```
// Queens.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"


/* Описание алгоритма:
 *
 * Для решения задачи испольщуется алгоритм перебора с возвратом.
 * Ферзи ставятся по очереди на любое свободное на каждом шаге место.
 * После того как поставили ферзя - битые клетки помечаются и другие
 * ферзи на них не ставятся, это обеспечивает то, что в любой момент времени
 * на доске никто никого не бьёт.
 * Если найдено решение - оно печатается, если ферзя поставить некуда,
 * то возвращаемся на предыдущий шаг и ставим предыдущего ферзя на другое,
 * допустимое для него место.
 * Сделана оптимизация: так как ферзей 8 и горизонталей 8, а 2 ферзя на одной
 * горизонтали не могут быть, то каждый ферзь размещается на горизонтали по номеру равной
 * (или большей для тупиковых вариантов перебора) номеру ферзя.
 *
 */


// Переменные, используемые в алгоритме:

int board[8][8];     // Шахматная доска 8x8.
                     // Каждая клетка может принимать следующие значения:
                     // CELL_FREE    (0) - пустая клетка и не под ударом поставленных на
    доске ферзей
                     // CELL_QUEEN   (1) - в клетке стоит ферзь
                     // CELL_ATTACK  (2) - клетка под ударом

// значения, принимаемые клетками шахматной доски
#define CELL_FREE   0   // пустая клетка и не под ударом поставленных на доске ферзей
#define CELL_QUEEN  1   // в клетке стоит ферзь
#define CELL_ATTACK 2   // клетка под ударом


// Начальная очистка доски
void init_board()
{
    int i, j;   // Переменные для работы с доской
    for (i=0; i<8; ++i)      // перебор по горизонтали
        for (j=0; j<8; ++j) // перебор по вертикали
            board[i][j] = CELL_FREE;    // клетка - пустая и не под ударом
}

// Пометить клетки, находящиеся под атакой ферзя в board[i][j]
void mark_attack(int queen_i, int queen_j)
{
    int i, j;   // Переменные для работы с доской

    // помечаем клетки по горизонтали под ударом
    for (j=0; j<8; ++j) board[queen_i][j] = CELL_ATTACK;

    // помечаем клетки по вертикали под ударом
    for (i=0; i<8; ++i) board[i][queen_j] = CELL_ATTACK;

    // по диагоналям
    for (i = queen_i, j = queen_j;
        (i < 8) && (j < 8);
        ++i, ++j) board[i][j] = CELL_ATTACK;
    for (i = queen_i, j = queen_j;
        (i < 8) && (j >=0);
        ++i, --j) board[i][j] = CELL_ATTACK;
    for (i = queen_i, j = queen_j;
        (i >= 0) && (j >= 0);
        --i, --j) board[i][j] = CELL_ATTACK;
    for (i = queen_i, j = queen_j;
        (i >= 0) && (j < 8);
        --i, ++j) board[i][j] = CELL_ATTACK;

    // первые 2 пункта затерли пометку ферзя, восстановим:
    board[queen_i][queen_j] = CELL_QUEEN;
```

```
}

// Снять пометки, находящиеся под атакой ферзя в board[i][j] и убрать самого ферзя
void unmark_attack(int queen_i, int queen_j)
{
    int i, j;   // Переменные для работы с доской

    // помечаем клетки по горизонтали пустыми
    for (j=0; j<8; ++j) board[queen_i][j] = CELL_FREE;

    // помечаем клетки по вертикали
    for (i=0; i<8; ++i) board[i][queen_j] = CELL_FREE;

    // по диагоналям
    for (i = queen_i, j = queen_j;
        (i < 8) && (j < 8);
        ++i, ++j) board[i][j] = CELL_FREE;
    for (i = queen_i, j = queen_j;
        (i < 8) && (j >= 0);
        ++i, --j) board[i][j] = CELL_FREE;
    for (i = queen_i, j = queen_j;
        (i >= 0) && (j >= 0);
        --i, --j) board[i][j] = CELL_FREE;
    for (i = queen_i, j = queen_j;
        (i >= 0) && (j < 8);
        --i, ++j) board[i][j] = CELL_FREE;
}

// Печать решения
void print_solution()
{
    int i, j;   // Переменные для работы с доской

    static int cnt = 0;

