Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Языки программирования (ЯП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

Программное средство решения кроссвордов «Судоку»

БГУИР КП 1-40 01 01 24 ПЗ

Студент: гр. 751003 Стубеда В.Д.

Руководитель: асс. Шостак Е.В.

Минск, 2018

**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 поможет в поиске решения даже самых замысловатых и сложных 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, решающих судоку. Большинство из них представляют собой сайты с возможностью ввести судоку онлайн или с помощью файла и решить ее.

## 1.2 Постановка задачи

Назначение приложения «Sudoku» заключается в возможности создания аккаунта и работы с ним. В программе пользователю доступны следующие возможности:

– добавление аккаунта;

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

– сортировка аккаунтов по имени, последней активности, количеству затраченных действий;

* сортировка может производиться по возрастанию или убыванию;

– редактирование и удаление выбранного аккаунта;

– сохранение выбранного решения судоку в текстовый файл;

– извлечение или сохранение условия судоку в текстовый файл;

– сохранение созданных аккаунтов в типизированный файл и последующая автоматическая загрузка аккаунтов при запуске программы.

# 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:

* запись выбранного результата решения судоку в файл;
* создание массива управляющего массива для судоку;
* запись и чтение условия судоку с помощью файла;
* рекурсивный способ решения судоку;
* рекурсивный усовершенствованный способ решения судоку;
* занесение в файл информации об аккаунтах;
* отображение истории пользования программой;

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

# 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 выбрать алгоритм, действие и файл, задать условие судоку, а также посмотреть информацию о предыдущем использовании программы.

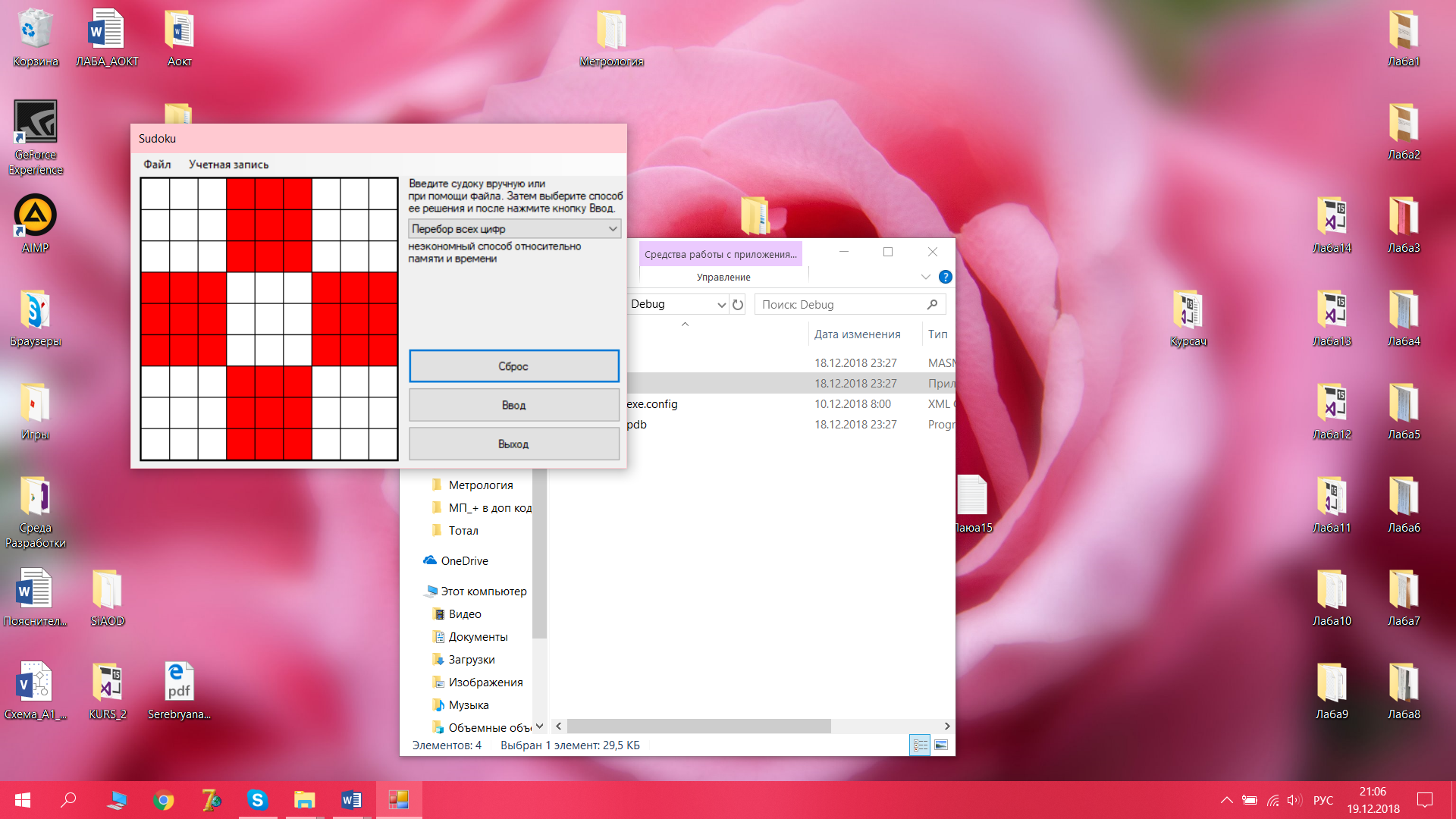


Рисунок 3.1 – Главное меню

Для работы пользователю нужно выбрать алгоритм («Перебор всех цифр», «С поиском одиночек»), задать условие судоку и нажать кнопу «Ввод». Заданное условие судоку можно сбросить с помощью кнопки «Сброс». После нажатия кнопки «Ввод» появится новое окно (рис 3.2).

