

**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(РУТ (МИИТ)**

**ИНСТИТУТ ТРАНСПОРТНОЙ ТЕХНИКИ И СИСТЕМ УПРАВЛЕНИЯ**

**Лабораторная работа №3**

**по дисциплине** «**Методы программирования**»

«Автоматизация компоновки ГПИ. Функции»

**Выполнил:** ст. гр. ТКИ-342

Речкалов Р.Ю.

Ярий А.А.

**Проверил:** к.т.н. Сафронов А.И.

**Москва – 2024 г**

# Цель работы

Закрепить навыки разработки визуального пользовательского интерфейса, освоить работу с текстовыми файлами в среде *Microsoft Visual Studio*, научиться взаимно увязывать одни элементы управления с другими, получить представление о перерисовке и перемасштабировании».

# Формулировка задачи

«В интегрированной среде разработки *Visual Studio* разработать программу в режиме *Windows Forms Application* на языке *Visual C#*, представляющую собой пользовательский интерфейс, содержащий главное меню, позволяющее:

1. Начать работу с приложением.
2. Прервать работу приложения.
3. Предоставить пользователю справочную информацию о работе с приложением.

Сама программа должна реализовывать вывод в списки значений аргумента и соответствующих им значений функций. Список функций должен обязательно содержать следующие пункты:

1. Логарифм по основанию 2,
2. Тангенс,
3. Возведение в квадрат,
4. Косинус,
5. Натуральный логарифм,
6. Арккосинус,
7. Извлечение корня,
8. Арктангенс,
9. Синус,
10. Десятичный логарифм,
11. Арксинус.

Согласно заданию, Варианты расположения расчётных функций в комбинированном или обычном списке Арктангенс, Тангенс, Десятичный логарифм, Извлечение корня, Возведение в квадрат, Арксинус. Натуральный логарифм, Синус, Арккосинус, Логарифм по основанию 2, Косинус,

Чётные варианты отключают стандартный блок управления экранной формой и создают авторские кнопки «Свернуть», «Развернуть», «Закрыть» внизу экранной формы.

Справочная информация должна быть вызвана в дочерней экранной форме и считана в статический по размеру ярлык из текстового файла (чётные варианты).

В режиме конструктора экранная форма не должна содержать ни одного элемента управления.

Решение (*solution*) должно содержать только одну единственную экранную форму. Экранная форма со справкой настраивается программным кодом.

Компоненты экранной формы со справкой должны быть чувствительными к изменению размера экранной формы по ширине и высоте (обязательно подключение события *Resize*).

Названия функций должны быть элементами комбинированного списка (чётные варианты).

Выбор конкретной функции из списка инициирует заполнение некоторого файла значениями аргумента и значениями функции. Разделитель данных подбирается согласно таблице вариантов:

++

Предусмотреть кнопку, нажатие на которую инициирует заполнение двух списков (один список под значения аргумента, другой список – под значения функции, округлённых по правилу *(N mod 5) + 1* для знаков после плавающей запятой, где *N* – номер варианта) данными о функциональной зависимости из файла. Обучающиеся с чётными номерами вариантов обновляют существующие основные списки.

В качестве исходного состояния функциональных компонентов главной экранной формы принимается либо недоступное состояние (*.Enabled = false*) – чётные варианты.

Кнопка «Начало работы с приложением» должна реализовывать активацию доступа пользователя к функциональным компонентам или отображение функциональных компонентов на экранной форме для пользователя.

Все элементы программы должны носить значащие имена переменных, в которых отражено существо этих элементов, например, экранная форма – *frmMain*, ярлык – *lblHelp*, комбинированный список – *cmbFigures* и т.д.

Выбор значения абсциссы в списке должен инициировать выбор соответствующего ему значения ординаты и наоборот – выбор значения ординаты должен инициировать выбор соответствующего ему значения абсциссы (работа с параметрами *SelectedIndex* или *SelectedItem* списков.

Варианты дополнительных функциональных зависимостей, рассчитываемых программой на 10 позицию:

|log2 |x| - 1|

# 3. Составление диаграммы классов, входящих в состав решения.

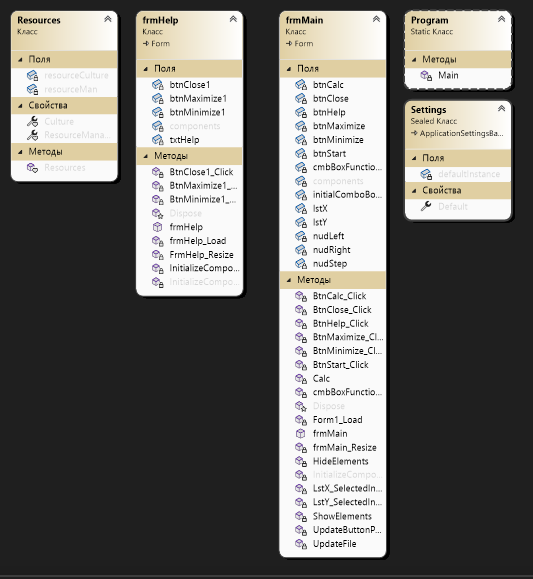


Рисунок 1 – Диаграмма классов

# 4. Составление сети Петри запрограммированного технологического процесса.

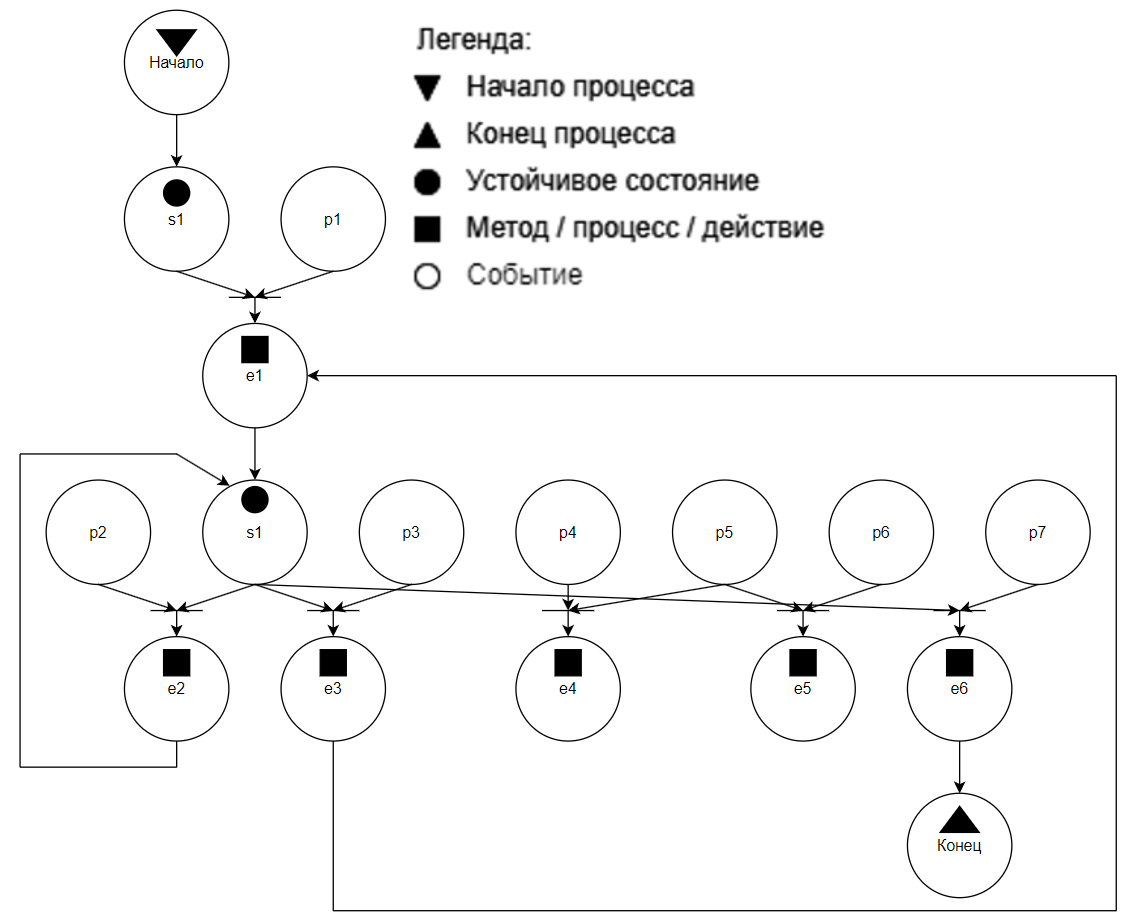


Рисунок 2 – Сеть Петри основной формы

**Описание сети Петри**

**Описание сети Петри**

- состояния (states)

s1 – форма ожидает действий пользователя

- действия (effects)

e1 – разблокируются кнопки взаимодействия;

кнопка «Начало работы с приложением» деактивируется

e2 – запись в файл аргументов и результатов функции

e3 – вывод результатов из файла в списки

e4 – выбор соответствующего элемента из списка ординат

e5 – выбор соответствующего элемента из списка абсцисс

e6 – закрытие формы

- события (prompts)

p1 – нажата кнопка «Начало работы с приложением»

p2 – в списке выбрана функция

p3 – нажатие кнопки рассчитать

p4 – пользователь выбрал элемент из списка абсцисс

p5 – списки заполнены значениями

p6 – пользователь выбрал элемент из списка ординат

p7 – нажата кнопка закрытия

# 5. Составление схем алгоритмов методов в составе решения, отмеченных на сети Петри в качестве «эффектов» (метка ).



Рисунок 3 – Алгоритм кнопки "Начать работу с приложением"

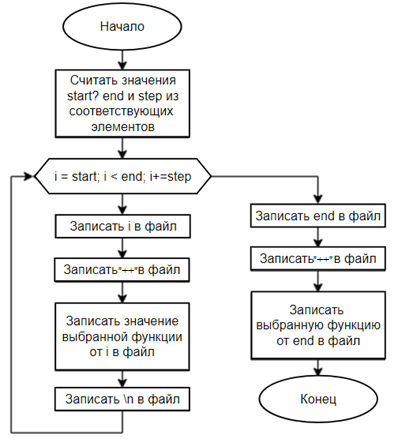


Рисунок 4 – Алгоритм записи в файл

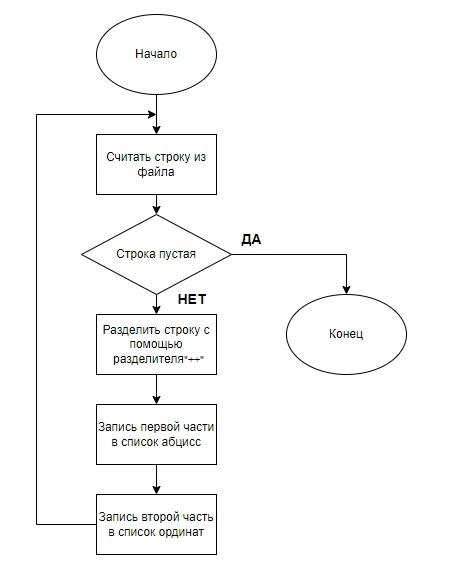


Рисунок 5 – Алгоритм чтения из файла

# 6. Подбор тестовых примеров.

Тестовые примеры:

