Integrantes: - Matías Franco

-Julio Fernández

**Funciones y Estructuras de datos utilizados**

Variable para determinar el turno.

turno = 1 le pertenece a la computadora siempre (no significa que la computadora sea la primera a jugar).

turno = 2 le pertenece al usuario (el 2 no quiere decir que el usuario no empiece, dependerá de la función seleccionar\_jugador1).

int turno;

Array que contiene los identificadores del equipo al que el jugador pertenece

Este array puede contener solo uno de dos numero posibles: 1 ó 3.

1 representa a los policías y 3 a las suffragettes.

bandos[0] pertenece a la computadora

bandos[1] pertenece al usuario

La variable turno se utiliza como subíndice del array bandos en la forma bandos[turno-1].

Por ejemplo:

si turno = 1 ==> bandos[1-1] = bandos[0]. Luego; juega la computadora...

int bandos[2];

Variable utilizada como bandera/contador para averiguar si existe alguna otra ficha alrededor

de una ficha que ha realizado un salto. Esto, junto con la variable cont se usa para determinar

si es posible ejecutar un segundo o tercer salto.

int B;

Contador/bandera, utilizado junto con la variable B para determinar si una ficha que realizo un

salto puede volver a saltar.

Se permiten hasta tres saltos, por lo tanto, cont comienza de cero cuando se realiza una jugada

y se incrementa si existe la posibilidad de volver a salta. Esta se incrementa hasta tres.

int cont;

Array auxiliar creado para guardar las coordenadas finales de una ficha que acaba de realizar

un salto. Esto es para, en el caso de que se pueda volver a realizar un salto, utilizar dichas

coordenadas como las coordenadas iniciales x,y utilizadas en la función Coordenadas\_destino

int coor[2];

void ventana\_fin\_del\_juego();

Función encargada de administrar los movimientos de la computadora

void Juega\_Computadora();

void rev\_carcel\_hosp(int \*num3, int \*num4);

Función que recibe dos contadores. Uno para la cárcel, otro para el hospital.

Esta funcion se utiliza para saber si el número de fichas habilitadas de ambos bandos es menor o igual a 5 puesto que, si algún equipo tiene solo 5 fichas, no podrá conquistar las seis casillas de la base enemiga.

Se alguno de estos dos contadores resulta ser mayor o igual a 16, algún equipo no podrá ganar. Luego, su oponente es el ganador.

int Limitacion\_Bases(int x, int y, int pieza);

Limitación para evitar que los bandos entren en sus propias bases.

int limitacion\_Suffrag(int x, int y);

Limitación para evitar que los bandos entren en sus propias bases.

int limitacion\_Polic(int x, int y);

Limitación para evitar que los policías entren en su propia base.

int limite\_movimiento(int x, int y);

Combinación con limite\_tablero para impedir que las piezas entren en la cárcel u hospital

int verificar\_adyacente(int x, int y, int x2, int y2);

Verifica que las coordenadas de movimiento sean las de una casilla adyacentes a la casilla seleccionada.

void rotar\_jugador();

Rotar jugadores.

void mostrar\_turno();

Mostrar turnos. Imprime un mensaje sobre a quién le toca jugar.

char \*ordenar(char \*v);

Función para mostrar primero en el archivo los nombres de los usuarios, con mayor cantidad de veces ganadas.

void Estadisticas(int ganador);

Función que maneja un archivo, cuyo contenido es la estadística de juego del jugador.

void fin\_del\_juego(int num1, int num2, int num3, int num4);

Imprime el Mensaje de fin del juego de acuerdo a las condiciones de fin del juego.

int limite\_tablero(int x, int y);

Maximo y Minimo valores posibles que pueden tomar las coordenadas x e y.

void verificar\_conquista(int \*num1, int \*num2);

Para verificar si alguna de las bases ha sido conquistada.

Se utilizan num1 y num2(o mejor dicho sus direcciones), ambos se van incrementando.

