#pragma once

#include <iostream>

#include <conio.h>

#define Check\_the\_req\_cel(ver, hor) (ver % 2 != 0 && hor % 2 == 0) || (ver % 2 == 0 && hor % 2 != 0)//макрос вычёркивающий из использования половину поля, которая по правилам не должна быть задействована

#define check\_poz\_pavn\_white\_or\_black(filed, ver, hor, my\_pavn, my\_queen, evil\_pavn, evil\_queen, i, j) ((((filed[hor + i][ver + i] == my\_pavn || filed[hor + i][ver + i] == my\_queen) || ((filed[hor + i][ver + i] == evil\_pavn || filed[hor + i][ver + i] == evil\_queen) && filed[hor + j][ver + j] != ' ') || filed[hor + i][ver + i] == '0') &&((filed[hor + i][ver - i] == my\_pavn || filed[hor + i][ver - i] == my\_queen) || ((filed[hor + i][ver - i] == evil\_queen || filed[hor + i][ver - i] == evil\_queen) && filed[hor + j][ver - j] != ' ') || filed[hor + i][ver - i] == '0') &&(filed[hor - i][ver + i] == my\_pavn || filed[hor - i][ver + i] == my\_queen || filed[hor - i][ver + i] == ' ' || (filed[hor - i][ver + i] != ' ' && filed[hor - j][ver + j] != ' ') || filed[hor - i][ver + i] == '0') &&(filed[hor - i][ver - i] == my\_pavn || filed[hor - i][ver - i] == my\_queen || filed[hor - i][ver - i] == ' ' || (filed[hor - i][ver - i] != ' ' && filed[hor - j][ver - j] != ' ') || filed[hor - j][ver - j] == '0')))

#define check\_poz\_queen\_white\_or\_black(filed, ver, hor, my\_pavn, my\_queen, evil\_pavn, evil\_queen, i, j) (((filed[hor + i][ver + i] == my\_pavn || filed[hor + i][ver + i] == my\_queen) || ((filed[hor + i][ver + i] == evil\_pavn || filed[hor + i][ver + i] == evil\_queen) && filed[hor + j][ver + j] != ' ') || filed[hor + i][ver + i] == '0') &&((filed[hor + i][ver - i] == my\_pavn || filed[hor + i][ver - i] == my\_queen) || ((filed[hor + i][ver - i] == evil\_queen || filed[hor + i][ver - i] == evil\_queen) && filed[hor + j][ver - j] != ' ') || filed[hor + i][ver - i] == '0') &&(filed[hor - i][ver + i] == my\_pavn || filed[hor - i][ver + i] == my\_queen || (filed[hor - i][ver + i] != ' ' && filed[hor - j][ver + j] != ' ') || filed[hor - i][ver + i] == '0') &&(filed[hor - i][ver - i] == my\_pavn || filed[hor - i][ver - i] == my\_queen || (filed[hor - i][ver - i] != ' ' && filed[hor - j][ver - j] != ' ') || filed[hor - j][ver - j] == '0'))

#define pawn\_black\_and\_white\_beating(filed, ver, hor, pawn, queen) (((filed[hor - 1][ver + 1] == pawn || filed[hor - 1][ver + 1] == queen) && filed[hor - 2][ver + 2] == ' ' && hor - 2 >= 2 && ver + 2 <= 9) || ((filed[hor + 1][ver + 1] == pawn || filed[hor + 1][ver + 1] == queen) && filed[hor + 2][ver + 2] == ' ' && hor + 2 <= 9 && ver + 2 <= 9) || ((filed[hor + 1][ver - 1] == pawn || filed[hor + 1][ver - 1] == queen) && filed[hor + 2][ver - 2] == ' ' && hor + 2 <= 9 && ver - 2 >= 2) || ((filed[hor - 1][ver - 1] == pawn || filed[hor - 1][ver - 1] == queen) && filed[hor - 2][ver - 2] == ' ' && hor - 2 >= 2 && ver - 2 >= 2))

using namespace std;

void PlayWithPC(int Konami);

void PlayWithHuman(int Konami);

void Menu\_Playing(int Konami);

void Rules();

void Menu();

void Menus();

void KonamI(int& Konami);

void space();

void spaceFIELD();

void creat\_a\_field(int size, char white, char black, char queen\_white, char queen\_black, char filed[][12], int Konami);//Формирование поля и расстановка пешек

void all\_game(char filed[][12], char white, char black, int size, int queen\_white, int queen\_black, int color);//Процесс игры

void check\_poz\_pawn(char filed[][12], int turn\_of\_the\_more, int& poz\_hor, int& poz\_ver, char white, char black, char queen\_white, char queen\_black, int AIturn);//Проверка вводимых координат пешки

void filed\_out\_durig\_the\_game(char filed[][12], int size);//Вывод поля после хода игрока

void pawn\_pos\_inp\_error(int& poz\_ver, int& poz\_hor, int AIturn);//При вводе позиции пешки допущена ошибка

void the\_queens\_move(int& new\_poz\_ver, int& new\_poz\_hor, int& poz\_ver, int& poz\_hor, bool& exit\_the\_str\_cucle, char filed[][12], int AIturn);//ход королевы

void check\_the\_ability\_to\_beat\_again(char field[][12], int& new\_poz\_hor, int& new\_poz\_ver, int& poz\_hor, int& poz\_ver, int size, int turn\_of\_the\_more, int& col\_vo\_white, int& col\_vo\_black, int AIturn, char evil\_pawn, char evil\_queen);//может ли пешка бить еще раз

void check\_a\_new\_position(bool& exit\_the\_str\_cucle, int& new\_poz\_ver, int& new\_poz\_hor, int& poz\_ver, int& poz\_hor, char filed[][12], char black, char queen\_black, char white, char queen\_white, int AIturn);//проверка новой позиции

void can\_the\_queen\_beat\_more( int& new\_poz\_ver, int& new\_poz\_hor, int& poz\_ver, int& poz\_hor, bool& exit\_the\_str\_cucle, char field[][12], int& col\_vo\_white, int& col\_vo\_black, int turn\_of\_the\_more, int size, int AIturn);//может ли королева бить еще раз

void who\_needs\_to\_take\_away\_figure(int turn\_of\_the\_more, int& col\_vo\_black, int& col\_vo\_white);//кому отнаять количество фигур

void the\_queen\_repeat\_bets(char field[][12], int& new\_poz\_ver, int& new\_poz\_hor, int& poz\_ver, int& poz\_hor, char own\_pawn, char own\_queen, char alien\_pawn, char alien\_queen, int AIturn, int k, int j);//королева бьет еще раз

void queen\_beat(char field[][12], int& new\_poz\_hor, int& new\_poz\_ver, int& poz\_hor, int& poz\_ver, int turn\_of\_the\_more, int& col\_vo\_white, int& col\_vo\_blakc, bool& exit\_the\_str\_cucle, int AIturn, int k, int j);//королева бьёт

void can\_a\_queen\_beat(int i, int j, int size, char filed[][12], char awn\_pawn, char own\_queen, char enemy\_pawn, char enemy\_queen, int pawn\_beat\_hor[], int pawn\_beat\_ver[], int& col\_vo\_voz\_beat);//Может королева бить?

void turning\_a\_pawn(int hor, int ver, char field[][12], char queen\_white, char queen\_black, int turn\_of\_the\_more);//Превращение пешки в дамку если она дошла до границы

void process\_of\_eating\_pawn(char field[][12], int& hor, int& ver, int& new\_hor, int& new\_ver, int turn\_of\_the\_more, int& col\_vo\_black, int& col\_vo\_white, int i, int j);//процесс бития пешки