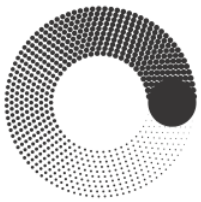
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

КУРСОВОЙ ПРОЕКТ

Дисциплина: Технологии прикладного программирования

Тема: Игра «Судоку»

Выполнил(а): студент(ка) группы 221-375

Орлова Е.А.

Дата, подпись 16.06.23 _____
(Дата) (Подпись)

Проверил: _____
(Фамилия И.О., степень, звание)

Дата, подпись _____ _____
(Дата) (Подпись)

Замечания: _____

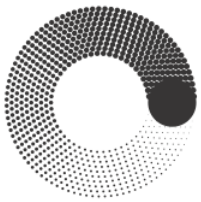
_____

_____

Москва

2023

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий
Кафедра Информатики и информационных технологий

_____ ***Утверждаю***
Зав. кафедрой Е.В. Булатников
« ___» _____ 2023 г.

ЗАДАНИЕ
на курсовое проектирование

Направление 09.03.02 «Информационные системы и технологии»

группа 221-375

дисциплина: Технологии прикладного программирования

Студент Орлова Екатерина Артемьевна

1. **Тема проекта** Игра «Судоку»
2. **Срок сдачи студентом законченного проекта** июнь 2023 года
3. **Исходные данные к проекту** Изучения языка C#, изучение Windows Forms, формирование пользовательского интерфейса, синтезирование алгоритма игры, реализация, выполнение проверки правильности игрового результата.
4. **Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)** Изучение игрового процесса, изучение алгоритмов игр на примере простой игры «Судоку»
5. **Перечень графического материала (с точным указанием обязательных чертежей)**
   листинг программы, скриншоты этапов программирования

6. **Литература и прочие материалы, рекомендуемые студенту для изучения**

1. Joe Mayo, «C# Cookbook»

2. Бессмертный, И. А. Системы искусственного интеллекта : учеб. пособие для СПО / И. А. Бессмертный. — 2-е изд., испр. и доп. — М. : Издательство Юрайт, 2018. — 130 с., 3. Гниденко, И. Г. Технология разработки программного обеспечения : учеб. пособие для СПО / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — М. : Издательство Юрайт, 2017. — 235 с., 4. Гордеев, С. И. Организация баз данных в 2 ч. Часть 2 : учебник для вузов / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — М. : Издательство Юрайт, 2019. — 501 с., 5. Жмудь, В. А. Моделирование замкнутых систем автоматического управления : учеб. пособие для академического бакалавриата / В. А. Жмудь. — 2-е изд., испр. и доп. — М. : Издательство Юрайт, 2019. — 128 с.

7. **Дата выдачи задания** февраль 2023 года

Руководитель проекта *Арсентьев Дмитрий Андреевич*

Задание принял студент *Орлова Екатерина Артемьевна*

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Судóку (яп. 数独 су:доку) - популярная головоломка-пазл с числами. В переводе с японского «су» - «цифра», «доку» - «стоящая отдельно». Иногда судоку называют «магическим квадратом», что, в общем-то, не верно, так как судоку является латинским квадратом 9-го порядка. Судоку активно публикуют газеты и журналы разных стран мира, сборники судоку издаются большими тиражами. Решение судоку - популярный вид досуга. Игровое поле представляет собой квадрат размером 9x9, разделённый на меньшие квадраты со стороной в 3 клетки. Таким образом, всё игровое поле состоит из 81 клетки. В них, уже в начале игры, стоят некоторые числа (от 1 до 9), так как незаполненное игровое поле не имеет смысла. От количества пустых клеток зависит сложность данной игры.

История судоку

Все знают судоку как японское развлечение, но на самом деле Япония только приложила руку к глобальному распространению этих головоломок, а действительным прообразом судоку можно назвать головоломки, публикуемые с 1892 года во французской газете Le Siècle. Эти «судоку» ещё не приобрели современный вид, но уже предполагали тот же механизм решения. Далее «судоку» продолжила своё путешествие по миру и вот в 1979 году Говард Ганс из Индианы публикует в «Dell Magazines» головоломку «Number Place» - это можно считать одним из первых выпусков современных судоку. Далее судоку была представлена японской публике японским же издателем Nikoli в своём ежемесячнике «Montly Nikolist» в апреле 1984 года и с этого момента можно начинать отсчёт судоку-мании. В конце 2004 года британская «Times» стала печатать судоку на своих страницах, чем прославила эту головоломку на всю Европу. Сегодня судоку - обязательный компонент многих газет. Среди них много изданий с многомиллионными тиражами, например немецкая газета «Die Zeit», австрийский «Der Standard». Также публикует судоку российская газета «Труд».

Виды судоку

Судоку различаются по уровню сложности в зависимости от размера квадрата - для профессионалов существуют судоку 15x15 и 16x16 клеток. Существуют судоку, в которых указываются не отдельные цифры для каждой клетки, а сумма цифр в той или иной группе клеток: судоку разбивается на несколько прямоугольных блоков разных размеров, а внутри каждого поля указывается сумма цифр, входящих в него. Ещё одним вариантом головоломки является судоку, у которой внутренние блоки не квадратные, а произвольной формы, но при этом они, как правило, включают те же 9 клеток. Для маленьких любителей головоломок делаются судоку с полями 2x2 клетки, а для любителей судоку со стажем существуют цветные судоку, где нужно расставить цифры в каждом из цветных блоков по отдельности. Существуют также диагональные судоку, в которых нет мини-квадратов, а цифры не должны совпадать в каждой строке, каждом столбце и на двух максимальных диагоналях. Встречаются и комбинации всех этих видов судоку, например, судоку с мини-квадратами и двумя максимальными диагоналями. Из всего многообразия судоку каждый сможет выбрать головоломку для себя. Судоку - это отличная зарядка для мозгов.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

