Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Компьютерные системы и сети (КСиС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

Игра «Судоку-онлайн»

БГУИР КП 1-40 01 01 24 ПЗ

Студент: гр. 751003 Стубеда В.Д.

Руководитель: асс. Красковский П.Н.

Минск, 2019

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc533025067)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc533025068)

[1.1 Анализ существующих приложений 6](#_Toc533025069)

[1.2 Постановка задачи 6](#_Toc533025070)

[2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 7](#_Toc533025071)

[2.1 Обоснование выбора языка программирования 7](#_Toc533025072)

[2.2 Архитектура программного средства 7](#_Toc533025073)

[3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА 8](#_Toc533025074)

[3.1 Визуальное представление приложения 8](#_Toc533025075)

[3.2 Анализ необходимы средств для отображения графики 10](#_Toc533025076)

[3.3 Разработка программного кода 10](#_Toc533025077)

[4 ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРИЛОЖЕНИЯ 14](#_Toc533025078)

[5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 15](#_Toc533025079)

[5.1 Начало работы 15](#_Toc533025080)

[5.2 Выбор параметров и начало работы 15](#_Toc533025081)

[5.3 Изучение истории использования программы 16](#_Toc533025082)

[ЗАКЛЮЧЕНИЕ 17](#_Toc533025083)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 18](#_Toc533025084)

[Приложение А. Код программы 19](#_Toc533025085)

# ВВЕДЕНИЕ

**Судо́ку** (яп. 数独 *су:доку*,) — головоломка с числами. Судоку активно публикуют газеты и журналы разных стран мира, сборники судоку издаются большими тиражами. Решение судоку — популярный вид досуга.

**Происхождение:**

В XVIII веке Леонард Эйлер изобрёл игру «*Carré latin*» («*Латинский квадрат*»). На основе этой игры в 70-х годах прошлого века в Северной Америке были придуманы специальные числовые головоломки. Так, в США судоку появилась впервые в 1979 году в журнале «*Dell Puzzle Magazine*». Тогда она называлась «*Number Place*». Настоящую популярность судоку завоевала в 1980—1990-х годах, когда японский журнал «Nikoli» начал регулярно публиковать на своих страницах эту головоломку (с 1986 года). 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».

**Правила игры:**

Игровое поле представляет собой квадрат размером 9×9, 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), называемые *подсказками*. От игрока требуется заполнить свободные клетки цифрами от 1 до 9 так, чтобы в каждой строке, в каждом столбце и в каждом малом квадрате 3×3 каждая цифра встречалась бы только один раз

Сложность судоку зависит от количества изначально заполненных клеток и от методов, которые нужно применять для её решения. Самые простые решаются дедуктивно: всегда есть хотя бы одна клетка, куда подходит только одно число. Некоторые головоломки можно решить за несколько минут, на другие можно потратить часы.

В связи с этим, было принято решение создать программный продукт, который реализует основы игры в судоку, с простым для понимания и функциональным интерфейсом.

В качестве языка программирования был выбран язык C#, а средой разработки – Visual Studio 2017.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

# Анализ существующих приложений

Первые упоминания о такой игре как «Судоку» были найдены в Китае (I век н. э.). Тогда она называлась «Магический квадрат» и уже тогда люди думали об алгоритмах быстрого решения этой игры. Однако значительный прорыв сделал только знаменитый швейцарский математик и физик Леонард Эйлер (1707 - 1783). В документе, озаглавленном "Научное исследование новых разновидностей магического квадрата" Эйлер помещал в клетки латинские буквы (получая Латинский квадрат), позже он заполнил клетки греческими буквами и называл квадрат греко-латинским. Исследуя различные варианты магического квадрата, Эйлер обратил внимание на проблему комбинации символов так, чтобы ни один из них не повторялся ни в одной строке и ни в одном столбце. С помощью своих исследований он выяснил, что в матрице размером 9х9 каждый ряд и каждую колонку можно заполнить цифрами от 1 до 9 в определенном порядке и без повторения.

В настоящее время существует множество электронных версий приложений судоку. Большинство из них представляют собой сайты с возможностью поиграть в судоку.

## 1.2 Постановка задачи

Назначение приложения «Sudoku» заключается в возможности создания аккаунта и игры с другими клиентами по локальной сети. В программе пользователю доступны следующие возможности:

– подключение к серверу;

– просмотр всех имеющихся на данный момент аккаунтов в игре в виде таблицы с основной информацией;

– сортировка аккаунтов по количеству очков;

– взаимодействие с пользователями;

– взаимодействие с сервером;

# 2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## 2.1 Обоснование выбора языка программирования

Для создания данного приложения решено использовать язык программирования C# и интегрированная среда разработки Visual Studio.

C# - это язык программирования, предназначенный для разработки самых разнообразных приложений. Поддержка Visual Studio обеспечивается с помощью полнофункционального редактора кода, компилятора, шаблонов проектов, конструкторов, мастеров кода, мощного и удобного отладчика и многих других средств. Данный язык позволяет стартовать разработчику быстрее, что позволяет быстрее получить прототип решения.

## 2.2 Архитектура программного средства

На этапе продумывания приложения курсового проекта и проектирования было принято решение о необходимости реализовать следующие основные функции:

* Подключение к серверу;
* Взаимодействие с другими клиентами через сервер;
* Взаимодействие со стороны сервера с несколькими пользователями;
* Генерация судоку;
* Проверка судоку на правильность;

Также есть необходимость реализации методов взаимодействия с графическим интерфейсом с помощью клавиатуры.

# 3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

## 3.1 Визуальное представление приложения

Важным качеством программного средства является простота в использовании. Программное средство, ориентированное на широкую аудиторию, понятно не только для опытных, но и для начинающих пользователей.

Данное приложение представляет небольшое окно, всплывающее посередине экрана. В окне представлено главное меню (рис 3.1), предлагающее сменить игрока, играть и выйти.

