Министерство образования Республики Беларусь

Оршанский колледж ВГУ имени П.М. Машерова

Отчет

По модулю №7

« Графический интерфейс»

по учебной практике по программированию

Выполнил учащийся Войтеховская К.В.

группа 3ПОИС23 29.09.2025 г.

Проверил Алейников М.А.

30.09.2025 г.

Орша, 2025

**Задания.**

**Задание 1.** Выберите проект, который вы разработали в рамках предмета Основы алгоритмизации и программирования

**Задание 2.** Сформулируйте требования к проекту и создайте план разработки.

**Задание 3.** Разработайте проект, следуя плану и используя полученные знания и навыки.

**Задание 4.** Проведите тестирование и отладку проекта.

**Задание 5**. Подготовьте документацию и презентацию для защиты проекта перед преподавателями.

**Ход работы.**

**Выполнение практических заданий.**

**Выполнение задания 1.** В рамках предмета «Основы алгоритмизации и программирования» был разработан проект реализующий метод наискорейшего спуска. Он был реализован на языке программирования Delphi.

**Выполнение задания 2.** Проект должен быть реализован на языке программирования C#. Также программа реализации метода наискорейшего спуска должна обеспечивать выполнение функций:

1)Управление интерфейсом.

2)Ввод данных.

Пользователь вводит функцию, минимум которой требуется найти. Программа поддерживает стандартные математические операции (сложение, вычитание, умножение, деление, возведение в степень) и функции (синус, косинус, экспонента, натуральный логарифм, квадратный корень). Дополнительно указываются начальная точка, точность вычислений и максимальное число итераций. Встроенные функции проверяют корректность введённых данных.

3)Вычисление минимума функции.

Программа «Метод наискорейшего спуска» использует метод наискорейшего спуска для итеративного приближения к минимуму. На каждом шаге вычисляется градиент функции с помощью численного дифференцирования, определяется оптимальный шаг и обновляется текущее значение аргумента. Процесс продолжается до достижения заданной точности или исчерпания лимита итераций.

4)Отображение результатов.

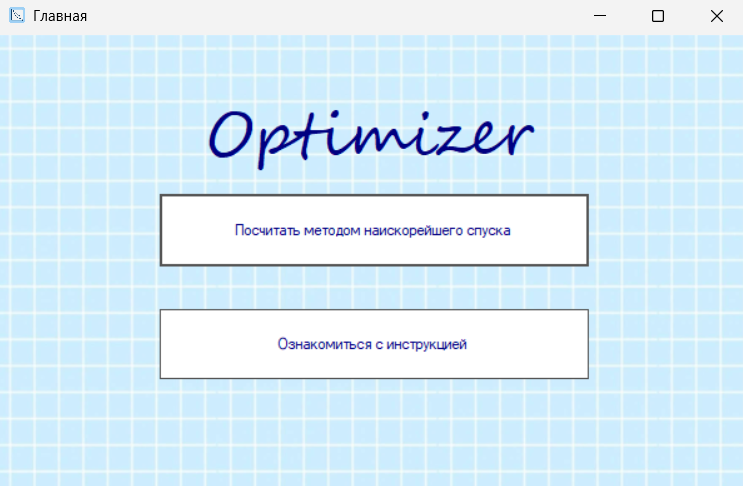
После завершения вычислений программа выводит найденную точку минимума, значение функции в этой точке, количество выполненных итераций, а также историю изменения градиента и шага оптимизации. Результаты отображаются в удобочитаемом формате с возможностью прокрутки.

