REVERSI!

GRUPO 2

Miguel Ferreiro Díaz
Sara Del Río Muñoz
Juan Carlos Fernándes González
Alberto Rumbo Nogueira

Tabla de contenidos:

Tabla de figuras:

1.INTRODUCCIÓN

El Reversi enfrenta a dos jugadores que desarrollan el juego sobre un tablero .Se utilizan además 64 fichas iguales de dos colores: el anverso de un color y el reverso de otro diferente (en nuestro caso usaremos los caracteres '**O**' y '**#**' para diferenciar las fichas de los jugadores.

El objetivo del juego para cada jugador es acabar la partida con un número de fichas de su color sobre el tablero superior al de su oponente.

Los movimientos consisten en incorporar fichas al tablero a razón de una por turno, nunca en desplazar fichas de las que ya estuvieran sobre el tablero.

Las incorporaciones deberán hacerse en orden a las siguientes normas:

-Sólo podrá incorporarse una ficha flanqueando (Por flanquear se entiende el hecho de colocar la nueva ficha en un extremo de una hilera de fichas del color del contrario (una o más fichas) en cuyo extremo opuesto hay una ficha del color de la que se incorpora, sin que existan casillas libres entre ninguna de ellas. Esta hilera puede ser indistintamente vertica1, horizontal o diagonal. De este modo, las fichas del contrincante quedan encerradas entre una que ya estaba en el tablero y la nueva ficha.) a una o varias fichas contrarias.

-Cada vez que un jugador incorpora una ficha, y por lo tanto encierra a otras del contrario, debe dar la vuelta a las fichas encerradas convirtiéndolas así en propias.

-Si en una sola incorporación se provocase esta situación de flanqueo en más de una línea, se voltearán todas las fichas contrarias que estuvieran implicadas en cada uno de los flanqueos.

-Si no fuera posible para un jugador encerrar a ninguna ficha, deberá pasar en su turno, volviendo el mismo a su oponente.

La partida finaliza cuando todas las casillas del tablero son ocupadas o ninguno de los 2 jugadores tiene posibilidad de incorporar una nueva ficha.

2.ALGORITMO

Al iniciar el juego, aparecerá en pantalla el menú principal, la cual está encuadrada en una función (**Menu_principal**) . Esta se encarga de pedirle al usuario la opción que desea escoger, comprueba que es una opción válida y devuelve ese valor. Si el usuario introduce una opción fuera de las que se le presentan, se le volverá a mostrar el menú avisando de que ha introducido una opción no válida. Entre las opciones a escoger están la de Nueva partida, cargar partida, instrucciones y salir del juego.

Luego de escoger una opción se devolverá esta al main, donde se ejecutará la opción escogida. Si desea empezar una Nueva partida lo que se hará es llamar a la función **Inicializar_tablero**, en la que cual se limpiará el tablero, se colocarán las 4 fichas iniciales (2 de cada jugador en diagonal) y se inicializará el contador de número de fichas a 4.

Si desea cargar una partida, se llamará a la función (**Cargar_partida**) donde se comprueba que sí existe una partida guardada, se cargará. Esto se realiza a través de ir poniendo en cada celda de la matriz tablero lo que hay en cada posición de la partida guardada anteriormente, además se coge del fichero un 1 o un 0 para saber a qué jugador le toca mover. También contará el número de fichas que hay ,a través de dos bucles, para modificar la variable número de fichas. Luego de realizar todo esto, se cerrará el archivo y se comprueba que se realiza sin problemas, se saldrá de la función y se reanudará la partida como hubiera sido guardada anteriormente.

Después de esto, se llama a la función partida, la cual hace que se cambie de jugador (haciendo que en cada turno, la ficha del jugador cambie), se muestra por pantalla la posición actual del tablero a través de la función **Visualización_tablero**, la cual dibuja el tablero e imprime por pantalla las nuevas fichas introducidas con los respectivos cambios de fichas,y que en cada turno de cada jugador, cada uno haga su movimiento. Para esto último lo que hace es llamar a la función **Turno_jugador**. Esta función estará cambiando de jugador mientras el número de fichas no llegue al máximo de fichas que se pueden colocar dentro del tablero o si el contador de movimientos válidos sea 2, ya que significa que los dos jugadores han tenido que pasar porque no pueden realizar más jugadas válidas.

En la función **Visualización_tablero** se imprimirá el tablero y las fichas que hayan sido guardadas en la variable **char tablero[TAMANHO][TAMANHO]**.