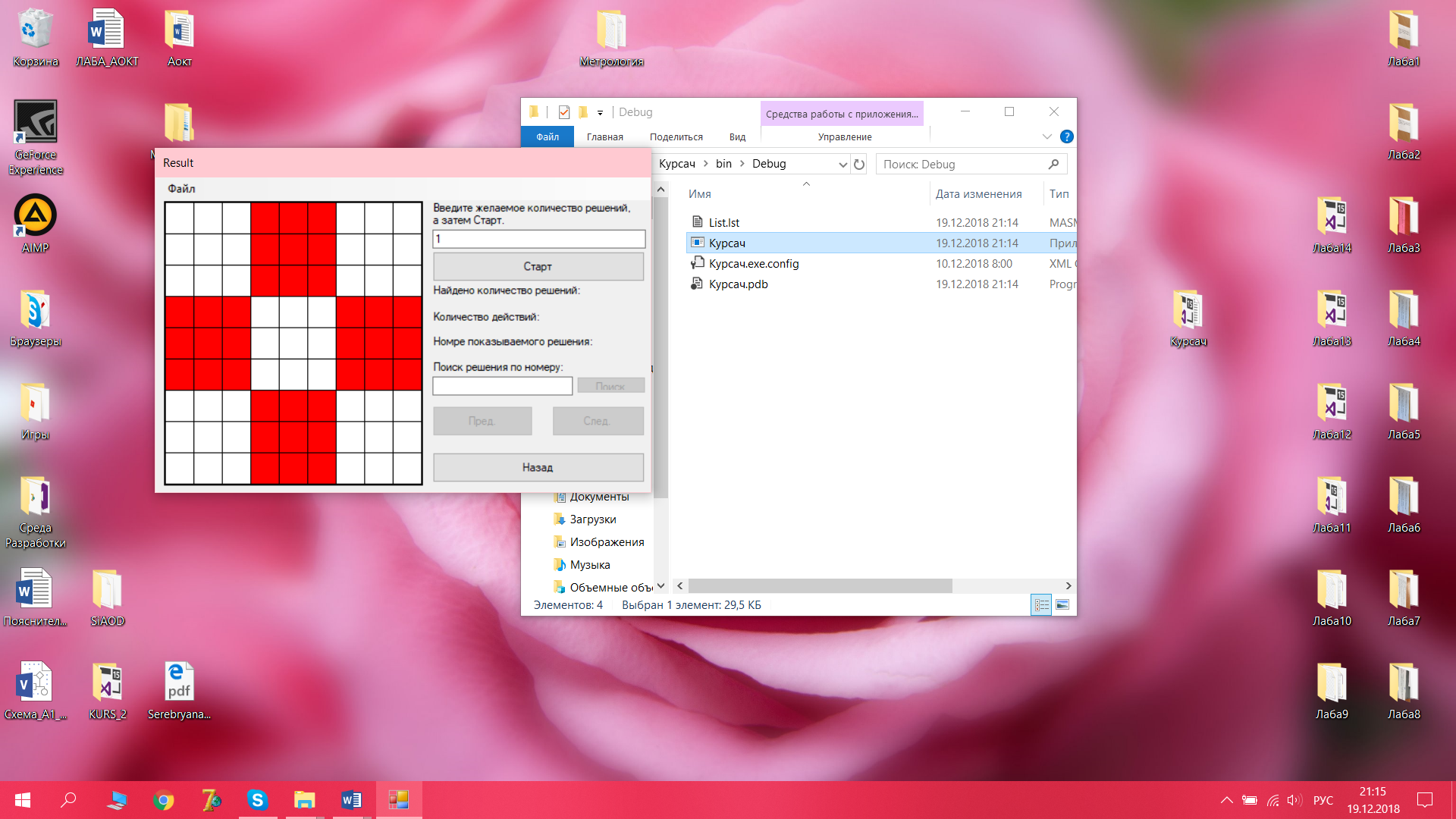


Рисунок 3.2 – Окно результата

Для запуска алгоритма нахождения решения судоку нужно нажать кнопку «Старт».

После этого, через некоторое время (в зависимости от количество указанных результатов), таблица заполнится правильным решением (рис 3.3).

