Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

Программное средство решения кроссвордов «Судоку»

БГУИР КП I–40 01 01 326 ПЗ

Выполнил

студент: гр. 751003 Стубеда В.Д.

Проверил: Фадеева Е.П.

Минск 2018

Учреждение образования

«Белорусский государственный университет информатики и

радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

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

(подпись)

Лапицкая Н.В. 2018г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Стубеде Вадиму Дмитриевичу* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема работы *Программное средство решения кроссвордов «Судоку» \_\_\_*

2. Срок сдачи законченной работы *11.06.2018г.*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе *Среда программирования Delphi. Реализовать работу с типизированным файлом для хранения данных о пользователях следующей структуры: имя, последняя активность, количество затраченных действий. Реализовать двунаправленный динамический список для работы с типизированным файлом. Использовать текстовый файлы для хранения судоку и извлечения ее. Предусмотреть контроль корректности ввода исходных данных при добавлении нового объекта в динамический список. Реализовать следующие алгоритмы: загрузка данных типизированного файла в динамическую структуру при запуске программы; перезапись новой версии типизированного файла при сохранении изменений пользователем; добавление новых объектов в динамическую структуру; удаление объектов из динамической структуры; редактирование объектов в динамической структуре; поиск в динамической структуре по введённым данным; сортировка динамической структуры по выбранным параметрам; сохранение выбранного решения судоку в текстовый файл.*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*1 Анализ литературных источников\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*2 Постановка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*3Разработка программного средства\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*4 Тестирование и проверка работоспособности программного средства\_\_\_\_*

*5 Руководство по установке и использованию программного средства\_\_\_\_\_\_*

*Заключение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Список использованных источников\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Приложения* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*Схема алгоритма в формате А1*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультант по курсовой работе *Фадеева Е.П.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

7.Дата выдачи задания *16.02.2018г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

*Раздел 1. Введение к 28.02.2018г. – 10 % готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 2 к 15.03.2018г. – 30% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 3 к 15.04.2018г. – 60% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 4 к 10.05.2018г. – 80% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 5.Заключение. Приложения к 20.05.2018г. – 90% готовности работы;*

*оформление пояснительной записки и графического материала к 01.06.2018г. – 100% готовности работы.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Защита курсового проекта с 02.06.2018г. по 11.06.2018г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

РУКОВОДИТЕЛЬ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Фадеева Е.П.*

*(подпись)*

Задание принял к исполнению *Стубеда В.Д, 16.02.2018г.*

*(дата и подпись студента)*

Содержание

[Введение 6](#_Toc516198982)

[1 Анализ литературных источников 7](#_Toc516198983)

[1.1 Анализ существующих приложений 7](#_Toc516198984)

[2 Постановка задачи 8](#_Toc516198985)

[3 Разработка программного средства 9](#_Toc516198986)

[3.1 Описание алгоритмов 9](#_Toc516198987)

[3.2 Структура данных 11](#_Toc516198988)

[3.2.1 Структура типов основного алгоритма 11](#_Toc516198989)

[3.2.2 Структура данных программы 12](#_Toc516198990)

[3.2.11 Описание графических компонентов формы FMain 14](#_Toc516198991)

[3.2.12 Описание графических компонентов формы FResult 16](#_Toc516198992)

[3.2.13 Описание графических компонентов формы FAuthorization 18](#_Toc516198993)

[3.2.14 Описание графических компонентов формы FCreateAccount 19](#_Toc516198994)

[3.2.15 Описание графических компонентов формы FStatistics 20](#_Toc516198995)

[3.3 Схема алгоритмов решения задачи по ГОСТ 19.701-90 21](#_Toc516198996)

[3.3.1 Схема алгоритма DataToList 21](#_Toc516198997)

[3.3.2 Схема алгоритма DataToFile 22](#_Toc516198998)

[3.3.3 Схема алгоритма ListToGrid 23](#_Toc516198999)

[3.3.4 Схема алгоритма DelUser 25](#_Toc516199000)

[3.3.5 Схема алгоритма CreateNew 26](#_Toc516199001)

[3.3.6 Схема алгоритма Sorting 28](#_Toc516199002)

[3.3.7 Схема алгоритма Search 30](#_Toc516199003)

[4 Тестирование и проверка работоспособности программного средства 31](#_Toc516199004)

[4.1 Просмотр аккаунтов 31](#_Toc516199005)

[4.1.1 Тест 1 31](#_Toc516199006)

[4.1.2 Тест 2 32](#_Toc516199007)

[4.2 Сортировка по выбранному параметру 33](#_Toc516199008)

[4.2.1 Тест 3 33](#_Toc516199009)

[4.2.2 Тест 4 34](#_Toc516199010)

[4.2.3 Тест 5 35](#_Toc516199011)

[4.3 Поиск по введённым данным 36](#_Toc516199012)

[4.3.1 Тест 6 36](#_Toc516199013)

[4.3.1 Тест 7 37](#_Toc516199014)

[4.3.1 Тест 8 38](#_Toc516199015)

[4.4 Создание аккаунта 39](#_Toc516199016)

[4.5.1 Тест 10 39](#_Toc516199017)

[4.5.2 Тест 11 40](#_Toc516199018)

[4.5.3 Тест 12 41](#_Toc516199019)

[4.5 Удаление аккаунта из списка 42](#_Toc516199020)

[4.6.1 Тест 13 42](#_Toc516199021)

[4.6.2 Тест 14 43](#_Toc516199022)

[4.1 Редактирование аккаунта из списка 44](#_Toc516199023)

[4.7.1 Тест 15 44](#_Toc516199024)

[4.7.2 Тест 16 45](#_Toc516199025)

[5 Руководство по установке и использованию программного средства 46](#_Toc516199026)

[5.1 Шаг 1. Установка 46](#_Toc516199027)

[5.2 Шаг 2. Авторизация 47](#_Toc516199028)

[5.3 Шаг 3. Ввод судоку в таблицу 48](#_Toc516199029)

[5.4 Шаг 4. Выбор способа решения и переход к результатам 49](#_Toc516199030)

[5.5 Шаг 5. Выбор количества искомых решений и отображение результатов 50](#_Toc516199031)

[5.6 Шаг 6. Сохранение выбранного результата 51](#_Toc516199032)

[5.7 Шаг 7. Выход из программы 52](#_Toc516199033)

[Заключение 53](#_Toc516199034)

[Список использованной литературы 54](#_Toc516199035)

[Приложение А 55](#_Toc516199036)

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.

Правильно составленная головоломка имеет только одно решение. Тем не менее, на некоторых сайтах в интернете под видом усложнённых головоломок пользователю предлагаются варианты судоку с несколькими вариантами решения, а также с ветвлениями самого хода решения.

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 Delphi, а средой разработки – Borland Delphi 7, которые изучаются в рамках предмета «Основы алгоритмизации и программирования».

# Анализ литературных источников

## 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.

# 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;

– перемещение аккаунтов в таблице;

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

* сортировка может производиться по возрастанию или убыванию;

– редактирование и удаление выбранного аккаунта;

– поиск аккаунта по имени или части имени;

– 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.

# Разработка программного средства

## Описание алгоритмов

Таблица – Описание основной программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование  алгоритма | Назначение  алгоритма | Формальные  параметры |  |
|  | Основной алгоритм | Создание динамической структуры |  |  |
|  | DataToList | Считывает данные из файла с именем Users и заносит их в список, начиная с адреса FirstUser |  | Процедура |
|  | ListToGrid | Вывод элементов из списка в SGAccounts, начиная с адреса FirstUser |  | Процедура |
|  | DelUser | Удаляет элемент с адресом CurUser из списка, который начинается с FirstUser |  | Процедура |
|  | DataToFile | Заносит информацию из списка, который начинается с FirstUser в файл с именем Users |  | Процедура |
|  | CreatNew | Добавляет новый аккаунт с именем из EName.Text в список, который начинается с FirstUser |  | Процедура |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sorting (NumOfSorting; Way; Sender) | Сортирует элементы в списке, который начинается с FirstUser по возрастанию/убыванию в зависимости от значения Way(если Way = true, то по возрастанию, иначе – по убыванию) и по параметру NubOfSorting | Way –параметр- значение, NumOfSorting – параметр-  значение | Процедура |
|  | Search | Ищет аккаунт в списке, который начинается с FirstUser, c именем или частью имени в ЕSearch.Text и выводит результат в SGAccounts |  | Процедура |

Продолжение таблицы 1

## Структура данных

### Структура типов основного алгоритма

При разработке программного средства использовалась такая динамическая структура данных, как двунаправленный список хранящий подробную информацию об аккаунтах.

Таблица 2 – Структура типов программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| PUser | ^TUser | Тип, предназначенный для представления информации об аккаунтах. |
| TUser | Record  Name: String[12];  LastEntrance:  TDateTime;  Score: Integer;  Next, Prev: PUser; | Name – Имя пользователя;  LastEntrance – последнее посещение;  Score – количество затраченных действий;  Next – следующий аккаунт;  Prev – предыдущий аккаунт. |
| TMainArray | Array [1..81] of  Integer; | Основной тип для хранения  судоку. |
| TDirectionArray | Array [0..8, 0..8] of Integer; | Тип для хранения маски для преобразования судоку из одномерного массива, в двумерный для представления в табличном виде. |
| TNeighbors | Array [1..81, 1..20] of Integer; | Тип для хранения 20 элементов, соответствующих ячейке i. |
| TPossibleArray | Array [1..81] of Boolean; | Тип для указывания на занятость ячейки. |
| TCandidates | Array [1..81, 1..9] of Integer; | Тип для хранения возможных кандидатов для ячейки k. |

### Структура данных программы

Таблица 3 – Структура данных основного алгоритма

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| FMain | TFMain | Обеспечение графического интерфейса главной страницы программы |
| FResult | TFResult | Обеспечение графического интерфейса вспомогательной страницы программы, отвечающую за результат |
| FirstUser | PUser | Адрес первого элемента в списке аккаунтов. |
| CurUser | PUser | Адрес аккаунта, параметры которого будут редактироваться. |
| TimeUser | PUser | Адрес аккаунта, который используется для переноса или удаления записи. |
| SourceArray | TMainArray | Исходная судоку. |
| ResultArrays | Аrray [1..999999] of TMainArray; | Массив решений судоку. |
| DirectionArray | TDirectionArray | Маска для преобразования судоку из одномерного массива, в двумерный для представления в табличном виде. |
| NumOfCandidates | TMainArray | Количество кандидатов для ячейки i. |
| Neighbors | TNeighbors | Возможные кандидаты для ячейки k. |
| Users | file of TUser | Типизированный файл для хранения аккаунтов. |
| CurSudoku | TextFile; | Текстовый файл для хранения судоку |
| CurStr | String; | Переменная для извлечения и записи судоку в текстовый файл |
| Authorized | String | Переменная хранящая имя текщего пользователя. |
| i | Integer | Переменная для цикла. |
| j | Integer | Переменная для цикла. |
| i0 | Integer | Переменная для цикла. |
| j0 | Integer | Переменная для цикла. |
| m | Integer | Переменная для цикла. |

Структура данных алгоритма Sorting (NumOfSort;Way;Sender)

Таблица 9 – Структура данных алгоритма Sort

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Комментарии |
| NumOfSorting | Integer | В зависимости от значения этой переменной данные будут сортироваться по одному из параметров(1-имени, 2-по последнему посещению, 3-по количеству затраченных действий) | Формальный |
| Way | Boolean | Условие сортировки(по убыванию/ по возростанию) | Формальный |
| f1 | Boolean | Флаг | Локальный |

Для организации графического интерфейса программного средства «Sudoku» было использовано 5 форм: FMain, FResul, FAuthorization, FCreateAccount, FStatistics. Взаимосвязь между формами представлена на рисунке 3.1.



Рисунок 3.1 – Взаимосвязь графических форм

### Описание графических компонентов формы FMain

Форма главной страницы программы FMain предоставляет возможность ввода судоку, ввода судоку, извлечение судоку из файла, запись созданной судоку в файл, выбора способа решения судоку. В зависимости от действий пользователя, форма может иметь виды, представленные на рисунках 3.2 и 3.3.



