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

**ОТЧЕТ**

**по Лабораторной работе №4**

по дисциплине

Технология программирования

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Багиров М.Б.

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Вихрова С.С

(подпись) (фамилия, и.,о.)

18-ИСТ-4

(шифр группы)

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

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

Нижний Новгород   
2020

Оглавление

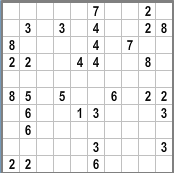
[Вариант №4 3](#_Toc41322033)

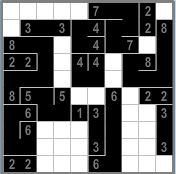
[Листинг программы 4](#_Toc41322034)

[Результат работы программы 12](#_Toc41322035)

# Вариант №4

**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 программы

-- Form2--

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace GameOver

{

public partial class Form2 : Form

{

List<DataGridViewCell> AllCells = new List<DataGridViewCell>(); // все ячейки с числами на данном поле

int countAlgoritm = 0; // количество запусков алгоритма заполнения поля

public Form2()

{

InitializeComponent();

Colums();//вызываем функцию, которая создаёт поле

ReadFile();//вызываем функцию, которая читает данные из файла

CountAllCellls();// считываем все ячейки с числами

//пока количество ячеек с числоми не равно 0 и количество запусков не превышает 5

while (AllCells.Count != 0 && countAlgoritm < 5)

{

Algoritm();//запускаем алгоритм нахождения пар

++countAlgoritm;

}

//елси после 5 попыток остались не задйствованные ячейки, то выводим сооющение

if (AllCells.Count() != 0 && countAlgoritm >= 5) {

MessageBox.Show("Игру нельзя пройти!!!");

}

}

//функция для чтения из файла

public void ReadFile()

{

Random random = new Random();

int x = random.Next(1, 4);

string nameFile = "Game" + 3 + ".txt";

FileStream file = new FileStream(nameFile, FileMode.Open, FileAccess.Read);

StreamReader reader = new StreamReader(file);

int k = 0; // переменная для определения строки в поле

string text;

while ((text = reader.ReadLine()) != null)

{ //чтение по строчно

k = k + 1;

string[] splitLine = text.Split(' '); // разделитель

for (int i = 0; i < splitLine.Length; i++)

{

if (splitLine[i] != "0")

{

dataGridView1.Rows[k].Cells[i].Value = splitLine[i];

dataGridView1.Rows[k].Cells[i].Style.BackColor = Color.White;

}

if (splitLine[i] == "1")

{

dataGridView1.Rows[k].Cells[i].Style.BackColor = Color.Black;

dataGridView1.Rows[k].Cells[i].Style.ForeColor = Color.White;

}

}

}

reader.Close();

file.Close();

}

//функция для разбиения поля на ячейки

public void Colums()

{

int k = this.dataGridView1.Size.Width / 30;

System.Windows.Forms.DataGridViewColumn[] f = new System.Windows.Forms.DataGridViewColumn[] { };

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

{

System.Windows.Forms.DataGridViewTextBoxColumn co = new System.Windows.Forms.DataGridViewTextBoxColumn();

co.Width = 30;

this.dataGridView1.Columns.Add(co);

}

k = this.dataGridView1.Size.Height / 30;

for (int i = 0; i < k - 1; i++)

{

this.dataGridView1.Rows.Add();

}

}

//фукнцтя для прохождения по всему полю и запоминание всех ячеек с числами

public void CountAllCellls() {

for (int c0 = 0; c0 < this.dataGridView1.Rows.Count; c0++)

{

for (int c1 = 0; c1 < this.dataGridView1.Columns.Count; c1++)

{

if (dataGridView1[c1, c0].Value != null && dataGridView1[c1, c0].Value != "" && dataGridView1[c1, c0].Value.ToString() != "1") AllCells.Add(dataGridView1[c1, c0]);

