МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра Информационная безопасность вычислительных систем и сетей

Рекурсия и головоломки
(наименование работы)

ОТЧЕТ

по лабораторной работе №4

по дисциплине

Технологии программирования
(наименование дисциплины)

РУКОВОДИТЕЛЬ:

_____ Капранов С.Н.
(подпись) (фамилия, и., о.)

СТУДЕНТ:

_____ Савельев М.А.
(подпись) (фамилия, и., о.)

18-ИСТ-4
(шифр группы)

Работа защищена «___» _____

С оценкой _____

Нижний Новгород, 2020

Содержание

Задача

16 вариант: Реализовать игру «Хидоку» – ひどく (или «Хидато» – ひだと).

Основная часть отчета

Программа написана на языке C++ в среде разработки VisualStudio 2017.

Листинг программы

```
#include <iostream>
#include <sstream>
#include <iterator>
#include <vector>

using namespace std;

struct node
{
	int val; // Значение текущей клетки
	unsigned char neighbors; // "Соседи" текущей клетки
};

class hSolver // Класс, в котором происходит расчет ("решение")
{
public:
	//--- Точки вокруг текущей точки ---\\
	hSolver()
	{
		dx[0] = -1;  dy[0] = -1;
		dx[1] =  0;  dy[1] = -1;
		dx[2] =  1;  dy[2] = -1;
		dx[3] = -1;  dy[3] =  0;                                //(-1, -1)|(0, -
1)|(1, -1)|
		dx[4] =  1;  dy[4] =  0;                                //(-1,  0)|(0,
0)|(1,  0)|  Конструкция для проверки точек вокруг текущей на проверку нужной
		dx[5] = -1;  dy[5] =  1;                                //(-1,  1)|(0,
1)|(1,  1)|
		dx[6] =  0;  dy[6] =  1;
		dx[7] =  1;  dy[7] =  1;
	}
	//--- ---\\

	int width, height, max, dx[8], dy[8];
	node* arr;
	bool* weHave;
	void solve(vector<string> & puzz, int max_wid) // Описание функции поиска решения
	{
		width = max_wid; // Присваивание значения полученной переменной
		height = static_cast<int>(puzz.size()) / width; // нахождение высоты
		int length = width * height; // Длина поля
		max = 0; // вычисление максимальной длины результирующей строки

		arr = new node[length]; // Массив структур
		memset(arr, 0, length * sizeof(node)); // определение массива структур для
хранения результатов

		weHave = new bool[length + 1]; // Булевый массив
		memset(weHave, 0, length + 1); // Определение булевого массива для
определения возможности хода

		//--- Алгоритм поиска следующего хода ---\\
		int c = 0;
		for (vector<string>::iterator i = puzz.begin(); i != puzz.end(); i++)
		{
			if ((*i) == "*")
			{
				arr[c++].val = -1;
				continue;
			} // Пропуск прохода цикла, если появилась звездочка
```

```
                        arr[c].val = atoi((*i).c_str()); // Преобразовать string в int (atoi)
("1" в 1)

                        if (arr[c].val > 0)
                                weHave[arr[c].val] = true; // Ход возможен
                        if (max < arr[c].val)
                                max = arr[c].val; // Нахождение максимума
                        c++;
                }
                //--- ---\\

                solveIt();
                c = 0;                                  // вызов функции для хода
                for (vector<string>::iterator i = puzz.begin(); i != puzz.end(); i++)
                {
                        if ((*i) == ".")
                        { // Вывод значения
                                ostringstream o; // Потоковый вывод данных
                                o << arr[c].val;
                                (*i) = o.str();
                        }
                        c++;
                }
                //--- Освобождение памяти ---\\
                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 * width];
                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]; // Присваивание
переменной значений "соседей"
                                        if (arr[a + b * width].val == w) // Вычисление