En la función **Turno_jugador**, lo primero que se comprueba es si hay movimientos posibles y esto se realiza a través del valor que devuelve la función **Validar_movimientos**. Si lo que devuelve es un 0, se pasa de turno porque significa que no hay movimientos válidos y suma uno al contador de movimientos invalidos y pasará de turno.Si los dos jugadores no pueden mover, se terminará la partida. Si hay movimientos válidos, es decir, se ha devuelto un número distinto de 0, se le pide al usuario que introduzca la fila y la columna de su jugada. Se adaptan las coordenadas de la jugada introducida al array del tablero ya que esta cuenta las filas y las columnas desde 0. Si la jugada introducida no es correcta, se avisa por pantalla que esa jugada no es válida y se le vuelve a pedir al usuario que vuelva a introducir su jugada hasta que la jugada sea correcta.

Si la jugada introducida es correcta, es decir está dentro del tablero y es movimiento válido (comprobado con la función **Validar_movimientos** que devuelve el array **Movimientos_validos[TAMANHO][TAMANHO]** en el cual están las casillas candidatas para realizar un movimiento, estas casillas están marcadas con un 1), se llama a la función **Hacer_movimientos** en la que se introduce la ficha del jugador en la casilla de su coordenada y se cambian las fichas flanqueadas del contrario por fichas del jugador, es decir se modifica el array **tablero[TAMANHO][TAMANHO].**
Además se le suma uno a la variable que lleva la cuenta de las fichas que hay en el tablero, se vuelve a poner a cero el contador de las jugadas inválidas ya que significa que el siguiente jugador ha podido mover y se cambia de jugador.

En la función **Validar_movimientos** se devolverá un array llamado ***Movimientos_validos[TAMANHO][TAMANHO]*** que contendrá 1s y 0s en las casillas donde es posible realizar un movimiento (es decir, esté en blanco y pueda moverse una ficha del jugador a esa casilla). Para comprobar si esa casilla es candidata a ser una casilla válida, se comprueba si hay fichas del oponente

alrededor y luego fichas del propio jugador para que haya la posibilidad de flanquearlas, si esto es posible introduce un 1 en el array anteriormente mencionado y se suma en 1 el contador de número de movimientos válidos para indicar que es posible realizar una jugada.

Luego de realizar la jugada, se llama a la función **Menu_juego** y le da al usuario la opción de abrir el menú del juego en pantalla . Si decide abrirlo, le aparecerá y le dá la opción de guardar la partida, cargar una partida anterior, hacer que la opción de abrir el menú del juego no vuelva a aparecer, salir de la partida o continuar la partida.

Si se escoge la opción de guardar se llamará a la función **Guardar_partida** en la cual modificará el archivo partida.txt,si existe, o se creará un archivo nuevo, si no existe, y se introducirán lo que hay en cada una de las celdas del array tablero, fichas de cada jugador y espacios, y al final del fichero se le introducirá un valor (0 o 1) que indicará el turno del jugador que le toque jugar al cargar la partida y así poder proseguir la partida como estaba. Además se comprueba si el archivo se ha abierto y cerrado correctamente.

Si se escoge la opción de cargar se llamará a la función **Cargar_partida**, anteriormente explicada, y si no existe ninguna partida guardada avisará al usuario y volverá al menú del juego.

Después de realizar el movimiento o en el caso de que no haya movimientos posibles, pasará al turno del otro jugador, el cual hará lo mismo que el primer jugador. Esto se realizará hasta que se llene el tablero de fichas o los jugadores tengan que pasar dos veces de turno, en cuyo caso finalizará la partida y se le mostrará al usuario por pantalla el tablero final. Además se llamará a la función **puntuación**, la cual contará las fichas e imprimirá la puntuación de cada jugador. Después volverá a ejecutarse el main, es decir, volverá al menú principal.

3.Programa:

3.1 Librerías utilizadas:

#include <stdio.h> ⮕Utilizamos esta librería para definir tres tipos de variables, varias macros y diversas funciones para la realización de entrada y salida.

#include <stdlib.h> ⮕Utilizamos esta libreria para definir cuatro tipos de variables, varias macros y diversas funciones para realizar funciones generales.

#include <locale.h> ⮕ Utilizada para utilizar la función *SetLocale* que nos permite añadir los acentos y que estos se visualicen por pantalla.

