Федеральное государственное автономное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет», ПНИПУ

ЛАБОРАТОРНАЯ РАБОТА

ЗАДАЧА О ВОСЬМИ ФЕРЗЯХ

Выполнил: студент группы РИС-23-3б

Артем Владимирович Швецов

Проверила: доцент кафедры ИТАС

Ольга Андреевна Полякова

Пермь 2023

**Постановка задачи**

Составить рекурсивную функцию, которая должна решать задачу о восьми ферзях. Формулировка задачи о восьми ферзях: «Расставить на стандартной 64-клеточной шахматной доске 8 ферзей так, чтобы ни один из них не находился под боем другого».

**Анализ задачи**

1. Для просмотра и сохранения данных о положении ферзей необходимо создать дополнительное хранилище. Так как необходим также и графический вывод положения ферзей, создается двумерный массив, в котором будут сохраняться положения ферзей и полей под боем.
2. В начале программы пользователю выводится запрос на ввод начального положения первого ферзя и запрос на включение функции вывода всех позиций ферзей следующей строки не под боем.
3. Для нахождения свободных клеток просматриваем позиции битых клеток, полученных на предыдущем уровне рекурсии. Если значение в клетке равно 0, то на эту клетку можно поставить ферзя.
4. Выполнение рекурсии продолжается до тех пор, пока не будет найдена позиция последнего ферзя.
5. Если пользователь хочет выводить шаги рекурсии, при нахождении свободной выводим текущее состояние доски.
6. При финальном выводе доски выводим последовательно расставления всех восьми ферзей.

**Блок-схема**

Все блок схемы расположены в конце документа.

**Код**

#include <iostream>

using namespace std;

static short int desk[66][8] = { {0, 1, 2, 3, 4, 5, 6, 7} };

bool show\_desk = false;

void desk\_rec(int n);

void desk\_print(int n, int curs);

void desk\_place(int n);

void desk\_print\_all();

int main()

{

system("chcp 1251");

int n;

cout << "Введите номер столбца в верхней строке, с которого хотите начать: ";

cin >> desk[1][0];

desk[1][0]--;

cout << "Если хотите просматривать каждый отдельный шаг рекурсии, введите 1, введите любую другую цифру для вывода только финальной позиции: ";

cin >> n;

switch (n)

{

case 1:

cout << "Вывод выполнен в псевдографическом стиле, где x - клетки под атакой, Q - ферзи." << endl;

show\_desk = true;

}

cout << endl;

system("chcp 866");

desk\_rec(0);

system("chcp 1251");

cout << "Финальный вариант расставления" << endl << endl;

system("chcp 866");

desk\_print\_all();

system("chcp 1251");

system("pause");

return 0;

}

void desk\_rec(int n)

{

if (desk[1][7] == 0)

{

if (n == 0)

{

desk\_place(0);

if (show\_desk)

{

desk\_print(0, -1);

system("chcp 1251");

cout << endl;

cout << "В столбце " << 1 << " строки " << desk[1][0] << " найдена свободная клетка." << endl;

cout << "\tПереход на более глубокий уровень рекурсии. Вызов desk\_rec(" << 1 << ")" << endl << endl;

system("pause");

system("chcp 866");

cout << endl;

}

desk\_rec(n + 1);

}

else

{

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

{

if (show\_desk)

{

desk\_print(n - 1, i);

system("pause");

}

if (desk[n + (n - 1) \* 8 + 2][i] == 0 && (desk[1][7] == 0))

{

desk[1][n] = i;

desk\_place(n);

if (show\_desk)

{

desk\_print(n, -1);

system("chcp 1251");

cout << endl;

cout << "В столбце " << i + 1 << " строки " << n + 1 << " найдена свободная клетка." << endl;

cout << "\tПереход на более глубокий уровень рекурсии. Вызов desk\_rec(" << n + 1 << ")" << endl << endl;

system("pause");

system("chcp 866");

cout << endl;

}

desk\_rec(n + 1);

if ((desk[1][7] == 0) && (show\_desk))

{

system("chcp 1251");

cout << endl;

cout << "Нет расстановок удовледтворяющих решению, возврат на более высокий уровень." << endl << endl;

system("pause");

system("chcp 866");

}

}

}

}

}

}

void desk\_print(int n, int curs)

{

cout << " A B C D E F G H" << endl;

cout << " \xDA";

for (int i = 0; i < 7; ++i)

{

cout << "\xC4\xC4\xC4" << '\xC2';

}

cout << "\xC4\xC4\xC4\xBF" << endl;

for (int i = 0; i < 7; ++i)

{

cout << 8 - i << "\xB3";

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

{

//cout << ' ' << desk[i + n \* 8 + 2][j] << ' ';

switch (desk[i + n \* 8 + 2][j])