Si alguno de estos dos llega a valer seis, quiere decir que una de las bases enemigas ha sido completamente invadida.

int verificar\_vacia(int x, int y);

Verifica si la casilla seleccionada esta vacía o no.

int revisar\_vecinos(int x, int y);

Revisa los alrededores de una ficha en busca de otras que estén en su adyacencia Auxiliar para saltos en cadena.

int verificar\_bando(int x, int y, int bando);

Verifica si la pieza que se encuentra en la casilla seleccionada pertenece al bando propio o al bando enemigo.

int verificar\_arena(int x, int y);

Para verificar que la ficha enemiga este dentro de la arena de combate de así serlo, podrá ser incapacitada por sus oponentes a la hora de ser saltada.

void Pieza\_Derrotada(int derrotado);

Función encargada de cargar en la cárcel/hospital a las piezas derrotadas.

int aux\_salto(int x, int y, int x2, int y2);

Auxiliar para la función Mov\_Salto. Recoge las direcciones del posible salto.

int Mov\_Salto(int x, int y, int x2, int y2, int pieza);

Recoge y verifica las coordenadas de los posibles saltos.

int Movimiento(int x, int y, int x2, int y2, int pieza);

Desplazamiento de piezas.

Si la casilla de destino esta vacía, se realiza un movimiento simple.

void Click\_Dos(int pieza, int x, int y, int x2, int y2);

Función que se encarga de las coordenadas de destino, después de ejecutarse la función del primer click.

int Primer\_Click(int bando, int x, int y);

Inicio. Se selecciona la pieza a mover y se guardan sus coordenadas.

void Click\_saltos(int pieza, int x, int y, int x2, int y2);

Función que se encargar de administrar las posibilidades del usuario para salta sobre la ficha.

**Descripción y justificación de la inteligencia/estrategia utilizada**

Se ha utilizado las propiedades matemáticas que nos brinda los vectores, como la distancia que hay entre dos puntos en un plano.

La computadora, toma una ficha de su bando y busca cual es la casilla de la base enemiga más cercana, selecciona hasta tres posibles movimientos para acercase los más posible a las casillas y entrar en ellas.

Mientras se repite este proceso, también la computadora va saltando o haciendo movimientos simples, siempre que esté dentro de sus posibilidades.

**Puntos fuertes y débiles del programa**

Algunos puntos débiles del programa son:

-La comunicación en redes.

-Las estadísticas de por si se realizan en un archivo correctamente, pero no son fáciles de implementar en la interfaz gráfica.

**Dificultades encontradas durante el trabajo**

A veces se presentaban problemas a la hora de implementar el código .c en la interfaz, ya que se debía tener un conocimiento buenos de las funciones que brinda gtk.

El manejo de archivos también tuvo su dificultad, ya que se tuvo que investigar funciones como strtok para lograr una buena optimización del código.

**Guía de utilización del programa**

En el menú principal tenemos:

-Jugar:

Si se quiere comenzar una nueva partida. Al comenzar se tiene los modos de juego a querer jugar, selecciona un modo y tiene luego la posibilidad de cargar el nombre de cada jugador. Luego se elige el bando para cada jugador y el turno que tendrá cada uno. Cabe recalcar que, al seleccionar una opción en las ventanas, se debe confirmar la elección haciendo click en el botón siguiente.

-Reglas del Juego:

Se especifica las reglas del juego. Si uno se quiere informar sobre el juego puede hacer click en ese botón.

-Estadísticas:

Se muestra un historial de la cantidad de jugadores que han jugado el juego, y como han terminado (perdiendo o ganando).

-Créditos:

Si uno quiere saber, quiénes fueron los creadores del juego a nivel de código.

-Salir:

Se sale del juego.

-Jugando:

Se cuenta con una barra de meno en la parte superior de la ventana, la cual cuenta con las opciones de ayuda (contiene las reglas del juego y los créditos) y juego (contiene la opción de nuevo juego y salir de este).

En la esquina derecha, se muestran mensajes, del estado actual del juego (turnos, bandos, etc.).

Para hacer un salto sobre una ficha se debe de hacer click en nuestra ficha y después click en la ficha a saltar o comer.

Al jugar el modo pc vs usuario, con la configuración pc juega primero, se debe de hacer click en el botón cerrar de la ventana saliente por defecto, la cual contiene las reglas del juego.