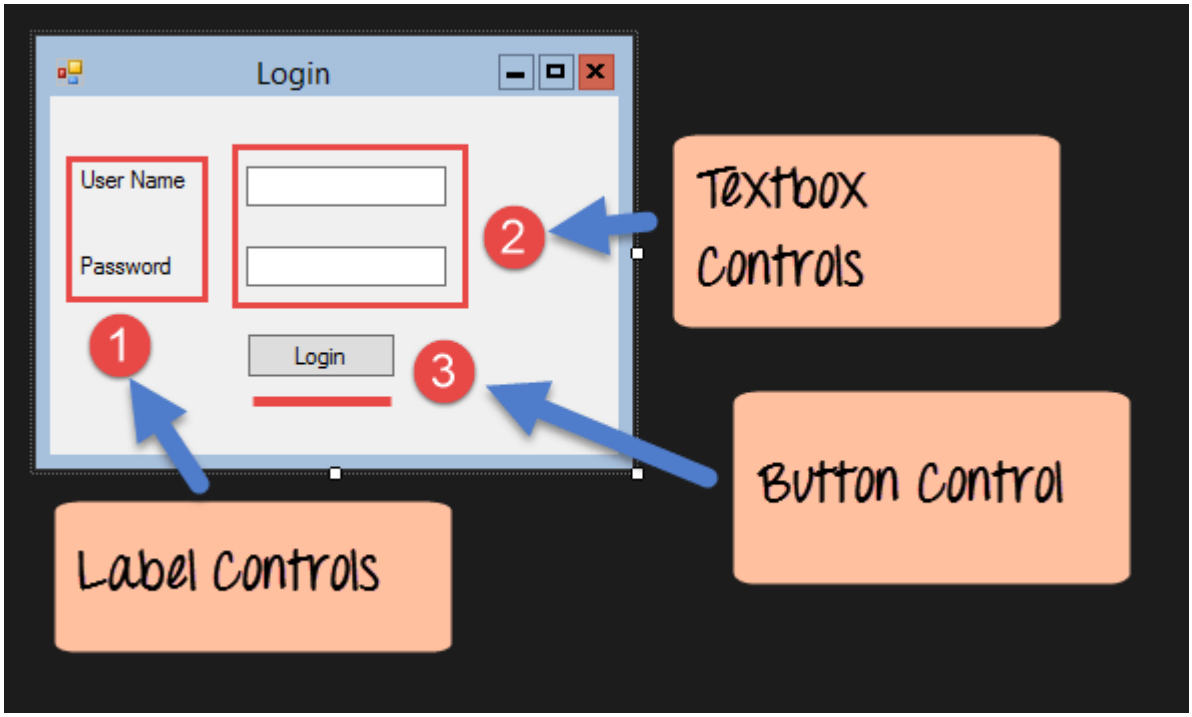
Windows Forms — это технология пользовательского интерфейса для .NET, представляющая собой набор управляемых библиотек, которые упрощают выполнение стандартных задач, таких как чтение из файловой системы и запись в нее. С помощью среды разработки, такой как Visual Studio, можно создавать интеллектуальные клиентские приложения Windows Forms, которые отображают информацию, запрашивают ввод пользователя и взаимодействуют с удаленными компьютерами по сети.

В Windows Forms форма — это визуальная поверхность, на которой выводится информация для пользователя. Обычно приложение Windows Forms строится путем добавления элементов управления в формы и создания кода для реагирования на действия пользователя, такие как щелчки мыши или нажатия клавиш. Элемент управления — это отдельный элемент пользовательского интерфейса, предназначенный для отображения или ввода данных.

При выполнении пользователем какого-либо действия с формой или одним из ее элементов управления создается событие. Приложение реагирует на эти события, как задано в коде, и обрабатывает события при их возникновении. В Windows Forms предусмотрено множество элементов управления, которые можно добавлять в формы. Например, элементы управления могут отображать текстовые поля, кнопки, раскрывающиеся списки, переключатели и даже веб-страницы. Если предусмотренные элементы управления не подходят для ваших целей, в Windows Forms можно создавать собственные пользовательские элементы управления с помощью класса UserControl.

Приложение Windows Forms работает на настольном компьютере. Приложение форм Windows обычно имеет набор элементов управления, таких как метки, текстовые поля, списки и т. Д.

Ниже приведен пример простого приложения для Windows. Он показывает простой экран входа в систему, который доступен пользователю. Пользователь введет необходимые учетные данные, а затем нажмет кнопку «Вход», чтобы продолжить.