Проверить корректность вычисления каждой из формул

Проверить запись в файл

# 7. Листинг (код) составленного программного обеспечения.

Код основной формы:

using System;

using System.Drawing;

using System.IO;

using System.Windows.Forms;

namespace Task3

{

public partial class frmMain : Form

{

private ListBox lstX;

private ListBox lstY;

private NumericUpDown nudLeft;

private NumericUpDown nudRight;

private NumericUpDown nudStep;

private Button btnCalc;

private Button btnStart;

private Button btnClose;

private Button btnMinimize;

private Button btnMaximize;

private Button btnHelp;

private ComboBox cmbBoxFunctions;

private Size initialComboBoxSize;

public frmMain()

{

InitializeComponent();

this.ControlBox = false;

HideElements();

}

private void HideElements()

{

foreach (Control control in this.Controls)

{

if (control != btnStart) // Скрываем все элементы, кроме кнопки "Старт"

{

control.Visible = false;

}

}

}

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

{

ShowElements(); // Показываем все элементы

btnStart.Visible = false;

// Активируем все элементы управления

foreach (Control control in this.Controls)

{

control.Enabled = true;

}

}

private void ShowElements()

{

foreach (Control control in this.Controls)

{

control.Visible = true;

}

}

private void frmMain\_Resize(object sender, EventArgs e)

{

// Вычисляем новое положение комбинированного списка после изменения размеров формы

cmbBoxFunctions.Location = new Point(230, 30);

UpdateButtonPositions();

cmbBoxFunctions.Location = new Point(230, 30);

cmbBoxFunctions.Size = initialComboBoxSize; // Используем сохраненный размер

UpdateButtonPositions();

}

private void UpdateButtonPositions()

{

int buttonWidth = 20; // Ширина кнопок

int margin = 5; // Отступ между кнопками и краями формы

// Кнопка "Закрыть"

btnClose.Location = new Point(this.ClientSize.Width - buttonWidth - margin, margin);

// Кнопка "Максимизировать"

btnMaximize.Location = new Point(btnClose.Left - buttonWidth - margin, margin);

// Кнопка "Минимизировать"

btnMinimize.Location = new Point(btnMaximize.Left - buttonWidth - margin, margin);

}

private void Form1\_Load(object sender, EventArgs e)

{

this.Text = "Задание №3 выполнили:Речкалов Р.Ю., Ярий А.А.; Номер варианта: 14; Дата выполнения: 25/04/2024";

this.BackColor = SystemColors.Control;

this.ClientSize = new Size(800, 600);

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.lstX = new ListBox();

this.lstY = new ListBox();

this.nudLeft = new NumericUpDown();

this.nudRight = new NumericUpDown();

this.nudStep = new NumericUpDown();

this.btnCalc = new Button();

this.btnStart = new Button();

this.btnClose = new Button();

this.btnMinimize = new Button();

this.btnMaximize = new Button();

this.btnHelp = new Button();

this.cmbBoxFunctions = new ComboBox();

//cmbBoxFunctions

this.cmbBoxFunctions.Anchor = ((System.Windows.Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Left)

| System.Windows.Forms.AnchorStyles.Right);

this.cmbBoxFunctions.DropDownWidth = 160;

this.cmbBoxFunctions.Items.AddRange(new object[] { "Арктангенс", "Тангенс", "Десятичный логарифм", "Извлечение корня", "Возведение в квадрат", "Арксинус", "Натуральный логарифм", "Синус", "Арккосинус", "|log2 |x| - 1|", "Логарифм по основанию 2", "Косинус" });

this.cmbBoxFunctions.Location = new System.Drawing.Point(230, 10);

this.cmbBoxFunctions.Size = new System.Drawing.Size(160, 100);

this.cmbBoxFunctions.TabIndex = 7;

this.cmbBoxFunctions.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;

//listX

this.lstX.Location = new Point(10, 10);

this.lstX.Size = new Size(150, 550);

this.lstX.SelectedIndexChanged += LstX\_SelectedIndexChanged;

//listY

this.lstY.Location = new Point(170, 10);

this.lstY.Size = new Size(150, 550);

this.lstY.SelectedIndexChanged += LstY\_SelectedIndexChanged;

//Выбор левой границы

this.nudLeft.Location = new Point(230, 40);

this.nudLeft.Size = new Size(50, 20);

this.nudLeft.Minimum = -100;

this.nudLeft.Maximum = 100;

this.nudLeft.Value = -5;

this.nudLeft.DecimalPlaces = 1;

this.nudLeft.Increment = (decimal)0.1;

//Выбор правой границы

this.nudRight.Location = new Point(285, 40);

this.nudRight.Size = new Size(50, 20);

this.nudRight.Minimum = -100;

this.nudRight.Maximum = 100;

this.nudRight.Value = 5;

this.nudRight.DecimalPlaces = 1;

this.nudRight.Increment = (decimal)0.1;

//Выбор шага

this.nudStep.Location = new Point(340, 40);

this.nudStep.Size = new Size(50, 20);

this.nudStep.Minimum = (decimal)0.1;

this.nudStep.Maximum = 10;

this.nudStep.Value = (decimal)0.1;

this.nudStep.DecimalPlaces = 1;

this.nudStep.Increment = (decimal)0.1;

//Кнопка "Вычислить"

this.btnCalc.Location = new Point(230, 70);

this.btnCalc.Size = new Size(160, 80);

this.btnCalc.FlatStyle = FlatStyle.Flat;

this.btnCalc.FlatAppearance.BorderSize = 1;

this.btnCalc.Text = "Вычислить";

this.btnCalc.Click += BtnCalc\_Click;

//Кнопка "Старт"

btnStart.Location = new Point((this.ClientSize.Width - btnStart.Size.Width) / 2, (this.ClientSize.Height - btnStart.Size.Height) / 2);

this.btnStart.Size = new Size(100, 40);

this.btnStart.FlatStyle = FlatStyle.Flat;

this.btnStart.FlatAppearance.BorderSize = 1;

this.btnStart.Text = "Старт";

this.btnStart.Click += BtnStart\_Click;