Рисунок 3.2 – Вид формы FMain

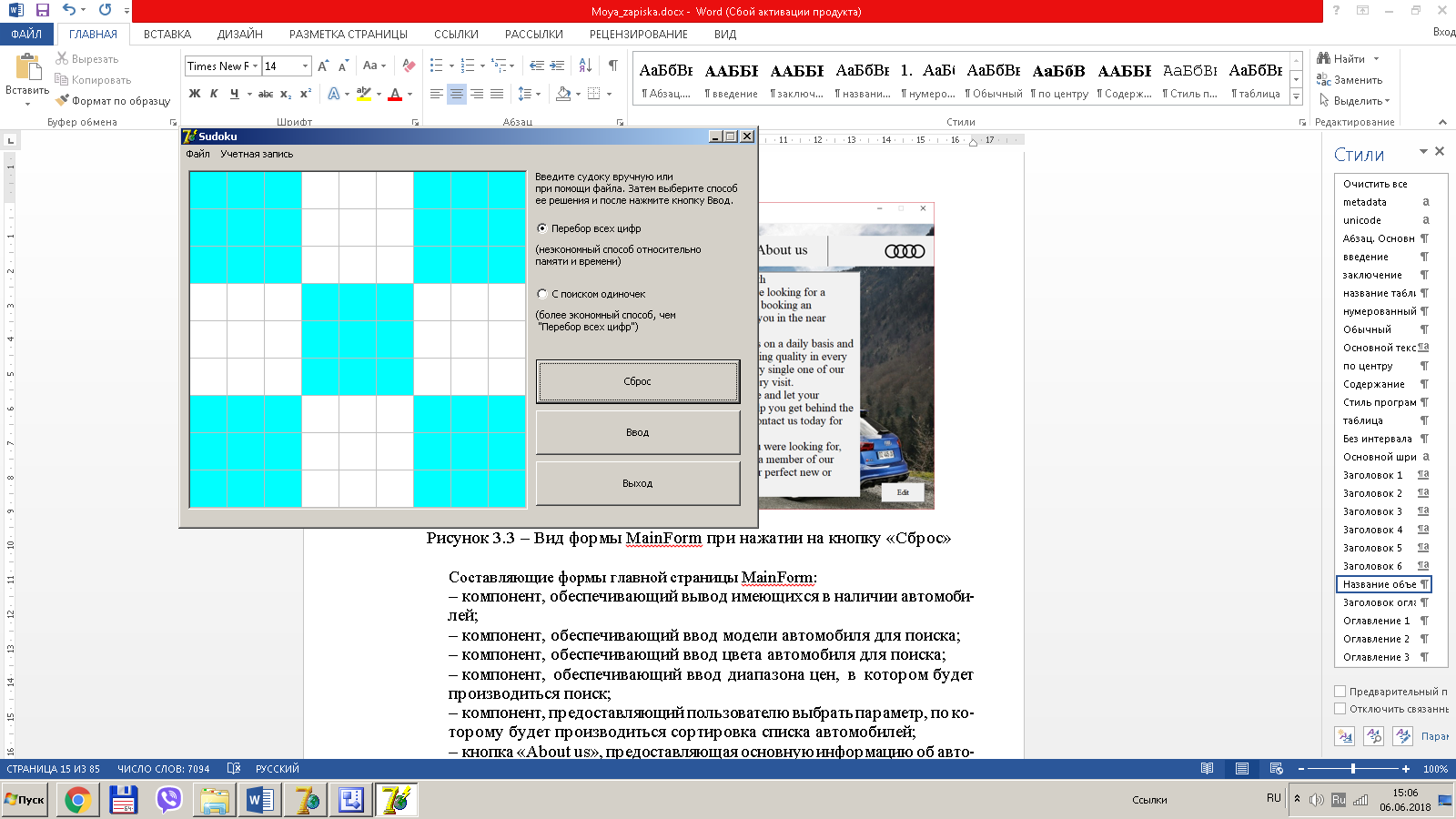


Рисунок 3.3 – Вид формы FMain при нажатии на кнопку «Сброс»

Составляющие формы главной страницы FMain:

* компонент, обеспечивающий ввод судоку;
* кнопка «Сброс», предоставляющая возможность очистить поле ввода судоку;
* кнопка «Ввод», позволяющая перейти к результатам решения судоку;
* кнопка «Выход», позволяющая закрыть программу;
* кнопка «Извлечь судоку из файла», предоставляющая пользователю возможность извлечь судоку из файла;
* кнопка «Сохранить судоку в файл», предоставляющая пользователю возможность сохранить судоку в файлы;
* кнопка «Сменить пользователя», позволяющая перейти в меню смены пользователя;
* кнопка «Статистика», позволяющая перейти к статистике пользователей;

### Описание графических компонентов формы FResult

Вспомогательная форма программы FResult предоставляет возможность просмотра выбранного количества решения судоку, переключения между правильными ответами, поиск решения по его номеру, сохранения выбранного решения судоку в текстовый файл. В зависимости от действий пользователя, форма может иметь виды, представленные на рисунках 3.4 и 3.5.

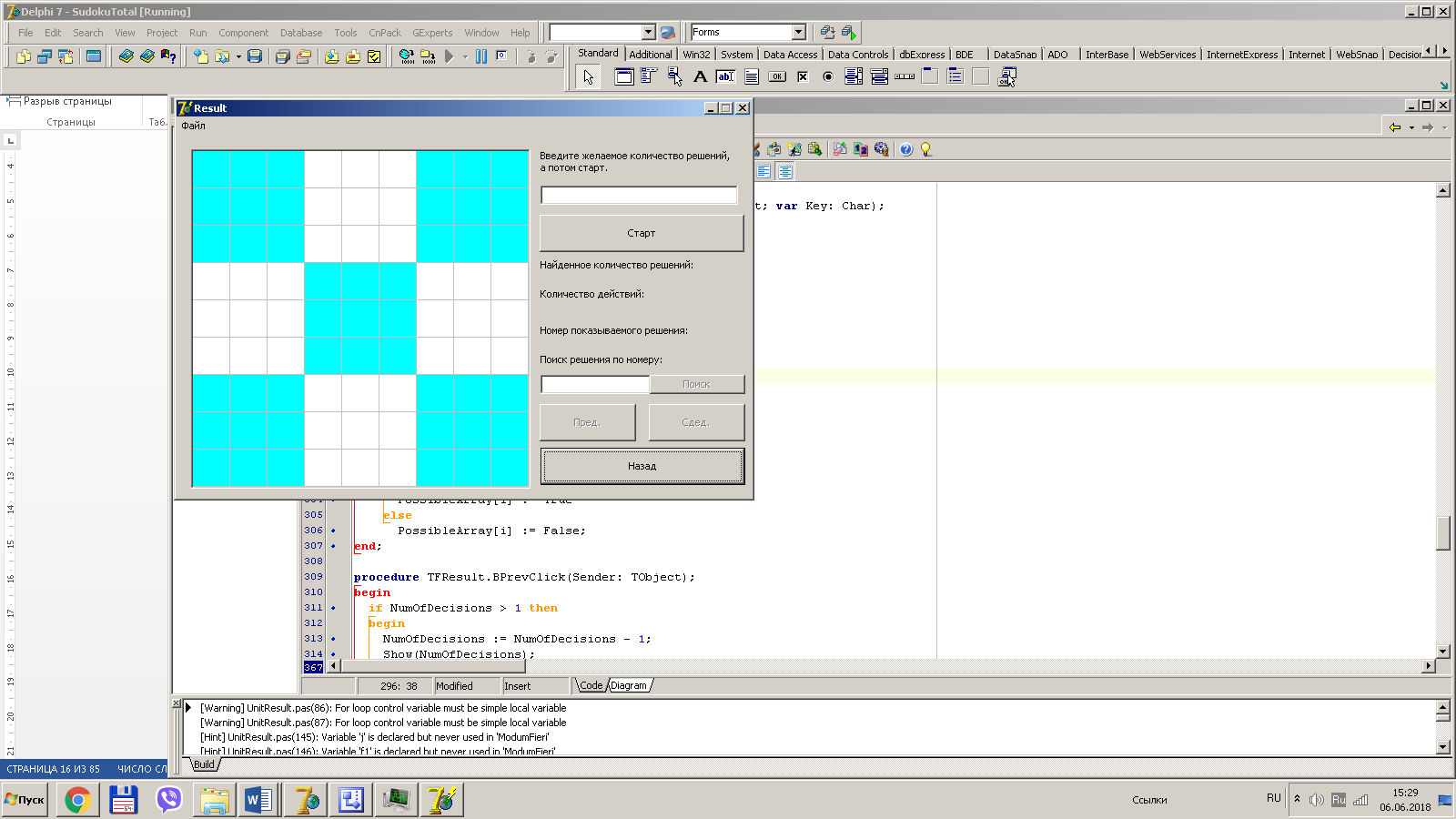


Рисунок 3.4 – Вид формы FResult

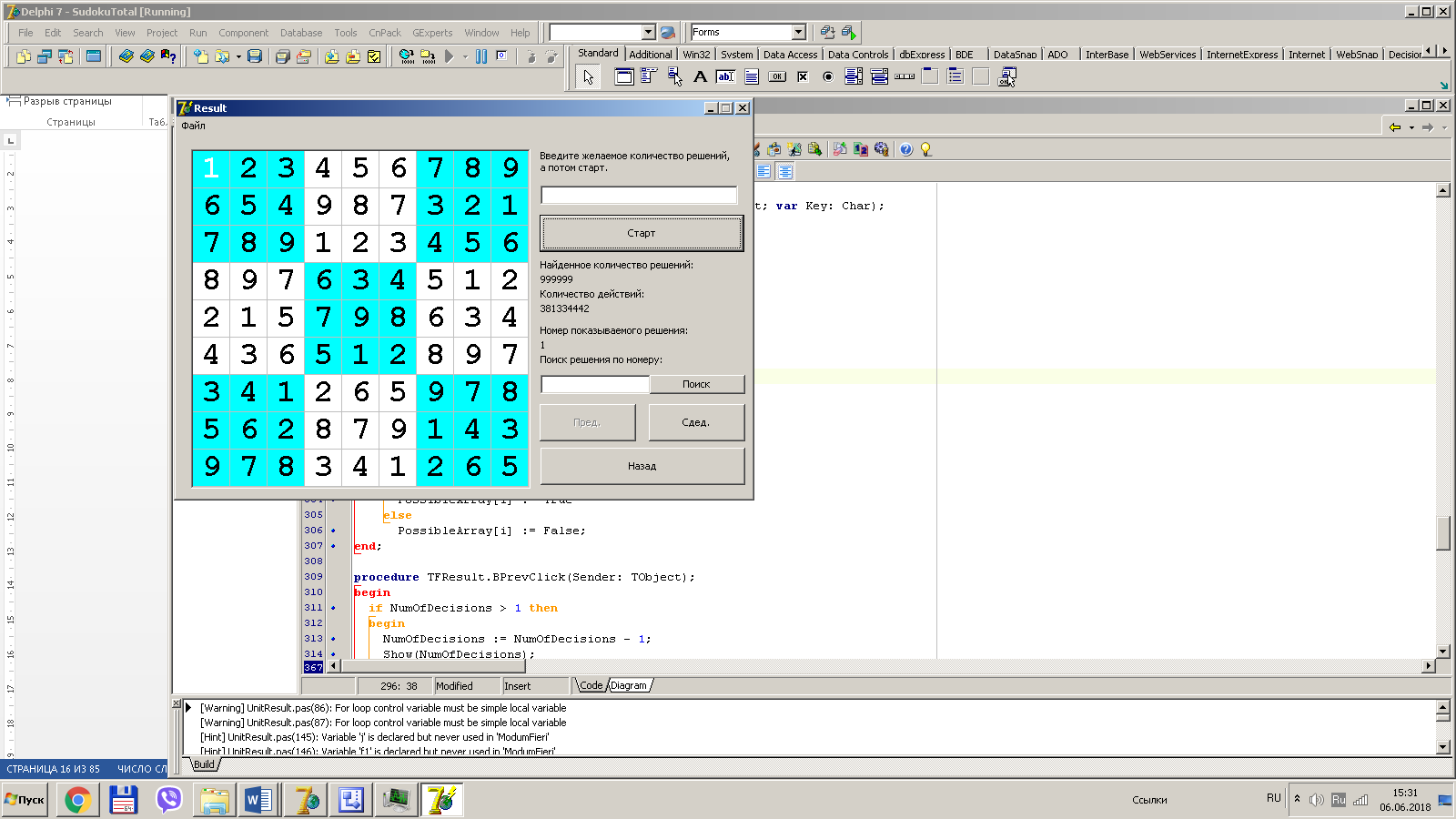


Рисунок 3.5 – Вид формы FResult при нажатии кнопки «Старт»

Составляющие вспомогательной формы FResult:

* компонент, обеспечивающий вывод найденных решений;
* компонент, отображающий количество найденных решений;
* компонент, отображающий количество затраченных действий;
* компонент, отображающий номер отображаемого решения;
* компонент, обеспечивающий ввод номера решения для поиска;
* компонент «Поиск», предоставляющий возможность поиска решения по заданному номеру;
* компонент, «След.», предоставляющий возможность просмотре следующего решения;
* компонент, «Пред.», предоставляющий возможность просмотре предыдущего решения;
* компонент «Назад», предоставляющий возможность перехода на главную форму
* компонент «Сохранить судоку в файл», предоставляющий возможность сохранить судоку в текстовый файл.

### Описание графических компонентов формы FAuthorization

Вспомогательная форма программы FAuthorization предоставляет возможность авторизоваться.

