Universidad Rafael Landívar
Facultad de Ingeniería.
Ingeniería Industrial.
Laboratorio de pensamiento computacional - Sección: 03
Catedrático: Ing. Cristian Estuardo Roldán Rodríguez.

PROYECTO DE LABORATORIO NO. 02

Estudiante: Chacón Flores Sara Fernanda Elizabeth
Carné: 1180524

Guatemala, 20 de mayo de 2024.

ACCIONES DEL PROGRAMA

1. **Agregar N cantidad de piezas en el tablero:**
   1.1 Para cada pieza, el usuario debe especificar y confirmar el tipo de pieza, por ejemplo: alfil, peón, caballo, etc., su color (blanco o negro), y la posición exacta usando notación estándar ('a1', 'e4', etc.).
   1.2 Antes de colocar la pieza, el programa verifica que la posición seleccionada esté libre y dentro del parámetro del tablero, garantizando que no se sobrepongan piezas en una misma casilla.
2. **Ingresar la pieza del caballo a evaluar:**
   2.1 Se solicita al usuario que especifique la posición y color de la pieza del caballo, que será analizada para determinar sus movimientos posibles.
   2.2 El programa asegura que la casilla destinada para el caballo esté vacía y dentro del parámetro antes de ubicarlo en el tablero.
3. **Determinar los posibles movimientos validos:**
   3.1 Calcula los movimientos factibles para la pieza del caballo desde la posición determinada por el usuario, siguiendo los movimientos típicos en forma de 'L' de esta pieza.
   3.2 Cada posible movimiento es verificado para confirmar que la casilla está dentro del tablero y no está ocupada por una pieza del mismo color.
   3.3 Se presenta una lista de los movimientos válidos, señalando si la casilla destino está vacía o no.
4. **Mostrar matriz:**
   4.1 Mostrar el estado actual del tablero en la consola, utilizando una representación visual clara y organizada con cada pieza ingresada por el usuario.

DATOS DE ENTRADA INGRESADOS POR EL USUARIO

1. **Cantidad de piezas a agregar:**

   - Tipo de dato: Entero
   - Mensaje: "Ingrese la cantidad de piezas a agregar: "
   - Validación: El valor debe ser un número entero positivo.

2. **Tipo de pieza:**

   - Tipo de dato: Cadena de texto
   - Mensaje: "Ingrese el tipo de pieza (alfil, peon, caballo, torre, rey, reina):”
   - Validación: El valor debe ser uno de los tipos de pieza válidos ('alfil', 'peon', 'caballo', 'torre', 'rey', 'reina').

3. **Color de la pieza:**

   - Tipo de dato: Cadena de texto
   - Mensaje: "Ingrese el color de la pieza (blanco/negro): "
   - Validación: El valor debe ser 'blanco' o 'negro'.

4. **Posición de la pieza:**

    - Tipo de dato: Cadena de texto
    - Mensaje: "Ingrese la posición de la pieza (ej., a1): "
    - Validación: La posición debe ser válida según la notación ajedrecística ('a1' a 'h8') y debe estar libre en el tablero.

5. **Posición del caballo a evaluar:**

    - Tipo de dato: Cadena de texto
    - Mensaje: "Ingrese la posición del caballo a evaluar (ej., e4): "
    - Validación: La posición debe ser válida según la notación ajedrecística ('a1' a 'h8') y no debe estar ocupada por otra pieza.

6. **Color del caballo a evaluar:**

    - Tipo de dato: Cadena de texto
    - Mensaje: "Ingrese el color del caballo (blanco/negro): "
    - Validación: El valor debe ser 'blanco' o 'negro'.

VARIABLES UTILIZADAS

1. **Tablero de ajedrez:**

    - Nombre: tablero
    - Tipo de dato: list[list[str]] (Matriz)
    - Matriz de 8x8 que representa el tablero de ajedrez. Cada celda contiene una cadena vacía (posición vacía) o una cadena que indica el tipo y color de la pieza

2. **Cantidad de piezas:**

    - Nombre: piezas
    - Tipo de dato: int (Entero)
    - Número entero que representa la cantidad de piezas que el usuario desea agregar al tablero.