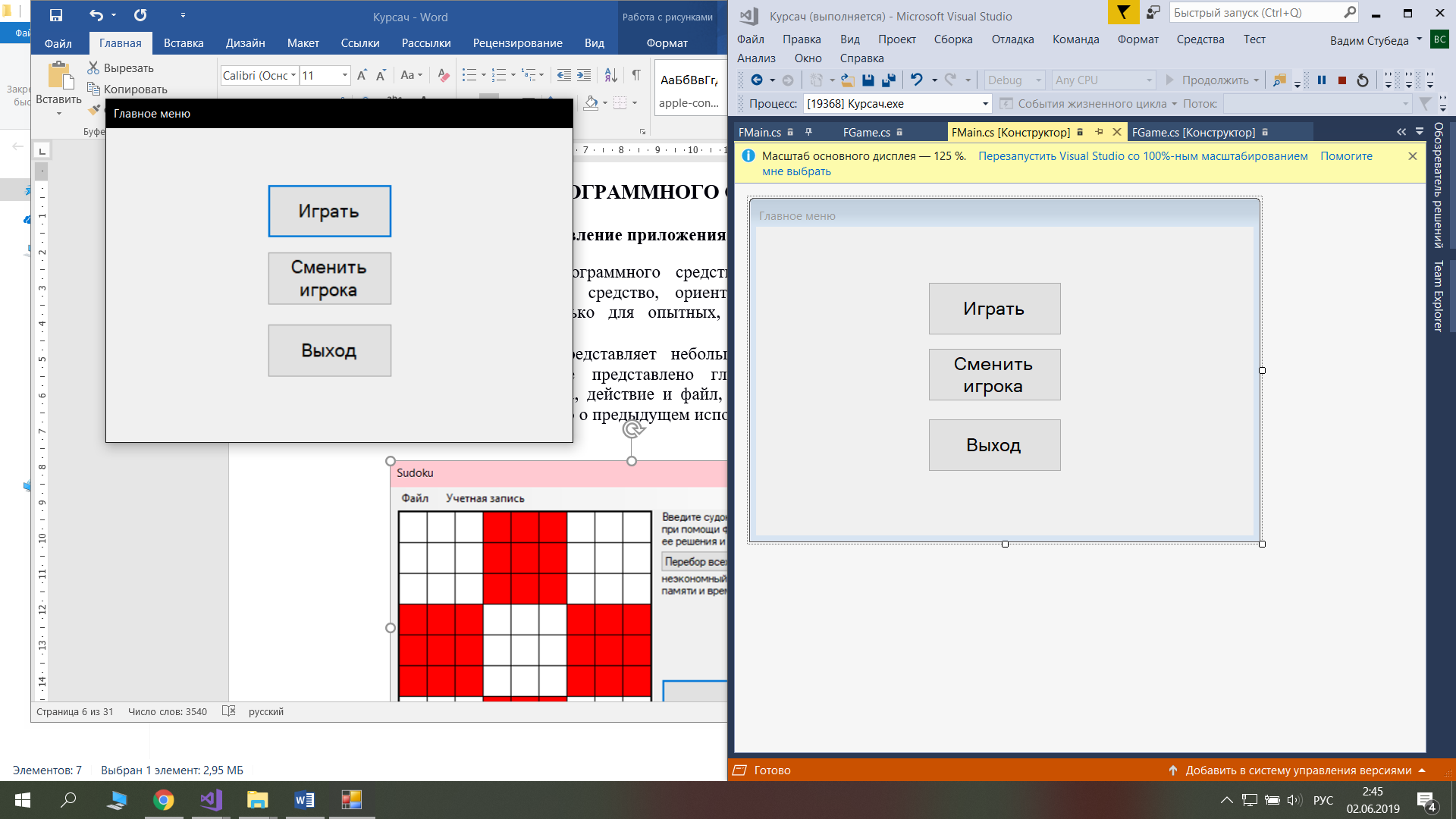


Рисунок 3.1 – Главное меню

Для работы пользователю нужно выбрать сменить игрока («Сменить игрока») и нажать кнопу «Играть». После нажатия кнопки «Играть» появится новое окно (рис 3.2).

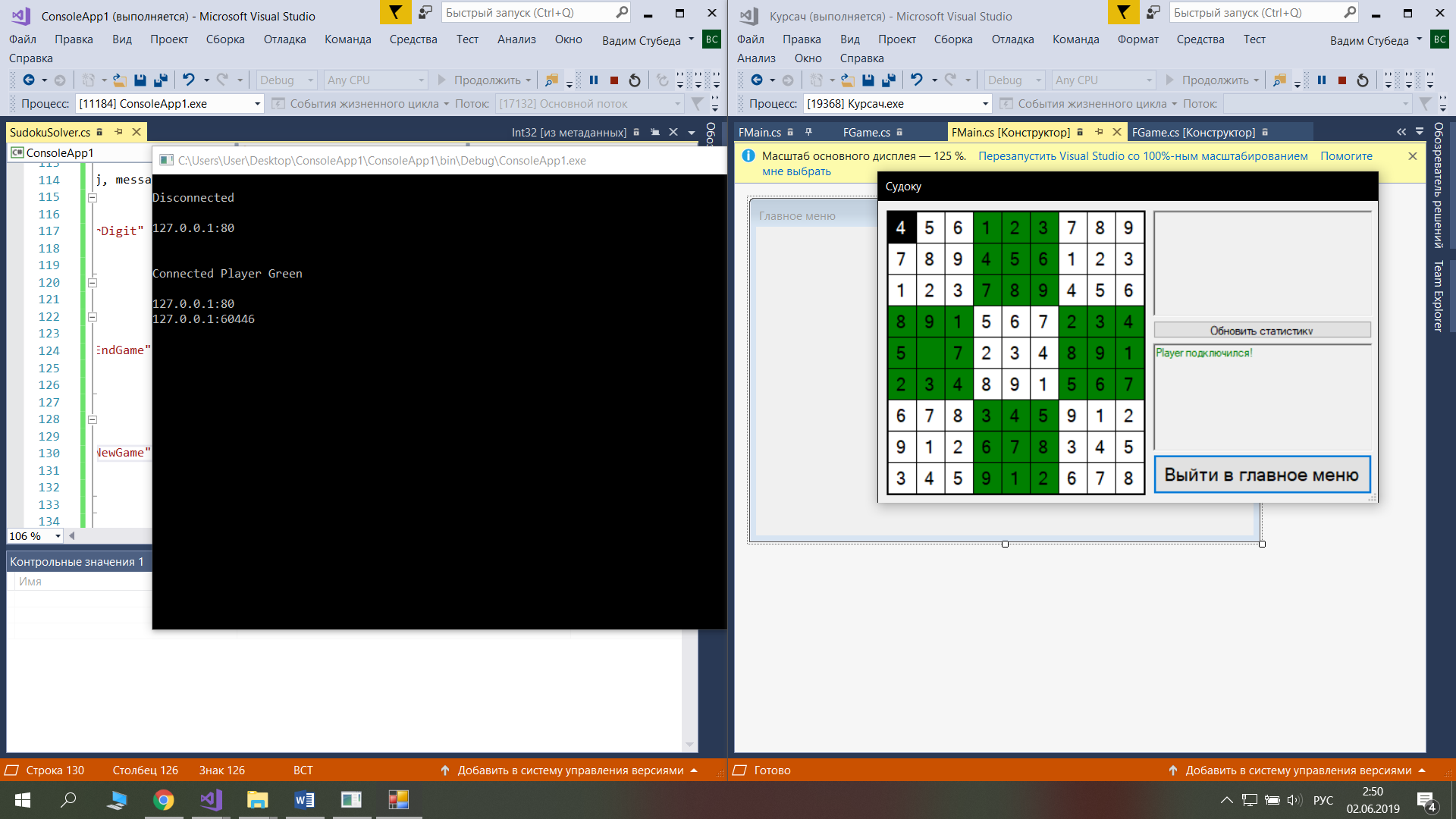


Рисунок 3.2 – Окно игры

Для запуска ввода значения нужно выбрать ячейку в таблице, нажать на цифру, а затем Enter.

После этого, в окне консоли появится сообщение от сервера (рис 3.3).

