#include <iostream>

#include <conio.h>

#include "Header.h"

using namespace std;

void can\_a\_queen\_beat(int i, int j, int size, char field[][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?

{

int ost\_v\_v = 1, ost\_v\_n = 1, ost\_n\_v = 1, ost\_n\_n = 1;

for (int k = 1; k < size; k++)

{

if (ost\_v\_v == 1)

{

if (((field[i + k][j + k] == enemy\_pawn || field[i + k][j + k] == enemy\_queen) && field[i + (k + 1)][j + (k + 1)] == ' ') && i + k + 1 <= 9 && j + k + 1 <= 9)

{

spaceFIELD();

cout << "Белая дамка на позиции " << j - 1 << " ; " << i - 1 << " может бить\n";

pawn\_beat\_hor[col\_vo\_voz\_beat] = i;

pawn\_beat\_ver[col\_vo\_voz\_beat] = j;

col\_vo\_voz\_beat++;

}

else if (field[i + k][j + k] != ' ' && field[i + k + 1][j + k + 1] != ' ' || field[i + k][j + k] == awn\_pawn || field[i + k][j + k] == own\_queen)

{

ost\_v\_v = 2;

}

}

if (ost\_v\_n == 1)

{

if (((field[i + k][j - k] == enemy\_pawn || field[i + k][j - k] == enemy\_queen) && field[i + (k + 1)][j - (k + 1)] == ' ') && i + k + 1 <= 9 && j - k - 1 >= 2)

{

spaceFIELD();

cout << "Белая дамка на позиции " << j - 1 << " ; " << i - 1 << " может бить\n";

pawn\_beat\_hor[col\_vo\_voz\_beat] = i;

pawn\_beat\_ver[col\_vo\_voz\_beat] = j;

col\_vo\_voz\_beat++;

}

else if (field[i + k][j - k] != ' ' && field[i + (k + 1)][j - (k + 1)] != ' ' || field[i + k][j - k] == awn\_pawn || field[i + k][j - k] == own\_queen)

{

ost\_v\_n = 2;

}

}

if (ost\_n\_v == 1)

{

if (((field[i - k][j + k] == enemy\_pawn || field[i - k][j + k] == enemy\_queen) && field[i - (k + 1)][j + (k + 1)] == ' ') && i - k - 1 >= 2 && j + k + 1 <= 9)

{

spaceFIELD();

cout << "Белая дамка на позиции " << j - 1 << " ; " << i - 1 << " может бить\n";

pawn\_beat\_hor[col\_vo\_voz\_beat] = i;

pawn\_beat\_ver[col\_vo\_voz\_beat] = j;

col\_vo\_voz\_beat++;

}

else if (field[i - k][j + k] != ' ' && field[i - (k + 1)][j + (k + 1)] != ' ' || field[i - k][j + k] == awn\_pawn || field[i - k][j + k] == own\_queen)

{

ost\_n\_v = 2;

}

}

if (ost\_n\_n == 1)

{

if (((field[i - k][j - k] == enemy\_pawn || field[i - k][j - k] == enemy\_queen) && field[i - (k + 1)][j - (k + 1)] == ' ') && i - k - 1 >= 2 && j - k - 1 >= 2)

{

spaceFIELD();

cout << "Белая дамка на позиции " << j - 1 << " ; " << i - 1 << " может бить\n";

pawn\_beat\_hor[col\_vo\_voz\_beat] = i;

pawn\_beat\_ver[col\_vo\_voz\_beat] = j;

col\_vo\_voz\_beat++;

}

else if (field[i - k][j - k] != ' ' && field[i - (k + 1)][j - (k + 1)] != ' ' || field[i - k][j - k] == awn\_pawn || field[i - k][j - k] == own\_queen)

{

ost\_n\_n = 2;

}

}

}

}

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

{

for (int i = 0; i < 8; i++)//цикл для проверки не пошла ли королева на недоступные поля, а для того чтоб это проверить надо от вертикали и горизонтали отнять одно и тоже число, чтоб получилась позиция королевы