//Кнопка "Справка"

this.btnHelp.Location = new Point(230, 160);

this.btnHelp.Size = new Size(160, 80);

this.btnHelp.FlatStyle = FlatStyle.Flat;

this.btnHelp.FlatAppearance.BorderSize = 1;

this.btnHelp.Text = "Справка";

this.btnHelp.Click += BtnHelp\_Click;

//Кнопка "Закрыть"

this.btnClose.Location = new Point(375, 280);

this.btnClose.Size = new Size(20, 20);

this.btnClose.FlatStyle = FlatStyle.Flat;

this.btnClose.FlatAppearance.BorderSize = 1;

this.btnClose.Text = "X";

this.btnClose.Click += BtnClose\_Click;

//Кнопка "Минимизировать"

this.btnMinimize.Location = new Point(325, 280);

this.btnMinimize.Size = new Size(20, 20);

this.btnMinimize.FlatStyle = FlatStyle.Flat;

this.btnMinimize.FlatAppearance.BorderSize = 1;

this.btnMinimize.Text = "-";

this.btnMinimize.Click += BtnMinimize\_Click;

//Кнопка "Максимизировать"

this.btnMaximize.Location = new Point(350, 280);

this.btnMaximize.Size = new Size(20, 20);

this.btnMaximize.FlatStyle = FlatStyle.Flat;

this.btnMaximize.FlatAppearance.BorderSize = 1;

this.btnMaximize.Text = "⬜";

this.btnMaximize.Click += BtnMaximize\_Click;

// Устанавливаем координаты кнопок сворачивания, максимизации, минимизации и закрытия

int buttonWidth = 20; // Ширина кнопок

int buttonHeight = 20; // Высота кнопок

int margin = 5; // Отступ между кнопками и краями формы

// Кнопка "Справка"

btnHelp.Location = new Point(530, 480);

btnHelp.Size = new Size(160, 80);

// Комбинированный список

cmbBoxFunctions.Location = new Point(330, 10);

cmbBoxFunctions.Size = new Size(160, 100);

// Кнопка "Вычислить"

btnCalc.Location = new Point(530, 45);

btnCalc.Size = new Size(160, 80);

// Место для числовых значений (например, nudLeft, nudRight, nudStep)

nudLeft.Location = new Point(530, 10);

nudRight.Location = new Point(585, 10);

nudStep.Location = new Point(640, 10);

// Кнопка "Закрыть"

btnClose.Location = new Point(this.ClientSize.Width - buttonWidth - margin, margin);

btnClose.Size = new Size(buttonWidth, buttonHeight);

// Кнопка "Максимизировать"

btnMaximize.Location = new Point(btnClose.Left - buttonWidth - margin, margin);

btnMaximize.Size = new Size(buttonWidth, buttonHeight);

// Кнопка "Минимизировать"

btnMinimize.Location = new Point(btnMaximize.Left - buttonWidth - margin, margin);

btnMinimize.Size = new Size(buttonWidth, buttonHeight);

// Переместите кнопку "Старт" на новую позицию, чтобы она не перекрывала кнопки сверху

btnStart.Location = new Point((this.ClientSize.Width - btnStart.Size.Width) / 2, (this.ClientSize.Height - btnStart.Size.Height) / 2);

foreach (Control control in this.Controls)

{

if (control != btnStart)

{

control.Visible = false;

}

}

this.Controls.AddRange(new Control[] { cmbBoxFunctions, lstX, lstY, nudLeft, nudRight, nudStep, btnCalc, btnClose, btnMinimize, btnMaximize, btnHelp });

foreach (Control control in this.Controls)

{

control.Enabled = false;

}

this.Controls.Add(btnStart);

foreach (Control control in this.Controls)

{

control.BackColor = SystemColors.Control;

control.ForeColor = SystemColors.ControlText;

}

this.Controls.Add(btnStart);

btnStart.BringToFront();

this.Resize += frmMain\_Resize;

initialComboBoxSize = cmbBoxFunctions.Size;

}

public partial class frmHelp : Form

{

private Button btnClose1;

private Button btnMinimize1;

private Button btnMaximize1;

private Label txtHelp;

public frmHelp()

{

InitializeComponent(); // Вызываем метод InitializeComponent() для инициализации компонентов формы

this.ControlBox = false;

}

private void InitializeComponent()

{

this.Text = "Справка";

this.ClientSize = new Size(700, 200);

this.txtHelp = new Label();

this.txtHelp.Location = new Point(0, 0);

this.txtHelp.Size = new Size(700, 150);

// Правый верхний угол

int rightTopMargin = 4;

int buttonWidth = 20;

int buttonHeight = 20;

this.btnClose1 = new Button();

this.btnClose1.Location = new Point(this.ClientSize.Width - rightTopMargin - buttonWidth, rightTopMargin);

this.btnClose1.Size = new Size(buttonWidth, buttonHeight);

this.btnClose1.FlatStyle = FlatStyle.Flat;

this.btnClose1.FlatAppearance.BorderSize = 1;

this.btnClose1.Text = "X";

this.btnClose1.Click += BtnClose1\_Click;

this.btnMaximize1 = new Button();

this.btnMaximize1.Location = new Point(this.ClientSize.Width - 3 \* rightTopMargin - 3 \* buttonWidth, rightTopMargin);

this.btnMaximize1.Size = new Size(buttonWidth, buttonHeight);

this.btnMaximize1.FlatStyle = FlatStyle.Flat;

this.btnMaximize1.FlatAppearance.BorderSize = 1;

this.btnMaximize1.Text = "⬜";

this.btnMaximize1.Click += BtnMaximize1\_Click;

this.btnMinimize1 = new Button();

this.btnMinimize1.Location = new Point(this.ClientSize.Width - 2 \* rightTopMargin - 2 \* buttonWidth, rightTopMargin);

this.btnMinimize1.Size = new Size(buttonWidth, buttonHeight);

this.btnMinimize1.FlatStyle = FlatStyle.Flat;

this.btnMinimize1.FlatAppearance.BorderSize = 1;

this.btnMinimize1.Text = "-";

this.btnMinimize1.Click += BtnMinimize1\_Click;

this.Controls.AddRange(new Control[] { txtHelp, btnClose1, btnMinimize1, btnMaximize1 });

foreach (Control control in this.Controls)