ПРАКТИЧЕСКАЯ ЧАСТЬ

ЛИСТИНГ КОДА

```
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 Sudoku
{
    public partial class Form1 : Form
    {
        const int block_ = 3; // Размер блока
        const int butt_ = 50; // Размер кнопки
        public int[,] arr = new int[block_ * block_, block_ * block_]; // Игровое
поле
        public Button[,] buttons = new Button[block_ * block_, block_ * block_];

        public Form1()
        {
            InitializeComponent();
            GenerateArr();
        }

        public void GenerateArr() // Заполняем поле
        {
            for(int i = 0; i < block_ * block_; i++)
            {
                for(int j = 0; j < block_ * block_; j++)
                {
                    arr[i, j] = (i * block_ + i / block_ + j) % (block_ * block_) +
1;
                    buttons[i, j] = new Button();
                }
            }

            Random r = new Random();
            for(int i = 0; i < 40; i++)
            {
                ShuffleMap(r.Next(0, 5));
            }

            CreateBut();
            HideCells();
        }

        public void CreateBut() // Создаем кнопки
        {
            for (int i = 0; i < block_ * block_; i++)
            {
                for (int j = 0; j < block_ * block_; j++)
                {
                    Button button = new Button();

                    buttons[i, j] = button;
                    button.Size = new Size(butt_, butt_);
                    button.Text = arr[i, j].ToString();
                    button.Click += OnCellPressed;
                    button.Location = new Point(j * butt_, i * butt_);
```

```
                this.Controls.Add(button);
            }
        }
    }


//////////////////////////////////////////////////////////////Перетасовка////////

        public void MatrixTransposition() // Транспонирование
        {
            int[,] tArr = new int[block_ * block_, block_ * block_];
            for (int i = 0; i < block_ * block_; i++)
            {
                for (int j = 0; j < block_ * block_; j++)
                {
                    tArr[i, j] = arr[j, i];
                }
            }
            arr = tArr;
        }

        public void SwapRowsInBlock() // Смена строк в пределах одного блока
        {
            Random r = new Random();

            var block = r.Next(0, block_);
            var row1 = r.Next(0, block_);
            var line1 = block * block_ + row1;
            var row2 = r.Next(0, block_);

            while (row1 == row2)
                row2 = r.Next(0, block_);

            var line2 = block * block_ + row2;

            for (int i = 0; i < block_ * block_; i++)
            {
                var temp = arr[line1, i];
                arr[line1, i] = arr[line2, i];
                arr[line2, i] = temp;
            }
        }

        public void SwapColumnsInBlock() // Смена столбцов в пределах одного блока
        {
            Random r = new Random();

            var block = r.Next(0, block_);
            var column1 = r.Next(0, block_);
            var line1 = block * block_ + column1;
            var column2 = r.Next(0, block_);

            while (column1 == column2)
                column2 = r.Next(0, block_);

            var line2 = block * block_ + column2;

            for (int i = 0; i < block_ * block_; i++)
            {
                var temp = arr[i, line1];
                arr[i, line1] = arr[i, line2];
                arr[i, line2] = temp;
            }
        }
```

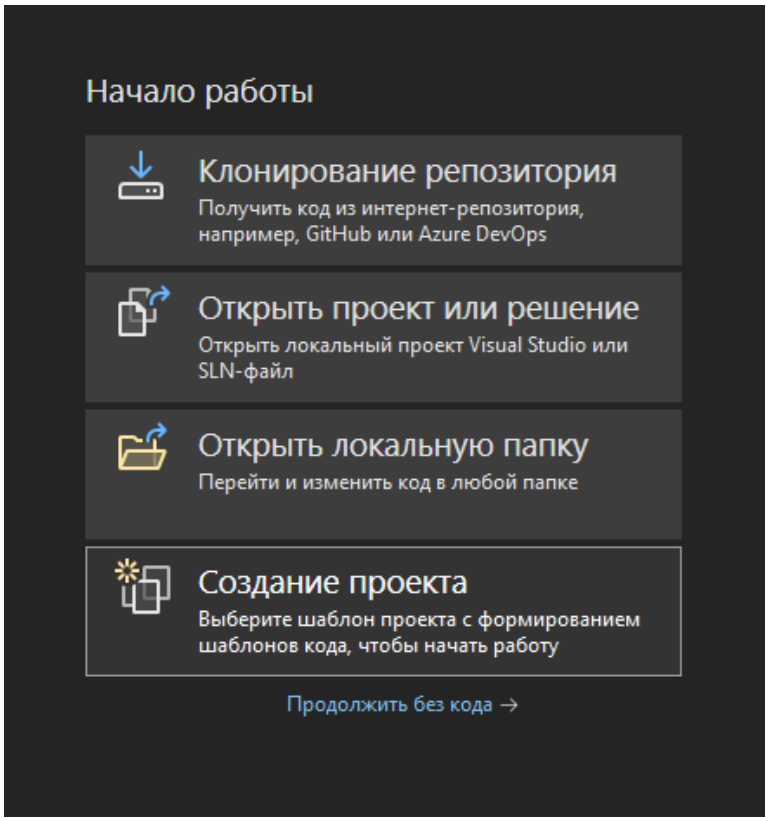
```
        public void SwapBlocksInRow() // Смена блоков по горизонтали
        {
            Random r = new Random();

            var block1 = r.Next(0, block_);
            var block2 = r.Next(0, block_);

            while (block1 == block2)
                block2 = r.Next(0, block_);

            block1 *= block_;
            block2 *= block_;

            for (int i = 0; i < block_ * block_; i++)
            {
                var k = block2;
                for (int j = block1; j < block1 + block_; j++)
                {
                    var temp = arr[j, i];
                    arr[j, i] = arr[k, i];
                    arr[k, i] = temp;
                    k++;
                }
            }
        }

        public void SwapBlocksInColumn()
        {
            Random r = new Random();

            var block1 = r.Next(0, block_);
            var block2 = r.Next(0, block_);

            while (block1 == block2)
                block2 = r.Next(0, block_);

            block1 *= block_;
            block2 *= block_;

            for (int i = 0; i < block_ * block_; i++)
            {
                var k = block2;
                for (int j = block1; j < block1 + block_; j++)
                {
                    var temp = arr[i, j];
                    arr[i, j] = arr[i, k];
                    arr[i, k] = temp;
                    k++;
                }
            }
        }

    ////////////////////////////////////////////////////////////////////////////////

        public void ShuffleMap(int i)
        {
            switch (i)
            {
                case 0:
                    MatrixTransposition();
                    break;
                case 1:
                    SwapRowsInBlock();
```

```
                break;
            case 2:
                SwapColumnsInBlock();
                break;
            case 3:
                SwapBlocksInRow();
                break;
            case 4:
                SwapBlocksInColumn();
                break;
            default:
                MatrixTransposition();
                break;
        }
    }

////////////////////////////////////////////////////////////////////////////////

    public void HideCells() // Кнопки для игры
    {
        int N = 40;
        Random r = new Random();
        while (N > 0)
        {
            for (int i = 0; i < block_ * block_; i++)
            {
                for (int j = 0; j < block_ * block_; j++)
                {
                    if (!string.IsNullOrEmpty(buttons[i, j].Text)){
                        int a = r.Next(0, 3);
                        buttons[i, j].Text = a == 0 ? "" : buttons[i, j].Text;
                        buttons[i, j].Enabled = a == 0 ? true : false;

                        if (a == 0)
                            N--;
                        if (N <= 0)
                            break;
                    }
                }
                if (N <= 0)
                    break;
            }
        }
    }


    public void OnCellPressed(object sender, EventArgs e)
    {
        Button pressedButton = sender as Button;
        string buttonText = pressedButton.Text;
        if (string.IsNullOrEmpty(buttonText))
        {
            pressedButton.Text = "1";
        }
        else
        {
            int num = int.Parse(buttonText);
            num++;
            if (num == 10)
                num = 1;
            pressedButton.Text = num.ToString();
        }

    }
```