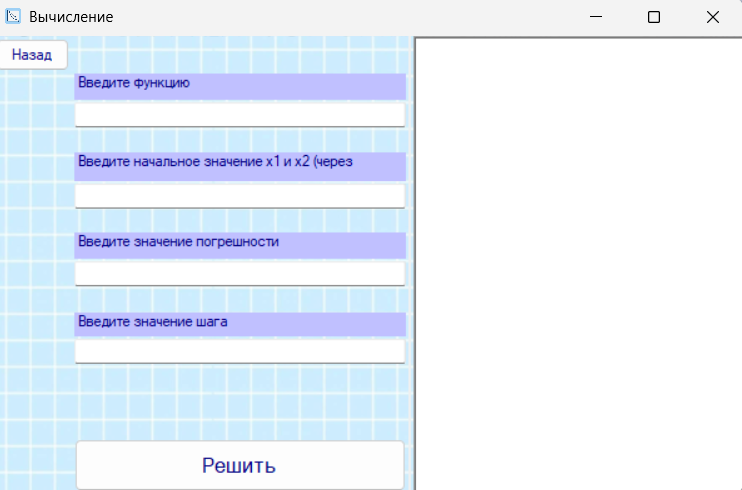
1. Работа с инструкцией.

Поскольку изучение предметной области и техническое задание было выполнено в ходе курсового проектирования, план разработки будет выглядеть так:

1. проектирование интерфейса программы;
2. обеспечение навигации между формами;
3. реализация математического метода;
4. обеспечение отказоустойчивости;
5. тестирование и отладка;
6. подготовка справочной информации;
7. подготовка документации.

**Выполнение задания 3.** Программа, реализующая метод наискорейшего спуска получила название «Optimizer». В приложении будет 3 формы, которые имеют гибкий интерфейс. Они представлены на рисунке 1. С помощью кнопок была обеспечена навигация между формами.





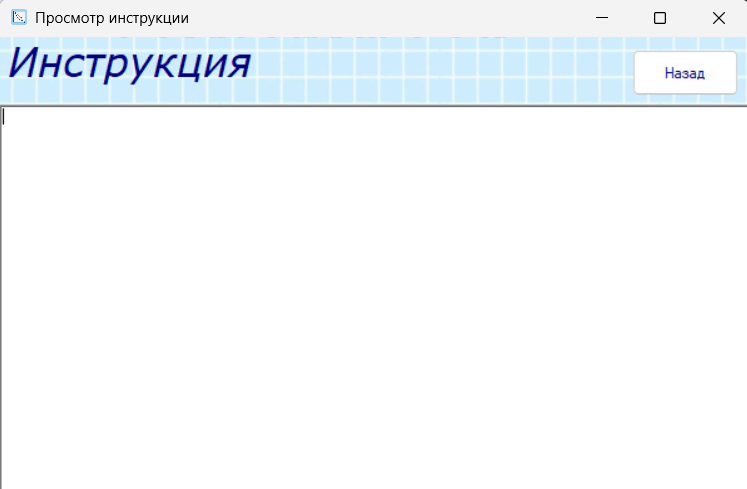


Рисунок 1 – Вид форм

После была разработана программа на языке программирования C# на основе проекта, сделанного в рамках предмета «Основы алгоритмизации и программирования» с исправлениями выполненных в прошлом ошибок. В листинге 1 представлен код первой формы, осуществляющая роль главной страницы программы.

Листинг 1. Главная форма

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 WindowsFormsApp1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

Form3 form3 = new Form3();

form3.Show();

this.Hide();

}

private void button2\_Click(object sender, EventArgs e)

{

Form2 form2 = new Form2();

form2.Show();

this.Hide();

}

}

}

В листинге 2 представлен код второй формы, предназначенной для просмотра инструкции. Текст инструкции загружается в нее из файла «Справка.txt».