{

control.BackColor = SystemColors.Control;

control.ForeColor = SystemColors.ControlText;

}

this.btnMaximize1.BringToFront();

this.btnMinimize1.BringToFront();

this.btnClose1.BringToFront();

this.Load += new EventHandler(frmHelp\_Load);

}

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

{

StreamReader sr = new StreamReader("..\\..\\Help.txt");

txtHelp.Text = sr.ReadToEnd();

sr.Close();

this.Resize += FrmHelp\_Resize;

}

private void FrmHelp\_Resize(object sender, EventArgs e)

{

this.txtHelp.Size = this.ClientSize;

}

private void BtnClose1\_Click(object sender, EventArgs e)

{

frmMain frm = new frmMain();

frm.Show();

this.Hide();

}

private void BtnMinimize1\_Click(object sender, EventArgs e)

{

this.WindowState = FormWindowState.Minimized;

}

private void BtnMaximize1\_Click(object sender, EventArgs e)

{

if (this.WindowState == FormWindowState.Normal)

{

this.WindowState = FormWindowState.Maximized;

}

else

{

this.WindowState = FormWindowState.Normal;

}

}

}

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

{

frmHelp fr2 = new frmHelp();

fr2.Show();

Hide();

}

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

{

Application.Exit();

}

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

{

this.WindowState = FormWindowState.Minimized;

}

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

{

if (this.WindowState == FormWindowState.Normal)

{

this.WindowState = FormWindowState.Maximized;

}

else

{

this.WindowState = FormWindowState.Normal;

}

}

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

{

this.lstX.SelectedIndex = this.lstY.SelectedIndex;

}

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

{

this.lstY.SelectedIndex = this.lstX.SelectedIndex;

}

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

{

this.UpdateFile();

this.lstX.Items.Clear();

this.lstY.Items.Clear();

StreamReader sr = new StreamReader(@"..\..\FunctionArguments.txt");

while (!sr.EndOfStream)

{

string str = sr.ReadLine();

string[] strarr = str.Split(new string[] { "++" }, StringSplitOptions.None);

// Проверяем, содержит ли массив strarr необходимое количество элементов

if (strarr.Length >= 2)

{

// Если да, добавляем в lstX и lstY соответствующие элементы

this.lstX.Items.Add(strarr[0]);

this.lstY.Items.Add(strarr[1]);

}

else

{

// Если нет, добавляем в lstX и lstY пустую строку или другое значение по умолчанию

this.lstX.Items.Add("");

this.lstY.Items.Add("");

}

}

sr.Close();

}

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

{

this.UpdateFile();

}

private void UpdateFile()

{

StreamWriter sw = new StreamWriter(@"..\..\FunctionArguments.txt");

decimal start = this.nudLeft.Value;

decimal end = this.nudRight.Value;

decimal step = this.nudStep.Value;

for (decimal i = start; i < end; i += step)

{

sw.Write(i);

sw.Write("++");

sw.Write(this.Calc(i));

sw.Write("\n");

}

sw.Write(end);

sw.Write("++");

sw.Write(this.Calc(end));

sw.Close();

}

private string Calc(decimal arg)

{

switch (this.cmbBoxFunctions.SelectedIndex)//Арктангенс, Тангенс, Десятичный логарифм,

//Извлечение корня, Возведение в квадрат,

//Арксинус. Натуральный логарифм,

//Синус, Арккосинус, лог модуль Логарифм по основанию 2, Косинус,

{

case 0:

return Math.Atan((double)arg).ToString("F3");//арктангенс

case 1:

if(Math.Cos((double)arg) != 0) return Math.Tan((double)arg).ToString("F3");//тангенс

else return "ERROR";

case 2:

if (arg > 0) return Math.Log((double)arg, 10).ToString("F3");//десятичный логарифм

else return "ERROR";

case 3:

if(arg >= 0) return Math.Sqrt((double)arg).ToString("F3");//квадратный корень

else return "ERROR";

case 4:

return Math.Pow((double)arg, 2).ToString("F3");//возведение в квадрат

case 5:

if (Math.Abs(arg) > 1) return "ERROR";

else return Math.Asin((double)arg).ToString("F3");//арксинус

case 6:

if (arg > 0) return Math.Log((double)arg, Math.E).ToString("F3");//натуральный логарифм

else return "ERROR";

case 7:

return Math.Sin((double)arg).ToString("F3");//синус

case 8:

if (Math.Abs(arg) > 1) return "ERROR";

else return Math.Acos((double)arg).ToString("F3");//арккосинус

case 9:

if ((arg != 0) && (arg > 0)) return Math.Abs(Math.Log((double)Math.Abs(arg), 2) - 1).ToString("F3");//лог модуль

else return "ERROR";

case 10:

if (arg > 0) return Math.Log((double)arg, 2).ToString("F3");//логарифм

else return "ERROR";

default:

return Math.Cos((double)arg).ToString("F3");//косинус

}

}

}

}

# 8. Графический пользовательский интерфейс программного обеспечения и его описание.



Рисунок 6 – Интерфейс основной формы

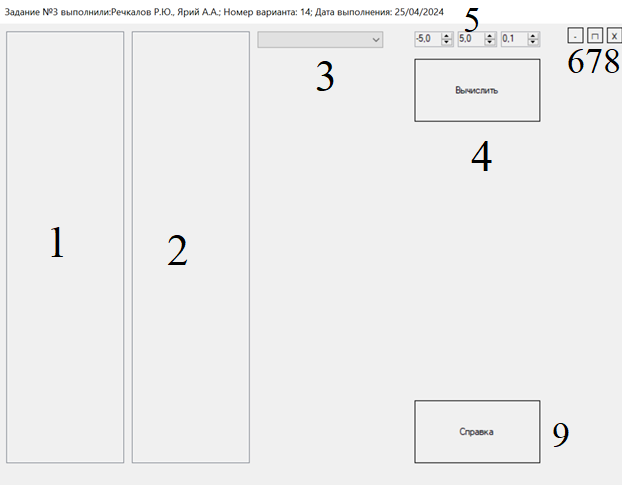


Рисунок 7 – Интерфейс основной формы после нажатия кнопки «Начало работы с приложением»

