Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №13**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: Рекурсивные функции. Задача о расстановке восьми ферзей

Выполнила:

Студент группы ИВТ-20-2б

Ананина Арина Юрьевна

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

1. Постановка задачи

С помощью метода поиска с возвратом найти расстановку ферзей на шахматной доске, таким образом, чтобы ни один из них не находился под боем другого.

2. Анализ задачи

1. Используемые типы данных.

Тип int[][] используется для хранения шахматной доски.

int board[8][8];

Тип int используется для хранения целочисленных значений и параметра в цикле.

for (int j = 0; j < 8; j++)

2. Действия над используемыми данными.

Действия над типом int[][]: переприсваивание.

++board[x][j];

++board[i][x];

Действия над типом int: переприсваивание, инкремент.

j++

3. Виды данных в программе.

Для хранения шахматного поля используется двумерный статический массив размером 8 на 8.

int board[8][8];

4. Структуры в программе не используются.

5. Организация ввода и вывода.

Ввод в программе не предусмотрен.

Вывод осуществляется в консоль через оператор cout.

if (board[i][j] == -1) cout << "[]";

else cout << ". ";

6. Действия для решения задачи.

Печать шахматного поля.

Печать организована с помощью 2-х циклов с параметром. Если ячейка содержит ферзя, то выводится символ [] .

for (int i = 0; i < 8; ++i)

{

for (int j = 0; j < 8; ++j)

{

if (board[i][j] == -1) cout << "[]";

else cout << ". ";

}

cout << endl;

}

Проверка на возможность постановки ферзя в определенную ячейку.

Изначально все клетки доски заполнены нулями

Сначала ферзь ставится в первую клетку доски, клетки под боем с помощью декремента изменяются на 1. Далее с помощью цикла осуществляется проход по следующей строке, если в строке есть ноль, то ставится ферзь, если в строке ноль не найден, необходимо вернуться на предыдущую строку, удалить ферзя и попробовать другую комбинацию. Для этого используется рекурсивная функция try\_queen.

bool try\_queen(int i) {

bool result = false;

for (int j = 0; j < 8; ++j)

{

if (board[i][j] == 0)

{

set\_queen(i, j);

if (i == 7)

{

result = true;

}

else if (!(result = try\_queen(i + 1))) {

reset\_queen(i, j);

}

}

if (result) break;

}

return result;

}

Очистка линии от ферзей.

void reset\_queen(int i, int j) {

for (int x = 0; x < 8; ++x)

{

--board[x][j];

--board[i][x];

int foo;

foo = j - i + x;

if (foo >= 0 && foo < 8)

--board[x][foo];

foo = j + i - x;

if (foo >= 0 && foo < 8)

--board[x][foo];

board[i][j] = 0;

}

}

Рекурсивная функция поиска.

3. Код программы

#include <iostream>

using namespace std;

int board[8][8];

void set\_queen(int i, int j) {

for (int x = 0; x < 8; ++x)

{

++board[x][j];

++board[i][x];

int foo;

foo = j - i + x;

if (foo >= 0 && foo < 8)

++board[x][foo];

foo = j + i - x;

if (foo >= 0 && foo < 8)

++board[x][foo];

board[i][j] = -1;

}

}

void reset\_queen(int i, int j) {

for (int x = 0; x < 8; ++x)

{

--board[x][j];

--board[i][x];

int foo;

foo = j - i + x;

if (foo >= 0 && foo < 8)

--board[x][foo];

foo = j + i - x;

if (foo >= 0 && foo < 8)

--board[x][foo];

board[i][j] = 0;

}

}

bool try\_queen(int i) {

bool result = false;

for (int j = 0; j < 8; ++j)

{

if (board[i][j] == 0)

{

set\_queen(i, j);

if (i == 7)

{

result = true;

}

else if (!(result = try\_queen(i + 1))) {

reset\_queen(i, j);

}

}

if (result) break;

}

return result;

}

int main()

{

for (int i = 0; i < 8; ++i)

for (int j = 0; j < 8; ++j)

board[i][j] = 0;

try\_queen(0);

for (int i = 0; i < 8; ++i)

{

for (int j = 0; j < 8; ++j)

{

if (board[i][j] == -1) cout << "[]";

else cout << ". ";

}

cout << endl;

}

}#include <iostream>

using namespace std;

int board[8][8];

void set\_queen(int i, int j) {

for (int x = 0; x < 8; ++x)

{

++board[x][j];

++board[i][x];

int foo;

foo = j - i + x;

if (foo >= 0 && foo < 8)

++board[x][foo];

foo = j + i - x;

if (foo >= 0 && foo < 8)

++board[x][foo];

board[i][j] = -1;

}

}

void reset\_queen(int i, int j) {

for (int x = 0; x < 8; ++x)

{

--board[x][j];

--board[i][x];

int foo;

foo = j - i + x;

if (foo >= 0 && foo < 8)

