**Федеральное государственное образовательное бюджетное учреждение  
высшего образования**

**«Финансовый университет при Правительстве Российской Федерации»**

**КОЛЛЕДЖ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ**

**ПМ.03 Участие в интеграции программных модулей**

**Группа: 1ПКС-114**

**УТВЕРЖДАЮ**

**Председатель цикловой комиссии**

**программирования и баз данных**

**( ) Пестов А.И.**

**\_\_\_\_.\_\_\_\_. 2017**

**ПРОЕКТ КУРСОВОЙ**

**На тему**: «Обучающая программа “Линейное программирование симплекс-метода”».

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**Руководитель курсового проекта**

**( )** Морозова М.В.

**Исполнитель курсового проекта**

**( )** Овчинников В.О

**Оценка за проект: \_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_.\_\_\_\_.2017**

Содержание

Введение…………………………………………….…………………………….3

1. Общий раздел……………………………………………………………...…...4

1.1 Системные требования…………………………………………………….…4

1.2 Характеристика системы программирования………………………………4

2. Технологический раздел………..……………………………………………..6

2.1 Предпроектное исследование предметной области……………………..…6

2.2 Анализ требований и определение спецификаций программного обеспечения……………………………………………………………….……….6

2.3 Проектирование программного обеспечения………………….…………....9

2.4 Разработка пользовательских интерфейсов программного обеспечения..10

2.5 Тестирование и отладка программного обеспечения……………………..13

3. Руководство по использованию программы………………...……………...18

3.1 Руководство системного программиста…………………………………...18

3.2 Руководство программиста…………………………………………………18

3.3 Руководство пользователя…………………………………………………..19

Заключение……………………………………………………………………….21

Список использованных источников……………...……………………………22

Приложение А…………………………………………………...……………….23

Приложение Б………………………………………………………..…………..37

Введение

Обучающая программа «Линейное программирование симплекс-метода» — это программа, которая предназначена помочь студентам ВУЗов в изучении симплекс-метода. Помимо уроков в программе интегрирован калькулятор симплекс-метода и практические задания, которые помогут студенту наглядно показать выполнение данного метода.

Целью курсовой работы является разработка обучающей программы линейного программирования симплекс-метода на языке программирования C#. Это программа содержащая уроки по изучению симплекс-метода и практические примеры применения данного метода.

Актуальность и обоснованность предлагаемого решения:

Обучающая программа линейного программирования симплекс-метода актуальна, как и другие обучающие программы оффлайн режима

В будущем возможны доработки в программе, такие как, загрузка уроков и практических заданий через файл Exel, автоматическое обновление программы.

1. Общий раздел
   1. Системные требования

Системные требования:

- процессор Intel(R) Core(TM) i5-2410M;

- оперативная память 2ГБ и выше;

- жесткий диск 10 Гб;

- устройство чтения компакт-диск;

- SVGA-видеокарта;

- USB-клавиатура;

- Наличие USB-мыши либо сенсорной панели (touchpad).

* 1. Характеристика системы программирования

В данной курсовой работе используется среда разработки Microsoft Visual Studio.

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

C# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Гейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java.

Windows Forms — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework.

Windows Forms позволяет разрабатывать интеллектуальные клиенты. Интеллектуальный клиент — это приложение с полнофункциональным графическим интерфейсом, простое в развертывании и обновлении, способное работать при наличии или отсутствии подключения к Интернету и использующее более безопасный доступ к ресурсам на локальном компьютере по сравнению с традиционными приложениями Windows. Windows Forms включает широкий набор элементов управления, которые можно добавлять на формы: текстовые поля, кнопки, раскрывающиеся списки, переключатели и даже веб-страницы. Список всех элементов управления, которые можно использовать в форме, представлены в разделе Элементы управления для использования в формах Windows Forms. Если существующий элемент управления не удовлетворяет потребностям, в Windows Forms можно создать пользовательские элементы управления с помощью класса UserControl.

Библиотеки языка программирования C# (стандартная и используемая для написания данного программного продукта):

- using System.Windows.Forms (Пространство имен содержит классы для создания приложений Windows, содержит всякие классы контролов, типа кнопок, тектовых полей предоставляемых в операционной системе Microsoft Windows.);

- usingSystem.Collections.Generic (работа с обобщёнными коллекциями);

- usingSystem.Drawing (классы для рисования на формах );

- using System.Text (для работы с текстом).

1. Технологический раздел
   1. Предпроектное исследование предметной области
      1. Описательную модель предметной области

Разработать обучающую программу линейного программирования симплекс-метода на языке программирования C#. Данный проект будет представлять собой приложение в котором реализованы интерактивные уроки с практическими заданиями и калькулятор линейного симплекс-метода. Цель программы — помочь студенту в изучении симплекс-метода.

2.1.2 Описание входной/выходной информации

Описание входной информации

- выбор урока или практического задания;

- ввод значений для калькулятора симплекс-метода.

Описание выходной информации

- текст выбранного урока или практического задания;

- результат решения задачи в калькуляторе симплекс-метода.

2.1.3 Требования к программному обеспечению

Функциональные требования:

- программа должна иметь уроки на тему «Линейное программирование симплекс-метода».

Нефункциональные требования:

- калькулятор симплекс-метода.

Функции выполняемые программой:

- демонстрация урока данной темы;

- демонстрация практического задания;

- решение задачи симплекс-метода;

- требования к интерфейсу – простой интерфейс и легкая навигация.

* 1. Анализ требований и определение спецификаций программного обеспечения.

- функциональная диаграмма Функциональная диаграмма представлена на рисунке 1. На ней можно увидеть, что на вход программе подаётся выбор урока. На выходе у программы вывод результатов.

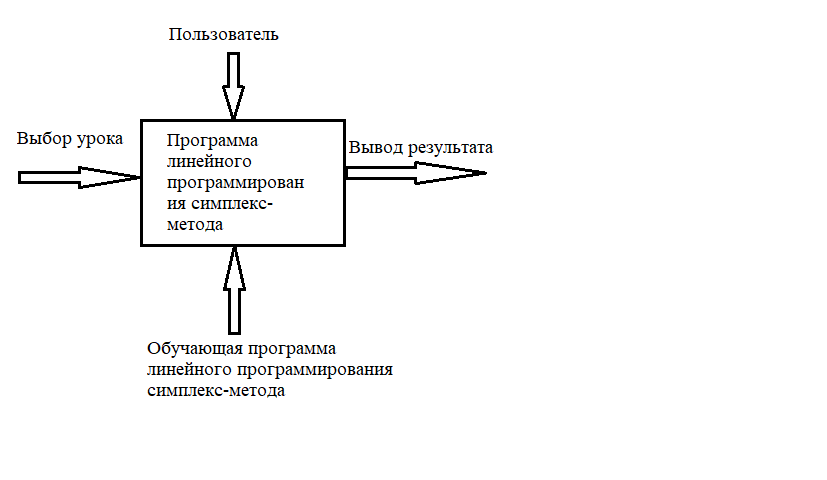


Рисунок 1 – Функциональная диаграмма

- детализированная функциональная диаграмма представлена на рисунке 2. Данная диаграмма содержит основные функции программы:

- вывод меню;

- вывод выбранного пункта меню;

- обработка введенных данных;

- вывод результата.

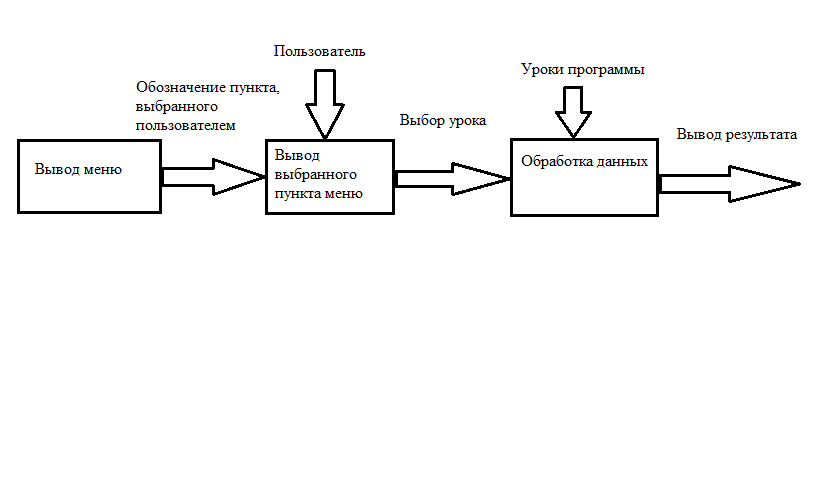
На вход программе подаётся данные о выбранном урок, а на выходе пользователь получает результат. 

Рисунок 2 – Детализированная функциональная диаграмма

- диаграмма потоков данных представлена на рисунке 3, и представляет собой взаимодействие пользователя и приложения, связанных потоками данных. На диаграмме можно увидеть, что от пользователя к программе поступают такие действия, как:

- выбор урока;

Программа, в свою очередь, выводит данный урок в окно программы.

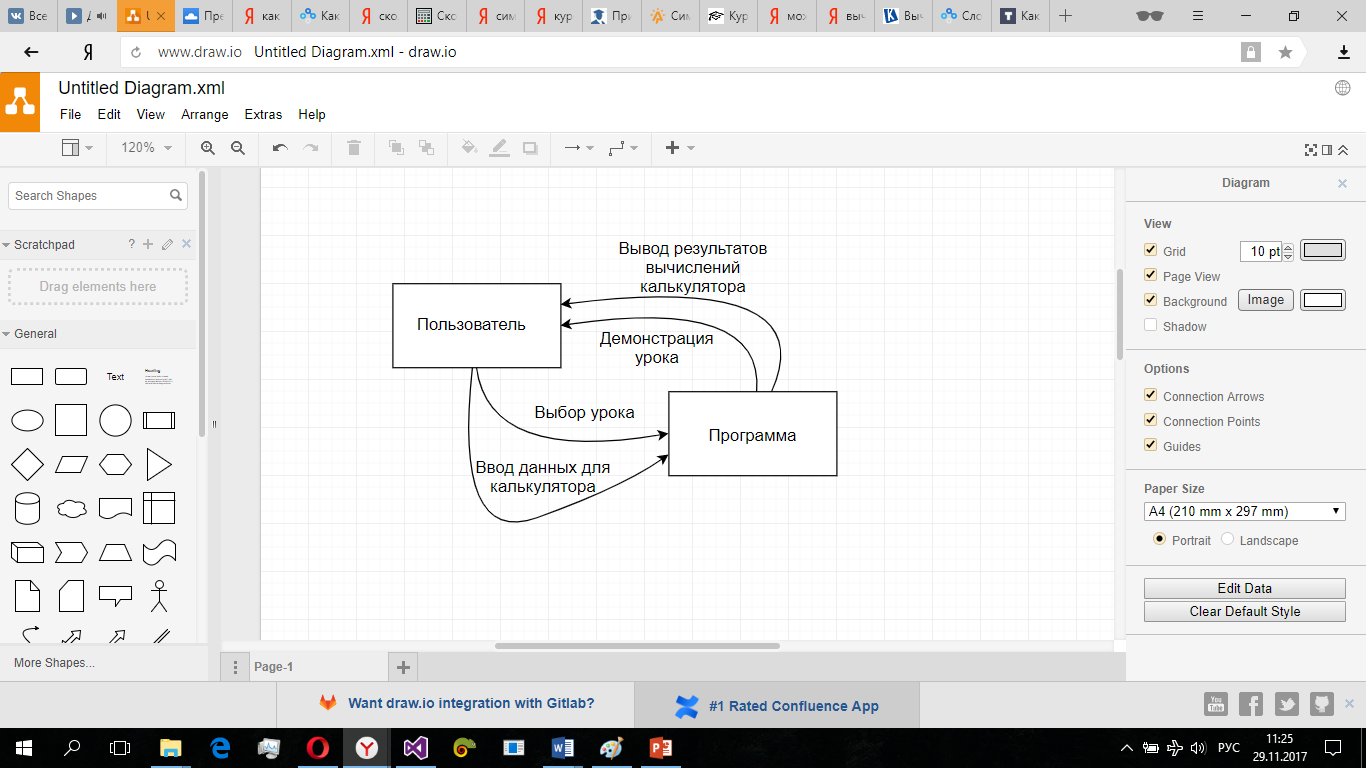


Рисунок 3 – Диаграмма потоков данных

-детализированная диаграмма потоков данных является более подробной и представлена на рисунке 4. На ней можно увидеть поэтапную работу и отклик программы на действие пользователя.

