

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

|  |  |
| --- | --- |
| Факультет | Информатика и вычислительная техника |
|  | (наименование факультета) |
| Кафедра | Кибербезопасность информационных систем |
|  | (наименование кафедры) |

**ОТЧЕТ по практической работе**

**по дисциплине “Методы программирования”**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Автор |  | | |  | Бобро М.Д. | | | |
|  | (подпись, дата) | | |  |  | | | |
| Обозначение | 10.05.01.550000.000 О | | Группа | | | | ВКБ31 |
| Направление подготовки | | 10.05.01 Компьютерная безопасность | | | | | |
| Профиль | Компьютерная безопасность | | | | | | |
| Преподаватель |  | | |  | | Савельев В.А. | |
|  | (подпись, дата) | | |  | |  | |

г. Ростов-на-Дону

2021 год

**Лабораторная работа №4**

На стандартной шахматной доске (8×88×8) живут 2 шахматных коня: Красный и Зелёный. Обычно они беззаботно скачут по просторам доски, пощипывая шахматную травку, но сегодня особенный день: у Зелёного коня день рождения. Зелёный конь решил отпраздновать это событие вместе с Красным. Но для осуществления этого прекрасного плана им нужно оказаться на одной клетке. Заметим, что Красный и Зёленый шахматные кони сильно отличаются от черного с белым: они ходят не по очереди, а одновременно, и, если оказываются на одной клетке, никто никого не съедает. Сколько ходов им потребуется, чтобы насладиться праздником?

Входные данные

Во входном файле содержатся координаты коней, записанные по стандартным шахматным правилам (т. е. двумя символами — маленькая латинская буква (от a до h) и цифра (от 1 до 8), задающие столбец и строку соответственно).

Выходные данные

Выходной файл должен содержать наименьшее необходимое количество ходов, либо −1−1, если кони не могут встретиться.

**Примеры**

|  |  |
| --- | --- |
| входные данные  a1 a3 | выходные данные  1 |

#include<fstream>

#include<queue>

#include <vector>

#include <string>

using namespace std;

int main() {

int x1, y1, x2, y2;

string a1;

queue <pair<int, int>> q; // будем хранить координаты исследуемой клетки

ifstream in("input.txt");

ofstream out("output.txt"); out.clear();

in >> a1; // координаты первого коня

x1 = a1[0] - 'a'; // пересчитываем в числа

y1 = a1[1] - '1';

in >> a1; // координаты второго коня

x2 = a1[0] - 'a'; // пересчитываем в числа

y2 = a1[1] - '1';

vector <vector <int>> a(8, vector <int>(8, 1000000)); // доска первого коня

vector <vector <int>> c(8, vector <int>(8, 1000000)); // доска второго коня

// заполним доску первого коня

a[x1][y1] = 0; // Начнем, помолясь. Путь 1-й ячейки равен 0

q.push(make\_pair(x1, y1)); // Ибо конь еще не ходил. Индекс в очередь!

while (!q.empty()) { // И пока очередь не опустеет и небеса не падут...

pair <int, int> b;

b = q.front(); // берем из очереди координаты ячейки

q.pop(); // и стираем верхушку очереди

// всего есть 8 вариантов хода конем

// в каждую подходящую клетку заносим длину пути

// и координаты отправляем в очередь

if (b.first < 6 && b.second < 7 && // 1

a[b.first + 2][b.second + 1] == 1000000) {

a[b.first + 2][b.second + 1] = a[b.first][b.second] + 1;

q.push(make\_pair(b.first + 2, b.second + 1));

}

if (b.first < 7 && b.second < 6 && // 2

a[b.first + 1][b.second + 2] == 1000000) {

a[b.first + 1][b.second + 2] = a[b.first][b.second] + 1;

q.push(make\_pair(b.first + 1, b.second + 2));

}

if (b.first < 6 && b.second > 0 && // 3

a[b.first + 2][b.second - 1] == 1000000) {

a[b.first + 2][b.second - 1] = a[b.first][b.second] + 1;

q.push(make\_pair(b.first + 2, b.second - 1));