{

case 0:

if ((i == n + 1) && (j == curs)) cout << "[ ]";

else cout << " ";

break;

case 1:

if ((i == n + 1) && (j == curs)) cout << "[\x78]";

else cout << " \x78 ";

break;

case 8:

if ((i == n + 1) && (j == curs)) cout << "[\x51]";

else cout << " \x51 ";

break;

}

cout << "\xB3";

}

cout << endl;

cout << " \xC3";

for (int j = 0; j < 7; ++j)

{

cout << "\xC4\xC4\xC4\xC5";

}

cout << "\xC4\xC4\xC4\xB4" << endl;

}

cout << 1 << "\xB3";

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

{

//cout << ' ' << desk[7 + n \* 8 + 2][j] << ' ';

switch (desk[7 + n \* 8 + 2][j])

{

case 0:

if ((n == 6) && (j == curs)) cout << "[ ]";

else cout << " ";

break;

case 1:

if ((n == 6) && (j == curs)) cout << "[\x78]";

else cout << " \x78 ";

break;

case 8:

if ((n == 6) && (j == curs)) cout << "[\x51]";

else cout << " \x51 ";

break;

}

cout << "\xB3";

}

cout << endl;

cout << " \xC0";

for (int j = 0; j < 7; ++j)

{

cout << "\xC4\xC4\xC4\xC1";

}

cout << "\xC4\xC4\xC4\xD9" << endl;

cout << endl;

}

void desk\_place(int n)

{

int x = desk[1][n];

if (n != 0)

{

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

{

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

{

desk[i + n \* 8 + 2][j] = desk[i + (n - 1) \* 8 + 2][j];

}

}

}

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

{

desk[n + n \* 8 + 2][i] = 1;

}

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

{

desk[i + n \* 8 + 2][x] = 1;

}

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

{

if (((x + n - i) > 0) && ((x + n - i) < 8))

{

desk[(x + n - i) + n \* 8 + 2][i] = 1;

}

if (((n - x + i) > 0) && ((n - x + i) < 8))

{

desk[(n - x + i) + n \* 8 + 2][i] = 1;

}

}

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

{

desk[i + n \* 8 + 2][desk[1][i]] = 8;

}

}

void desk\_print\_all()

{

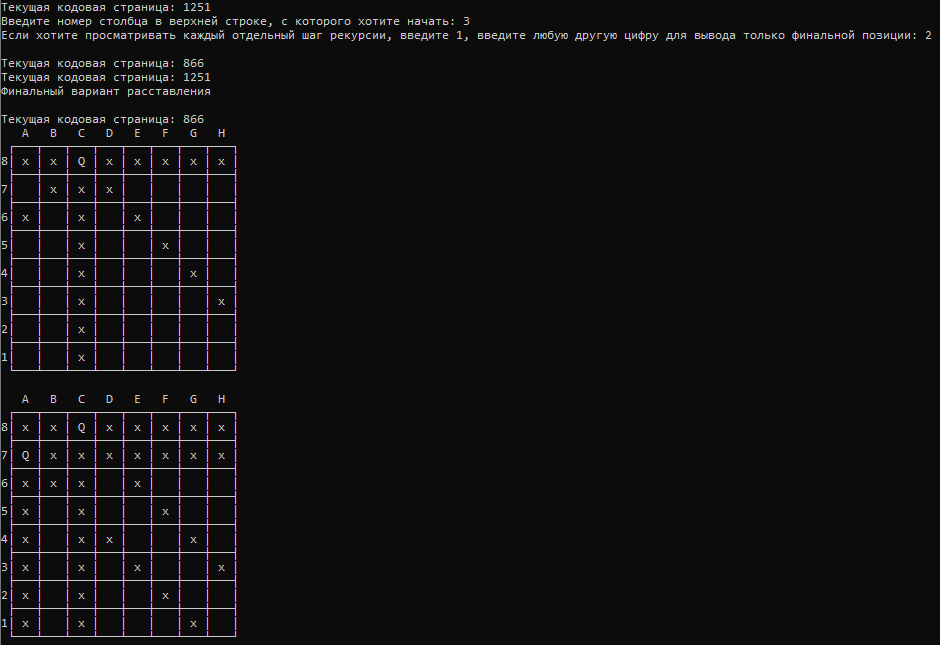
for (int n = 0; n < 8; ++n) {

desk\_print(n, - 1);

}

}

**Решение**



**Выводы**

Данная задача позволяет изучить возможности решения задач, которые могут решаться методами схожими с перебором. Кроме того, часть работы, связанная с визуализацией, позволила изучить некоторые особенности работы с ASCII таблицами.

**Github**

<https://github.com/Hitikov/Lab_Queens>