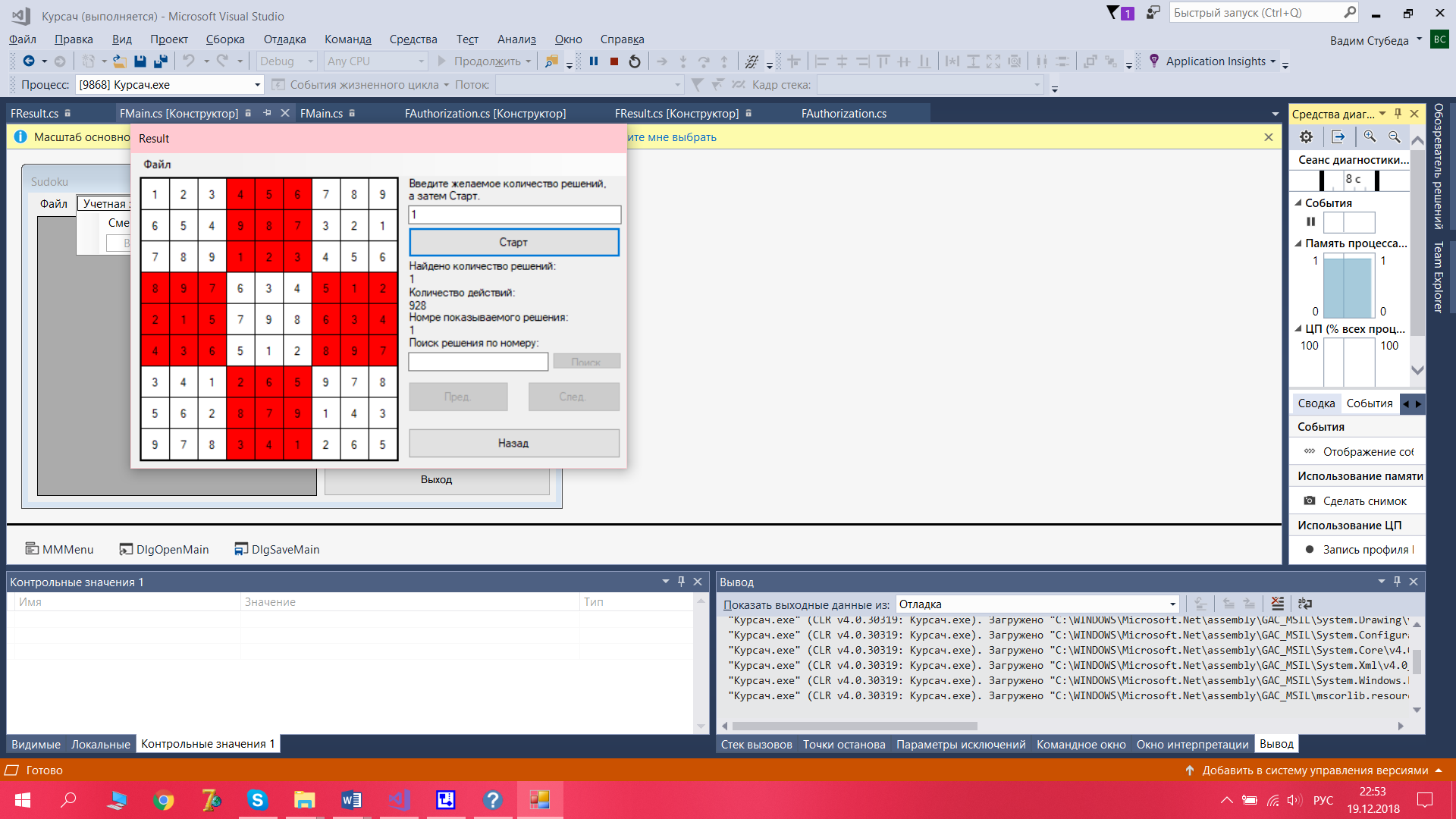


Рисунок 3.3 – Окно результата

Если решений несколько, то их можно перебирать кнопками «Пред.» и «След.». Также нужное решение можно с помощью поля «Поиск». Выбранный результат можно сохранить. Для этого нужно нажать кнопку «Файл->Сохранить». Всплывет окно, приведенное на рисунке 3.4.

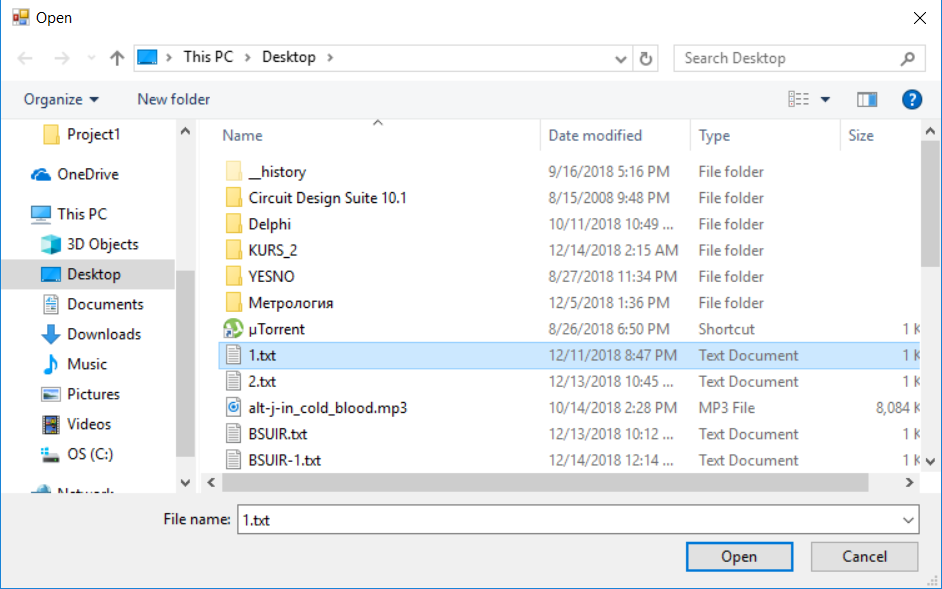


Рисунок 3.4 – Выбора файла

## 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 обычным и рекурсивным усовершенствованным способом, а также граф-схема алгоритма «Поиск «скрытых одиночек»».