3. **Tipo de pieza:**

    - Nombre: pieza
    - Tipo de dato: str (Cadena de texto)
    - Cadena que representa el tipo de pieza a agregar ('alfil', 'peon', 'caballo', 'torre', 'rey', 'reina').

4. **Color de la pieza:**

    - Nombre: color
    - Tipo de dato: str (Cadena de texto)
    - Cadena que representa el color de la pieza ('blanco' o 'negro').

5. **Posición de la pieza:**

    - Nombre: posición
    - Tipo de dato: str (Cadena de texto)
    - Cadena en notación ajedrecística que representa la posición en el tablero donde se agregará la pieza.

6. **Posición del caballo a evaluar:**

    - Nombre: pos_caballo
    - Tipo de dato: str (Cadena de texto)
    - Descripción: Cadena en notación ajedrecística que representa la posición inicial del caballo cuyo movimiento se evaluará (e.g., 'e4').

7. **Color de la pieza de caballo a evaluar:**

    - Nombre: color_caballo
    - Tipo de dato: str (cadena de texto)
    - Cadena que representa el color del caballo ('blanco' o 'negro').

8. **Movimientos válidos del caballo:**

    - Nombre: movimientos_caballo
    - Tipo de dato: list[tuple[int, int]] (Lista de listas asociativas)
    - Lista de tuplas que representan las posiciones en el tablero (índices de la matriz) a las que el caballo puede moverse.

CONDICIONES O RESTRICCIONES

Validación de la cantidad de piezas: La cantidad de piezas debe ser un número entero positivo.

- Condición: piezas.isdigit() and int(piezas) > 0

Tipos de pieza válidos: Solo se aceptan tipos de piezas estándar del ajedrez: 'alfil', 'peon', 'caballo', 'torre', 'rey', 'reina'.

- Condición: pieza in ['alfil', 'peon', 'caballo', 'torre', 'rey', 'reina']

Colores válidos: Solo se aceptan los colores 'blanco' y 'negro'.

- Condición: color in ['blanco', 'negro']

Validación de posiciones: La posición debe tener un formato de dos caracteres: una letra entre 'a' y 'h' seguida de un número entre '1' y '8'. La posición debe estar dentro del rango del tablero.

- Condición: posicion[0] in 'abcdefgh' and posicion[1] in '12345678'

Verificación de celdas ocupadas: No se debe agregar una pieza en una celda ya ocupada.

- Condición: tablero[fila][col] == ''

Restricciones de movimiento del caballo: El caballo puede moverse en forma de "L": dos casillas en una dirección y una en dirección perpendicular. Debe evitar movimientos fuera del tablero. No puede moverse a una celda ocupada por una pieza del mismo color.

Cálculos:

Conversión de posiciones a índices de matriz: Convierte las posiciones en notación ajedrecística a índices de matriz para facilitar el acceso.

- Cálculo: col = ord(posicion[0]) - ord('a'), fila = 8 - int(posicion[1])

Generación de movimientos del caballo: Generar todas las posibles posiciones a las que puede moverse un caballo desde una posición dada. Verificar que las nuevas posiciones estén dentro de los límites del tablero y no estén ocupadas por piezas del mismo color. Basado en la posición actual del caballo, el programa calcula dónde podría moverse, siguiendo un patrón en forma de 'L' (moviéndose dos casillas en una dirección y una en perpendicular). Esto implica ajustar las coordenadas actuales del caballo en diferentes combinaciones.

FUNCIONES IMPLEMENTADAS

1. **crear_tablero** Crea una matriz 8x8 que representa el tablero de ajedrez vacío.

- Entrada: Ninguna.
- Salida: list[list[str]] - Tablero vacío.

2. **posicion_a_indice** Convierte una posición en notación ajedrecística (e.g., 'e4') a índices de matriz.

- Entrada: str - Posición en notación ajedrecística.
- Salida: tuple[int, int] - Índices de la matriz correspondientes a la posición.

3. **posicion_valida** Verifica si una posición en notación ajedrecística es válida (e.g., 'e4').