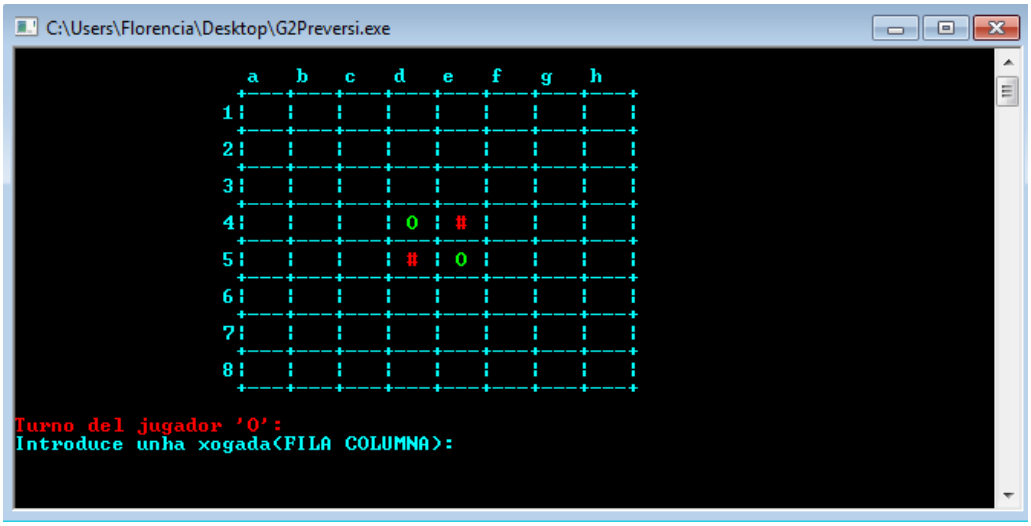
#include <windows.h> ⮕ Librería que nos permite cambiar el color del texto que aparece por pantalla.

#include <time.h> ⮕ Librería utilizada para poder definir variables de tipo clock.

3.2 Funciones empleadas:

 void Visualizar_tablero(char tablero[TAMANHO][TAMANHO]) ➞

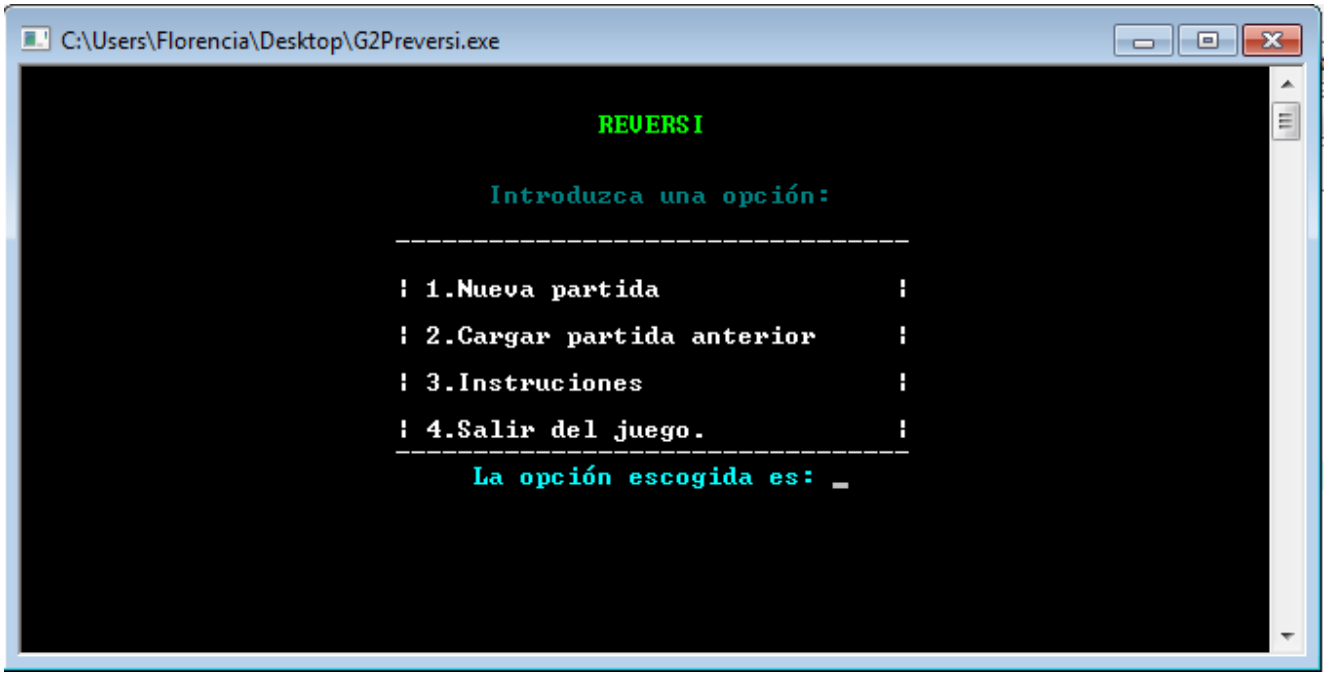
Es una función de tipo void (no devuelve ningún valor) y se le pasa como argumentos la variable **char tablero[TAMANHO][TAMANHO]** la cual es pasada por referencia y no por valor al ser un array (esto significa que los cambios que se realicen dentro de la función hará que se modifique el contenido del array).En esta función el array tablero no se modifica.
Esta función dibuja el tablero y lo muestra en pantalla. En esta función se imprime en pantalla el tablero con la posición actual, pasada a través del array tablero, que tiene las fichas de cada jugador. Además introducirá las fichas de un color diferente para cada jugador.



 int Menu_principal() ➞

Es una función tipo entero (devuelve un valor entero) y no se pasan argumentos. Devolverá la opción escogida por el usuario.