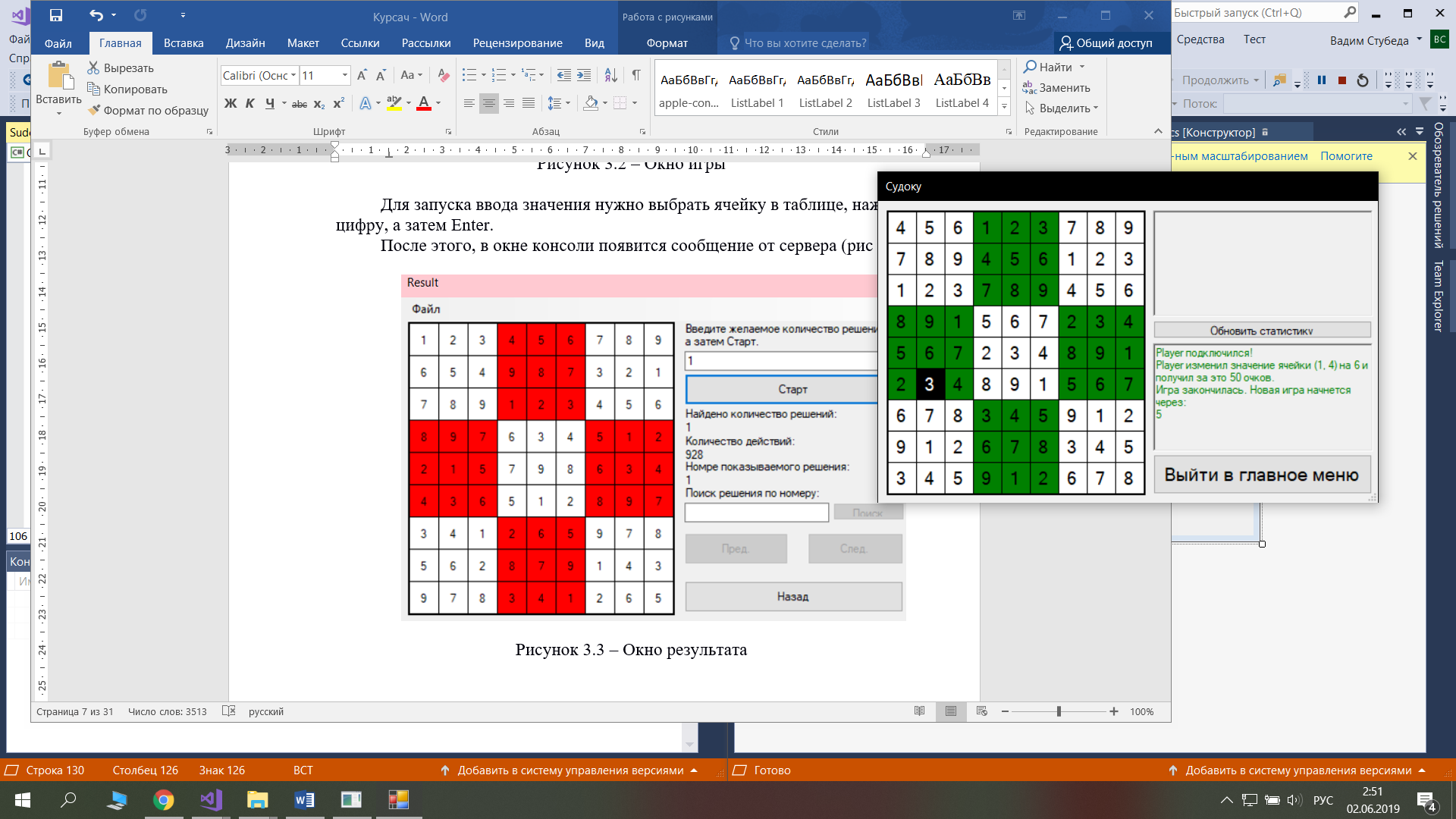


Рисунок 3.3 – Окно результата

## 3.2 Анализ необходимы средств для отображения графики

Так как данное программное средство создано для работы с кроссвордами, т.е. на входе получает условие, на выходе выдает заполненную таблицу, при этом действия над данными отображаются визуально, то графическое представление вызывает сложности и требует больших затрат. Оно представлено стандартными компонентами. Относительную сложность, также, представляет отображение и обработка данных в таблице, т.к. эта часть программы доступна и видна пользователю.

## 3.3 Разработка программного кода

Плотное взаимодействие данными пользователя обязывает пристально следить за поступающей и выходящей информацией, искать и устранять любые неоднозначности и уязвимости алгоритма.

Таким образом было добавлено большое количество проверок и ответвлений, что привело к усложнению и ухудшению читаемости кода, однако позволило обезопасить пользовательские данные.

Ниже представлен псевдоалгоритм нахождения решения судоку рекурсивным способом.

Рекурсивный способ

Если уже элемент под номером 82

Занесение текущей решенной судоку в результаты

Увеличение количества найденных решений

Конец если

Если сейчас элемент с номером меньшим 82 и количество найденных решений меньше чем нужно

Если элемент задан условием

Вызвать «Рекурсивный обычный метод» со следующим элементом

Иначе

Перебираем кандидатов от 1 до 9

Если кандидат подходит и не нарушает правила

Заносим текущего кандидата в текущее решение

Увеличиваем количество действий на один

Вызвать «Рекурсивный обычный метод» со следующим элементом

Конец если

Убираем кандидата из текущего решения

Увеличиваем количество действий на один

Конец перебора

Конец если

Конец если

# ****4 ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРИЛОЖЕНИЯ****

Осуществлялось функциональное тестирование; отчёт о проведённом тестировании представлен в таблице 4.1.

Таблица 4.1 – Тестирование программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестируемая функция | Ожидаемый результат | Полученный результат |
| 1 | Открытие программы, нажатие кнопки «Игра», с отключенным сервером | - | Соответствует ожидаемому |
| 2 | Открытие программы, нажатие кнопки «Сменить игрока» | Появления окна выбора игрока | Соответствует ожидаемому |
| 3 | Открытие программы, нажатие кнопки «Выход» | Выход из программы | Соответствует ожидаемому |
| 4 | Ввод цифры в таблицу | Ответ сервера | Соответствует ожидаемому |
| 5 | Открытие программы, нажатие кнопки «Обновить статистику» | Обновление статистики | Соответствует ожидаемому |
| 7 | Закрытие окна главного меню | Завершение работы программы | Соответствует ожидаемому |

В ходе итогового тестирования не было выявлено каких-либо ошибок либо некорректной работы приложения.

# 5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

5.1 Начало работы

Для начала работы необходимо запустить приложение. Перед пользователем появится окно, в котором отобразится главное меню.

## 5.2 Выбор параметров и начало работы

После появления стартового окна следует выбрать. Затем подтвердить выбор. После нажать кнопку «Играть». Появится окно игры. Ввести в пустую клетку нужное значение и нажать кнопку Enter, либо покинуть окно редактирования ячейки. Вид окна главного меню представлен на рисунке 5.1