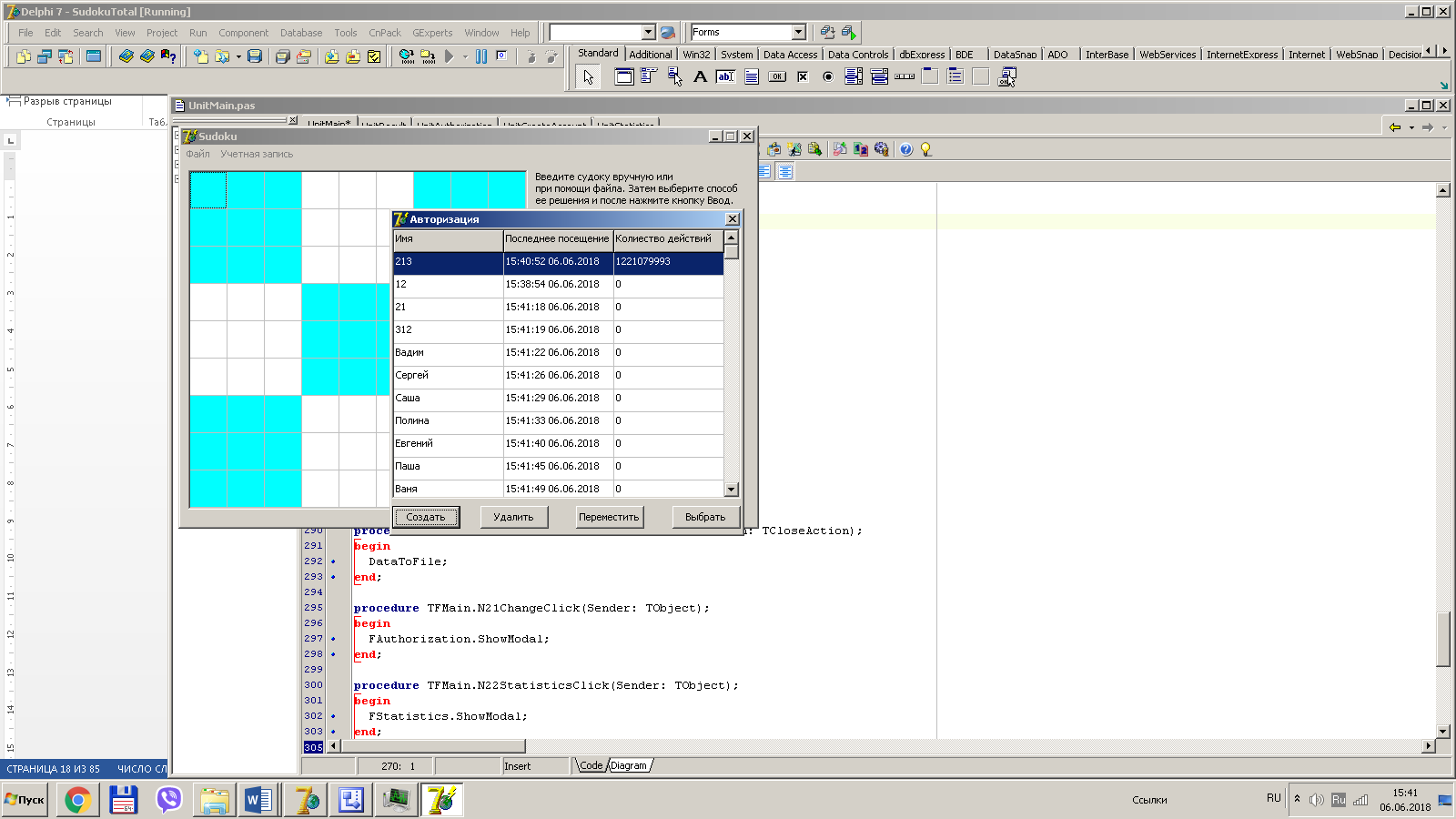


Рисунок 3.7 – Вид формы FAuthorization

Составляющие вспомогательной формы FAuthorization

* компонент, предоставляющий возможность отобразить имеющиеся аккаунты;
* компонент, предоставляющий возможность редактировать имя в имеющихся аккаунтах;
* кнопка «Создать», обеспечивающая добавления нового аккаунта в список;
* кнопка «Удалить», обеспечивающая удаление выбранного аккаунта из списка;
* кнопка «Переместить», обеспечивающая перемещение выбранного аккаунта в списке;
* кнопка «Выбрать», обеспечивающая выбор аккаунта из списка.

### Описание графических компонентов формы FCreateAccount

Вспомогательная форма программы FCreateAccount предоставляет пользователю возможность создания нового аккаунта. Данные пользователя затем сохраняются в отдельный типизированный файл.

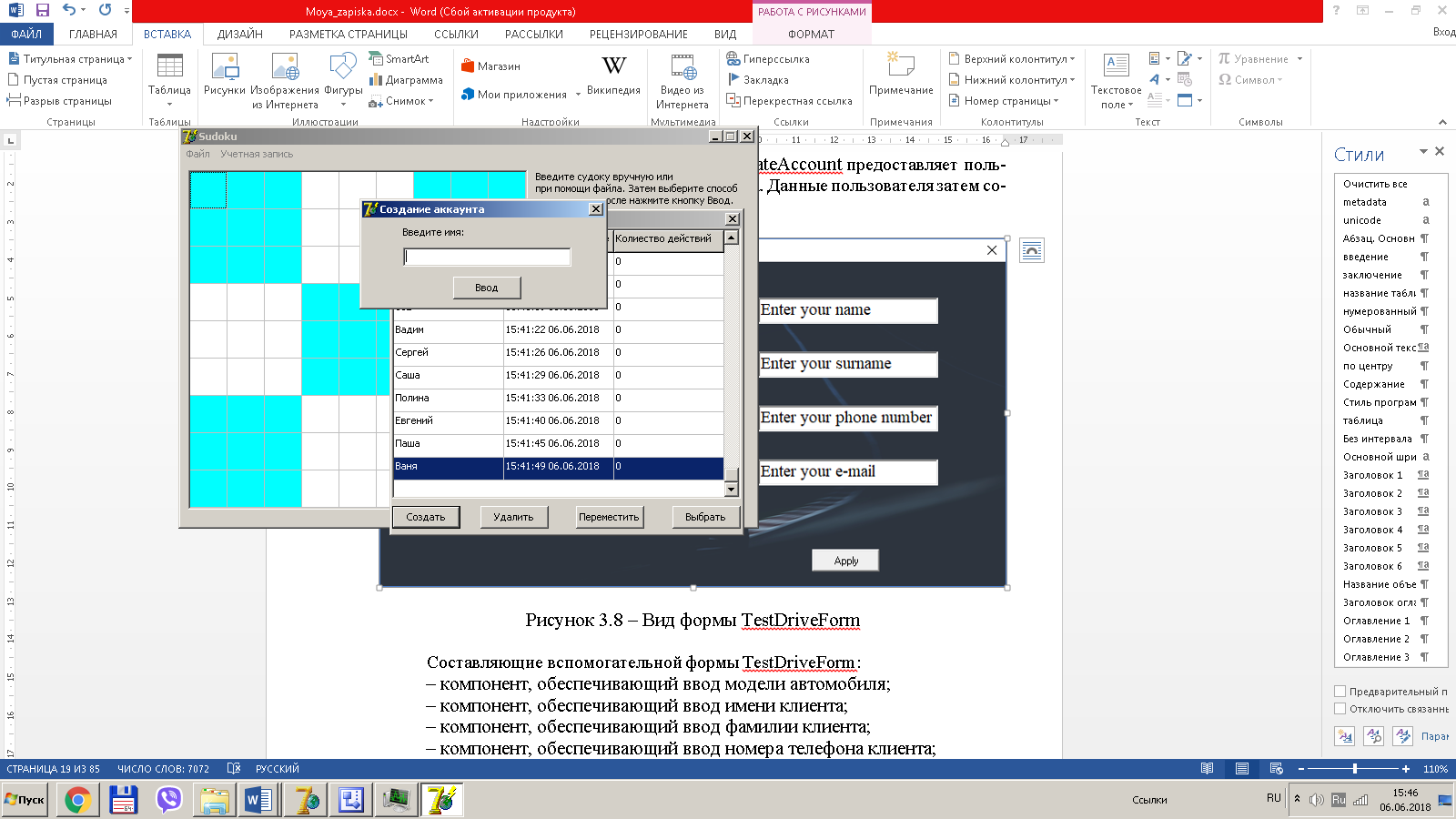


Рисунок 3.8 – Вид формы FCreateAccount

Составляющие вспомогательной формы FCreateAccount:

* компонент, обеспечивающий ввод имени;
* компонент «Ввод», обеспечивающий ввод имени в его аккаунт.

### Описание графических компонентов формы FStatistics

Вспомогательная форма программы FStatistics предоставляет основную информацию об аккаунтах, возможность поиска аккаунта по имени или его части и возможность сортировать аккаунты по одному из трех параметров: имя, последнее посещение, количество действий.

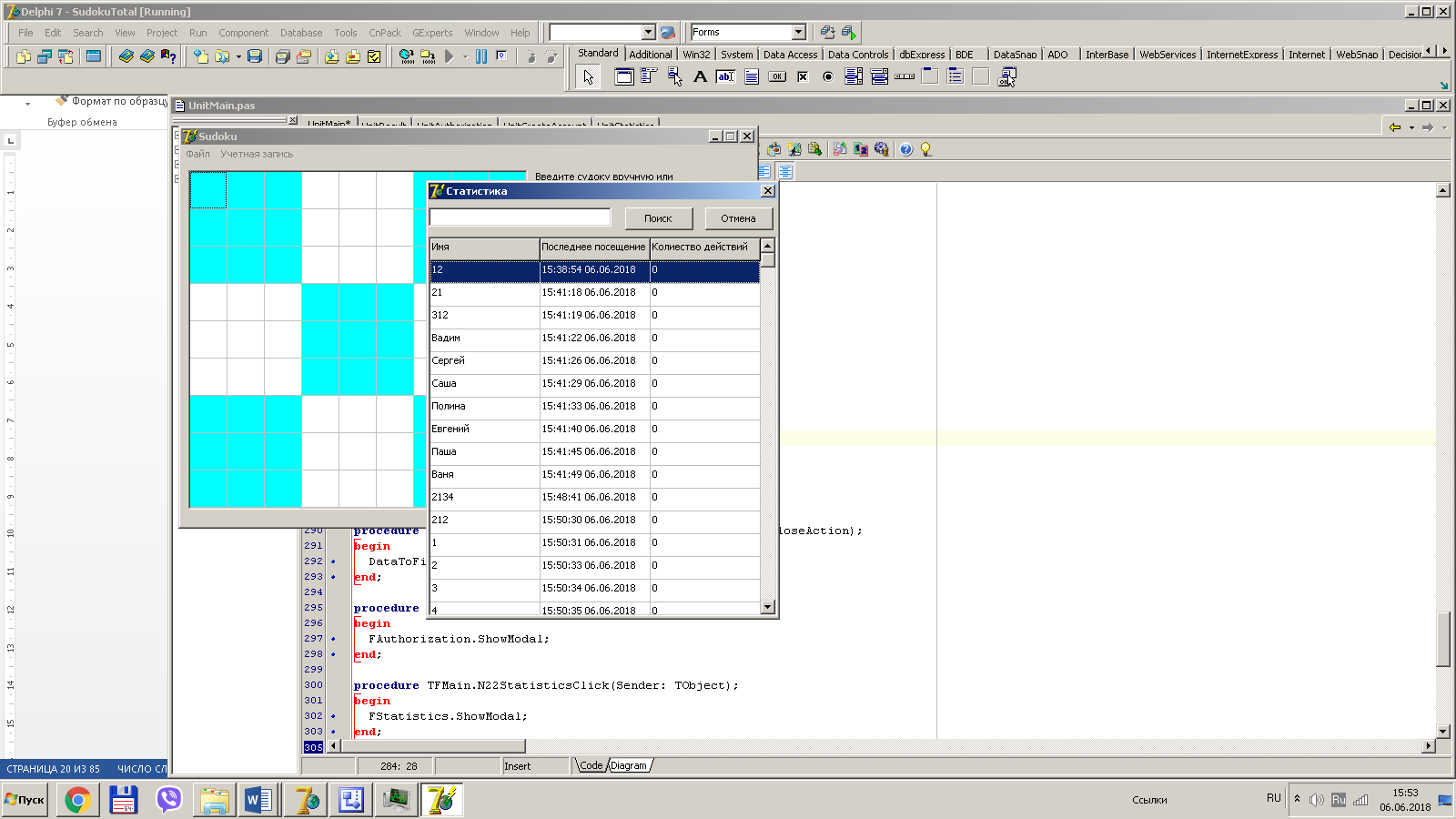


Рисунок 3.9 – Вид формы FStatistics

Составляющие вспомогательной формы FStatistics:

* компонент, отображающий основную информацию об аккаунтах;
* компонент, обеспечивающий ввода имени или его части для поиска аккаунта;
* кнопка «Поиск», обеспечивающая возможность поиска введённого имени или его части;
* кнопка «Отмена», обеспечивающая отмену всех фильтров;
* кнопка «Имя», обеспечивающая сортировку аккаунтов по имени.
* кнопка «Последнее посещение», обеспечивающая сортировку аккаунтов по времени последнего посещения;
* кнопка «Количество действий», обеспечивающая сортировку аккаунтов по количеству затраченных действий;

## Схема алгоритмов решения задачи по ГОСТ 19.701-90

## 3.3.1 Схема алгоритма DataToList



Рисунок 3.11 – Схема алгоритма DataToList

## 3.3.2 Схема алгоритма DataToFile



Рисунок 3.12 – Схема алгоритма DataToFile

### Схема алгоритма ListToGrid



Рисунок 3.13 – Схема алгоритма (часть 1)



Рисунок 3.14 – Схема алгоритма ListToGrid (часть 2)

### Схема алгоритма DelUser



Рисунок 3.15 – Схема алгоритма DelUser

### Схема алгоритма CreateNew



Рисунок 3.17 – Схема алгоритма CreateNew (часть 1)



Рисунок 3.18 – Схема алгоритма CreateNew (часть 2)

### Схема алгоритма Sorting



Рисунок 3.19 – Схема алгоритма Sorting (часть 1)



Рисунок 3.18 – Схема алгоритма Sorting (часть 2)

### Схема алгоритма Search



Рисунок 3.20 – Схема алгоритма Search

# Тестирование и проверка работоспособности программного средства

В ходе тестирования были рассмотрены в действии все функциональные возможности программы «Sudoku». Было изучено поведение программы при различных действиях пользователя.

