INFORME TÉCNICO: PROYECTO AJEDREZ C++

1. INTRODUCCIÓN

Este documento describe la implementación de un juego de Ajedrez en C++ para consola. El proyecto utiliza Programación Orientada a Objetos (POO) para modelar las piezas, el tablero y la lógica del juego, permitiendo una partida entre dos jugadores (Humano vs Humano).

2. ESTRUCTURA DEL PROYECTO

El proyecto está organizado en las siguientes carpetas:
- src/: Contiene los archivos de implementación (.cpp).
- include/: Contiene los archivos de cabecera (.h).
- docs/: Documentación del proyecto.

Archivos principales:
- main.cpp: Punto de entrada. Muestra el menú y gestiona el ciclo de vida de la aplicación.
- Partida.cpp/h: Controla el flujo de la partida (turnos, validaciones globales, input de usuario).
- Tablero.cpp/h: Representa la cuadrícula de 8x8 y gestiona la posición de las piezas.
- Pieza.cpp/h: Clase base abstracta para todas las piezas.
- [Piezas Concretas]: Peon, Torre, Caballo, Alfil, Reina, Rey (cada una con su .h y .cpp).
- Extra.h: Definiciones auxiliares (Enums Color, TipoPieza y struct Posicion).

3. ARQUITECTURA DE CLASES

3.1. Clase Base: Pieza

Es una clase abstracta que define la interfaz común para todas las piezas.
- Atributos: color (BLANCO/NEGRO), tipo (PEON, TORRE...), pos (Posicion), seHaMovido (bool).
- Métodos Virtuales Puros:
  - movimientovalido(destino, tablero): Cada pieza implementa su propia lógica de movimiento.
  - getSimbolo(): Retorna la representación en texto (ej: "PB", "TN").

3.2. Clases Derivadas (Piezas)

- Peon: Implementa movimiento simple hacia adelante, captura diagonal, movimiento doble inicial.
- Torre: Movimiento lineal horizontal/vertical.
- Alfil: Movimiento diagonal.
- Caballo: Movimiento en "L" (salta piezas).
- Reina: Combina movimientos de Torre y Alfil.
- Rey: Movimiento de una casilla en cualquier dirección y lógica de Enroque.

3.3. Clase Tablero

Gestiona una matriz de punteros a Pieza (Pieza* casillas[8][8]).
- Responsabilidades: Inicializar el tablero, mover piezas (actualizar punteros), imprimir el estado actual, gestionar capturas (memory management).

3.4. Clase Partida

Controlador principal del juego.

- Responsabilidades:
  - Alternar turnos (Blancas/Negras).
  - Validar input del usuario (coordenadas tipo "E2").
  - Orquestar movimientos especiales:
    - Enroque: Verifica condiciones y mueve Rey y Torre.
    - Coronación: Detecta peón al final y solicita nueva pieza.

4. LÓGICA DEL JUEGO

4.1. Flujo Principal

1. Menú Principal (Jugar, Instrucciones, etc.).
2. Inicialización del Tablero (colocación de piezas).
3. Bucle de Juego:
   - Mostrar Tablero.
   - Solicitar movimiento al jugador actual.
   - Validar movimiento (sintaxis, reglas de pieza, reglas de tablero).
   - Ejecutar movimiento.
   - Cambiar turno.
   - (Repetir hasta condición de fin - actualmente bucle infinito o salida manual).

4.2. Validaciones

Cada movimiento pasa por varias capas de validación:

1. Validación de Input: Formato correcto (A-H, 1-8).
2. Validación de Origen: ¿Hay pieza? ¿Es de mi color?
3. Validación de Pieza: ¿La pieza puede moverse así geométricamente? (Lógica en cada clase derivada).
4. Validación de Camino: ¿Hay obstáculos? (Excepto Caballo).

5. COMPILACIÓN Y EJECUCIÓN

Requisitos: Compilador C++ (g++ o clang++).

Comando de compilación (desde la raíz del proyecto):

```
g++ src/main.cpp src/Partida.cpp src/Tablero.cpp src/Pieza.cpp src/Peon.cpp src/Torre.cpp src/Caballo.cpp src/Alfil.cpp src/Reina.cpp src/Rey.cpp -I include -o ajedrez.exe
```

Ejecución:

```
./ajedrez.exe
```