МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ижевский государственный технический университет имени М. Т. Калашникова»

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение»

Работа защищена с оценкой

«\_\_\_\_»

Дата \_\_\_\_\_\_\_\_\_\_

Подпись \_\_\_\_\_\_\_\_\_\\_\_\_\_\_\_\_\_\_\_

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

на тему:«Проекты на основе WF»

Выполнил

Студент группы Б03-191-3 Р. А. Гумметов

Руководитель

д.т.н., профессор В. Г. Тарасов

Рецензия:

степень достижения поставленной цели работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

полнота разработки темы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

уровень самостоятельности работы обучающегося\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

недостатки работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

СОДЕРЖАНИЕ

ВВЕДЕНИЕ .............................................................................................................................................3

1. ПОСТАНОВКА ЗАДАЧИ И ИНСТРУМЕНТЫ ДЛЯ ЕЕ РЕШЕНИЯ...........................................4

2. СОЗДАНИЕ ГРАФИЧЕСКОГО ПРИЛОЖЕНИЯ...........................................................................5

2.1 Установка среды разработки и ее компонентов.............................................................................5

2.2 Создание нового проекта в Visual Studio.......................................................................................5

3. ОСНОВЫ РАБОТЫ С WINDOWS FORMS.....................................................................................7

3.1 Графический дизайнер......................................................................................................................7

3.2 Панель инструментов........................................................................................................................7

3.3 Запуск приложения.........................................................................................................................10

3.4 Свойства...........................................................................................................................................11

4. РАЗРАБОТКА ПРОГРАММЫ........................................................................................................13

4.1 Подготовка объектов......................................................................................................................13

4.2 Подготовка элементов....................................................................................................................13

4.3 Алгоритм поиска кратчайшего пути и его построения...............................................................13

4.4 Логика работы программы.............................................................................................................15

5. КОНТРОЛЬНЫЙ ПРИМЕР.............................................................................................................20

ЗАКЛЮЧЕНИЕ.....................................................................................................................................21

ПРИЛОЖЕНИЕ 1..................................................................................................................................22

ВВЕДЕНИЕ

Цель работы заключается в разработке программы, ищущей кратчайший путь и ее графической визуализации.

Программа состоит в том, что пользователь выбирает размер поля, в левом верхнем углу которого находится точка старта, а в правом нижнем - точка финиша. Объект может передвигаться буквой "Г". Нажав кнопку "Начать", появляется поле, программа визуализирует кратчайший путь от точки старта до точки финиша. Если кратчайших путей несколько, программа показывает один из них. По достижению точки финиша, программа показывает сколько всего шагов сделано, а также появляется кнопка "Начать сначала", нажав на которую, всё сбрасывается и программа переходит в самое начало.

1. ПОСТАНОВКА ЗАДАЧИ И ИНСТРУМЕНТЫ ДЛЯ ЕЕ РЕШЕНИЯ

Разработать программу, ищущую кратчайший путь и визуализирующую данный путь.

Расположить на форме кнопки, позволяющие пользователю взаимодействовать с программой, а также само поле.

Объект начинает свое движение из левого верхнего угла поля. Передвижение возможно только буквой "Г". На каждом шаге идет подсчет, какой номер хода происходит в данный момент. Когда объект достигает правый нижний угол, программа выводит количество сделанных шагов. Появляется кнопка "Начать сначала", нажав которую, программа возвращается к началу.

Для реализации данной задачи выбрана платформа .Net, технология Windows forms, язык C#. Программный код пишется в среде разработки Visual Studio 2017.

2. СОЗДАНИЕ ГРАФИЧЕСКОГО ПРИЛОЖЕНИЯ

2.1 Установка среды разработки и ее компонентов

Для создания графического проекта нам потребуется среда разработки Visual Studio. Устанавливаем среду разработки и ее компоненты с помощью Visual Studio Installer (Разработка классических приложений .Net):

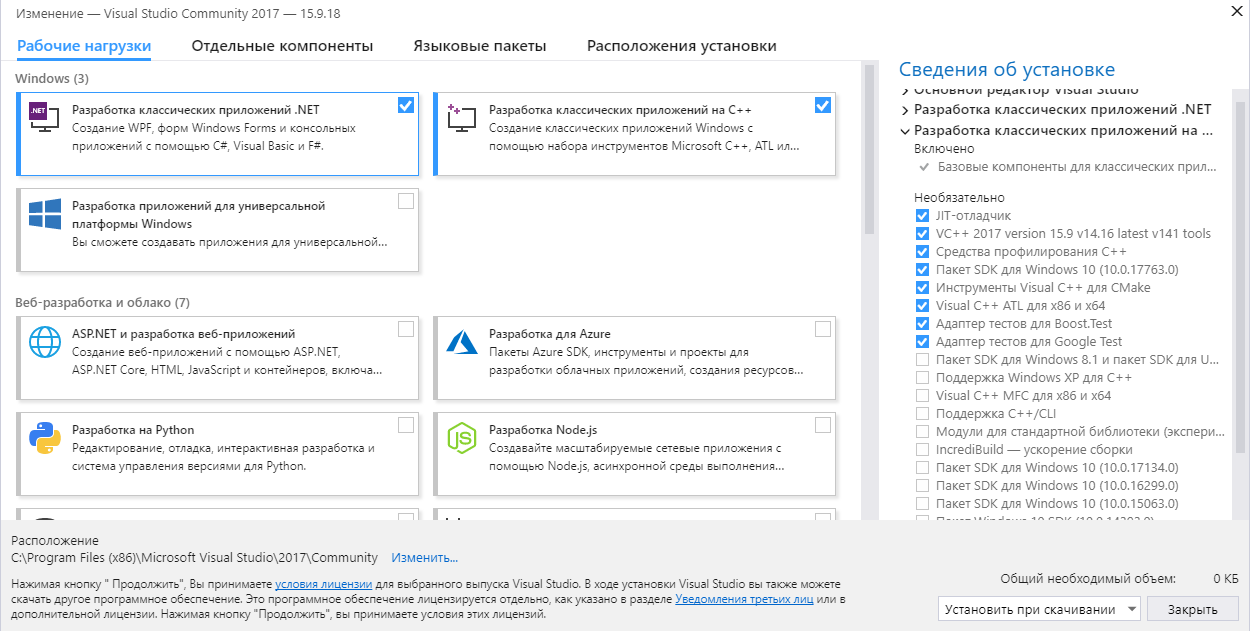


Рис 2.1 Установка среды разработки и ее компонентов

2.2 Создание нового проекта в Visual Studio

После установки среды и всех ее компонентов, запустим Visual Studio и создадим проект графического приложения. Для этого в меню выберем пункт File (Файл) и в подменю выберем New - > Project (Создать - > Проект). После этого перед нами откроется диалоговое окно создания нового проекта:

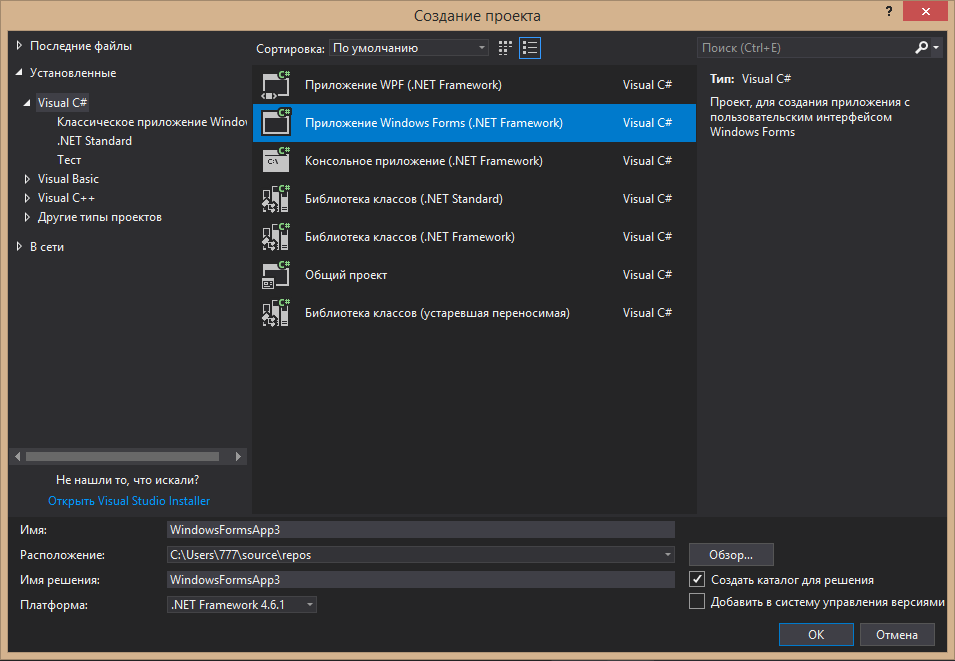


Рис. 2.2 Создание нового проекта

Среди типов проектов выбираем Приложение Windows Forms (.Net Framework). После этого нажимаем OK. После этого Visual Studio откроет наш проект с созданными по умолчанию файлами:

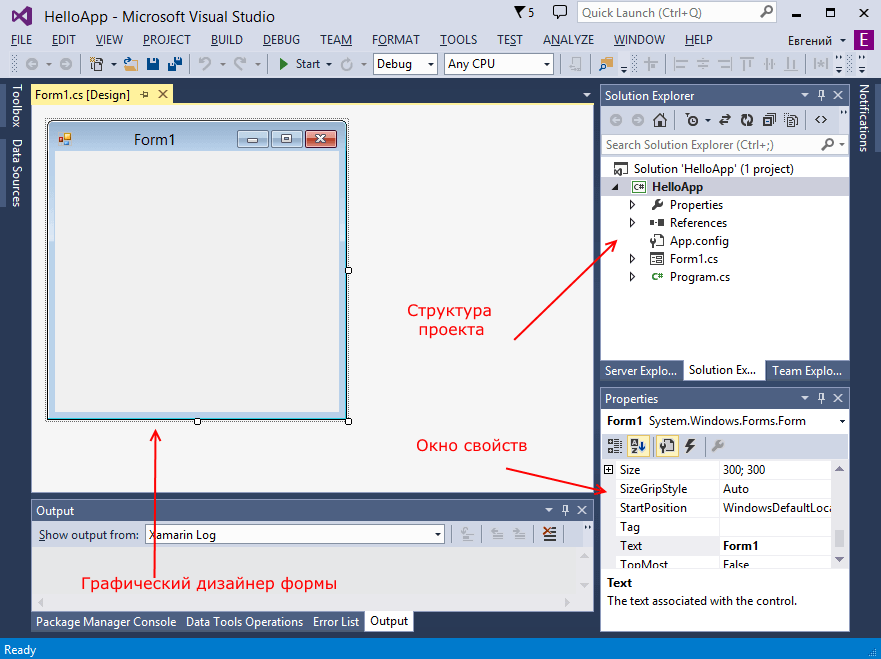


Рис. 2.3 Открытие графического приложения

3. ОСНОВЫ РАБОТЫ С WINDOWS FORMS

3.1 Графический дизайнер

Большую часть пространства Visual Studio занимает графический дизайнер, который содержит форму будущего приложения. Пока она пуста и имеет только заголовок Form1. Справа находится окно файлов решения/проекта - Solution Explorer (Обозреватель решений). Там и находятся все связанные с нашим приложением файлы, в том числе файлы формы Form1.cs.

Внизу справа находится окно свойств - Properties. Так как в данный момент выбрана форма как элемент управления, то в этом поле отображаются свойства, связанные с формой.

Теперь найдем в этом окне свойство формы Text и изменим его значение на любое другое:

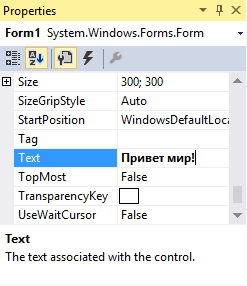


Рис. 3.1 Свойства формы

Таким образом мы поменяли заголовок формы.

3.2 Панель инструментов

Теперь перенесем на поле какой-нибудь элемент управления, например, кнопку. Для этого найдем в левой части Visual Studio вкладку Toolbox (Панель инструментов). Нажмем на эту вкладку, и у нас откроется панель с элементами, откуда мы можем с помощью мыши перенести на форму любой элемент:

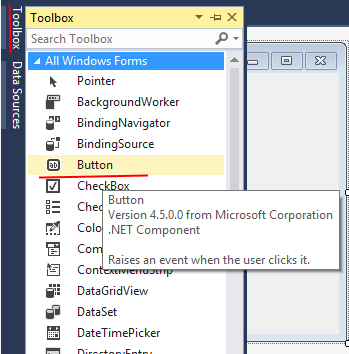


Рис. 3.2 Панель инструментов

Найдем среди элементов кнопку и, захватив ее указателем мыши, перенесем на форму:

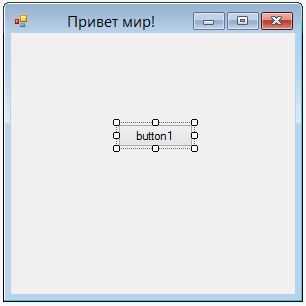


Рис. 3.3 Добавление кнопки на форму

Это визуальная часть. Теперь приступим к самому программированию. Добавим простейший код на языке C#, который бы выводил сообщение по нажатию кнопки. Для этого мы должны перейти в файл кода, который связан с этой формой. Если у нас не открыт файл кода, мы можем нажать на форму правой кнопкой мыши и в появившемся меню выбрать View Code (Посмотреть файл кода):