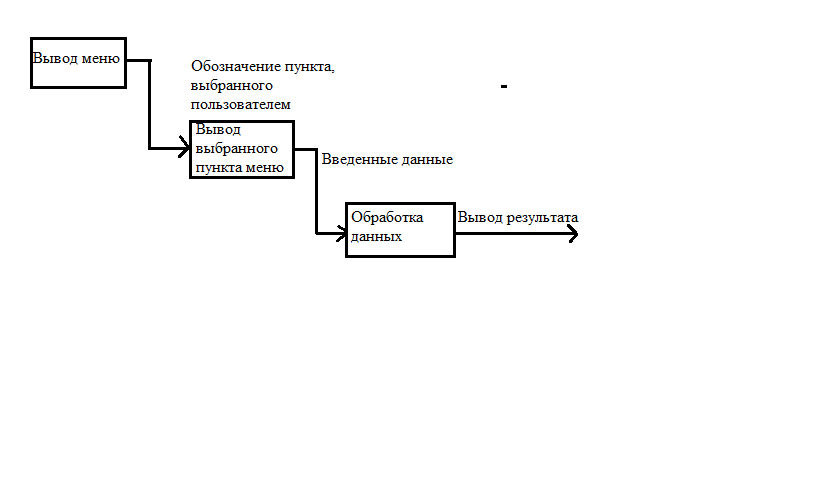


Рисунок 4 – Детализированная диаграмма потоков данных

- диаграмма вариантов использования — диаграмма, отражающая отношения между пользователем и программой. Как видно на этой диаграмме, пользователь может выбрать урок или практическое задание, использовать калькулятор симплекс-метода для решения задачи линейного программирования, выбор настроек программы, получить информацию о программе.

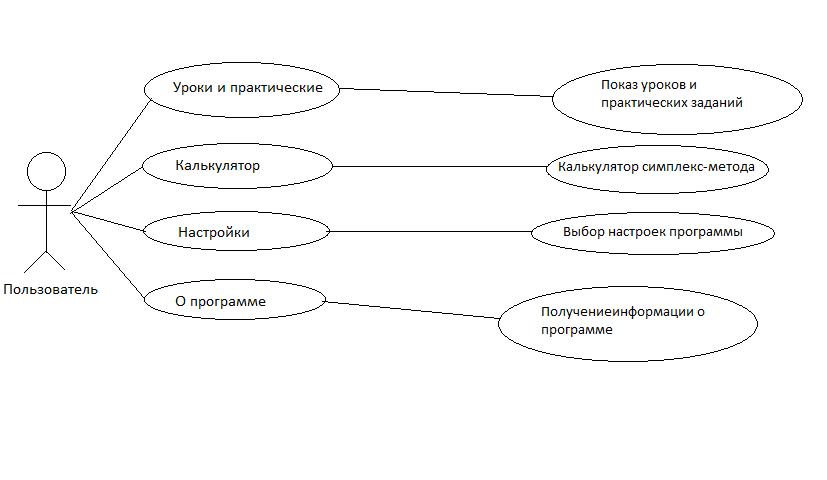


Рисунок 5 – Диаграмма вариантов использования

* 1. Проектирование программного обеспечения

- структурная схема — это совокупность элементарных звеньев объекта и связей между ними. Выбирая урок, программа выводит результат. Введя все данные для решения задачи линейного программирования в калькуляторе, программа выведет ответ данной задачи. Также пользователь может посмотреть раздел о программе и настройки.

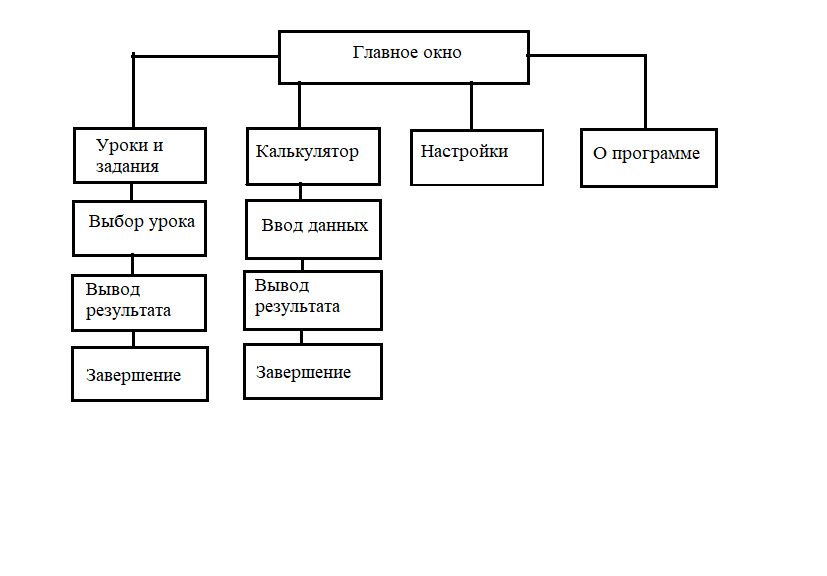


Рисунок 6 – Структурная схема

- функциональная схема – описывает функциональные механизмы программы.

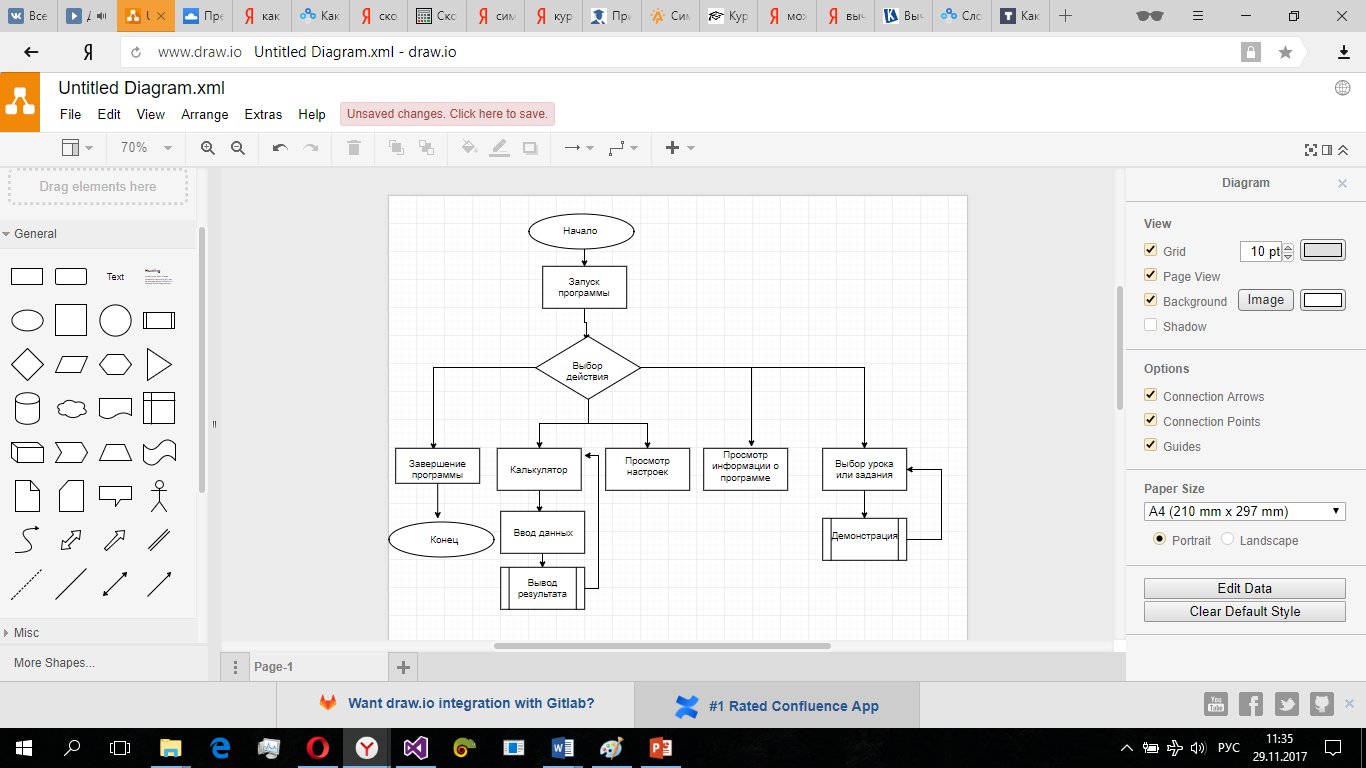


Рисунок 7 – Функциональная схема

* 1. Разработка пользовательских интерфейсов программного обеспечения

На рисунке 8 “Главное меню” показана панель где пользователь может выбрать нужный ему раздел.

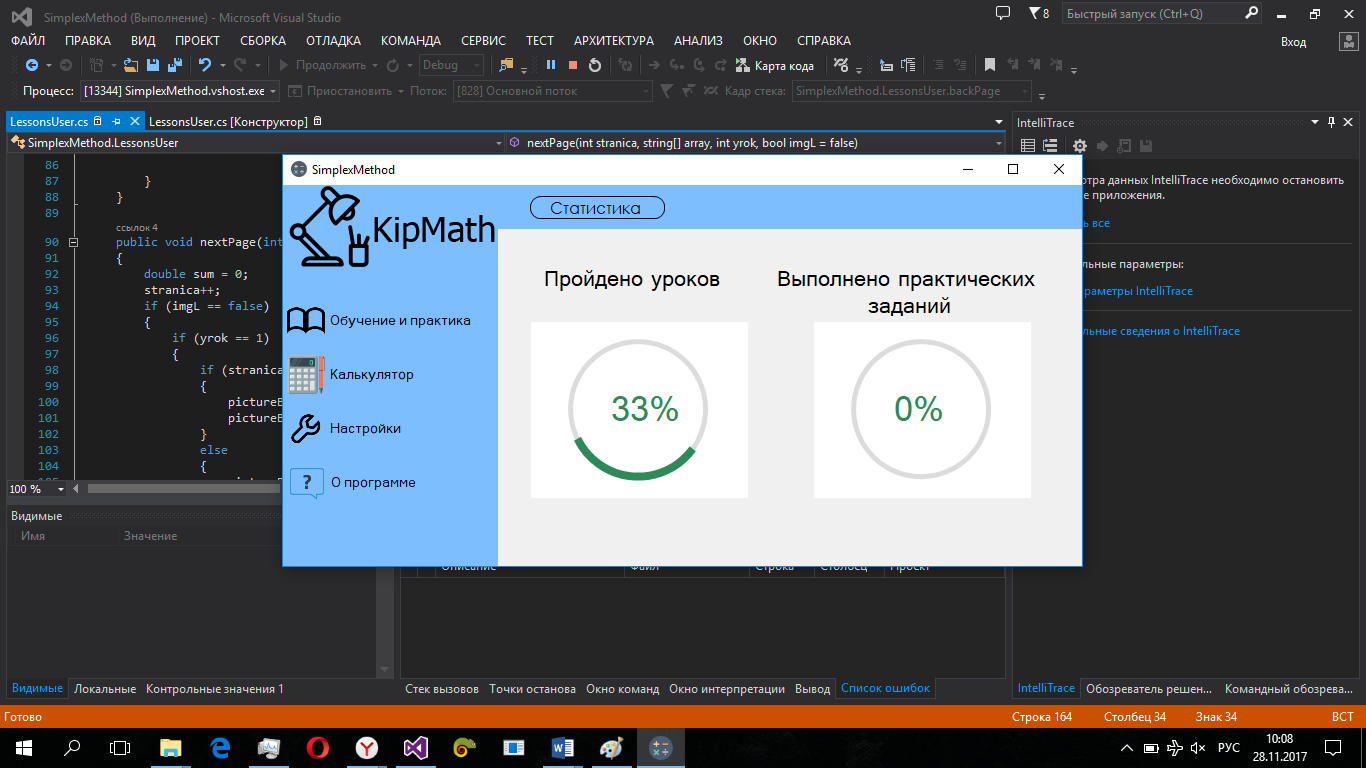


Рисунок 8 – Главное меню

На рисунке 9 “Уроки и практические задания” изображена панель программы.

В окне имеются следующие поля и кнопки:

- Уроки;

- Практические задания.