## 4.1 Просмотр аккаунтов

### Тест 1

Таблица 13 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка возможности просмотра аккаунтов |
| Исходный набор данных: | Имя: 1  Последнее посещение: 15:50:31 27.01.2016  Количество действий: 0 |
| Ожидаемый результат: | Открытие окна с подробной информацией об аккаунтах |
| Фактический результат: |  |

### Тест 2

Таблица 14 – Тест 2

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка возможности просмотра информации об аккаунтах после сортировки по параметру «Имя» |
| Исходный набор данных: | Имя: 1  Последнее посещение: 15:50:31 27.01.2016  Количество действий: 0  Имя: 2  Последнее посещение: 15:50:31 27.01.2016  Количество действий: 0 |
| Ожидаемый результат: | Открытие окна с подробной информацией об аккаунтах |
| Фактический результат: |  |

## Сортировка по выбранному параметру

### 4.2.1 Тест 3

Таблица 15 – Тест 3

|  |  |
| --- | --- |
| Тестовая ситуация: | Сортировка исходного списка аккаунтов по параметру «Имя» |
| Исходный набор данных: |  |
| Ожидаемый результат: | Список аккаунтов, отсортированный по имени в порядке возрастания |
| Фактический результат: |  |

### 4.2.2 Тест 4

Таблица 16 – Тест 4

|  |  |
| --- | --- |
| Тестовая ситуация: | Сортировка исходного списка аккаунтов по параметру «Последнее посещение», с выбором способа сортировки по убыванию |
| Исходный набор данных: |  |
| Фактический результат: |  |

### 4.2.3 Тест 5

Таблица 17 – Тест 5

|  |  |
| --- | --- |
| Тестовая ситуация: | Сортировка исходного списка аккаунтов по параметру «Количество действий», с выбором способа сортировки по убыванию |
| Исходный набор данных: |  |
| Ожидаемый результат: | Список аккаунтов, отсортированный по количеству действий в порядке возрастания |
| Фактический результат: |  |

## Поиск по введённым данным

### 4.3.1 Тест 6

Таблица 18 – Тест 6

|  |  |
| --- | --- |
| Тестовая ситуация: | Поиск по имени |
| Исходный набор данных: |  |
| Ожидаемый результат: | Отображение в таблице аккаунта, имя которого совпадает с введённым |
| Фактический результат: |  |

### 4.3.1 Тест 7

Таблица 19 – Тест 7

|  |  |
| --- | --- |
| Тестовая ситуация: | Поиск по частично введённому имени |
| Исходный набор данных: |  |
| Ожидаемый результат: | Отображение в таблице аккаунта, имя которого частично соответствуют введённым в таблице |
| Фактический результат: |  |

### 4.3.1 Тест 8

Таблица 20 – Тест 8

|  |  |
| --- | --- |
| Тестовая ситуация: | Отмена поиска |
| Исходный набор данных: |  |
| Ожидаемый результат: | Отображение таблицы аккаунтов, до введения фильтров |
| Фактический результат: |  |

## Создание аккаунта

### 4.5.1 Тест 10

Таблица 22 – Тест 10

|  |  |
| --- | --- |
| Тестовая ситуация: | Попытка создания аккаунта, со всеми корректно заполненными полями |
| Исходный набор данных: |  |
| Ожидаемый результат: | Регистрация нового пользователя |
| Фактический результат: |  |

### 4.5.2 Тест 11

Таблица 23 – Тест 11

|  |  |
| --- | --- |
| Тестовая ситуация: | Попытка создания аккаунта, со существующим именем |
| Исходный набор данных: |  |
| Ожидаемый результат: | Сообщение о занятости имени |
| Фактический результат: |  |

### 4.5.3 Тест 12

Таблица 24 – Тест 12

|  |  |
| --- | --- |
| Тестовая ситуация: | Попытка создания аккаунта, без заполнения имени |
| Исходный набор данных: |  |
| Ожидаемый результат: | Сообщение с просьбой заполнить поле |
| Фактический результат: |  |

## Удаление аккаунта из списка

### 4.6.1 Тест 13

Таблица 25 – Тест 13

|  |  |
| --- | --- |
| Тестовая ситуация: | Попытка удаления аккаунта из списка |
| Исходный набор данных: |  |

Продолжение таблицы 25

|  |  |
| --- | --- |
| Ожидаемый результат: | Список аккаунтов, в котором отсутствует удалённый |
| Фактический результат: |  |

### 4.6.2 Тест 14

Таблица 26 – Тест 14

|  |  |
| --- | --- |
| Тестовая ситуация: | Перемещение аккаунта в списке |
| Исходный набор данных: |  |
| Ожидаемый результат: | Список аккаунтов |
| Фактический результат: |  |

## Редактирование аккаунта из списка

### 4.7.1 Тест 15

Таблица 27 – Тест 15

|  |  |
| --- | --- |
| Тестовая ситуация: | Попытка редактирования аккаунта из списка |
| Исходный набор данных: |  |
| Ожидаемый результат: | Вывод обновлённой информации в список аккаунтов |
| Фактический результат: |  |

### 4.7.2 Тест 16

Таблица 26 – Тест 16

|  |  |
| --- | --- |
| Тестовая ситуация: | Ввод некорректных данных в поле года выпуска |
| Исходный набор данных: |  |
| Ожидаемый результат: | Вывод сообщения о том, что введённое имя слишком длинное |
| Фактический результат: |  |

# Руководство по установке и использованию программного средства

## Шаг 1. Установка

Для открытия приложения необходимо распаковать архив и запустить файл Sudoku.exe. Далее появится главная страница программы, представленная на рисунке 5.1.

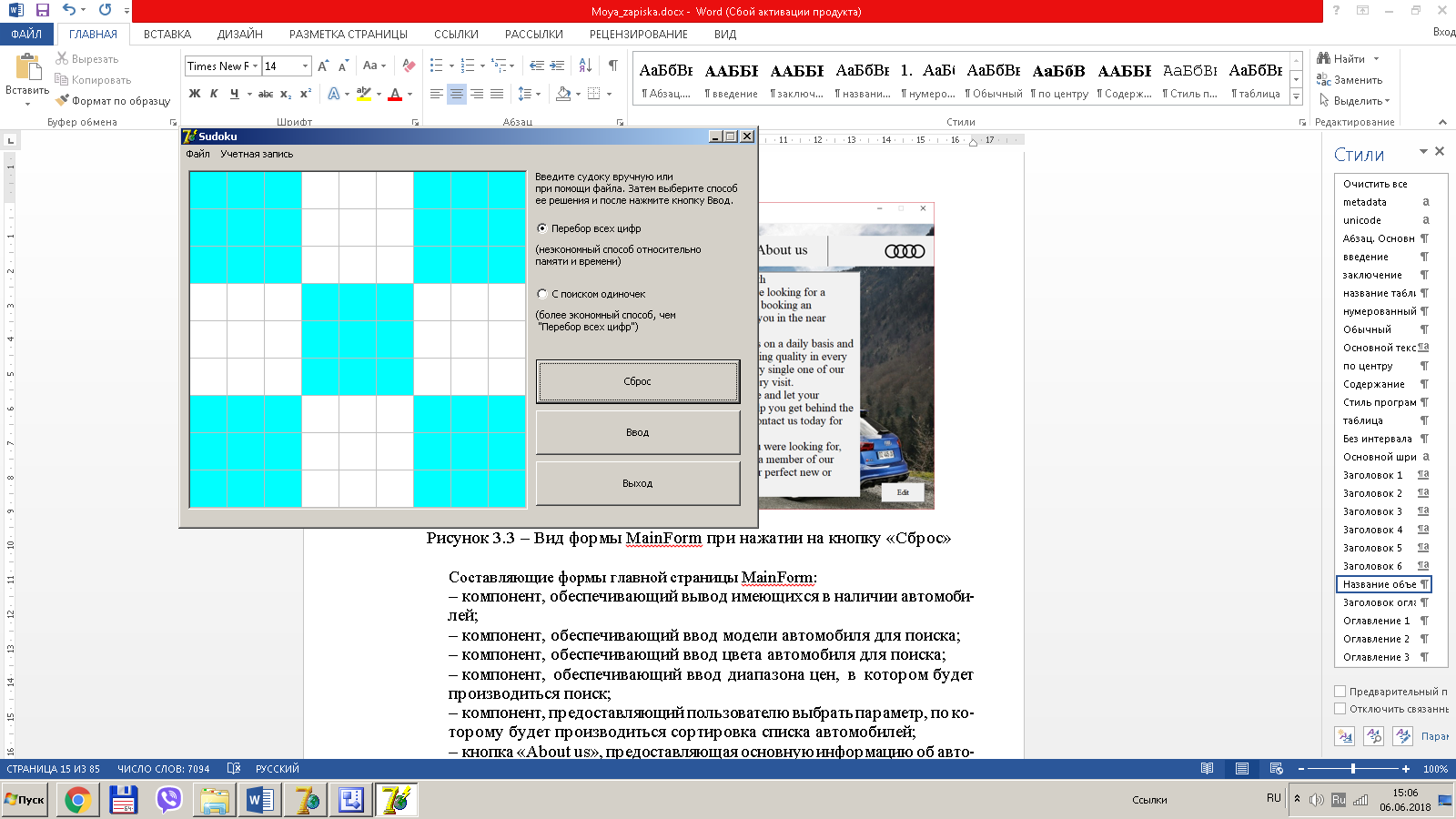


Рисунок 5.1 – Главная страница

## Шаг 2. Авторизация

Для того, чтобы выполнить поиск решения для судоку – нужно авторизоваться. Для этого нужно перейти по вкладке «Учетная запись-> Сменить пользователя». После нужно выбрать или создать новый аккаунт.

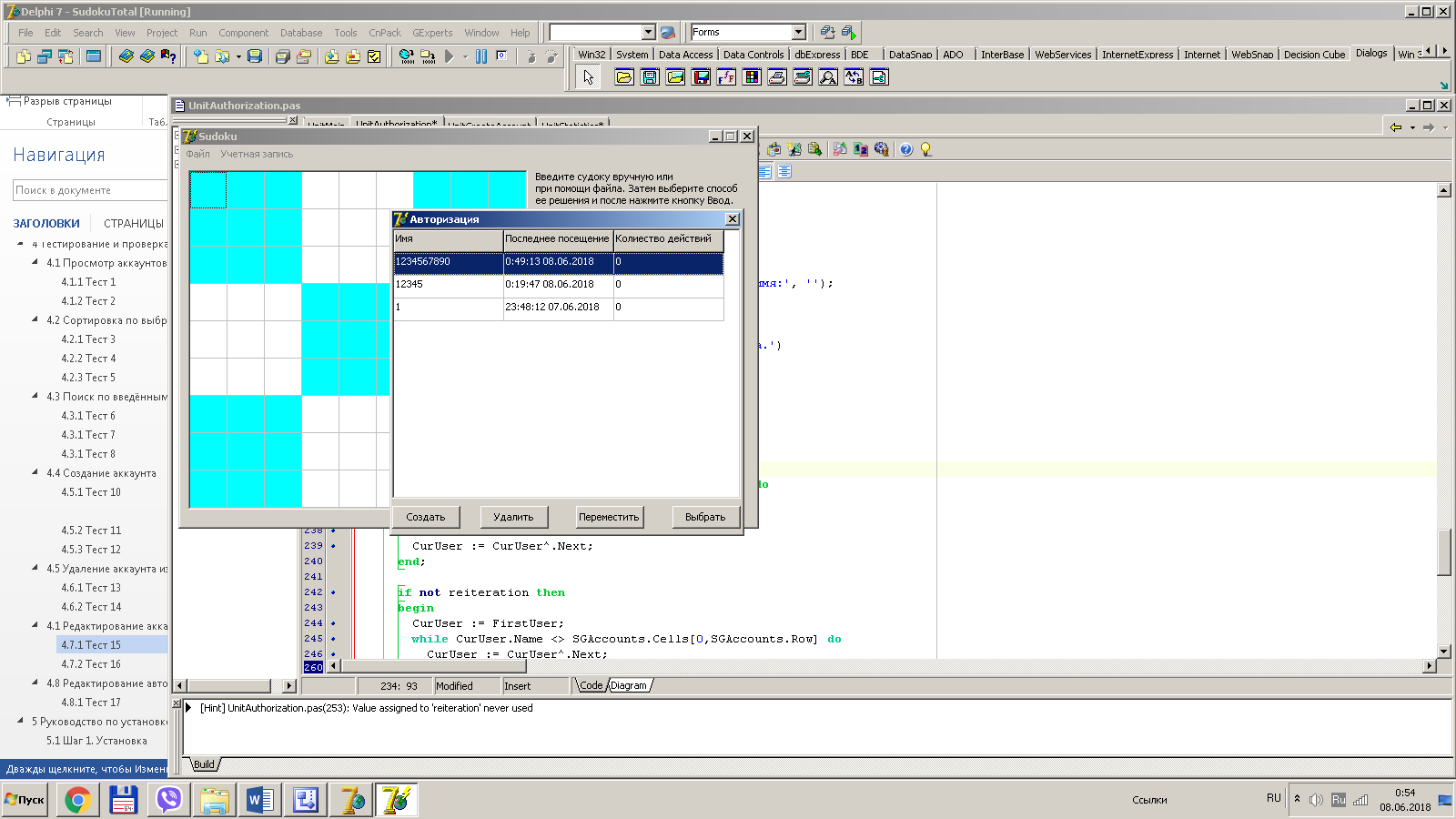


Рисунок 5.2 – Список аккаунтов

## Шаг 3. Ввод судоку в таблицу

Для того, чтобы найти решение (или несколько решений) судоку – ее надо ввести. Сделать это можно вручную, а также извлечь судоку из файла. Чтобы ввести судоку вручную нужно: выбрать нужную вам ячейку и нажмите клавишу с цифрой, которая вам нужна. Любая клавиша кроме цифры и стереть не будет учитываться. Если поле не заполнено – оно рассматривается как «0». Чтобы извлечь судоку из файла вам нужно перейти по вкладке «Файл-> Извлечь судоку из файла» и в высветившемся окне выбрать нужный вам файл. Судоку автоматически перенесется на таблицу.