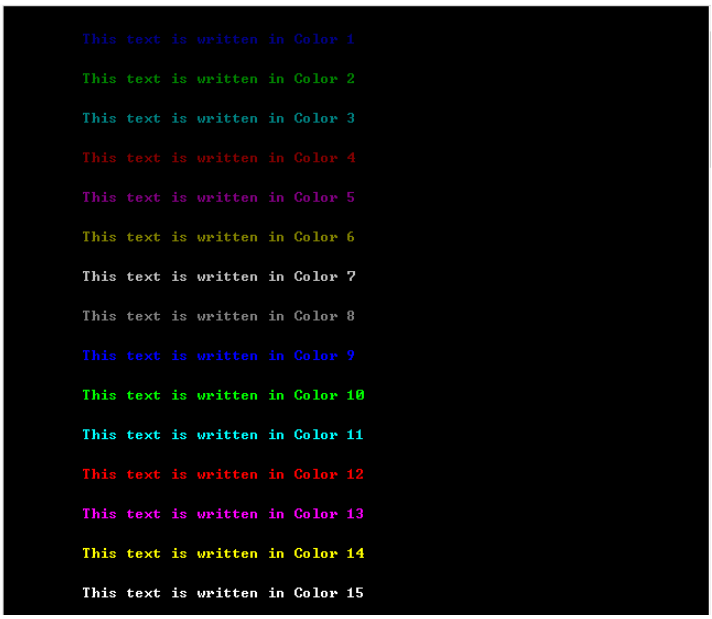
En esta función se imprimen las opciones que puede escoger el usuario al comienzo del programa. Si escoge la opción instrucciones, se le imprimirá por pantalla las instrucciones del juego, y si escoge otra opción esta función devolverá el valor asociado a esa opción. Si la opción introducida por el usuario es incorrecta, se volverá a pedir que introduzca la opción.



 void SetColor(int Color) ➞

Es una función de tipo void (no devuelve ningún valor) y se pasa como parámetro **int Color.**

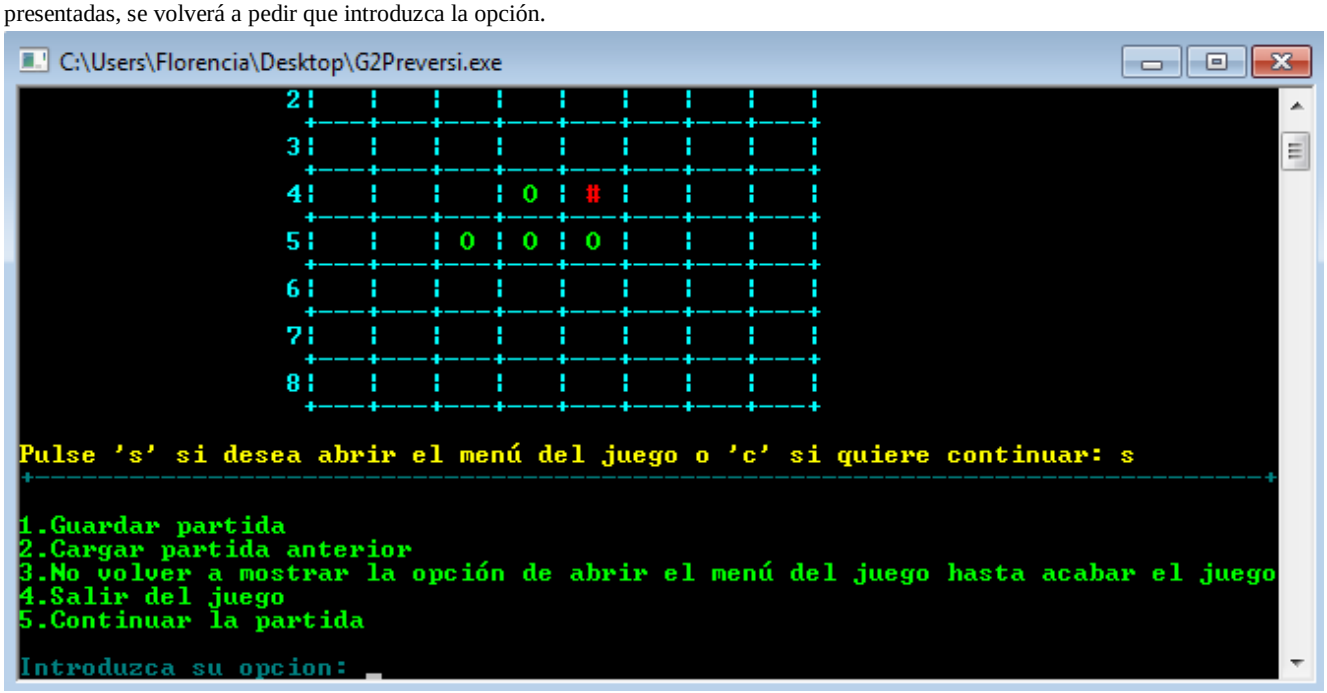
Esta función nos sirve para cambiar el color al texto mostrado en pantalla, en la llamada de la función se le pasa un argumento de tipo entero con un número del 1 al 15 que está asociado a un color.