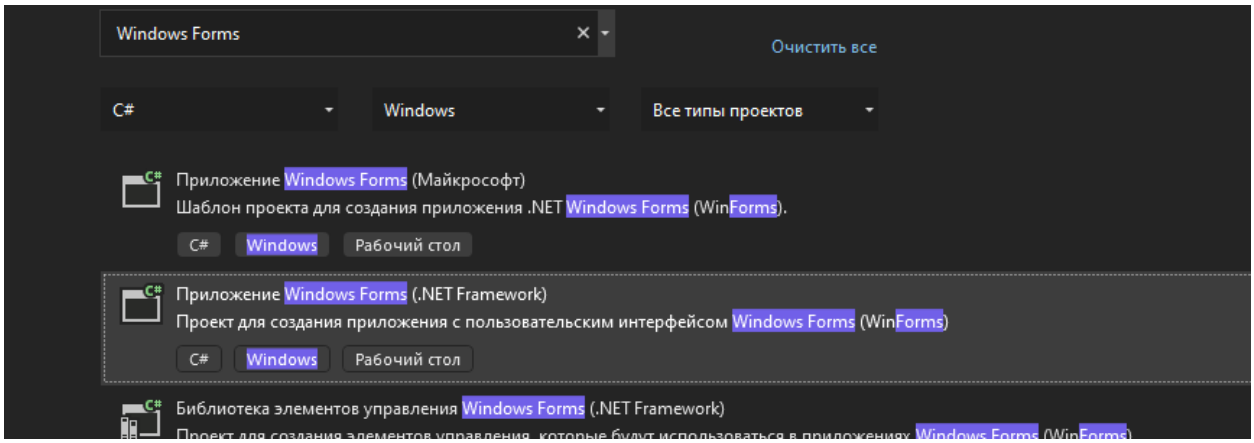
```
        private void button1_Click(object sender, EventArgs e)
        {
            for(int i = 0; i < block_ * block_; i++)
            {
                for(int j = 0; j < block_ * block_; j++)
                {
                    var btnText = buttons[i, j].Text;
                    if(btnText != arr[i, j].ToString())
                    {
                        MessageBox.Show("Обнаружены совпадения!\nРешение неверно");
                        return;
                    }
                }
            }
            MessageBox.Show("Все верно!");
            for(int i = 0; i < block_ * block_; i++)
            {
                for (int j = 0; j < block_ * block_; j++)
                {
                    this.Controls.Remove(buttons[i, j]);
                }
            }
            GenerateArr();
        }
    }
}
```

СКРИНШОТЫ ВЫПОЛНЕНИЯ

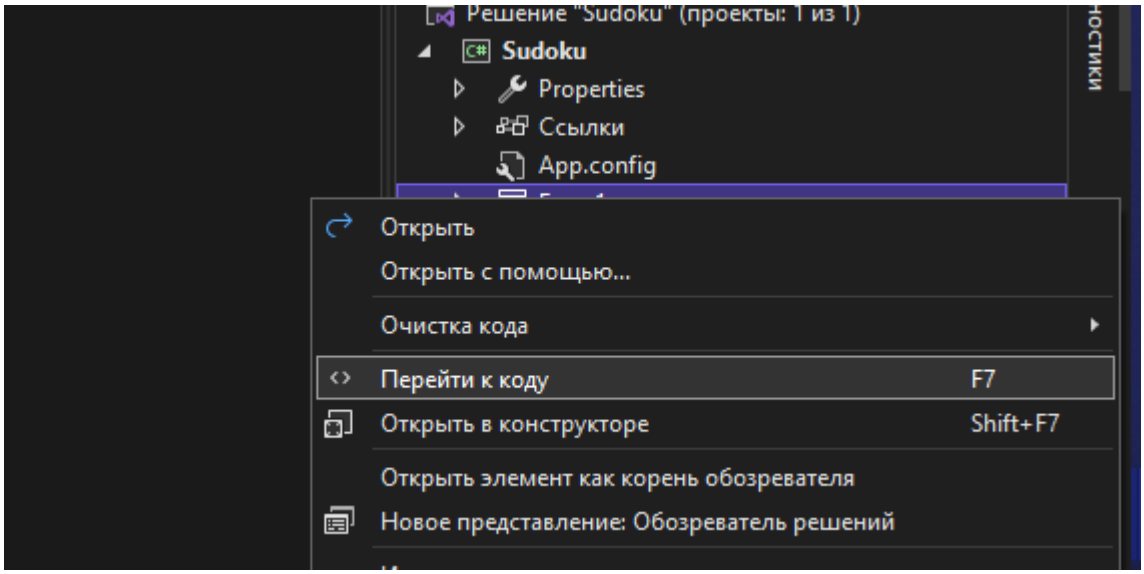
Создаем новый проект



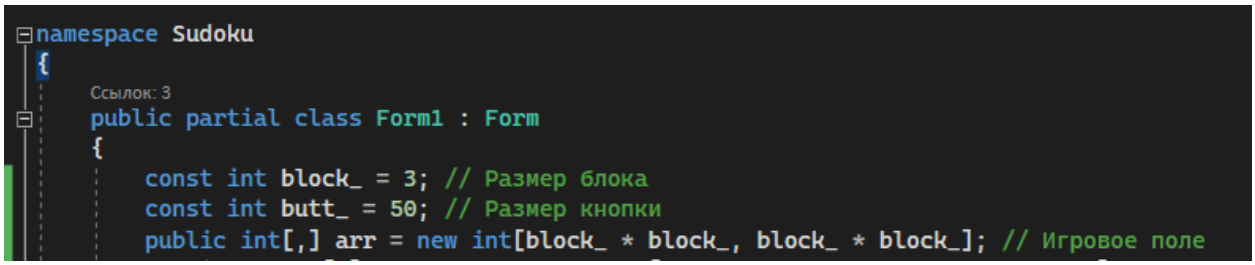
Выбираем приложение Windows Forms (.NET Framework)



