

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу «Программирование»

на тему «Разработка программы сложной структуры методом
нисходящего программирования»

Выполнил:

студент группы 21BB2

Изосин А.А.

Принял:

Голотенков.Н.О

07.06.2022



07.06.2022

Пенза 2022

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники
Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ



М.А. Матрозин

« » 20 г.

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Цзосину Алексею Александровичу Группа 21ВВ2

Тема проекта: Разработка программы сложной структуры методом нисходящего программирования.

Исходные данные (технические требования) на проектирование

Приложение: игра "Батрис"
Функции: сохранение имени пользователя и его
результата, вывод лучиц по запросу, вывод индеек
мачи автора программы

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.
5. Пользовательский интерфейс должен быть встроен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер
9. Среда разработки ПО – Microsoft Visual Studio.

Объем работы по курсу

1. Расчетная часть

Разработка программы.

2. Графическая часть

Схема данных, схема ресурсов системы, схема работы системы,
иерархическая структура программы, схема взаимодействия программы.

3. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания "22" декабря

Дата защиты проекта " " "

Руководитель Голотенков Н. О.

Задание получил "22" декабря 2022г.

Студент Ильин А. А.

Оглавление

Введение.....	5
Выбор решения.....	7
Постановка задачи.....	7
Описание разработки программы	8
Отладка и тестирование.....	8
Описание программы	9
Руководство пользователя.....	10
Заключение	13
Список используемых источников	14
Приложение А. Иерархическая структура программы	15
Приложение В. Схема данных программы	16
Приложение С. Листинг программы	17
Приложение D. Схемы функций программы	31
Приложение Е. Схема взаимодействия программы	33

Введение

В данном документе описывается программа "Тетрис". Программа осуществляет вывод на экран случайным образом падение различных фигур. Входными данными является управление падающими фигурами. Назначение программы - развлечение играющих, совершенствование их координации и логического мышления. Программа может применяться в качестве игровой на разных типах персональных компьютеров.

История игры Тетрис начинается в июне 1985 года. Тетрис был изобретен Алексеем Пажитновым, а затем был интегрирован на ПК IBM Вадимом Герасимовым. После чего игра Тетрис начала распространяться по всей Москве, а затем уже и по всему миру. Сначала она была доставлена в Венгрию, где венгерские программисты интегрировали Тетрис для Apple II и Commodore 64.

Игра была замечена в мире, и несколько представителей крупных компаний, обращались к автору Тетриса, чтобы купить права на распространение игры. Алексей подписывает контракт с Mirrorsoft UK и Spectrum Holobyte, предоставляя им права на компьютерным версии Тетрис. После того, как первые копии Тетриса для домашних компьютеров были преданы, игра приобрела популярность среди населения и стала самой продаваемой компьютерной игрой в Англии и США в 1988 году.

Существует множество способов реализации данной программы. Их можно разделить по функциональности:

- математическое описание движений фигур;

- графическое отображение движений фигур;

В математической части рассматриваются основные принципы и законы движений фигур. Это самая важная часть программы. От неё зависит правильная работоспособность программы. Для её реализации можно использовать различные алгоритмы. Например, описать движение фигуры двумя линейными функциями. Одна будет отвечать за

расположение фигуры по горизонтали, другая по вертикале. Меняя за определенные промежутки времени значения переменных этих функций, будет меняться положение фигуры на плоскости. Третья функция будет отвечать за очистку полностью заполненных горизонталей. Основными недостатками этого способа является объявление большого числа переменных, отвечающих за описание уже упавших фигур, и создание большого числа дополнительных функций, отвечающих за поворот фигур вокруг своей оси.

Другой способ математического описания движения фигур и заполнения поля тетриса - создание двумерной матрицы $i*j$. Через определенный промежуток времени будут изменяться значения, соответствующие положению фигур на плоскости и уже упавшим фигурам. Т.е. значение матрицы, расположенное на I -ой строке и в J -ом столбце, будет соответствовать части фигуры, расположенной на I -ой горизонтали и J -ой вертикали.

Приведенные выше способы не являются единственными. Они лишь наиболее популярны в реализации программы "Тетрис" среди программистов. Для графического отображения фигур и поля тетриса также существует большое количество различных способов. Например, двигать на плоскости уже готовые рисунки фигур "Тетриса". Тогда рисунки фигур будут храниться отдельными графическими файлами в памяти компьютера. Сложность данного способа реализации заключается в отображении уже упавших фигур и отображение повернутой фигуры вокруг своей оси. Для этого придется постоянно создавать новый рисунок поля из рисунков, хранимых в памяти компьютера.

При разработке программы для игры Тетрис был использован объектно-ориентированный язык программирования Visual C#. Математическая часть программы была создана с помощью двумерной матрицы. Графическое отображение было реализовано с помощью графических возможностей языка Visual C#.

Выбор решения

Приложение “Tetris” состоит из нескольких модулей:

- 1) Form1.cs
- 2) MapController.cs
- 3) Shape.cs
- 4) Sortirovka.cs

Файл Form1.cs является основным модулем, который контролирует основные элементы интерфейса и вызывает различные функции из других файлов. А также содержит функции, необходимые для реализации игрового процесса.

Все выше сказанное удовлетворяет требованию «многомодульность» в данной курсовой работе. Управление программой осуществляется с помощью клавиатуры.

Постановка задачи

Требуется написать программу - тетрис, осуществляющую падение фигур и наложение их друг на друга, а также подсчет заполненных линий.

Многомодульность программы. Программа должна быть поделена на логические модули. Это упростит поиск ошибок при отладке и тестировании программы, а также позволит легко расширять функционал программы.

Использование сложных типов данных – массивов, структур, файлов. Это необходимо для более простой и интуитивной обработки данных в коде программы.

Режим работы видеосистемы – текстовый/графический. Для начала необходимо определиться с типом интерфейса и с элементами управления, затем необходимо изучить способы их реализации.

