INFORME DE PROYECTO

Parcial 2 – Informática II

Elaborado por Sergio Berbesí Builes y Salomé Bermúdez Macias

Contextualización del Problema

El proyecto se enfoca en la creación de un juego inspirado en el juego de Othello por medio de la programación orientada a objetos, un juego de estrategia para dos jugadores (cada uno con fichas blancas o negras) que se juega en un tablero de 8x8 casillas. El objetivo del juego es capturar las fichas del oponente logrando llenar el tablero con más fichas de su color que su oponente al final de la partida. Othello es un juego desafiante que requiere de estrategia y pensamiento táctico, lo que lo convierte en un interesante proyecto para la implementación de lógica y algoritmos de juego.

El programa debe gestionar las jugadas, verificar la validez de las mismas y determinar al ganador al final de la partida.

Análisis

Para abordar el desarrollo del juego de Othello, se utilizó programación orientada a objetos (POO) en C++ empleando una estructura de clases para representar los elementos del juego lo que permite un código modular, en el que cada módulo se encarga de llevar una tarea específica debido. La POO es eficiente en el manejo de estructuras de datos, lo que es fundamental para representar el tablero, las fichas y las reglas del juego.

A continuación, se describen las principales consideraciones y estrategias de solución:

- Representación del Tablero: El tablero se representa mediante una matriz de casillas, donde cada casilla puede estar vacía, ocupada por una ficha blanca o por una ficha negra.

- Jugadores: Se implementa una clase Jugador que representa a los dos jugadores (blanco y negro) con un atributo que almacena su color.

- Turnos de Jugadores: Los jugadores se alternarán en sus turnos, y el programa solicitará al jugador en turno que ingrese su jugada.

- Movimientos Válidos: Se verifica la validez de un movimiento antes de realizarlo. Un movimiento se considera válido si se cumple con las reglas del juego: rodear fichas del oponente con fichas del jugador actual en las multiples direcciones.

- Reglas del Juego: Las reglas del juego se implementan mediante algoritmos que determinan si un movimiento es válido y permiten realizar cambios en el tablero, volteando las fichas del oponente capturadas.

- Finalización de la Partida: La partida se da por finalizada cuando no hay más movimientos posibles para ninguno de los jugadores o cuando el tablero se llene por completo.

- Determinación del Ganador: Se determina el ganador contando las fichas de cada color al final de la partida.

Estrategia de Solución:

La estrategia de solución se basa en el siguiente sistema:

1. **Inicialización:** Se crea un tablero de juego y se colocan las cuatro fichas iniciales en el centro.
2. **Turnos de Jugadores:** Los jugadores alternan sus turnos. En cada turno, un jugador realiza un movimiento válido seleccionando una casilla vacía en el tablero.
3. **Validación de Jugadas:** Se verifica si el movimiento es válido y, en caso afirmativo, se aplican las reglas para cambiar el color de las fichas del oponente capturadas.
4. **Continuación y Finalización:** El juego continúa hasta que no haya más movimientos posibles o el tablero esté lleno. Luego, se determina el ganador según la cantidad de fichas de cada color.

Cada jugador se representará como una instancia de la clase "Jugador" con un atributo de color (blanco o negro).

El tablero se gestionará a través de la clase "Tablero", que contendrá una matriz de casillas y proporcionará métodos para inicializar el tablero, mostrarlo, realizar jugadas y determinar el ganador.

La lógica detrás de los principales algoritmos se basará en la detección de jugadas válidas y la captura de fichas del oponente en múltiples direcciones. Las jugadas se realizarán utilizando un enfoque de "fichas a cambiar" en las direcciones válidas.

Es fundamental comprender las reglas del juego de Othello para detectar jugadas no validas

- Los jugadores se dividen en blancos y negros, siendo el blanco el color del usuario.
- Al inicio, el tablero contiene 4 fichas en el centro, dispuestas dos fichas blancas y dos fichas negras, en diagonal.
- Las fichas deben ser colocadas en forma de sándwich de manera que encierren una o más fichas del oponente en cualquier dirección (vertical, horizontal o diagonal).

- Este encierro hará que las fichas encerradas sean cambiadas al color del jugador actual.
- Si un jugador no puede realizar un movimiento válido que encierre fichas del oponente, debe pasar su turno.

Diseño

El diseño del código está estructurado en una serie de clases que representan los elementos del juego. Además, hay una clase Juego que orquesta la partida y almacena los resultados. A continuación, se describen algunos aspectos clave del diseño:

- El código utiliza clases para representar conceptos como Jugador, Casilla y Tablero, lo que facilita la organización y el mantenimiento del código. Cada clase tiene métodos que realizan acciones relacionadas con ese concepto.

- Se utiliza una estructura de datos vector<vector<Casilla>> para representar el tablero del juego. Esto permite acceder y modificar fácilmente las casillas en el tablero.