{

if (new\_poz\_ver - i == poz\_ver && new\_poz\_hor - i == poz\_hor)//если новая позиция королевы выше изначальной по горизонтали и вертикали

{

for (int j = 1; j <= i; j++)

{

if (field[poz\_hor + j][poz\_ver + j] != ' ' && j != i)

{

pawn\_pos\_inp\_error(new\_poz\_ver, new\_poz\_hor, AIturn);

break;

}

else if (j == i)

{

exit\_the\_str\_cucle = true;

i = 10;

break;

}

}

if (i == 10)

{

break;

}

}

else if (new\_poz\_ver + i == poz\_ver && new\_poz\_hor + i == poz\_hor)//если новая позиция королевы ниже по вертикали и горизонтали

{

for (int j = 1; j <= i; j++)

{

if (field[poz\_hor - j][poz\_ver - j] != ' ' && j != i)

{

pawn\_pos\_inp\_error(new\_poz\_ver, new\_poz\_hor, AIturn);

break;

}

else if (j == i)

{

exit\_the\_str\_cucle = true;

i = 10;

break;

}

}

if (i == 10)

{

break;

}

}

else if (new\_poz\_ver + i == poz\_ver && new\_poz\_hor - i == poz\_hor)//если новая позиция королевы по вертикали ниже, а по горизонтали выше

{

for (int j = 1; j <= i; j++)

{

if (field[poz\_hor + j][poz\_ver - j] != ' ' && j != i)

{

pawn\_pos\_inp\_error(new\_poz\_ver, new\_poz\_hor, AIturn);

break;

}

else if (j == i)

{

exit\_the\_str\_cucle = true;

i = 10;

break;

}

}

if (i == 10)

{

break;

}

}

else if (new\_poz\_ver - i == poz\_ver && new\_poz\_hor + i == poz\_hor)//если новая позиция королевы по вертикали выше, а по горизонтали ниже

{

for (int j = 1; j <= i; j++)

{

if (field[poz\_hor - j][poz\_ver + j] != ' ' && j != i)

{

pawn\_pos\_inp\_error(new\_poz\_ver, new\_poz\_hor, AIturn);

break;

}

else if (j == i)

{

exit\_the\_str\_cucle = true;

i = 10;

break;

}

}

if (i == 10)

{

break;

}

}

else if (i == 7)//ошибка в случае если вообще ниразу не совпало

{

pawn\_pos\_inp\_error(new\_poz\_ver, new\_poz\_hor, AIturn);

}

}

}

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

{

int ver = new\_poz\_ver, hor = new\_poz\_hor;

field[new\_poz\_hor][new\_poz\_ver] = ' ';

if (AIturn == 1)

{

new\_poz\_ver = rand() % 7 + 2;

new\_poz\_hor = rand() % 7 + 2;

}

else

{

spaceFIELD();

cout << "Введите поле, на которое походит дамка\n";

cin >> new\_poz\_ver >> new\_poz\_hor;

new\_poz\_hor++;

new\_poz\_ver++;

}

for(int i = 1; i <= 8; i++)

{

if (new\_poz\_ver - j \* i == poz\_ver && new\_poz\_hor - k \* i == poz\_hor && new\_poz\_hor > hor)

{

if (k > 0)

{

if (j > 0)

{

if (new\_poz\_hor > hor && new\_poz\_ver > ver)

{

break;

}

}

else

{

if (new\_poz\_hor > hor && new\_poz\_ver < ver)

{

break;

}

}

}

else

{

if (j > 0)

{

if (new\_poz\_hor < hor && new\_poz\_ver > ver)

{

break;

}

}

else

{

if (new\_poz\_hor < hor && new\_poz\_ver < ver)

{

break;

}

}

}

}

else

{

if (i == 8)

{

pawn\_pos\_inp\_error(new\_poz\_ver, new\_poz\_hor, AIturn);

i = 0;

