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»

ОТЧЕТ по лабораторной работе

по дисциплине

Технологии программирования

Вариант 4

РУКОВОДИТЕЛЬ:

_____ Капранов С.Н.

(подпись) (фамилия,и.,о.)

СТУДЕНТ:

_____ Гудима И.А.

(подпись) (фамилия,и.,о.)

18-ИСТ-2

(шифр группы)

Работа защищена«____»_____

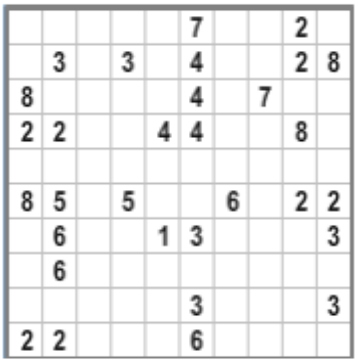
С оценкой _____

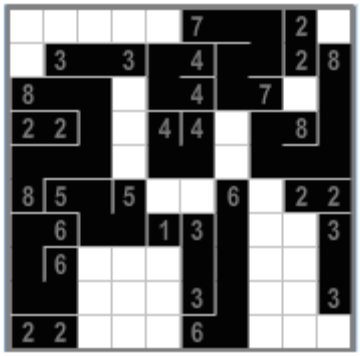
Нижний Новгород

2020

Задание:

Филиппинский кроссворд (*"Link-a-Pix", "Paint by Pairs"*) - это головоломка с числами. Все числа, расположенные в сетке, кроме единицы, имеют свою пару. Необходимо найти каждую пару чисел и соединить их линиями. Количество клеток в ней должно равняться числам на ее концах. Линии, соединяющие пары, могут преломляться и идти в горизонтальном или вертикальном направлениях (но не по диагоналям). Линии не могут пересекаться друг с другом или проходить через одни и те же клетки.





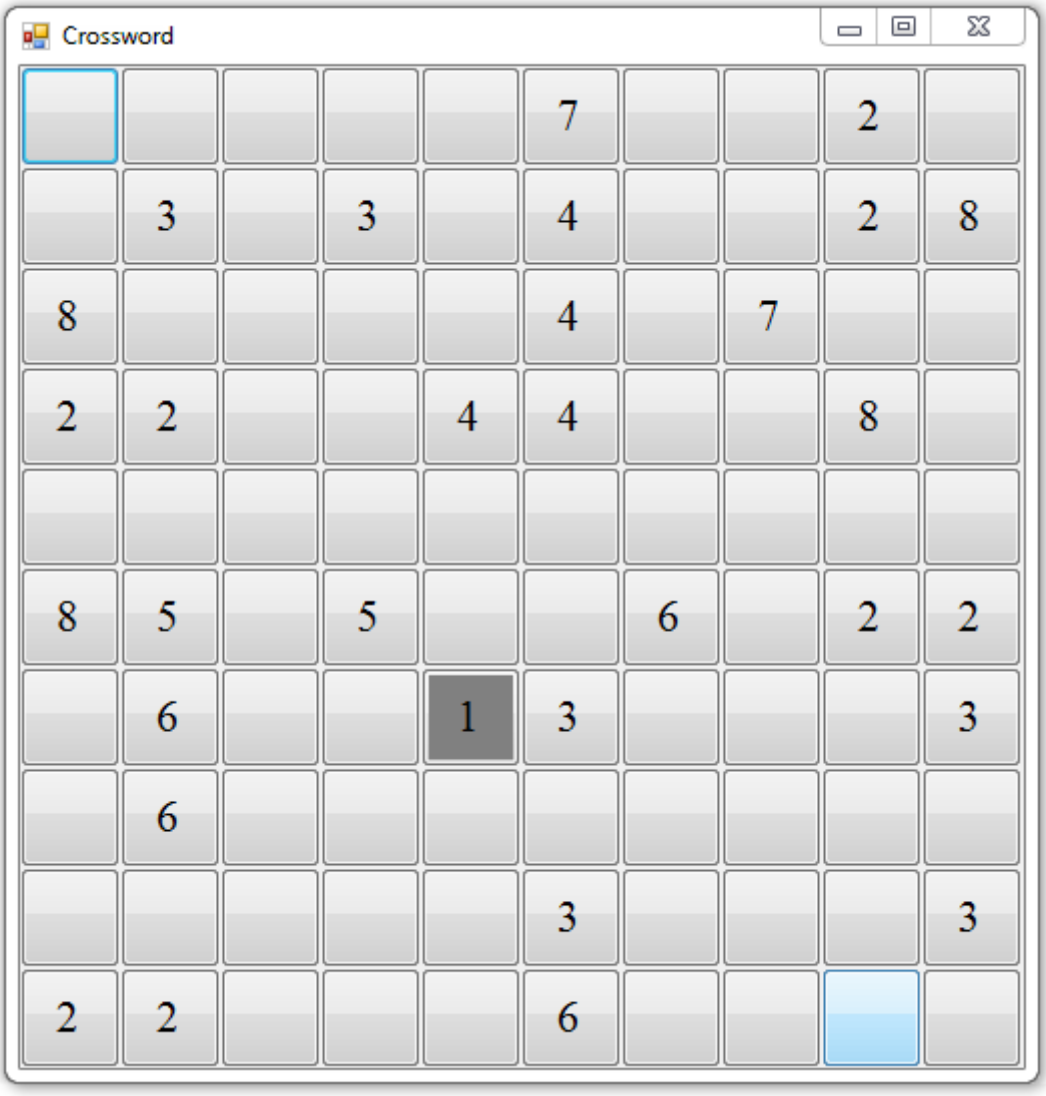
Листинг:

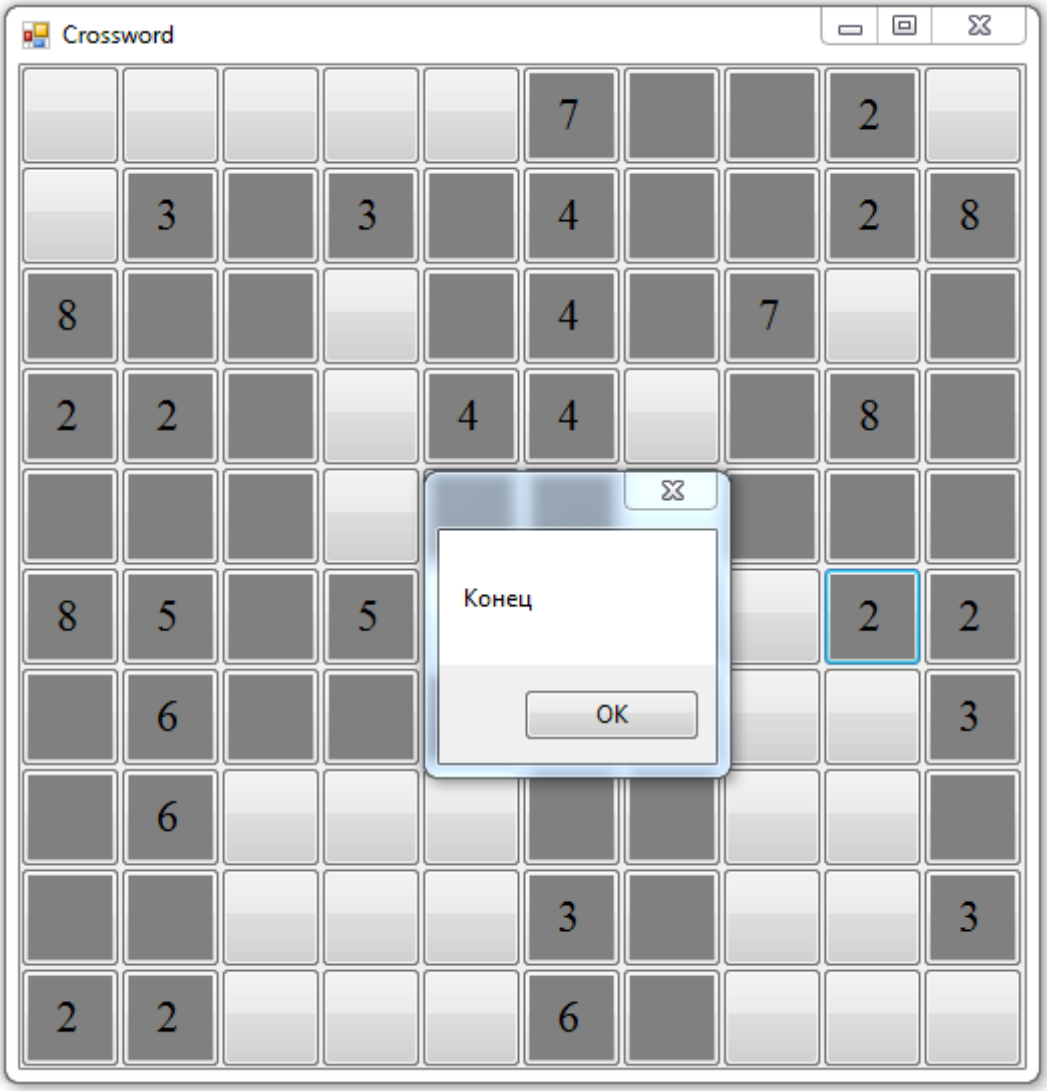
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace qwe.Crossword
{
    public partial class Form1 : Form
    {
        const int mapSize = 10; // колво клеток

        const int cellSize = 50; // размер клетки

        int[,] map = new int[mapSize, mapSize]; //

        List<string> way = new List<string>(); //Путь

        List<List<string>> listOfWay = new List<List<string>>(); // Список путей

        bool btn = false; // свичер

        public Form1()
        {
            InitializeComponent();
            Text = "Crossword";
            Init();
        }

        public void Init() // инициализация игры
        {
            map = new int[mapSize, mapSize]
```