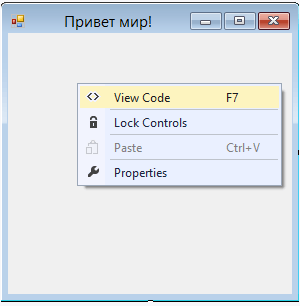


Рис. 3.4 Посмотреть файл кода

Однако можно воспользоваться другим способом, чтобы не писать много лишнего кода. Нужно указатель мыши на кнопку и щелкнуть по ней двойным щелчком. Мы автоматически попадаем в файл кода Form1.cs. Добавим вывод сообщения по нажатию кнопки, изменив код следующим образом:

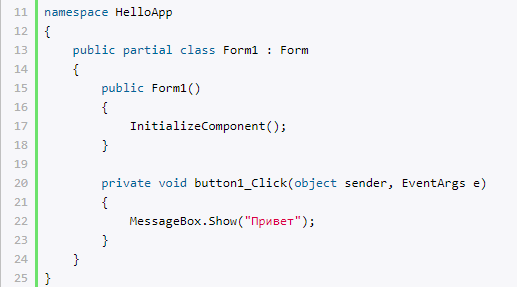


Рис. 3.5 Сообщение по нажатию кнопки

3.3 Запуск приложения

Чтобы запустить приложение в режиме отладки, нажмем на клавишу F5 или на зеленую стрелочку на панели Visual Studio. После этого запустится наша форма с одинокой кнопкой. И если мы нажмем на кнопку на форме, то нам будет отображено сообщение с приветствием.

После запуска приложения студия компилирует его в файл с расширением exe. Найти данный файл можно, зайдя в папку проекта и далее в каталог bin/Debug или bin/Release

Рассмотрев вкратце создание проекта графического приложения, мы можем перейти к обзору основных компонентов и начнем мы с форм.

Внешний вид приложения является нам преимущественно через формы. Формы являются основными строительными блоками. Они предоставляют контейнер для различных элементов управления. А механизм событий позволяет элементам формы отзываться на ввод пользователя, и, таким образом, взаимодействовать с пользователем.

При открытии проекта в Visual Studio в графическом редакторе мы можем увидеть визуальную часть формы - ту часть, которую мы видим после запуска приложения и куда мы переносим элементы с панели управления. Но на самом деле форма скрывает мощный функционал, состоящий из методов, свойств, событий и прочее. Рассмотрим основные свойства форм.

Если мы запустим приложение, то нам отобразится одна пустая форма. Однако даже такой простой проект с пустой формой имеет несколько компонентов:

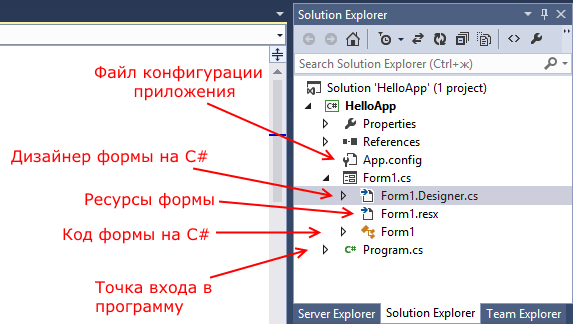


Рис. 3.6 Компоненты

3.4 Свойства

С помощью специального окна Properties (Свойства) справа Visual Studio предоставляет нам удобный интерфейс для управления свойствами элемента:

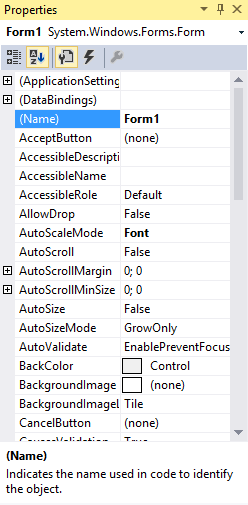


Рис. 3.7 Свойства элемента

Большинство этих свойств оказывает влияние на визуальное отображение формы. Основные свойства:

* Name: устанавливает имя формы - точнее имя класса, который наследуется от класса Form
* BackColor: указывает на фоновый цвет формы. Щелкнув на это свойство, мы сможем выбрать тот цвет, который нам подходит из списка предложенных цветов или цветовой палитры
* BackgroundImage: указывает на фоновое изображение формы
* BackgroundImageLayout: определяет, как изображение, заданное в свойстве BackgroundImage, будет располагаться на форме.
* ControlBox: указывает, отображается ли меню формы. В данном случае под меню понимается меню самого верхнего уровня, где находятся иконка приложения, заголовок формы, а также кнопки минимизации формы и крестик. Если данное свойство имеет значение false, то мы не увидим ни иконку, ни крестика, с помощью которого обычно закрывается форма
* Cursor: определяет тип курсора, который используется на форме
* Enabled: если данное свойство имеет значение false, то она не сможет получать ввод от пользователя, то есть мы не сможем нажать на кнопки, ввести текст в текстовые поля и т.д.
* Font: задает шрифт для всей формы и всех помещенных на нее элементов управления. Однако, задав у элементов формы свой шрифт, мы можем тем самым переопределить его
* ForeColor: цвет шрифта на форме
* FormBorderStyle: указывает, как будет отображаться граница формы и строка заголовка. Устанавливая данное свойство в None можно создавать внешний вид приложения произвольной формы
* HelpButton: указывает, отображается ли кнопка справки формы
* Icon: задает иконку формы
* Location: определяет положение по отношению к верхнему левому углу экрана, если для свойства StartPosition установлено значение Manual
* MaximizeBox: указывает, будет ли доступна кнопка максимизации окна в заголовке формы
* MinimizeBox: указывает, будет ли доступна кнопка минимизации окна
* MaximumSize: задает максимальный размер формы
* MinimumSize: задает минимальный размер формы
* Opacity: задает прозрачность формы
* Size: определяет начальный размер формы
* StartPosition: указывает на начальную позицию, с которой форма появляется на экране
* Text: определяет заголовок формы
* TopMost: если данное свойство имеет значение true, то форма всегда будет находиться поверх других окон
* Visible: видима ли форма, если мы хотим скрыть форму от пользователя, то можем задать данному свойству значение false
* WindowState: указывает, в каком состоянии форма будет находиться при запуске: в нормальном, максимизированном или минимизированном

4. РАЗРАБОТКА ПРОГРАММЫ

4.1 Подготовка объектов

Для игры потребуется окно размером 500x600 px, название окна "Поиск кратчайшего пути".

4.2 Подготовка элементов

Создадим Label размером 220x60 с текстом "Выберите размер поля", Label размером 35х35 с текстом "m", Label размером 35х35 с текстом "n", 4 кнопки с размерами 35х35, с помощью которых можно выбирать размер поля, кнопку с размером 100х30 с текстом "ОК", кнопку с размером 100х30 "Начать", кнопку с размером 150х60 "Начать сначала", PictureBox с размером 301х301.