1. Список абсцисс
2. Список ординат
3. Комбинированный список с функциями
4. Кнопка расчета
5. Поля для выбора границ
6. Кнопка сворачивания функции
7. Кнопка максимизации/минимизации функции
8. Кнопка закрытия формы
9. Кнопка вызова формы со справкой

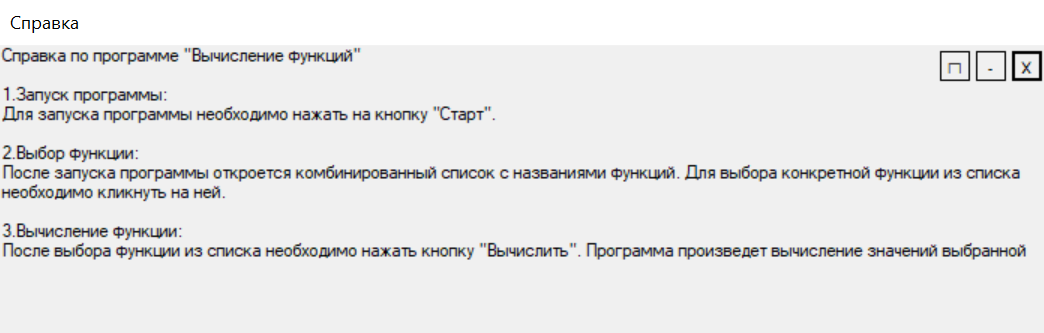


Рисунок 8 – Интерфейс вспомогательной формы

# Подтверждение соответствия графического пользовательского интерфейса требованиям к оформлению.

1. Заголовок экранной формы должен содержать надпись вида: «Задание №3 выполнил: [Фамилия И.О. автора]; Номер варианта: [Номер]; Дата выполнения: [дд/мм/гггг].



Рисунок 9 – Название формы

2. Дата выполнения проставляется в момент, когда программа считается законченной и по ней можно готовить итоговый отчёт о выполнении работы.

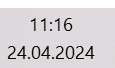


Рисунок 10 - Дата на момент написания отчета

3. Чётные варианты отключают стандартный блок управления экранной формой и создают авторские кнопки «Свернуть», «Развернуть», «Закрыть» внизу экранной формы.



Рисунок 11 - Авторские кнопки «Свернуть», «Развернуть», «Закрыть»

1. Справочная информация должна быть вызвана в дочерней экранной форме и считана в статический по размеру ярлык из текстового файла (чётные варианты).

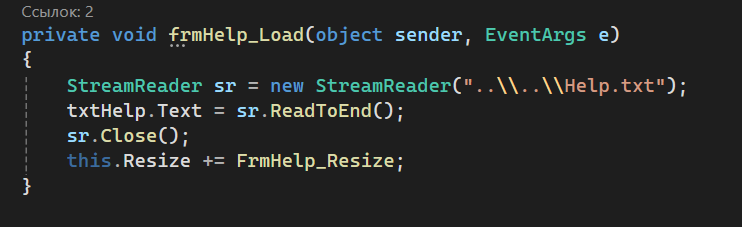


Рисунок 12 - Фрагмент кода с созданием ярлыка для выгрузки справочной информации

6. В режиме конструктора экранная форма не должна содержать ни одного элемента управления.

