МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Лабораторная работа №4 Рекурсия и головоломки

ОТЧЁТ

По лабораторной работе

по дисциплине

Технологии программирования

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Капранов С.Н.

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Редькина О.С.

(подпись) (фамилия, и.,о.)

18-ИСТ-2

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

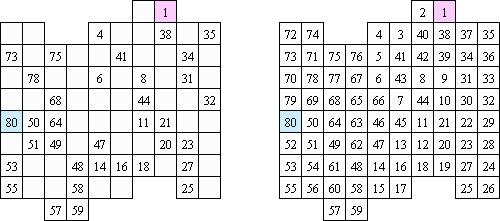
С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2020

# Задача 16. Хидато

**Хидато**(*"Hidato"*; в переводе с идиш:*"моя головоломка"*; еще одно название-*"Hidoku"*)была изобретена израильским математиком ГиоромБенедеком (GyoraBenedek). Головоломка представляет собой поле произвольной формы (чаще всего прямоугольное или квадратное), состоящее из клеток. Необходимо заполнить все клетки последовательными числами, которые соединены горизонтально, вертикально или по диагонали. В каждой головоломке уже присутствуют наименьшее и наибольшее числа. Также на поле могут стоять и другие числа, чтобы облегчить игроку процесс разгадывания и обеспечить единственность решения задачи.

* России головоломку еще называют *"Путь короля"*, так как движение от одного числа к другому напоминает перемещение шахматного короля по доске.



# Листинг программы:

#include <iostream>

#include <sstream>

#include <iterator>

#include <vector>

using namespace std;

struct node

{

int val;

unsigned char neighbors;

};

class hSolver //определение класса для решения задачи

{

public:

hSolver() //задание конструктора класса по умолчанию

{

dx[0] = -1; dx[1] = 0; dx[2] = 1; dx[3] = -1; dx[4] = 1; dx[5] = -1; dx[6] = 0; dx[7] = 1;

dy[0] = -1; dy[1] = -1; dy[2] = -1; dy[3] = 0; dy[4] = 0; dy[5] = 1; dy[6] = 1; dy[7] = 1;

}

void solve(vector<string>& puzz, int max\_wid) //описание функции поиска решения

{

if (puzz.size() < 1)

return; //прерывание работы программы в случае, когда в строке задано менее 1 элемента

wid = max\_wid;

hei = static\_cast<int>(puzz.size()) / wid;

int len = wid \* hei, c = 0;

max = 0; // вычисление максимальной длины результирующей строки

arr = new node[len];

memset(arr, 0, len \* sizeof(node)); // определение массива структур для хранения результатов

weHave = new bool[len + 1];

memset(weHave, 0, len + 1); // определение массива бувевых значений для определения возможности хода

for (vector<string>::iterator i = puzz.begin(); i != puzz.end(); i++)

{ // алгоритм поиска следующего хода

if ((\*i) == "\*") { arr[c++].val = -1; continue; } // пропуск прохода цикла в случае окончания игрового поля

arr[c].val = atoi((\*i).c\_str());//преобразование строки в число

if (arr[c].val > 0) weHave[arr[c].val] = true; //ход возможен

if (max < arr[c].val) max = arr[c].val;

c++;

}

solveIt(); c = 0; // вызов функции для хода

for (vector<string>::iterator i = puzz.begin(); i != puzz.end(); i++)

{

if ((\*i) == ".")

{

// запись найденного значения в строку

ostringstream o; o << arr[c].val;

(\*i) = o.str();

}

c++;

}

delete[] arr;

delete[] weHave;

}

private:

bool search(int x, int y, int w) // функция поиска

{

if (w == max) return true; // найден максимум, прекращение поиска

node\* n = &arr[x + y \* wid];

n->neighbors = getNeighbors(x, y); // взятие соседних квадратов поля

if (weHave[w])

{

// поиск возможности хода относительно выбранных квадратов поля

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

{

if (n->neighbors & (1 << d))

{

int a = x + dx[d], b = y + dy[d];

// вычисление предполагаемого хода и сравнение его с ранее найденным значением

if (arr[a + b \* wid].val == w)

if (search(a, b, w + 1)) return true;

}

}

return false;

}

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

{

if (n->neighbors & (1 << d))

{

int a = x + dx[d], b = y + dy[d];

// вычисление предполагаемого хода и его реализация

if (arr[a + b \* wid].val == 0)

{

arr[a + b \* wid].val = w;

if (search(a, b, w + 1)) return true;

arr[a + b \* wid].val = 0;

}

}

}

return false;

}

unsigned char getNeighbors(int x, int y)

{ // описание функции взятия двух соседних квадратов игрового поля

unsigned char c = 0; int m = -1, a, b;

for (int yy = -1; yy < 2; yy++)

for (int xx = -1; xx < 2; xx++)

{

if (!yy && !xx) continue;

m++; a = x + xx, b = y + yy;

if (a < 0 || b < 0 || a >= wid || b >= hei) continue;

if (arr[a + b \* wid].val > -1) c |= (1 << m);

}

return c;

}

void solveIt() // описание функции для возможности хода

{

int x, y; findStart(x, y); // ход возможен, вызов функции поиска хода

if (x < 0) { cout << "\nCan't find start point!\n"; return; } //ход невозможен

search(x, y, 2);

}

void findStart(int& x, int& y) // описание функции поиска хода

{

for (int b = 0; b < hei; b++)

for (int a = 0; a < wid; a++)

if (arr[a + wid \* b].val == 1) { x = a; y = b; return; }

x = y = -1;

}

int wid, hei, max, dx[8], dy[8];

node\* arr;

bool\* weHave;

};

int main(int argc, char\* argv[])

{

setlocale(LC\_ALL, "Russian");

int wid; //переменная для хранения ширины поля

string p = ". 33 35 . . \* \* \* . . 24 22 . \* \* \* . . . 21 . . \* \* . 26 . 13 40 11 \* \* 27 . . . 9 . 1 \* \* \* . . 18 . . \* \* \* \* \* . 7 . . \* \* \* \* \* \* 5 .";

wid = 8; //задание начального поля

istringstream iss(p);

vector<string> puzz; //считывание начальной строки

int c = 0;

copy(istream\_iterator<string>(iss), istream\_iterator<string>(), back\_inserter<vector<string> >(puzz));

cout << "Изначальная таблица" << endl;

for (vector<string>::iterator i = puzz.begin(); i != puzz.end(); i++)

{

if ((\*i) != "\*" && (\*i) != ".")

{

if (atoi((\*i).c\_str()) < 10) cout << "0";

cout << (\*i) << " ";

}

else cout << "- ";

if (++c >= wid)

{

cout << endl; c = 0;

}

}

cout << endl << endl;

hSolver s;

s.solve(puzz, wid); //создание объекта класса и вызов функции решения

cout << "Заполненая таблица" << endl; // вывод результата

for (vector<string>::iterator i = puzz.begin(); i != puzz.end(); i++)

{

if ((\*i) != "\*" && (\*i) != ".")

{

if (atoi((\*i).c\_str()) < 10) cout << "0";

cout << (\*i) << " ";

}

else cout << "- ";

if (++c >= wid)

{

cout << endl; c = 0;

}

}

cout << endl << endl;

return system("pause");

}

# Результат работы:

