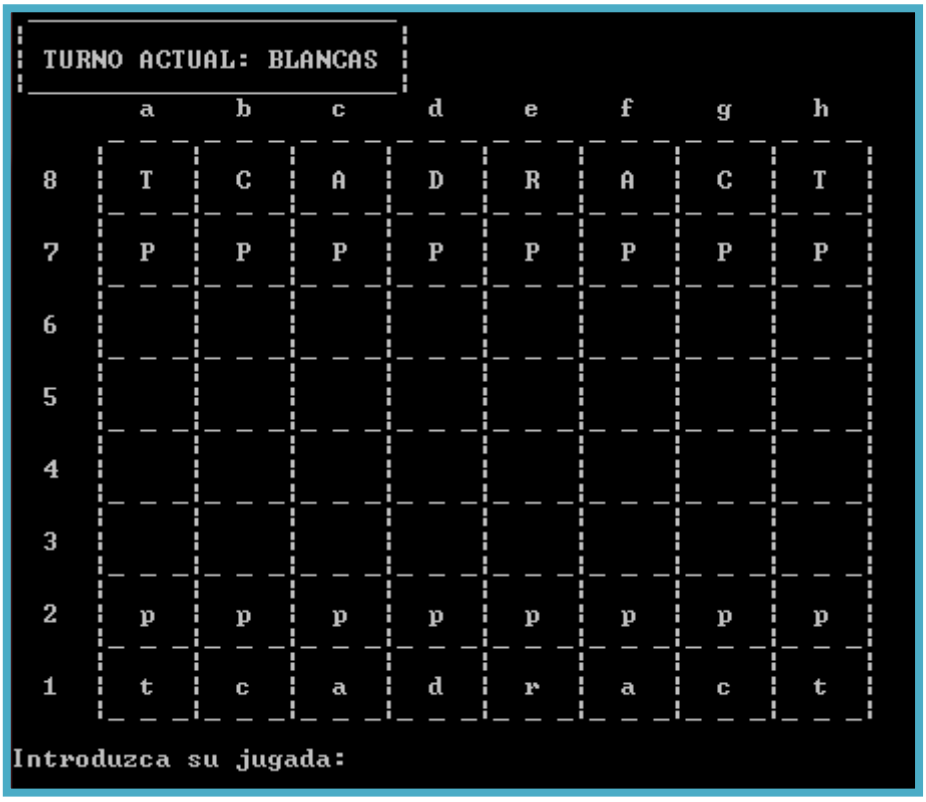
Programación de un ajedrez en C++

Programación de Sistemas - Curso 2015/2016

Javier Díaz Mena 12100
Eduardo Ballesteros Abellán 12027
Marta Ures Ciriaco 13442



INTRODUCCIÓN

Se trata de una modesta aplicación que permite jugar al ajedrez a dos personas. No se trata de un ajedrez contra el que se pueda jugar, debe haber dos oponentes humanos. A continuación se expone una pequeña guía de uso del programa, y más adelante una explicación completa de la lógica de los algoritmos.

GUÍA DE USO

El juego interacciona con los usuarios mediante la salida por pantalla de un marcador del turno, el tablero con las piezas y una línea donde teclear la jugada deseada.

Toda entrada y salida es en modo texto. El tablero muestra una matriz de 8x8 casillas, algunas de ellas con las letras asignadas a cada pieza. Las letras minúsculas corresponden al bando de las blancas, y las mayúsculas a las negras. El marcador del turno consiste en un recuadro en cuyo interior se indica de qué bando es el turno actual.

Es importante destacar que la entrada por teclado debe seguir la notación algebraica que dictan las reglas del ajedrez, que de forma general responde al patrón:

Letra_mayúscula_piezaColumna_destinoFila_destino

Las piezas se representan por las siguientes letras: R, rey; D, dama; A, alfil; C, caballo; T, torre. En el caso del peón, no se pone letra, solamente se indica la columna y fila destino, que van desde la ‘a’ hasta la ‘h’ y del 1 al 8 respectivamente.

Ejemplos: Cb8, e4, Dh7.

Dicha notación abarca todos los casos especiales del ajedrez como enroques, ambigüedades, capturas, jaques, etc. Más adelante se explica cuáles se han tenido en cuenta en nuestro programa.

ESTRUCTURAS DE DATOS Y CLASES UTILIZADAS. EXPLICACIÓN DE FUNCIONES

Procedemos a explicar las partes más significativas del código realizado para poder tener una conocimiento global de las ideas implementadas.

-La función moverse

Una de las funciones más importantes en nuestro código es la función moverse. Se trata de una función virtual de la clase pieza, ya que todas las piezas van a emplearla pero cada una lo hará de una manera determinada. Esta función recibe la columna y fila de destino introducidas por el usuario, de manera que al haber pasado los requisitos mínimos en el main para considerarse sintaxis correcta, comienza a comprobar si coincide con el tipo de movimiento de la pieza y si puede mover adecuadamente a la casilla destino. En caso afirmativo mueve la pieza y devuelve un 1 al main.

-La función capturar

Se basa en prácticamente el mismo fundamento que la función moverse, solo que en esta función será requisito fundamental que haya una pieza rival en la casilla de destino. Se perderá

la ubicación de la pieza que había en esa casilla y se moverá la pieza a esa casilla. Sabemos que no es adecuado retirar directamente el puntero, pero los objetos son alcanzables en todo momento por ser globales. Además, destruir una pieza en estas condiciones supondría errores en las llamadas a sus funciones, que no encontrarían el objeto.