- Entrada: str - Posición en notación ajedrecística.
- Salida: bool - True si la posición es válida, False en caso contrario.

4. **agregar_pieza** Agrega una pieza al tablero si la posición es válida y está libre.

- Entrada:

list[list[str]] - Tablero.

str - Tipo de pieza.

str - Color de la pieza.

str - Posición en notación ajedrecística.

- Salida: bool - True si la pieza se agrega correctamente, False en caso contrario.

5. **obtener_movimientos_caballo** Calcula los movimientos válidos de un caballo desde una posición dada, evitando posiciones ocupadas por piezas del mismo color.

- Entrada:

list[list[str]] - Tablero.

str - Posición inicial del caballo en notación ajedrecística.

str - Color del caballo.

- Salida: list[tuple[int, int]] - Lista de posiciones válidas a las que el caballo puede moverse.

6. **imprimir_tablero** Imprime el tablero de ajedrez con un formato legible.

- Entrada: list[list[str]] - Tablero.
- Salida: Ninguna.

7. **obtener_input_valido** Solicita entrada del usuario y la valida hasta que sea correcta.

- Entrada:

Str - Mensaje a mostrar al usuario.

callable - Función de validación.

- Salida: str - Entrada válida del usuario.

8. **validar_entero_positivo** Verifica si una entrada es un número entero positivo.

- Entrada: str - Entrada del usuario.
- Salida: bool - True si la entrada es un número entero positivo, False en caso contrario.

9. **validar_tipo_pieza** Verifica si el tipo de pieza es válido.

- Entrada: str - Tipo de pieza.
- Salida: bool - True si el tipo de pieza es válido, False en caso contrario.

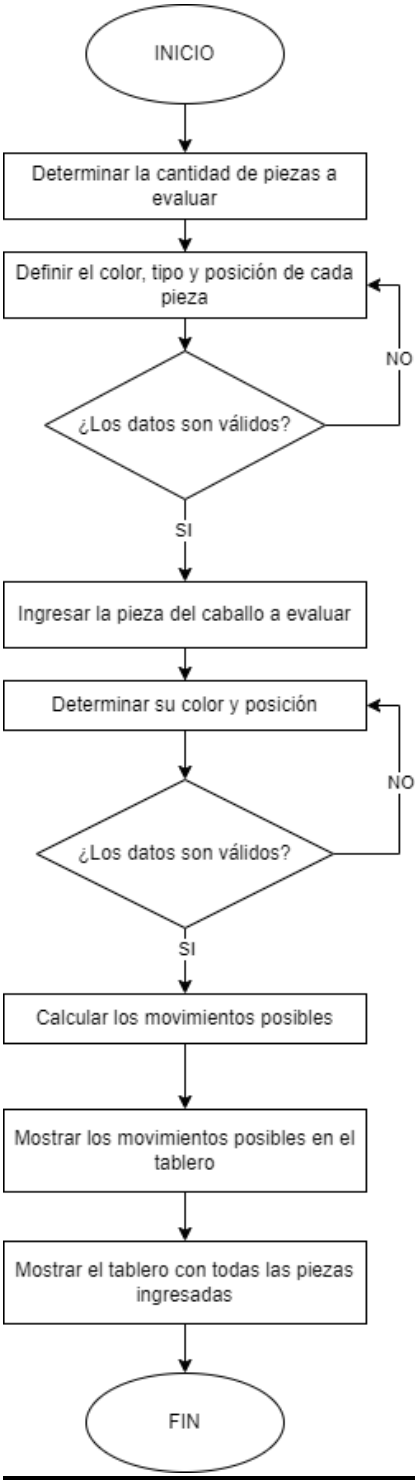
10. **validar_color** Verifica si el color de la pieza es válido.

- Entrada: str - Color de la pieza.
- Salida: bool - True si el color es válido, False en caso contrario.

11. **principal** Función principal que coordina la ejecución del programa.

- Entrada: Ninguna.
- Salida: Ninguna

DIAGRAMA DE FLUJO



Fuente: Elaboración propia en draw.io - Free Flowchart Maker and Diagrams Online, n.d.