Устройство ввода-вывода – клавиатура. Необходимо различать и идентифицировать действия, произведенные с их помощью, это облегчит использование программы.

Интерфейс должен быть построен на основе меню. Это необходимо для создания интуитивно понятного интерфейса для пользователя.

Описание разработки программы

Разработанная программа состоит из нескольких модулей, которые реализовывались в следующем порядке:

- 1) Form1.cs – основной файл программы
- 2) MapController.cs – файл предназначенный для отрисовки и обработки игрового поля
- 3) Shape.cs – файл для обработки матриц и их позиционировании на игровом поле
- 4) Sortirovka.cs – сортировка игроков для таблицы лидеров

По порядку реализации модулей можно понять, что использовался нисходящий подход реализации программы. Этот метод основан на том, что сначала проектируются основные компоненты программы, а затем уже дорабатываются её мелкие детали. Это позволяет упростить её тестирование, так как каждый модуль тестируется сразу после его реализации, что помогает в создании более структурированных программ.

Отладка и тестирование

В качестве среды разработки была выбрана программа Microsoft Visual Studio 2019. Программа обладает всеми средствами необходимыми при разработке и отладке программы. Для отладки использовались несколько возможностей Visual Studio: точка останова, трассировка, анализ содержимого переменных.

Тестирование проводилось во время разработки и также после завершения разработки. В ходе нее было выявлено огромное количество проблем, связанных с работой с файлами.

Описание программы

1. Приветственное меню.

После запуска программы появляется окно с приветствием игрока и предоставлением возможности ввести свое игровое имя. Оно вызывается в Form1.cs с помощью функции `Microsoft.VisualBasic.Interaction.InputBox`.

2. Отрисовка игрового поля и фигур.

После того, как игрок записал свое игровое имя в файле MapController.cs происходит отрисовка игрового поля, а отрисовка фигур происходит в файле Shape.cs.

3. Игровой процесс

Во время игры файл MapController.cs постоянно проверяет расположение фигур на игровом поле, количество сожжённых линий и пользовательский счет, который постоянно обновляется в зависимости от количества сожжённых линий.

4. Таблица лидеров

После того как игрок запишет свое имя в приветственное поле, оно сразу считается функцией `StreamWriter` в файле Form1.cs, таким же образом записывается счет пользователя, когда игрок проиграет, та же функция считает его счет и запишет в текстовый файл. Далее функция `StreamReader` считывает данные из текстовых файлов `names.txt` и `score.txt` в файле Sortirovka.cs. После считывания данных в файл происходит сортировка по «ключам» и «значениям», где в роли «ключей» выступает счет игрока, а в роли «значений» - имя пользователя.

Руководство пользователя

Программа Tetris предназначена для игры в тетрис. Она имеет стандартный интерфейс с подсказками по управлению.

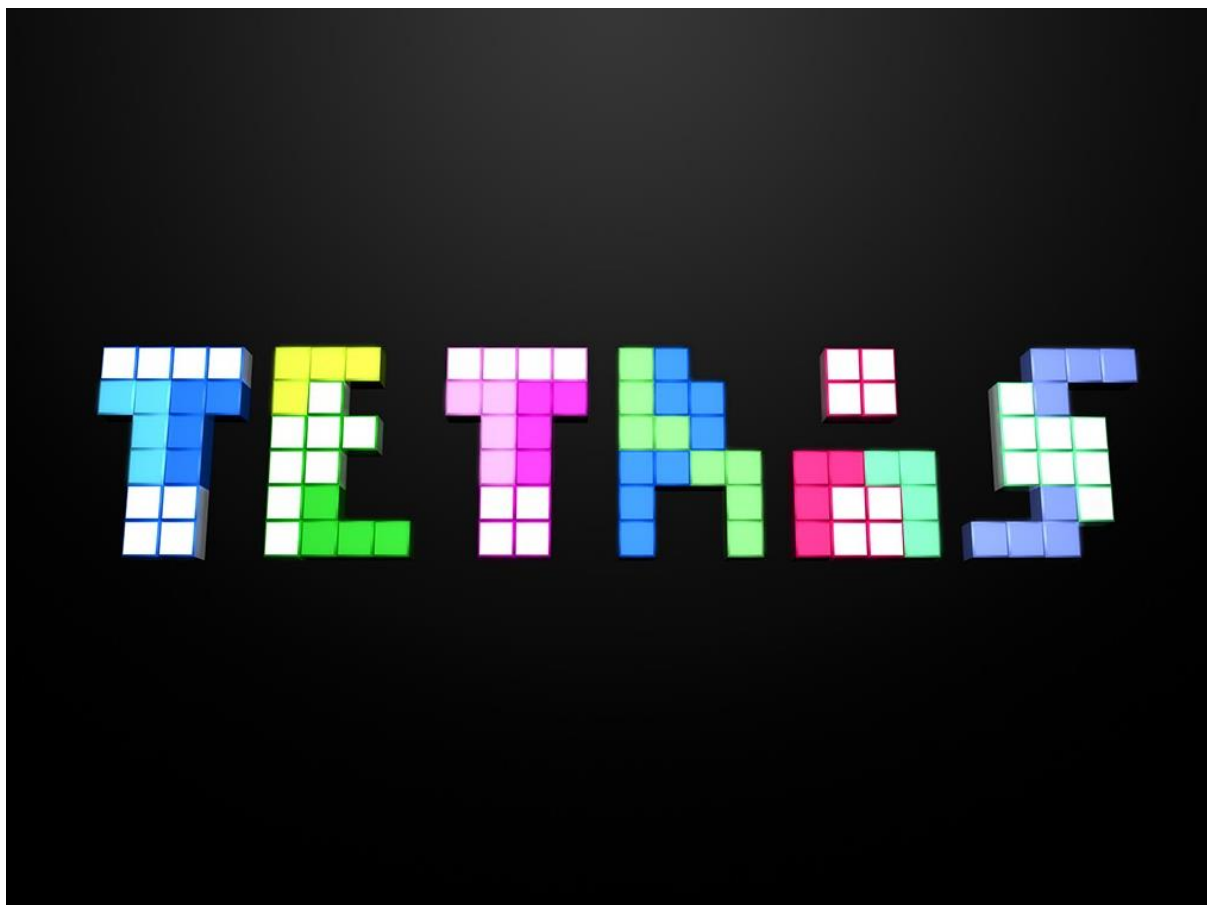


Рисунок 1.1 – Заставка

После заставки появляется окошко для ввода имени игрока.

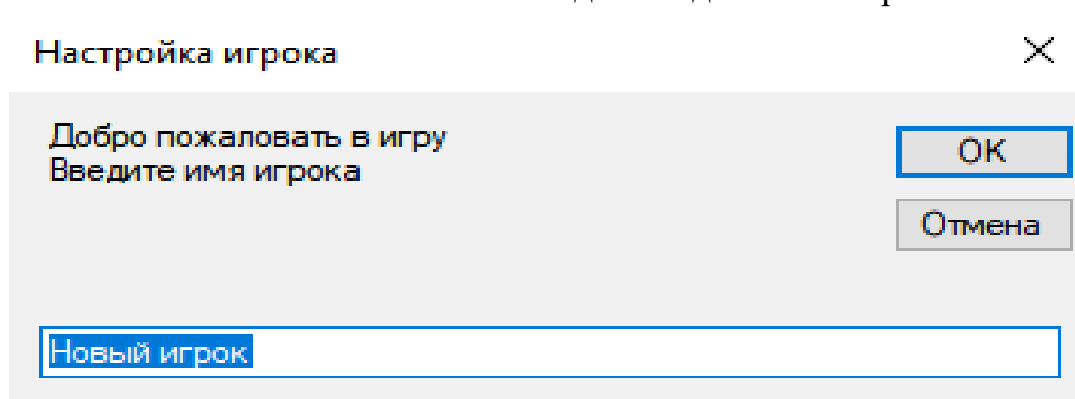


Рисунок 1.2 – Ввод имени игрока

После ввода имени пользователя появляется игровое поле и начинается игра.



Рисунок 1.3 – игровое поле

Меню выполнено в виде раскрывающегося списка.

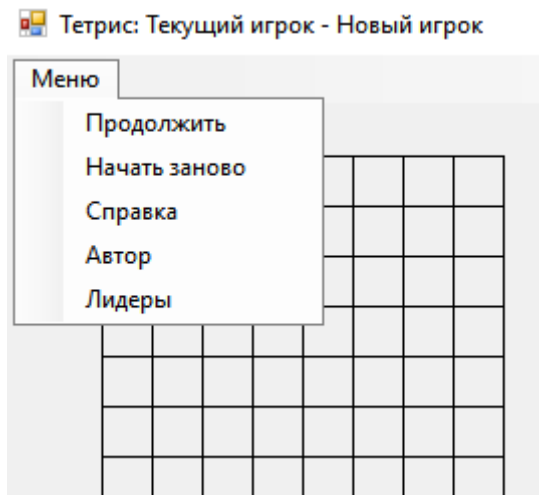


Рисунок 1.4 – меню

В меню находится информация об авторе программы.

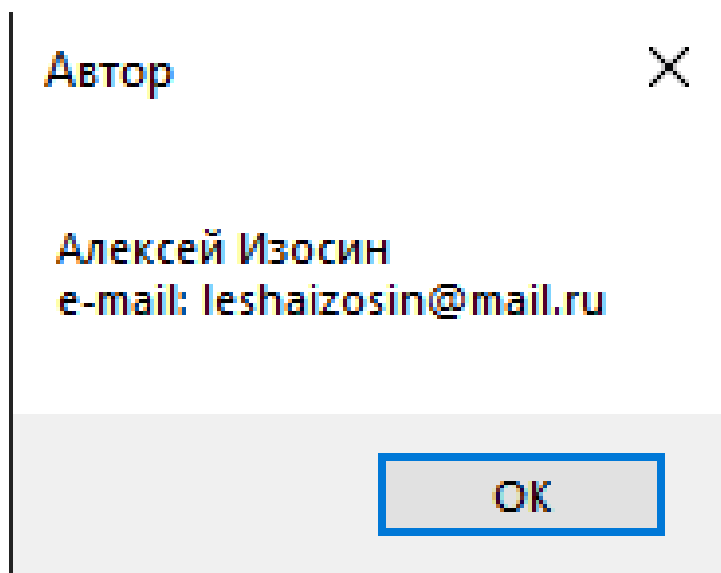


Рисунок 1.5 – сведения об авторе программы

И на последней строчке в меню расположились лидеры.

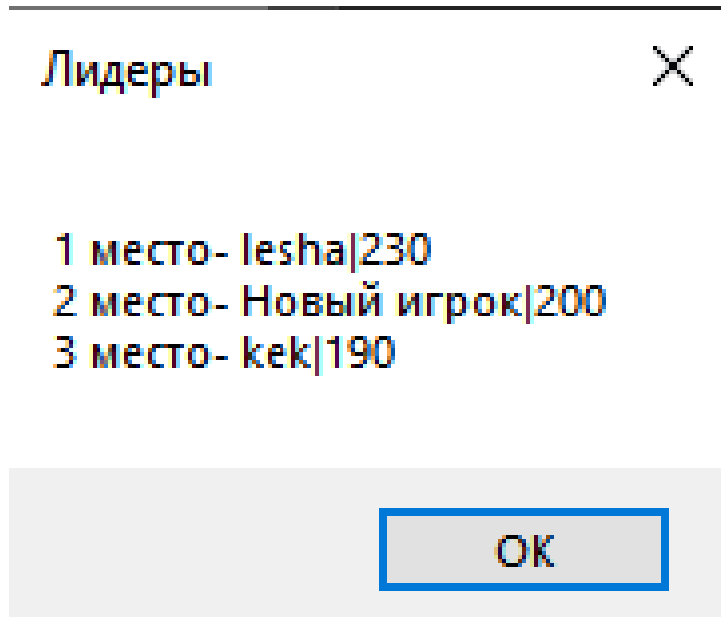


Рисунок 1.6 – Лидеры игры

Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульных программ. Были освоены способы обработки событий от клавиатуры. Усвоены принципы реализации интерфейса. Изучены возможности среды программирования VisualStudio 2019. Получены навыки отладки и тестирования программ.