}

if (b.first > 0 && b.second < 6 && // 4

a[b.first - 1][b.second + 2] == 1000000) {

a[b.first - 1][b.second + 2] = a[b.first][b.second] + 1;

q.push(make\_pair(b.first - 1, b.second + 2));

}

if (b.first > 1 && b.second < 7 && // 5

a[b.first - 2][b.second + 1] == 1000000) {

a[b.first - 2][b.second + 1] = a[b.first][b.second] + 1;

q.push(make\_pair(b.first - 2, b.second + 1));

}

if (b.first < 7 && b.second > 1 && // 6

a[b.first + 1][b.second - 2] == 1000000) {

a[b.first + 1][b.second - 2] = a[b.first][b.second] + 1;

q.push(make\_pair(b.first + 1, b.second - 2));

}

if (b.first > 0 && b.second > 1 && // 7

a[b.first - 1][b.second - 2] == 1000000) {

a[b.first - 1][b.second - 2] = a[b.first][b.second] + 1;

q.push(make\_pair(b.first - 1, b.second - 2));

}

if (b.first > 1 && b.second > 0 && // 8

a[b.first - 2][b.second - 1] == 1000000) {

a[b.first - 2][b.second - 1] = a[b.first][b.second] + 1;

q.push(make\_pair(b.first - 2, b.second - 1));

}

}

// заполним доску второго коня

c[x2][y2] = 0; // Начнем, помолясь. Путь 1-й ячейки равен 0

q.push(make\_pair(x2, y2)); // Ибо конь еще не ходил. Индекс в очередь!

while (!q.empty()) { // И пока очередь не опустеет и небеса не падут...

pair <int, int> b;

b = q.front(); // берем из очереди координаты ячейки

q.pop(); // и стираем верхушку очереди

// всего есть 8 вариантов хода конем

// Но сначала проверим, не добрались ли мы до финиша

if (a[b.first][b.second] == c[b.first][b.second]) { // кони встретились

out << a[b.first][b.second];

return 0;

}

// в каждую подходящую клетку заносим длину пути

// и координаты отправляем в очередь

if (b.first < 6 && b.second < 7 && // 1

c[b.first + 2][b.second + 1] == 1000000) {

c[b.first + 2][b.second + 1] = c[b.first][b.second] + 1;

q.push(make\_pair(b.first + 2, b.second + 1));

}

if (b.first < 7 && b.second < 6 && // 2

c[b.first + 1][b.second + 2] == 1000000) {

c[b.first + 1][b.second + 2] = c[b.first][b.second] + 1;

q.push(make\_pair(b.first + 1, b.second + 2));

}

if (b.first < 6 && b.second > 0 && // 3

c[b.first + 2][b.second - 1] == 1000000) {

c[b.first + 2][b.second - 1] = c[b.first][b.second] + 1;

q.push(make\_pair(b.first + 2, b.second - 1));

}

if (b.first > 0 && b.second < 6 && // 4

c[b.first - 1][b.second + 2] == 1000000) {

c[b.first - 1][b.second + 2] = c[b.first][b.second] + 1;

q.push(make\_pair(b.first - 1, b.second + 2));

}

if (b.first > 1 && b.second < 7 && // 5

c[b.first - 2][b.second + 1] == 1000000) {

c[b.first - 2][b.second + 1] = c[b.first][b.second] + 1;

q.push(make\_pair(b.first - 2, b.second + 1));

}

if (b.first < 7 && b.second > 1 && // 6

c[b.first + 1][b.second - 2] == 1000000) {

c[b.first + 1][b.second - 2] = c[b.first][b.second] + 1;

q.push(make\_pair(b.first + 1, b.second - 2));

}

if (b.first > 0 && b.second > 1 && // 7

c[b.first - 1][b.second - 2] == 1000000) {

c[b.first - 1][b.second - 2] = c[b.first][b.second] + 1;

q.push(make\_pair(b.first - 1, b.second - 2));

}

if (b.first > 1 && b.second > 0 && // 8

c[b.first - 2][b.second - 1] == 1000000) {

c[b.first - 2][b.second - 1] = c[b.first][b.second] + 1;

q.push(make\_pair(b.first - 2, b.second - 1));

}

}

out << -1;

return 0;

}