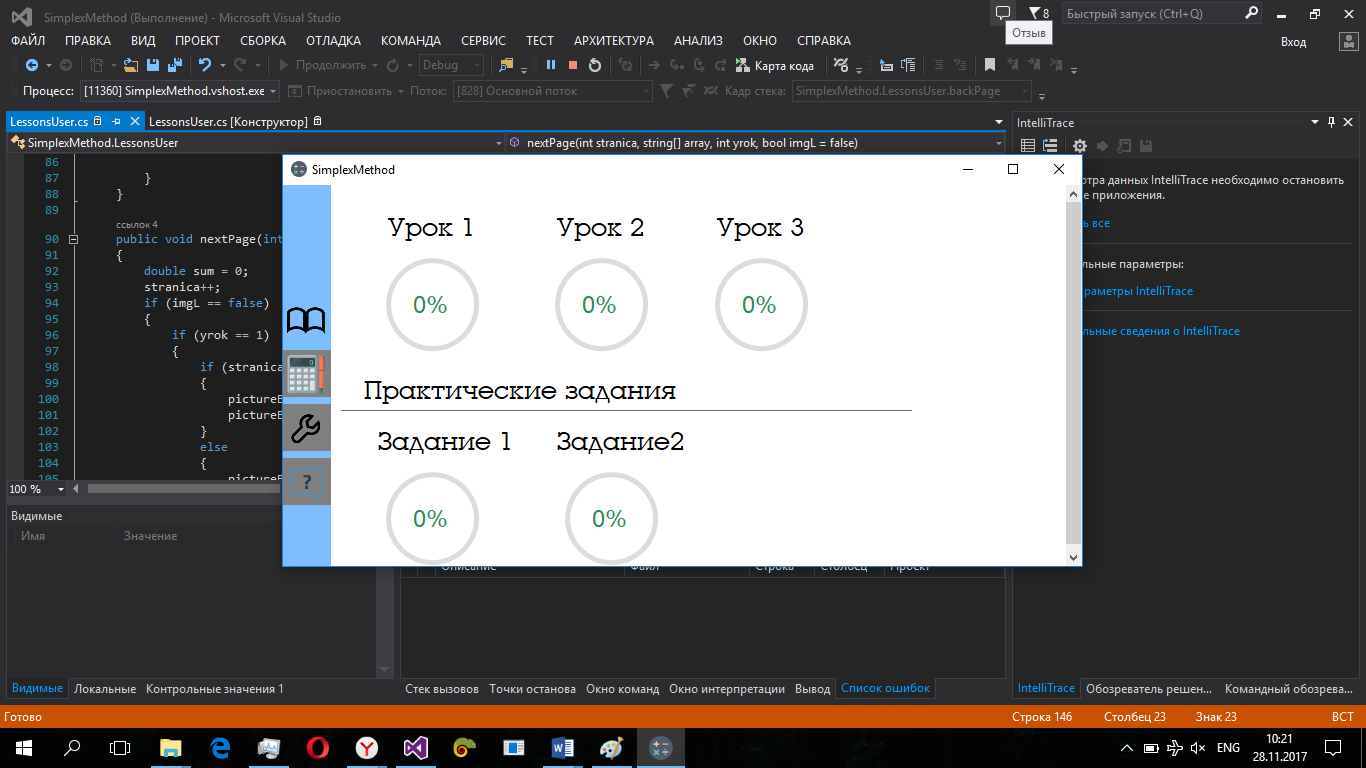


Рисунок 9 – Уроки и практические задания

На рисунке 10 "Калькулятор симплекс-метода" изображена панель калькулятора симплекс-метода.

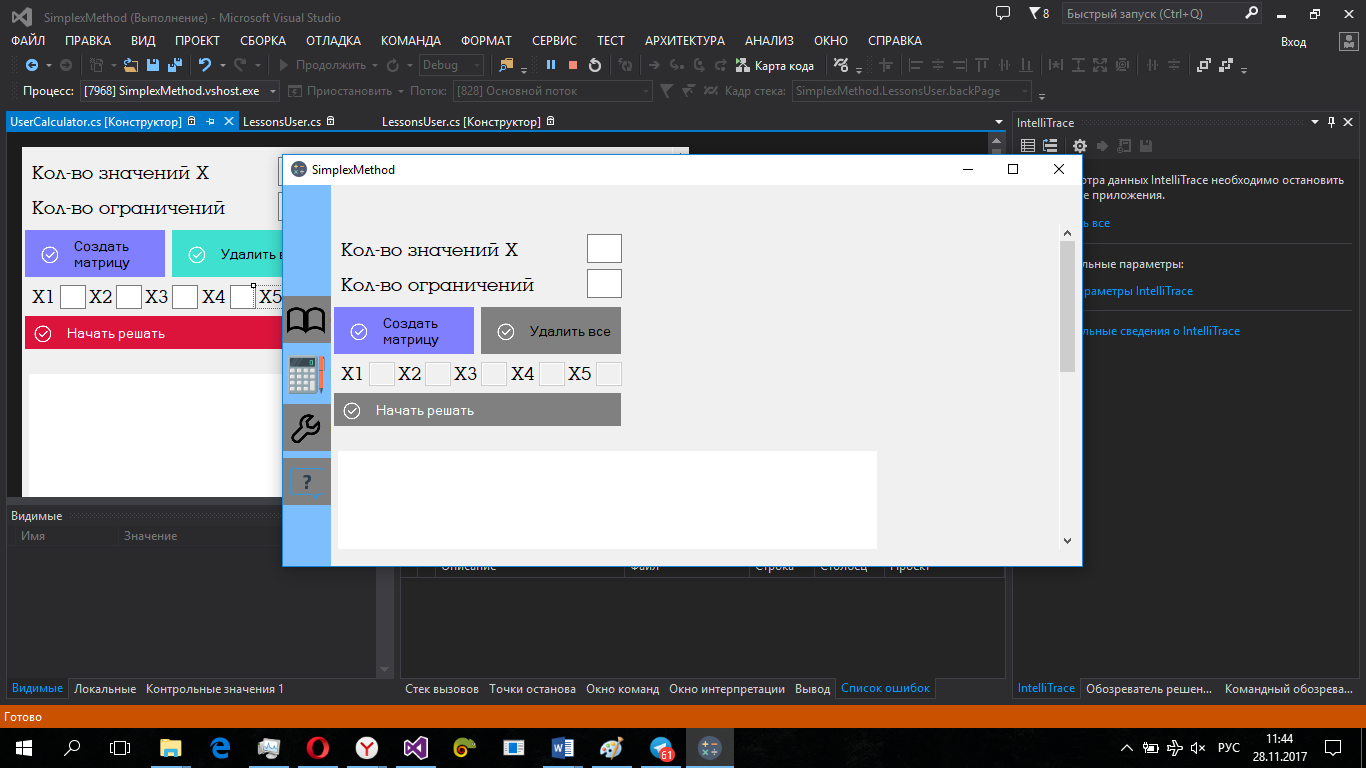


Рисунок 10 – Калькулятор симплекс-метода

На рисунке 11 "Настройки" изображена панель настроек.

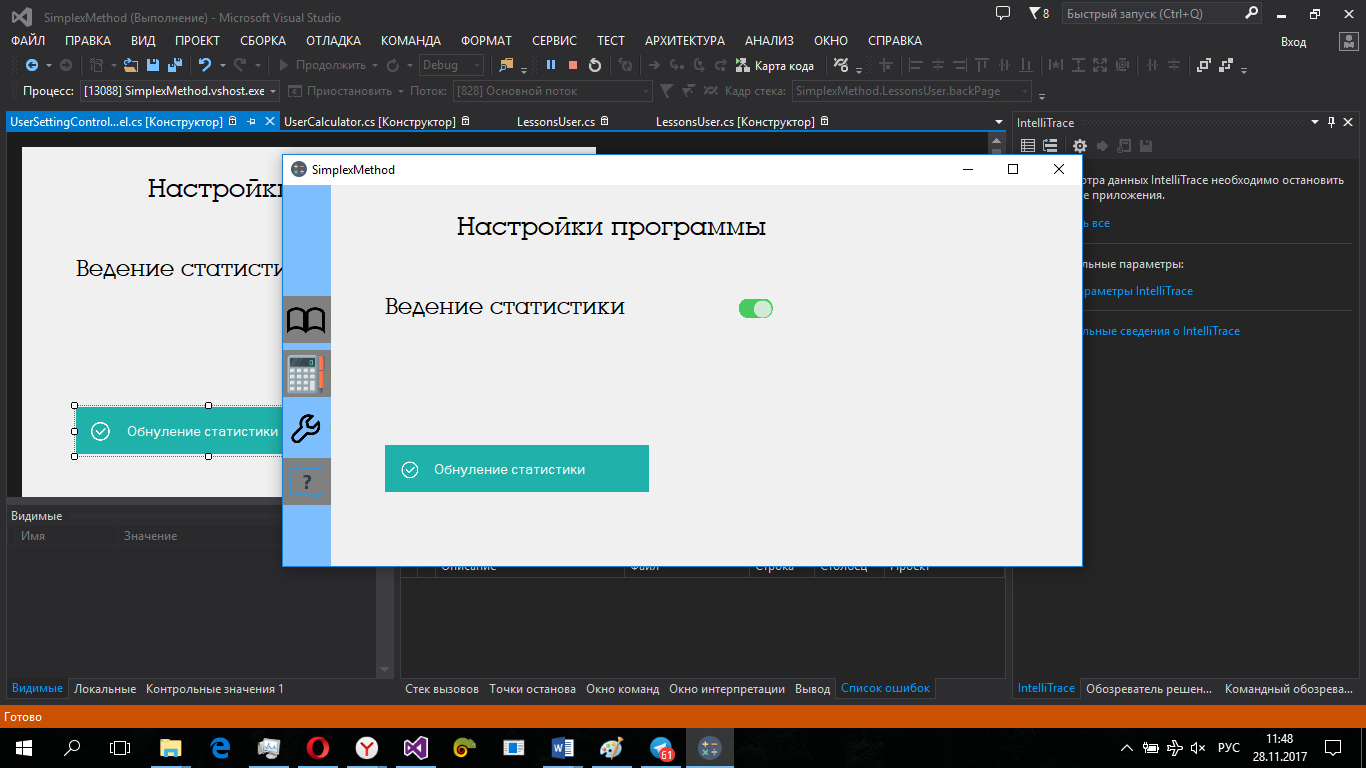


Рисунок 11 – Настройки

На рисунке 12 "О программе" изображена панель информации о программе.

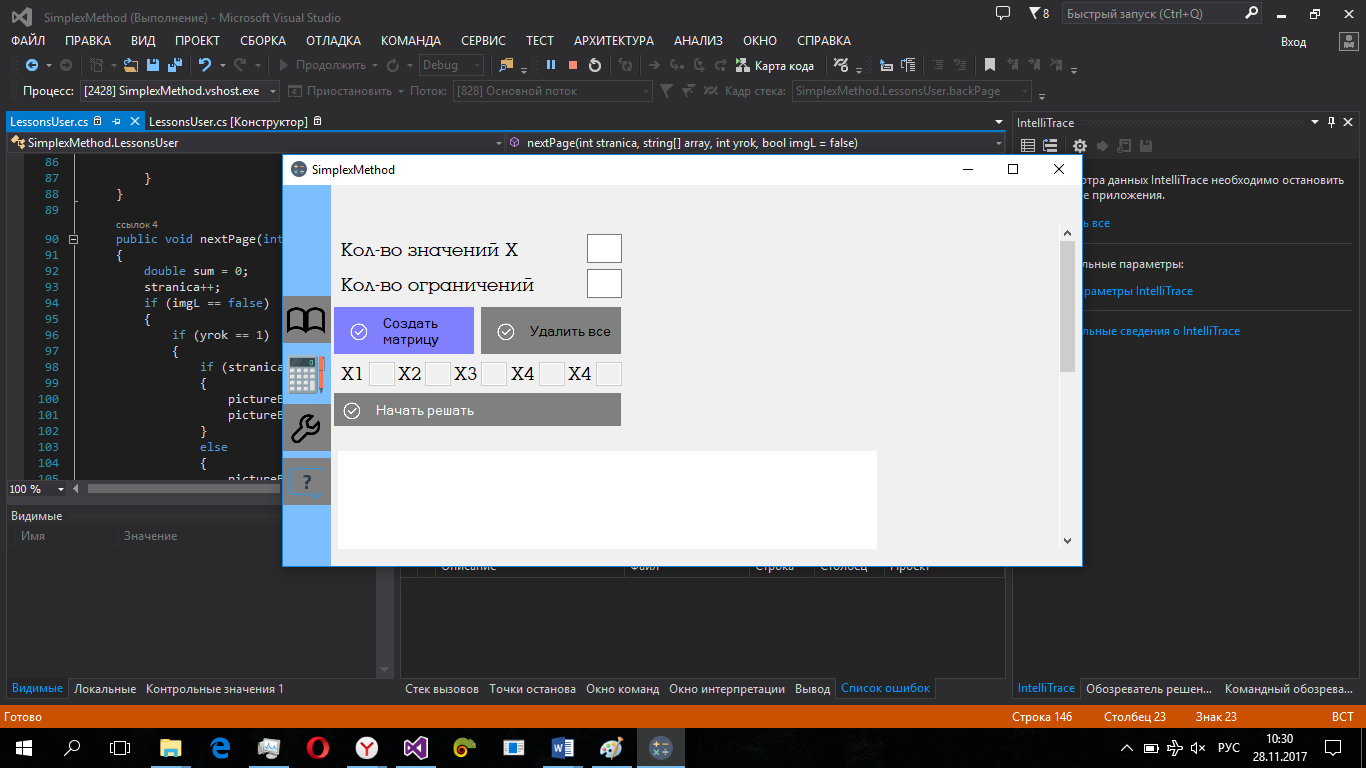


Рисунок 12 – О программе

* 1. Тестирование и отладка программного обеспечения

На рисунке 13 “Главное меню” показана панель где пользователь может перейти к урокам, открыть калькулятор симплекс-метода, настройки и информация о программе.