Листинг 2. Код формы просмотра инструкции

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace WindowsFormsApp1

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

private void Form2\_Load(object sender, EventArgs e)

{

// Загрузка текста из файла при загрузке формы

LoadHelpText();

}

private void LoadHelpText()

{

try

{

string filePath = Path.Combine(Application.StartupPath, "Справка.txt");

if (File.Exists(filePath))

{

string helpText = File.ReadAllText(filePath, Encoding.UTF8);

richTextBox1.Text = helpText;

}

else

{

richTextBox1.Text = "Файл справки не найден: " + filePath;

}

}

catch (Exception ex)

{

richTextBox1.Text = "Ошибка при загрузке файла справки: " + ex.Message;

}

}

private void button1\_Click(object sender, EventArgs e)

{

Form1 form1 = new Form1();

form1.Show();

this.Hide();

}

}

}

В листинге 3 представлен код программы формы 3, на которой происходит ввод данных для расчёта, реализация метода наискорейшего спуска и вывод результатов.

Листинг 3. Форма ввода данных и вычисления

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data;

namespace WindowsFormsApp1

{

public partial class Form3 : Form

{

public Form3()

{

InitializeComponent();

InitializeScrollBar();

}

private void InitializeScrollBar()

{

// Настройка вертикального скроллбара

vScrollBar1.Minimum = 0;

vScrollBar1.Maximum = 100;

vScrollBar1.Value = 0;

vScrollBar1.LargeChange = 10;

vScrollBar1.SmallChange = 1;

vScrollBar1.Visible = false; // Изначально скрыт

vScrollBar1.Scroll += VScrollBar1\_Scroll;

}

private void VScrollBar1\_Scroll(object sender, ScrollEventArgs e)

{

// Прокрутка RichTextBox при изменении положения скроллбара

int scrollValue = vScrollBar1.Value;

richTextBox1.SelectionStart = 0;

richTextBox1.ScrollToCaret();

// Альтернативный метод прокрутки

int lineToScroll = (int)((double)scrollValue / vScrollBar1.Maximum \* GetLineCount());

if (lineToScroll < richTextBox1.Lines.Length)

{

richTextBox1.SelectionStart = richTextBox1.GetFirstCharIndexFromLine(lineToScroll);

richTextBox1.ScrollToCaret();

}

}

private int GetLineCount()

{

return richTextBox1.Text.Split('\n').Length;

}

private void UpdateScrollBar()

{

// Обновление скроллбар в зависимости от содержимого RichTextBox

int lineCount = GetLineCount();

int visibleLines = richTextBox1.Height / richTextBox1.Font.Height;

if (lineCount > visibleLines)

{

vScrollBar1.Visible = true;

vScrollBar1.Maximum = Math.Max(0, lineCount - visibleLines);

vScrollBar1.LargeChange = Math.Max(1, visibleLines / 2);

}

else

{

vScrollBar1.Visible = false;

}

}

private void button1\_Click(object sender, EventArgs e)

{

Form1 form1 = new Form1();

form1.Show();

this.Hide();

}

// Парсинг функции и вычисление значения

private double EvaluateFunction(string function, double x1, double x2)

{

try

{

// Заменяем x1 и x2 на числовые значения

string expression = function

.Replace("x1", x1.ToString(System.Globalization.CultureInfo.InvariantCulture))

.Replace("x2", x2.ToString(System.Globalization.CultureInfo.InvariantCulture))

.Replace(",", "."); // Для корректного парсинга

// Используем DataTable для вычисления выражения

DataTable table = new DataTable();

table.Columns.Add("expression", typeof(string), expression);

DataRow row = table.NewRow();

table.Rows.Add(row);

return double.Parse(row["expression"].ToString());

}

catch (Exception ex)

{

throw new Exception($"Ошибка вычисления функции: {ex.Message}");

}

}

// Численное вычисление градиента по x1

private double NumericGradX1(string function, double x1, double x2, double h = 0.0001)

{

double f1 = EvaluateFunction(function, x1 + h, x2);

double f2 = EvaluateFunction(function, x1 - h, x2);

return (f1 - f2) / (2 \* h);

}

// Численное вычисление градиента по x2

private double NumericGradX2(string function, double x1, double x2, double h = 0.0001)