В созданном приложении переходим к коду

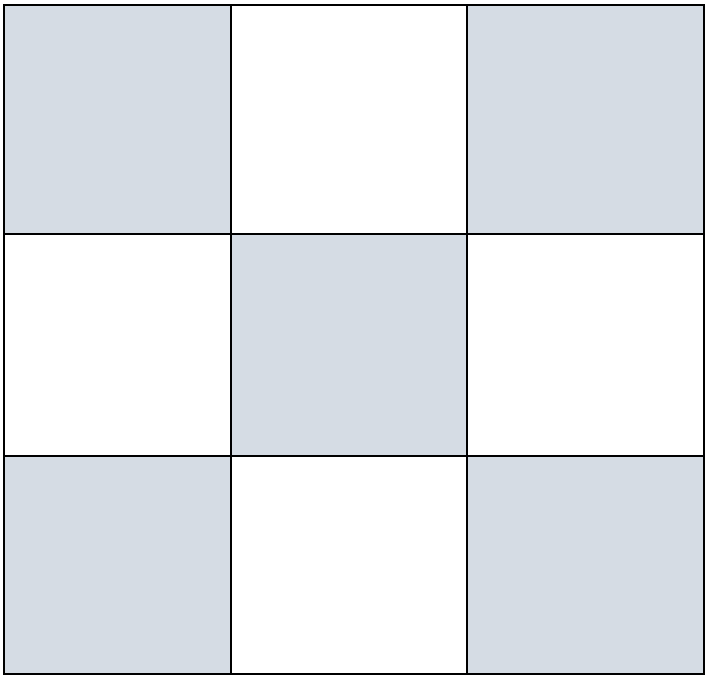


Задаем размер блока, кнопки и создаем массив для работы с игровым полем



Далее нам нужно создать функцию, которая будет заполнять наше игровое поле числами, но для этого нам нужно соблюсти некоторые принципы игры

Размещение блоков стандартного игрового поля:



Базовое размещение чисел

1	2	3	4	5	6	7	8	9
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
2	3	4	5	6	7	8	9	1
5	6	7	8	9	1	2	3	4
8	9	1	2	3	4	5	6	7
3	4	5	6	7	8	9	1	2
6	7	8	9	1	2	3	4	5
9	1	2	3	4	5	6	7	8

Для этого пропишем функцию

```
Ссылок: 2
public void GenerateArr() // Заполняем поле
{
    for(int i = 0; i < block_ * block_; i++)
    {
        for(int j = 0; j < block_ * block_; j++)
        {
            arr[i, j] = (i * block_ + i / block_ + j) % (block_ * block_) + 1;
            buttons[i, j] = new Button();
        }
    }

    Random r = new Random();
    for(int i = 0; i < 40; i++)
    {
        ShuffleMap(r.Next(0, 5));
    }

    CreateBut();
    HideCells();
}
```

А также пропишем функцию для создания кнопок с сгенерированными числами

```
Ссылок: 1
public void CreateBut() // Создаем кнопки
{
    for (int i = 0; i < block_ * block_; i++)
    {
        for (int j = 0; j < block_ * block_; j++)
        {
            Button button = new Button();

            buttons[i, j] = button;
            button.Size = new Size(butt_, butt_);
            button.Text = arr[i, j].ToString();
            button.Click += OnCellPressed;
            button.Location = new Point(j * butt_, i * butt_);

            this.Controls.Add(button);
        }
    }
}
```

Теперь мы имеем базу для игры

Но так играть нельзя, поэтому мы пропишем ряд функций для перетасовки чисел

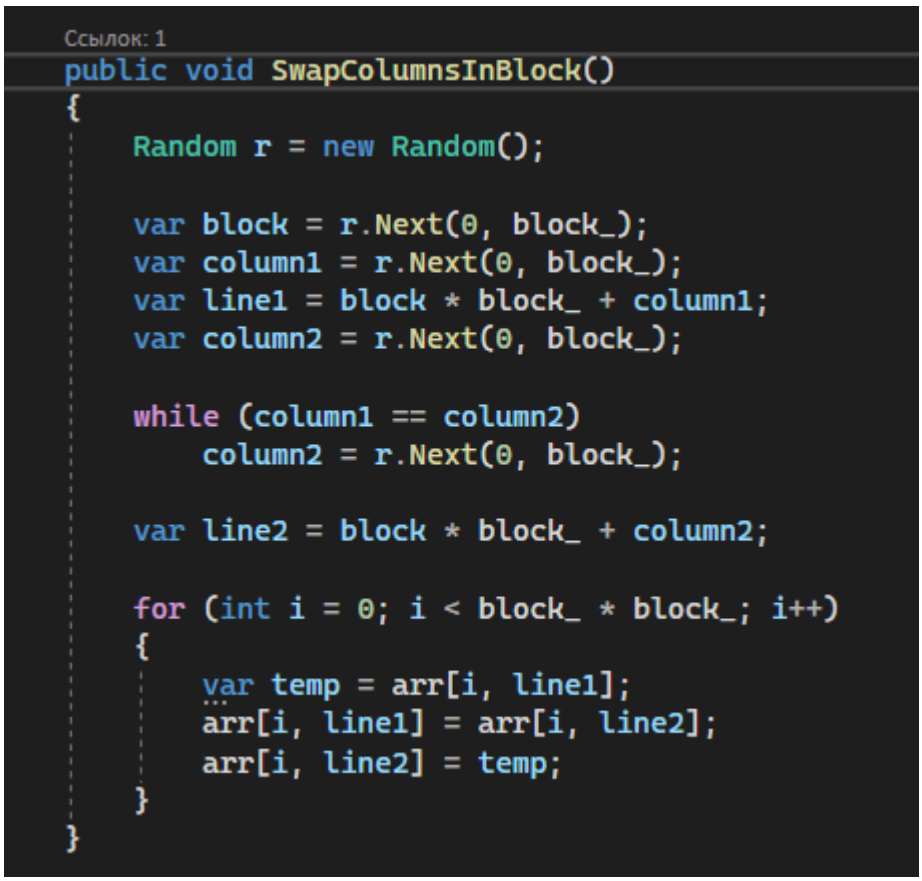
Транспонирование массива

```
//////////////////////////////////////////////////////////////Перетасовка//////////

Ссылок: 2
public void MatrixTransposition() // Транспонирование
{
    int[,] tArr = new int[block_ * block_, block_ * block_];
    for (int i = 0; i < block_ * block_; i++)
    {
        for (int j = 0; j < block_ * block_; j++)
        {
            tArr[i, j] = arr[j, i];
        }
    }
    arr = tArr;
}
```

