INFORME TRABAJO PRÁCTICO FINAL

PROGRAMACIÓN 2

Alumno : Alenny Tiziano

Explicacion del codigo en C :

Antes de explicar cómo el proceso de resolución y las decisiones tomadas creo conveniente sintetizar lo que debe hacer el **Programa en C**

Programa en C :

El programa en C simula una partida de **Othello** y valida que las jugadas realizadas sean correctas a partir de un archivo de texto de entrada.
Este archivo contiene :

- El nombre de los jugadores junto con su color en partida
- El color que comienza la partida
- Las jugadas realizadas en orden

El programa deberá leer el archivo de entrada y verificar que el formato del nombre de los jugadores junto con su color sea correcto, así como también que el formato del color inicial sea válido.

En caso de que el archivo cumpla con el formato, se procederá a simular la partida leyendo y ejecutando jugada a jugada.

Si durante el procesamiento se detecta una jugada inválida, el programa dejará de leer el archivo, indicará quién realizó la jugada incorrecta y mostrará el estado del tablero hasta ese momento, sin incluir la jugada.
Si no se encuentra ninguna jugada inválida, el programa deberá determinar si la partida ha finalizado.

- En caso de que el juego haya terminado se indicará el ganador
- En caso contrario , se generará un archivo mostrando el tablero y el color del jugador que continúa jugando.

Código en C

Antes de comenzar a programar, el primer paso fue pensar qué elementos debían ser representados, qué procesos debía realizar el código y cuál sería la forma más simple y clara de implementarlos.

El programa realiza modificaciones constantes sobre un **tablero**. Dado que Othello se juega sobre un tablero de **8 × 8**, se decidió representarlo mediante una **matriz de caracteres de 8 filas por 8 columnas**, Esta estructura permite:

- Acceso directo a cualquier posición del tablero.
- Recorridos eficientes por filas, columnas y diagonales.
- Una representación clara del estado del tablero.

Validación de jugadas

El proceso más importante del programa es la **validación de una jugada**.
En una primera implementación, hice que la validación de la jugada y la aplicación de los cambios sobre el tablero se realizaban dentro de una misma función . Sin embargo, esta solución resultó poco clara y no representa correctamente la finalidad de cada función.

Por este motivo, decidí **dividir el proceso en dos etapas independientes**. Validar si la jugada es correcta. y luego aplicar los cambios al tablero

Para implementar esta separación se utilizaron dos estructuras

- Una estructura para representar **posiciones del tablero** (fila y columna).
- Una estructura para almacenar la **lista de fichas que deben ser modificadas** como resultado de una jugada válida (lista de cambios).

Para acotar la Lista de cambios se investigó cuál es cambios que puede realizar una jugada , y dado que este es limitado y pequeño no se requiere de memoria dinámica

De esta manera, el programa puede determinar si una jugada es válida **sin modificar el tablero**, y solo en caso de serlo aplicar los cambios correspondientes.

Lógica de validación

Para determinar si una jugada es válida, la función recorre las **8 direcciones posibles**(dos horizontales, dos verticales y cuatro diagonales). a partir de la posición jugada .

En cada dirección se realiza este procedimiento:

- Se avanza en la dirección (arriba, abajo, diagonales, etc.).
- Se verifica si la casilla adyacente está ocupada por una ficha del oponente.
- Si es así, se continúa avanzando en esa dirección guardando temporalmente las posiciones recorridas hasta encontrar una ficha del propio color o una casilla vacía (o salir del tablero).

Si se encuentra una ficha del propio color encerrando las fichas del oponente, la dirección se considera válida y las fichas encerradas se agregan a la lista de cambios. Si la lista final tiene al menos un cambio, la jugada es válida.

Opté por esta forma ya que, en mi primera implementación del código, debía crear copias del tablero porque no había separado correctamente las etapas del proceso.

Explicacion del codigo en Python:

Antes de explicar cómo el proceso de resolución y las decisiones tomadas creo conveniente sintetizar lo que debe hacer el programa en python

Programa en Python :

El programa en Python toma como entrada el archivo generado por el programa en C junto con el color que usará la persona para jugar y el nivel del programa.
Lo primero que hará es imprimir el tablero. Luego se jugará al othello usando lo datos que hay en el archivo generado (tablero , color actual)

Turno del jugador:

Se le pedirá que ingrese la jugada y se validará si la misma es correcta , en caso de no serlo se pedirá que vuelva a ingresar una jugada . Si la jugada es válida se imprime el tablero resultante.

Turno del programa :

El programa contará con 2 niveles de dificultad , el nivel 0 elige jugadas aleatoriamente , mientras que el nivel 1 elige la que más cambios genera en el tablero.

Sin importar el nivel siempre se imprimirá la jugada elegida junto con el tablero resultante.

Código en Python

Como el código de python necesita hacer cosas similares al código de C , no hubo mayor complicación que transcribir el código y las funciones.

Para que el programa pueda elegir sus movimientos, primero es necesario conocer todos los movimientos posibles en ese momento. Esto se logró recorriendo el tablero y probando las posiciones vacías con una función idéntica a la lógica de validación utilizada en C.

Aproveché esta lista de movimientos posibles para resolver tres problemas al mismo tiempo:

- El programa usa la lista para elegir su jugada (tanto nivel 0 como nivel 1)
- Si la lista está vacía , se pasa de turno automáticamente tanto para el jugador como para el programa
- En el turno de la persona , no es necesario validar si la jugada ingresada es válida , basta con verificar que esta esté dentro de la lista de movimientos

Cómo ejecutar los programas

1. Programa en C (Juego Principal)

Compilación Desde la carpeta raíz othello, ejecuta el siguiente comando para compilar:

```
gcc -Wall -Wextra -Werror programa/Main.c programa/othello.c -o othello.exe
```

(Esto generará un archivo ejecutable llamado othello.exe dentro de la carpeta)

Ejecución Para correr el programa, ejecuta el siguiente comando en othello:

```
.\othello.exe partida.txt
```

(Dónde partida.txt es el archivo de entrada que contiene la partida a validar).

2. Tests en C

Compilación Desde la carpeta othello, ejecuta:

```
gcc -Wall -Wextra -Werror TEST/test.c programa/othello.c -Iprograma -o test_othello.exe
```

Ejecución Para correr los tests, utiliza:

```
.\test_othello.exe
```

3. Programa en Python

Ejecución para correr el programa , ejecuta el siguiente comando con los argumentos necesarios :

ejemplo :

```
python programa/othello.py partida_guardada.txt B 1
```

4. Tests en en Python

Para correr los test se utiliza desde la carpeta othello :

```
pytest
```

(Se incluye un archivo.py vacío para permitir la correcta ejecución de pytest dado que los test están en distintas carpetas)

(1) https://othelloacademy.weebly.com/rules.html
(2) https://stackoverflow.com/questions/11152160/initializing-a-struct-to-0
(3) https://unstop.com/blog/extend-in-python#:~:text=Frequently%20Asked%20Questions-,Python%20extend()%20Function%20%7C%20Syntax

%2C%20Techniques%20&%20More%20(+Examples,in%20place%20and%20returns%20None.&text=In%20Python%2C%20the%20extend(),for%20using%20extend()%20effectively.
(4) https://www.geeksforgeeks.org/c/character-arithmetic-c-c/
(5) https://stackoverflow.com/questions/12582503/test-discovery-failure-when-tests-in-different-directories-are-called-the-same
(6) https://keepcoding.io/blog/como-generar-una-excepcion-valueerror-en-python/