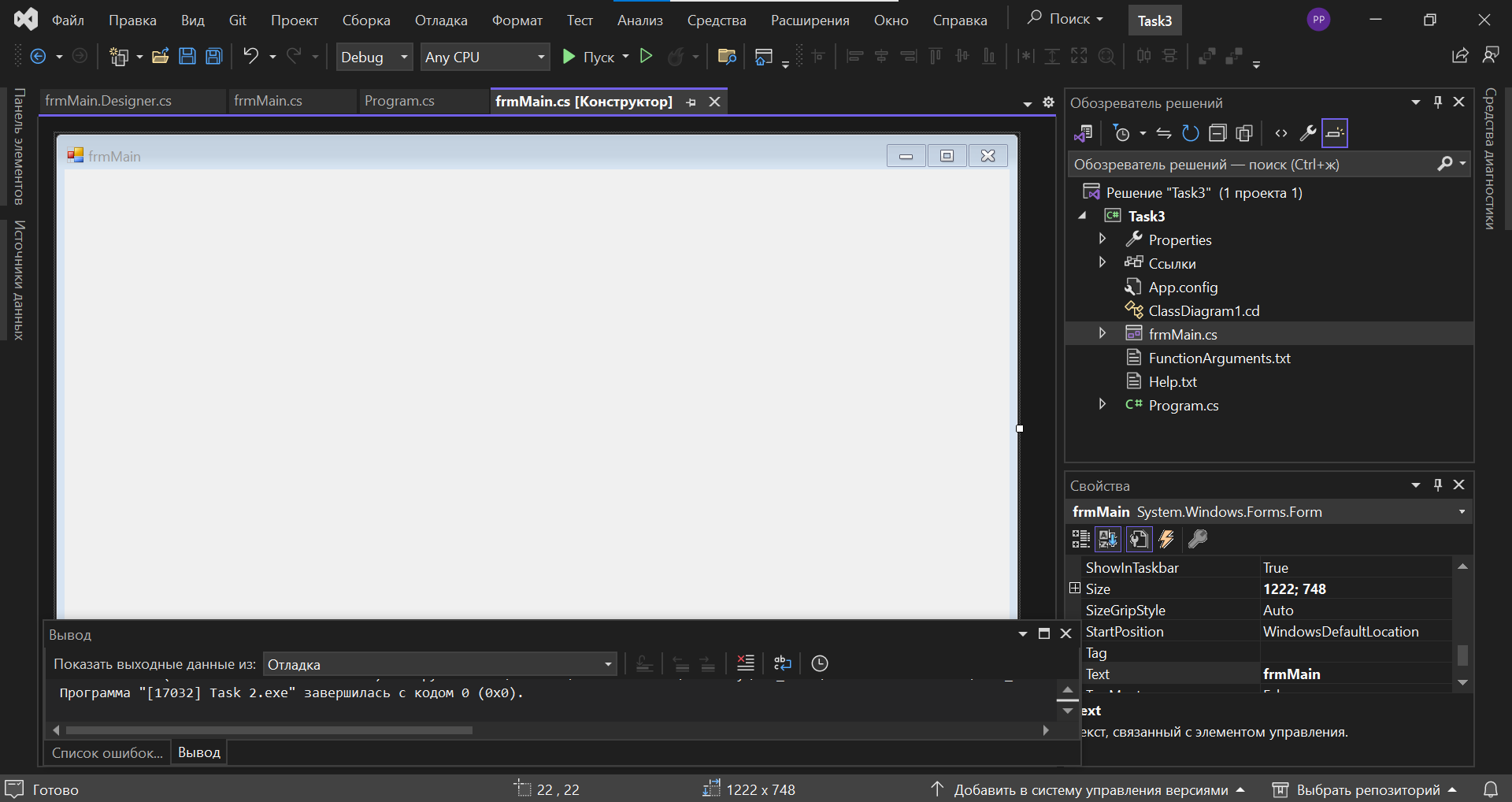


Рисунок 13 – Конструктор главной формы

7. Решение (*solution*) должно содержать только одну единственную экранную форму. Экранная форма со справкой настраивается программным кодом.

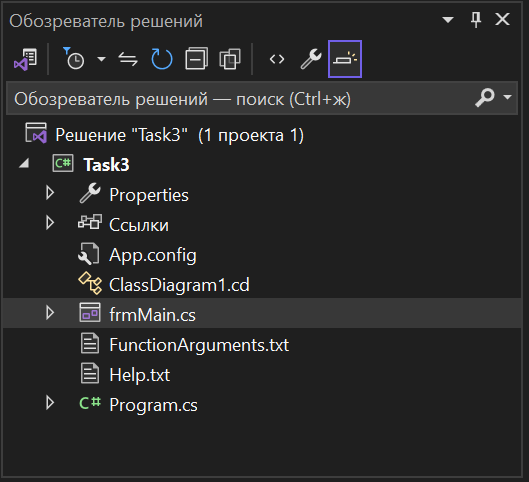


Рисунок 14 – Решение

8. Компоненты экранной формы со справкой должны быть чувствительными к изменению размера экранной формы по ширине и высоте (обязательно подключение события *Resize*).

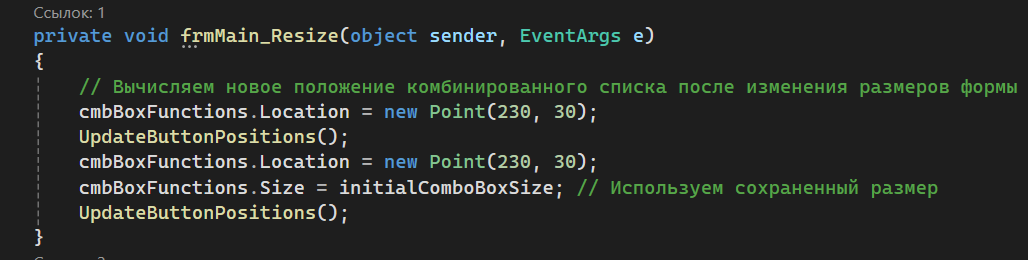


Рисунок 15 - Фрагмент кода с изменением размера компонента экранной формы со справкой

9. Названия функций должны быть элементами комбинированного списка (чётные варианты).

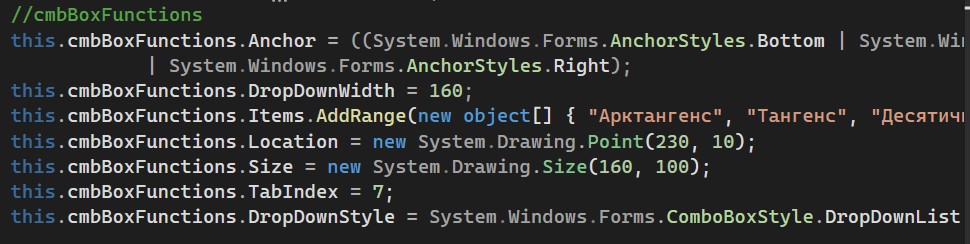


Рисунок 16 – Фрагмент кода с комбинированным списком

10.Выбор конкретной функции из списка инициирует заполнение некоторого файла значениями аргумента и значениями функции. Разделитель данных подбирается согласно таблице вариантов.