Рекурсивный обычный 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

Рекурсивный усовершенствованный 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 возможных кандидатов

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

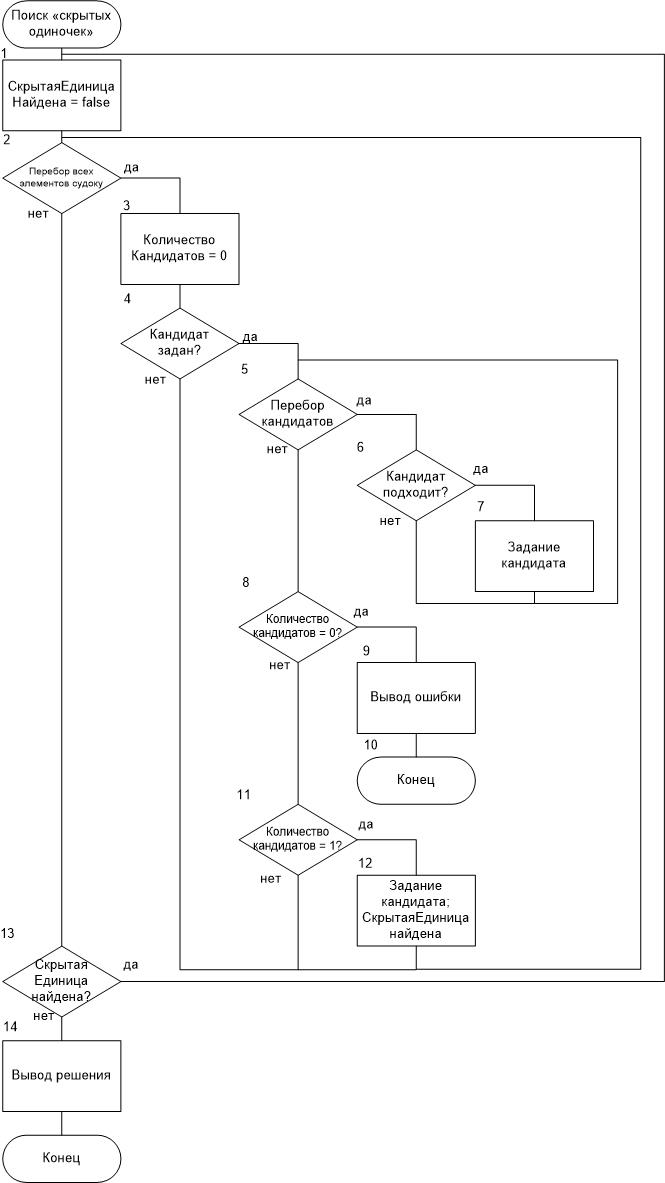


Рисунок 3.4 – Алгоритм сжатия LZW

# ****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 «Ввод», с введенной некорректной судоку | Появление сообщения о неправильности судоку | Соответствует ожидаемому |
| 2 | Открытие программы, выбор файла для извлечения судоку | Появления окна выбора файла для открытия | Соответствует ожидаемому |
| 3 | Выбор файла для сохранения судоку | Появления окна выбора файла для сохранения | Соответствует ожидаемому |
| 4 | Нажатие ссылки «Сменить пользователя» | Открытие окна с данными об использовании программы | Соответствует ожидаемому |
| 5 | Попытка воспользоваться поиском | Вывод искомого результата | 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 алгоритмпоиска решения судоку. Затем ввести судоку. После нажать кнопку «Ввод». Появится окно результата. Выбрать нужное количество решений. Нажать кнопку «Старт». Вид окна главного меню представлен на рисунке 5.1