В рамках выполнения курсовой работы была написана программа (игра) «тетрис». Программа не требует большого количества системных ресурсов, обеспечивает быструю обработку данных и показ результата пользователю по требованию.

Программу можно улучшить, оптимизировав код для более быстрого выполнения функций.

Список используемых источников

- 1) METANIT.COM
- 2) CyberForum.ru
- 3) docs.microsoft.com
- 4) programforyou.ru

Приложение А. Иерархическая структура программы

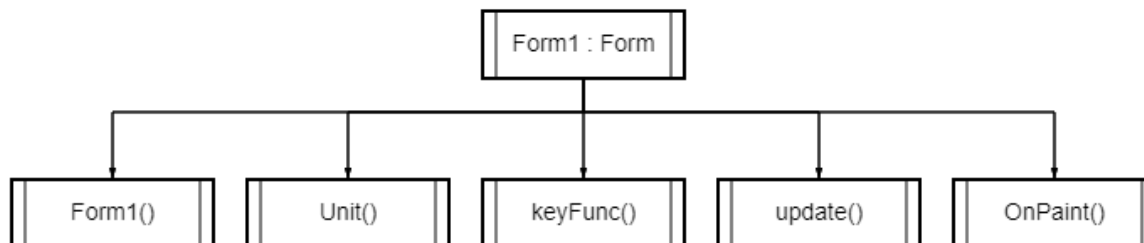


Рисунок 2 — Иерархическая структура программы

Приложение В. Схема данных программы

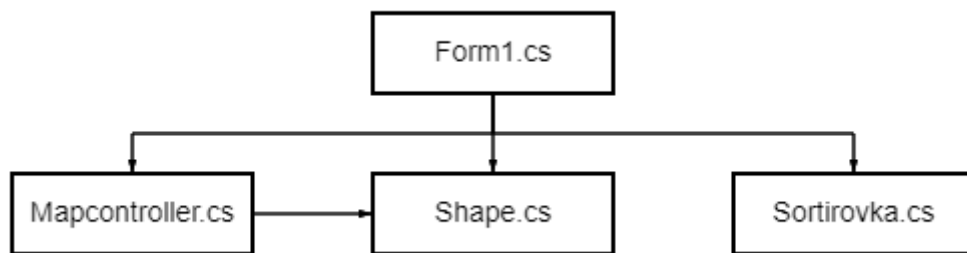


Рисунок 3 — Схема данных программы

Приложение С. Листинг программы

Form1.cs

```
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;
using Tetris.Controllers;
using System.Collections;

namespace Tetris
{
    public partial class Form1 : Form
    {
        string playerName;
        public Form1()
        {
            InitializeComponent();
            playerName =
Microsoft.VisualBasic.Interaction.InputBox("Добро пожаловать в
игру\nВведите имя игрока", "Настройка игрока", "Новый игрок");
            if (playerName == "")
            {
                playerName = "Новый игрок";
            }
            this.KeyUp += new KeyEventHandler(keyFunc);

            string path =
@"C:\Users\lesha\source\repos\Tetris\Tetris\names.txt";
            using (StreamWriter writer = new
StreamWriter(path, true))
            {
                writer.WriteLineAsync(playerName);
            }
            Init();
        }

        public void Init()
        {
            this.Text = "Тетрис: Текущий игрок - " +
playerName;
            MapController.size = 25;
            MapController.score = 0;
            MapController.linesRemoved = 0;
            MapController.currentShape = new Shape(3, 0);
            MapController.Interval = 300;
        }
    }
}
```

```

        label1.Text = "Score: " + MapController.score;
        label2.Text = "Lines: " +
MapController.linesRemoved;

        timer1.Interval = MapController.Interval;
        timer1.Tick += new EventHandler(update);
        timer1.Start();

        Invalidate();
    }

    private void keyFunc(object sender, KeyEventArgs e)
    {
        switch (e.KeyCode)
        {
            case Keys.A:

                if (!MapController.IsIntersects())
                {
                    MapController.ResetArea();

MapController.currentShape.RotateShape();
                    MapController.Merge();
                    Invalidate();
                }
                break;
            case Keys.Space:
                timer1.Interval = 10;
                break;
            case Keys.Right:
                if (!MapController.CollideHor(1))
                {
                    MapController.ResetArea();

MapController.currentShape.MoveRight();
                    MapController.Merge();
                    Invalidate();
                }
                break;
            case Keys.Left:
                if (!MapController.CollideHor(-1))
                {
                    MapController.ResetArea();
                    MapController.currentShape.MoveLeft();
                    MapController.Merge();
                    Invalidate();
                }
                break;
        }
    }
}

