Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

ПНИПУ

**Лабораторная работа Рекурсия**

**«Восемь ферзей»**

Выполнил:

студент группы РИС-23-2б

Береснев Александр Сергеевич

Проверила:

доцент кафедры ИТАС

О.А. Полякова

Пермь 2024 г.

**Постановка задачи:**

Дана шахматная доска 8 на 8, расставить на доске 8 ферзей так, чтобы каждой строке доски стоял один ферзь, при этом ни один ферзь не бил никакого другого.

**Примечания:**

1. Ферзь ходит по вертикали, горизонтали и диагонали;
2. Выполнить расстановку через рекурсивную функцию.

**Анализ:**

1. Вначале ферзь ставится на первую строку в первую встретившуюся свободную клетку, в матрице это координаты [0;0];
2. Отмечаем все клетки, которые бьёт ферзь;
3. Переходя на следующую строку, повторяем ранее описанные действия, с тем условием, что мы не можем ставить ферзя на отмеченные клетки;
4. В случае, если в следующей строке нет клетки для постановки ферзя, рекурсивно возвращаемся на столько строк обратно, чтобы расстановка ферзей удовлетворяла условию задачи.

**Реализация:**

1. В первую очередь создадим массив 8x8 и заполним его элементы нулями;
2. Далее создадим несколько функциональных блоков, каждый из которых отвечает за вверенную ему функциональную особенность:
3. Блок, отвечающий за решение задачи и соблюдение условий решения.

В первую очередь проверяет возможность постановки ферзя, если такая возможность есть, вызываем аналогичную функцию. Если возможности нет, то вызываем функцию удаления.

1. Блок постановки ферзя и отметки клеток.

Основная идея заключается в том, что при постановке ферзя нулевое значение элемента массива принимает значение, равное -1, а элементы массива, по которым ферзь может бить принимают значение +1. Таким образом, при постановке следующего ферзя, нас будут интересовать только нулевые элементы массива, означающие, что клетка доступна для постановки.

Если в следующей строке массива нет места для постановки, вызывается функция делающая «шаг назад» - удаление предыдущего ферзя.

1. Блок удаления ферзя.

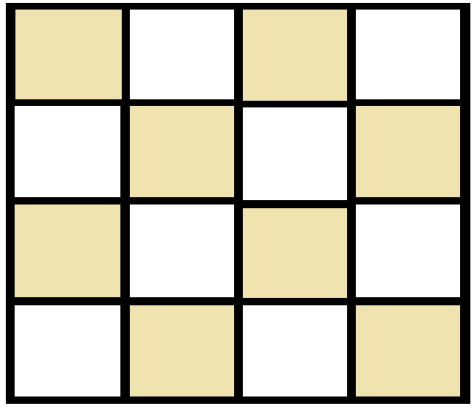
Как уже было описано ранее, отвечает за удаление ферзя, путём математического изменения значений элементов массива для достижения нулевого значения – признака свободной клетки;

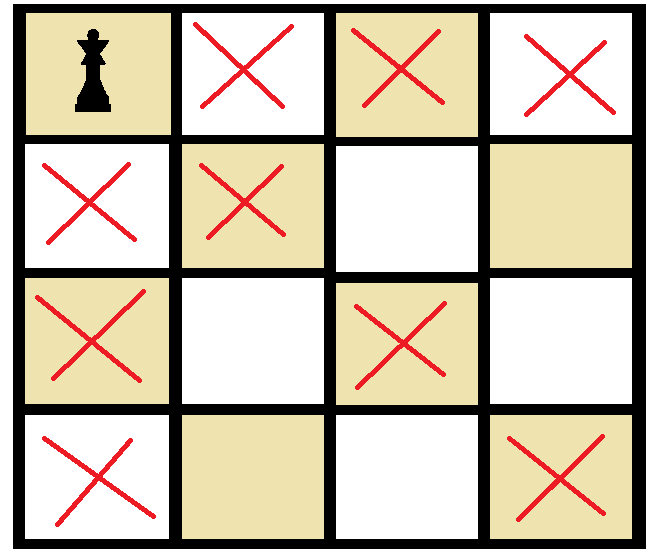
После выполнения этой функции в блоке постановки ферзя, индекс доступной клетки увеличивается на 1.

1. Блок «визуализации» - отвечает за пошаговую демонстрацию выполнения алгоритма.
2. В функции main вызываем ранее описанные функции создания массива и функцию, отвечающую за выполнение задачи.