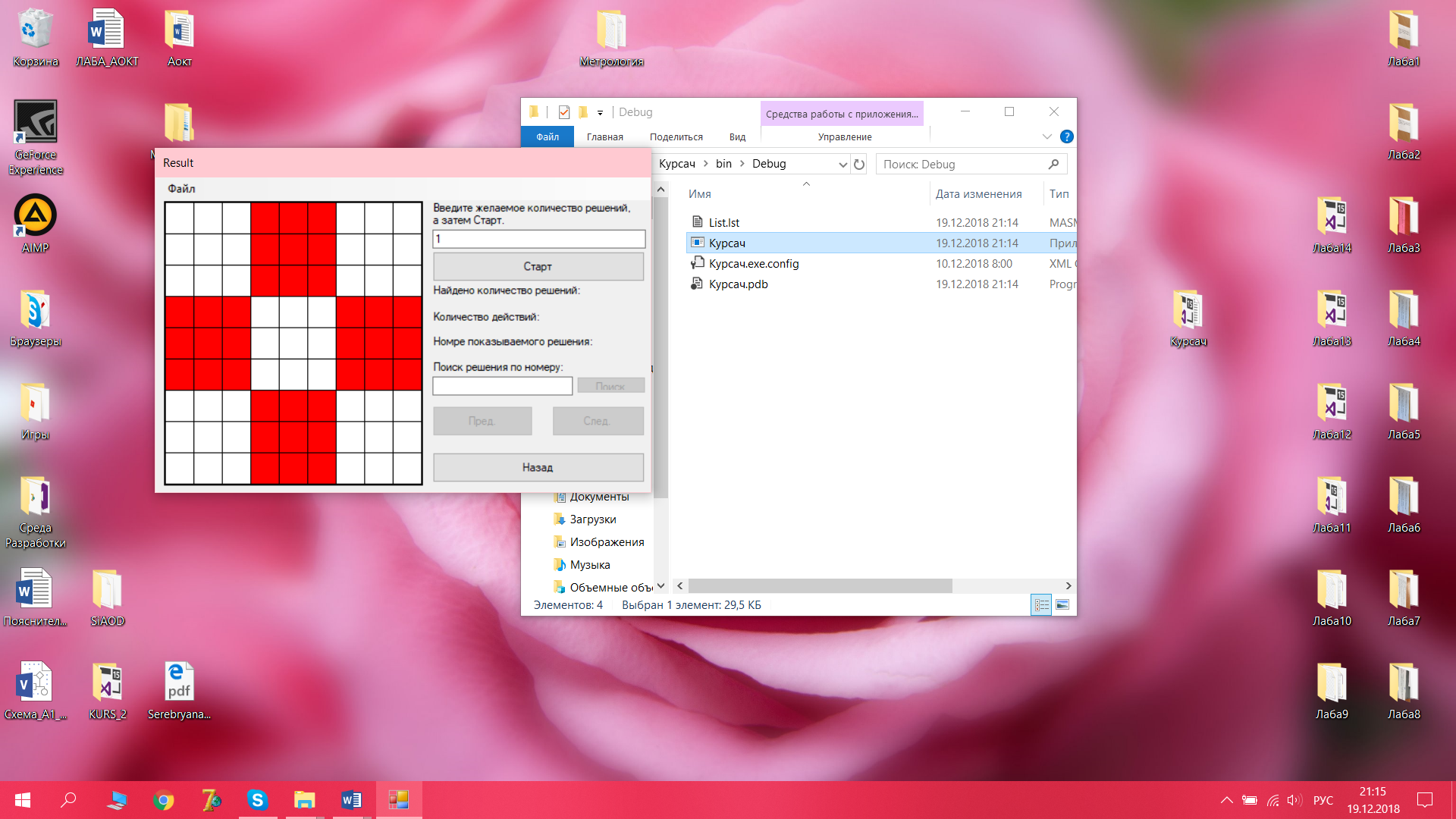


Рисунок 5.1 – Окно главного меню

## 5.3 Изучение истории использования программы

Нажать ссылку «Сменить пользователя». Всплывет окно, представленное на рисунке 5.2. В таблице представлены поля: Имя, Последнее посещение аккаунта, Количество затраченных действий пользователем.

При желании воспользоваться сортировкой, пользователю нужно нажать на заглавие выбранного столбца. Для отмены сортировки нажать еще раз на заглавие.

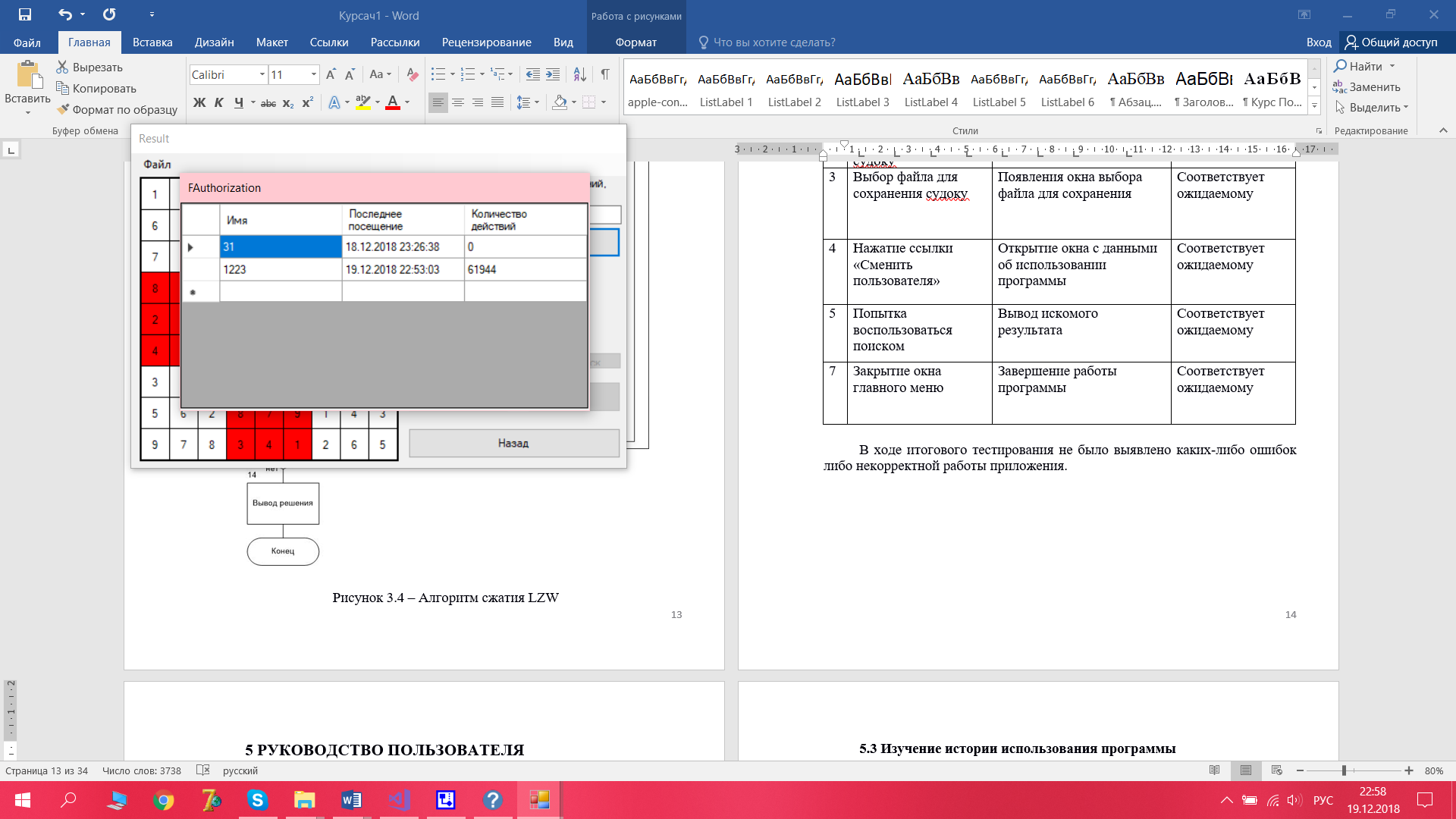


Рисунок 5.2 – Окно Аккаунтов

# ЗАКЛЮЧЕНИЕ

В ходе разработки приложения был создан продукт, предоставляющий возможность решения кроссвордов «Судоку», сохранение результатов в текстовый файл, возможность добавления, удаления, редактирования аккаунта. Так же реализована функция поиска по имени и сортировки по выбранному параметру. Данный программный продукт может быть полезен людям, которые не могут решить судоку.