```

```

private void update(object sender, EventArgs e)
{
    MapController.ResetArea();
    if (!MapController.Collide())
    {
        MapController.currentShape.MoveDown();
    }
    else
    {
        MapController.Merge();
        MapController.SliceMap(label1, label2);
        timer1.Interval = MapController.Interval;
        MapController.currentShape.ResetShape(3, 0);
        if (MapController.Collide())
        {
            MapController.ClearMap();
            timer1.Tick -= new EventHandler(update);
            timer1.Stop();
            MessageBox.Show("Ваш результат: " +
MapController.score);

            string cscore =
MapController.score.ToString();
            string path =
@"C:\Users\lesha\source\repos\Tetris\Tetris\score.txt";
            using (StreamWriter writer = new
StreamWriter(path, true))
            {
                writer.WriteLineAsync(cscore);
            }

            Close();

            Init();
        }
    }
    MapController.Merge();
    Invalidate();
}

private void OnPaint(object sender, PaintEventArgs e)
{
    MapController.DrawGrid(e.Graphics);
    MapController.DrawMap(e.Graphics);
    MapController.ShowNextShape(e.Graphics);
}

private void OnPauseButtonClick(object sender,
EventArgs e)
{
    var pressedButton = sender as ToolStripMenuItem;
    if (timer1.Enabled)
    {

```

```

        pressedButton.Text = "Продолжить";
        timer1.Stop();
    }
    else
    {
        pressedButton.Text = "Пауза";
        timer1.Start();
    }
}

private void OnAgainButtonClick(object sender,
EventArgs e)
{
    timer1.Tick -= new EventHandler(update);
    timer1.Stop();
    MapController.ClearMap();
    Init();
}

private void OnInfoPressed(object sender, EventArgs e)
{
    string infoString = "";
    infoString = "Для управление фигурами используйте
стрелочку влево/вправо.\n";
    infoString += "Чтобы ускорить падение фигуры -
нажмите 'Пробел'.\n";
    infoString += "Для поворота фигуры используйте
'A'.\n";
    MessageBox.Show(infoString, "Справка");
}

private void авторToolStripMenuItem_Click(object
sender, EventArgs e)
{
    string infoString = "";
    infoString = "Алексей Изосин\n";
    infoString += "e-mail: leshaizosin@mail.ru\n";
    MessageBox.Show(infoString, "Автор");
}

private void лидерыToolStripMenuItem_Click(object
sender, EventArgs e)
{
    string infoString = "";
    infoString = "1 место- " +
Sortirovk.myValuesNames[0] + "|" +
Sortirovk.myKeysScore[0] + "\n";
    infoString += "2 место- " +
Sortirovk.myValuesNames[1] + "|" +
Sortirovk.myKeysScore[1] + "\n";
    infoString += "3 место-
" + Sortirovk.myValuesNames[2] + "|" +
Sortirovk.myKeysScore[2] + "\n";
    MessageBox.Show(infoString, "Лидеры");
}

```

```

    }
}
}

```

MapController.cs

```

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;

namespace Tetris.Controllers
{
    public static class MapController
    {
        public static Shape currentShape;
        public static int size;
        public static int[,] map = new int[16, 8];
        public static int linesRemoved;
        public static int score;
        public static int Interval;
        public static void ShowNextShape(Graphics e)
        {
            for (int i = 0; i < currentShape.sizeNextMatrix;
i++)
            {
                for (int j = 0; j <
currentShape.sizeNextMatrix; j++)
                {
                    if (currentShape.nextMatrix[i, j] == 1)
                    {
                        e.FillRectangle(Brushes.Red, new
Rectangle(300 + j * (size) + 1, 50 + i * (size) + 1, size - 1,
size - 1));
                    }
                    if (currentShape.nextMatrix[i, j] == 2)
                    {
                        e.FillRectangle(Brushes.Yellow, new
Rectangle(300 + j * (size) + 1, 50 + i * (size) + 1, size - 1,
size - 1));
                    }
                    if (currentShape.nextMatrix[i, j] == 3)
                    {
                        e.FillRectangle(Brushes.Green, new
Rectangle(300 + j * (size) + 1, 50 + i * (size) + 1, size - 1,
size - 1));
                    }
                    if (currentShape.nextMatrix[i, j] == 4)
                    {

```

```

        e.FillRectangle(Brushes.Blue, new
Rectangle(300 + j * (size) + 1, 50 + i * (size) + 1, size - 1,
size - 1));
    }
    if (currentShape.nextMatrix[i, j] == 5)
    {
        e.FillRectangle(Brushes.Purple, new
Rectangle(300 + j * (size) + 1, 50 + i * (size) + 1, size - 1,
size - 1));
    }
}
}

public static void ClearMap()
{
    for (int i = 0; i < 16; i++)
    {
        for (int j = 0; j < 8; j++)
        {
            map[i, j] = 0;
        }
    }
}

public static void DrawMap(Graphics e)
{
    for (int i = 0; i < 16; i++)
    {
        for (int j = 0; j < 8; j++)
        {
            if (map[i, j] == 1)
            {
                e.FillRectangle(Brushes.Red, new
Rectangle(50 + j * (size) + 1, 50 + i * (size) + 1, size - 1,
size - 1));
            }
            if (map[i, j] == 2)
            {
                e.FillRectangle(Brushes.Yellow, new
Rectangle(50 + j * (size) + 1, 50 + i * (size) + 1, size - 1,
size - 1));
            }
            if (map[i, j] == 3)
            {
                e.FillRectangle(Brushes.Green, new
Rectangle(50 + j * (size) + 1, 50 + i * (size) + 1, size - 1,
size - 1));
            }
            if (map[i, j] == 4)
            {
                e.FillRectangle(Brushes.Blue, new
Rectangle(50 + j * (size) + 1, 50 + i * (size) + 1, size - 1,
size - 1));
            }
        }
    }
}