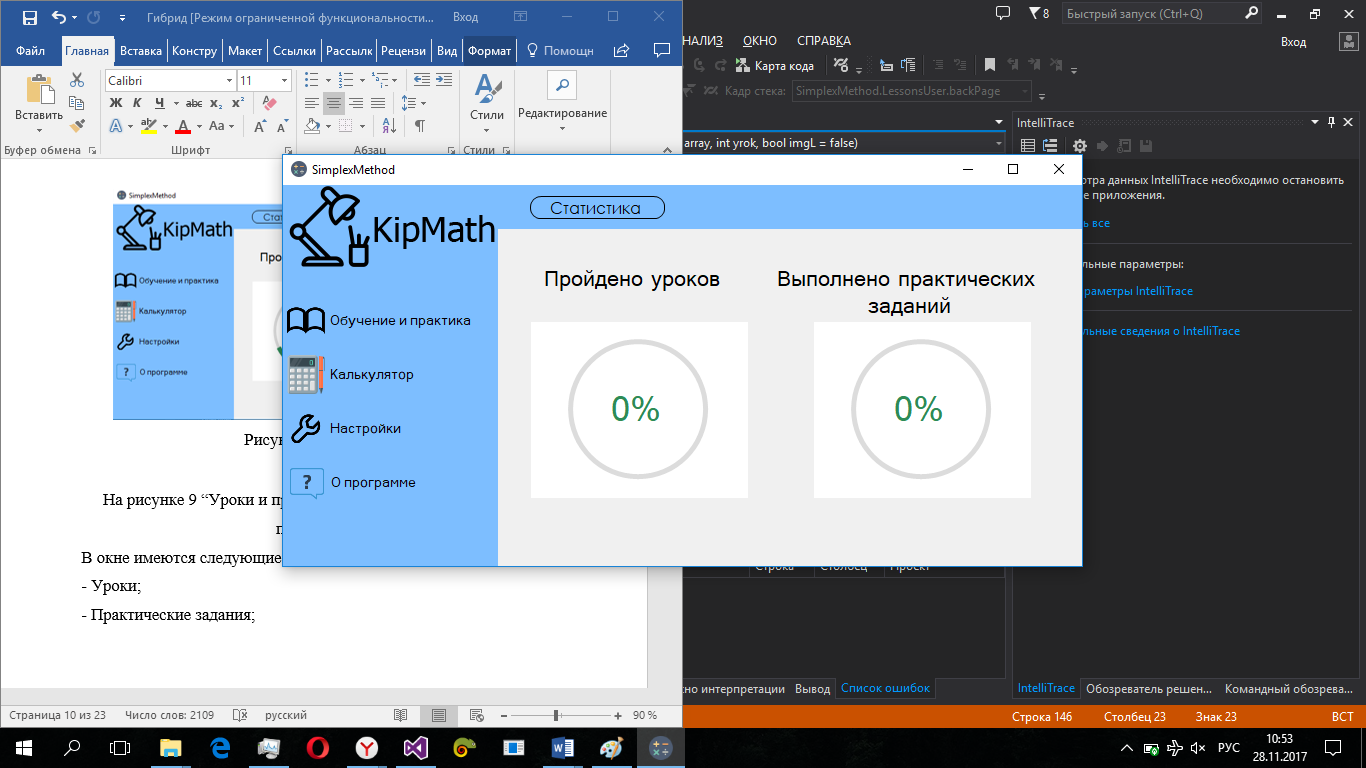


Рисунок 13 – Главное меню

На рисунке 14 “Уроки и практические задания” изображена панель выбора уроков или практического задания.

В окне имеются следующие поля и кнопки:

- Номера уроков и практических заданий;

- Статистика изучения урока ил практического задания;

- Меню – возврат в стартовое меню.

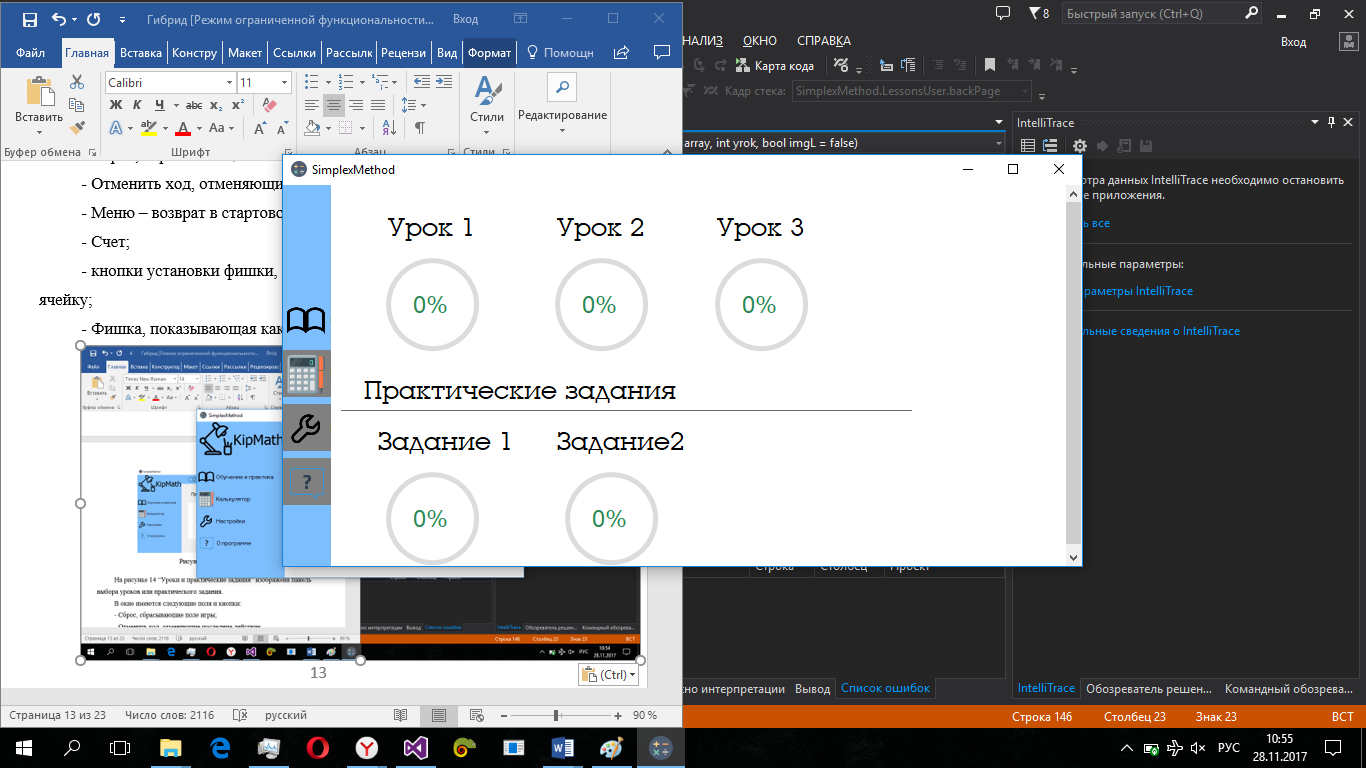


Рисунок 14 – Уроки и практические задания

На рисунке 15 “ Процесс демонстрации урока” изображен урок выбранный пользователем

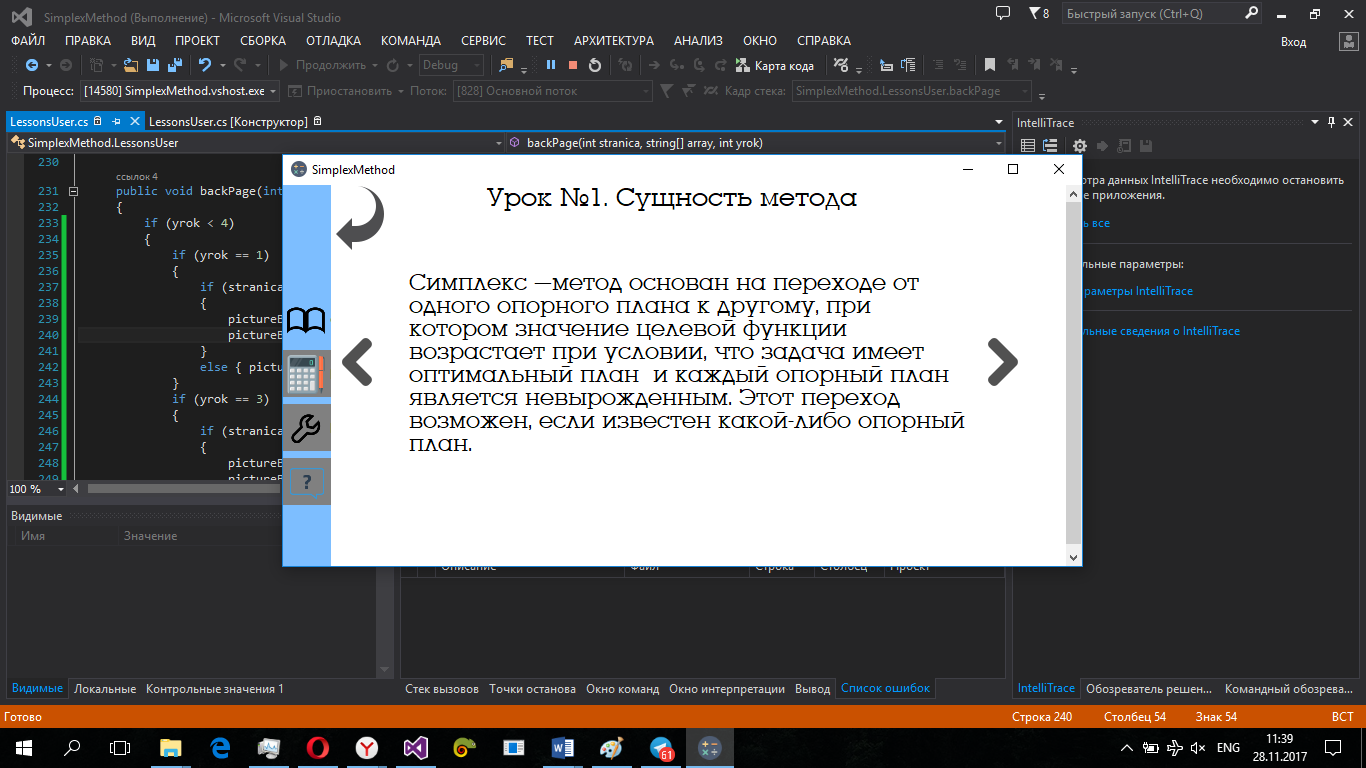


Рисунок 15 – Процесс демонстрации урока

На рисунке 16 “ Процесс демонстрации практического задания ” изображено практическое задание выбранное пользователем.

