Documentation:

Classes:

Struct position: contains int x, and int y which represent the position on the board

Class enum colour: have black and white which represent the colour of the pieces

Class pieces: used as a piece_code to identify the type of the piece

Base class piece:

> Child classes: king, queen, bishop, knight, rook, pawn
>
> Using polymorphism to determine the available moves for each piece according to the rules of the game.

Class smartpiece: contains a pointer and handles deleting and copying the pointer to the piece class which contains child class.

Class board: contains a 2d array of smartpiece which represents the board and holds all the pieces according to their position. Handles the move option by using move operator= overloaded in smartpiece and changes the pos of the piece according to the destination moved to.

Class renderwindow: contains the renderer, window, height, width, edge (the size of the edge of the square calculated using the width and height since the window is resiable), title, running (bool to determine whether the window is closed or not). Renderwindow handles the graphic input and output of the program.

Class texture: contains an SDL_texture created from and image from the file “res”. Handles drawing squares and images onto the renderer of the renderwindow so the renderer can output them onto the window later using renderwindow class.

Nmespace main_game: contains the sdl loop of the main game which uses renderwindow to handle graphic output and input for the main game.

Namespace move_input: contains the sdl loop for the output of the legal moves and the input of the move.

Namespace promotion: contains the sdl loop for the output and input of the promotion.

Nammesppace menu: contains the sdl loop for the output and input of the menu. // not finished

Main:

Start by initializing renderwindow variable, an array of pointers of the textures which is filled using static functions in the texture class, and finally initialize the board.

Then using the sdl loop of the main game the game starts once the game until the sdl_quit is hit which set the running in renderwindow to false thus close the sdl loop.

Then the array of texture pointers manually allocated through texture class is deleted using static fundtion in the texture class.

Finally return