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.Windows.Forms;

using System.IO;

using System.Collections.Generic;

namespace Курсач

{

public struct Zveno

{

public string Name;

public string LastEntrance;

public int Score;

}

public partial class FMain : Form

{

public int[] SourceArray = new int[81];

public int[,] DirectionArray = new int[9, 9];

public int[] NumOfCandidates = new int[81];

public int[,] Neighbors = new int[81, 20];

public int[,] ResultArrays = new int[999999, 81];

public bool Way;

List<Zveno> list = new List<Zveno>();

public string CurStr;

public static int i, j, i0, j0, m, Id;

public FMain()

{

InitializeComponent();

}

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

{

//настройа таблицы/////////////////////////////////////

DGV1.AllowUserToAddRows = false;

DGV1.AllowUserToDeleteRows = false;

DGV1.AllowUserToResizeColumns = false;

DGV1.AllowUserToResizeRows = false;

DGV1.RowHeadersVisible = false;

DGV1.ColumnHeadersVisible = false;

DGV1.ScrollBars = ScrollBars.None;

DGV1.GridColor = Color.Black;

DGV1.CellBorderStyle = DataGridViewCellBorderStyle.Single;

DGV1.AlternatingRowsDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

DGV1.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

DGV1.ColumnCount = 9;

DGV1.Rows.Add(9);

DGV1.MultiSelect = false;

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

{

DGV1.Columns[i].Width = (int)(DGV1.Width / 9);

DGV1.Rows[i].Height = (int)(DGV1.Height / 9);

}

DGV1.Width = DGV1.Columns[1].Width \* 9 + 3;

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

for (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)))

DGV1.Rows[i].Cells[j].Style.BackColor = Color.Red;

DGV1.Rows[i].Cells[j].Value = "";

}

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

// инициализация /////////////////////////////////////////

CBWay.SelectedItem = "Перебор всех цифр";

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

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

if (j % 2 == 0)

DirectionArray[i, j] = i + 9 \* j;

else

DirectionArray[i, j] = 9 - i + 9 \* j - 1;

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

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

{

m = 0;

for (i0 = 0; i0 < 9; i0++)

for (j0 = 0; j0 < 9; j0++)

if ((i == i0) & (j != j0) |

(i != i0) & (j == j0) |

((i != i0) & (j != j0) & (i / 3 == i0 / 3) & (j / 3 == j0 / 3)))

Neighbors[DirectionArray[i, j], m++] = DirectionArray[i0, j0];

}

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

}

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

{

Close();

}

private void N12Save\_Click(object sender, EventArgs e)

{

if (DlgSaveMain.ShowDialog() == DialogResult.OK)

{

FileStream CurSudoku = new FileStream(DlgSaveMain.FileName, FileMode.Create);

StreamWriter writer = new StreamWriter(CurSudoku);

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

{

CurStr = "";

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

if (DGV1.Rows[i].Cells[j].Value.ToString() == "")

CurStr += '0';

else

CurStr += DGV1.Rows[i].Cells[j].Value.ToString();

writer.Write(CurStr + "\r\n");

}

writer.Close();

}

}

private void N11Open\_Click(object sender, EventArgs e)

{

if (DlgOpenMain.ShowDialog() == DialogResult.OK)

{

FileStream CurSudoku = new FileStream(DlgOpenMain.FileName, FileMode.Open);

StreamReader reader = new StreamReader(CurSudoku);

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

{

CurStr = reader.ReadLine();

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

DGV1.Rows[i].Cells[j].Value = CurStr[j];

}

reader.Close();

}

}

private void FMain\_Shown(object sender, EventArgs e)

{

FileStream CurList = new FileStream("List.lst", FileMode.Open);

StreamReader reader = new StreamReader(CurList);

Zveno cur;

while ((cur.Name = reader.ReadLine()) != null)

{

cur.LastEntrance = reader.ReadLine();

cur.Score = Convert.ToInt32(reader.ReadLine());

list.Add(cur);

}

reader.Close();

}

private void FMain\_FormClosing(object sender, FormClosingEventArgs e)

{

FileStream CurList = new FileStream("List.lst", FileMode.Create);

StreamWriter writer = new StreamWriter(CurList);

foreach (Zveno cur in list)

{

writer.WriteLine(cur.Name);

writer.WriteLine(cur.LastEntrance);

writer.WriteLine(cur.Score.ToString());

}

writer.Close();

}

private void N21Change\_Click(object sender, EventArgs e)

{

FAuthorization Authorization = new FAuthorization(list, 0);

Authorization.ShowDialog();

}

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

{

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

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

if (DGV1.Rows[j].Cells[i].Value.ToString() == "")

SourceArray[DirectionArray[i, j]] = 0;

else

if ((DGV1.Rows[j].Cells[i].Value.ToString() == "0") |

(DGV1.Rows[j].Cells[i].Value.ToString() == "1") |

(DGV1.Rows[j].Cells[i].Value.ToString() == "2") |

(DGV1.Rows[j].Cells[i].Value.ToString() == "3") |

(DGV1.Rows[j].Cells[i].Value.ToString() == "4") |

(DGV1.Rows[j].Cells[i].Value.ToString() == "5") |

(DGV1.Rows[j].Cells[i].Value.ToString() == "6") |

(DGV1.Rows[j].Cells[i].Value.ToString() == "7") |

(DGV1.Rows[j].Cells[i].Value.ToString() == "8"))