{

double f1 = EvaluateFunction(function, x1, x2 + h);

double f2 = EvaluateFunction(function, x1, x2 - h);

return (f1 - f2) / (2 \* h);

}

private void button2\_Click(object sender, EventArgs e)

{

try

{

// Получаем функцию из первого текстбокса

string function = textBox1.Text.Trim();

if (string.IsNullOrEmpty(function))

{

MessageBox.Show("Введите функцию!", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Получаем начальные значения из второго текстбокса

string[] values = textBox2.Text.Split(new[] { ' ', ',' }, StringSplitOptions.RemoveEmptyEntries);

if (values.Length != 2)

{

MessageBox.Show("Введите два значения x1 и x2 через пробел!", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

double x1 = Convert.ToDouble(values[0]);

double x2 = Convert.ToDouble(values[1]);

double eps = Convert.ToDouble(textBox3.Text);

double step = Convert.ToDouble(textBox4.Text);

int k = 0;

double prevValue, currentValue, x1\_prev, x2\_prev;

// Тестовая проверка функции

double testValue = EvaluateFunction(function, x1, x2);

// Алгоритм градиентного спуска

do

{

x1\_prev = x1;

x2\_prev = x2;

prevValue = EvaluateFunction(function, x1\_prev, x2\_prev);

// Вычисляем градиенты численным методом

double grad1 = NumericGradX1(function, x1\_prev, x2\_prev);

double grad2 = NumericGradX2(function, x1\_prev, x2\_prev);

// Обновляем значения

x1 = x1\_prev - step \* grad1;

x2 = x2\_prev - step \* grad2;

currentValue = EvaluateFunction(function, x1, x2);

k++;

// Защита от бесконечного цикла

if (k > 10000)

{

throw new Exception("Превышено максимальное количество итераций. Возможно, функция не сходится.");

}

}

while (Math.Abs(currentValue - prevValue) > eps);

// Выводим результаты в RichTextBox

string result = $"Функция: {function}\r\n" +

$"Начальные значения: x1={values[0]}, x2={values[1]}\r\n" +

$"Точность: {eps}, Шаг: {step}\r\n" +

$"Результаты:\r\n" +

$"x1 = {x1:F6}\r\n" +

$"x2 = {x2:F6}\r\n" +

$"f(x1,x2) = {currentValue:F6}\r\n" +

$"Количество итераций: {k}\r\n" +

$"Время вычисления: {DateTime.Now:HH:mm:ss}\r\n" +

$"----------------------------------------\r\n\r\n";

// Добавляем результат в начало RichTextBox

richTextBox1.Text = result + richTextBox1.Text;

// Прокручиваем к началу

richTextBox1.SelectionStart = 0;

richTextBox1.ScrollToCaret();

// Обновляем скроллбар

UpdateScrollBar();

}

catch (FormatException)

{

MessageBox.Show("Ошибка ввода! Пожалуйста, введите корректные числовые значения.",

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка: {ex.Message}",

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button3\_Click(object sender, EventArgs e)

{

richTextBox1.Clear();

vScrollBar1.Visible = false;

}

// Обработка изменения размера RichTextBox

private void richTextBox1\_SizeChanged(object sender, EventArgs e)

{

UpdateScrollBar();

}

// Обработка изменения текста в RichTextBox

private void richTextBox1\_TextChanged(object sender, EventArgs e)

{

UpdateScrollBar();

}

}

}

**Выполнение задания 4.** В таблицах 1-4 представлены результаты тестирования приложения.

Таблица 1 – Результат теста 1.