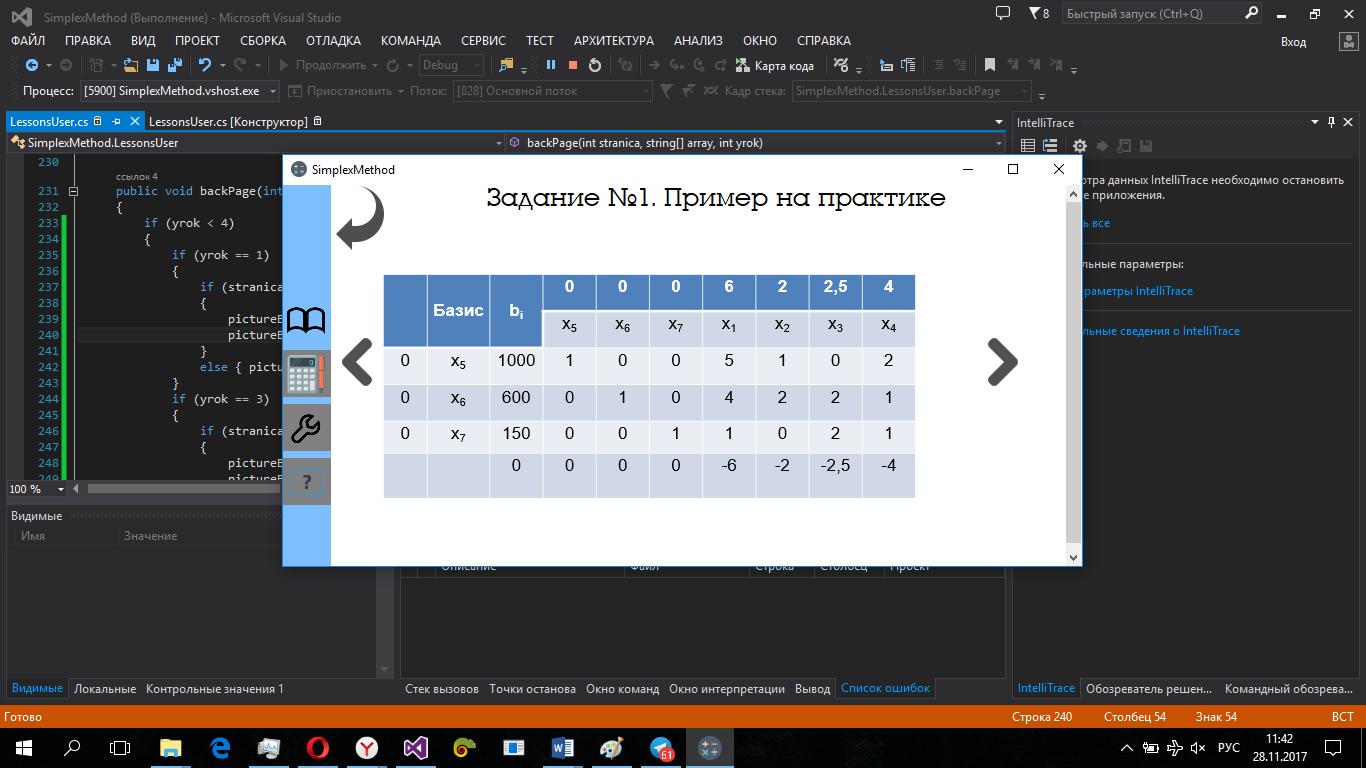


Рисунок 16 – Процесс демонстрации практического задания

На рисунке 17 "Калькулятор симплекс-метода" калькулятора симплекс-метода.

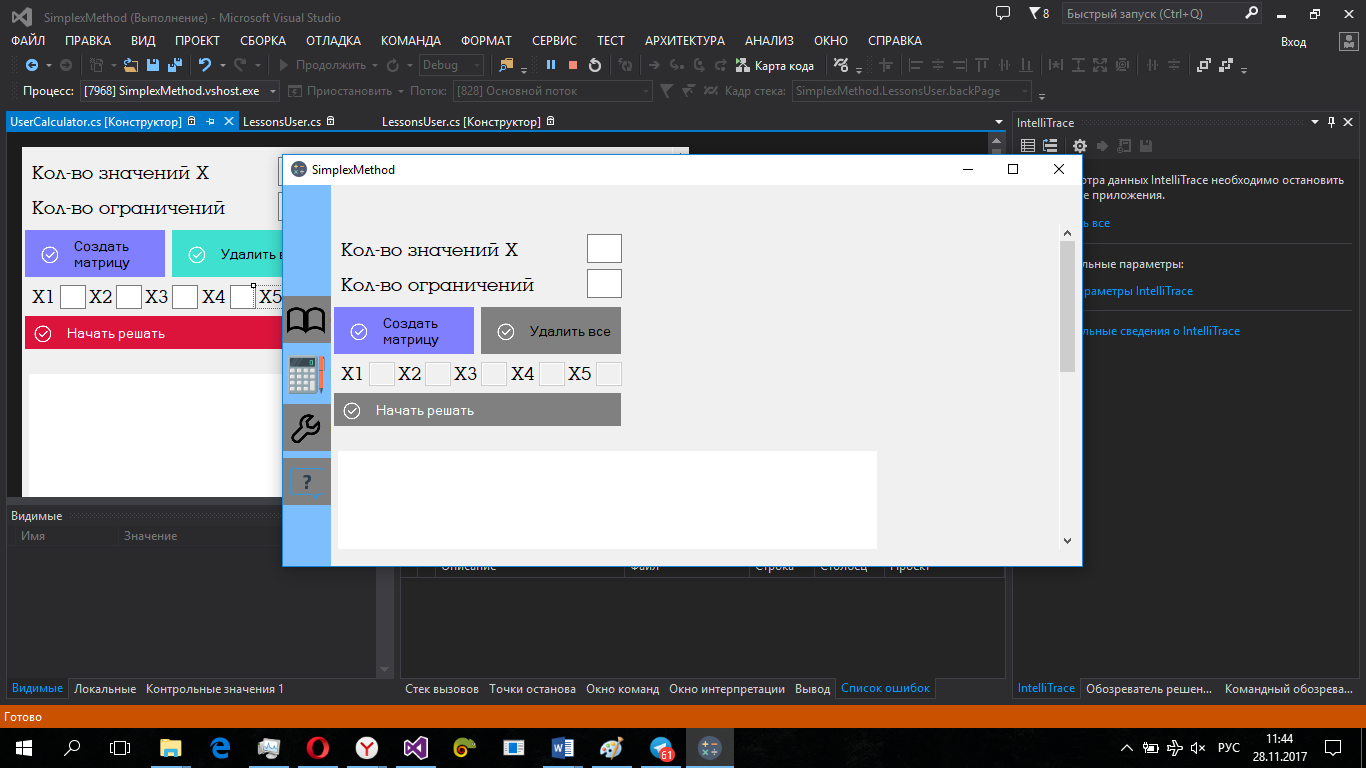


Рисунок 17 – Калькулятор симплекс-метода

На рисунке 18 "Настройки" изображена панель настроек программы.

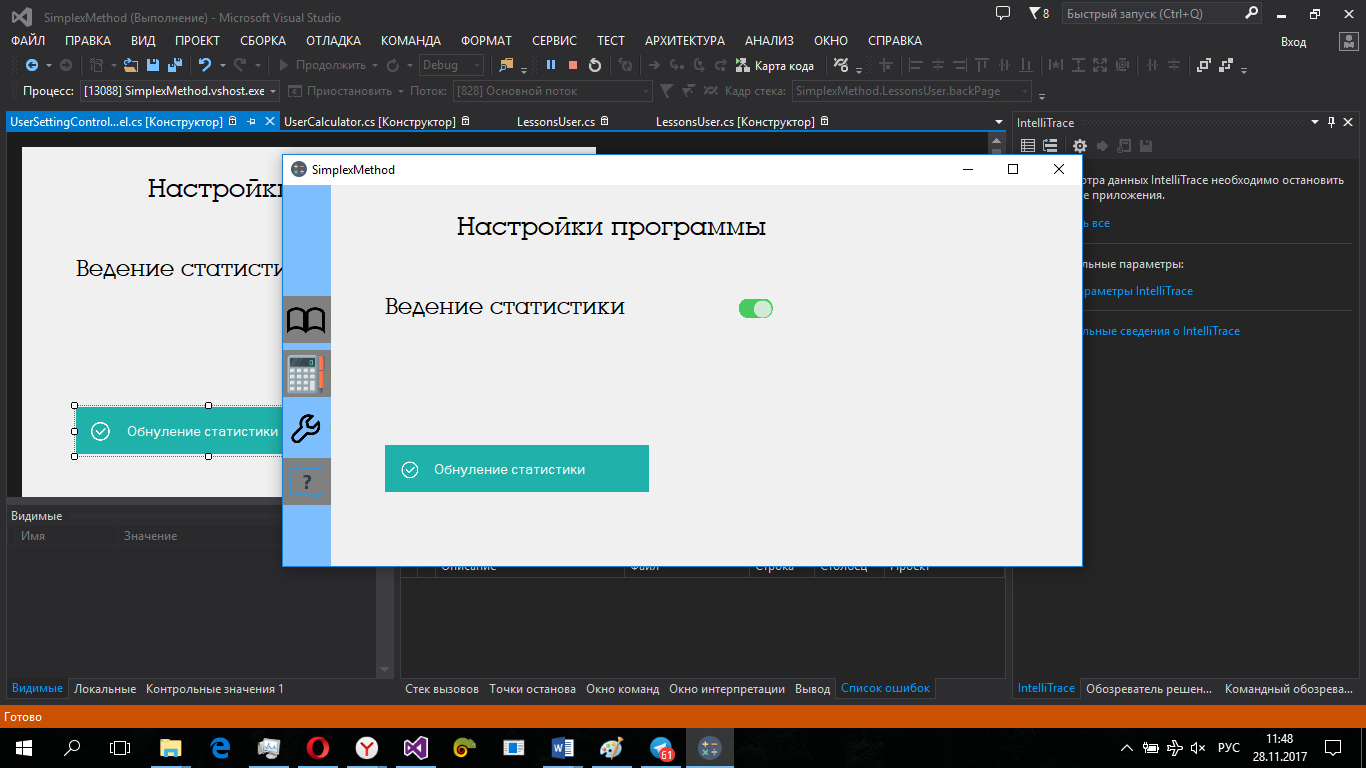


Рисунок 18 – Настройки

На рисунке 19 "О программе" изображена панель с информацией о программе.

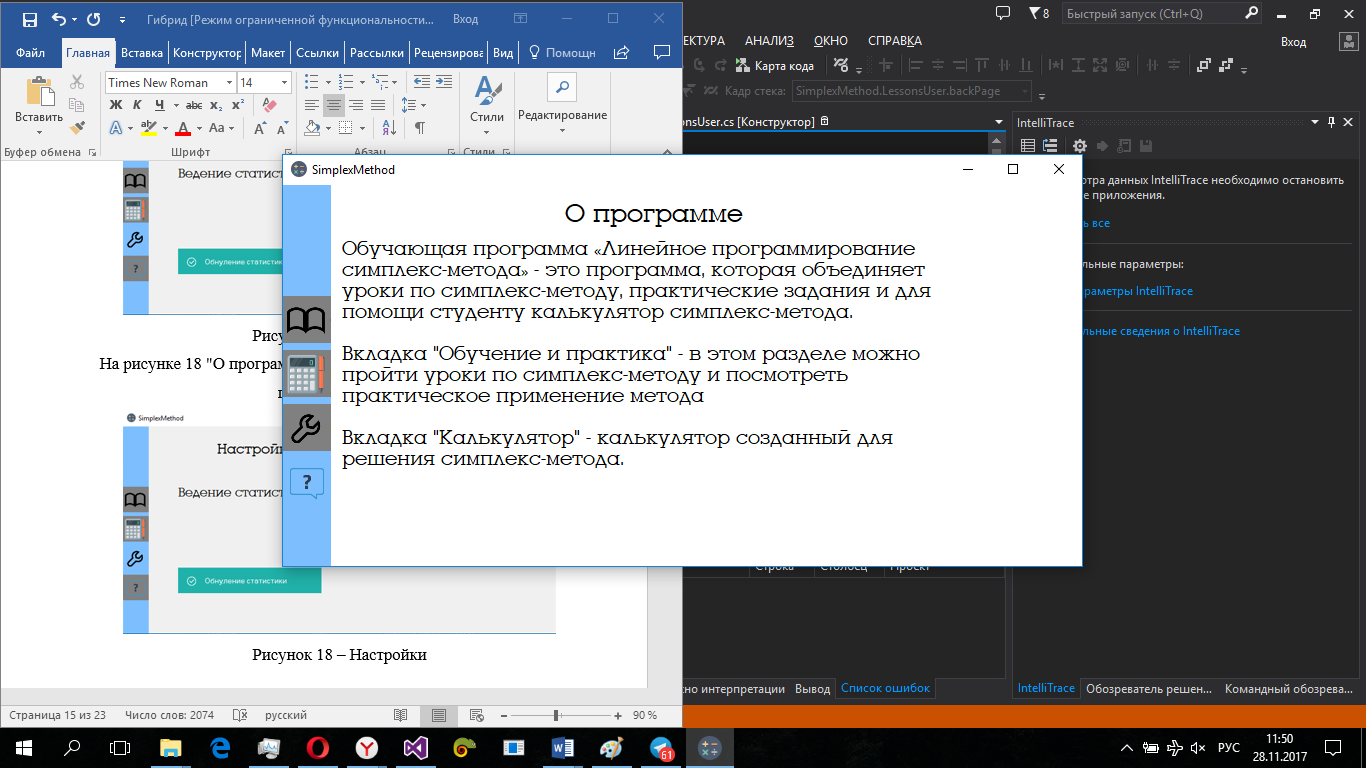


Рисунок 19 – О программе

* + 1. Анализ результатов тестирования

На рисунке 13, “Главное меню” У нас есть выбор: перейти к изучению уроков, открыть калькулятор симплекс-метода, открыть настройки, открыть информационную панель. При выборе кнопки “Обучение и практика” открывается панель с выбором урока, показано на рисунке 9. При выборе кнопки “Калькулятор” открывается окно с панелью калькулятора, показано на рисунке 10. При выборе кнопки “Настройки” открывается панель с настройками программы. При выборе кнопки “О программе” открывается панель с информацией о программе.

