Universidad de Antioquia - Facultad de Ingeniería



Parcial II

Bohórquez Ardila, Luisa María - Morón Acacio, David Alejandro

Informática II

Docentes a cargo: Guerra Soler, Aníbal José - Salazar Jiménez, Augusto Enrique

Medellín, Antioquia. 2023

1. **Contextualización del problema:**

Se tiene para la presente entrega el programar un Othello, juego de estrategia de mesa para dos personas, escrito en el lenguaje C++ direccionado a la POO, teniendo en cuenta la eficiencia del programa y el correcto uso de la memoria, así como el manejo de excepciones y la documentación adecuada del programa. Entonces, el reto consiste en diseñar un tablero de 8x8 (donde se muestre todo el tiempo el estado de todas las fichas), cuyas posiciones se van actualizando según las fichas que jueguen los dos jugadores que estén involucrados en el juego. También, debe ser posible guardar un historial de partidas en un archivo de texto plano, con detalles sobre la fecha y la hora, el ganador y las cifras que cada jugador logró.

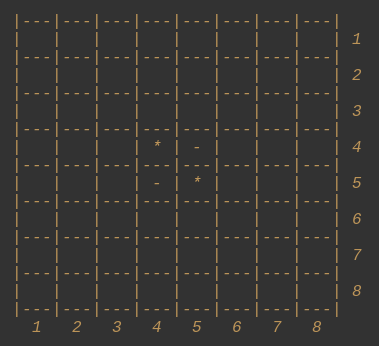
1. **Consideraciones a tener en cuenta:**

Se debe desplegar un cuadro de texto que indique las reglas del juego, por tanto el juego entero debe estar en un loop que se detenga sólo cuando una partida termina y algún usuario decida no jugar otra partida. Además del cuadro de texto con las reglas, se tiene que imprimir por pantalla el tablero de juego (el cual compondremos a base de guiones y barras que den la ilusión de casillas) acompañado de las coordenadas laterales y verticales (para que los jugadores puedan elegir dónde poner su ficha), de manera que sea visible y claro para el usuario lo que está pasando. También. al momento de comenzar el juego, ambos jugadores podrán registrar su nombre y se darán los colores de las fichas a cada uno (en cuanto se registren, el programa será quien asigne cuál juega con fichas negras y cuál juega con fichas blancas por orden de llegada, siendo respectivamente guiones y asteriscos).

Teniendo en cuenta todas las validaciones de los movimientos y posiciones legales el el juego, así como las validaciones de entrada por consola, sólo resta tener en cuenta la disposición del histórico de jugadas, donde se guardan los nombres de los jugadores, las fichas que acumularon y quién ganó, todo en un archivo de texto que debe estar actualizándose cada vez que se juega. En el archivo, por cada jugada se irá añadiendo un bloque de texto en el que se registren los datos por partida, el formato se debe dar como:

| OTHELLO  Fecha y hora: //aquí se registra la fecha y la hora de inicio de la partida  Negras(“nombrejugador”): #fichas  Blancas(“nombrejugador): #fichas  Ganador: “nombrejugador”  …//las partidas posteriores |
| --- |

Dado que todo va a construirse a base de arreglos de arreglos, punteros, memoria dinámica, clases y librerías con manejo de excepciones y archivos de texto, se tendrán en cuenta los headers donde se guarden las funciones de mostrar el “tablero”, cambiar el contenido del “tablero” (el objeto que modela todos los arreglos que contienen los caracteres que van cambiando según la jugada), poder recibir las jugadas y poder verificarlas, validaciones, entre otros auxiliares, y en el main iría todo lo referente a las invocaciones de dichas funciones y el menú del juego. El tablero se formaría con arreglos que sólo posean los caracteres jugados en las posiciones indicadas, y a medida que se va imprimiendo el tablero, se van concatenando guiones, barras y espacios en blanco para lograr una interfaz de juego visualmente amigable para el usuario, mostrando la siguiente figura:



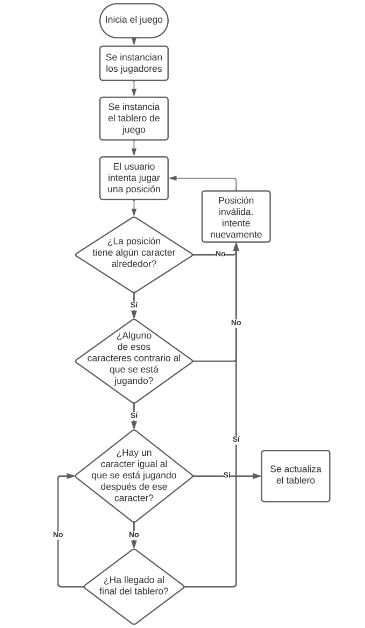
1. **Diseño de funciones, clases y algoritmos:**

Vamos a definir dos clases:

1. Clase Jugador, cuyos atributos serán el tipo de ficha (es decir, el caracter que van a estar jugando), el nombre del jugador (ingresado por el usuario) y el estado de victoria (si al final de la partida ganó o perdió), y cuyos métodos serán el recibir el nombre que ingrese el usuario, definir el caracter que está representando a su personaje, y actualizar la variable que verifica si ganó o perdió (recibiendo al final de la partida el número de fichas acumuladas).
2. Clase Tablero, cuyo atributo será una matriz que almacene todo el tablero, y cuyos métodos sean actualizar la matriz (de manera que, cada vez que un jugador ingrese una jugada, el método se encargue de cambiar los caracteres necesarios de la matriz, y de indicarle al usuario que es un movimiento inválido en caso de que no pueda hacer nada), un método que imprima en pantalla la matriz para indicar el estado del juego, y un método que contabilice los caracteres al final de cada partida.

De esta manera, mediante una función llamada *verificarGanador*, sólo se deberán leer los atributos de las instancias de las clase Jugador para saber quién ganó y con cuantas fichas.

Entonces el programa comienza, se inicializan las clases, se pide al usuario que ingrese su nombre y se comienza a definir el objeto, empieza el juego, luego se comienza el ciclo en el que el juego se dedica a evaluar los elementos que hay en las posiciones de los arreglos alrededor de las posiciones ingresadas por el usuario para verificar si cambia un caracter, o si no puede poner una ficha en esa posición. Entonces se evalúan las posiciones por debajo, por arriba, por derecha, por izquierda, por la diagonal inferior izquierda, diagonal inferior derecha, diagonal superior derecha y diagonal superior izquierda. Nuestro algoritmo va a basarse en el siguiente diagrama:



Es decir, se va a buscar posición por posición siempre y cuando hayan fichas contrarias, hasta llegar a una posición vacía o al final del tablero. De esta forma, se garantizan las jugadas válidas y al mismo tiempo se realizan dichas jugadas, se actualizan los objetos y se guardan para ser imprimidos en pantalla, hasta que termine la partida y se guarde el registro de los atributos de los objetos en el archivo de texto en el formato que está indicado, también en tiempo de ejecución, para que se puedan jugar múltiples partidas continuas y que cada una se guarde en el archivo con los datos especificados (es decir, que no deba cerrarse el programa para tener que guardar las partidas, sino que se vaya escribiendo el archivo al terminar cada partida).

Funciones a diseñar:

1. leerTiempo(recibe la hora y la fecha de la partida y la guarda en el archivo)
2. **Método** recibirPosicion(recibe la coordenada de la posición que juegue el usuario)
3. **Método** evaluarPosicionIngresada(evalúa en la matriz la jugada que se ingresó)
4. **Método** cambiarCaracter(cambia los caracteres cuando se cumplen las condiciones de la jugada)
5. **Método** recibirUsuario(recibe el nombre del jugador)
6. **Método** leerUsuario(lee la información contenida en el objeto)
7. **Método** contarFichas(cuenta las fichas de cada jugador al final de la partida)
8. **Método** leerTablero(lee la información de la cantidad de fichas al final de cada partida)
9. comprobarArchivo(verifica que exista el archivo de registro)
10. escribirArchivo(escribe la información en el archivo de registro)
11. **Método** imprimirTablero(va a imprimir la serie de barras, guiones y espacios concatenados con los elementos de la matriz del juego)
12. validarEntradas(validaciones de entrada de datos)
13. **Implementación de algoritmos: (enlace a github)**
14. **Observaciones del desarrollo y conclusiones:**