```

```

        }
        if (map[i, j] == 5)
        {
            e.FillRectangle(Brushes.Purple, new
Rectangle(50 + j * (size) + 1, 50 + i * (size) + 1, size - 1,
size - 1));
        }
    }
}

public static void DrawGrid(Graphics g)
{
    for (int i = 0; i <= 16; i++)
    {
        g.DrawLine(Pens.Black, new Point(50, 50 + i *
size), new Point(50 + 8 * size, 50 + i * size));
    }
    for (int i = 0; i <= 8; i++)
    {
        g.DrawLine(Pens.Black, new Point(50 + i *
size, 50), new Point(50 + i * size, 50 + 16 * size));
    }
}

public static void SliceMap(Label label1, Label label2)
{
    int count = 0;
    int curRemovedLines = 0;
    for (int i = 0; i < 16; i++)
    {
        count = 0;
        for (int j = 0; j < 8; j++)
        {
            if (map[i, j] != 0)
                count++;
        }
        if (count == 8)
        {
            curRemovedLines++;
            for (int k = i; k >= 1; k--)
            {
                for (int o = 0; o < 8; o++)
                {
                    map[k, o] = map[k - 1, o];
                }
            }
        }
    }
    for (int i = 0; i < curRemovedLines; i++)
    {
        score += 10 * (i + 1);
    }
    linesRemoved += curRemovedLines;
}

```

```

        if (linesRemoved % 5 == 0)
        {
            if (Interval > 60)
                Interval -= 10;
        }

        label1.Text = "Score: " + score;
        label2.Text = "Lines: " + linesRemoved;
    }

    public static bool IsIntersects()
    {
        for (int i = currentShape.y; i < currentShape.y +
currentShape.sizeMatrix; i++)
        {
            for (int j = currentShape.x; j <
currentShape.x + currentShape.sizeMatrix; j++)
            {
                if (j >= 0 && j <= 7)
                {
                    if (map[i, j] != 0 &&
currentShape.matrix[i - currentShape.y, j - currentShape.x] ==
0)

                        return true;
                }
            }
        }
        return false;
    }

    public static void Merge()
    {
        for (int i = currentShape.y; i < currentShape.y +
currentShape.sizeMatrix; i++)
        {
            for (int j = currentShape.x; j <
currentShape.x + currentShape.sizeMatrix; j++)
            {
                if (currentShape.matrix[i -
currentShape.y, j - currentShape.x] != 0)
                    map[i, j] = currentShape.matrix[i -
currentShape.y, j - currentShape.x];
            }
        }
    }

    public static bool Collide()
    {
        for (int i = currentShape.y +
currentShape.sizeMatrix - 1; i >= currentShape.y; i--)
        {

```

```

        for (int j = currentShape.x; j <
currentShape.x + currentShape.sizeMatrix; j++)
        {
            if (currentShape.matrix[i -
currentShape.y, j - currentShape.x] != 0)
            {
                if (i + 1 == 16)
                    return true;
                if (map[i + 1, j] != 0)
                {
                    return true;
                }
            }
        }
    }
    return false;
}

public static bool CollideHor(int dir)
{
    for (int i = currentShape.y; i < currentShape.y +
currentShape.sizeMatrix; i++)
    {
        for (int j = currentShape.x; j <
currentShape.x + currentShape.sizeMatrix; j++)
        {
            if (currentShape.matrix[i -
currentShape.y, j - currentShape.x] != 0)
            {
                if (j + 1 * dir > 7 || j + 1 * dir <
0)

                    return true;

                if (map[i, j + 1 * dir] != 0)
                {
                    if (j - currentShape.x + 1 * dir
>= currentShape.sizeMatrix || j - currentShape.x + 1 * dir <
0)

                        {
                            return true;
                        }
                    if (currentShape.matrix[i -
currentShape.y, j - currentShape.x + 1 * dir] == 0)
                        return true;
                }
            }
        }
    }
    return false;
}

public static void ResetArea()
{

```

```

        for (int i = currentShape.y; i < currentShape.y +
currentShape.sizeMatrix; i++)
        {
            for (int j = currentShape.x; j <
currentShape.x + currentShape.sizeMatrix; j++)
            {
                if (i >= 0 && j >= 0 && i < 16 && j < 8)
                {
                    if (currentShape.matrix[i -
currentShape.y, j - currentShape.x] != 0)
                    {
                        map[i, j] = 0;
                    }
                }
            }
        }
    }
}

```

Shape.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Tetris
{
    public class Shape
    {
        public int x;
        public int y;
        public int[,] matrix;
        public int[,] nextMatrix;
        public int sizeMatrix;
        public int sizeNextMatrix;

        public int[,] tetr1 = new int[4, 4]{
            {0,0,1,0 },
            {0,0,1,0 },
            {0,0,1,0 },
            {0,0,1,0 },
        };

        public int[,] tetr2 = new int[3, 3]{
            {0,2,0 },
            {0,2,2 },
            {0,0,2 },
        };
    };
}

```

```

public int[,] tetr3 = new int[3, 3]{
    {0,0,0 },
    {3,3,3 },
    {0,3,0 },
};

public int[,] tetr4 = new int[3, 3]{
    { 4,0,0 },
    {4,0,0 },
    {4,4,0 },
};
public int[,] tetr5 = new int[2, 2]{
    { 5,5 },
    {5,5 },
};

public Shape(int _x,int _y)
{
    x = _x;
    y = _y;
    matrix = GenerateMatrix();
    sizeMatrix = (int)Math.Sqrt(matrix.Length);
    nextMatrix = GenerateMatrix();
    sizeNextMatrix =
(int)Math.Sqrt(nextMatrix.Length);
}

public void ResetShape(int _x, int _y)
{
    x = _x;
    y = _y;
    matrix = nextMatrix;
    sizeMatrix = (int)Math.Sqrt(matrix.Length);
    nextMatrix = GenerateMatrix();
    sizeNextMatrix =
(int)Math.Sqrt(nextMatrix.Length);
}

public int[,] GenerateMatrix()
{
    int[,] _matrix = tetr1;
    Random r = new Random();
    switch (r.Next(1, 6))
    {
        case 1:
            _matrix = tetr1;
            break;
        case 2:
            _matrix = tetr2;
            break;
        case 3:
            _matrix = tetr3;
            break;
    }
}

```