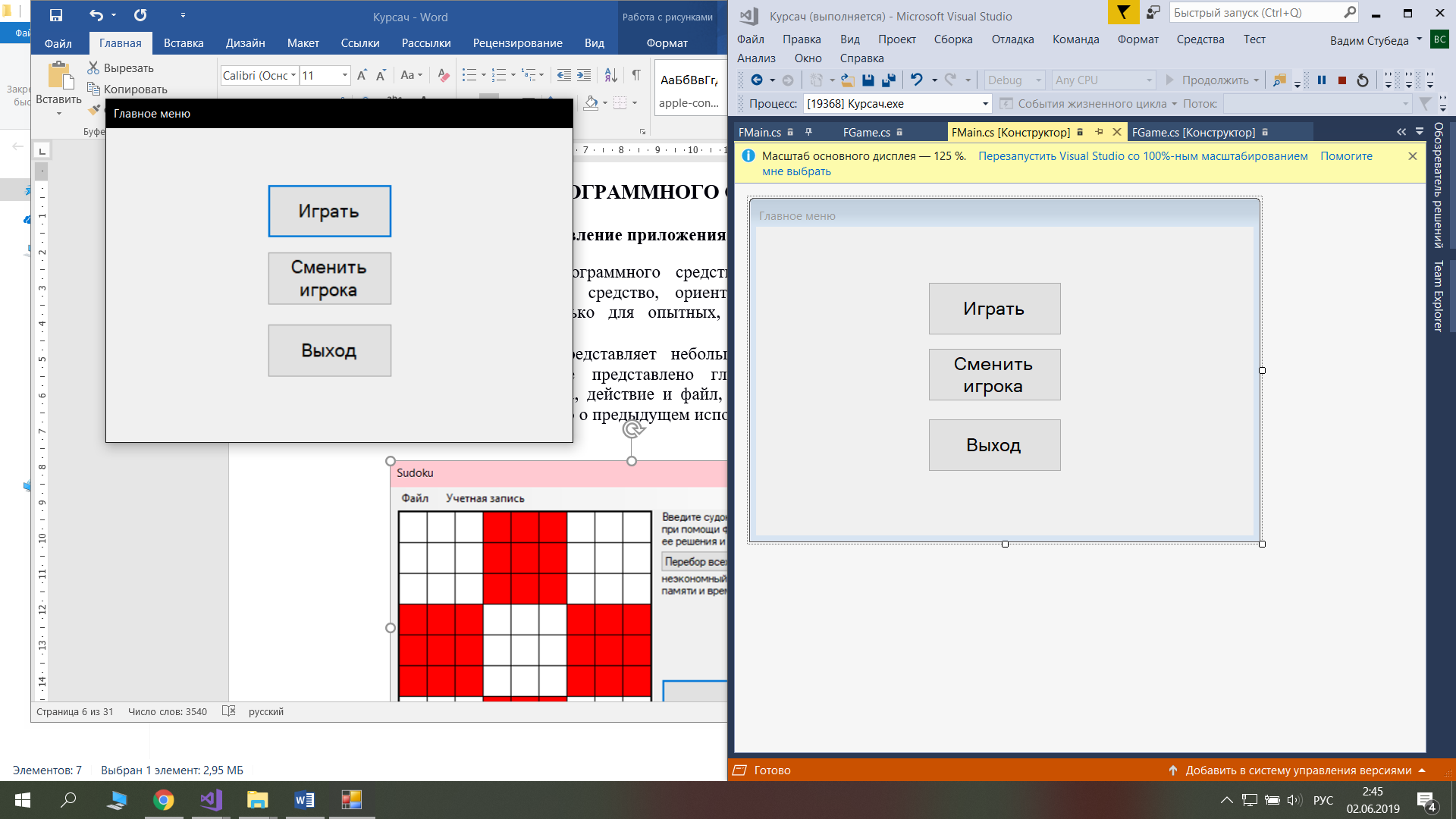


Рисунок 5.1 – Окно главного меню

## 5.3 Изучение истории использования программы

Нажать кнопку «Обновить статистику». Всплывет окно, представленное на рисунке 5.2. В таблице представлены поля: Имя, Последнее посещение аккаунта, Количество очков.

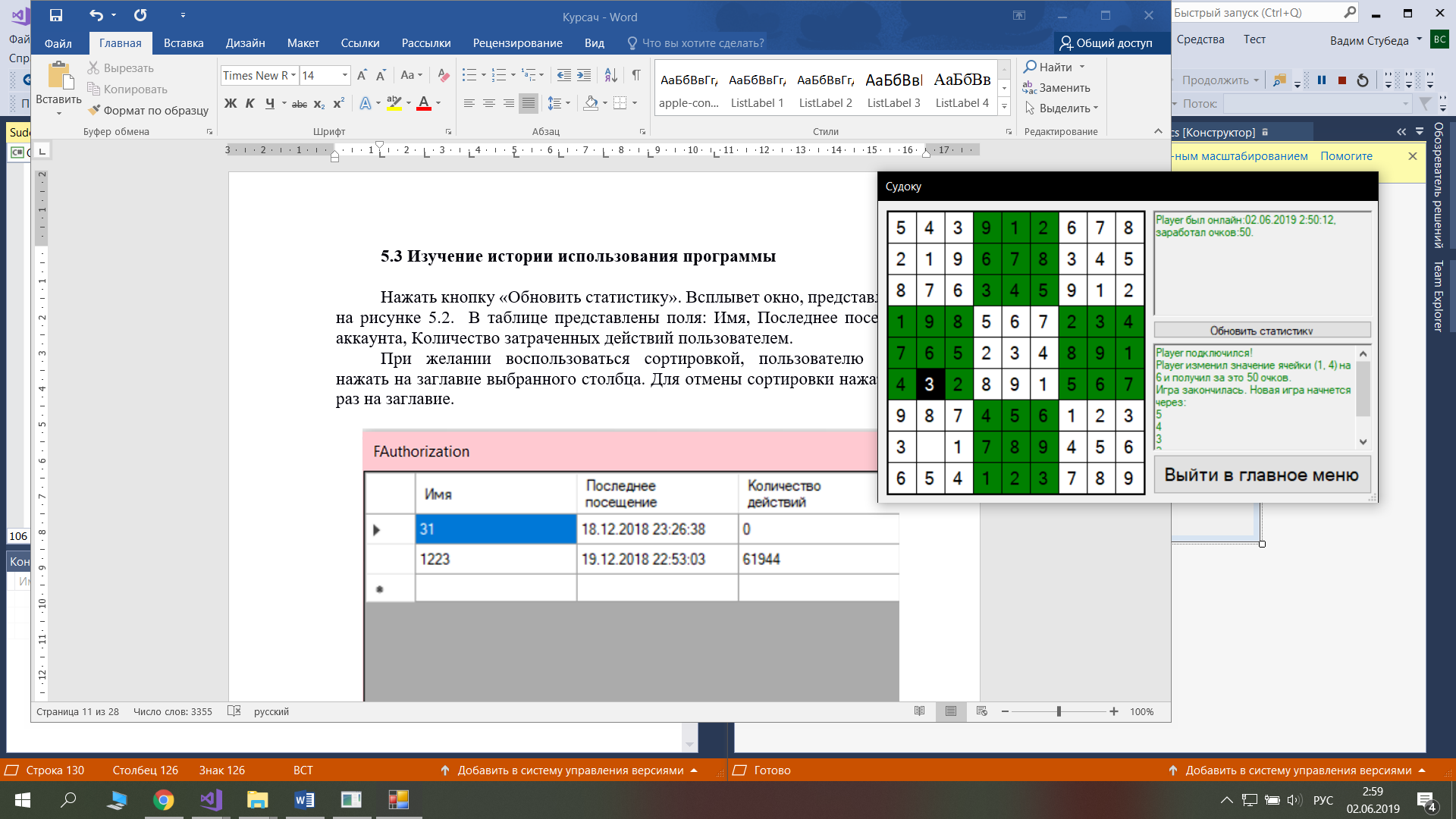


Рисунок 5.2 – Окно игры.