}

}

}

//основной алгоритм нахождения пар

public void Algoritm()

{

Console.WriteLine("Количество ячеек без пары " + AllCells.Count());

for (int c0 = 0; c0 < this.dataGridView1.Rows.Count; c0++)

{

for (int c1 = 0; c1 < this.dataGridView1.Columns.Count; c1++)

{

//проверка на не пустую ячейку

if (this.dataGridView1.Rows[c0].Cells[c1].Value != "" && Convert.ToInt32(this.dataGridView1.Rows[c0].Cells[c1].Value) >= 1 && this.dataGridView1.Rows[c0].Cells[c1].Style.BackColor != Color.Black)

{

List<DataGridViewCell> nn = new List<DataGridViewCell>();//массив совпавших ячеек

//проход по всем соседним ячейкам к выбранной ячейки в шахматном порядке

//Например, к ячейке в середине будут проверятся все ячейки с плюсами

// |\_|\_|+|\_|\_|

// |\_|+|\_|+|\_|

// |+|\_|3|\_|+|

// |\_|+|\_|+|\_|

// |\_|\_|+|\_|\_|

for (int i = c0 - (Convert.ToInt32(this.dataGridView1.Rows[c0].Cells[c1].Value) - 1); i < c0 + Convert.ToInt32(this.dataGridView1.Rows[c0].Cells[c1].Value); i++)

{

for (int j = c1 - ((Convert.ToInt32(this.dataGridView1.Rows[c0].Cells[c1].Value) - 1) - Math.Abs(i - c0)); j <= c1 + ((Convert.ToInt32(this.dataGridView1.Rows[c0].Cells[c1].Value) - 1) - Math.Abs(i - c0)); j = j + 2)

{

if (i >= 0 && j >= 0 && i < this.dataGridView1.Rows.Count && j < this.dataGridView1.ColumnCount)

if (this.dataGridView1[j, i].Value != null && this.dataGridView1[j, i] != this.dataGridView1[c1, c0] && this.dataGridView1[j, i].Value.ToString() == this.dataGridView1.Rows[c0].Cells[c1].Value.ToString() && this.dataGridView1.Rows[i].Cells[j].Style.BackColor == Color.White)

{

nn.Add(this.dataGridView1[j, i]);

}

}

}

//проверка на то, что совпадающих с данной ячейкой всего одна ячейка

if (nn.Count == 1)

{

//если можно сразу соединить эти две ячейки

//по вертикали

if (Math.Abs(c0 - nn[nn.Count - 1].RowIndex) == Convert.ToInt32(nn[nn.Count - 1].Value) - 1)

{

AllCells.Remove(dataGridView1[c1, c0]);

for (int h = 0; h < Convert.ToInt32(dataGridView1[c1, c0].Value); h++)

{

dataGridView1[c1, c0 + h].Style.BackColor = Color.Black;

dataGridView1[c1, c0 + h].Style.ForeColor = Color.White;

}

AllCells.Remove(dataGridView1[c1, c0 + Convert.ToInt32(dataGridView1[c1, c0].Value)-1]);

}

//по горизонтали

if (Math.Abs(c1 - nn[nn.Count - 1].ColumnIndex) == Convert.ToInt32(nn[nn.Count - 1].Value) - 1)

{

AllCells.Remove(dataGridView1[c1, c0]);

for (int h = 0; h < Convert.ToInt32(dataGridView1[c1, c0].Value); h++)

{

dataGridView1[c1 + h, c0].Style.BackColor = Color.Black;

dataGridView1[c1 + h, c0].Style.ForeColor = Color.White;

}

AllCells.Remove(dataGridView1[c1 + Convert.ToInt32(dataGridView1[c1, c0].Value) - 1, c0]);

}

//если нельзя соединить сразу, то будут строиться возможные пути соединения двух выбранных ячеек

if (Math.Abs(c0 - nn[nn.Count - 1].RowIndex) != Convert.ToInt32(nn[nn.Count - 1].Value) - 1 && Math.Abs(c1 - nn[nn.Count - 1].ColumnIndex) != Convert.ToInt32(nn[nn.Count - 1].Value) - 1)

