Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №13**

Дисциплина: основы алгоритмизации и программирования

Тема: “Метод поиска с возвратом.

Задача о восьми ферзях”

Выполнил:

студент группы ИВТ-20-2Б

Брейкин Алексей

Проверила: доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**

Используя метод поиска с возвратом, расставить на стандартной 64-клеточной шахматной доске 8 ферзей так, чтобы ни один из них не находился под боем другого.

2

**Анализ задачи**

1. Для решения задачи необходимо:
   1. Организовать фукнцию DeskInit( ), которая будет заполнять изначально все клетки доски нулями
   2. Организовать функцию set\_ferz(), которая будет ставить ферзя и менять клетки под боем.
   3. Организовать функцию del, которая будет удалять ферзя и очищать клетки под его боем.
   4. Организовать функцию void ChoicePlace(int i, int j), в которой методом возврата будут вызываться функции bool SearchPlace(int i)и del для решения поставленной задачи.

**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Двумерный статический массив chessboard типа int.

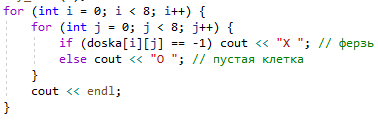
**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Через функцию SearchPlace отправляется позиция первого ферзя (первая клетка) в качестве параметра.

**4.** Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** В качестве ввода используется число как параметр в функцию SearchPlace.

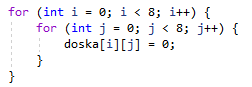
**4.2.** Для вывода используется цикл for в котором через функцию cout выводятся ферзи и пустые клетки.



**5.** Поставленные задачи будут решены следующими действиями:

3

**5.1.** В главной функции все клетки доски заполняются нулями, что означает, что они пустые.



**5.2.** В функции SearchPlace () в цикле for происходит итерирование по строке на доске (по горизонтали), в ветвлении идёт проверка на то, является ли клетка под боем одного из ферзя, если она пустая, то вызывается функция ChoicePlace (), в которой ставится ферзь (ферзь в памяти компьютера представляется как -1) и клетки под его боем (обозначаются инкрементом).

void ChoicePlace(int i, int j)

{

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

{

chessboard[k][j]++;

chessboard[i][k]++;

int diag1 = j - i + k;

int diag2 = j + i - k;

if (diag1 >= 0 && diag1 < 8)

{

chessboard[k][diag1]++;

}

if (diag2 >= 0 && diag2 < 8)

{

chessboard[k][diag2]++;

}

}

chessboard[i][j] = -1;

}

 После обработки функции ChoicePlace () используется ветвление, в котором проверяется, на каком по счёту ряду происходит итерация: если на последнем (7), то функция SearchPlace () может быть досрочно завершена через переменную result, иначе происходит вызов функции SearchPlace () с переходом на другой ряд рекурсивно, пока не будет достигнут последний ряд. Если на каком-то из рядов не получается поставить ферзя, то вызывается функция del.

void Del(int i, int j)

{

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

{

chessboard[k][j]--;

chessboard[i][k]--;

int diag1 = j - i + k;

int diag2 = j + i - k;

if (diag1 >= 0 && diag1 < 8)

{

chessboard[k][diag1]--;

}

if (diag2 >= 0 && diag2 < 8)

{

chessboard[k][diag2]--;

}

}

chessboard[i][j] = 0;

}

В функции del происходит декремент клеток под боем ферзя, которого необходимо удалить, а также удаление самого ферзя.

bool SearchPlace(int i)

{

bool res = false;

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

{

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

{

ChoicePlace(i, j);

if (i == 7)

{

res = true;

}

else if ((res = SearchPlace(i + 1)) == false)

{

Del(i, j);

}

}

if (res == true)

{

return res;

}

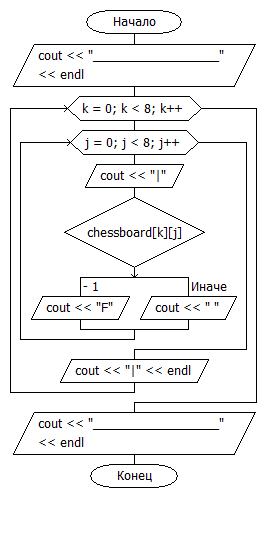
}

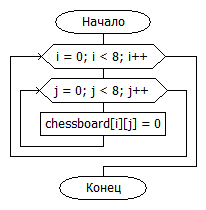
return res;