}

}

}

check\_a\_new\_position(exit\_the\_str\_cucle, new\_poz\_ver, new\_poz\_hor, poz\_ver, poz\_hor, field, 'B', 'D', 'A', 'C', AIturn);

field[new\_poz\_hor][new\_poz\_ver] = field[poz\_hor][poz\_ver];

field[poz\_hor][poz\_ver] = ' ';

who\_needs\_to\_take\_away\_figure(turn\_of\_the\_more, col\_vo\_blakc, col\_vo\_white);

can\_the\_queen\_beat\_more(new\_poz\_ver, new\_poz\_hor, poz\_ver, poz\_hor, exit\_the\_str\_cucle, field, col\_vo\_white, col\_vo\_blakc, turn\_of\_the\_more, 8, AIturn);

exit\_the\_str\_cucle = true;

}

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

{

int verM\_horM = 0, verM\_horB = 0, verB\_horM = 0, verB\_horB = 0;

for(int i = 1; i < 8; i++)

{

poz\_ver = new\_poz\_ver;

poz\_hor = new\_poz\_hor;

if (verM\_horM == 0 && i != 0)

{

if (field[poz\_hor - i][poz\_ver - i] != ' ' && field[poz\_hor - (i + 1)][poz\_ver - (i + 1)] == ' ')

{

if (turn\_of\_the\_more % 2 == 0)

{

the\_queen\_repeat\_bets(field, new\_poz\_ver, new\_poz\_hor, poz\_ver, poz\_hor, 'A', 'C', 'B', 'D', AIturn, -1, -1);

}

else

{

the\_queen\_repeat\_bets(field, new\_poz\_ver, new\_poz\_hor, poz\_ver, poz\_hor, 'B', 'D', 'A', 'C', AIturn, -1, -1);

}

who\_needs\_to\_take\_away\_figure(turn\_of\_the\_more, col\_vo\_black, col\_vo\_white);

verM\_horM = 0;

verB\_horM = 0;

verB\_horB = 0;

verM\_horB = 0;

i = 0;

}

else if (field[poz\_hor - i][poz\_ver - i] != ' ' && field[poz\_hor - (i + 1)][poz\_ver - (i + 1)] != ' ')

{

verM\_horM = 1;

}

}

if (verM\_horB == 0 && i != 0)

{

if (field[poz\_hor + i][poz\_ver - i] != ' ' && field[poz\_hor + (i + 1)][poz\_ver - (i + 1)] == ' ')

{

if (turn\_of\_the\_more % 2 == 0)

{

the\_queen\_repeat\_bets(field, new\_poz\_ver, new\_poz\_hor, poz\_ver, poz\_hor, 'A', 'C', 'B', 'D', AIturn, 1, -1);

}

else

{

the\_queen\_repeat\_bets(field, new\_poz\_ver, new\_poz\_hor, poz\_ver, poz\_hor, 'B', 'D', 'A', 'C', AIturn, 1, -1);

}

who\_needs\_to\_take\_away\_figure(turn\_of\_the\_more, col\_vo\_black, col\_vo\_white);

verM\_horB = 0;

verB\_horM = 0;

verM\_horM = 0;

verB\_horB = 0;

i = 0;

}

else if (field[poz\_hor + i][poz\_ver - i] != ' ' && field[poz\_hor + (i + 1)][poz\_ver - (i + 1)] != ' ')

{

verM\_horB = 1;

}

}

if (verB\_horM == 0 && i != 0)

{

if (field[poz\_hor - i][poz\_ver + i] != ' ' && field[poz\_hor - (i + 1)][poz\_ver + (i + 1)] == ' ')

{

if (turn\_of\_the\_more % 2 == 0)

{

the\_queen\_repeat\_bets(field, new\_poz\_ver, new\_poz\_hor, poz\_ver, poz\_hor, 'A', 'C', 'B', 'D', AIturn, -1, 1);

}

else

{

the\_queen\_repeat\_bets(field, new\_poz\_ver, new\_poz\_hor, poz\_ver, poz\_hor, 'B', 'D', 'A', 'C', AIturn, -1, 1);