Смена строк в пределах одного блока

```
Ссылок: 1
public void SwapRowsInBlock()
{
    Random r = new Random();

    var block = r.Next(0, block_);
    var row1 = r.Next(0, block_);
    var line1 = block * block_ + row1;
    var row2 = r.Next(0, block_);

    while (row1 == row2)
        row2 = r.Next(0, block_);

    var line2 = block * block_ + row2;

    for (int i = 0; i < block_ * block_; i++)
    {
        var temp = arr[line1, i];
        arr[line1, i] = arr[line2, i];
        arr[line2, i] = temp;
    }
}
```

Смена столбцов в пределах одного блока

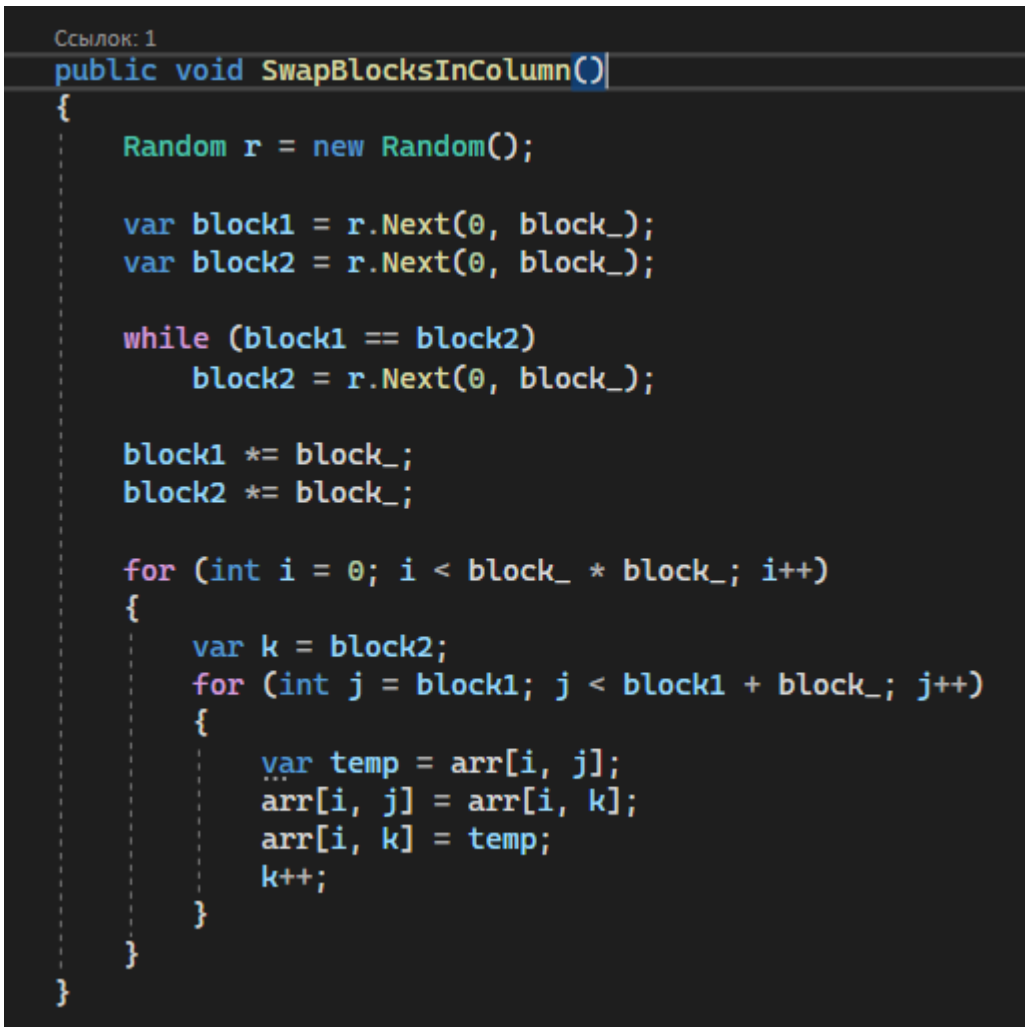


```
Ссылок: 1
public void SwapColumnsInBlock()
{
    Random r = new Random();

    var block = r.Next(0, block_);
    var column1 = r.Next(0, block_);
    var line1 = block * block_ + column1;
    var column2 = r.Next(0, block_);

    while (column1 == column2)
        column2 = r.Next(0, block_);

    var line2 = block * block_ + column2;

    for (int i = 0; i < block_ * block_; i++)
    {
        var temp = arr[i, line1];
        arr[i, line1] = arr[i, line2];
        arr[i, line2] = temp;
    }
}
```