1. Руководство по использованию программы
   1. Руководство системного программиста
      1. Общие сведения о программе

Обучающая программа линейного программирования симплекс-метода предназначена помогать студентам в изучении симплекс-метода.

Функции, выполняемые программой:

- демонстрация уроков;

- демонстрация практических заданий;

- калькулятор симплекс-метода.

* + 1. Структура программы

Программа состоит из одной формы и 4 панелей, которые отображаются в зависимости от выбранного раздела.

* + 1. Проверка программы

Для проверки работоспособности программы можно начать изучение урока, при переходе на следующую страницу текст урока будет изменяться, что свидетельствует о том что программа работает корректно.

* 1. Руководство программиста
     1. Назначение и условия применения программы

Программа представляет собой набор уроков для студента с практическими заданиями. Уроки будут разбиты на разделы и пользователь сможет изучить тот раздел, который ему необходим. Так же в программе присутствует калькулятор для решения линейного симплекс-метода, настройка программы и панель с информацией о программе.

* + 1. Характеристики программы

Системные требования:

- процессор Intel(R) Core(TM) i5-2410M;

- оперативная память 2ГБ и выше;

- жесткий диск 10 Гб;

- устройство чтения компакт-диск;

- SVGA - видеокарта;

- USB - клавиатура;

- Наличие USB-мыши либо сенсорной панели (touchpad).

* + 1. Обращение к программе

Для запуска программы необходимо зайти в среду разработки Visual Studio. Также ее можно запустить с помощью соответствующего ехе файла.

* + 1. Входные и выходные данные

Входной информацией в данной программе является выбор пользователем урока, ввод данных в калькулятор.

Выходной информацией демонстрация уроков, результат решения задачи в калькуляторе.

* 1. Руководство пользователя
     1. Назначение программы

Программа представляет собой набор уроков для студента с практическими заданиями. Уроки будут разбиты на разделы и пользователь сможет изучить тот раздел, который ему необходим. Так же в программе присутствует калькулятор для решения линейного симплекс-метода, настройка программы и панель с информацией о программе.

* + 1. Условия выполнения программы

Системные требования:

- процессор Intel(R) Core(TM) i5-2410M;

- оперативная память 2ГБ и выше;

- жесткий диск 10 Гб;

- устройство чтения компакт-диск;

- SVGA - видеокарта;

- USB - клавиатура;

- Наличие USB-мыши либо сенсорной панели (touchpad).

* + 1. Выполнение программы

Для запуска программы необходимо зайти в среду разработки Visual Studio. Также ее можно запустить с помощью соответствующего ехе файла. После чего необходимо нажать кнопку «Начать игру» . После игроки ходят по очереди, роняя фишки в ячейки доски, пока один из них расположить подряд по горизонтали, вертикали или диагонали четыре фишки своего цвета. После этого ряд из четырех собранных фишек поменяет цвет и появиться сообщение о выигрыше игрока, при этом поле очистится, а игрок получит одно очко.

Заключение

В данной курсовой работе представлен проект обучающей программы линейного программирования симплекс-метода, реализованного на языке программирования C#. Дополнительные средства разработки и возможности среды программирования позволили осуществить вывод необходимых форм, создать удобный пользовательский интерфейс.

В ходе выполнения курсовой работы были закреплены знания программирования, и подробно изучена новая среда. Написание программы способствовало закреплению теоретического материала на практике.

Целью курсовой работы являлось создание и разработка обучающей программы линейного программирования симплекс-метода в графическом режиме.

Для ее разработки я использовал среду Visual Studio, язык программирования C#. Он позволил создать графический интерфейс для взаимодействия пользователя с программой.

Итог курсовой работы соответствует поставленной задачи. В будущем возможны доработки в программе, такие как, автоматического обновления программы и добавление уроков через Exel файл.

Список использованных источников

* 1. Введение в язык C# и .NET Framework. – [Электронный ресурс]. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/z1zx9t92.aspx
  2. ГОСТ 7.1. – 2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2004. – 169 с.
  3. ГОСТ 7.32 – 2001. Система стандартов по информацию, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: ИПК Издательство стандартов, 2001. – 21 с.
  4. Гуриков С.Р. Введение в программирование на языке Visual C#: учебное пособие. – М.: ФОРУМ: ИНФРА-М, 2013.
  5. Макконелл С. Совершенный код – Microsoft Press, Русская Редакция, 2014.
  6. Полное руководство по языку программирования C# 6.0 и платформе .NET 4.6. – [Электронный ресурс]. – Режим доступа: http://metanit.com/sharp/tutorial/
  7. Рудаков А.В. Технология разработки программных продуктов: учебник для студ. учреждений сред. проф. образования. – М.: Академия, 2013.

Приложение А

Программный модуль Form1.cs

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 SimplexMethod

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

public void enabledElement(bool action)

{

LogoPanel.Visible = action;

statControlPanel.Visible = action;

HeaderPanel.Visible = action;

bunifuThinButton21.Visible = action;

buttonTrainig.Enabled = action;

buttonCalculator.Enabled = action;

buttonSettings.Enabled = action;

buttonAbout.Enabled = action;

}

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

{

if (SidePanel.Width == 215)

{

enabledElement(false);

SidePanel.Width = 48;

userSettingControlPanel1.Visible = true;

buttonSettings.Enabled = true;

}

else if (SidePanel.Width == 48)

{

enabledElement(true);

SidePanel.Width = 215;

userSettingControlPanel1.Visible = false;

}

}

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

{

if(SidePanel.Width == 215)

{

enabledElement(false);

SidePanel.Width = 48;

lessonsUser1.Visible = true;

buttonTrainig.Enabled = true;

}

else if (SidePanel.Width == 48 )

{

SidePanel.Width = 215;

lessonsUser1.Visible = false;

enabledElement(true);

}

}

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

{

if (SidePanel.Width == 215)

{

enabledElement(false);

SidePanel.Width = 48;

userCalculator1.Visible = true;

buttonCalculator.Enabled = true;

}

else if (SidePanel.Width == 48 )

{

userCalculator1.Visible = false;

SidePanel.Width = 215;

enabledElement(true);

}

}

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

{

if (SidePanel.Width == 215)

{

enabledElement(false);

SidePanel.Width = 48;

userAbout1.Visible = true;

buttonAbout.Enabled = true;

}

else if (SidePanel.Width == 48 )

{

userAbout1.Visible = false;

SidePanel.Width = 215;

enabledElement(true);

}

}

}

}

Программный модуль LessonsUser.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace SimplexMethod

{

public partial class LessonsUser : UserControl

{

int count1 = 0,count2=0,count3=0,countImg1=0;

string[] arrayLesson1 = {

"",

"\nСимплекс –метод основан на переходе от одного опорного плана к другому, при котором значение целевой функции возрастает при условии, что задача имеет оптимальный план и каждый опорный план является невырожденным. Этот переход возможен, если известен какой-либо опорный план.",

"\nВ этом случае каноническая задача линейного программирования должна содержать единичную подматрицу порядка m \nТогда очевиден первоначальный опорный план( неотрицательное базисное решение системы ограничений КЗЛП).",

"Метод применим к любой задаче линейного программирования в канонической форме",

"Основные этапы решения задачи:\n1)Приведение задачи к каноническому виду.\n2)Приведение задачи к допустимому виду (выделение базиса) и преобразование целевой функции.\n3)Нахождение первого допустимого базисного решения системы ограничений или установление факта ее несовместности.",

"\n4)Проверка полученного решения на оптимальность.\n5)Если решение не оптимально, то поиск другого допустимого базисного решения, при котором целевая функция достигает как минимум не меньшего значения. \nП. 4 и 5 повторяются до нахождения оптимального решения",

"\nДля приведения задачи к каноническому виду необходимо добавить в каждое из ограничений задачи, представленных неравенствами, по одной переменной.\nДля того, чтобы привести систему уравнений к допустимому виду, необходимо выразить любые m неизвестных через остальные",

"Неизвестные, которые выражаются через остальные неизвестные, называются базисными, а весь набор этих неизвестных – базисом.\nОстальные неизвестные называются свободными",

"Далее необходимо преобразовать целевую функцию, исключив из нее базисные переменные.\n Для исключения базисных переменных из целевой функции нужно умножить первое уравнение системы ограничений на c1, второе на c2, и т.д., сложить полученные произведения и вычесть целевую функцию",

"Таким образом симплекс-метод представ-ляет собой процедуру направленного перебора опорных решений. "};

string[] arrayLesson2 = {

"",

"1. Просматривается последняя строка таблицы, среди коэффициентов этой строки (исключая столбец свободных членов) выбирается наименьшее отрицательное число. Если такового нет, то исходное базисное решение является оптимальным.",

"2. Столбец таблицы, соответствующий выбранному отрицательному коэффици-енту в последней строке называется ключевым. В этом столбце выбираются положительные коэффициенты. Если таковых нет, то задача решений не имеет.",

"3. Среди положительных коэффициентов ключевого столбца выбирается тот, для которого абсолютная величина отношения соответствующего свободного члена к этому элементу минимальна. Этот коэффициент называется разрешающим или ключевым, а строка, в которой он находится, ключевой.",

"4. Базисная переменная, отвечающая строке ключевого элемента, должна быть переведена в разряд свободных, а свободная переменная, отвечающая столбцу ключевого элемента, вводится в число базисных. Новая таблица строится по следующему алгоритму.",

"4а) в обозначениях строк и столбцов переменная, вводимая в базис и переменная, выводимая из него, меняются местами; \n4б) на месте ключевого элемента записываем обратное ему число;\n4в) ключевую строку (за исключением ключевого элемента) делим на ключевой элемент; полученную строку вписываем на место ключевой;",

"4г) ключевой столбец (за исключением ключевого элемента) делим на ключевой элемент с противоположным знаком; полученный столбец вписываем на место ключевого;\n4д) все остальные элементы таблицы, включая строку оценок и столбец свободных членов, пересчитываем по так называемому «правилу прямоугольника»: на основе пересчитываемой клетки и клетки с ключевым элементом мысленно составляем прямоугольник, далее перемножаем элементы, стоящие в двух оставшихся его вершинах, полученное произведение делим на ключевой элемент и вычитаем из пересчитываемого элемента.",

"5. Новая симплекс-таблица отвечает новому допустимому базисному решению. Проверяем новое решение на оптимальность, если решение не оптимально, то повторяем алгоритм."};

string[] arrayLesson3 = {

"",

"",

"Неизвестные, которые выражаются через остальные неизвестные, называются базисными, а весь набор этих неизвестных – базисом.\nОстальные неизвестные называются свободными. Количество базисных переменных должно равняться количеству уравнений в системе. ",

"Далее необходимо преобразовать целевую функцию, исключив из нее базисные переменные.\nДля исключения базисных переменных из целевой функции нужно умножить первое уравнение системы ограничений на c1, второе на c2, и т.д., сложить полученные произведения и вычесть целевую функцию. ",

"",

"Приравняем свободные переменные к нулю и найдем значения базисных переменных. Получим одно из базисных решений системы ограничений.\nБазисное решение называется допустимым базисным решением или опорным решением, если значения базисных переменных в нем неотрицательны.",

"Среди оптимальных планов задачи линейного программирования в канонической форме обязательно есть опорное решение ее системы ограничений.\nТаким образом симплекс-метод представ-ляет собой процедуру направленного перебора опорных решений. "};

string[] arrayLesson4 = { "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""};

Image formula1 = Properties.Resources.formula1;

Image formula2 = Properties.Resources.formula2;

Image formula3 = Properties.Resources.formula3;

Image src1 = Properties.Resources.Scr1;

Image src2 = Properties.Resources.Scr2;

Image src3 = Properties.Resources.Scr3;

Image src4 = Properties.Resources.Scr4;

Image src5 = Properties.Resources.Scr5;

Image src6 = Properties.Resources.Scr6;

Image src7 = Properties.Resources.Scr7;