|  |  |
| --- | --- |
| Вид тест | Нажатие кнопок |
| Что тестируем | Функционирование кнопок |
| Входные данные | Нажатие на кнопки |
| Настройка тестового окружения | Тест выполняется на компьютере со следующими характеристиками:  Процессор 12th Gen Intel(R) Core(TM) i3-1215U 1.20 GHz  Оперативная память 8,00 ГБ (доступно: 7,75 ГБ)  Тип системы 64-разрядная операционная система, процессор x64  Оперативная система: Windows 11 Домашняя |
| Шаги теста | Запустить программу и нажать на все кнопки |
| Ожидаемые выходные данные | Отображение форм |
| Ожидаемый результат | Пользователь без ошибок переходит между формами программы |
| Фактический результат | Происходит переход между формами приложения |
| Обнаруженный дефект | Дефекты не обнаружены |
| Возможная причина дефекта |  |
| Предполагаемое решение |  |

Таблица 2 – Результат теста 2.

|  |  |
| --- | --- |
| Вид тест | Загрузка программы |
| Что тестируем | Корректный запуск программы на стороннем компьютере |
| Входные данные | Готовый к запуску файл программы. |
| Настройка тестового окружения | Тест выполняется на компьютере со следующими характеристиками:  Оперативная система: Windows XP |
| Шаги теста | Установить, запустить программу и переключиться на формы. |
| Ожидаемые выходные данные | Отображение форм на экране и открытие файла инструкции. |
| Ожидаемый результат | Программа запущена без ошибок, пользователь может переключаться между формами. |
| Фактический результат | Программа не устанавливается |
| Обнаруженный дефект | Программа не запускается |
| Возможная причина дефекта | Файл для 64-битных систем, а Windows XP поддерживает только 32-битные приложения. |
| Предполагаемое решение | Устанавливать программу на 64-битные системы |

Таблица 3 – Результат теста 3

|  |  |
| --- | --- |
| Вид тест | Ввод неполных данных. |
| Что тестируем | Вывод сообщения об ошибке |
| Входные данные | Отсутствие ввода в поля для данных. |
| Настройка тестового окружения | Тест выполняется на компьютере со следующими характеристиками:  Процессор 12th Gen Intel(R) Core(TM) i3-1215U 1.20 GHz  Оперативная память 8,00 ГБ (доступно: 7,75 ГБ)  Тип системы 64-разрядная операционная система, процессор x64  Оперативная система: Windows 11 Домашняя |
| Шаги теста | Переход на форму заполнения данных и сразу нажатие кнопки «решить» |
| Ожидаемые выходные данные | Сообщение об ошибке с просьбой заполнить поля |
| Ожидаемый результат | Программа обрабатывает отсутствие данных и выводит пользователю сообщение об ошибки |
| Фактический результат | Сообщение выведено |
| Обнаруженный дефект | Дефекты не обнаружены |
| Возможная причина дефекта |  |
| Предполагаемое решение |  |

Таблица 4 – Результат теста 4

|  |  |
| --- | --- |
| Вид тест | Ввод дробных чисел в разном формате |
| Что тестируем | Обработка ввода дробных чисел в различных форматах |
| Входные данные | В поле точности вводится число с точкой: 0.001  В поле точности вводится число с запятой: 0.001  В поле точности вводится число в стандартном виде: 1e-1 |
| Настройка тестового окружения | Тест выполняется на компьютере со следующими характеристиками:  Процессор 12th Gen Intel(R) Core(TM) i3-1215U 1.20 GHz  Оперативная память 8,00 ГБ (доступно: 7,75 ГБ)  Тип системы 64-разрядная операционная система, процессор x64  Оперативная система: Windows 11 Домашняя |
| Шаги теста | Переходится на окно ввода данных. В поля ввода функции и количества итераций вводятся корректные данные. Поля заполняются в соответсивии с выходными данными. |
| Ожидаемые выходные данные | При вводе с точкой выводится ошибка, запятая и стандартный вид обрабатываются. |
| Ожидаемый результат | Вывод сообщения с ошибкой при вводе числа с точкой, в остальных случаях решение |
| Фактический результат | Выводится ошибка при вводе с точкой    Выводится результат вычисления в остальных случаях. |
| Обнаруженный дефект | Дефекты не обнаружены |
| Возможная причина дефекта |  |
| Предполагаемое решение |  |

**Выполнение задания 5.** В ходе выполнения задания был написан справочный файл, который помогает пользователю пользоваться приложением. На рисунке 2 представлено отображение файла справки в приложении.