Смена блоков по горизонтали

```
Ссылок: 1
public void SwapBlocksInRow()
{
    Random r = new Random();

    var block1 = r.Next(0, block_);
    var block2 = r.Next(0, block_);

    while (block1 == block2)
        block2 = r.Next(0, block_);

    block1 *= block_;
    block2 *= block_;

    for (int i = 0; i < block_ * block_; i++)
    {
        var k = block2;
        for (int j = block1; j < block1 + block_; j++)
        {
            var temp = arr[j, i];
            arr[j, i] = arr[k, i];
            arr[k, i] = temp;
            k++;
        }
    }
}
```

Смена блоков по вертикали



```
Ссылок: 1
public void SwapBlocksInColumn()
{
    Random r = new Random();

    var block1 = r.Next(0, block_);
    var block2 = r.Next(0, block_);

    while (block1 == block2)
        block2 = r.Next(0, block_);

    block1 *= block_;
    block2 *= block_;

    for (int i = 0; i < block_ * block_; i++)
    {
        var k = block2;
        for (int j = block1; j < block1 + block_; j++)
        {
            var temp = arr[i, j];
            arr[i, j] = arr[i, k];
            arr[i, k] = temp;
            k++;
        }
    }
}
```

Далее создадим функцию для выбора методов перетасовки и пропишем цикл, с рандомным выбором методов

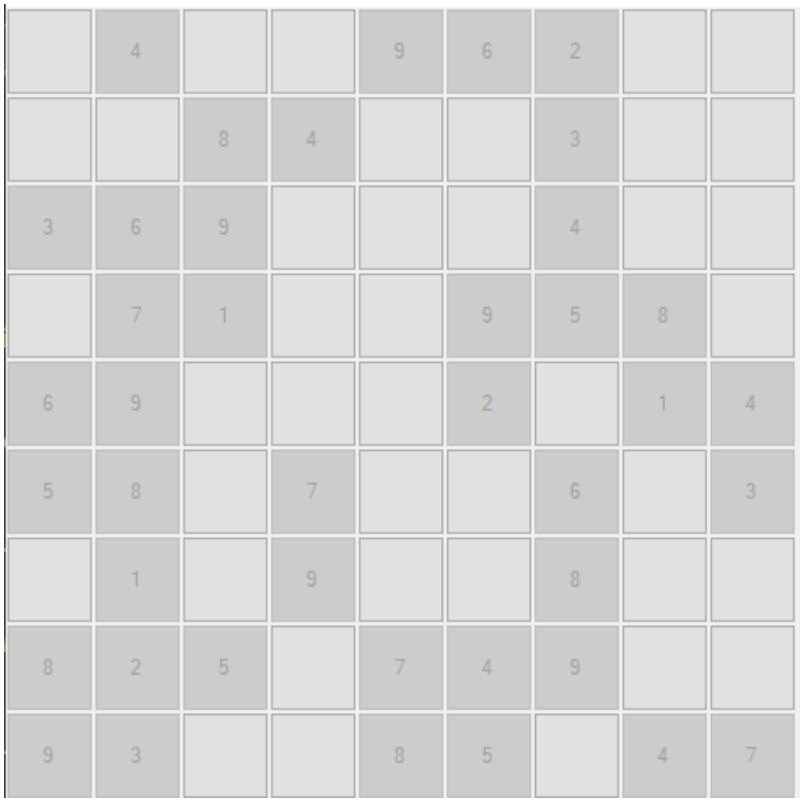
```
Ссылок: 1
public void ShuffleMap(int i)
{
    switch (i)
    {
        case 0:
            MatrixTransposition();
            break;
        case 1:
            SwapRowsInBlock();
            break;
        case 2:
            SwapColumnsInBlock();
            break;
        case 3:
            SwapBlocksInRow();
            break;
        case 4:
            SwapBlocksInColumn();
            break;
        default:
            MatrixTransposition();
            break;
    }
}
```

```
Random r = new Random();
for(int i = 0; i < 40; i++)
{
    ShuffleMap(r.Next(0, 5));
}
```

Далее нам нужно подготовить поле к игре, поэтому пропишем функцию, чтобы скрыть некоторые кнопки

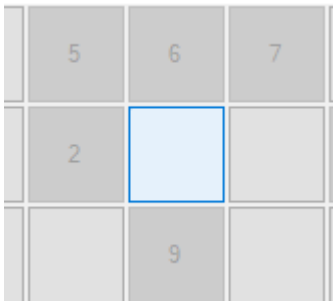
```
Ссылок: 1
public void HideCells() // Кнопки для игры
{
    int N = 40;
    Random r = new Random();
    while (N > 0)
    {
        for (int i = 0; i < block_ * block_; i++)
        {
            for (int j = 0; j < block_ * block_; j++)
            {
                if (!string.IsNullOrEmpty(buttons[i, j].Text)){
                    int a = r.Next(0, 3);
                    buttons[i, j].Text = a == 0 ? "" : buttons[i, j].Text;
                    buttons[i, j].Enabled = a == 0 ? true : false;

                    if (a == 0)
                        N--;
                    if (N <= 0)
                        break;
                }
            }
            if (N <= 0)
                break;
        }
    }
}
```

Выглядит это следующим образом



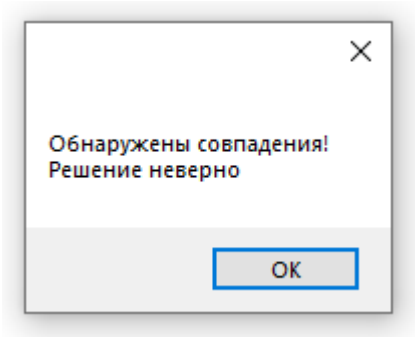
Теперь нам нужно прописать функцию, чтобы мы могли добавлять числа в пустые ячейки