 int Menu_juego (char tablero[TAMANHO][TAMANHO],struct Estado E) ➞

Es una función de tipo entero (devuelve un valor entero) y se le pasan como parámetros la variable **char tablero[TAMANHO][TAMANHO]** la cual es pasada por referencia y no por valor al ser un array (esto significa que los cambios que se realicen dentro de la función hará que se modifique el contenido del array). En esta función el array tablero no se modifica. También se pasa la estructura Estado E. El valor que devuelve la función es la opción escogida por el usuario.

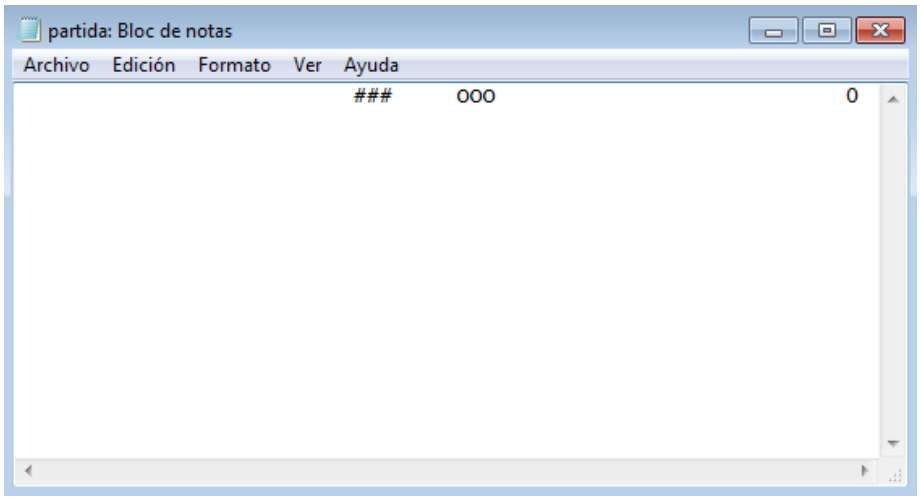
Esta función permite colocar un menú si el jugador lo desea (tiene la opción de decidir si se muestra o no) tras cada jugada. En el cual se le da la opción al usuario de escoger las opciones que se muestran en la imagen. Si introduce cualquier opción que no sea entre las presentadas, se volverá a pedir que introduzca la opción.



 int Guardar_partida (char tablero[TAMANHO][TAMANHO],struct Estado E) ➞

Es una función de tipo entero (devuelve un valor entero) y se pasa como argumento la variable **char tablero[TAMANHO][TAMANHO]** la cual es pasada por referencia y no por valor al ser un array (esto significa que los cambios que se realicen dentro de la función hará que se modifique el contenido del array). En esta función el array tablero no se modifica. También se pasa la estructura Estado E. El valor de retorno de la función es 0, si no ha habido ningún problema al guardar la partida.

Nos permite guardar la partida en un archivo de texto (.txt) con la posición de las fichas y el turno de jugador (un 0 si le toca al jugador 1 y 1 si le toca al jugador 2.