```

        case 4:
            _matrix = tetr4;
            break;
        case 5:
            _matrix = tetr5;
            break;
    }
    return _matrix;
}

public void RotateShape()
{
    int[,] tempMatrix = new
int[sizeMatrix,sizeMatrix];
    for(int i = 0; i < sizeMatrix; i++)
    {
        for (int j = 0; j < sizeMatrix; j++)
        {
            tempMatrix[i, j] = matrix[j, (sizeMatrix -
1) - i];
        }
    }
    matrix = tempMatrix;
    int offset1 = (8 - (x + sizeMatrix));
    if (offset1 < 0)
    {
        for (int i = 0; i < Math.Abs(offset1); i++)
            MoveLeft();
    }

    if (x < 0)
    {
        for (int i = 0; i < Math.Abs(x)+1; i++)
            MoveRight();
    }
}

public void MoveDown()
{
    y++;
}
public void MoveRight()
{
    x++;
}
public void MoveLeft()
{
    x--;
}
}
}

```

Sortirovka.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Collections;
namespace Tetris
{
    class Sortirovka
    {
        string path =
@"C:\Users\lesha\source\repos\Tetris\Tetris\names.txt";
        string path1 =
@"C:\Users\lesha\source\repos\Tetris\Tetris\score.txt";
        public string[] myKeysScore = new string[20];
        public string[] myValuesNames = new string[20];

        public class Sort : IComparer
        {
            int IComparer.Compare(Object x, Object y)
            {
                return ((new
CaseInsensitiveComparer()).Compare(y, x));
            }
        }

        public void main()
        {
            IComparer myComparer = new Sort();
            using (StreamReader reader = new
StreamReader(path))
            {
                string line;
                int g;
                while ((line = reader.ReadLine()) != null)
                {
                    for (g = 0; g < 20; g++)
                    {
                        myValuesNames[g] = line;
                    }
                }
            }
            using (StreamReader reader = new
StreamReader(path1))
            {
                string line;
                int g;
                while ((line = reader.ReadLine()) != null)
```

```

        {
            for (g = 0; g < 20; g++)
            {
                myKeysScore[g] = line;
            }
        }
        Array.Sort(myKeysScore, myValuesNames,
myComparer);
        PrintKeysAndValues(myKeysScore, myValuesNames);
    }

    public void PrintKeysAndValues(String[] myKeysScore,
String[] myValuesNames)
    {
        for (int i = 0; i < myKeysScore.Length; i++)
        {
            Console.WriteLine("    {0,-10}: {1}",
myKeysScore[i], myValuesNames[i]);
        }
    }
}

```

Form2.cs

```

using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Tetris
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }
    }
}