# ЗАКЛЮЧЕНИЕ

В ходе разработки приложения был создан продукт, предоставляющий возможность игры в «Судоку-онлайн», взаимодействия с другими пользователями, возможность подключения, отключения и выхода из игры. Данный программный продукт может быть полезен людям, которые поиграть в судоку онлайн.

Данное приложение имеет простой интерфейс и высокую скорость работы. Все выявленные в процессе тестирования неполадки, были устранены.

При дальнейшей доработке программного средства, возможно добавление изображений для каждого из аккаунтов, автоматическая генерация имени для аккаунта. Также есть вероятность значительного расширения функционала отчётов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Статься об игре «Sudoku» [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/ Sudoku – Дата доступа: 25.04.2018
2. Сайт игры «Sudoku» [Электронный ресурс] – Режим доступа: http://www.sudoku.com/ – Дата доступа: 20.04.2018
3. А. Хейлсбегр, М. Торгерсен, С. Вилтамут, П. Голд: Язык прогроммирования C#. Четвертое издание. – 773с.
4. Дж. Шарп. Microsoft Visual **C#**. Подробное руководство – 848c.

# Приложение А. Код программы

Главное меню:

using System;

using System.Drawing;

using System.Net.Sockets;

using System.Text;

using System.Windows.Forms;

namespace Курсач

{

public partial class FMain : Form

{

public string PlayerNickname = "Player";

public Color PlayerColor = Color.Green;

FAuthorization PlayerInfo = new FAuthorization();

FGame Game = new FGame();

const string ip = "127.0.0.1";

const int port = 80;

public FMain()

{

InitializeComponent();

}

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

{

Game.PlayerNickname = PlayerNickname;

Game.PlayerColor = PlayerColor;

Hide();

Game.ShowDialog();

Show();

}

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

{

PlayerInfo.PlayerNickname = PlayerNickname;

PlayerInfo.PlayerColor = PlayerColor;

PlayerInfo.ShowDialog();

PlayerNickname = PlayerInfo.PlayerNickname;

PlayerColor = PlayerInfo.PlayerColor;

}

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

{

}

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

{

Game.udpSocket.SendTo(Encoding.Default.GetBytes("Exit" + ' '), Game.ServerEndPoint);

Game.udpSocket.Shutdown(SocketShutdown.Both);

Game.udpSocket.Close();

Close();

}

}

}

Авторизация:

using System;

using System.Drawing;

using System.Drawing.Text;

using System.Windows.Forms;

namespace Курсач

{

public partial class FAuthorization : Form

{

public string PlayerNickname { get; set; }

public Color PlayerColor { get; set; }

public FAuthorization()

{

InitializeComponent();

CBColor.Items.Add(Color.Green);

CBColor.Items.Add(Color.Brown);

CBColor.Items.Add(Color.Yellow);

CBColor.Items.Add(Color.SpringGreen);

CBColor.Items.Add(Color.Violet);

CBColor.Items.Add(Color.SkyBlue);

CBColor.Items.Add(Color.SeaGreen);

CBColor.Items.Add(Color.RoyalBlue);

CBColor.Items.Add(Color.Purple);

}

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

{

Hide();

}

private void CBColor\_DrawItem(object sender, DrawItemEventArgs e)

{

int MarginWidth = 4;

int MarginHeight = 4;

// Draw a ComboBox item that is displaying a color sample

if (e.Index < 0) return;

// Clear the background appropriately.

e.DrawBackground();

// Draw the color sample.

int hgt = e.Bounds.Height - 2 \* MarginHeight;

Rectangle rect = new Rectangle(

e.Bounds.X + MarginWidth,

e.Bounds.Y + MarginHeight,

hgt, hgt);

ComboBox cbo = sender as ComboBox;

Color color = (Color)cbo.Items[e.Index];

using (SolidBrush brush = new SolidBrush(color))

{

e.Graphics.FillRectangle(brush, rect);

}

// Outline the sample in black.

e.Graphics.DrawRectangle(Pens.Black, rect);

// Draw the color's name to the right.

using (Font font = new Font(cbo.Font.FontFamily,

cbo.Font.Size \* 0.75f, FontStyle.Bold))

{

using (StringFormat sf = new StringFormat())

{

sf.Alignment = StringAlignment.Near;

sf.LineAlignment = StringAlignment.Center;

int x = hgt + 2 \* MarginWidth;

int y = e.Bounds.Y + e.Bounds.Height / 2;

e.Graphics.TextRenderingHint =

TextRenderingHint.AntiAliasGridFit;

e.Graphics.DrawString(color.Name, font,

Brushes.Black, x, y, sf);

}

}

// Draw the focus rectangle if appropriate.

e.DrawFocusRectangle();

}

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

{

TBNickname.Text = PlayerNickname;

CBColor.SelectedItem = PlayerColor;

}

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

{

PlayerNickname = TBNickname.Text;

PlayerColor = (Color)CBColor.SelectedItem;

Hide();

}

}

}

Игра:

using System;

using System.Drawing;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсач

{

public partial class FGame : Form

{

public string PlayerNickname { get; set; }

public Color PlayerColor { get; set; }

const string ip = "127.0.0.1";

const int port = 80;

public EndPoint ServerEndPoint { get; set; } = new IPEndPoint(IPAddress.Parse(ip), port);

public Socket udpSocket { get; set; } = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);

string[] message;

public Task RecipientTask;

public bool UserKeyPress = false;

public FGame()

{

InitializeComponent();

DGVMainArea.AllowUserToAddRows = false;

DGVMainArea.AllowUserToDeleteRows = false;

DGVMainArea.AllowUserToResizeColumns = false;

DGVMainArea.AllowUserToResizeRows = false;

DGVMainArea.RowHeadersVisible = false;

DGVMainArea.ColumnHeadersVisible = false;

DGVMainArea.ScrollBars = ScrollBars.None;

DGVMainArea.GridColor = Color.Black;

DGVMainArea.CellBorderStyle = DataGridViewCellBorderStyle.Single;

DGVMainArea.AlternatingRowsDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

DGVMainArea.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

DGVMainArea.ColumnCount = 9;

DGVMainArea.Rows.Add(9);

DGVMainArea.MultiSelect = false;

for (int i = 0; i < 9; i++)

{

DGVMainArea.Columns[i].Width = DGVMainArea.Width / 9;

DGVMainArea.Rows[i].Height = DGVMainArea.Height / 9;

}

DGVMainArea.Width = DGVMainArea.Columns[1].Width \* 9 + 3;

RecipientTask = new Task(Recipient);

}

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

{

RTBConsole.Text = "";

for (int i = 0; i < 9; i++)

for (int j = 0; j < 9; j++)

if (((i == 3) | (i == 4) | (i == 5)) & !((j == 3) | (j == 4) | (j == 5)) |

!((i == 3) | (i == 4) | (i == 5)) & ((j == 3) | (j == 4) | (j == 5)))

DGVMainArea.Rows[i].Cells[j].Style.BackColor = PlayerColor;

if (RecipientTask.Status.GetHashCode() != 3)

{

RecipientTask.Start();

}

udpSocket.SendTo(Encoding.Default.GetBytes("Connected" + ' ' + PlayerNickname + ' ' + PlayerColor.Name + ' '), ServerEndPoint);

}

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

{

udpSocket.SendTo(Encoding.Default.GetBytes("Disconnected" + ' '), ServerEndPoint);