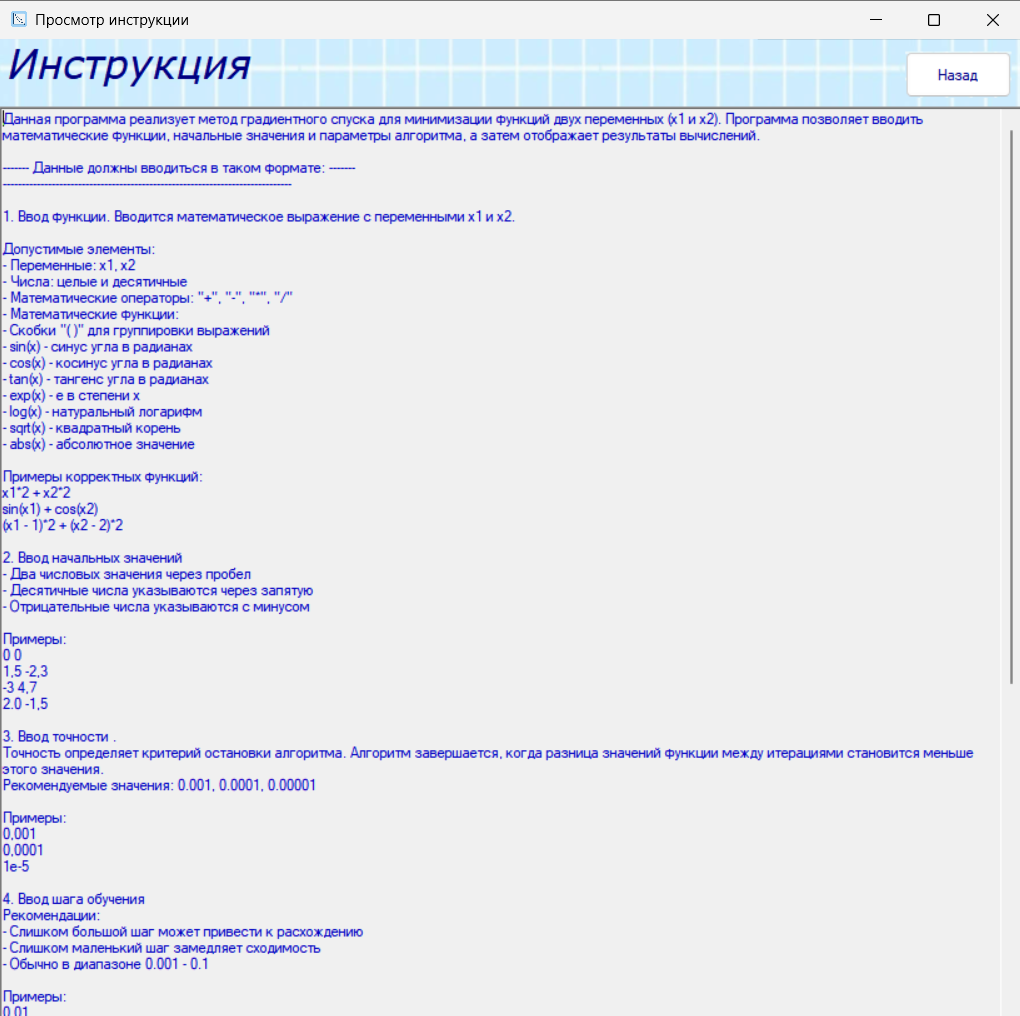


Рисунок 2 – Отображение инструкции в приложении

Была создана презентация, слайды которой представлены на рисунке 3.