предполагаемого хода и сравнение его с ранее найденным значением
                                                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 * width].val == 0)
                                {
                                        arr[a + b * width].val = w;
                                        if (search(a, b, w + 1))
                                                return true;
```

```
                                arr[a + b * width].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 >= width || b >= height)
                                        continue;
                                if (arr[a + b * width].val > -1)
                                        c |= (1 << m); // Побитовый сдвиг влево: равносильно --
-> (1 * Math.pow(2,m))
                        }
                return c;
        }

        void solveIt() // Описание функции для возможности хода (покажет, если игровое
поле построено неправильно)
        {
                int x, y; // Координаты
                findStart(x, y); // Функция выхода первого хода, возвращает координаты "1"
                if (x < 0)
                {
                        cout << "\nCan't find start point!\n";
                        exit(0);
                } // Ход невозможен
                search(x, y, 2); // Функция поиска
        }

        void findStart(int& x, int& y) // Поиск единицы, чтобы начать
        {
                for (int b = 0; b < height; b++)
                        for (int a = 0; a < width; a++)
                                if (arr[a + width * b].val == 1)
                                {
                                        x = a;
                                        y = b;
                                        return;
                                }
                x = y = -1;
        }
};

int main(int argc, char* argv[])
{
        int size; // Размер поля (size*size)
        int difficulty; // Сложность поля (два варианта)
        string width; // Передается в обработку (выбранный размер поля)

        //--- Игровые поля разных ширин ---\\
        string w_1_e = "1 . 3 . 5";
        string w_1_h = "1 . . . . . . . . 10";
        string w_2_e = "4 . . . . 1 . . 10 . ";
        string w_2_h = "1 . 4 . . * 6 * 12 . . . . .";
```

```
	string w_3_e = "1 . . . . . . . 9";
	string w_3_h = ". . 10 14 . . . 1 . . 7 . . . . 18 4 .";
	string w_4_e = ". . 9 . 5 . 16 . . . . . 1 2 13 14";
	string w_4_h = ". . . 7 . 10 . . 14 . . 5 . . . 20 16 3 . . . * * . 1 * * 24";
	string w_5_e = "1 . 3 . 5";
	string w_5_h = ". . . 11 * . 6 . . . . 15 * * * 4 . 16 * * . * 1 * *";
	string w_6_e = "16 . 18 * * * 14 . 1 . . 4 . . . 6 . * . 9 . . * * ";
	string w_6_h = "5 . 13 . . 16 * . 7 . . * 3 . 10 . 22 18 * . . . . * * * 1 20 *
*";
	string w_7_e = "1 13 3 11 5 9 7 14 . . . . . 8";
	string w_7_h = "20 22 . 24 . 3 . . . 28 1 . 6 . . 14 18 . 26 . . . . . 12 . 8 .";
	string w_8_e = ". 33 35 . . * * * . . 24 22 . * * * . . . 21 . . * * . 26 . 13 40
11 * *      27 . . . 9 . 1 * * * . . 18 . . * * * * * . 7 . . * * * * * * 5 .";
	string w_8_h = "1 . * * * * * * * 3 . 5 . 7 * * * * . 11 . . 8 * * * 13 . 15 . 17 *
* * . 21 . 19 . * * * * 23 . . . 33 * * * * * * 27 . . * * * * * . . 31 ";
	string w_9_e = "* . 8 10 . . . . 17 * * . . . 12 * * * * * * 5 . * * * * * * * 1 .
3 * * *";
	string w_9_h = "20 1 . 17 . 15 . . 11 * . . 3 . 5 14 . 8 * * * * * * . 7 .";
	string w_10_e = " 1 . . . . 6 . . . 10";
	string w_10_h = ". . 8 12 . . 15 21 . . . 1 . . . 23 . . . 18 4 . 40 . 10 . . . .
32 . . 42 . 38 . 35 25 . . 50 . 47 * * * * * . 29 44 . . * * * * * 28 . ";
	//--- ---\\

	bool flag = true;
	int check;