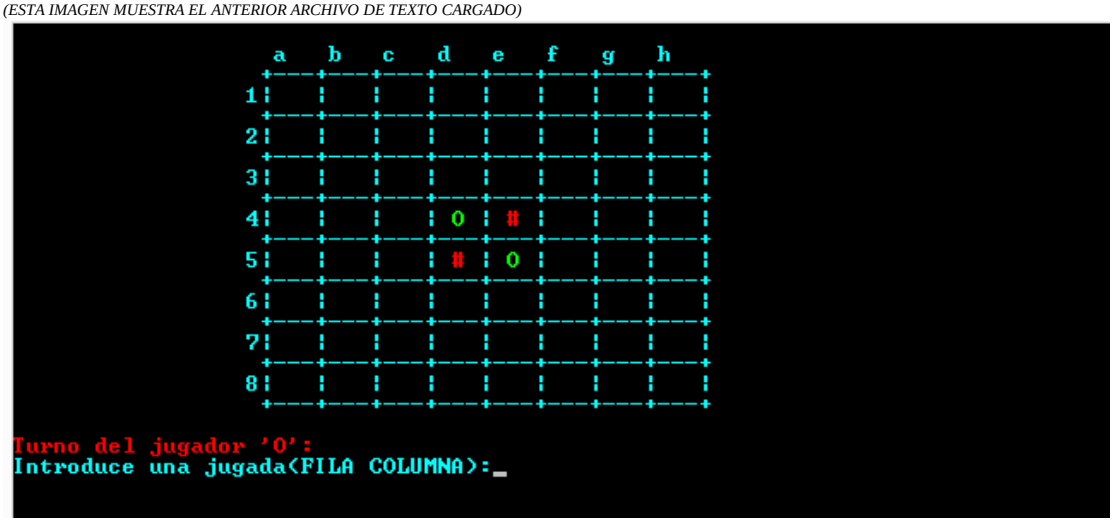


 void Cargar_partida(char tablero[TAMANHO][TAMANHO],struct Estado *E) ➞

Es una función de tipo void (no devuelve ningún valor) y se pasan como argumentos la variable **char tablero[TAMANHO][TAMANHO]** la cual es pasada por referencia y no por valor al ser un array (esto significa que los cambios que se realicen dentro de la función hará que se modifique el contenido del array). También se pasa el puntero a la estructura Estado E, lo que permitirá modificar las variables de la estructura y así no perder sus valores modificados al salir de la función.

Esta función carga una partida guardada en un archivo .txt, si existe. Coloca las fichas de la anterior partida en la variable **char tablero[TAMANHO][TAMANHO]** y, en la estructura Estado E,se asigna el turno del jugador a través del puntero a la variable **Turno_jugador** y le modifica el número de fichas que hay a través del puntero a la variable **Num_fichas**.

(ESTA IMAGEN MUESTRA EL ANTERIOR ARCHIVO DE TEXTO CARGADO)



 int Validar_movimientos(char tablero[TAMANHO][TAMANHO], int Movimientos_validos[TAMANHO][TAMANHO],struct Estado E) ➞ Es una función de tipo entero (devuelve un valor entero) y se pasa como argumento la estructura **Estado E**, la variable **char tablero[TAMANHO][TAMANHO]** la cual es pasada por referencia y no por valor al ser un array (esto significa que los cambios que se realicen dentro de la función hará que se modifique el contenido del array). Además se la variable **Movimientos_validos**, pasado por referencia al ser un array, el cual se modificará y estas modificaciones se mantendrán al acabar la función. El valor de retorno de la función es el número de movimientos válidos que hay en el tablero.

Esta función comprueba que los movimientos realizados por los jugadores sean movimientos correctos. Comprobamos las casillas en blanco, si se encuentra una ficha contraria nos moveremos en la dirección en la cual la ha encontrado hasta que encuentra una casilla del jugador. Entonces en la casilla en blanco inicial ,si es un movimiento válido, se coloca un 1 (en el array de **Movimientos_validos**) y devuelve el número jugadas posibles.

 void Hacer_movimientos (char tablero[TAMANHO][TAMANHO],struct Jugada J, struct Estado E) ➞Es una función de tipo void (no devuelve ningún valor) y se pasa como argumentos las estructura Jugada J y la estructura Estado E. Además la variable **char tablero[TAMANHO][TAMANHO]** la cual es pasada por referencia y no por valor al ser un array (esto significa que los cambios que se realicen dentro de la función hará que se modifique el contenido del array).

Coloca la ficha en la casilla elegida y recorre las casillas adyacentes para encontrar las fichas del oponente y poder moverse en esa dirección. Luego,si no encuentra una ficha del jugador no hace nada(es decir, si encuentra una casilla en blanco o se sale del tablero), y si la encuentra hace cambiar las fichas contrarias hasta volver a la casilla inicial.

 void puntuacion (char tablero[TAMANHO][TAMANHO]) ➞Es una función de tipo void(no devuelve ningún valor) y se la pasa como argumento la variable **char tablero[TAMANHO][TAMANHO]** la cual es pasada por referencia y no por valor al ser un array (esto significa que los cambios que se realicen dentro de la función hará que se modifique el contenido del array).
La función recorre el tablero,cuenta las fichas de cada jugador y, con dos contadores indica el número final de fichas de cada jugador. La función muestra también el tiempo que dura abierto el ejecutable.

 void Turno_jugador(char tablero[TAMANHO][TAMANHO],struct Estado *E) ➞
Es una función de tipo void (no devuelve ningún valor) y se le pasa como argumentos la variable **char tablero [TAMANHO][TAMANHO]** la cual es pasada por referencia y no por valor al ser un array (esto significa que los cambios que se realicen dentro de la función hará que se modifique el contenido del array). También se pasa el puntero a la **estructura Estado E**, lo que permitirá modificar las variables de la estructura y así no perder sus valores modificados al salir de la función.