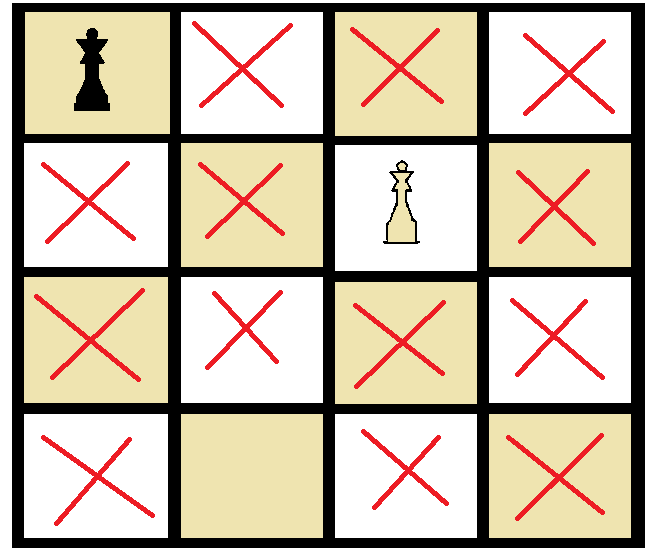
**Визуальная демонстрация принципа решения задачи:**

1. В самую первую свободную ячейку доски ставим ферзя и отмечаем клетки, которые он бьет:

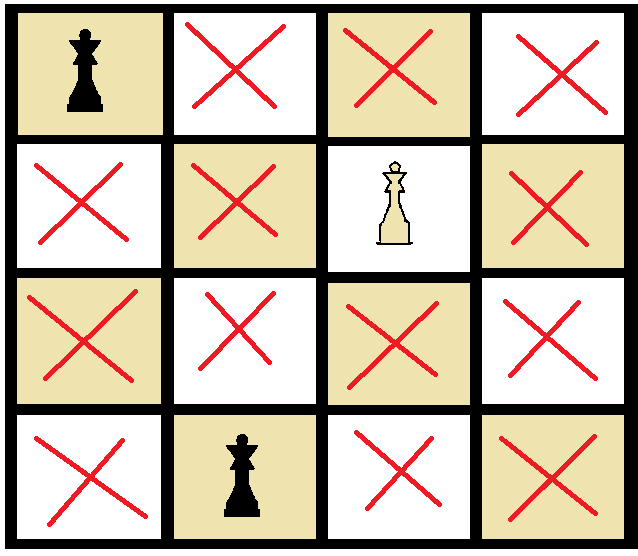




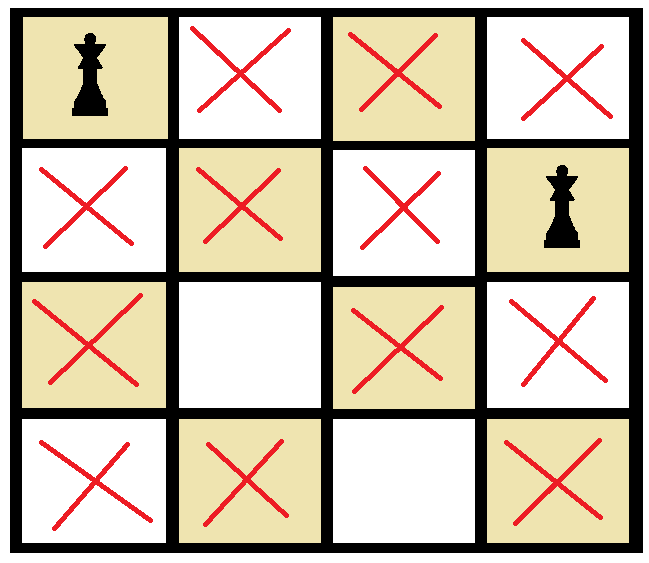
1. На следующую строку таблицы (доски), в свободную клетку подставляем следующего ферзя и также отмечаем клетки, по которым он будет бить:



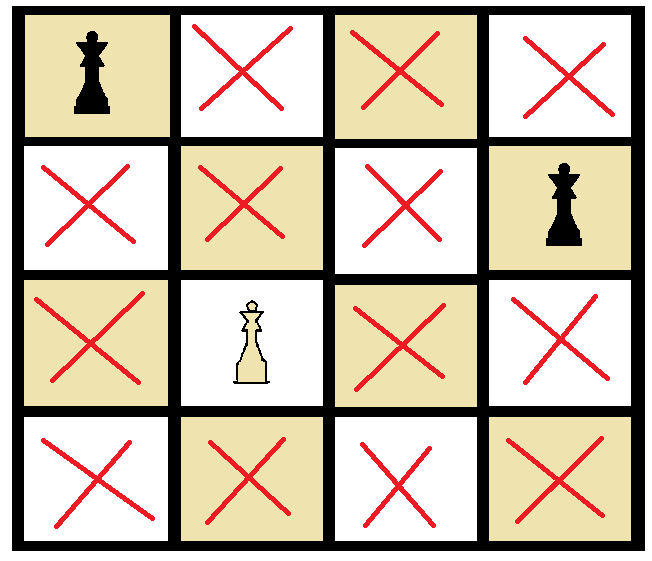
* Не смотря на то, что у нас, очевидно, не хватит места на 4 ферзя, для наглядной демонстрации, аналогично подставим следующего в единственную свободную ячейку:



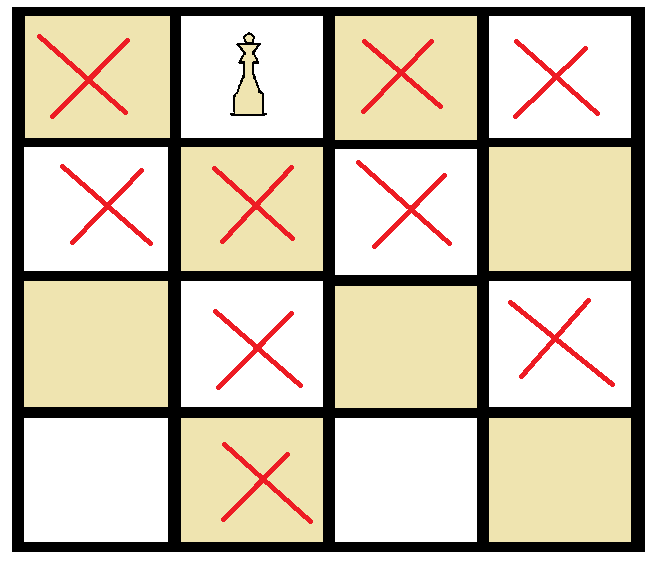
1. Вернёмся на второй шаг и попробуем подставить второго ферзя в другую клетку. В таком случае дальнейшая постановка будет выгладить следующим образом:



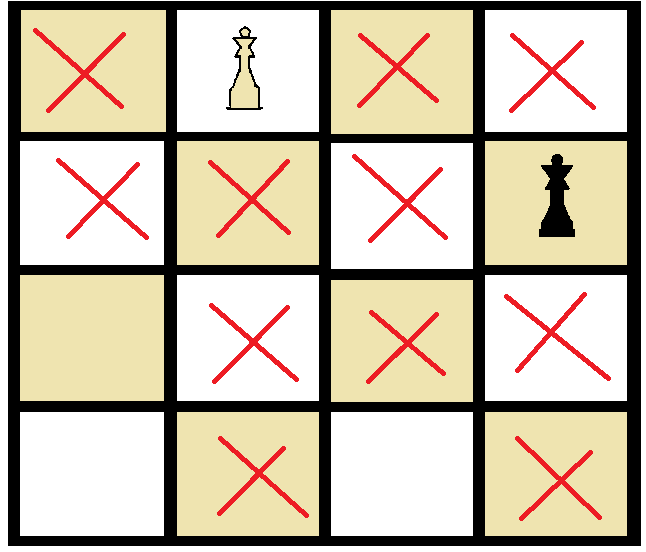
* Итоговый вариант:



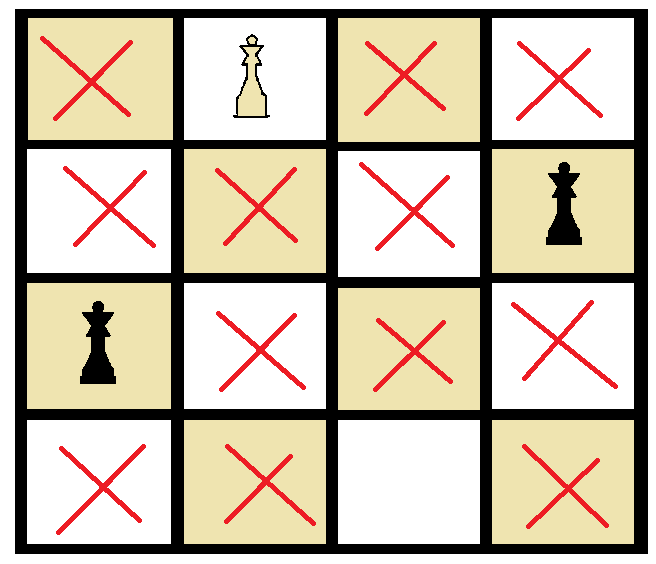
1. В связи с тем, что на втором шаге мы использовали все варианты решения, вернёмся на первый шаг и поставим ферзя в следующую свободную ячейку в строке:



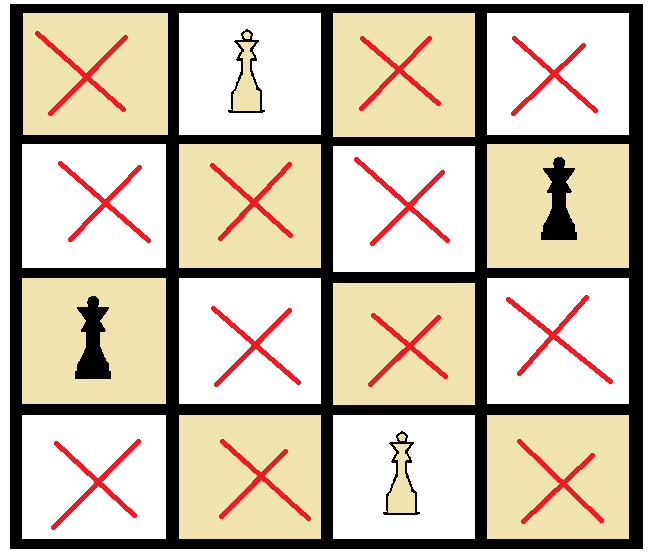
1. Подставим следующего ферзя в единственную свободную клетку в новой «строке»:



1. Повторим предыдущий шаг:



1. Аналогично повторим предыдущий шаг и получим необходимый вариант решения:



**Код в С++:**

#include <iostream>

#include <clocale>

using namespace std;

const int n = 8;

int board[n][n];

void Board() // Создание массива "доски" и заполнение её нулями

{

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

board[i][j] = 0;

}

}

}

void Display() // Отображение доски с учётом изменений

{

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

if (board[i][j] == -1)

{

cout << " Q "; // Ферзь

}

else if (board[i][j] > 0) {

cout << " X ";

}

else {

cout << " \_ "; // Клетка

}

}

cout << endl;

}

}

void Substitution(int i, int j) // Постановка ферзя

{

for (int x = 0; x < n; ++x)

{

board[x][j] += 1;

board[i][x] += 1;

int d;

d = j - i + x;

if (d >= 0 && d < n)

{

board[x][d] += 1;

}

d = j + i - x;

if (d >= 0 && d < n) {

board[x][d] += 1;

}

}

board[i][j] = -1;

cout << endl;

Display();

}

void Delete(int i, int j) // Удаление ферзя

{

for (int x = 0; x < n; ++x) {

board[x][j] -= 1;

board[i][x] -= 1;

int d;

d = j - i + x;

if (d >= 0 && d < n) {

board[x][d] -= 1;

}

d = j + i - x;

if (d >= 0 && d < n) {

board[x][d] -= 1;

}

}

cout << endl << " Возврат на шаг назад:" << endl;

Display();

board[i][j] = 0;

}

bool Solution(int i) // Запуск выполнения алгоритма

{

bool result = false; // "флаг" - для завершения работы функции

for (int j = 0; j < n; ++j) // Перебор строк

{

if (board[i][j] == 0) // Условие для поиска свободной клетки

{

Substitution(i, j); // Фызов функции для постановки в найденую клетку ферзя

if (i == 7) // Условие для проверки окончания работы функции

{

result = true; // смена "флага"

}

else if (!(result = Solution(i + 1)))

{

Delete(i, j); // Вызов функции для удаления ферзя

}

}

if (result) break;

}

return result;

}

int main()

{

setlocale(LC\_ALL, "Russian");

Board();

cout << endl << " Постановка Первого ферзя";

Solution(0);

cout << endl << " Итоговый вариант:" << endl;

Display();

return 0;

}

**Итоговый вывод в консоль:**