--board[x][foo];

foo = j + i - x;

if (foo >= 0 && foo < 8)

--board[x][foo];

board[i][j] = 0;

}

}

bool try\_queen(int i) {

bool result = false;

for (int j = 0; j < 8; ++j)

{

if (board[i][j] == 0)

{

set\_queen(i, j);

if (i == 7)

{

result = true;

}

else if (!(result = try\_queen(i + 1))) {

reset\_queen(i, j);

}

}

if (result) break;

}

return result;

}

int main()

{

for (int i = 0; i < 8; ++i)

for (int j = 0; j < 8; ++j)

board[i][j] = 0;

try\_queen(0);

for (int i = 0; i < 8; ++i)

{

for (int j = 0; j < 8; ++j)

{

if (board[i][j] == -1) cout << "[]";

else cout << ". ";

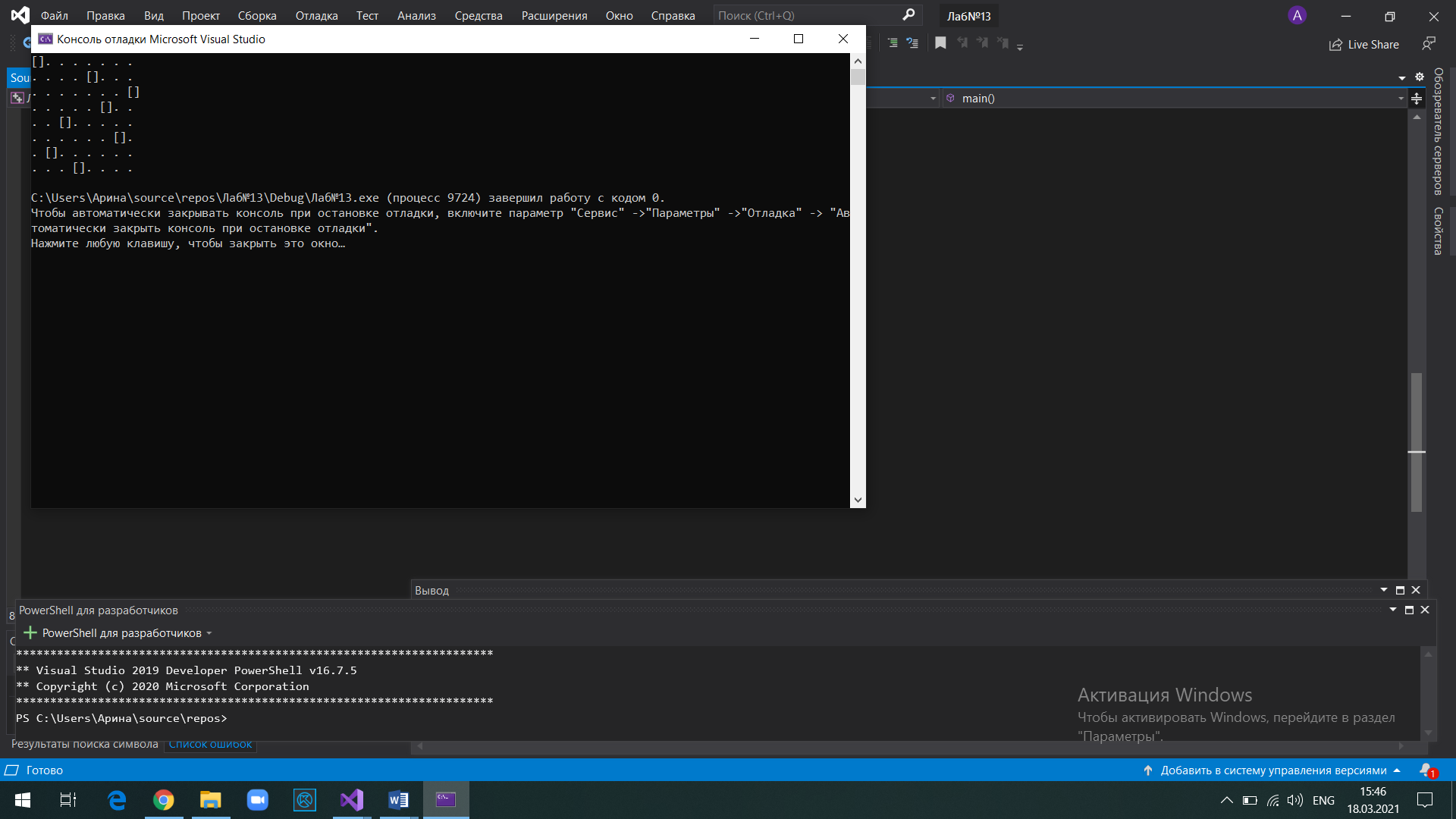
}

cout << endl;

}

}

4. Выполнение программы.



5. Блок схема