Hide();

}

private void DGVMainArea\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

TextBox tb = (TextBox)e.Control;

tb.KeyPress += new KeyPressEventHandler(tb\_KeyPress);

tb.MaxLength = 1;

}

void tb\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Char.IsNumber(e.KeyChar) && (e.KeyChar != (char)Keys.Back) || (e.KeyChar == '0'))

{

e.Handled = true;

}

else

{

UserKeyPress = true;

}

}

private void Recipient()

{

while (true)

{

EndPoint CurSenderEndPoint = ServerEndPoint;

byte[] buffer = new byte[200];

try

{

udpSocket.ReceiveFrom(buffer, ref CurSenderEndPoint);

}

catch (Exception e)

{

if (e.HResult != 10004)

{

Console.WriteLine(e.ToString());

}

}

if (CurSenderEndPoint == ServerEndPoint)

{

message = Encoding.Default.GetString(buffer).Split(' ');

while (!RTBConsole.IsHandleCreated || !DGVMainArea.IsHandleCreated)

{

}

switch (message[0])

{

case "Connected":

RTBConsole.Invoke((MethodInvoker)delegate

{

RTBConsole.AppendText(message[1] + " подключился!" + '\n', Color.FromName(message[2]));

});

break;

case "Disconnected":

RTBConsole.Invoke((MethodInvoker)delegate

{

RTBConsole.AppendText(message[1] + " отключился." + '\n', Color.FromName(message[2]));

});

break;

case "Exit":

RTBConsole.Invoke((MethodInvoker)delegate

{

RTBConsole.AppendText(message[1] + " вышел." + '\n', Color.FromName(message[2]));

});

break;

case "NewGame":

DGVMainArea.Invoke((MethodInvoker)delegate

{

for (int i = 0; i < 9; i++)

for (int j = 0; j < 9; j++)

if (message[1][i \* 9 + j] != '0')

{

DGVMainArea.Rows[j].Cells[i].Value = message[1][i \* 9 + j];

DGVMainArea.Rows[j].Cells[i].Style.ForeColor = Color.Black;

DGVMainArea.Rows[j].Cells[i].ReadOnly = true;

}

else

{

DGVMainArea.Rows[j].Cells[i].Value = ' ';

DGVMainArea.Rows[j].Cells[i].Style.ForeColor = Color.Black;

DGVMainArea.Rows[j].Cells[i].ReadOnly = false;

}

});

break;

case "EnterDigit":

RTBConsole.Invoke((MethodInvoker)delegate

{

RTBConsole.AppendText(message[1] + " изменил значение ячейки (" + message[3] + ", " + message[4] + ") на " + message[5] + " и получил за это " + message[6] + " очков." + '\n', Color.FromName(message[2]));

});

DGVMainArea.Invoke((MethodInvoker)delegate

{

DGVMainArea.Rows[int.Parse(message[4])].Cells[int.Parse(message[3])].Value = message[5];

DGVMainArea.Rows[int.Parse(message[4])].Cells[int.Parse(message[3])].Style.ForeColor = Color.FromName(message[2]) == PlayerColor ? Color.Black : Color.FromName(message[2]);

DGVMainArea.Rows[int.Parse(message[4])].Cells[int.Parse(message[3])].ReadOnly = true;

});

break;

case "Error":

RTBConsole.Invoke((MethodInvoker)delegate

{

RTBConsole.AppendText(message[1] + " изменил значение ячейки (" + message[3] + ", " + message[4] + ") на " + message[5] + " и ошибся." + '\n', Color.FromName(message[2]));

});

DGVMainArea.Invoke((MethodInvoker)delegate

{

DGVMainArea.Rows[int.Parse(message[4])].Cells[int.Parse(message[3])].Value = message[5];

DGVMainArea.Rows[int.Parse(message[4])].Cells[int.Parse(message[3])].Style.ForeColor = Color.FromName(message[2]);

DGVMainArea.Rows[int.Parse(message[4])].Cells[int.Parse(message[3])].ReadOnly = true;

});

switch (message[6])

{

case "One":

DGVMainArea.Invoke((MethodInvoker)delegate

{

DGVMainArea.Rows[int.Parse(message[4])].Cells[int.Parse(message[3])].Style.BackColor = Color.Red;

});

break;

case "Horizontal":

DGVMainArea.Invoke((MethodInvoker)delegate

{

for (int i = 0; i < 9; i++)

DGVMainArea.Rows[int.Parse(message[4])].Cells[i].Style.BackColor = Color.Red;

});

break;

case "Vertical":

DGVMainArea.Invoke((MethodInvoker)delegate

{

for (int i = 0; i < 9; i++)

DGVMainArea.Rows[i].Cells[int.Parse(message[3])].Style.BackColor = Color.Red;

});

break;

case "Square":

DGVMainArea.Invoke((MethodInvoker)delegate

{

for (int i = 0; i < 3; i++)

for (int j = 0; j < 3; j++)

DGVMainArea.Rows[int.Parse(message[4]) / 3 \* 3 + i].Cells[int.Parse(message[3]) / 3 \* 3 + j].Style.BackColor = Color.Red;

});

break;

}

Thread.Sleep(1000);

DGVMainArea.Invoke((MethodInvoker)delegate

{

DGVMainArea.Rows[int.Parse(message[4])].Cells[int.Parse(message[3])].Value = ' ';

DGVMainArea.Rows[int.Parse(message[4])].Cells[int.Parse(message[3])].Style.ForeColor = Color.Black;

DGVMainArea.Rows[int.Parse(message[4])].Cells[int.Parse(message[3])].ReadOnly = false;

});

for (int i = 0; i < 9; i++)

for (int j = 0; j < 9; j++)

if (((i == 3) | (i == 4) | (i == 5)) & !((j == 3) | (j == 4) | (j == 5)) |

!((i == 3) | (i == 4) | (i == 5)) & ((j == 3) | (j == 4) | (j == 5)))

DGVMainArea.Rows[i].Cells[j].Style.BackColor = PlayerColor;

else

DGVMainArea.Rows[i].Cells[j].Style.BackColor = Color.White;

break;

case "GetStatistics":

RTBStatistics.Invoke((MethodInvoker)delegate

{

RTBStatistics.AppendText(message[1] + " был онлайн:" + message[3] + ' ' + message[4] + ", заработал очков:" + message[5] + '.' + '\n', Color.FromName(message[2]));

});

break;

case "EndGame":

DGVMainArea.Invoke((MethodInvoker)delegate

{

DGVMainArea.ReadOnly = true;

});

RTBConsole.Invoke((MethodInvoker)delegate

{

RTBConsole.AppendText("Игра закончилась. Новая игра начнется через:\n", PlayerColor);

});

RTBConsole.Invoke((MethodInvoker)delegate

{

RTBConsole.AppendText("5\n", PlayerColor);

});

Thread.Sleep(1000);

RTBConsole.Invoke((MethodInvoker)delegate

{

RTBConsole.AppendText("4\n", PlayerColor);

});

Thread.Sleep(1000);

RTBConsole.Invoke((MethodInvoker)delegate

{

RTBConsole.AppendText("3\n", PlayerColor);

});

Thread.Sleep(1000);

RTBConsole.Invoke((MethodInvoker)delegate

{

RTBConsole.AppendText("2\n", PlayerColor);

});

Thread.Sleep(1000);