Image src8 = Properties.Resources.Scr8;

Image src9 = Properties.Resources.Scr9;

Image src10 = Properties.Resources.Scr10;

public LessonsUser()

{

InitializeComponent();

closeElement(true);

}

private void closeElement(bool active)

{

lesson1.Visible = active; lesson2.Visible = active; lesson3.Visible = active;

label1.Visible = active; label3.Visible = active;

label4.Visible = active;

bunifuCircleProgressbar1.Visible = active; bunifuCircleProgressbar2.Visible = active;

bunifuCircleProgressbar3.Visible = active;

bunifuCircleProgressbar5.Visible = active; bunifuCircleProgressbar6.Visible = active;

bunifuSeparator1.Visible = active;

if(active == false)

{

label5.Visible = true;

label6.Visible = true;

labelText.Visible = true;

labelBack.Visible = true;

labelText.Visible = true;

}

else

{

label5.Visible = false;

label6.Visible = false;

labelText.Visible = false;

labelBack.Visible = false;

//pictureBox1.Visible = false;

nameLesson.Text = "";

}

}

public void nextPage(int stranica, string[] array, int yrok, bool imgL = false)

{

double sum = 0;

stranica++;

if (imgL == false)

{

pictureBox1.Visible = false;

if (yrok == 1)

{

if (stranica == 3)

{

pictureBox1.Image = formula1;

pictureBox1.Visible = true;

}

else

{

pictureBox1.Visible = false;

}

}

if (yrok == 3)

{

if (stranica == 1)

{

pictureBox1.Image = formula3;

pictureBox1.Visible = true;

}

else

{

pictureBox1.Visible = false;

}

if (stranica == 4)

{

pictureBox1.Image = formula2;

pictureBox1.Visible = true;

}

else if (stranica != 1 && stranica != 5) { pictureBox1.Visible = false; }

}

if (stranica < array.Length)

{

labelText.Text = array[stranica];

}

else

{

stranica--;

}

sum = (stranica + 1) \* 100 / array.Length;

if (yrok == 1)

{

count1 = stranica;

if (sum > Properties.Settings.Default.lesson1App)

Properties.Settings.Default.lesson1App = Convert.ToInt32(sum);

}

if (yrok == 2)

{

count2 = stranica;

if (sum > Properties.Settings.Default.lesson2App)

Properties.Settings.Default.lesson2App = Convert.ToInt32(sum);

}

if (yrok == 3)

{

count3 = stranica;

if (sum > Properties.Settings.Default.lesson3App)

Properties.Settings.Default.lesson3App = Convert.ToInt32(sum);

}

stranica = 0;

}

else

{

if (yrok == 4)

{

pictureBox1.Visible = true;

labelText.Visible = false;

if (stranica+1 < 12)

{

countImg1 = stranica;

}

else

{

return;

}

switch (stranica)

{

case 0:

pictureBox1.Image = src1;

break;

case 1:

pictureBox1.Image = src1;

break;

case 2:

pictureBox1.Image = src2;

break;

case 3:

pictureBox1.Image = src3;

break;

case 4:

pictureBox1.Image = src4;

break;

case 5:

pictureBox1.Image = src5;

break;

case 6:

pictureBox1.Image = src6;

break;

case 7:

pictureBox1.Image = src7;

break;

case 8:

pictureBox1.Image = src8;

break;

case 9:

pictureBox1.Image = src9;

break;

case 10:

pictureBox1.Image = src10;

break;

}

countImg1 = stranica;

if (stranica \* 100 / 10 > Properties.Settings.Default.lesson4App)

Properties.Settings.Default.lesson4App = Convert.ToInt32(countImg1 \* 100 / 10);

}

}

Properties.Settings.Default.Save();

}

public void backPage(int stranica, string[] array, int yrok)

{

if (yrok == 1)

{

if (stranica == 3)

{

pictureBox1.Visible = true;

}

else{ pictureBox1.Visible = false;}

}

if (yrok == 3)

{

if (stranica == 1)

{

pictureBox1.Image = formula3;

pictureBox1.Visible = true;

}

else

{

pictureBox1.Visible = false;

}

if (stranica == 4)

{

pictureBox1.Image = formula2;

pictureBox1.Visible = true;

}

else if (stranica != 1 && stranica != 5) { pictureBox1.Visible = false; }

}

if (yrok == 4)

{

pictureBox1.Visible = true;

labelText.Visible = false;

if (stranica - 1 > -1)

{

countImg1 = stranica;

}

else

{

return;

}

switch (stranica)

{

case 0:

pictureBox1.Image = src1;

break;

case 1:

pictureBox1.Image = src1;

break;

case 2:

pictureBox1.Image = src2;

break;

case 3:

pictureBox1.Image = src3;

break;

case 4:

pictureBox1.Image = src4;

break;

case 5:

pictureBox1.Image = src5;

break;

case 6:

pictureBox1.Image = src6;

break;

case 7:

pictureBox1.Image = src7;

break;

case 8:

pictureBox1.Image = src8;

break;

case 9:

pictureBox1.Image = src9;

break;

case 10:

pictureBox1.Image = src10;

break;

}

countImg1 = stranica;

}

if (stranica > 0)

{

labelText.Text = array[stranica];

stranica--;

}

if (yrok == 1)

count1 = stranica;

if (yrok == 2)

count2 = stranica;

if (yrok == 3)

count3 = stranica;

if (yrok == 4)

countImg1 = stranica;

}

bool les1 = false;

bool les2 = false;

bool les3 = false;

bool les4 = false;

private void bunifuCircleProgressbar1\_Click(object sender, EventArgs e)

{

closeElement(false);

nameLesson.Text = "Урок №1. Сущность метода";

les1 = true;

}

private void lesson2\_Click(object sender, EventArgs e)

{

closeElement(false);

nameLesson.Text = "Урок №3. Алгоритм решения";

les2 = true;

}

private void lesson3\_Click(object sender, EventArgs e)

{

closeElement(false);

nameLesson.Text = "Урок №2. Приведение задачи \nк допустимому виду";

les3 = true;

}

private void label4\_Click(object sender, EventArgs e)

{

nameLesson.Text = "Задание №1. Пример на практике";

closeElement(false);

les4 = true;

}

private void label5\_Click(object sender, EventArgs e)

{

if (les1 == true)

nextPage(count1, arrayLesson1, 1);

if (les2 == true)

nextPage(count2, arrayLesson2, 2);

if (les3 == true)

nextPage(count3, arrayLesson3, 3);

if (les4 == true)

nextPage(countImg1, arrayLesson1, 4,true);

}

private void label6\_Click(object sender, EventArgs e)

{

if (les1 == true)

backPage(count1, arrayLesson1, 1);

if (les2 == true)

backPage(count2, arrayLesson2, 2);

if (les3 == true)

backPage(count3, arrayLesson3, 3);

if (les4 == true)

backPage(countImg1, arrayLesson1, 4);

}

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

{

les1 = false;

les2 = false;

les3 = false;

les4 = false;

bunifuCircleProgressbar1.Value = Properties.Settings.Default.lesson1App;

bunifuCircleProgressbar2.Value = Properties.Settings.Default.lesson2App;

bunifuCircleProgressbar3.Value = Properties.Settings.Default.lesson3App;

bunifuCircleProgressbar6.Value = Properties.Settings.Default.lesson4App;

closeElement(true);

pictureBox1.Visible = false;

labelText.Text = "Нажмите далее";

}

private void LessonsPanel\_Layout(object sender, LayoutEventArgs e)

{

bunifuCircleProgressbar1.Value = Properties.Settings.Default.lesson1App;

bunifuCircleProgressbar2.Value = Properties.Settings.Default.lesson2App;

bunifuCircleProgressbar3.Value = Properties.Settings.Default.lesson3App;

bunifuCircleProgressbar6.Value = Properties.Settings.Default.lesson4App;

}

}

}

Программный модуль StatControlPanel.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace SimplexMethod

{

public partial class StatControlPanel : UserControl

{

public StatControlPanel()

{

InitializeComponent();

}

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

{

}

private void StatControlPanel\_Layout(object sender, LayoutEventArgs e)

{

double sum = (Properties.Settings.Default.lesson1App + Properties.Settings.Default.lesson2App +

Properties.Settings.Default.lesson3App) / 3;

bunifuCircleProgressbar1.Value = Convert.ToInt32(sum);

sum = (Properties.Settings.Default.lesson3App + Properties.Settings.Default.lesson4App) / 2;

bunifuCircleProgressbar2.Value = Convert.ToInt32(0);

}

}

}

Программный модуль UserAbout.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace SimplexMethod

{

public partial class UserAbout : UserControl

{

public UserAbout()

{

InitializeComponent();

}

}

}

Программный модуль UserCalculator.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace SimplexMethod

{

public partial class UserCalculator : UserControl

{

public UserCalculator()

{

InitializeComponent();

}

List<ComboBox> \_boxes = new List<ComboBox>();

List<TextBox> \_func = new List<TextBox>();

TextBox[,] tb;

TextBox[] restb;

Random rdn = new Random();

/\*

\* Массив для хранения введенных чисел

\*/

double[,] arrayMatr;

/\*

\* Массив для дублирования исходного обработанного массива

\*/

double[,] saveMatr;

/\*

\* Массив для хранения значений введенных знаков

\*/

int[] arrayZnak;

/\*

\* Массив для хранения исходного обработанного массива (значения после знака '=')

\*/

double[] myZnach;

/\*

\* Массив для хранения общей матрицы

\*/

double[,] resArray;

double[] save2;

int a, b, addElement;

double[] elementFunk;

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

{

try

{

a = Convert.ToInt32(textBox1.Text);

b = Convert.ToInt32(textBox2.Text);

if(a>0 && b>0)

{

if(a>=2 && b>=2)

{

if(a<=5 && b<=5)

{

/\*

\* Если все условия выполнены

\*/

pasteCombo(a, b);

unlockFunc(a);

/\*

\* Блокируем необходимые кнопки управления

\*/

bunifuFlatButton2.Enabled = false;

bunifuFlatButton1.Enabled = true;

bunifuFlatButton3.Enabled = true;

textBox1.Enabled = false;

textBox2.Enabled = false;

}

else

{

MessageBox.Show("Кол-во значений Х или ограничений не должно превышать 5!","Ошибка");

}

}

else

{

MessageBox.Show("Кол-во значений Х или ограничений должно быть не меньше 2!", "Ошибка");

}

}

else

{

MessageBox.Show("Введите числа больше 0!", "Ошибка");

}

}

catch

{

MessageBox.Show("Введите корректные данные!", "Ошибка");

}

}

public void pasteCombo(int a, int b)

{

tb = new TextBox[a, b];

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

{

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

{

tb[i, j] = new TextBox();

tb[i, j].Parent = this;

tb[i, j].Bounds = new Rectangle((i + 5) \* 60, j\*30, 30, 30);

tb[i, j].Font = new System.Drawing.Font(FontFamily.Families[20], this.Height / 26);

}

}

/\*

\* Создаем объект для создания определенного кол-ва ComboBox после всей матрицы

\*/

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

{

var combo = new ComboBox() { Location = new Point((a + 5) \* 60, j\*30), Width = 40, Height = 40};

combo.Font = new System.Drawing.Font(FontFamily.Families[20], this.Height / 30);

combo.Items.Add("<=");

combo.Items.Add("=>");

combo.Items.Add("=");

combo.DropDownStyle = ComboBoxStyle.DropDownList;

this.Controls.Add(combo);

\_boxes.Add(combo);

}

/\*

\* Создаем объект для создания определенного кол-ва TextBox после знака равенства

\*/

restb = new TextBox[a];

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

{

restb[i] = new TextBox();

restb[i].Parent = this;

restb[i].Bounds = new Rectangle((a + 6) \* 60, i\*30, 30, 30);

restb[i].Font = new System.Drawing.Font(FontFamily.Families[20], this.Height / 26);

}

}

public void unlockFunc(int a)

{

switch (a)

{

case 2: textBox3.Enabled = true; textBox4.Enabled = true; textBox5.Text = "0"; textBox6.Text = "0"; textBox7.Text = "0"; break;

case 3: textBox3.Enabled = true; textBox4.Enabled = true; textBox5.Enabled = true; textBox6.Text = "0"; textBox7.Text = "0"; break;

case 4: textBox3.Enabled = true; textBox4.Enabled = true; textBox5.Enabled = true; textBox6.Enabled = true; textBox7.Text = "0"; break;

case 5: textBox3.Enabled = true; textBox4.Enabled = true; textBox5.Enabled = true; textBox6.Enabled = true; textBox7.Enabled = true; break;

}

}

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

{

}

/\*

\* Конка "Удалить все"