-Tablero

El constructor de tablero declara una matriz de 8x8 casillas y posteriormente asocia a las casillas las direcciones de las piezas.

La salida por pantalla del tablero con las piezas situadas en él se realiza a través de una gran serie de cout que imprimen la matriz. De hecho imprimen la letra a la que apunta la casilla. Alrededor de las piezas hemos decidido imprimir los límites del tablero y las casillas usando los caracteres barra vertical y guión bajo . Todo ello se implementa en un procedimiento de la clase tablero denominado imprimetablero.

DIAGRAMA DE CLASES

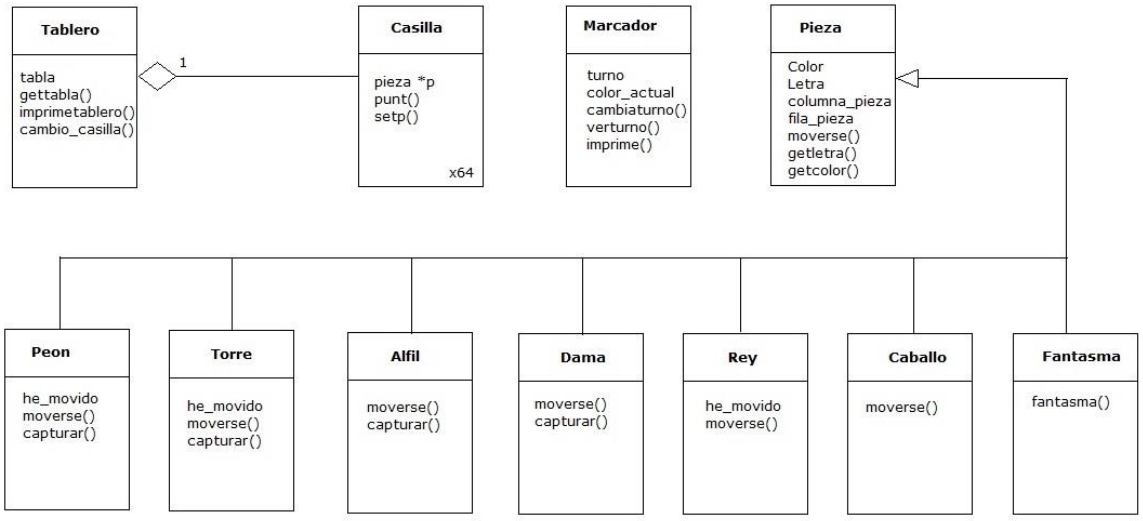
A continuación se muestra el diagrama de clases pensado para desarrollar el programa de manera más eficiente. Podemos ver que no es muy complejo en nuestro caso.

La clase pieza tiene 7 derivadas, que son los 6 tipos de pieza y el "fantasma", a quien apunta una casilla si está vacía. Se trata de una generalización puesto que todas son piezas.

Más abajo podemos ver métodos polimórficos como son moverse y capturar, que en pieza son virtuales y que en peon, rey, dama, etc se redefinen para su adecuado uso.

El tablero se compone de una matriz de 64 casillas, capaz de imprimirse junto con las piezas. Las casillas disponen de un puntero a la pieza que tengan encima, que puede ser "real" o el fantasma en su defecto.

La única clase que no tiene relación con las demás es el marcador, que a cada iteración del bucle principal, cambia el turno y lo muestra, sin mayor influencia en el resto del programa.



PRUEBAS DE VALIDACION

A continuación se exponen una serie de pruebas a las que se ha sometido el programa, y si las pasan con éxito. Hemos decidido incluir una tercera columna con los test que pasaba el programa anterior de ajedrez que hicimos, y a partir del cual está hecho el que se entrega. Tuvimos que reescribir el código y eliminar muchas tareas que se habían implementado con alto detalle por errores de ejecución que no pudimos solucionar. Tal y como se puede ver, el ajedrez entregado tiene varias funcionalidades menos que el originalmente ideado.

PRUEBA	Actual	Anterior
Representación del estado de la partida	Si	Si
Reconocimiento de la solicitud de movimientos	Si	Si
Comprueba legalidad de la jugada introducida	Si	Si
Movimiento de las piezas	Si	Si
Movimiento propio de cada pieza	Si	Si
Reloj de cada bando	No	No
Recuento del número de jugadas	No	No
Marcador piezas capturadas	No	Si
Marcador del turno	Si	Si
Situación de jaque	No	Si
Enroque	No	Si
Reconocimiento de ambigüedades	No	Si
Captura de peón al paso	No	No
Fin de la partida	No	Si

REPARTO DE ROLES

Marta Ures

- Salida por pantalla del ajedrez

- Clases casilla y tablero

- Realización del código de captura de todas las piezas

- Reescritura del programa a uno más sencillo

- Elaboración de comentarios explicativos en código

- Corrección de errores de sintaxis y compilación

Javier Díaz

- Diagrama de clases

- Realiza el makefile

- Marcadores

- Práctica y ejecución de las pruebas de validación

- Realización del código de movimiento de todas las piezas

- Corrección de errores de sintaxis y compilación

Eduardo Ballesteros

- Escritura del main

- Implementación de estados clasificatorios según entrada por teclado del usuario

- Ideado del marcador

- Orden inicial de llamadas a funciones

- Corrección de errores de sintaxis y compilación

- Explicación del código en la documentación