RTBConsole.Invoke((MethodInvoker)delegate

{

RTBConsole.AppendText("1\n", PlayerColor);

});

Thread.Sleep(1000);

DGVMainArea.Invoke((MethodInvoker)delegate

{

DGVMainArea.ReadOnly = false;

});

break;

}

}

}

}

private void DGVMainArea\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

if (UserKeyPress)

{

udpSocket.SendTo(Encoding.Default.GetBytes(

"EnterDigit" + ' ' +

e.ColumnIndex.ToString() + ' ' +

e.RowIndex.ToString() + ' ' +

DGVMainArea.Rows[e.RowIndex].Cells[e.ColumnIndex].Value + ' '),

ServerEndPoint);

UserKeyPress = false;

}

}

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

{

RTBStatistics.Text = "";

udpSocket.SendTo(Encoding.Default.GetBytes("GetStatistics" + ' '), ServerEndPoint);

}

}

public static class RichTextBoxExtensions

{

public static void AppendText(this RichTextBox box, string text, Color color)

{

box.SelectionStart = box.TextLength;

box.SelectionLength = 0;

box.SelectionColor = color;

box.AppendText(text);

box.SelectionColor = box.ForeColor;

}

}

}

Сервер:

using System;

using System.Collections.Generic;

using System.Net;

using System.Net.Sockets;

using System.Text;

namespace ConsoleApp1

{

public struct Player

{

public EndPoint PlayerEndPoint;

public string Nickname;

public string Color;

public string LastEntrance;

public int Score;

}

class Server

{

public string Sudoku = new Sudoku().ToString();

const string ip = "127.0.0.1";

const int port = 80;

EndPoint NewSenderEndPoint = new IPEndPoint(IPAddress.Any, 0);

EndPoint udpEndPoint = new IPEndPoint(IPAddress.Parse(ip), port);

Socket udpSocket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);

List<EndPoint> PlayersEndPoint = new List<EndPoint>();

List<Player> Players = new List<Player>();

Player CurPlayer;

int CurPlayerId;

public string[] message;

EndPoint LastMove;

int Score;

public Server()

{

udpSocket.Bind(udpEndPoint);

byte[] buffer = new byte[200];

while(true)

{

EndPoint CurSenderEndPoint = NewSenderEndPoint;

udpSocket.ReceiveFrom(buffer, ref CurSenderEndPoint);

CurPlayerId = PlayersEndPoint.IndexOf(CurSenderEndPoint);

if (CurPlayerId == -1)

{

PlayersEndPoint.Add(CurSenderEndPoint);

Players.Add(new Player());

CurPlayerId = Players.Count - 1;

}

message = Encoding.Default.GetString(buffer).Split(' ');///////

Console.WriteLine("");

Console.WriteLine(Encoding.Default.GetString(buffer));

buffer = new byte[200];

Console.WriteLine(udpEndPoint.ToString());

switch (message[0])

{

case "Connected":

CurPlayer.PlayerEndPoint = CurSenderEndPoint;

CurPlayer.Nickname = message[1];

CurPlayer.Color = message[2];

CurPlayer.LastEntrance = DateTime.Now.ToString();

CurPlayer.Score = Players[CurPlayerId].Score;

Players[CurPlayerId] = CurPlayer;

PlayersEndPoint[CurPlayerId] = CurSenderEndPoint;

PlayersEndPoint.ForEach(CurSender =>

{

udpSocket.SendTo(Encoding.Default.GetBytes("Connected" + ' ' + CurPlayer.Nickname + ' ' + CurPlayer.Color + ' '), CurSender);

Console.WriteLine(CurSender);

});

udpSocket.SendTo(Encoding.Default.GetBytes("NewGame" + ' ' + Sudoku + ' '), CurPlayer.PlayerEndPoint);

break;

case "Disconnected":

PlayersEndPoint.ForEach(CurSender =>

{

if (!CurSender.Equals(CurSenderEndPoint))

{

udpSocket.SendTo(Encoding.Default.GetBytes("Disconnected" + ' ' + Players[CurPlayerId].Nickname + ' ' + Players[CurPlayerId].Color + ' '), CurSender);

Console.WriteLine(CurSender.ToString());

}

});

break;

case "Exit":

CurPlayer = Players[CurPlayerId];

Players.RemoveAt(CurPlayerId);

PlayersEndPoint.RemoveAt(CurPlayerId);

if (CurPlayer.Nickname != "")

PlayersEndPoint.ForEach(CurSender =>

{

udpSocket.SendTo(Encoding.Default.GetBytes("Exit" + ' ' + CurPlayer.Nickname + ' ' + CurPlayer.Color + ' '), CurSender);

Console.WriteLine(CurSender.ToString());

});

break;

case "EnterDigit":

int i = int.Parse(message[1]);

int j = int.Parse(message[2]);

try

{

checkValidity(message[3][0], j, i);

if (new Sudoku().Check(Sudoku))

{

if (LastMove == null || PlayersEndPoint.IndexOf(LastMove) != CurPlayerId)

{

LastMove = PlayersEndPoint[CurPlayerId];

Score = 0;

}

Score += 50;

CurPlayer.PlayerEndPoint = LastMove;

CurPlayer.Nickname = Players[CurPlayerId].Nickname;

CurPlayer.Color = Players[CurPlayerId].Color;

CurPlayer.LastEntrance = Players[CurPlayerId].LastEntrance;

CurPlayer.Score = Players[CurPlayerId].Score + Score;

Players[CurPlayerId] = CurPlayer;

PlayersEndPoint[CurPlayerId] = LastMove;

Sudoku = Sudoku.Remove(i \* 9 + j, 1).Insert(i \* 9 + j, message[3]);

PlayersEndPoint.ForEach(CurSender =>

{

udpSocket.SendTo(Encoding.Default.GetBytes("EnterDigit" + ' ' + Players[CurPlayerId].Nickname + ' ' + Players[CurPlayerId].Color + ' ' + i + ' ' + j + ' ' + message[3] + ' ' + Score.ToString() + ' '), CurSender);

Console.WriteLine(CurSender);

});

if (Sudoku.IndexOf('0') == -1)

{

PlayersEndPoint.ForEach(CurSender =>

{

udpSocket.SendTo(Encoding.Default.GetBytes("EndGame" + ' '), CurSender);

Console.WriteLine(CurSender);

});

Sudoku = new Sudoku().ToString();

PlayersEndPoint.ForEach(CurSender =>

{

udpSocket.SendTo(Encoding.Default.GetBytes("NewGame" + ' ' + Sudoku + ' '), CurSender);

Console.WriteLine(CurSender);

});

}

}

else

{

throw new Exception("One");

}

}

catch (Exception e)

{

if (LastMove == PlayersEndPoint[CurPlayerId])

{

LastMove = null;

}

PlayersEndPoint.ForEach(CurSender =>

{

udpSocket.SendTo(Encoding.Default.GetBytes("Error" + ' ' + Players[CurPlayerId].Nickname + ' ' + Players[CurPlayerId].Color + ' ' + i + ' ' + j + ' ' + message[3] + ' ' + e.Message + ' '), CurSender);

Console.WriteLine(CurSender);

});

}

break;

case "GetStatistics":

Players.Sort((a, b) => a.Score.CompareTo(b.Score));