В тексте создаем лейблы, которые отображают выбранные размеры поля:

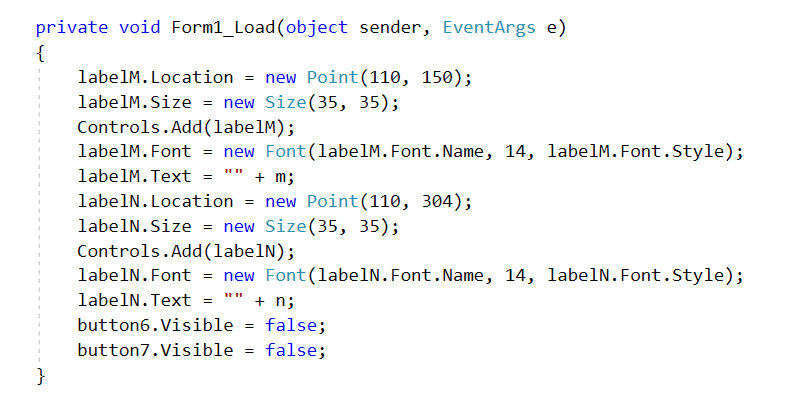


Рис. 4.1 Создание лейблов для отображения размеров поля

4.3 Алгоритм поиска кратчайшего пути и его построения

В программе для поиска кратчайшего пути используется волновой алгоритм. Алгоритм работает на дискретном рабочем поле (ДРП), представляющем собой ограниченную замкнутой линией фигуру, не обязательно прямоугольную, разбитую на прямоугольные ячейки, в частном случае — квадратные. Множество всех ячеек ДРП разбивается на подмножества: «проходимые» (свободные), т. е при поиске пути их можно проходить, «непроходимые» (препятствия), путь через эту ячейку запрещён, стартовая ячейка (источник) и финишная (приемник). Назначение стартовой и финишной ячеек условно, достаточно — указание пары ячеек, между которыми нужно найти кратчайший путь.

Алгоритм предназначен для поиска кратчайшего пути от стартовой ячейки к конечной ячейке, если это возможно, либо, при отсутствии пути, выдать сообщение о непроходимости.

Работа алгоритма включает в себя три этапа: инициализацию, распространение волны и восстановление пути.

Во время инициализации строится образ множества ячеек обрабатываемого поля, каждой ячейке приписываются атрибуты проходимости/непроходимости, запоминаются стартовая и финишная ячейки. При выполнении условий проходимости и непринадлежности её к ранее помеченным в пути ячейкам, в атрибут ячейки записывается число, равное количеству шагов от стартовой ячейки, от стартовой ячейки на первом шаге это будет 1. Каждая ячейка, меченная числом шагов от стартовой ячейки, становится стартовой и из неё порождаются очередные шаги в соседние ячейки. Очевидно, что при таком переборе будет найден путь от начальной ячейки к конечной, либо очередной шаг из любой порождённой в пути ячейки будет невозможен. Восстановление кратчайшего пути происходит в обратном направлении: при выборе ячейки от финишной ячейки к стартовой на каждом шаге выбирается ячейка, имеющая атрибут расстояния от стартовой на единицу меньше текущей ячейки. Очевидно, что таким образом находится кратчайший путь между парой заданных ячеек. Трасс с минимальной числовой длиной пути, как при поиске пути в окрестностях Мура, так и фон Неймана может существовать несколько. Выбор окончательного пути в приложениях диктуется другими соображениями, находящимися вне этого алгоритма.

4.4 Логика работы программы

Вид формы при запуске программы:

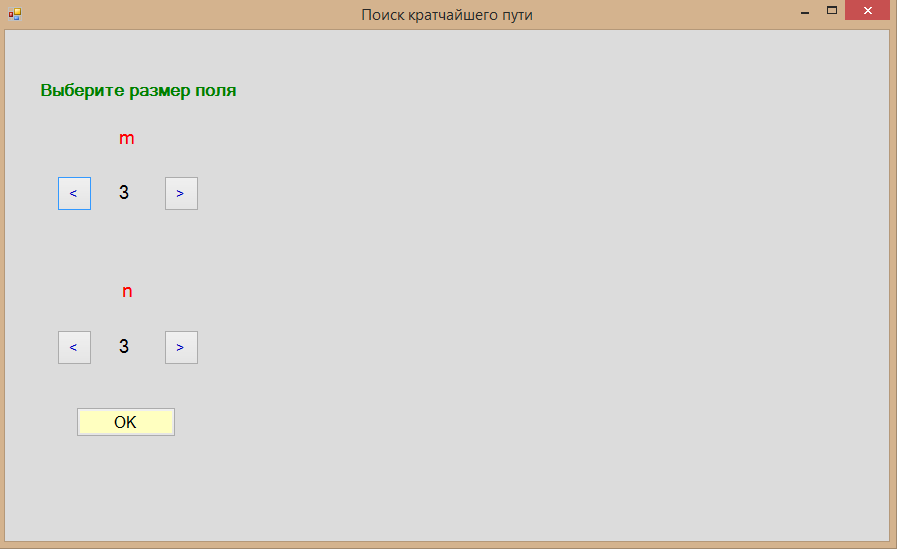


Рис. 4.2 Форма при запуске программы

После выбора размеров поля, и нажатии кнопки "ОК" под двумерный массив field выделяется память в соответствии с выбранными размерами поля, стартовая точка добавляется в очередь, программа вызывает функцию testP поиска кратчайшего пути волновым алгоритмом, после этого вызывается функция find\_only\_way восстановления пути в обратном направлении. В массив way записываются индексы клеток, по которым построен путь:

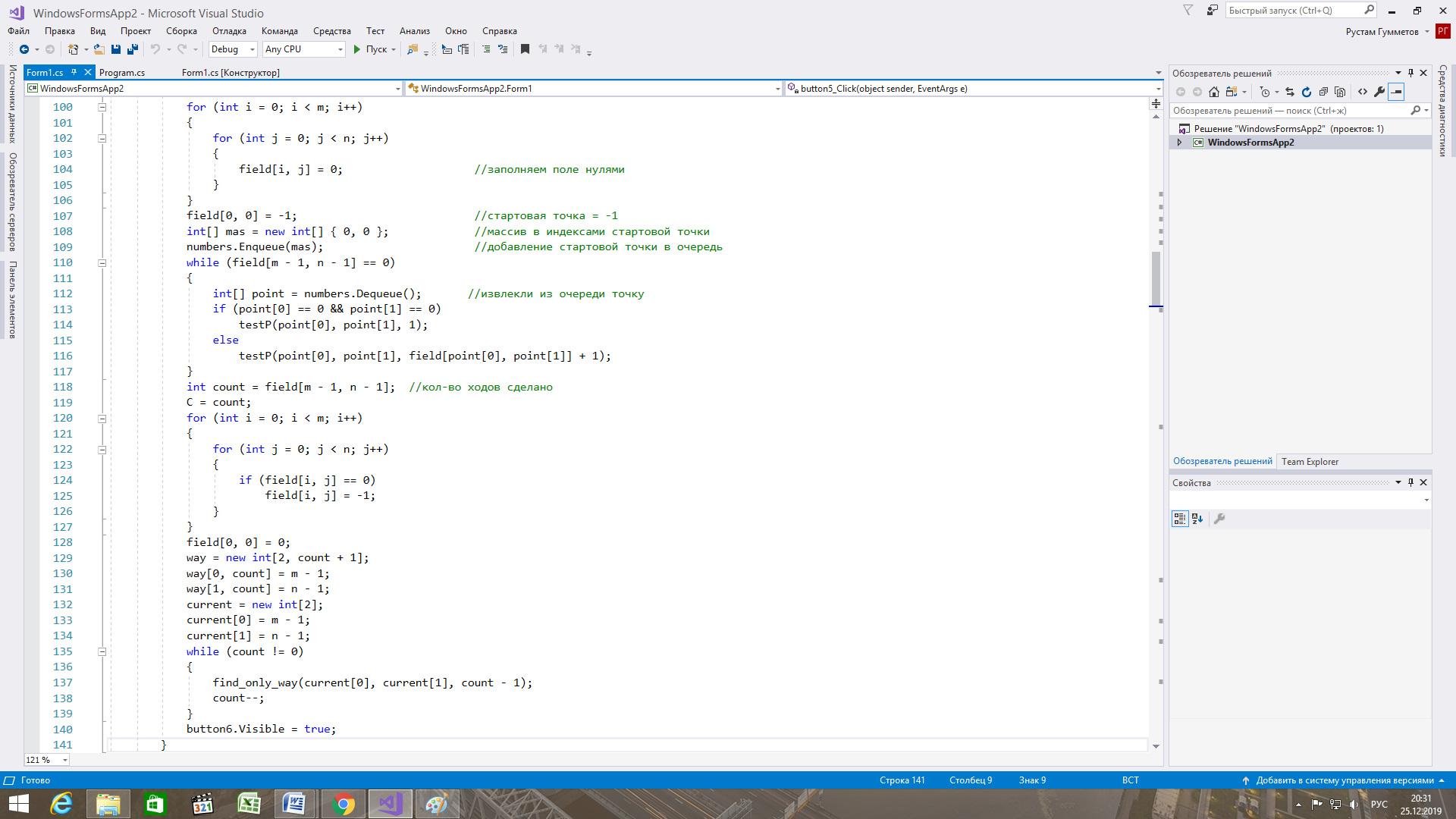


Рис. 4.3 Создание поля, построение пути

Кнопки для изменения размеров поля блокируются, появляется кнопка "Начать", которая в этот момент единственная доступная для нажатия:

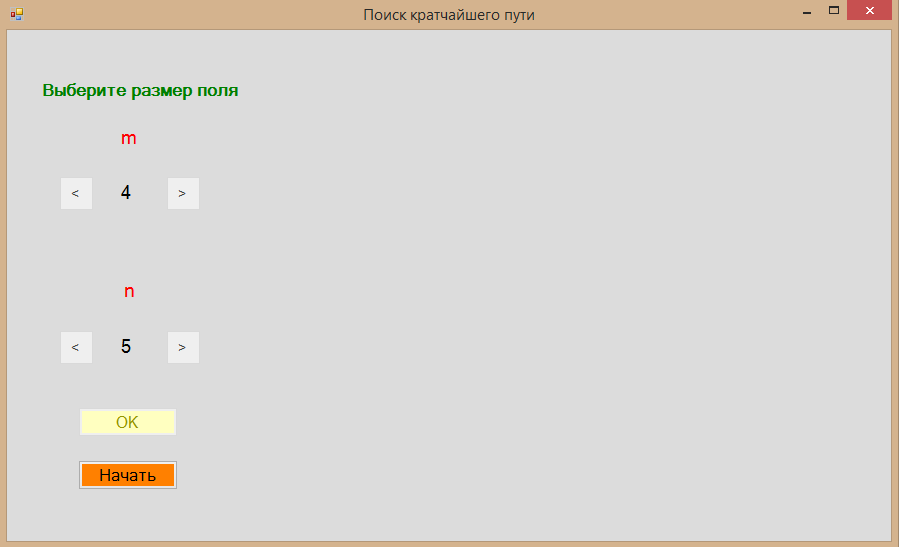


Рис. 4.4 Появление кнопки "Начать"

После нажатия на кнопку начать в тексте создаем лейблы с надписями "Путь", "Ход", номером шага:

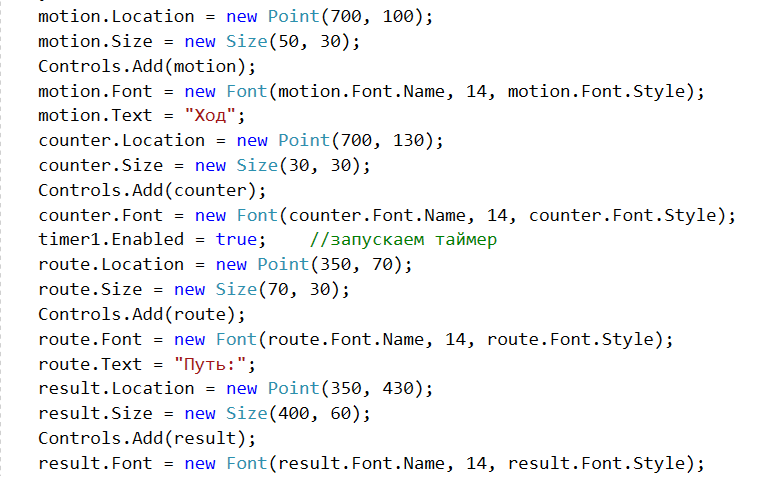


Рис. 4.5 Создание лейблов для надписи путь, отображения номера хода и общего количества шагов

Появляется поле с выбранными размерами mn, запускается таймер с интервалом 1с. Каждую секунду на поле закрашивается одна клетка - место, в которое сделан очередной шаг. Справа от поля выводится номер сделанного хода:

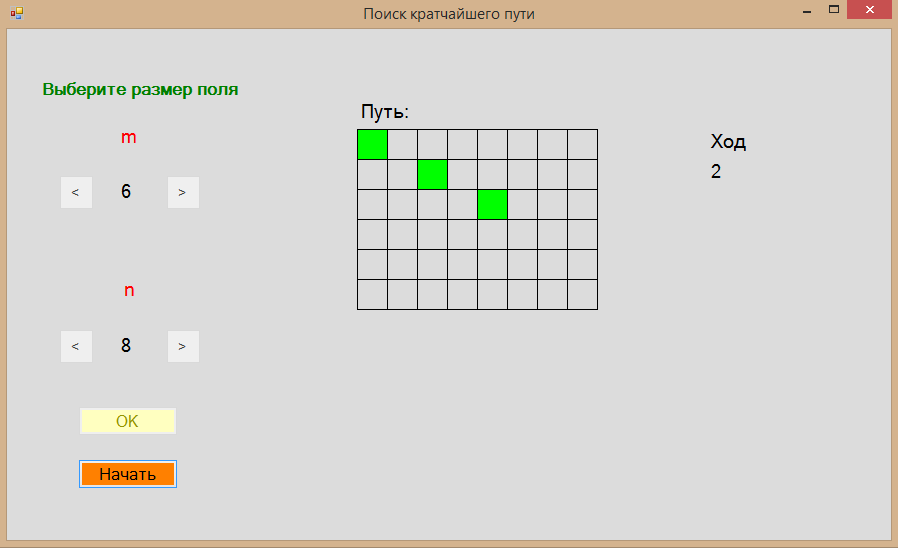


Рис. 4.6 Визуализация пути

В момент, когда закрасится клетка в правом нижнем углу, таймер останавливается, снизу от поля выводится число, показывающее общее количество сделанных шагов и появляется кнопка "Начать сначала":