- Se ha implementado un algoritmo para determinar si una jugada es válida, lo que implica verificar si se pueden voltear fichas del oponente en las direcciones adecuadas.

- El juego sigue un bucle principal en la función main(), que permite jugar múltiples partidas hasta que el jugador decida no jugar más. Los resultados de cada partida se guardan en un archivo de texto.

- La clase Juego está diseñada para manejar el flujo del juego, mostrar el tablero, solicitar las jugadas de los jugadores y determinar al ganador al final de la partida. También proporciona la funcionalidad para guardar la información de la partida en un archivo de resultados.

El diseño del proyecto se organiza en las siguientes clases:

Jugador

Atributos:

- Color (blanco o negro)

Métodos:

- hacer_jugada(): Realiza una jugada en el tablero.

- es_movimiento_valido(): Verifica si un movimiento es válido.

Casilla

Atributos:

- Estado (vacia, blanca, negra)

Métodos:

- establecer_estado(): Establece el estado de la casilla.
- obtener_estado(): Obtiene el estado de la casilla.
- esta_vacia(): Verifica si la casilla está vacía.
- es_blanca() y es_negra(): Verifica si la casilla es blanca o negra.
- es_del_oponente() y es_del_jugador(): Verifica si la casilla pertenece al oponente o al jugador.

Tablero

Atributos:

- Matriz de casillas (8x8)

Métodos:

- inicializar_tablero(): Inicializa el tablero con las fichas iniciales.
- mostrar_tablero(): Muestra el tablero en la consola.
- hacer_jugada(): Realiza una jugada en el tablero y captura fichas del oponente.
- jugador_puede_jugar(): Verifica si un jugador puede realizar una jugada.
- determinar_ganador(): Determina al ganador de la partida.

Juego

Atributos:

- Tablero
- Jugador blanco
- Jugador negro

Métodos:

- jugar_partida(): Controla el flujo de una partida.
- guardar_resultados(): Guarda los resultados de la partida en un archivo.
- obtener_fecha_actual(): Obtiene la fecha y hora actual.

Algoritmos Implementados

El código implementa diferentes algoritmos para:

- Realizar jugadas válidas.
- Capturar fichas del oponente en múltiples direcciones.
- Determinar al ganador de la partida.

El código completo, incluyendo los algoritmos implementados, se encuentra en el repositorio del proyecto: https://github.com/berbu25/Parcial-2.git

La lógica detrás de los algoritmos implementados en las clases es:

- Jugador::hacer_jugada(): Este método permite a un jugador realizar un movimiento. Verifica si la casilla seleccionada es válida y si es posible capturar fichas del oponente en las direcciones correspondientes.

- Tablero::hacer_jugada(): Este método verifica si una jugada en una casilla específica es válida y, en caso afirmativo, cambia las fichas del oponente capturadas en las direcciones correspondientes.

- Tablero::jugador_puede_jugar(): Este método verifica si un jugador tiene movimientos válidos disponibles en el tablero.

- Tablero::determinar_ganador(): Este método determina el ganador de la partida contando las fichas de cada color en el tablero.

Experiencia de Aprendizaje

Durante el desarrollo de este proyecto, se han enfrentado varios desafíos y se han aprendido importantes lecciones entre ellas están:

- Diseño Orientado a Objetos: Se ha profundizado en la programación orientada a objetos, lo que ha permitido organizar el código de manera más eficiente y reutilizable.

- Algoritmos del juego: La implementación de algoritmos para determinar jugadas válidas y capturar fichas del oponente ha requerido un profundo entendimiento de las reglas del juego.

- Gestión de Archivos: Se reforzó el conocimiento sobre la escritura y lectura de archivos para guardar los resultados de las partidas.

- Interacción con el usuario: La interacción agradable y amena con el usuario, permitiendo una experiencia fácil y ágil.

- Resolución de Errores: La depuración y corrección de errores fue una parte integral del proceso de desarrollo y una oportunidad para fortalecer las habilidades de resolución de problemas.

- Trabajo en Equipo: La colaboración entre el equipo en la planificación y desarrollo del proyecto.

El proyecto de desarrollo del juego de Othello ha sido una experiencia de aprendizaje en la aplicación de conceptos de programación y estructuras de datos en un proyecto de software. Se logró implementar un juego funcional con reglas y lógica de juego precisas. Además, se mejoraron habilidades de diseño orientado a objetos, algoritmos de juego y manejo de la interfaz de usuario en consola.

Este proyecto representa una introducción al mundo de la programación de juegos y brinda una base sólida para futuros desarrollos y la exploración de juegos más complejos. El juego de Othello no solo es un desafío para los jugadores, sino también para los desarrolladores que trabajan en su implementación. Proporciona una base sólida para futuras mejoras y adiciones.