    cout << "Найдено решение: " << ++cnt << endl;
    cout << endl;

    cout << "+-+-+-+-+-+-+-+-+" << endl;
    for (i=0; i<8; ++i)     // перебор по горизонтали
    {
        cout << "|" ;
        for (j=0; j<8; ++j) // перебор по вертикали
            cout << (board[i][j] == CELL_QUEEN ? "X" : " " ) << "|";
        cout << endl;
        cout << "+-+-+-+-+-+-+-+-+" << endl;
    }

    cout << endl;
}

// Перебор вариантов расстановок ферзей
// queen_number - номер (по порядку) расставляемого ферзя
void gen(int queen_number)
{
    int i, j;   // Переменные для работы с доской:
                // номер клетки по горизонтали и номер по вертикали
    int k, p;   // Переменные для работы с доской:
                // номер клетки по горизонтали и номер по вертикали


    ////////////////////////////
    // 1. Поиск свободного места

    for (i=queen_number-1; i<8; ++i)        // перебор по горизонтали
        for (j=0; j<8; ++j) // перебор по вертикали
            if (board[i][j] == CELL_FREE)   // если клетка свободна
                                            // и не под ударом ранее поставленных ферзей
            {
                ////////////////////////////////////
                // 2. Ставим ферзя и помечаем клетку

                board[i][j] = CELL_QUEEN;   // ставим ферзя
                mark_attack(i, j);      // помечаем область атаки этого ферзя
```

```
                ////////////////////////////////////////////////////////////////////////////////
    /
                // 3. Если это не 8-ой ферзь, то перебираем варианты расстановки следующего
                //    иначе печатаем решение

                if (queen_number != 8)
                {
                    // это не 8-ой ферзь, перебираем варианты расстановки следующего
                    gen(queen_number+1);
                } else {
                    // печатаем решение
                    print_solution();
                    cout << i << ":" << j << endl;
                }

                ////////////////////////////////////////////
                // 2. Снимаем ферзя и восстанавливаем пометки

                unmark_attack(i, j);
                // Мы могли затереть лишние пометки атак других ферзей, поэтому
    перепомечаем
                for (k=0; k<8; ++k)      // перебор по горизонтали
                    for (p=0; p<8; ++p) // перебор по вертикали
                        if (board[k][p] == CELL_ATTACK)
                            board[k][p] = CELL_FREE;    // стираем все пометки атак
                for (k=0; k<8; ++k)      // перебор по горизонтали
                    for (p=0; p<8; ++p) // перебор по вертикали
                        if (board[k][p] == CELL_QUEEN)
                            mark_attack(k, p);          // расставляем пометки атак для
    каждого ферзя
            }
}

// краткий вариант записи начала перебора с первого ферзя
void gen_all()
{
    gen(1);
}


int main(int argc, char* argv[])
{
    // Выводим на консоль заголовок программы
    cout << "Лабораторная работа N6" << endl;
    cout << "Тема: Алгоритм перебора с возвратом" << endl;
    cout << endl;
    cout << "'Задача о расстановке ферзей'" << endl;
    cout << endl;
    cout << "Найти все варианты расстановки ферзей" << endl;
    cout << "на шахматной доске размера 8x8 так, чтобы" << endl;
    cout << "ни один ферзь не был под ударом другого." << endl;
    cout << endl;

    // Начальная очистка доски
    init_board();
    // Перебор вариантов расстановок ферзей
    cout << "Поиск решений..." << endl;
    cout << endl;
    gen_all();

    system("pause");

    return 0;

}
```