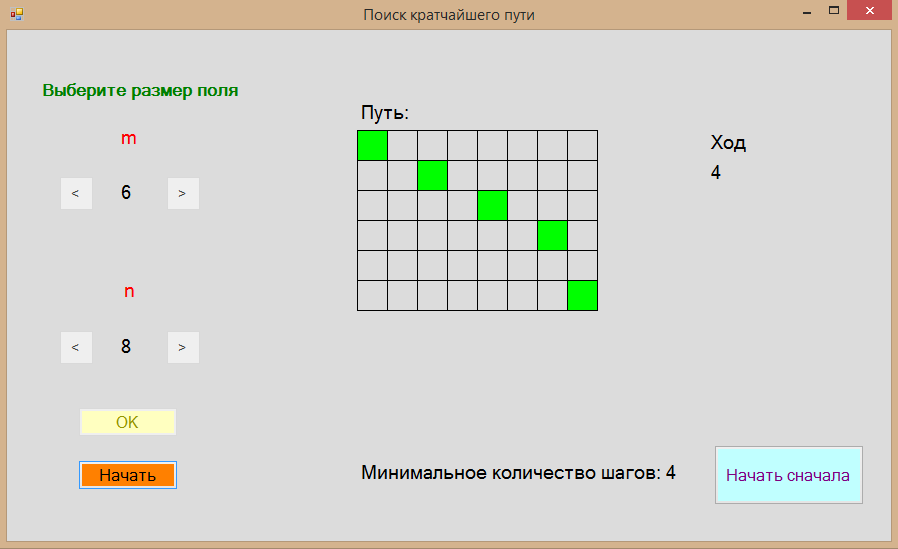


Рис. 4.7 Конец пути

После нажатия кнопки "Начать сначала" кнопки для выбора размера поля снова становятся доступны, поле стирается, очищаются массивы для поля, найденного пути. Все оставшиеся в очереди элементы из очереди извлекаются. Переменные принимают свои начальные значения:

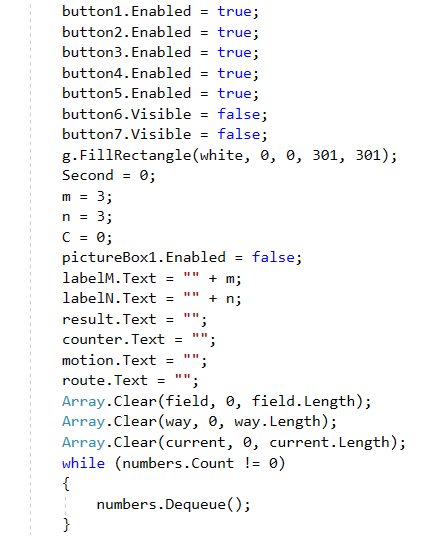


Рис. 4.8

После этого форма приобретает начальный вид, программу можно начать сначала:

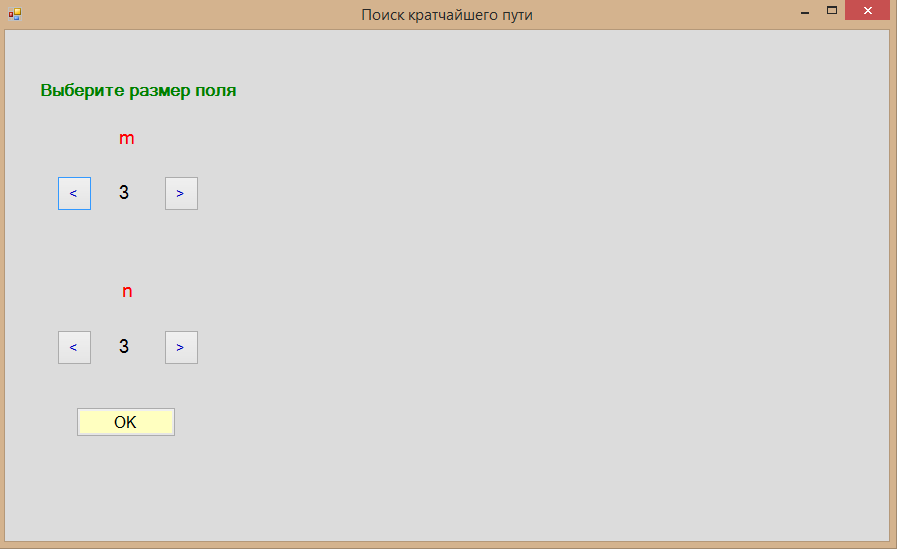
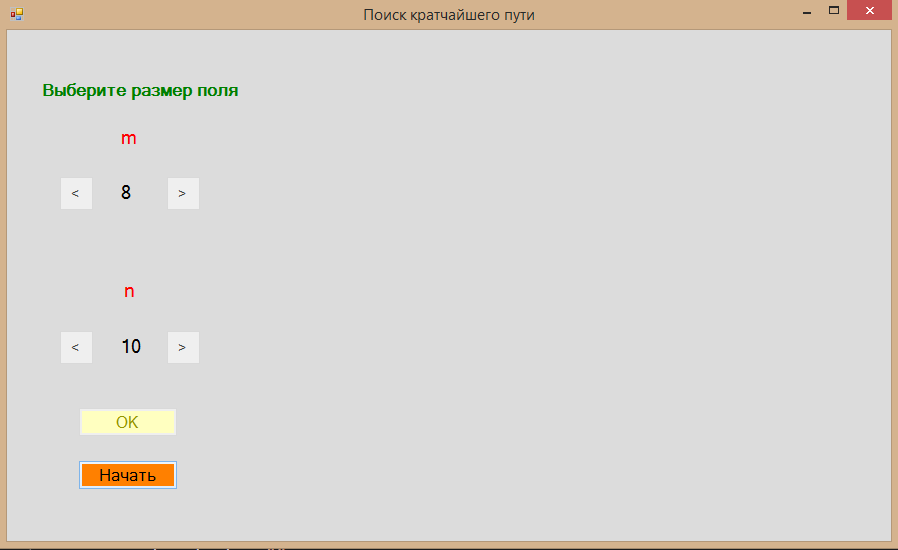
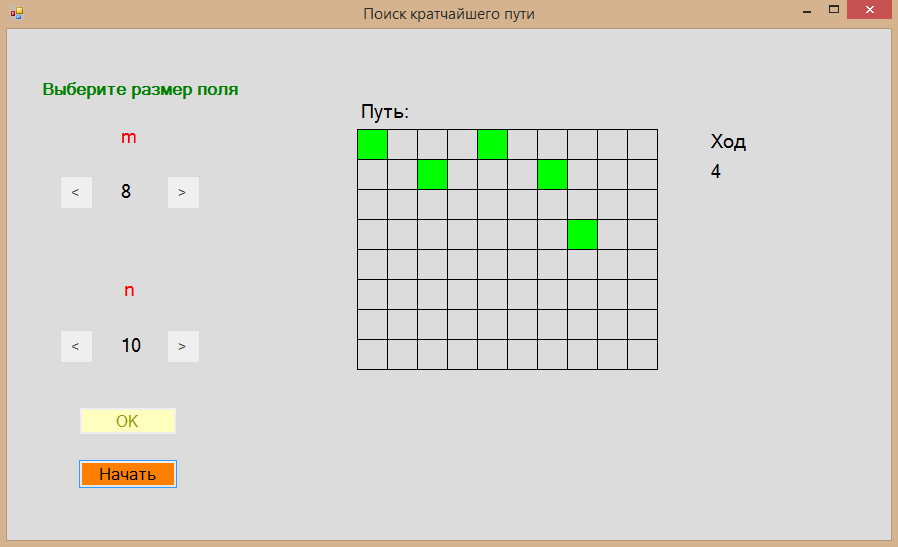