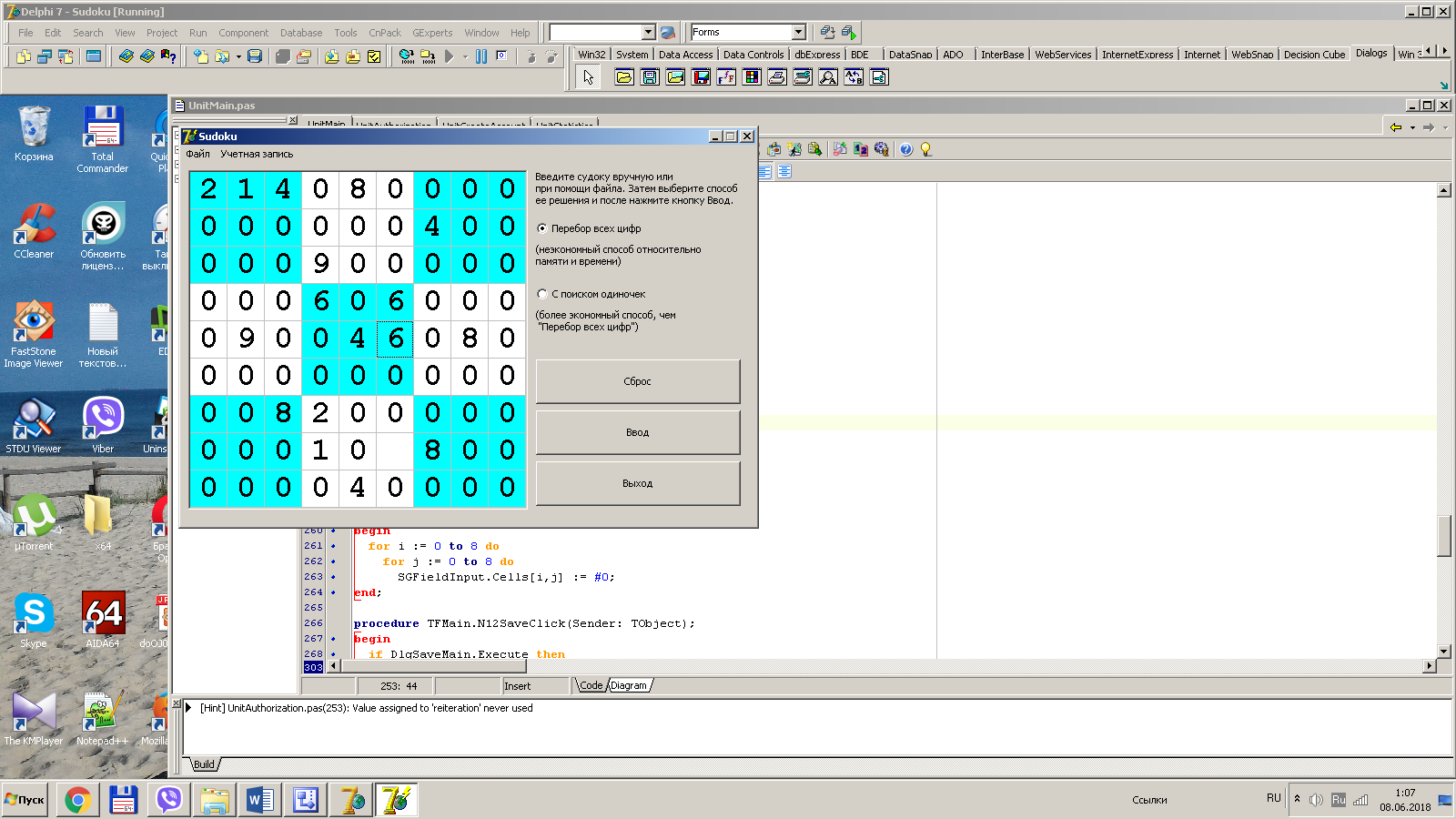


Рисунок 5.3 – Введенная судоку

## Шаг 4. Выбор способа решения и переход к результатам

После ввода и авторизации вам нужно выбрать способ решения. Есть два способа: «перебор всех цифр» и «с поиском одиночек». Затем нажмите ввод и перед вами появится окно «Результаты».

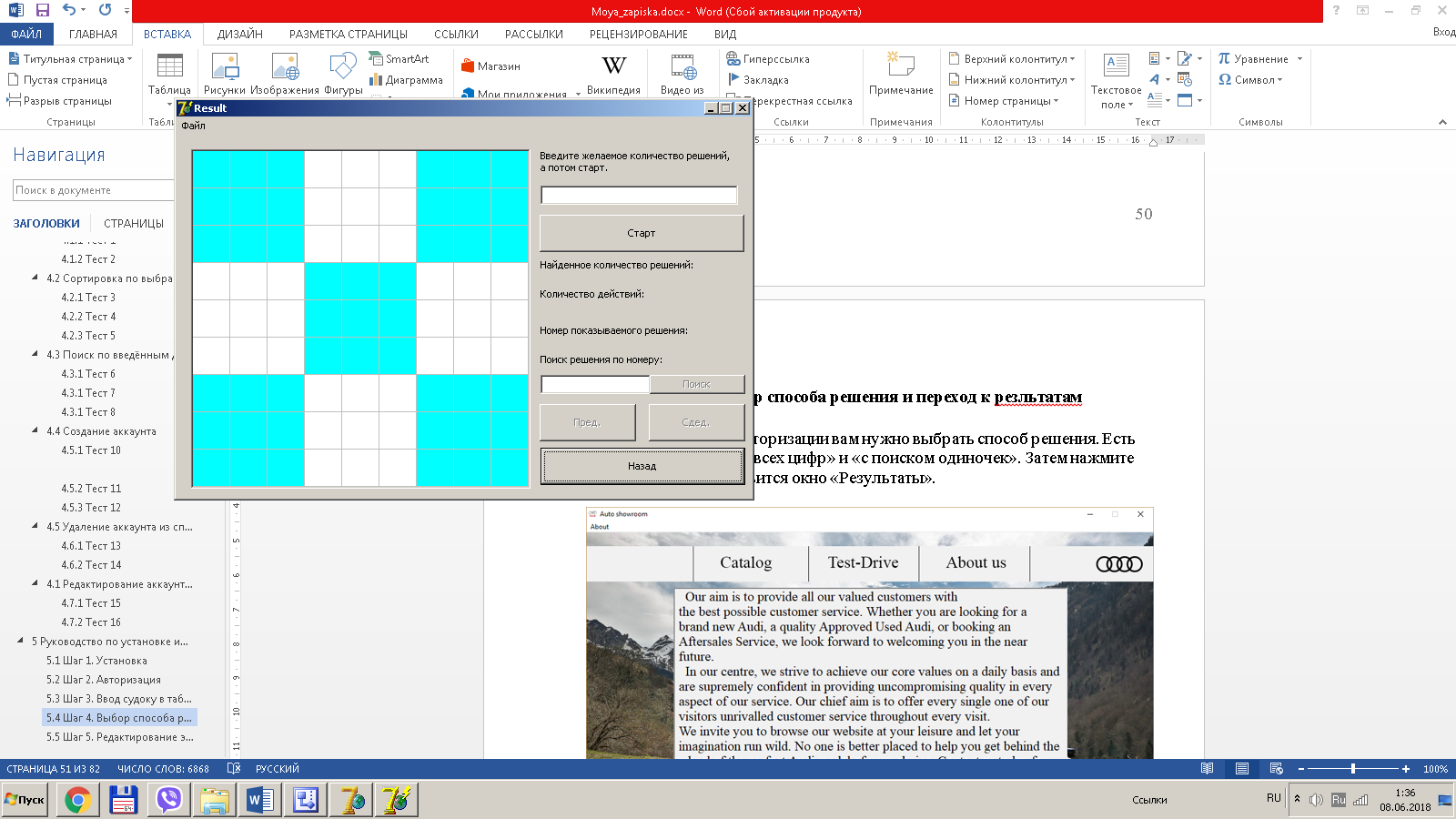


Рисунок 5.6 – Просмотр окна с результатами

## Шаг 5. Выбор количества искомых решений и отображение результатов

Для поиска решений вам нужно ввести количество желаемых решений. Если вы его не введете, программа найдет всевозможные решения (не более 999999). После выбора количества решений нажмите кнопку «Старт». Перед вами отобразится первое найденное решение, если судоку правильная. Также покажется количество действий затраченных на поиск результатов (одна подставленная цифра – одно действие).



Рисунок 5.7 – Отображение результатов

## Шаг 6. Сохранение выбранного результата

С помощью кнопок «Пред.» и «След.», а также объекта «Поиск» вы можете найти нужное вам решение по номеру. Также вы можете сохранить решение в текстовый файл. Для этого перейдите по вкладке «Файл-> Сохранить судоку в файл». Перед вами высветится окно где с помощью которого вы можете дать этому решению имя и сохранить его.

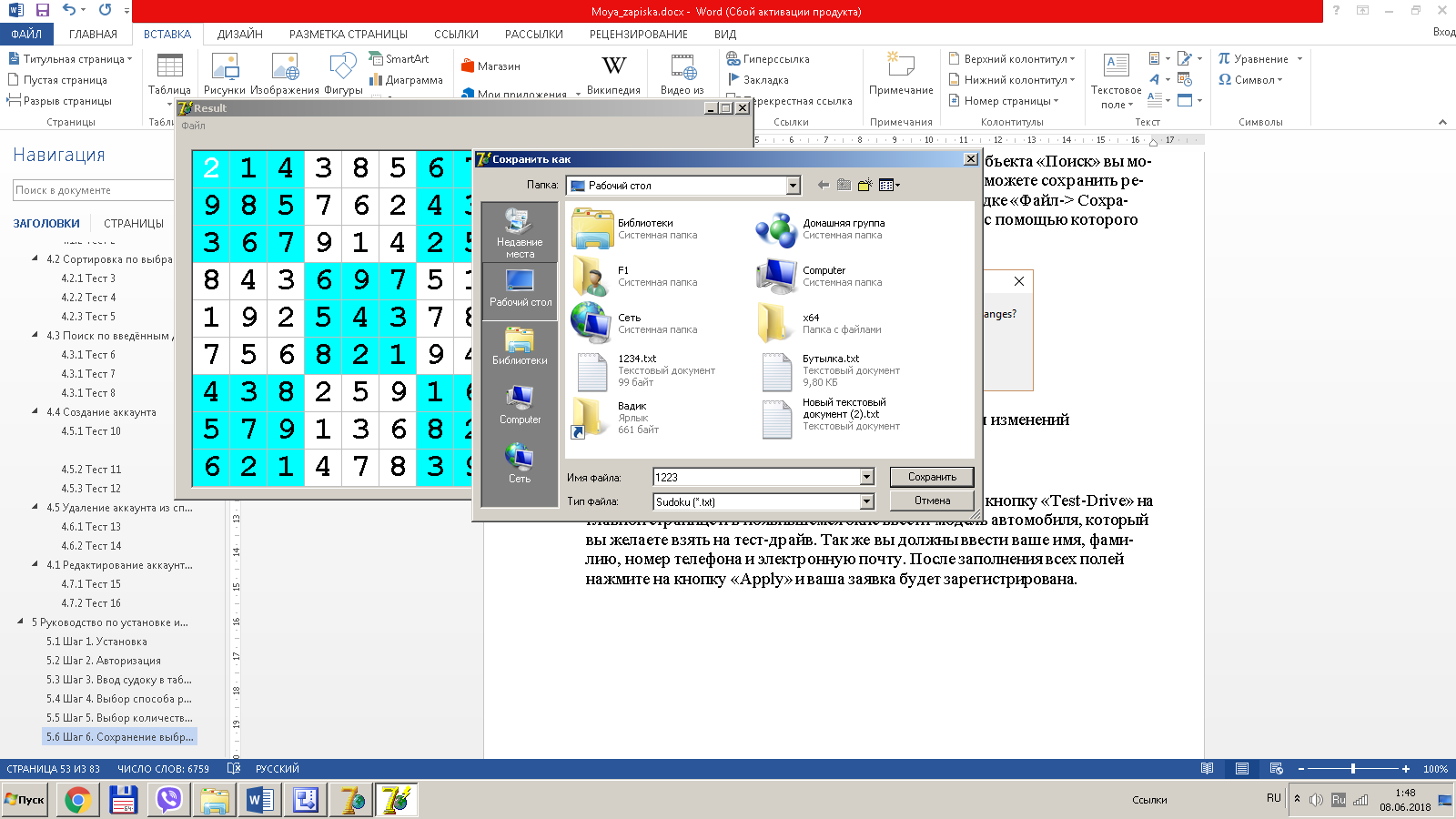


Рисунок 5.8 – Подтверждение сохранения изменений

## Шаг 7. Выход из программы

Для того чтобы выйти из программы вам достаточно нажать кнопку «Выход».