SourceArray[DirectionArray[i, j]] = Convert.ToInt32(DGV1.Rows[j].Cells[i].Value.ToString());

else

SourceArray[DirectionArray[i, j]] = 0;

if (CBWay.SelectedItem.ToString() == "Перебор всех цифр")

Way = true;

else

Way = false;

FResult Result = new FResult(this, SourceArray, Neighbors, Way, DirectionArray, list);

if (Control(SourceArray))

{

Hide();

Result.Show();

}

else

MessageBox.Show("Судоку не может быть решена. Исправьте ее и повторите попытку.");

}

private void CBWay\_SelectedIndexChanged(object sender, EventArgs e)

{

if (CBWay.SelectedItem.ToString() == "Перебор всех цифр")

{

LSingles.Visible = false;

LRecursively.Visible = true;

}

else if (CBWay.SelectedItem.ToString() == "С поиском одиночек")

{

LSingles.Visible = true;

LRecursively.Visible = false;

}

}

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

{

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

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

DGV1.Rows[j].Cells[i].Value = "";

}

private bool Control(int[] Sudoku)

{

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

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

if ((Sudoku[i] == Sudoku[Neighbors[i, j]]) & (Sudoku[i] > 0))

return false;

return true;

}

}

}

Результат:

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;

using System.IO;

namespace Курсач

{

public partial class FResult : Form

{

public int[] SourceArray = new int[81];

public bool[] PossibleArray = new bool[81];

public int[,] DirectionArray = new int[9, 9];

public int[] NumOfCandidates = new int[81];

public int[,] Neighbors = new int[81, 20];

public int[,] ResultArrays = new int[999999, 81];

public int[,] Candidates = new int[81, 9];

public int NumOfResults, NumOfDecisions, NumOfSteps;

public bool Way;

public int i, j, i0, j0, m;

private FMain Main = new FMain();

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

{

if (ENumOfResults.Text != "")

NumOfResults = Convert.ToInt32(ENumOfResults.Text);

else

NumOfResults = 99999;

if (NumOfResults != 0)

Heart();

else

MessageBox.Show("Не может быть 0 решений. Введите натуральное число.");

FAuthorization Authorization = new FAuthorization(list, NumOfSteps);

Authorization.ShowDialog();

}

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

{

if (NumOfDecisions > 1)

{

NumOfDecisions = NumOfDecisions - 1;

Show(NumOfDecisions);

}

}

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

{

if (NumOfDecisions < NumOfResults)

{

NumOfDecisions = NumOfDecisions + 1;

Show(NumOfDecisions);

}

}

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

{

if (ESearch.Text != "")

if ((Convert.ToInt32(ESearch.Text) > 0) & (Convert.ToInt32(ESearch.Text) <= NumOfResults))

Show(NumOfDecisions = Convert.ToInt32(ESearch.Text));

else

MessageBox.Show("Решение с введенным номером не найдено");

else

MessageBox.Show("Введите номер решения.");

}

private void N12Save\_Click(object sender, EventArgs e)

{

if (NumOfDecisions != 0)

if (DlgSaveResult.ShowDialog() == DialogResult.OK)

{

FileStream CurSudoku = new FileStream(DlgSaveResult.FileName, FileMode.Create);

StreamWriter writer = new StreamWriter(CurSudoku);

string CurStr;

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

{

CurStr = "";

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

if (DGV1.Rows[i].Cells[j].Value.ToString() == "")

CurStr += '0';

else

CurStr += DGV1.Rows[i].Cells[j].Value.ToString();

writer.Write(CurStr + "\r\n");

}

writer.Close();

}

else

MessageBox.Show("Нечего сохранять!");

}

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

{

Hide();

Main.Show();

}

List<Zveno> list = new List<Zveno>();

public FResult(FMain Main, int[] SourceArray, int[,] Neighbors, bool Way, int[,] DirectionArray, List<Zveno> list)

{

InitializeComponent();

this.Main = Main;

this.SourceArray = SourceArray;

this.Neighbors = Neighbors;

this.Way = Way;

this.DirectionArray = DirectionArray;

this.list = list;

}

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

{

//настройа таблицы///////////////////////////////////////

DGV1.AllowUserToAddRows = false;

DGV1.AllowUserToDeleteRows = false;

DGV1.AllowUserToResizeColumns = false;

DGV1.AllowUserToResizeRows = false;

DGV1.RowHeadersVisible = false;

DGV1.ColumnHeadersVisible = false;

DGV1.ScrollBars = ScrollBars.None;

DGV1.GridColor = Color.Black;

DGV1.CellBorderStyle = DataGridViewCellBorderStyle.Single;

DGV1.AlternatingRowsDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

DGV1.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

DGV1.ColumnCount = 9;

DGV1.Rows.Add(9);

DGV1.MultiSelect = false;

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

{

DGV1.Columns[i].Width = (int)(DGV1.Width / 9);

DGV1.Rows[i].Height = (int)(DGV1.Height / 9);

}

DGV1.Width = DGV1.Columns[1].Width \* 9 + 3;

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

for (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)))

DGV1.Rows[i].Cells[j].Style.BackColor = Color.Red;

DGV1.Rows[i].Cells[j].Value = "";

}

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

LNumOfDecision.Text = "";

ENumOfResults.Text = "1";

LNumOfSteps.Text = "";

LNumOfDecisionShown.Text = "";

BPrev.Enabled = false;

BNext.Enabled = false;

BSearch.Enabled = false;

for (i = 0; i < 81; i++)

if (SourceArray[i] > 0)

PossibleArray[i] = true;

else

PossibleArray[i] = false;

}

private void Heart()

{

bool CandidateCheck(int Candidate, int NumOfEl)

{

int i;

i = 0;

while (i < 20)

if (Candidate == SourceArray[Neighbors[NumOfEl, i++]])

return false;

return true;