{

//Создание объекта класса Tree

Tree parent = new Tree(dataGridView1, this.dataGridView1.Rows[c0].Cells[c1], Convert.ToInt32(this.dataGridView1.Rows[c0].Cells[c1].Value), this.dataGridView1.Rows[c0].Cells[c1].Value.ToString());

parent.CalculatingMoves();//функция вычисления всех возможныъ ходов

Console.WriteLine(" ======== Дети =========");

int countLastNode = 0;//количество всех путей, которые могут образовать пару

Console.WriteLine(" Количество возможных ходов" + countLastNode);

parent.LastNode(ref countLastNode);

Console.WriteLine(" Количество возможных ходов" + countLastNode);

if (countLastNode == 1)

{

List<DataGridViewCell> \_moves = new List<DataGridViewCell>();

List<DataGridViewCell> \_moves2 = new List<DataGridViewCell>();

parent.AllChild(\_moves, \_moves2);//возвращение единственно верного пути

AllCells.Remove(dataGridView1[c1, c0]);

AllCells.Remove(dataGridView1[\_moves2[\_moves2.Count()-1].ColumnIndex, \_moves2[\_moves2.Count() - 1].RowIndex]);

for (int i = 0; i < \_moves2.Count(); i++)

{

Console.WriteLine(" ======== Индексы ячеек " + \_moves2[i].RowIndex + ", " + \_moves2[i].ColumnIndex + "=========");

dataGridView1[\_moves2[i].ColumnIndex, \_moves2[i].RowIndex].Style.BackColor = Color.Black;

dataGridView1[\_moves2[i].ColumnIndex, \_moves2[i].RowIndex].Style.ForeColor = Color.White;

}

}

}

}

}

}

}

Console.WriteLine("////Количество ячеек без пары " + AllCells.Count() + "////");

}

}

public class Tree : Form

{

DataGridView dataGridView1;

public DataGridViewCell thisNodeCell;

List<Tree> movesCell = new List<Tree>();// все возможные ходы

int count;//переменная отвечающая за то, сколько ещё ячеек нужно до составления пары

string value;//значение выбранной ячейки

int IndexRowMain;

int IndexColMain;

public Tree(DataGridView dv, DataGridViewCell cell, int \_count, string \_value, int Row, int Col)

{

dataGridView1 = dv;

thisNodeCell = cell;

count = \_count;

value = \_value;

IndexColMain = Col;

IndexRowMain = Row;

}

public Tree (DataGridView dv, DataGridViewCell cell, int \_count, string \_value)

{

dataGridView1 = dv;

thisNodeCell = cell;

count = \_count;

value = \_value;

IndexColMain = cell.ColumnIndex;

IndexRowMain = cell.RowIndex;

}

//функция для отыскания единственного возможного пути для соединения ячеек

public void AllChild(List<DataGridViewCell> moves, List<DataGridViewCell> \_Moves)

{

if (movesCell.Count() != 0)

{

moves.Add(thisNodeCell);

if (count == 2)

{

for (int i=0; i< moves.Count; i++) \_Moves.Add(moves[i]);

\_Moves.Add(movesCell[0].thisNodeCell);

}

for (int i = 0; i < movesCell.Count(); i++)

{

movesCell[i].AllChild(moves,\_Moves);

}

moves.Remove(thisNodeCell);

}

}

//фуекция для проверки на то, что путь соединения ячеек один

public void LastNode(ref int countLast) {

if (movesCell.Count() != 0)

{

if (count == 2 && movesCell.Count() == 1)

{

++countLast;

Console.WriteLine("Добавление" + countLast);

}

Console.WriteLine("Раскрытие детей");

for (int i = 0; i < movesCell.Count(); i++)