}

who\_needs\_to\_take\_away\_figure(turn\_of\_the\_more, col\_vo\_black, col\_vo\_white);

verB\_horM = 0;

verM\_horM = 0;

verB\_horB = 0;

verM\_horB = 0;

i = 0;

}

else if (field[poz\_hor - i][poz\_ver + i] != ' ' && field[poz\_hor - (i + 1)][poz\_ver + (i + 1)] != ' ')

{

verB\_horM = 1;

}

}

if (verB\_horB == 0 && i != 0)

{

if (field[poz\_hor + i][poz\_ver + i] != ' ' && field[poz\_hor + (i + 1)][poz\_ver + (i + 1)] == ' ')

{

if (turn\_of\_the\_more % 2 == 0)

{

the\_queen\_repeat\_bets(field, new\_poz\_ver, new\_poz\_hor, poz\_ver, poz\_hor, 'A', 'C', 'B', 'D', AIturn, 1, 1);

}

else

{

the\_queen\_repeat\_bets(field, new\_poz\_ver, new\_poz\_hor, poz\_ver, poz\_hor, 'B', 'D', 'A', 'C', AIturn, 1, 1);

}

who\_needs\_to\_take\_away\_figure(turn\_of\_the\_more, col\_vo\_black, col\_vo\_white);

verB\_horB = 0;

verB\_horM = 0;

verM\_horM = 0;

verM\_horB = 0;

i = 0;

}

else if (field[poz\_hor + i][poz\_ver + i] != ' ' && field[poz\_hor + (i + 1)][poz\_ver + (i + 1)] != ' ')

{

verB\_horB = 1;

}

}

}

}

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

{

bool exit\_the\_str\_cucle = false;

system("cls");

filed\_out\_durig\_the\_game(field, 12);

if (AIturn == 1)

{

new\_poz\_ver = rand() % 7 + 2;

new\_poz\_hor = rand() % 7 + 2;

}

else

{

spaceFIELD();

cout << "У дамки есть возможность побить, бейте\n";

cin >> new\_poz\_ver >> new\_poz\_hor;

new\_poz\_hor++;

new\_poz\_ver++;

}

while (field[new\_poz\_hor][new\_poz\_ver] != alien\_pawn && field[new\_poz\_hor][new\_poz\_ver] != alien\_queen)//проверка новой позиции

{

pawn\_pos\_inp\_error(new\_poz\_ver, new\_poz\_hor, AIturn);

}

check\_a\_new\_position(exit\_the\_str\_cucle, new\_poz\_ver, new\_poz\_hor, poz\_ver, poz\_hor, field, 'B', 'D', 'A', 'C', AIturn);

field[new\_poz\_hor][new\_poz\_ver] = ' ';

if (AIturn == 1)

{

new\_poz\_ver = rand() % 7 + 2;

new\_poz\_hor = rand() % 7 + 2;

}

else

{

spaceFIELD();

cout << "Введите поле, на которое походит дамка\n";

cin >> new\_poz\_ver >> new\_poz\_hor;

new\_poz\_hor++;

new\_poz\_ver++;

}

for(int i = 1; i < 8; i++)

{

if (new\_poz\_ver - j \* i == poz\_ver && new\_poz\_hor - k \* i == poz\_ver)

{

break;

}

else

{

if (i < 8)

{

i++;

}

else

{

pawn\_pos\_inp\_error(new\_poz\_ver, new\_poz\_hor, AIturn);

i = 1;

}

}

}

check\_a\_new\_position(exit\_the\_str\_cucle, new\_poz\_ver, new\_poz\_hor, poz\_ver, poz\_hor, field, 'B', 'D', 'A', 'C', AIturn);

field[new\_poz\_hor][new\_poz\_ver] = field[poz\_hor][poz\_ver];

field[poz\_hor][poz\_ver] = ' ';

poz\_hor = new\_poz\_hor;

poz\_ver = new\_poz\_ver;

}