}

void RecursiveMethod(int NumOfEl)

{

int Candidate;

if (NumOfEl == 81)

{

for (i = 0; i < 81; i++)

ResultArrays[NumOfDecisions - 1, i] = SourceArray[i];

NumOfDecisions++;

}

if ((NumOfEl < 81) & (NumOfDecisions <= NumOfResults))

if (PossibleArray[NumOfEl])

RecursiveMethod(NumOfEl + 1);

else

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

{

if (CandidateCheck(Candidate, NumOfEl))

{

SourceArray[NumOfEl] = Candidate;

NumOfSteps++;

RecursiveMethod(NumOfEl + 1);

}

SourceArray[NumOfEl] = 0;

NumOfSteps++;

}

}

bool ModumFieri()

{

bool Easy;

int Candidate, NumOfEl;

do

{

Easy = false;

for (NumOfEl = 0; NumOfEl < 81; NumOfEl++)

{

NumOfCandidates[NumOfEl] = 0;

if (SourceArray[NumOfEl] == 0)

{

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

if (CandidateCheck(Candidate, NumOfEl))

Candidates[NumOfEl, NumOfCandidates[NumOfEl]++] = Candidate;

if (NumOfCandidates[NumOfEl] == 0)

return false;

if (NumOfCandidates[NumOfEl] == 1)

{

SourceArray[NumOfEl] = Candidates[NumOfEl, 0];

Easy = true;

}

}

}

}

while (Easy);

return true;

}

void RecursiveMethod\_Smart(int NumOfEl)

{

if (NumOfEl == 81)

{

for (i = 0; i < 81; i++)

ResultArrays[NumOfDecisions - 1, i] = SourceArray[i];

NumOfDecisions++;

}

if ((NumOfEl < 81) & (NumOfDecisions <= NumOfResults))

{

if (NumOfCandidates[NumOfEl] == 0)

RecursiveMethod\_Smart(NumOfEl + 1);

if (NumOfCandidates[NumOfEl] > 1)

{

for (int t = 0; t < NumOfCandidates[NumOfEl]; t++)

{

if (CandidateCheck(Candidates[NumOfEl, t], NumOfEl))

{

SourceArray[NumOfEl] = Candidates[NumOfEl, t];

NumOfSteps++;

RecursiveMethod\_Smart(NumOfEl + 1);

}

}

SourceArray[NumOfEl] = 0;

NumOfSteps++;

}

}

}

{

NumOfDecisions = 1;

NumOfSteps = 0;

if (Way)

RecursiveMethod(0);

else

if (ModumFieri())

RecursiveMethod\_Smart(0);

else

MessageBox.Show("Судоку не может быть решена. Исправьте ее и повторите попытку.");

LNumOfDecision.Text = Convert.ToString(NumOfDecisions - 1);

LNumOfSteps.Text = Convert.ToString(NumOfSteps);

NumOfDecisions = 1;

Show(NumOfDecisions);

}

}

void Show(int Id)

{

LNumOfDecisionShown.Text = Convert.ToString(Id);

if (Id == 1)

BPrev.Enabled = false;

else

BPrev.Enabled = true;

if (Id == NumOfResults)

BNext.Enabled = false;

else

BNext.Enabled = true;

if (NumOfResults == 1)

BSearch.Enabled = false;

else

BSearch.Enabled = true;

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

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

DGV1.Rows[j].Cells[i].Value = Convert.ToString(ResultArrays[Id - 1, DirectionArray[i, j]]);

}

}

}

Авторизация:

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 Курсач

{

public partial class FAuthorization : Form

{

public int i, Id;

List<Zveno> list = new List<Zveno>();

public FAuthorization(List<Zveno> list, int Id)

{

InitializeComponent();

this.list = list;

this.Id = Id;

}

private void ListToGrid()

{

list.Clear();

Zveno cur;

for (i = 0; i < dataGridView1.RowCount - 1; i++)

{

cur.Name = dataGridView1.Rows[i].Cells[0].Value.ToString();

cur.LastEntrance = dataGridView1.Rows[i].Cells[1].Value.ToString();

cur.Score = Convert.ToInt32(dataGridView1.Rows[i].Cells[2].Value.ToString());

list.Add(cur);

}

}

private void dataGridView1\_UserAddedRow(object sender, DataGridViewRowEventArgs e)

{

dataGridView1.Rows[dataGridView1.RowCount - 2].Cells[1].Value = DateTime.Now.ToString();

dataGridView1.Rows[dataGridView1.RowCount - 2].Cells[2].Value = "0";

}

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

{

foreach (Zveno cur in list)

dataGridView1.Rows.Add(cur.Name, cur.LastEntrance, cur.Score.ToString());

}

private void dataGridView1\_RowHeaderMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (dataGridView1.CurrentRow.Index < dataGridView1.RowCount - 1)

{

dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[1].Value = DateTime.Now.ToString();

dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[2].Value = Convert.ToInt32(Convert.ToInt32(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[2].Value) + Id);

ListToGrid();

Hide();

}

}

}

}

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КП 1–40 01 01 020 ПЗ | | | | Пояснительная записка | | | | 38 с. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| ГУИР 751003 008 СП | | | | Приложение «Сетевой чат». Схема | | | | Формат А1 | | | |
|  | | | | программы | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КП 1-40 01 01 020 Д1 | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Приложение «Сетевой чат». Ведомость курсового  проекта |  | | | | Лист | Листов |
| Разраб. | | Сергей П.Д |  | 13.12.18 |  |  |  | |  | 38 | 38 |
| Пров. | | Шостак Е.В. |  | 26.11.18 |  | Кафедра ПОИТ  гр. 751003 | | | | | |