{

movesCell[i].LastNode(ref countLast);

}

}

}

//функция для вычмсления всех путей, которые можно построить от выбранной ячейки

public void CalculatingMoves() {

if (count > 2)

{

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

{

if (i != 0)

{

if (thisNodeCell.RowIndex + i >= 0 && thisNodeCell.RowIndex + i < this.dataGridView1.RowCount)

{

//если ячейка пустая и у нее белый цвет фона

if (this.dataGridView1.Rows[thisNodeCell.RowIndex + i].Cells[thisNodeCell.ColumnIndex].Value == null && this.dataGridView1[thisNodeCell.ColumnIndex, thisNodeCell.RowIndex + i].Style.BackColor != Color.Black)

{

int c = count - 1;

Tree child = new Tree(dataGridView1, this.dataGridView1.Rows[thisNodeCell.RowIndex + i].Cells[thisNodeCell.ColumnIndex], c, value, IndexRowMain, IndexColMain);

child.CalculatingMoves();

movesCell.Add(child);

}

}

if (thisNodeCell.ColumnIndex + i >= 0 && thisNodeCell.ColumnIndex + i < this.dataGridView1.ColumnCount)

{

//если ячейкая пустая и белого цвета

if (this.dataGridView1[thisNodeCell.ColumnIndex + i, thisNodeCell.RowIndex].Value == null && dataGridView1[thisNodeCell.ColumnIndex + i, thisNodeCell.RowIndex].Style.BackColor != Color.Black)

{

int c = count - 1;

Tree child = new Tree(dataGridView1, this.dataGridView1.Rows[thisNodeCell.RowIndex].Cells[thisNodeCell.ColumnIndex + i], c, value, IndexRowMain, IndexColMain);

child.CalculatingMoves();

movesCell.Add(child);

}

}

}

}

}

//если это предпоследний ход

//то проверяется, есть ли вокруг свободная ячейка с нужным значение

if (count == 2)

{

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

{

if (j != 0 && thisNodeCell.RowIndex + j >= 0 && thisNodeCell.RowIndex + j < this.dataGridView1.RowCount)

{

Console.WriteLine("----------" + count);

if (dataGridView1[thisNodeCell.ColumnIndex, thisNodeCell.RowIndex + j].Value != null)

if (thisNodeCell.ColumnIndex != IndexColMain && thisNodeCell.RowIndex + j != IndexRowMain && dataGridView1[thisNodeCell.ColumnIndex, thisNodeCell.RowIndex + j].Style.BackColor!=Color.Black && dataGridView1[thisNodeCell.ColumnIndex, thisNodeCell.RowIndex + j].Value != null && dataGridView1[thisNodeCell.ColumnIndex, thisNodeCell.RowIndex + j].Value.ToString() == value)

{

Console.WriteLine("Последний ход добавление");

Tree child = new Tree(dataGridView1, dataGridView1[thisNodeCell.ColumnIndex, thisNodeCell.RowIndex + j], 2, value);

movesCell.Add(child);

}

}

if (j != 0 && thisNodeCell.ColumnIndex + j >= 0 && thisNodeCell.ColumnIndex + j < this.dataGridView1.RowCount)

{

if (thisNodeCell.ColumnIndex + j != IndexColMain && thisNodeCell.RowIndex != IndexRowMain && dataGridView1[thisNodeCell.ColumnIndex + j, thisNodeCell.RowIndex].Style.BackColor != Color.Black && dataGridView1[thisNodeCell.ColumnIndex + j, thisNodeCell.RowIndex].Value != null && dataGridView1[thisNodeCell.ColumnIndex + j, thisNodeCell.RowIndex].Value.ToString() == value)

{

Tree child = new Tree(dataGridView1, dataGridView1[thisNodeCell.ColumnIndex + j, thisNodeCell.RowIndex], 2, value);

movesCell.Add(child);

}

}

}

}

}

}

}

# Результат работы программы