Esta función se encarga de realizar todo lo referente al turno del jugador, es decir, llama a diferentes funciones para comprobar si hay movimientos válidos, para realizar la jugada y para que se abra el menú del juego. A parte, se actualizan el valor de las variables agrupadas en las **estructura Estado E**.

Primero se comprueba si el valor que devuelve la función que valida los movimientos es diferente de 0, es decir, hay movimientos posibles. Si es igual a 0, el contador de los movimientos inválidos aumenta y se pasa de turno si todavía no llega a 2. Si llega a 2 significa que ninguno de los jugadores puede mover y se termina la partida.
Si hay una jugada válida, se pide al jugador que introduzca las coordenadas de su movimiento y se comprueba si es válido. Si lo es, se realiza el movimiento y se pasa de turno, se reinicia el contador de movimientos inválidos y aumenta en uno el número de fichas. Por último se llama la función del menú del juego

 void Partida (char tablero[TAMANHO][TAMANHO],struct Estado E) ➞
Es una función de tipo void (no devuelve ningún valor) y se le pasa como argumentos la variable **char tablero[TAMANHO][TAMANHO]** la cual es pasada por referencia y no por valor al ser un array (esto significa que los cambios que se realicen dentro de la función hará que se modifique el contenido del array). Además se le pasa la estructura Estado E.

En esta función se realiza el cambio de turno mientras no se llena el tablero de fichas o el número de movimientos inválidos es menor que dos. Si el contador del turno del jugador(**Cont_jugador**) == 0, le toca al primer jugador y se le asigna a la variable char jugador la ficha 'O' y se muestra el tablero después del movimiento del otro jugador.
En cambio si **Cont_jugador == 1**, le toca al segundo jugador y char jugador pasa a ser '#' y se vuelve a mostrar el tablero después del movimiento del otro jugador.

 void Inicializar_tablero(char tablero[TAMANHO][TAMANHO],struct Estado *E) ➞Es una función de tipo void (no devuelve ningún valor) y se le pasa como argumentos la variable **char tablero[TAMANHO][TAMANHO]** la cual es pasada por referencia y no por valor al ser un array (esto significa que los cambios que se realicen dentro de la función hará que se modifique el contenido del array). También se pasa el puntero a la **estructura Estado E**, lo que permitirá modificar las variables de la estructura y así no perder sus valores modificados al salir de la función.

Vacía el tablero para empezar la partida e inicializa el número de fichas a 4 (modificará la variable **Num_fichas** de la **estructura Estado E**).

 void Tiempo_de_ejecucion(): Define dos variables de tipo clock_t, una para el momento en el que el reloj se inicia y otra para cuando se termina el programa (gracias a la librería time.h) que después emplea en la función puntuación junto con la fórmula *[(fin-inicio)/(double)CLOCKS_PER_SEC]*.

3.3 Estructuras

Hay dos estructuras, una del estado del juego (**struct Estado E**) y otra sobre la jugada introducida por los jugadores (**struct Jugada J**). La primera estructura, agrupa las variables que llevan la cuenta de la situación de la partida y se van modificando en las diferentes funciones. Las variables que contiene son: el número de fichas que hay en el tablero (**int Num_fichas**), el turno del jugador al que le toca(**int Cont_jugador**), el número de movimientos inválidos (**int Mov_invalidos**), la opción escogida en el menú del juego(**int Opcion_juego**) y la variable que contiene la ficha del jugador, que puede ser O o # (char jugador).
La segunda estructura, agrupa las variables que se refieren a la jugada introducida por el jugador, es decir, la fila (**int Fila_jugada**) y la columna (**char Columna_jugada**).

3.4 Procesos de acceso a array

Se utilizan tres arrays en total dentro del juego que son:

-char tablero[TAMANHO][TAMANHO]
-int Movimientos_validos[TAMANHO][TAMANHO]
-char tablero_cargado [TAMANHO*TAMANHO+1]

El array de tablero utilizamos para guardar las posiciones de las fichas de los jugadores y poder pasar la situación del tablero a las funciones que necesitan saber donde están colocadas las fichas. Este array se modifica en la función **void Hacer_movimientos (char tablero[TAMANHO][TAMANHO],struct Jugada J,struct Estado E)** en la cual se colocan las fichas de los jugadores en las coordenadas introducidas por estos mismos y se cambian las fichas que han sido flanqueadas. Por otro lado, donde se imprime el array el tablero, es decir las fichas de los jugadores, es en la función **void Visualizar_tablero(char tablero[TAMANHO][TAMANHO])**, la cual a parte de dibujar el tablero, tambien se imprimen las fichas de los jugadores.