Рис. 4.9 Начать сначала

5. КОНТРОЛЬНЫЙ ПРИМЕР

На рис. 5.1 представлен пример работы приложения:





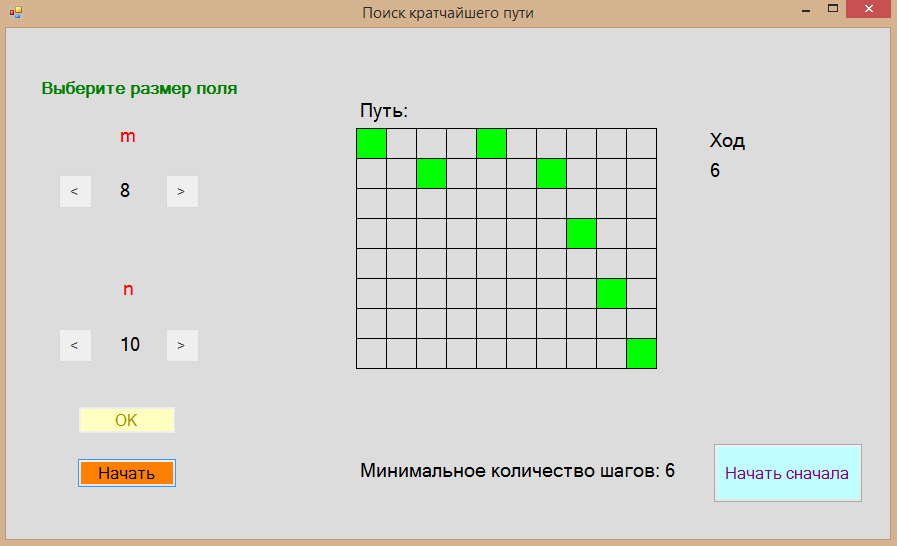


Рис. 5.1 Работа программы

ЗАКЛЮЧЕНИЕ

В ходе курсовой работы разработана программа, ищущая кратчайший путь и ее графическая визуализация.

Для реализации приложения выбрана платформа .Net, технология Windows forms, язык C#. Программный код написан в среде разработки Visual Studio 2017.

ПРИЛОЖЕНИЕ 1

ТЕКСТ ПРОГРАММЫ

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 WindowsFormsApp2

{

public partial class Form1 : Form

{

static public Queue<int[]> numbers = new Queue<int[]>(); //создание очереди, элементы которой - массивы (индексы точек)

static public int m = 3, n = 3; //размеры поля

static public int[,] field;

static public int[,] way; //массив индексов пути

static public int[] current; //текущие координаты

static public int C; //количество ходов

Label labelM = new Label();

Label labelN = new Label();

Label route = new Label();

Label result = new Label();

Label motion = new Label(); //для надписи ход

Label counter = new Label(); //номер хода

Graphics g;

Pen line = new Pen(Color.Black, 1);

SolidBrush rec = new SolidBrush(Color.Lime);

SolidBrush white = new SolidBrush(Color.Gainsboro);

static public int Second = 0; //для таймера

public Form1()

{

InitializeComponent();

timer1.Interval = 1000;

}

private void Form1\_Load(object sender, EventArgs e)

{

labelM.Location = new Point(110, 150);

labelM.Size = new Size(35, 35);

Controls.Add(labelM);

labelM.Font = new Font(labelM.Font.Name, 14, labelM.Font.Style);

labelM.Text = "" + m;

labelN.Location = new Point(110, 304);

labelN.Size = new Size(35, 35);

Controls.Add(labelN);

labelN.Font = new Font(labelN.Font.Name, 14, labelN.Font.Style);

labelN.Text = "" + n;

button6.Visible = false;

button7.Visible = false;

}

private void button1\_Click(object sender, EventArgs e)

{

if (m > 3)

{

m--;

labelM.Text = "" + m;

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (m < 10)

{

m++;

labelM.Text = "" + m;

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (n > 3)

{

n--;

labelN.Text = "" + n;

}

}

private void button4\_Click(object sender, EventArgs e)

{

if (n < 10)

{

n++;

labelN.Text = "" + n;

}

}

private void button5\_Click(object sender, EventArgs e)

{

button1.Enabled = false;

button2.Enabled = false;

button3.Enabled = false;

button4.Enabled = false;

button5.Enabled = false;

field = new int[m, n]; //создаем поле

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

{

for (int j = 0; j < n; j++)

{

field[i, j] = 0; //заполняем поле нулями

}

}

field[0, 0] = -1; //стартовая точка = -1

int[] mas = new int[] { 0, 0 }; //массив в индексами стартовой точки

numbers.Enqueue(mas); //добавление стартовой точки в очередь

while (field[m - 1, n - 1] == 0)

{

int[] point = numbers.Dequeue(); //извлекли из очереди точку

if (point[0] == 0 && point[1] == 0)

testP(point[0], point[1], 1);

else

testP(point[0], point[1], field[point[0], point[1]] + 1);

}

int count = field[m - 1, n - 1]; //кол-во ходов сделано

C = count;

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

{

for (int j = 0; j < n; j++)

{

if (field[i, j] == 0)

field[i, j] = -1;

}

}

field[0, 0] = 0;

way = new int[2, count + 1];

way[0, count] = m - 1;

way[1, count] = n - 1;

current = new int[2];

current[0] = m - 1;

current[1] = n - 1;

while (count != 0)

{

find\_only\_way(current[0], current[1], count - 1);

count--;

}

button6.Visible = true;