Players.ForEach(CurPlayer =>

{

udpSocket.SendTo(Encoding.Default.GetBytes("GetStatistics" + ' ' + CurPlayer.Nickname + ' ' + CurPlayer.Color + ' ' + CurPlayer.LastEntrance + ' ' + CurPlayer.Score.ToString() + ' '), CurSenderEndPoint);

Console.WriteLine(CurSenderEndPoint);

});

break;

}

Console.WriteLine("");

}

}

public bool checkValidity(char val, int x, int y)

{

for (int i = 0; i < 9; i++)

{

if (Sudoku[y \* 9 + i] == val)

throw new Exception("Vertical");

else if (Sudoku[i \* 9 + x] == val)

throw new Exception("Horizontal");

}

int startX = (x / 3) \* 3;

int startY = (y / 3) \* 3;

for (int i = startY; i < startY + 3; i++)

{

for (int j = startX; j < startX + 3; j++)

{

if (Sudoku[i \* 9 + j] == val)

throw new Exception("Square");

}

}

return true;

}

public static void Main(String[] args)

{

new Server();

Console.Read();

}

}

class Sudoku

{

public bool OneSolution;

public Random CurRandom = new Random();

private char[] Grid;

public string CurSudoku = "";

public Sudoku()

{

CreateBase();

if (CurRandom.Next(2) == 1)

{

Transposing();

}

for (int i = 0; i < CurRandom.Next(5); i++)

{

SwapColums(CurRandom.Next(3), CurRandom.Next(3), CurRandom.Next(3));

}

for (int i = 0; i < CurRandom.Next(5); i++)

{

SwapRows(CurRandom.Next(3), CurRandom.Next(3), CurRandom.Next(3));

}

for (int i = 0; i < CurRandom.Next(5); i++)

{

SwapColumsArea(CurRandom.Next(3), CurRandom.Next(3));

}

for (int i = 0; i < CurRandom.Next(5); i++)

{

SwapRowsArea(CurRandom.Next(3), CurRandom.Next(3));

}

Solve();

}

public bool Check(string s)

{

OneSolution = false;

Grid = s.ToCharArray();

try

{

PlaceNumber(0);

}

catch (Exception e)

{

if (e.Message == "Sudoku is wrong")

{

}

}

if (OneSolution)

return true;

else

return false;

}

public void CreateBase()

{

for (int i = 0; i < 9; i++)

{

for (int j = 0; j < 9; j++)

{

CurSudoku += ((i \* 3 + i / 3 + j) % 9 + 1).ToString();

}

}

}

public void Transposing()

{

string[] TimeSudoku = new string[9];

for (int i = 0; i < 9; i++)

{

TimeSudoku[i] = CurSudoku.Substring(i \* 9, 9);

}

CurSudoku = "";

for (int i = 0; i < 9; i++)

for (int j = 0; j < 9; j++)

{

CurSudoku += TimeSudoku[j][i];

}

}

public void SwapRows(int Area, int Row1, int Row2)

{

if (Row1 != Row2)

{

string[] TimeSudoku = new string[9];

for (int i = 0; i < 9; i++)

{

TimeSudoku[i] = CurSudoku.Substring(i \* 9, 9);

}

TimeSudoku[Area \* 3 + Row1] += TimeSudoku[Area \* 3 + Row2];

TimeSudoku[Area \* 3 + Row2] = TimeSudoku[Area \* 3 + Row1].Substring(0, 9);

TimeSudoku[Area \* 3 + Row1] = TimeSudoku[Area \* 3 + Row1].Substring(9);

CurSudoku = "";

for (int i = 0; i < 9; i++)

{

CurSudoku += TimeSudoku[i];

}

}

}

public void SwapColums(int Area, int Row1, int Row2)

{

if (Row1 != Row2)

{

Transposing();

SwapRows(Area, Row1, Row2);

Transposing();

}

}

public void SwapRowsArea(int Area1, int Area2)

{

if (Area1 != Area2)

{

string PartArea1 = CurSudoku.Substring(Area1 \* 9 \* 3, 9 \* 3);

string PartArea2 = CurSudoku.Substring(Area2 \* 9 \* 3, 9 \* 3);

CurSudoku = CurSudoku.Remove(Area1 \* 9 \* 3, 9 \* 3).Insert(Area1 \* 9 \* 3, PartArea2)

.Remove(Area2 \* 9 \* 3, 9 \* 3).Insert(Area2 \* 9 \* 3, PartArea1);

}

}

public void SwapColumsArea(int Area1, int Area2)

{

if (Area1 != Area2)

{

Transposing();

SwapRowsArea(Area1, Area2);

Transposing();

}

}

public void Solve()

{

bool[] Views = new bool[81];

bool SudokuIsWrong;

int Position;

string CurValue;

int EmptyCells = 0;

do

{

while(Views[Position = CurRandom.Next(81)])

{

}

Views[Position] = true;

CurValue = CurSudoku[Position].ToString();

Grid = (CurSudoku = CurSudoku.Remove(Position, 1).Insert(Position, "0")).ToCharArray();

OneSolution = false;

SudokuIsWrong = false;

try

{

PlaceNumber(0);

}

catch (Exception e)

{

if (e.Message == "Sudoku is wrong")

{

SudokuIsWrong = true;

}

}

if (OneSolution && !SudokuIsWrong)

{

EmptyCells++;

}

if (!OneSolution || SudokuIsWrong)

{

CurSudoku = CurSudoku.Remove(Position, 1).Insert(Position, CurValue);

}

}

while (!OneSolution || SudokuIsWrong || EmptyCells < 1);

Grid = CurSudoku.ToCharArray();

}

public void PlaceNumber(int pos)

{

if (pos == 81)

{

if (!OneSolution)

{

OneSolution = true;

}

else

{

throw new Exception("Sudoku is wrong");

}

}

else

{

if (Grid[pos] > '0')

{

PlaceNumber(pos + 1);

}

else

{

for (char Candidate = '1'; Candidate <= '9'; Candidate++)

{

if (checkValidity(Candidate, pos % 9, pos / 9))

{

Grid[pos] = Candidate;

PlaceNumber(pos + 1);

Grid[pos] = '0';

}

}

}

}

}

public bool checkValidity(char val, int x, int y)

{

for (int i = 0; i < 9; i++)

{

if (Grid[y \* 9 + i] == val || Grid[i \* 9 + x] == val)

return false;

}

int startX = (x / 3) \* 3;

int startY = (y / 3) \* 3;

for (int i = startY; i < startY + 3; i++)

{

for (int j = startX; j < startX + 3; j++)

{

if (Grid[i \* 9 + j] == val)

return false;

}

}

return true;

}

public override string ToString()

{

return CurSudoku;

}

}

}

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КП 1–40 01 01 020 ПЗ | | | | Пояснительная записка | | | | 33 с. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| ГУИР 751003 008 СП | | | | Приложение «Судоку-онлайн». Схема | | | | Формат А1 | | | |
|  | | | | программы | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КП 1-40 01 01 020 Д1 | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Приложение  «Судоку-онлайн». Ведомость курсового  проекта |  | | | | Лист | Листов |
| Разраб. | | Стубеда В.Д. |  | 3.06.19 |  |  |  | |  | 33 | 33 |
| Пров. | | Красковски П.В. |  | 3.06.19 |  | Кафедра ПОИТ  гр. 751003 | | | | | |