	while (flag) { // Весь цикл программы

		//--- Обработка введенной размерности ---\\
		cout << "Enter field's size (1-10)\n";
		while (!(cin >> size) || size < 1 || size > 10)
		{
			cout << "Uncorrect size, try again \n";
			cout << "Enter field's size (1-10)\n";
			cin.clear();
			cin.ignore(numeric_limits<streamsize>::max(), '\n');
		}
		//--- ---\\

		//--- Выбор сложности уровня ---\\
		cout << "Enter difficulty (1 - easy, 2 - hard)\n";
		while (!(cin >> difficulty) || difficulty < 1 || difficulty > 2)
		{
			cout << "Uncorrect difficulty, try again \n";
			cout << "Enter difficulty (1 - easy, 2 - hard)\n";
			cin.clear();
			cin.ignore(numeric_limits<streamsize>::max(), '\n');
		}
		//--- ---\\

		//--- Задание параметров поля ---\\
		if (size == 1) { if (difficulty == 1) width = w_1_e;  else width = w_1_h; }
		if (size == 2) { if (difficulty == 1) width = w_2_e;  else width = w_2_h; }
		if (size == 3) { if (difficulty == 1) width = w_3_e;  else width = w_3_h; }
		if (size == 4) { if (difficulty == 1) width = w_4_e;  else width = w_4_h; }
		if (size == 5) { if (difficulty == 1) width = w_5_e;  else width = w_5_h; }
		if (size == 6) { if (difficulty == 1) width = w_6_e;  else width = w_6_h; }
		if (size == 7) { if (difficulty == 1) width = w_7_e;  else width = w_7_h; }
		if (size == 8) { if (difficulty == 1) width = w_8_e;  else width = w_8_h; }
		if (size == 9) { if (difficulty == 1) width = w_9_e;  else width = w_9_h; }
		if (size == 10){ if (difficulty == 1) width = w_10_e; else width = w_10_h;}
		//--- ---\\

		istringstream iss(width); // Потоковый ввод данных
		vector<string> puzz; // Считывание начальной строки
```

```
            copy(istream_iterator<string>(iss), istream_iterator<string>(),
back_inserter<vector<string> >(puzz)); // Ввод элементов, добавляя новые элементы в конце
контейнера
            hSolver s; // Экземпляр класса
            s.solve(puzz, size); // Создание объекта класса и вызов функции решения

            //--- Вывод результата ---\\
            int c = 0;
            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 >= size)
                  {
                        cout << endl; c = 0;
                  }
            }
            cout << endl << endl;
            //--- ---\\

            //--- Возможность повторной игры ---\\
            cout << "Continue? (1 - Yes, 0 - No)";
            while (!(cin >> check) || check < 0 || check > 1)
            {
                  cout << "Uncorrect choice, try again \n";
                  cout << "Continue? (1 - Yes, 0 - No) \n";
                  cin.clear();
                  cin.ignore(numeric_limits<streamsize>::max(), '\n');
            }
            if (check == 0) flag = false;
            else flag = true;
            //--- ---\\
      }
      return system("pause");
}
```

Входные и выходные данные

```
Enter field's size (1-10)
error
Uncorrect size, try again
Enter field's size (1-10)
-8
Uncorrect size, try again
Enter field's size (1-10)
6
Enter difficulty (1 - easy, 2 - hard)
error
Uncorrect difficulty, try again
Enter difficulty (1 - easy, 2 - hard)
-7
Uncorrect difficulty, try again
Enter difficulty (1 - easy, 2 - hard)
2
```

Рис. 1. Входные значения (1)

```
05 06 13 14 15 16
   04 07 12 17
03 08 10 11 22 18
   02 09 21 19
      01 20
```

Рис. 2. Выходные значения (1)

```
Continue? (1 - Yes, 0 - No)
yes
Uncorrect choice, try again
Continue? (1 - Yes, 0 - No)
1
Enter field's size (1-10)
8
Enter difficulty (1 - easy, 2 - hard)
1
```

Рис. 3. Входные значения (2)

```
32 33 35 36 37
31 34 24 22 38
30 25 23 21 12 39
29 26 20 13 40 11
27 28 14 19 09 10 01
      15 16 18 08 02
            17 07 06 03
                  05 04
```

Рис. 4. Выходные значения (2)