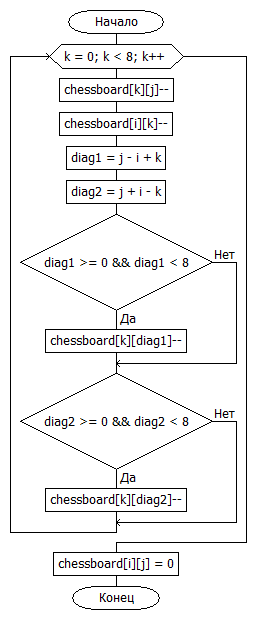
}

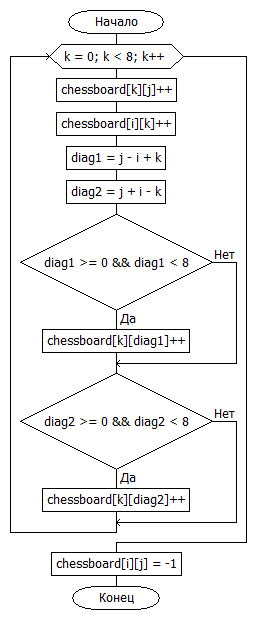
**Блок-схема**

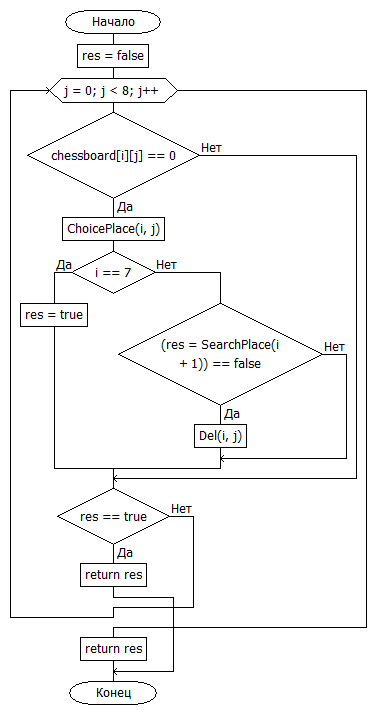












**Код**

#include <iostream>

using namespace std;

int chessboard[8][8];

void DeskInit();

void ChoicePlace(int i, int j);

void Del(int i, int j);

bool SearchPlace(int i);

void PrintDesk();

int main()

{

DeskInit();

SearchPlace(0);

PrintDesk();

}

void PrintDesk()

{

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

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

{

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

{

cout << "|";

switch (chessboard[k][j])

{

case -1: cout << "F"; break;

default: cout << " ";

}

}

cout << "|" << endl;

}

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

void DeskInit()

{

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

{

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

{

chessboard[i][j] = 0;

}

}

}

void Del(int i, int j)

{

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

{

chessboard[k][j]--;

chessboard[i][k]--;

int diag1 = j - i + k;

int diag2 = j + i - k;

if (diag1 >= 0 && diag1 < 8)

{

chessboard[k][diag1]--;

}

if (diag2 >= 0 && diag2 < 8)

{

chessboard[k][diag2]--;

}

}

chessboard[i][j] = 0;

}

void ChoicePlace(int i, int j)

{

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

{

chessboard[k][j]++;

chessboard[i][k]++;

int diag1 = j - i + k;

int diag2 = j + i - k;

if (diag1 >= 0 && diag1 < 8)

{

chessboard[k][diag1]++;

}

if (diag2 >= 0 && diag2 < 8)

{

chessboard[k][diag2]++;

}

}

chessboard[i][j] = -1;

}

bool SearchPlace(int i)

{

bool res = false;

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

{

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

{

ChoicePlace(i, j);

if (i == 7)

{

res = true;

}

else if ((res = SearchPlace(i + 1)) == false)

{

Del(i, j);

}

}

if (res == true)

{

return res;

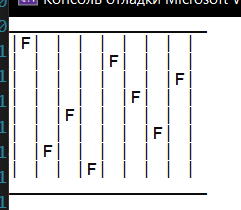
}

}

return res;

}

**Скриншоты**



12