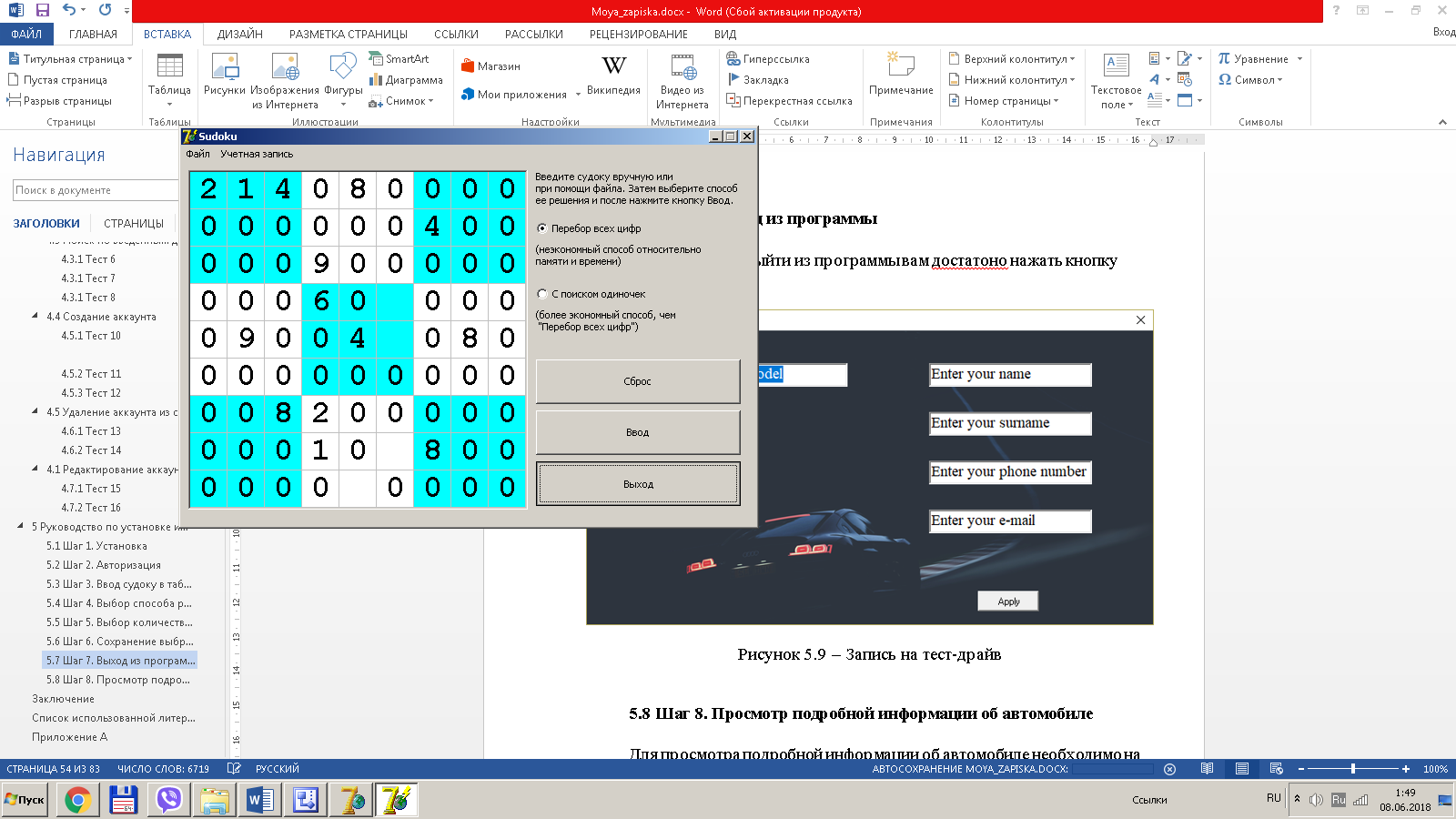


Рисунок 5.9 – Выход из программы

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 использованной литературы

[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] Объектно-ориентированное программирование [Электронный ресурс] – Режим доступа: http://library.bsuir.by/m/12\_101945\_1\_90135.pdf. – Дата доступа: 14.03.2016.

[4] Глухова, Л.А. Основы алгоритмизации и программирования: лабораторный практикум / Л.А. Глухова, Е.П. Фадеева, Е.Е. Фадеева. – Минск: БГУИР, 2007. – 4 ч.

[5] Серебряная, Л.В. Структуры и алгоритмы обработки данных: учеб.метод. Пособие / Л.В. Серебряная, И.М. Марина. – Минск: БГУИР, 2013. – 51 с.

[6] Вирт Н. Алгоритмы и структуры данных / Н. Вирт. – Москва: Мир 1989. – 360 с.

[7] Кнут Д.Э. Искусство программирования: в 4 т. / Д.Э. Кнут. – Москва: Вильямс, 2013. – 4 т.

[8] Сурков К.А. Программирование на языке Delphi [Электронный ресурс] – режим доступа: <http://pc.sitedirect.kz/soft/other/delphi.pdf> - Дата доступа: 10.04.2018

Приложение А

(обязательное)

Исходный код программы

program Sudoku;

uses

Forms,

UnitMain in UnitMain.pas' {Main},

UnitResult in UnitResult.pas' {Result},

UnitAuthorization inUnitAuthorization.pas'{Authorization},

UnitCreateAccount inUnitCreateAccount.pas'{CreateAccount},

UnitStatistics in UnitStatistics.pas' {Statistics};

{$R \*.res}

begin

Application.Initialize;

Application.CreateForm(TFMain, FMain);

Application.CreateForm(TFResult, FResult);

Application.CreateForm(TFAuthorization, FAuthorization);

Application.CreateForm(TFCreateAccount, FCreateAccount);

Application.CreateForm(TFStatistics, FStatistics);

Application.Run;

end.

{Модуль формы окна главной страницы}

unit UnitMain;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, StdCtrls, Grids, Unit2, Unit3, Menus;

type

TMainArray = array [1..81] of Integer;

TDirectionArray = array [0..8, 0..8] of Integer;

TNeighbors = array [1..81, 1..20] of Integer;

{ Тип для хранения данных об аккаунтах}

PUser = ^TUser;

TUser = record

Name: String[12];

LastEntrance: TDateTime;

Score: Integer;

Next, Prev: PUser;

TFMain = class(TForm)

SGFieldInput: TStringGrid;

BInput: TButton;

MMMenu: TMainMenu;

N1File: TMenuItem;

BExit: TButton;

RBRecursively: TRadioButton;

RBSingles: TRadioButton;

LAdmonition: TLabel;

LRecursively: TLabel;

LSingles: TLabel;

N11Open: TMenuItem;

BReser: TButton;

DlgOpenMain: TOpenDialog;

N12Save: TMenuItem;

DlgSaveMain: TSaveDialog;

N2Account: TMenuItem;

N21Change: TMenuItem;

N22Statistics: TMenuItem;

procedure SGFieldInputKeyPress(Sender: TObject; var Key: Char);

procedure SGFieldInputDrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

procedure BInputClick(Sender: TObject);

procedure BExitClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure N11OpenClick(Sender: TObject);

procedure BReserClick(Sender: TObject);

procedure N12SaveClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure N21ChangeClick(Sender: TObject);

procedure N22StatisticsClick(Sender: TObject);

private

procedure DataToList;

procedure DataToFile;

public

function Control(Sudoku: TMainArray): Boolean;

end;

var

FMain: TFMain;

SourceArray: TMainArray;

ResultArrays: array [1..999999] of TMainArray;

DirectionArray: TDirectionArray;

NumOfCandidates: TMainArray;

Neighbors: TNeighbors;

i, j, i0, j0, m: Integer;

Users: file of TUser;

CurUser, FirstUser, TimeUser: PUser;

UserForRecord: TUser;

CurSudoku: TextFile;

CurStr: String;

Authorized: String;

implementation

uses UnitResult, UnitAuthorization, UnitStatistics;

{$R \*.dfm}

{Процедура сохранения динамического списка в типизированного файл}

procedure TFMain.DataToFile;

begin

rewrite(Users);

while FirstUser <> nil do

begin

UserForRecord.Name := FirstUser^.Name;

UserForRecord.LastEntrance := FirstUser^.LastEntrance;

UserForRecord.Score := FirstUser^.Score;

write(Users, UserForRecord);

seek(Users,FilePos(USers) + 1);

CurUser := FirstUser^.Next;

Dispose(FirstUser);

FirstUser := CurUser;

end;

closefile(Users);

end;

{Процедура извлечения динамического списка из файла}

procedure TFMain.DataToList;

begin

assignfile (Users,ExtractFilePath(Application.ExeName) + 'Users.sdk');

reset (Users);

CurUser := nil;

FirstUser := nil;

while not EoF(Users) do

begin

if FirstUser = nil then

begin

new(CurUser);

FirstUser := CurUser;

CurUser^.Prev := nil;

end

else

begin

new(CurUser^.Next);

CurUser^.Next^.Prev := CurUser;

CurUser := CurUser^.Next;

end;

read(Users, UserForRecord);

seek(Users, FilePos(Users) + 1);

CurUser^.Name := UserForRecord.Name;

CurUser^.LastEntrance := UserForRecord.LastEntrance;

CurUser^.Score := UserForRecord.Score;

end;

if CurUser <> nil then

CurUser^.Next := nil;

end;

{Функция проверяющая судоку на правильность}

function TFMain.Control(Sudoku: TMainArray): Boolean;

begin

Result := True;

for i := 1 to 81 do

for j := 1 to 20 do

if (Sudoku[i] = Sudoku[Neighbors[i,j]]) and

(Sudoku[i] > 0) then

Result := False;

end;

procedure TFMain.SGFieldInputKeyPress(Sender: TObject; var Key: Char);

begin

if not (key in ['0'..'9']) then

key:=#0;

SGFieldInput.Cells[SGFieldInput.col,SGFieldInput.row] := key;

end;

procedure TFMain.SGFieldInputDrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

begin

if (ACol in [0..2, 6..8]) and (ARow in [0..2, 6..8])

or (ACol in [3..5]) and (ARow in [3..5]) then

begin

SGFieldInput.Canvas.Brush.Color := TColor($FFFF00);

SGFieldInput.Canvas.FillRect(Rect);

end

else

begin

SGFieldInput.Canvas.Brush.Color := TColor($FFFFFF);

SGFieldInput.Canvas.FillRect(Rect);

end;

SGFieldInput.Canvas.TextOut(Rect.Left + 10,Rect.Top,SGFieldInput.Cells[Acol,Arow]);

end;

procedure TFMain.BInputClick(Sender: TObject);

begin

if Authorized <> '' then

begin

for i := 0 to 8 do

for j := 0 to 8 do

if SGFieldInput.Cells[i,j] = '' then

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

else

if SGFieldInput.Cells[i,j][1] in ['0'..'9'] then

SourceArray[DirectionArray[i,j]]:= StrToInt(SGFieldInput.Cells[i,j])

else

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

if RBRecursively.Checked or RBSingles.Checked then

if Control(SourceArray) then

begin

FMain.Hide;

FResult.ShowModal;

end

else

ShowMessage ('Ñóäîêó íå ìîæåò áûòü ðåøåíà. Èñïðàâüòå åå è ïîâòîðèòå ïîïûòêó.')

else

ShowMessage('Âûáåðèòå ñïîñîá ðåøåíèÿ ñóäîêó.');

end

else

begin

ShowMessage('Äëÿ íà÷àëà àâòîðèçèðóéòåñü.');

FAuthorization.ShowModal;

end;

end;

procedure TFMain.BExitClick(Sender: TObject);

begin

FMain.Close;

end;

procedure TFMain.FormCreate(Sender: TObject);

begin

DataToList;

Authorized := '';

for i := 0 to 8 do

for j := 0 to 8 do

if j mod 2 = 0 then

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

else

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

for i := 0 to 8 do

for j := 0 to 8 do

begin

m := 0;

for i0 := 0 to 8 do

for j0 := 0 to 8 do

if ((i=i0) and (j<>j0))

or ((i<>i0) and (j=j0))

or ((i<>i0) and (j<>j0)

and (i div 3 = i0 div 3)

and (j div 3 = j0 div 3)) then

begin

m := m + 1;

Neighbors[DirectionArray[i,j],m] :=

DirectionArray[i0,j0]

end;

end;

end;

procedure TFMain.N11OpenClick(Sender: TObject);

begin

if DlgOpenMain.Execute then

begin

assignfile (CurSudoku,DlgOpenMain.FileName);

reset (CurSudoku);

for i := 0 to 8 do

begin

readln (CurSudoku, CurStr);

for j := 0 to 8 do

SGFieldInput.Cells[j,i] := CurStr[j + 1];

end;

closefile (CurSudoku);

end;

end;

procedure TFMain.BReserClick(Sender: TObject);

begin

for i := 0 to 8 do

for j := 0 to 8 do

SGFieldInput.Cells[i,j] := #0;

end;

procedure TFMain.N12SaveClick(Sender: TObject);

begin

if DlgSaveMain.Execute then

begin

assignfile (CurSudoku, DlgSaveMain.FileName);

rewrite (CurSudoku);

for i := 0 to 8 do

begin

CurStr := '';

for j := 0 to 8 do

if SGFieldInput.Cells[j,i] = '' then

CurStr := CurStr + '0'

else

CurStr := CurStr + SGFieldInput.Cells[j,i];

writeln(CurSudoku,CurStr);

end;

closefile (CurSudoku);

end;

end;

procedure TFMain.FormClose(Sender: TObject; var Action: TCloseAction);

begin

DataToFile;

end;

procedure TFMain.N21ChangeClick(Sender: TObject);

begin

FAuthorization.ShowModal;

end;

procedure TFMain.N22StatisticsClick(Sender: TObject);

begin

FStatistics.ShowModal;

end;

end.