\*/

private void bunifuFlatButton1\_Click(object sender, EventArgs e)

{

bunifuFlatButton1.Enabled = false;

bunifuFlatButton2.Enabled = true;

bunifuFlatButton3.Enabled = false;

textBox3.Text = null;

textBox3.Enabled = false;

textBox4.Text = null;

textBox4.Enabled = false;

textBox5.Text = null;

textBox5.Enabled = false;

textBox6.Text = null;

textBox6.Enabled = false;

textBox7.Text = null;

textBox7.Enabled = false;

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

{

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

{

Controls.Remove(tb[i, j]);

}

}

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

{

Controls.Remove(restb[i]);

}

tb = null;

restb = null;

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

{

var box = \_boxes[j];

Controls.Remove(box);

}

a = 0;

b = 0;

textBox1.Enabled = true;

textBox2.Enabled = true;

\_boxes.Clear();

}

/\*

\* Нажатие кнопки "Начать решать"

\*/

private void bunifuFlatButton3\_Click(object sender, EventArgs e)

{

int i, j;

/\*

\* Проверка на пустоту TextBox

\*/

foreach (var element in Controls)

{

if (element is TextBox)

{

if (((TextBox)element).Text == "")

{

MessageBox.Show("Заполни поля!");

return;

}

}

}

/\*

\* Проверка на пустоту ComboBox

\*/

foreach(var element in Controls)

{

if(element is ComboBox)

{

if(((ComboBox)element).Text == String.Empty)

{

MessageBox.Show("Выберете все ограничения!");

return;

}

}

}

/\*

\* Если введенные значения прошли проверку

\* Блокируем все поля

\*

\*/

foreach(var element in Controls)

{

if (element is TextBox)

((TextBox)element).Enabled = false;

if (element is ComboBox)

((ComboBox)element).Enabled = false;

}

//arrayMatr = new double[a, b];

setElementCombo();

/\*

\* Алгоритм решения

\*/

a = Convert.ToInt32(textBox1.Text);//Строка

b = Convert.ToInt32(textBox2.Text);//Столбец

int updM = b + addElement + 1; //Увеличиваем кол-во столбцов до требуемого значения

resArray = new double[a, updM];

saveMatr = new double[a, updM];

save2 = new double[updM];

/\*

\* Переменная addElement - будет хранить кол-во ComboBox;

\*/

int n, m, PositionX = 0, PositionY = 0,complete=0;

double num1, num2, max = -999, Glavn = 0, x, Minimum = 999;

n = a;

m = b;

/\*

\* Зададим массив в котором будем хранить минимальные значения каждого столбца

\*/

double[] min;

min = new double[b+addElement];

/\*

\* Зададим 2 массива в которых будем хранить позиции каждого минимального элемента

\*/

int[] posx;

int[] posy;

posx = new int[b+addElement];

posy = new int[b+addElement];

int update =0;

/\*

\* Начало перебора матрицы

\*/

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

{

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

{

resArray[i, j] = Convert.ToDouble(tb[j, i].Text);

}

}

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

{

for ( j = b; j < updM - 1; j++)

{

resArray[i, j] = 0;

}

}

int c = b;

bool complete1 = false;

/\*

\* Установим значения в фиктивной матрице в соответствии с заданными знаками

\*/

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

{

complete1 = false;

if (arrayZnak[i] == 1 && complete1 == false)

{

resArray[i, c] = 1;

//label8.Text += "\n Если знак <= позиции i " + i + " j " + c;

c++;

complete1 = true;

}

if (arrayZnak[i] == -1 && complete1 == false)

{

//label8.Text += "\n Если знак >= позиции i " + i + " j " + c;

resArray[i, c] = -1;

c++;

complete1 = true;

}

if (arrayZnak[i] == 0 && complete1 == false)

{

//label8.Text += "\n Если знак = позиция i " + i + " j " + c;

complete1 = true;

}

}

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

{

for ( j = updM - 1; j < updM; j++)

{

resArray[i, j] = myZnach[i];

}

}

label1.Text += "\n\n ";

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

{

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

{

saveMatr[i, j] = resArray[i, j];

}

label8.Text += "\n";

}

for (int k = 1; k < 5; k++)

{

for (i = 0; i < b + addElement; i++)

{

min[i] = 0;

}

label8.Text += " \n Иттерация №" + k;

/\*

\* Делим последний элемент на каждый элемент в столбце и ищем минимальный элемент в столбце

\*/

if (k > 0)

{

update = b + addElement;

}

else

{

update = b;

}

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

{

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

{

num1 = saveMatr[j, updM - 1];

num2 = resArray[j, i];

if (num1 != 0 && num2 != 0)

{

x = num1 / num2;

x = x \* elementFunk[i];

if (k > 1)

{

//debugLabel.Text += "\n Что " + num1 + " делим на " + Math.Round(num2, 2) + " умножаю на " + Math.Round(elementFunk[i], 2) + " = " + x;

}

if (x < Minimum)

{

Minimum = x;

min[i] = x;

posx[i] = i;

posy[i] = j;

}

}

else

break;

}

Minimum = 999;

}

//debugLabel.Text += "\n Минимальные";

//for (i = 0; i < b; i++)

//{

// debugLabel.Text += min[i] + " ";

//}

/\*

\* Поиск максимального элемента среди минимальных каждого столбца

\*/

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

{

if (min[i] > max)

max = min[i];

}

/\*

\* Записываем ведущий элемент и его позицию

\*/

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

{

if (max == min[i])

{

PositionX = posy[i];

PositionY = posx[i];

Glavn = resArray[PositionX, PositionY];

label8.Text += "\n Ведущий элемент: " + Glavn + "\n Столбец X " + PositionX + "\n Cтрока Y " + PositionY;

break;

}

}

label8.Text += "\n Преобразование матрицы: \n ";

/\*

\* Начало обработки всей матрицы

\*/

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

{

for (j = 0; j < m + addElement + 1; j++)

{

if (PositionX != i && PositionY != j)

{

saveMatr[i, j] = (resArray[i, j] \* Glavn - resArray[PositionX, j] \* resArray[i, PositionY]) / Glavn;

//label3.Text += "\n Глав 1 [" + Glavn + "] Глав 2 [" + resArray[i, j] + "] Побоч 1 [" + resArray[PositionX, j] + "] Побоч 2 [" + resArray[i, PositionY] + "] =" + Math.Round(((resArray[i, j] \* Glavn - resArray[PositionX, j] \* resArray[i, PositionY]) / Glavn), 2);

}

if (PositionY == j && resArray[i, j] != resArray[PositionX, PositionY])

{

saveMatr[i, j] = 0;

// elementFunk[i] = 0;

}

if (PositionX == i)

{

//label3.Text += "\n Что деллю" + resArray[i, j] + " на что делю " + Glavn;

saveMatr[i, j] = resArray[i, j] / Glavn;

}

}

}

for (i = 0; i < b + addElement; i++)

{

// debugLabel.Text += "\n Глав 1 [" + Glavn + "] Глав 2 [" + save2[i] + "] Побоч 1 [" + resArray[PositionX, i] + "] Побоч 2 [" + save2[PositionY] + "] = " + Math.Round(((save2[i] \* Glavn) - (resArray[PositionX, i] \* save2[PositionY])) / Glavn, 2);

elementFunk[i] = ((save2[i] \* Glavn) - (resArray[PositionX, i] \* save2[PositionY])) / Glavn;

}

label8.Text += "\n";

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

{

for (j = 0; j < m + addElement + 1; j++)

{

resArray[i, j] = saveMatr[i, j];

if (resArray[i, j] > 0)

{

label8.Text += " " + Math.Round(resArray[i, j], 2);

}

else

{

label8.Text += " " + Math.Round(resArray[i, j], 2);

}

}

label8.Text += "\n";

}

for (i = 0; i < addElement + b; i++)

{

if (elementFunk[i] > 0)

{

complete++;

}

label8.Text += " " + Math.Round(elementFunk[i], 2);

}

label8.Text += "\n----------------------------------\n";

if (complete == 0)

{

break;

}

else

{

complete = 0;

max = -999; Minimum = 999;

//PositionX = 0; PositionY = 0;Glavn = 0;

}

}

label8.Text += "\n-----------Ответ-----------\n";

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

{

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

{

if (resArray[i, j] > 0)

{

label8.Text += " " + Math.Round(resArray[i, j], 2);

}

else

{

label8.Text += " " + Math.Round(resArray[i, j], 2);

}

}

label8.Text += "\n";

}

for (int h = 0; h < addElement + b; h++)

{

label8.Text += " " + Math.Round(elementFunk[h], 2);

}

}

public void processingMatr()

{

int a = Convert.ToInt32(textBox1.Text);//Строка

int b = Convert.ToInt32(textBox2.Text);//Столбец

int updM = b + addElement + 1; //Увеличиваем кол-во столбцов до требуемого значения

/\*

\* Создаем общий массив который включит в себя

\* 1) Обработанную матрицу

\* 2) Фиктивную матрицу с заданнами значениями неравенства

\* 3) Значения после неравенства

\*/

resArray = new double[a, updM];

/\*

\* Дублирующая матрица 1

\*/

saveMatr = new double[a, updM];

/\*

\* Дублирующая матрица 2

\*/

save2 = new double[updM];

/\*

\* Запишем в общую матрицу значения матрицы заданных элементов

\*/

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

{

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

{

resArray[i, j] = arrayMatr[i, j];

}

}

/\*

\* Обнулим все значения в фиктивной матрице

\*/

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

{

for (int j = b; j < updM - 1; j++)

{

resArray[i, j] = 0;

}

}

int k = b;

bool complete = false;

/\*

\* Установим значения в фиктивной матрице в соответствии с заданными знаками

\*/

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

{

complete = false;

if (arrayZnak[i] == 1 && complete == false)

{

resArray[i, k] = 1;

label8.Text += "\n Если знак <= позиции i " + i + " j " + k;

k++;

complete = true;

}

if (arrayZnak[i] == -1 && complete == false)

{

label8.Text += "\n Если знак >= позиции i " + i + " j " + k;

resArray[i, k] = -1;

k++;

complete = true;

}

if (arrayZnak[i] == 0 && complete == false)

{

label8.Text += "\n Если знак = позиция i " + i + " j " + k;

complete = true;

}

}

label8.Text += "\n";

/\*

\* Запишем в конец общей матрицы значения после неравенства

\*/

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

{

for (int j = updM - 1; j < updM; j++)

{

resArray[i, j] = myZnach[i];

}

}

/\*

\* Выведем полученный результат

\*/

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

{

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

{

saveMatr[i, j] = resArray[i, j];

if (resArray[i, j] > 0)

{

label8.Text += " " + resArray[i, j];

}

else

{

label8.Text += " " + resArray[i, j];

}

}

label8.Text += "\n";

}

for (int i = 0; i < a + b; i++)

{

label8.Text += " " + elementFunk[i];

}

}

public void setElementMatr(TextBox[,] td, int a = 0, int b = 0)

{

int n, m;

n = Convert.ToInt32(textBox1.Text);

m = Convert.ToInt32(textBox2.Text);

arrayMatr = new double[n, m];

}

public void setElementCombo()

{

/\*

\* Массив хранит значения с ComboBox

\*

\* <= - это 1

\* => - это -1

\* = - это 0

\*/

a = Convert.ToInt32(textBox1.Text);//Строка

b = Convert.ToInt32(textBox2.Text);//Столбец

arrayZnak = new int[a];

int i = 0;

addElement = 0;

foreach (var element in \_boxes)

{

if (element is ComboBox)

{

if (((ComboBox)element).Text == "<=")

{

arrayZnak[i] = 1;

addElement++;

i++;

}

if (((ComboBox)element).Text == "=>")

{

arrayZnak[i] = -1;

addElement++;

i++;

}

if (((ComboBox)element).Text == "=")

{

arrayZnak[i] = 0;

i++;

}

}

}

myZnach = new double[a];

elementFunk = new double[b + a+1];

try

{

for (int k = 0; k < a; k++)

{

myZnach[k] = Convert.ToDouble(restb[k].Text);

}

}

catch { MessageBox.Show("Заполни все поля"); }

elementFunk[0] = Convert.ToDouble(textBox3.Text);

elementFunk[1] = Convert.ToDouble(textBox4.Text);

elementFunk[2] = Convert.ToDouble(textBox5.Text);

elementFunk[3] = Convert.ToDouble(textBox6.Text);

elementFunk[4] = Convert.ToDouble(textBox7.Text);

for (int k = b; k < a; k++)

{

elementFunk[k] = 0;

}

}

}

}

Приложение Б

Титульный лист презентации к пояснительной записке.

