Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

ПНИПУ

**Лабораторная работа  
“8 ферзей”**

Выполнил:   
студент группы РИС-23-1б   
Чунарев Дмитрий Николаевич

Проверила:   
доцент кафедры ИТАС   
О.А. Полякова

2024 г.

**Условие:** Расставить на стандартной 64-клеточной шахматной доске 8

ферзей так, чтобы ни один из них не находился под боем другого».

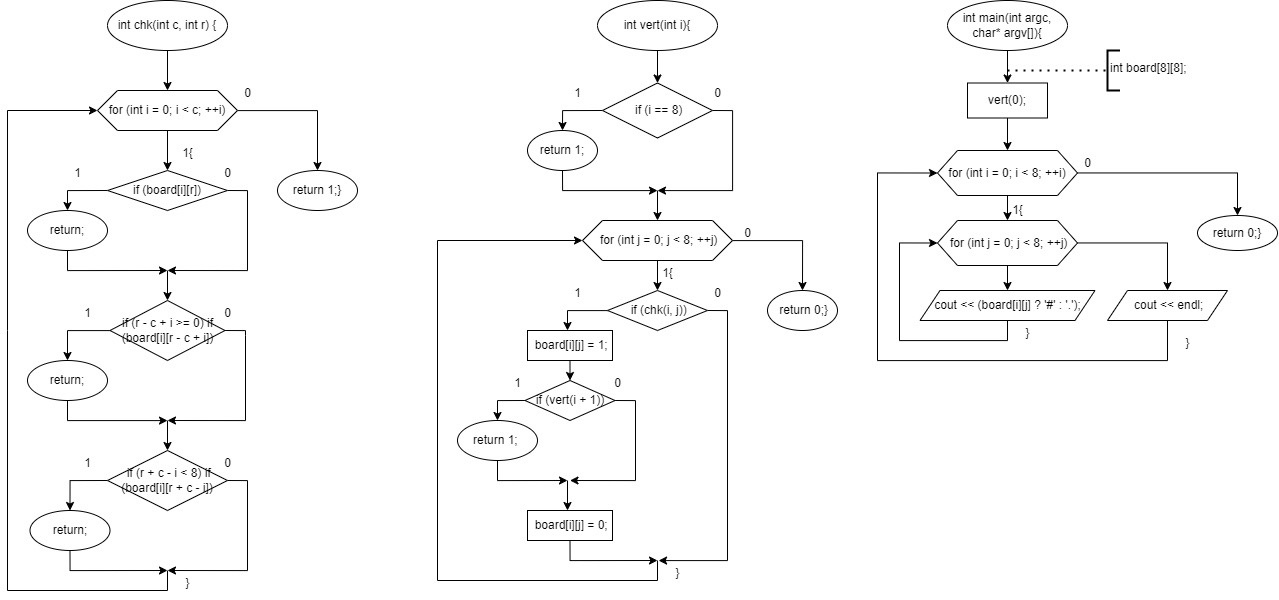
Подразумевается, что ферзь бьёт все клетки, расположенные по

вертикалям, горизонталям и обеим диагоналям.

**Анализ задачи:**

1. Начинаем с пустой шахматной доски.
2. Рекурсивно размещаем ферзей на доске, начиная с первого столбца.
3. На каждом шаге рекурсии перебираем строки текущего столбца.
4. Проверяем, можно ли разместить ферзя в текущей позиции, учитывая угрозы со стороны уже размещенных ферзей.
5. Если ферзя можно разместить, помечаем текущую позицию и переходим к следующему столбцу.
6. Повторяем шаги 3-5 для всех столбцов.
7. Если все ферзи размещены успешно, выводим решение.
8. Если на каком-то шаге невозможно разместить ферзя, возвращаемся на предыдущий шаг и пробуем другую позицию.
9. Если все возможные варианты перебраны и решение не найдено, выводим сообщение о том, что решение не существует.

**Блок-схема:**



**Код:**

#include <iostream>

using namespace std;

const int N = 8; // Размер доски

// Функция для проверки, можно ли поставить ферзя на доску в позиции (row, col)

bool isSafe(int board[N][N], int row, int col) {

int i, j;

// Проверяем эту строку слева

for (i = 0; i < col; ++i)

if (board[row][i])

return false;

// Проверяем верхнюю диагональ слева

for (i = row, j = col; i >= 0 && j >= 0; --i, --j)

if (board[i][j])

return false;

// Проверяем нижнюю диагональ слева

for (i = row, j = col; j >= 0 && i < N; ++i, --j)

if (board[i][j])

return false;

return true;

}

// Функция для вывода шахматной доски с цветовой разметкой

void printBoard(int board[N][N]) {

cout << " ";

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

cout << "\033[1;37m" << i << " ";

cout << endl;

for (int i = 0; i < N; ++i) {

cout << "\033[1;37m" << i << " ";

for (int j = 0; j < N; ++j) {

if ((i + j) % 2 == 0)

cout << "\033[48;5;235m"; // Зеленый фон для четных клеток

else

cout << "\033[48;5;234m"; // Белый фон для нечетных клеток

if (board[i][j])

cout << "\033[1;31mQ\033[0m "; // Красный цвет для ферзей

else

cout << " ";

}

cout << "\033[0m" << endl;

}

cout << endl;

}

// Рекурсивная функция для решения задачи о 8 ферзях

bool solveNQueensUtil(int board[N][N], int col) {

// Базовый случай: если все ферзи размещены, возвращаем true

if (col >= N)

return true;

// Перебираем строки в текущем столбце

for (int i = 0; i < N; ++i) {

// Проверяем, можно ли поставить ферзя в позицию (i, col)

if (isSafe(board, i, col)) {

// Расставляем ферзя

board[i][col] = 1;

// Выводим положение на текущем шаге

cout << "Шаг " << col + 1 << ": Размещаем ферзя в позиции (" << i << ", " << col << ")" << endl;

printBoard(board);

// Рекурсивно вызываем solveNQueensUtil для следующего столбца

if (solveNQueensUtil(board, col + 1))

return true;

// Если не можем разместить ферзя в данной позиции, возвращаемся и пробуем другую

board[i][col] = 0;

cout << "Шаг " << col + 1 << ": Удаляем ферзя из позиции (" << i << ", " << col << ")" << endl;

printBoard(board);

}

}

// Если не удалось разместить ферзя в текущем столбце, возвращаем false

return false;

}

// Функция для решения задачи о 8 ферзях

void solveNQueens() {

int board[N][N] = { 0 }; // Создаем пустую доску

// Вызываем вспомогательную рекурсивную функцию для размещения ферзей

if (!solveNQueensUtil(board, 0))

cout << "Решение не существует" << endl;

}

int main() {

setlocale(LC\_ALL, "ru");

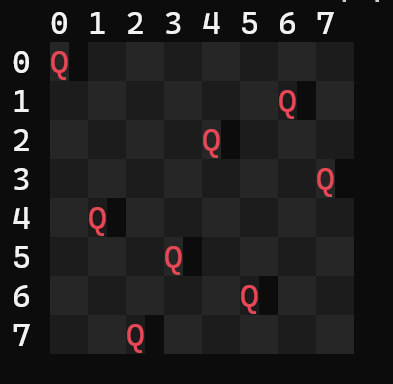
solveNQueens();

return 0;

}

**Вывод:**

****

****

**Итог:**

В конце работы, программа успешно выводит одно из решений задачи.