```
            {
                {0,0,0,0,0,7,0,0,2,0},
                {0,3,0,3,0,4,0,0,2,8},
                {8,0,0,0,0,4,0,7,0,0},
                {2,2,0,0,4,4,0,0,8,0},
                {0,0,0,0,0,0,0,0,0,0},
                {8,5,0,5,0,0,6,0,2,2},
                {0,6,0,0,1,3,0,0,0,3},
                {0,6,0,0,0,0,0,0,0,0},
                {0,0,0,0,0,3,0,0,0,3},
                {2,2,0,0,0,6,0,0,0,0}
            };
            CreateMap();
        }
        public void CreateMap()
        {
            Width = mapSize * cellSize + 17;
            Height = mapSize * cellSize + 38;
            for (int i = 0; i < mapSize; i++)
            {
                for (int j = 0; j < mapSize; j++)
                {
                    Button button = new Button();
                    button.Location = new Point(j * cellSize, i * cellSize);
                    button.Size = new Size(cellSize, cellSize);
                    button.Font = new Font ("Times New Roman", 16f);
                    button.BackColor = Color.Transparent;
                    //button.Tag = new Tuple<int, int>(i, j); //Связывание номера ij map с
кнопкой
                    button.Name = $"button{i},{j}"; // Задаёт имя кнопкам типа: buttoni,j
                    Controls.Add(button);
                    if (map[i, j] != 0)
                        button.Text = Convert.ToString(map[i, j]);
                    if (map[i, j] == 1)
                        button.BackColor = Color.Gray;
                    else
                    {
                        if (button.Text != "" && button.Text != "1")
                        {
                            button.Click += Button_Click;
                        }
                        else button.BackColor = Color.Transparent;
                        button.MouseEnter += Button_MouseEnter;
                    }
                }
            }
        }

        public List<string> DefaultColorCastingFunction(List<string>a)
        {
            for (int i = 0; i < a.Count; i++)
            {
                Controls[$"{a[i]}"].BackColor = Color.Transparent;
            }
            return a;
        }

        private bool EndGame(List<List<string>> a)
        {
            int p = 0;
            List<string> b = new List<string>();
            for (int i = 0; i < listOfWay.Count; i++)
            {
                b = listOfWay[i];
                for (int j = 0; j < b.Count; j++)
                {
                    if (b[j] != "")
                    {
```

```
                        p++;
                    }
                }
            }
            if (p == 65)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        private void Button_Click(object sender, EventArgs e)
        {
            if ((sender as Button).BackColor == Color.Transparent)
            {
                (sender as Button).BackColor = Color.Gray;
                way.Add((sender as Button).Name);
                if (btn == true)
            {
                    btn = false;
                    if ((Controls[$"{way[0]}"] as Button).Text ==
(Controls[$"{way[way.Count-1]}"] as Button).Text && Convert.ToInt32((Controls[$"{way[0]}"]
as Button).Text) == way.Count)///
                    {
                        listOfWay.Add(way);
                        way = new List<string>();
                    }
                    else
                    {
                        DefaultColorCastingFunction(way);
                        way = new List<string>();
                    }
            }
            else
            {
                    btn = true;
            }
            }
            if (EndGame(listOfWay) == true)
            {
                MessageBox.Show("Конец");
                Close();
            }
        }

        private bool SearchButtonInList(Button a)
        {
            List<string> b = new List<string>(); //создаю список b, который будет являться
обёрткой для списов, которые хронятся в списке listOfWay
            for (int i = 0; i < listOfWay.Count; i++)
            {
                b = listOfWay[i];
                for (int j = 0; j < b.Count; j++)
                {
                    if (a.Name == b[j])
                    {
                        DefaultColorCastingFunction(listOfWay[i]);
                        listOfWay[i] = new List<string>();
                        return true;
                    }
                }
            }
            return false;
        }

        private void Button_MouseEnter(object sender, EventArgs e)
```

```
        {
            if ((sender as Button).Text == "")
            {
                if (btn == true)
                {
                    SearchButtonInList((sender as Button));
                    if (SearchButtonInList((sender as Button)) == true)
                    {
                        way = new List<string>();
                    }
                    if ((sender as Button).BackColor == Color.Transparent)
                    {
                        (sender as Button).BackColor = Color.Gray;
                    }
                    else (sender as Button).BackColor = Color.Transparent;
                    way.Add((sender as Button).Name);
                }
                else
                {
                    if ((sender as Button).BackColor == Color.Gray)
                    {
                        (sender as Button).BackColor = Color.Gray;
                    }
                    else (sender as Button).BackColor = Color.Transparent;
                }
            }
        }
    }
}
```




Crossword
Конец
OK