores ingresen sus movimientos a través de la consola, por ejemplo, "A4" para colocar una ficha en la fila A y columna 4.

- Reglas del Juego: Implementar las reglas del juego de Othello, que incluyen la validación de movimientos, la captura de fichas del oponente y la actualización del tablero.

- Finalización del Juego: Detectar cuándo finaliza el juego (por ejemplo, cuando el tablero está lleno o ambos jugadores no pueden realizar movimientos válidos).

- Visualización del Tablero: Diseñar una función para mostrar el estado actual del tablero en la consola.

- Gestión de Turnos: Alternar entre los turnos de los jugadores y verificar la validez de los movimientos en cada turno.

- Conteo de Fichas: Realizar el recuento de fichas al final del juego para determinar al ganador.

- Interfaz de Usuario Amigable: Proporcionar mensajes claros y solicitudes de entrada para que los jugadores comprendan y disfruten el juego.

- Loop Principal: Crear un bucle principal que permita a los jugadores realizar

movimientos hasta que el juego termine.

- Diseño Modular: Considerar la modularidad para separar funciones relacionadas con la lógica del juego, entrada/salida y visualización del tablero.

- Con el anterior análisis comenzaremos al desarrollo de nuestro juego de Othello en C++, centrándonos en cada uno de estos aspectos para lograr una implementación funcional, intuitiva y entretenida para la consola. En **Figura 01**. Se Muestra el diseño de la salida en consola del tablero de juego (Propuesto En Consola):

Figura 01

4. Diagrama Y Diseño De La Solución

Describiremos las estructuras de esta clase, sus miembros y la lógica detrás de los principales algoritmos implementados en sus métodos, atributos y funciones (Diagrama en el Anexo 01):

- **Clase Othello:**

```
class Othello {
public:
  // Miembros públicos
private:
  // Miembros privados
};
```

Miembros públicos:

No se definen miembros públicos en esta clase, pero todos los métodos necesarios para jugar y gestionar el juego están implementados como métodos públicos.

Miembros privados:

char jugadorActual: Representa el jugador actual que tiene el turno. Puede ser '*' o '-'.

char tablero:** Representa el tablero del juego como una matriz de caracteres. Cada celda puede contener '*', '-', o '(vacío)´.

Lógica detrás de los principales algoritmos:

- **Constructor Othello():**

- Inicializa al jugador actual con '*'.
- Inicializa el tablero como una matriz 8x8 y coloca las fichas iniciales en el centro del tablero ('*' y '-').
- Justificación: La lógica del constructor se encarga de establecer un estado inicial válido para el juego, configurando al jugador actual y preparando el tablero.

- **Destructor ~Othello():**

- Libera la memoria dinámica utilizada por el tablero.
- Justificación: El destructor se encarga de evitar fugas de memoria, liberando la memoria dinámica utilizada para el tablero cuando se destruye una instancia de la clase.

- **Método jugar():**

- Inicia el juego y maneja el ciclo del juego.
- Muestra el tablero actual.
- Solicita las jugadas de los jugadores, verifica su validez y las ejecuta.
- **Justificación, Método jugar:**
  Este método es el núcleo del juego, gestionando el flujo del juego, las jugadas de los jugadores y la actualización del tablero. El ciclo while permite que el juego continúe hasta que se cumpla una condición de finalización. Este es crucial para asegurarse de que las jugadas de los jugadores sean válidas. Comprueba tanto los límites del tablero como las reglas del juego y determina si la jugada puede capturar fichas del oponente.

- **Método esJugadaValida(char fila, int columna):**

- Verifica si una jugada es válida siguiendo las reglas del juego.
- Comprueba si la fila y la columna son válidas.
- Comprueba si la celda seleccionada está vacía.
- Explora las direcciones alrededor de la jugada para determinar si hay fichas del oponente que pueden ser capturadas.

- **Método hacerJugada(char fila, int columna):**
- Realiza una jugada válida, actualizando el tablero y capturando fichas del oponente si es necesario.
- Explora las direcciones alrededor de la jugada y actualiza las fichas según las reglas del juego.

- **Método dibujarTablero():**

- Imprime el estado actual del tablero en la consola, mostrando las fichas de los jugadores y las posiciones disponibles.

- **Metodo ContarFichas():** Cuenta las fichas de ambos jugadores en el tablero.
- **Metodo CostrarResultados():** Muestra los resultados finales del juego y guarda los datos en un archivo.
- **Metodo ObtenerFechaActual():** Obtiene la fecha actual para los resultados.
- **Metodo OtenerHoraActual():** Obtiene la hora actual para los resultados.
- **Metodo TableroLleno():** Verifica si el tablero está lleno.
- **Metodo ContarFichasJugador(char ficha):** Cuenta las fichas de un jugador específico en el tablero.
- **Metodo GuardarResultados(int fichasJugador1, int fichasJugador2):** Guarda los resultados en un archivo.

La clase Othello encapsula la lógica del juego y sus miembros y métodos se diseñan para facilitar la gestión de las reglas del juego, la interacción con el usuario y el mantenimiento del estado del tablero. Cada método cumple una función específica en el flujo del juego y se ha implementado siguiendo las reglas de la Clase Othello para garantizar su corrección (**Figura 02**).

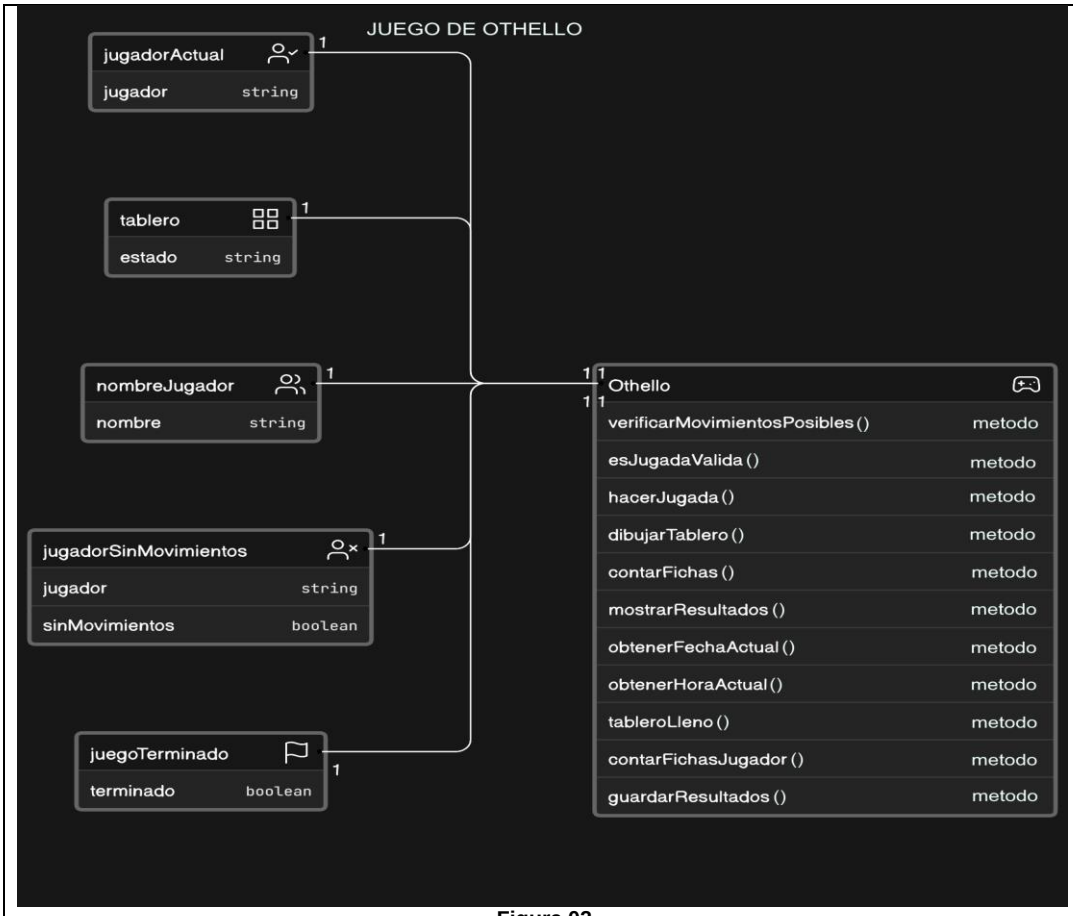

Figura 02

Experiencia de Aprendizaje:

- La experiencia de aprendizaje implicó comprender en profundidad las reglas y mecánicas del juego de Othello, aplicadas en un juego de consola C++.
- Se adquirieron habilidades en diseño de software, estructuras de datos y algoritmos para representar y gestionar el juego.
- Se mejoró la comprensión de C++ y la programación orientada a objetos en la práctica.
- Se fortaleció la habilidad de documentar y comentar el código de manera clara.

Problemas de Desarrollo que se Afrontó:

- La elección de la estructura de datos para representar el tablero fue un desafío, ya que debía ser eficiente y permitir la fácil verificación de jugadas válidas.
- La implementación de algoritmos para verificar la validez de las jugadas y capturar fichas del oponente requería un pensamiento lógico y detallado.
- La detección de fin de juego y el recuento de fichas fue una parte crítica, ya que involucraba múltiples condiciones y cálculos.

Evolución de la Solución:

- La solución evolucionó desde una idea conceptual hasta una implementación funcional y probada.
- Se incorporaron mejoras en la interfaz de usuario y mensajes claros para una experiencia más amigable.

Consideraciones:

La modularidad en el diseño del código facilitó la gestión de diferentes aspectos del juego, como la lógica, la entrada/salida y la visualización.
Se prestaron especial atención a las reglas del juego de Othello para garantizar que la aplicación cumpliera con precisión con ellas.
La documentación y los comentarios en el código fueron esenciales para facilitar el mantenimiento y comprensión del programa por otros desarrolladores.

Resultado del Aprendizaje Alcanzado durante el Desarrollo:

- La experiencia de desarrollo proporcionó un entendimiento profundo de cómo abordar problemas de software complejos.
- Se fortalecieron las habilidades en C++ y programación orientada a objetos.
- Se adquirieron conocimientos sobre la implementación de algoritmos y estructuras de datos en situaciones prácticas.
- Se mejoró la capacidad de documentación y comunicación efectiva en el desarrollo de software

5. Referencias

[1] "Othello Programming." Stanford Encyclopedia of Philosophy. 2017.
[Enlace:https://plato.stanford.edu/archives/fall2017/entries/games-and-gaming/othello-programming.html]

[2] Stanford University. "Programming Othello (Reversi)."
[Enlace:https://web.stanford.edu/class/cs221/assignments/reversi/]

Diagrama (Anexo 01)