El array de **Movimientos_validos** lo utilizamos para marcar las casillas donde los jugadores pueden mover con un 1, lo que permite comparar la jugada introducida con las coordenadas de este array y ver si hay un 1, lo que significaría que la jugada es válida. Este array se modifica en la función ***int Validar_movimientos (char tablero[TAMANHO][TAMANHO],int Movimientos_validos[TAMANHO][TAMANHO],struct Estado E).***
No se imprime por pantalla.
El array de tablero_cargado se encarga de guardar la cadena de texto que hay en el archivo partida.txt. Se utiliza en la función **void Cargar_partida(char tablero[TAMANHO][TAMANHO],struct Estado *E)** y no se imprime por pantalla.

3.5 Ficheros

Se utilizan los ficheros para realizar dos acciones complementarias, la opción de guardar la partida y la opción de cargarla luego. Para esto cada una de las opciones está encuadrada en dos funciones diferentes:

- **int Guardar_partida (char tablero[TAMANHO][TAMANHO],struct Estado E)**
- **void Cargar_partida(char tablero[TAMANHO][TAMANHO],struct Estado *E)**

Para guardar la partida(acción realizada en la función **Guardar_partida**, se crea el apuntador de tipo FILE*,el cual se usará como apuntador a la información del fichero. Luego, en el lugar donde esté el archivo del juego, se abrirá el fichero partida.txt o se crea, si no existe, en modo "w" (escritura). Luego se comprueba si se ha abierto correctamente, es decir, si no devuelve NULL (Un 0).
Para guardar la posición del tablero en el fichero se usará un dos bucles for anidados para recorrer el array **tablero[TAMANHO][TAMANHO]** y se irá guardando(fputc) lo que hay en cada casilla del tablero en el fichero.
Despues de guardar el tablero, se guardará el turno del jugador al que le toque un 0 si le toca jugar al jugador 1 o un 1 si le toca al jugador 2.
Por último se cierra el archivo y se comprueba si se cierra correctamente, es decir no devuelve NULL.

Para cargar la partida(acción realizada en la función **Cargar_partida**), se crea el apuntador de tipo FILE*, el cual se usará como apuntador a la información del fichero. Luego, en el lugar donde esté el archivo, se abrirá el fichero partida.txt, si existe, en modo "r"(lectura). Luego se comprueba si se ha abierto correctamente, es decir, si no devuelve 0(NULL). Si no se abre significa que no existe y se vuelve al main, ya que la opción de cargar está al principio de la partida, y en la otra opción de cargar la partida (en el menú del juego) ya comprobamos antes de entrar en la función cargar de si existe el fichero, ya que sino existiera se volvería al main pero mientras dura la partida se perdería la posición en la que el usuario le da la opción de cargar una partida anterior.
Si se abre el fichero, se obtendrá lo que hay en el fichero con fgets y se guardará el array **tablero_cargado** con la longitud de de TAMANHO*TAMANHO +1. Luego se colocará lo recogido en **tablero_cargado** en el array **tablero[TAMANHO][TAMANHO]** con dos bucles for. Cuando se terminen de ejecutarse los bucles for, es decir se han colocado todas las fichas en el tablero, se mirará el último carácter de la cadena contenida en **tablero_cargado** el cual indica el turno del jugador y se modificará el contador del turno del jugador.
Por último se contarán las fichas que hay en en el tablero para actualizar el contador **Num_fichas** y se cierra el fichero, comprobando que lo hace correctamente

3.6 Validación de errores

La validación de errores afecta principalmente al apartado de los ficheros, es decir si hay problemas al abrir o cerrar los archivos, lo cual está explicado en el apartado anterior y a la comprobación de las entradas del usuario.
El usuario a lo largo del juego tendrá que ir introduciendo las opciones que desea escoger y las coordenadas de la jugada, obviamente cabe la posibilidad que introduzca valores fuera de los esperados para ello se han utilizado estructuras do-while para que vuelva se vuelva a pedir que introduzca una opción correcta y un la estructura condicional if para que si es incorrecto lo introducido, se imprima por pantalla un mensaje avisando al usuario de que lo que ha introducido no es válido y debe volver a introducirlo.

3.6 Referencias y glosario de términos:

- **setlocale(LC_CTYPE, "Spanish")** ⟶ Función que nos permite poner acentos y caracteres especiales.

- **fflush(stdin)** ⟶ Funcion para limpiar el buffer de entrada de teclado. Sirve para evitar que guarde un enter en el buffer y lo malinterprete en siguientes bucles.

- **#define TAMANHO 8** ⟶ Definimos una variable global, la cual establecerá el tamaño del tablero.

- **http://www.tutorialspoint.com/cprogramming/index.htm**