```

Приложение D. Схемы функций программы

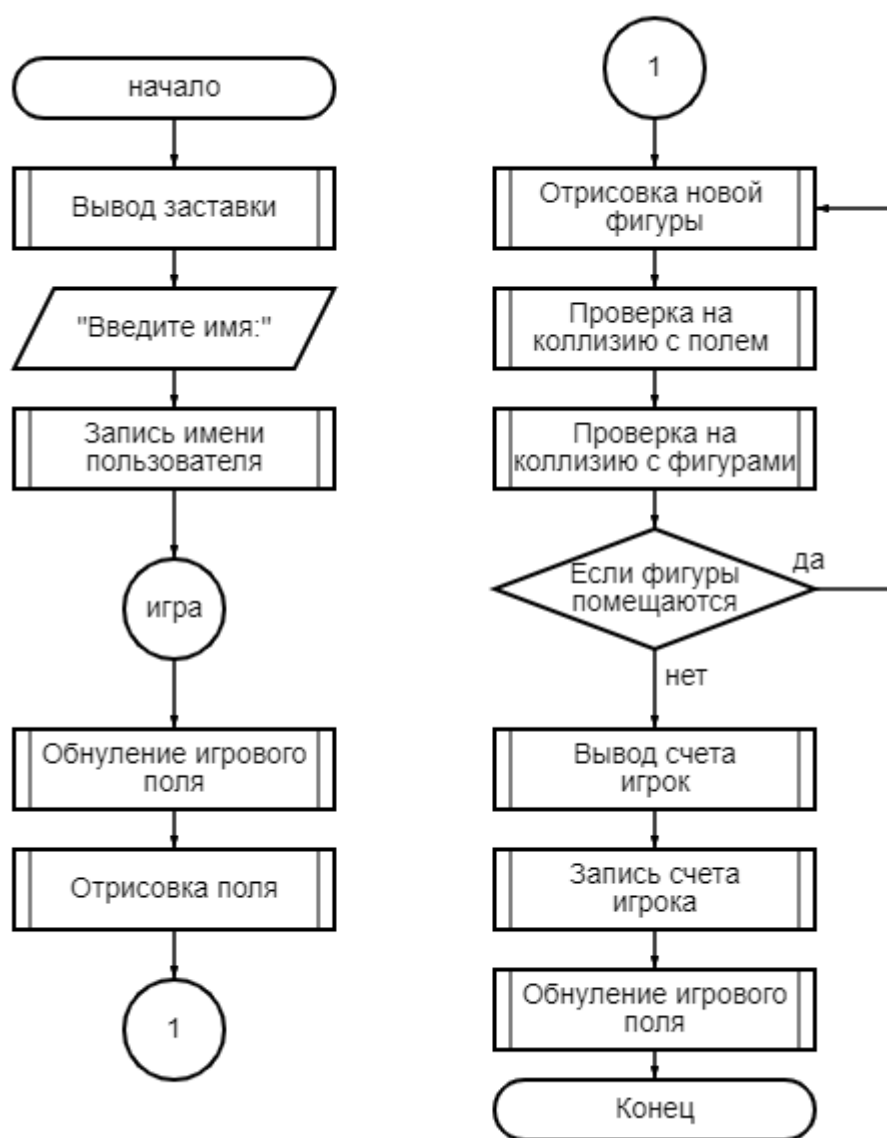


Рисунок 4.1 – Form1:Form

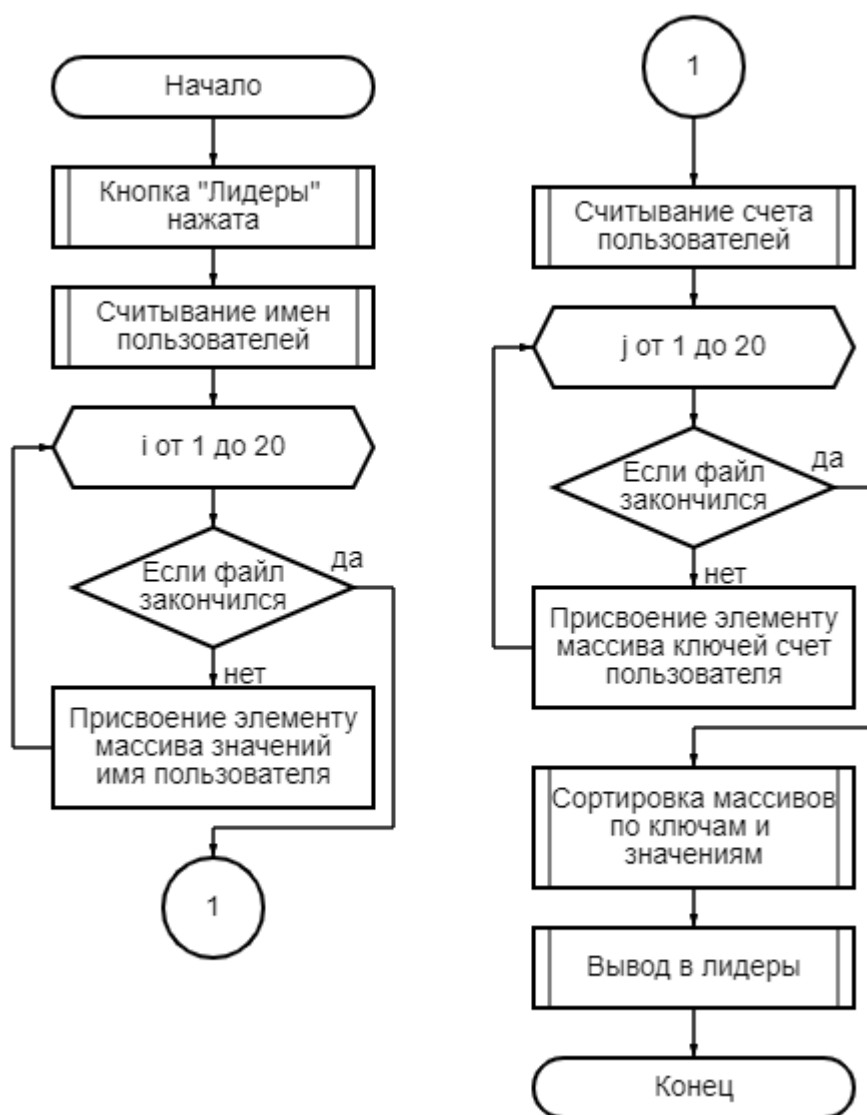


Рисунок 4.4 – Sortirovka.cs

Приложение Е. Схема взаимодействия программы

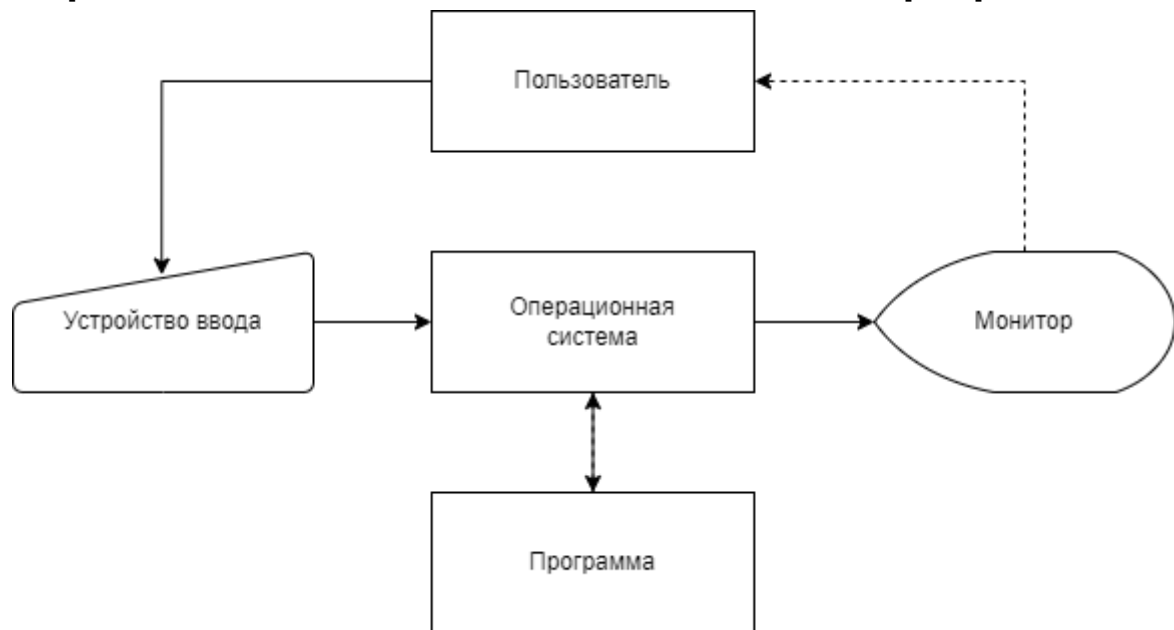


Рисунок 6 – Схема взаимодействия программы