{ Модуль формы окна с результатами }

unit UnitResult;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, Grids, Menus;

type

TPossibleArray = array [1..81] of Boolean;

TCandidates = array [1..81, 1..9] of Integer;

TFResult = class(TForm)

BBack: TButton;

SGFieldOutput: TStringGrid;

BStart: TButton;

LAdmonition1: TLabel;

ENumOfResults: TEdit;

BPrev: TButton;

BNext: TButton;

LAdmonition2: TLabel;

LNumOfDecision: TLabel;

LAdmonition3: TLabel;

LNumOfDecisionShown: TLabel;

LAdmonition4: TLabel;

ESearch: TEdit;

BSearch: TButton;

LAdmontion5: TLabel;

LNumOfSteps: TLabel;

MMMenu: TMainMenu;

N1File: TMenuItem;

N12Save: TMenuItem;

DlgSaveResult: TSaveDialog;

procedure BBackClick(Sender: TObject);

procedure SGFieldOutputDrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

procedure BStartClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure ENumOfResultsKeyPress(Sender: TObject; var Key: Char);

procedure FormActivate(Sender: TObject);

procedure BPrevClick(Sender: TObject);

procedure BNextClick(Sender: TObject);

procedure BSearchClick(Sender: TObject);

procedure N12SaveClick(Sender: TObject);

private

procedure Heart;

procedure Show(Id: Integer);

public

end;

var

FResult: TFResult;

PossibleArray: TPossibleArray;

Candidates: TCandidates;

NumOfDecisions: Integer;

NumOfSteps: Integer;

NumOfResults: Integer;

i, j: Integer;

implementation

uses UnitMain;

{$R \*.dfm}

{Процедура для отображения казанного решения судоку}

procedure TFResult.Show(Id: Integer);

begin

LNumOfDecisionShown.Caption := IntToStr(Id);

if Id = 1 then

BPrev.Enabled := False

else

BPrev.Enabled := True;

if Id = NumOfResults then

BNext.Enabled := False

else

BNext.Enabled := True;

if NumOfResults = 1 then

BSearch.Enabled := False

else

BSearch.Enabled := True;

for i := 0 to 8 do

for j := 0 to 8 do

SGFieldOutput.Cells[i,j] := IntToStr(ResultArrays[Id,DirectionArray[i,j]]);

end;

{Процедура отвечающая за поиск решений судоку}

procedure TFResult.Heart;

{Функция проверяющая подходит ли кандидат}

function CandidateCheck(Candidate, NumOfEl: Integer): Boolean;

var

i: Integer;

begin

Result := True;

i := 1;

while Result and (i <= 20) do

begin

if Candidate = SourceArray[Neighbors[NumofEl,i]] then

Result := False;

inc(i);

end;

end;

{Рекурсивный способ поиска решений}

procedure RecursiveMethod(NumOfEl: Integer);

var

Candidate: Integer;

begin

if NumOfEl = 82 then

begin

ResultArrays[NumOfDecisions] := SourceArray;

NumOfDecisions := NumOfDecisions + 1;

end;

if (NumOfEl <= 81) and (NumOfDecisions <= NumOfResults) then

if PossibleArray[NumOfEl] then

RecursiveMethod(NumOfEl + 1)

else

for Candidate := 1 to 9 do

begin

if CandidateCheck(Candidate,NumOfEl) then

begin

SourceArray[NumofEl] := Candidate;

NumofSteps := NumofSteps + 1;

RecursiveMethod(NumOfEl + 1);

end;

SourceArray[NumofEl] := 0;

NumofSteps := NumofSteps + 1;

end;

end;

{Подготовка к экономному решению судоку}

function ModumFieri: Boolean;

var

Easy: Boolean;

Candidate, NumOfEl: Integer;

j: Integer;

f1: Boolean;

begin

repeat

Result := True;

Easy := False;

for NumOfEl := 1 to 81 do

begin

NumOfCandidates[NumOfEl] := 0;

if SourceArray[NumOfEl] = 0 then

begin

for Candidate := 1 to 9 do

if CandidateCheck(Candidate,NumOfEl) then

begin

NumOfCandidates[NumOfEl] :=

NumOfCandidates[NumOfEl] + 1;

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

end;

if NumOfCandidates[NumOfEl] = 0 then

Result := False;

if NumOfCandidates[NumOfEl] = 1 then

begin

SourceArray[NumOfEl] := Candidates[NumOfEl, 1];

Easy := True;

end;

end;

end

until not Easy or not Result;

end;

{Уьный способ решения судоку}

procedure RecursiveMethod\_Smart(NumOfEl: Integer);

var

Candidate: Integer;

begin

if NumOfEl = 82 then

begin

ResultArrays[NumOfDecisions] := SourceArray;

FResult.LNumOfDecision.Caption := IntToStr

(NumOfDecisions);

NumOfDecisions := NumOfDecisions + 1;

end;

if (NumOfEl <= 81) and (NumOfDecisions <= NumOfResults) then

begin

if (NumOfCandidates[NumOfEl] = 0) or

(NumOfCandidates[NumOfEl] = 1) then

RecursiveMethod\_Smart(NumOfEl + 1);

if NumOfCandidates[NumOfEl] > 1 then

begin

for Candidate := 1 to NumOfCandidates[NumOfEl] do

begin

if CandidateCheck(Candidate,NumOfEl) then

begin

SourceArray[NumOfEl] := Candidates[NumOfEl, Candidate];

NumOfSteps := NumOfSteps + 1;

RecursiveMethod\_Smart(NumOfEl + 1)

end;

end;

SourceArray[NumOfEl] := 0;

NumOfSteps := NumOfSteps + 1

end;

end;

end;

begin

NumOfDecisions := 1;

NumOfSteps := 0;

if FMain.RBRecursively.Checked then

RecursiveMethod(1);

if FMain.RBSingles.Checked then

if ModumFieri then

RecursiveMethod\_Smart(1)

else

ShowMessage('Ñóäîêó íå ìîæåò áûòü ðåøåíà. Èñïðàâüòå åå è ïîâòîðèòå ïîïûòêó.');

FResult.LNumOfDecision.Caption := IntToStr(

NumOfDecisions - 1);

FResult.LNumOfSteps.Caption := IntToStr(NumOfSteps);

CurUser := FirstUser;

while CurUser^.Name <> Authorized do

CurUser := CurUser^.Next;

CurUser^.Score := CurUser^.Score + NumOfSteps;

NumOfDecisions := 1;

Show(NumOfDecisions);

end;

procedure TFResult.BBackClick(Sender: TObject);

begin

for i := 0 to 8 do

for j := 0 to 8 do

SGFieldOutPut.Cells[i,j] := '';

FMain.Show;

FResult.Close;

end;

procedure TFResult.SGFieldOutputDrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

begin

if (ACol in [0..2, 6..8]) and (ARow in [0..2, 6..8])

or (ACol in [3..5]) and (ARow in [3..5]) then

begin

SGFieldOutput.Canvas.Brush.Color := TColor($FFFF00);

SGFieldOutput.Canvas.FillRect(Rect);

end

else

begin

SGFieldOutput.Canvas.Brush.Color := TColor($FFFFFF);

SGFieldOutput.Canvas.FillRect(Rect);

end;

SGFieldOutput.Canvas.TextOut(Rect.Left + 10,Rect.Top,SGFieldOutput.Cells[Acol,Arow]);

end;

procedure TFResult.BStartClick(Sender: TObject);

begin

if ENumOfResults.Text <> '' then

NumOfResults := StrToInt(ENumOfResults.Text)

else

NumOfResults := 999999;

if NumOfResults <> 0 then

Heart

else

ShowMessage('Íå ìîæåò áûòü 0 ðåøåíèé. Âââåäèòå íàòóðàëüíîå ÷èñëî.')

end;

procedure TFResult.FormClose(Sender: TObject; var Action: TCloseAction);

begin

FMain.Show;

FResult.Close

end;

procedure TFResult.ENumOfResultsKeyPress(Sender: TObject; var Key: Char);

begin

if not (key in ['0'..'9', #8]) then

key := #0;

end;

procedure TFResult.FormActivate(Sender: TObject);

begin

FResult.LNumOfDecision.Caption := '';

FResult.ENumOfResults.Text := '';

FResult.LNumOfSteps.Caption := '';

FResult.LNumOfDecisionShown.Caption := '';

BPrev.Enabled := False;

BNext.Enabled := False;

BSearch.Enabled := False;

for i := 1 to 81 do

if SourceArray[i] > 0 then

PossibleArray[i] := True

else

PossibleArray[i] := False;

end;

procedure TFResult.BPrevClick(Sender: TObject);

begin

if NumOfDecisions > 1 then

begin

NumOfDecisions := NumOfDecisions - 1;

Show(NumOfDecisions);

end;

end;

procedure TFResult.BNextClick(Sender: TObject);

begin

if NumOfDecisions < NumOfResults then

begin

NumOfDecisions := NumOfDecisions + 1;

Show(NumOfDecisions);

end;

end;

procedure TFResult.BSearchClick(Sender: TObject);

begin

if ESearch.Text <> '' then

if (StrToInt(ESearch.Text) > 0) and (StrToInt(ESearch.Text) <= NumOfResults) then

begin

Show(StrToInt(ESearch.Text));

NumOfDecisions := StrToInt(ESearch.Text);

end

else

ShowMessage ('Ðåøåíèå ñ ââåäåííûì íîìåðîì íå íàéäåíî.')

else

ShowMessage ('Ââåäèòå íîìåð ðåøåíèÿ.');

end;

procedure TFResult.N12SaveClick(Sender: TObject);

begin

if NumOfDecisions <> 0 then

if DlgSaveResult.Execute then

begin

assignfile (CurSudoku, DlgSaveResult.FileName);

rewrite (CurSudoku);

for i := 0 to 8 do

begin

CurStr := '';

for j := 0 to 8 do

if SGFieldOutPut.Cells[j,i] = '' then

CurStr := CurStr + '0'

else

CurStr := CurStr + SGFieldOutPut.Cells[j,i];

writeln(CurSudoku,CurStr);

end;

closefile (CurSudoku);

end

else

else

ShowMessage ('Íå÷åãî ñîõðàíÿòü!');

end;

end.

{ Модуль формы окна авторизации }

unit UnitAuthorization;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, Grids, StdCtrls;

type

TFAuthorization = class(TForm)

BCreate: TButton;

BSelect: TButton;

SGAccounts: TStringGrid;

BDelete: TButton;

BMove: TButton;

procedure BSelectClick(Sender: TObject);

procedure BCreateClick(Sender: TObject);

procedure FormActivate(Sender: TObject);

procedure BDeleteClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure BMoveClick(Sender: TObject);

procedure FormKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure SGAccountsDblClick(Sender: TObject);

private

procedure ListToGrid;

procedure DelUser;

public

end;

var

FAuthorization: TFAuthorization;

implementation

uses UnitMain, UnitCreateAccount;

{$R \*.dfm}

{Процедура удаляющая запись из списка}

procedure TFAuthorization.DelUser;

begin

if SGAccounts.Row > 2 then

begin

CurUser := FirstUser;

while CurUser^.Next^.Name <> SGAccounts.Cells[0,

SGAccounts.Row] do

CurUser := CurUser^.Next;

TimeUser := CurUser^.Next;

CurUser^.Next := CurUser^.Next^.Next;

if CurUser^.Next <> nil then

CurUser^.Next^.Prev := CurUser;

end

else

begin

TimeUser := FirstUser;

FirstUser := FirstUser^.Next;

if FirstUser <> nil then

FirstUser^.Prev := nil;

end;

Dispose(TimeUser);

end;

{Процедура заносящая список в таблицу}

procedure TFAuthorization.ListToGrid;

begin

CurUser := FirstUser;

if FirstUser = nil then

begin

BDelete.Enabled := False;

BMove.Enabled := False;

BSelect.Enabled := False;

Authorized := '';

SGAccounts.Rows[1].Clear;

end

else

begin

BDelete.Enabled := True;

BMove.Enabled := True;

BSelect.Enabled := True;

end;

SGAccounts.RowCount := 2;

while CurUser <> nil do

begin

UserForRecord.LastEntrance := CurUser^.LastEntrance;

UserForRecord.Score := CurUser^.Score;

SGAccounts.Cells[0, SGAccounts.RowCount - 1] := CurUser^.Name;

SGAccounts.Cells[1, SGAccounts.RowCount - 1] :=

TimeToStr(UserForRecord.LastEntrance) + ' ' +

DateToStr(UserForRecord.LastEntrance);

SGAccounts.Cells[2, SGAccounts.RowCount - 1] :=

IntToStr(UserForRecord.Score);