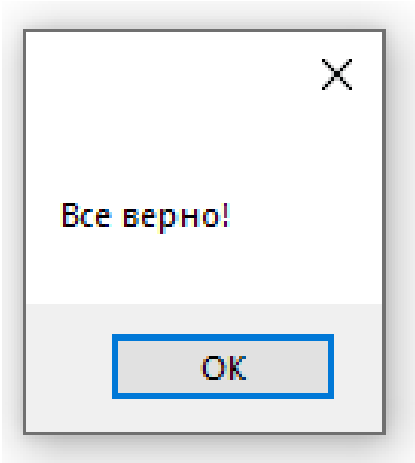
```
Ссылок: 1
public void OnCellPressed(object sender, EventArgs e)
{
    Button pressedButton = sender as Button;
    string buttonText = pressedButton.Text;
    if (string.IsNullOrEmpty(buttonText))
    {
        pressedButton.Text = "1";
    }
    else
    {
        int num = int.Parse(buttonText);
        num++;
        if (num == 10)
            num = 1;
        pressedButton.Text = num.ToString();
    }

}
```



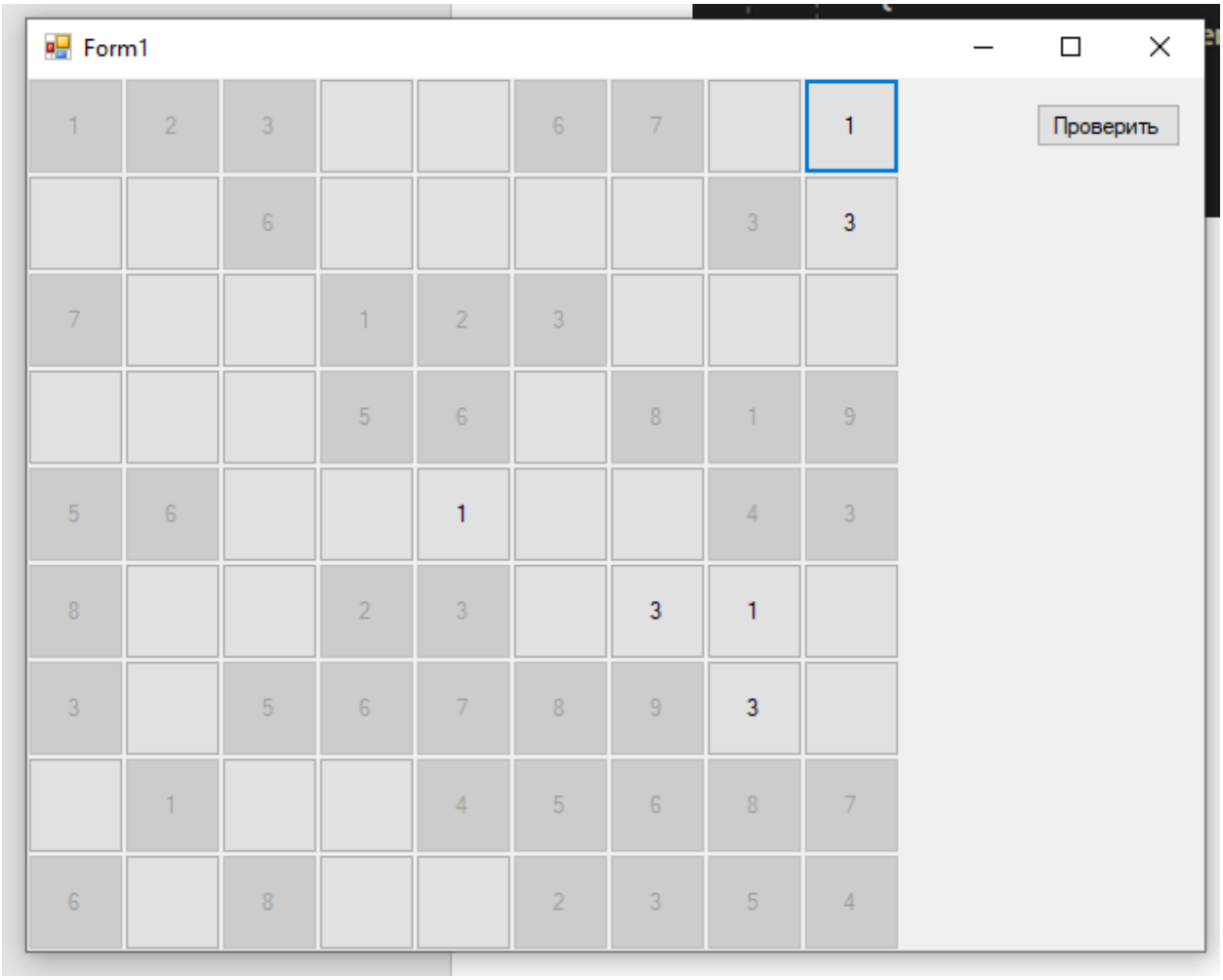
И финальный штрих, нужно прописать функцию для осуществления проверки, введенных чисел

```
ссылок: 1
private void button1_Click(object sender, EventArgs e)
{
    for(int i = 0; i < block_ * block_; i++)
    {
        for(int j = 0; j < block_ * block_; j++)
        {
            var btnText = buttons[i, j].Text;
            if(btnText != arr[i, j].ToString())
            {
                MessageBox.Show("Обнаружены совпадения!\nРешение неверно");
                return;
            }
        }
    }
    MessageBox.Show("Все верно!");
    for(int i = 0; i < block_ * block_; i++)
    {
        for (int j = 0; j < block_ * block_; j++)
        {
            this.Controls.Remove(buttons[i, j]);
        }
    }
    GenerateArr();
}
```





Готово!


Form1
Проверить
1 2 3 6 7 1
6 3 3
7 1 2 3
5 6 8 1 9
5 6 1 4 3
8 2 3 3 1
3 5 6 7 8 9 3
1 4 5 6 8 7
6 8 2 3 5 4