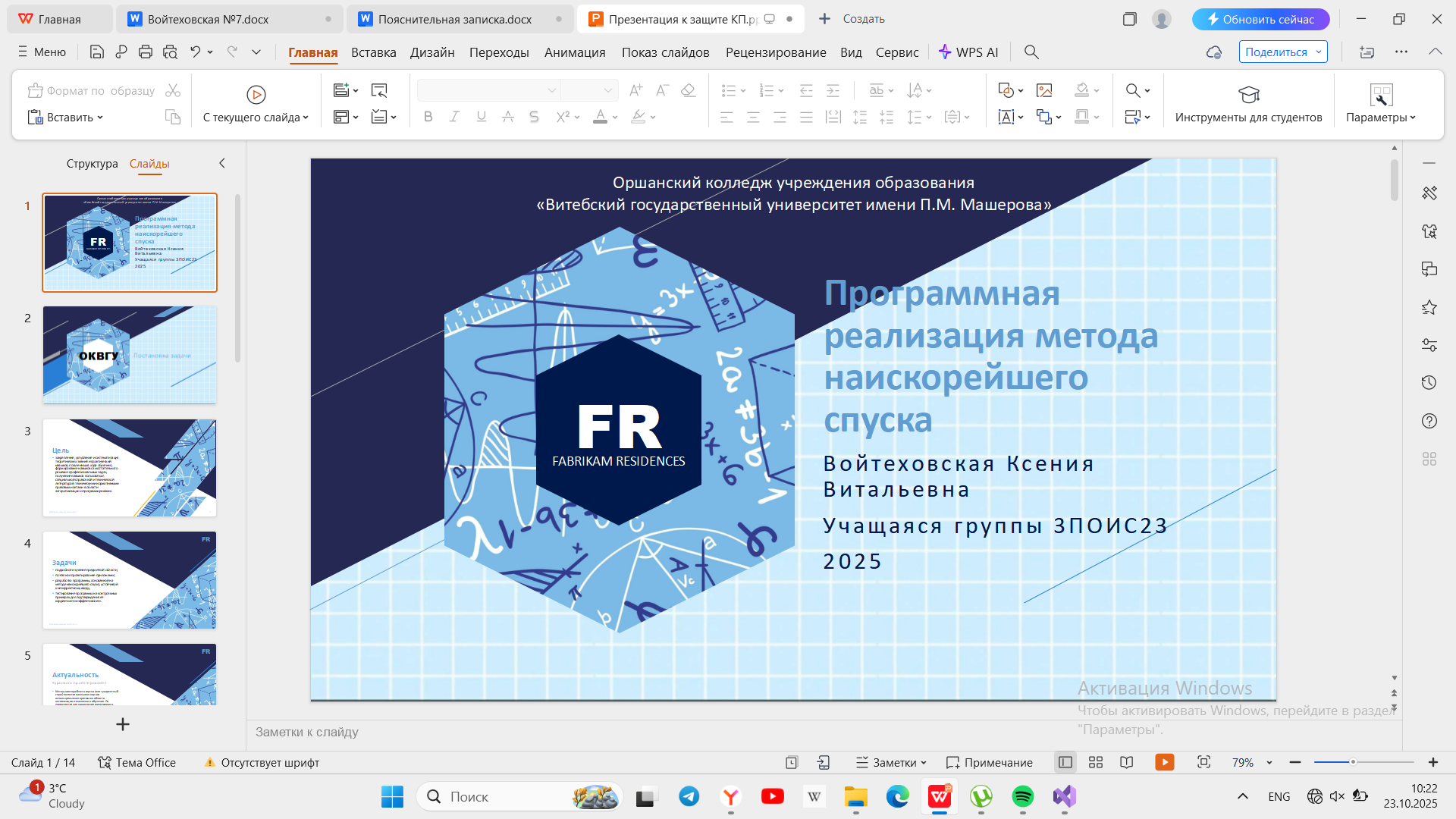


Рисунок 3 – Презентация