CurUser := CurUser^.Next;

if CurUser <> nil then

SGAccounts.RowCount := SGAccounts.RowCount + 1;

end;

end;

procedure TFAuthorization.BSelectClick(Sender: TObject);

begin

if SGAccounts.Cells[0, 1] <> '' then

begin

Authorized := SGAccounts.Cells[0, SGAccounts.Row];

CurUser := FirstUser;

while CurUser^.Name <> Authorized do

CurUser := CurUser^.Next;

CurUser^.LastEntrance := Time + Date;

FAuthorization.Close;

end

else

ShowMessage('Ñîçäàéòå íîâîãî ïîëüçîâàòåëÿ.');

end;

procedure TFAuthorization.BCreateClick(Sender: TObject);

begin

FCreateAccount.ShowModal;

ListToGrid;

end;

procedure TFAuthorization.FormActivate(Sender: TObject);

begin

ListToGrid;

end;

procedure TFAuthorization.BDeleteClick(Sender: TObject);

begin

DelUser;

ListToGrid;

end;

procedure TFAuthorization.FormCreate(Sender: TObject);

begin

SGAccounts.Cells[0,0] := 'Èìÿ';

SGAccounts.Cells[1,0] := 'Ïîñëåäíåå ïîñåùåíèå';

SGAccounts.Cells[2,0] := 'Êîëèåñòâî äåéñòâèé';

end;

procedure TFAuthorization.BMoveClick(Sender: TObject);

begin

CurUser := FirstUser;

while CurUser.Name <> SGAccounts.Cells[0,SGAccounts.Row] do

CurUser := CurUser^.Next;

BCreate.Enabled := False;

BDelete.Enabled := False;

BMove.Enabled := False;

BSelect.Enabled := False;

SGAccounts.Enabled := False;

end;

procedure TFAuthorization.FormKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

var

RowForMov: String;

begin

if not BMove.Enabled then

begin

if (key = VK\_UP) and (CurUser <> FirstUser) then

begin

if CurUser^.Next <> nil then

CurUser^.Next^.Prev := CurUser^.Prev;

CurUser^.Prev^.Next := CurUser^.Next;

CurUser^.Next := CurUser^.Prev;

CurUser^.Prev := CurUser^.Prev^.Prev;

CurUser^.Next^.Prev := CurUser;

if CurUser^.Prev <> nil then

CurUser^.Prev^.Next := CurUser

else

FirstUser := CurUser;

RowForMov := SGAccounts.Rows[SGAccounts.Row].Text;

SGAccounts.Rows[SGAccounts.Row].Text :=

SGAccounts.Rows[SGAccounts.Row - 1].Text;

SGAccounts.Rows[SGAccounts.Row - 1].Text := RowForMov;

SGAccounts.Row := SGAccounts.Row - 1;

end;

if (key = VK\_DOWN) and (CurUser^.Next <> nil) then

begin

if CurUser^.Prev <> nil then

begin

CurUser^.Prev^.Next := CurUser^.Next;

CurUser^.Next^.Prev := CurUser^.Prev;

end

else

begin

FirstUser := CurUser^.Next;

FirstUser^.Prev := nil;

end;

CurUser^.Prev := CurUser^.Next;

CurUser^.Next := CurUser^.Prev^.Next;

CurUser^.Prev^.Next := CurUser;

if CurUser^.Next <> nil then

CurUser^.Next^.Prev := CurUser;

SGAccounts.Row := SGAccounts.Row + 1;

RowForMov := SGAccounts.Rows[SGAccounts.Row].Text;

SGAccounts.Rows[SGAccounts.Row].Text := SGAccounts.Rows[SGAccounts.Row - 1].Text;

SGAccounts.Rows[SGAccounts.Row - 1].Text := RowForMov;

end;

if key = VK\_RETURN then

begin

BCreate.Enabled := True;

BDelete.Enabled := True;

BMove.Enabled := True;

BSelect.Enabled := True;

SGAccounts.Enabled := True;

end;

end;

end;

procedure TFAuthorization.SGAccountsDblClick(Sender: TObject);

var

NewName: String;

reiteration: Boolean;

begin

NewName := InputBox('Ðåäàêòèðîâàíèå', 'Ââåäèòå íîâîå èìÿ:', '');

if Length(NewName) <> 0 then

if Length(NewName) > 12 then

begin

ShowMessage('Ñëèøêîì äëèííîå èìÿ. Ïîïðîáóéòå ñíîâà.')

end

else

begin

reiteration := False;

CurUser := FirstUser;

if CurUser^.Name = NewName then

reiteration := True;

while (CurUser^.Next <> nil) and not reiteration do

begin

if CurUser^.Next^.Name = NewName then

reiteration := True;

CurUser := CurUser^.Next;

end;

if not reiteration then

begin

CurUser := FirstUser;

while CurUser.Name <> SGAccounts.Cells[0,SGAccounts.Row] do

CurUser := CurUser^.Next;

CurUser^.Name := NewName;

end

else

begin

ShowMessage('Èìÿ çàíÿòî.');

reiteration := False;

end;

end;

ListToGrid;

end;

end.

{Модуль формы окна добавления аккаунта }

unit UnitCreateAccount;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls;

type

TFCreateAccount = class(TForm)

EInputName: TEdit;

LAdmonition: TLabel;

BInput: TButton;

procedure BInputClick(Sender: TObject);

procedure FormActivate(Sender: TObject);

procedure FormCreate(Sender: TObject);

private

procedure CreatNew;

public

end;

var

FCreateAccount: TFCreateAccount;

implementation

uses UnitMain;

var

reiteration: Boolean;

{$R \*.dfm}

{Процедура создания нового аккаунта}

procedure TFCreateAccount.CreatNew;

begin

if FirstUser = nil then

begin

New(CurUser);

FirstUser := CurUser;

CurUser^.Prev := nil;

end

else

begin

CurUser := FirstUser;

if CurUser^.Name = EInputName.Text then

reiteration := True;

while (CurUser^.Next <> nil) and not reiteration do

begin

if CurUser^.Next^.Name = EInputName.Text then

reiteration := True;

CurUser := CurUser^.Next;

end;

if not reiteration then

begin

New(CurUser^.Next);

CurUser^.Next^.Prev := CurUser;

CurUser := CurUser^.Next;

end;

end;

if not reiteration then

begin

CurUser^.Name := EInputName.Text;

CurUser^.LastEntrance := Time + Date;

CurUser^.Score := 0;

CurUser^.Next := nil;

FCreateAccount.Close;

end

else

begin

ShowMessage('Èìÿ çàíÿòî.');

reiteration := False;

end;

end;

procedure TFCreateAccount.BInputClick(Sender: TObject);

begin

if EInputName.Text <> '' then

begin

CreatNew

end

else

ShowMessage ('Ââåäèòå èìÿ');

end;

procedure TFCreateAccount.FormActivate(Sender: TObject);

begin

EInputName.Text := '';

FCreateAccount.FocusControl(EInputName);

end;

procedure TFCreateAccount.FormCreate(Sender: TObject);

begin

reiteration := False;

end;

end.

{Модуль формы окна статистики}

unit UnitStatistics;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, Grids;

type

TFStatistics = class(TForm)

SGAccounts: TStringGrid;

ESearch: TEdit;

BSearch: TButton;

BCancel: TButton;

procedure BSearchClick(Sender: TObject);

procedure FormActivate(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure SGAccountsMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

private

procedure Sorting(NumOfSorting: Integer; Way: Boolean; Sender: TObject);

procedure Search;

public

end;

var

FStatistics: TFStatistics;

implementation

uses UnitMain;

var

TextForSearch: String;

ByName, Date, Muvs: Boolean;

{$R \*.dfm}

{Процедура поиска записей по указанному имени}

procedure TFStatistics.Search;

begin

SGAccounts.RowCount := 2;

SGAccounts.Rows[1].Clear;

TextForSearch := ESearch.Text;

CurUser := FirstUser;

while CurUser <> nil do

begin

UserForRecord.Name := CurUser^.Name;

if AnsiPos(TextForSearch, UserForRecord.Name) <> 0 then

begin

UserForRecord.LastEntrance := CurUser^.LastEntrance;

UserForRecord.Score := CurUser^.Score;

SGAccounts.Cells[0, SGAccounts.RowCount- 1] := UserForRecord.Name;

SGAccounts.Cells[1, SGAccounts.RowCount - 1] :=

TimeToStr(UserForRecord.LastEntrance) + ' ' +

DateToStr(UserForRecord.LastEntrance);

SGAccounts.Cells[2, SGAccounts.RowCount - 1] :=

IntToStr(UserForRecord.Score);

if CurUser^.Next <> nil then

begin

SGAccounts.RowCount := SGAccounts.RowCount + 1;

SGAccounts.Rows[SGAccounts.RowCount - 1].Clear;

end;

end;

CurUser := CurUser^.Next;

end;

if (SGAccounts.Cells[0, SGAccounts.RowCount - 1] = '') and (SGAccounts.RowCount <> 2) then

SGAccounts.RowCount := SGAccounts.RowCount - 1;

if (SGAccounts.Cells[0, SGAccounts.RowCount - 1] = '') and (SGAccounts.RowCount = 2) then

ShowMessage('Óêàçàííîå èìÿ èëè ÷àñòü èìåíè íå íàéäåíî.');

end;

{Процедура сортировки записейпо указанному параметру}

procedure TFStatistics.Sorting(NumOfSorting: Integer; Way: Boolean; Sender: TObject);

var

f1: Boolean;

begin

f1 := True;

while f1 do

begin

f1 := False;

CurUser := FirstUser;

if CurUser <> nil then

while CurUser^.Next <> nil do

if (NumOfSorting = 1) and (CurUser^.Name < CurUser^.Next^.Name) and Way or

(NumOfSorting = 1) and (CurUser^.Name > CurUser^.Next^.Name) and not Way or

(NumOfSorting = 2) and (CurUser^.LastEntrance < CurUser^.Next^.LastEntrance) and Way or

(NumOfSorting = 2) and (CurUser^.LastEntrance > CurUser^.Next^.LastEntrance) and not Way or

(NumOfSorting = 3) and (CurUser^.Score < CurUser^.Next^.Score) and Way or

(NumOfSorting = 3) and (CurUser^.Score > CurUser^.Next^.Score) and not Way then

begin

if CurUser^.Prev <> nil then

begin

CurUser^.Prev^.Next := CurUser^.Next;

CurUser^.Next^.Prev := CurUser^.Prev;

end

else

begin

FirstUser := CurUser^.Next;

FirstUser^.Prev := nil;

end;

CurUser^.Prev := CurUser^.Next;

CurUser^.Next := CurUser^.Prev^.Next;

CurUser^.Prev^.Next := CurUser;

if CurUser^.Next <> nil then

CurUser^.Next^.Prev := CurUser;

f1 := True;

end

else

CurUser := CurUser^.Next;

end;

case NumOfSorting of

1: ByName := not ByName;

2: Date := not Date;

3: Muvs := not Muvs;

else

end;

FormActivate(Sender);

end;

procedure TFStatistics.BSearchClick(Sender: TObject);

begin

if ESearch.Text <> '' then

begin

Search

end

else

ShowMessage('Ââåäèòå èìÿ èëè ÷àñòü èìåíè.')

end;

procedure TFStatistics.FormActivate(Sender: TObject);

begin

CurUser := FirstUser;

if FirstUser = nil then

SGAccounts.Rows[1].Clear;

SGAccounts.RowCount := 2;

while CurUser <> nil do

begin

UserForRecord.Name := CurUser^.Name;

UserForRecord.LastEntrance := CurUser^.LastEntrance;

UserForRecord.Score := CurUser^.Score;

UserForRecord.Next := CurUser^.Next;

SGAccounts.Cells[0, SGAccounts.RowCount - 1] := UserForRecord.Name;

SGAccounts.Cells[1, SGAccounts.RowCount - 1] := TimeToStr(UserForRecord.LastEntrance) + ' ' + DateToStr(UserForRecord.LastEntrance);

SGAccounts.Cells[2, SGAccounts.RowCount - 1] := IntToStr(UserForRecord.Score);

CurUser := CurUser^.Next;

if CurUser <> nil then

SGAccounts.RowCount := SGAccounts.RowCount + 1;

end;

end;

procedure TFStatistics.FormCreate(Sender: TObject);

begin

SGAccounts.Cells[0,0] := 'Èìÿ';

SGAccounts.Cells[1,0] := 'Ïîñëåäíåå ïîñåùåíèå';

SGAccounts.Cells[2,0] := 'Êîëèåñòâî äåéñòâèé';

ByName := True;

Date := True;

Muvs := True;

end;

procedure TFStatistics.SGAccountsMouseDown(Sender: TObject;

Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

begin

CurUser := FirstUser;

if (Y > SGAccounts.Top - 40) and (Y < SGAccounts.Top + 24 - 40) and (CurUser <> nil) then

begin

if (X > SGAccounts.Left) and (X < SGAccounts.Left + 120) then

if CurUser^.Next <> nil then

Sorting(1, ByName, Sender);

if (X > SGAccounts.Left + 120) and (X < SGAccounts.Left + 240) then

if CurUser^.Next <> nil then

Sorting(2, Date, Sender);

if (X > SGAccounts.Left + 240) and (X < SGAccounts.Left + 360) then

if CurUser^.Next <> nil then

Sorting(3, Muvs, Sender);

end;

end;

end.