}

static public void testP(int i, int j, int count) //проверка стен,

{ //доступных ходов и заполнение поля

if ((i - 2) >= 0) //на 2 вверх

{

if ((j - 1) >= 0) //на 1 влево

{

if (field[i - 2, j - 1] == 0)

{

field[i - 2, j - 1] = count;

int[] mas = new int[] { i - 2, j - 1 };

numbers.Enqueue(mas);

}

}

if ((j + 1) < n) //на 1 вправо

{

if (field[i - 2, j + 1] == 0)

{

field[i - 2, j + 1] = count;

int[] mas = new int[] { i - 2, j + 1 };

numbers.Enqueue(mas);

}

}

}

if ((i + 2) < m) //на 2 вниз

{

if ((j - 1) >= 0) //на 1 влево

{

if (field[i + 2, j - 1] == 0)

{

field[i + 2, j - 1] = count;

int[] mas = new int[] { i + 2, j - 1 };

numbers.Enqueue(mas);

}

}

if ((j + 1) < n) //на 1 вправо

{

if (field[i + 2, j + 1] == 0)

{

field[i + 2, j + 1] = count;

int[] mas = new int[] { i + 2, j + 1 };

numbers.Enqueue(mas);

}

}

}

if ((j - 2) >= 0) //на 2 влево

{

if ((i - 1) >= 0) //на 1 вверх

{

if (field[i - 1, j - 2] == 0)

{

field[i - 1, j - 2] = count;

int[] mas = new int[] { i - 1, j - 2 };

numbers.Enqueue(mas);

}

}

if ((i + 1) < m) //на 1 вниз

{

if (field[i + 1, j - 2] == 0)

{

field[i + 1, j - 2] = count;

int[] mas = new int[] { i + 1, j - 2 };

numbers.Enqueue(mas);

}

}

}

if ((j + 2) < n) //на 2 вправо

{

if ((i - 1) >= 0) //на 1 вверх

{

if (field[i - 1, j + 2] == 0)

{

field[i - 1, j + 2] = count;

int[] mas = new int[] { i - 1, j + 2 };

numbers.Enqueue(mas);

}

}

if ((i + 1) < m) //на 1 вниз

{

if (field[i + 1, j + 2] == 0)

{

field[i + 1, j + 2] = count;

int[] mas = new int[] { i + 1, j + 2 };

numbers.Enqueue(mas);

}

}

}

}

static public void find\_only\_way(int i, int j, int search) //ищем обратный путь

{

bool f = false;

while (f == false)

{

if ((i - 2) >= 0) //на 2 вверх

{

if ((j - 1) >= 0) //на 1 влево

{

if (field[i - 2, j - 1] == search)

{

current[0] = i - 2;

current[1] = j - 1;

way[0, search] = i - 2;

way[1, search] = j - 1;

f = true;

break;

}

}

if ((j + 1) < n) //на 1 вправо

{

if (field[i - 2, j + 1] == search)

{

current[0] = i - 2;

current[1] = j + 1;

way[0, search] = i - 2;

way[1, search] = j + 1;

f = true;

break;

}

}

}

if ((i + 2) < m) //на 2 вниз

{

if ((j - 1) >= 0) //на 1 влево

{

if (field[i + 2, j - 1] == search)

{

current[0] = i + 2;

current[1] = j - 1;

way[0, search] = i + 2;

way[1, search] = j - 1;

f = true;

break;

}

}

if ((j + 1) < n) //на 1 вправо

{

if (field[i + 2, j + 1] == search)

{

current[0] = i + 2;

current[1] = j + 1;

way[0, search] = i + 2;

way[1, search] = j + 1;

f = true;

break;

}

}

}

if ((j - 2) >= 0) //на 2 влево

{

if ((i - 1) >= 0) //на 1 вверх

{

if (field[i - 1, j - 2] == search)

{

current[0] = i - 1;

current[1] = j - 2;

way[0, search] = i - 1;

way[1, search] = j - 2;

f = true;

break;

}

}

if ((i + 1) < m) //на 1 вниз

{

if (field[i + 1, j - 2] == search)

{

current[0] = i + 1;

current[1] = j - 2;

way[0, search] = i + 1;

way[1, search] = j - 2;

f = true;

break;

}

}

}

if ((j + 2) < n) //на 2 вправо

{

if ((i - 1) >= 0) //на 1 вверх

{

if (field[i - 1, j + 2] == search)

{

current[0] = i - 1;

current[1] = j + 2;

way[0, search] = i - 1;

way[1, search] = j + 2;

f = true;

break;

}

}

if ((i + 1) < m) //на 1 вниз

{

if (field[i + 1, j + 2] == search)

{

current[0] = i + 1;

current[1] = j + 2;

way[0, search] = i + 1;

way[1, search] = j + 2;

f = true;

break;

}

}

}

}

}

private void button6\_Click(object sender, EventArgs e)

{

g = pictureBox1.CreateGraphics();

for (int x = 0; x < n \* 30; x += 30)

{

for (int y = 0; y < m \* 30; y += 30)

{

g.DrawRectangle(line, x, y, 30, 30);

}

}

motion.Location = new Point(700, 100);

motion.Size = new Size(50, 30);

Controls.Add(motion);

motion.Font = new Font(motion.Font.Name, 14, motion.Font.Style);

motion.Text = "Ход";

counter.Location = new Point(700, 130);

counter.Size = new Size(30, 30);

Controls.Add(counter);

counter.Font = new Font(counter.Font.Name, 14, counter.Font.Style);

timer1.Enabled = true; //запускаем таймер

route.Location = new Point(350, 70);

route.Size = new Size(70, 30);

Controls.Add(route);

route.Font = new Font(route.Font.Name, 14, route.Font.Style);

route.Text = "Путь:";

result.Location = new Point(350, 430);

result.Size = new Size(400, 60);

Controls.Add(result);

result.Font = new Font(result.Font.Name, 14, result.Font.Style);

}

private void timer1\_Tick(object sender, EventArgs e)

{

if (Second > C)

{

result.Text = "Минимальное количество шагов: " + C;

button7.Visible = true;

timer1.Stop();

}

else

{

g.FillRectangle(rec, 1 + way[1, Second] \* 30, 1 + way[0, Second] \* 30, 29, 29);

counter.Text = "" + Second;

Second++;

}

}

private void button7\_Click(object sender, EventArgs e)

{

button1.Enabled = true;

button2.Enabled = true;

button3.Enabled = true;

button4.Enabled = true;

button5.Enabled = true;

button6.Visible = false;

button7.Visible = false;

g.FillRectangle(white, 0, 0, 301, 301);

Second = 0;

m = 3;

n = 3;

C = 0;

pictureBox1.Enabled = false;

labelM.Text = "" + m;

labelN.Text = "" + n;

result.Text = "";

counter.Text = "";

motion.Text = "";

route.Text = "";

Array.Clear(field, 0, field.Length);

Array.Clear(way, 0, way.Length);

Array.Clear(current, 0, current.Length);

while (numbers.Count != 0)

{

numbers.Dequeue();

}

}

}

}