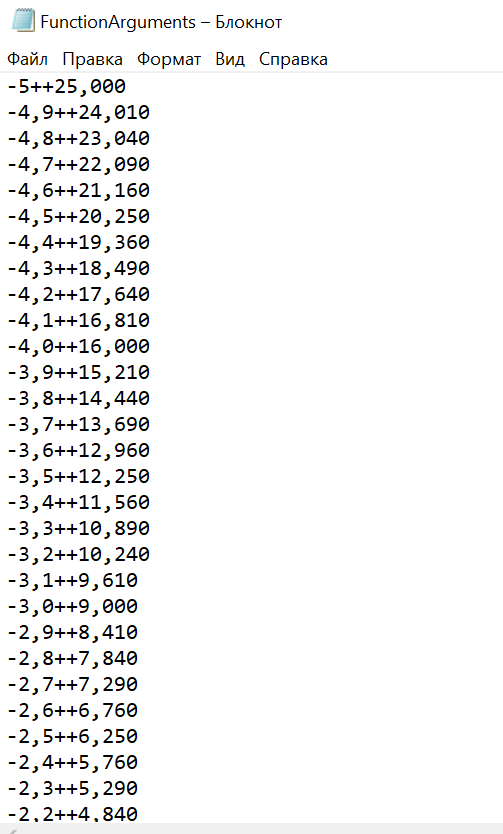


Рисунок 17 – Файл с данными

12. Предусмотреть кнопку, нажатие на которую инициирует заполнение двух списков (один список под значения аргумента, другой список – под значения функции, округлённых по правилу *(N mod 5) + 1* для знаков после плавающей запятой, где *N* – номер варианта) данными о функциональной зависимости из файла. Обучающиеся с чётными номерами вариантов обновляют существующие основные списки, с нечётными номерами вариантов выводят значения, считанные из файла, в дополнительные списки.

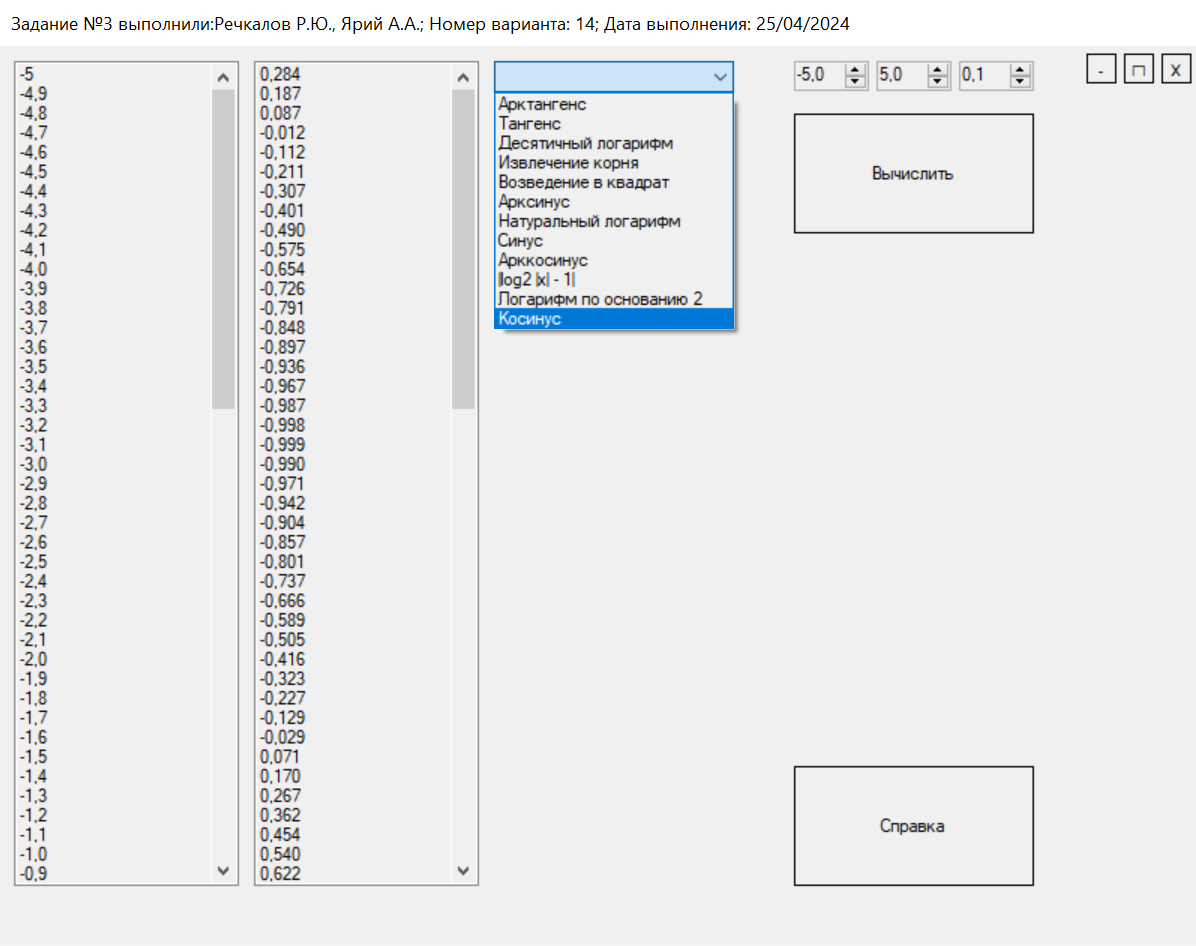


Рисунок 18 – Результат работы функции расчёта

13. В качестве исходного состояния функциональных компонентов главной экранной формы принимается либо недоступное состояние (*.Enabled = false*) – чётные варианты.

см. Рисунок 6

14. Кнопка «Начало работы с приложением» должна реализовывать активацию доступа пользователя к функциональным компонентам или отображение функциональных компонентов на экранной форме для пользователя.

см. Рисунок 6

15. Все элементы программы должны носить значащие имена переменных, в которых отражено существо этих элементов, например, экранная форма – *frmMain*, ярлык – *lblHelp*, комбинированный список – *cmbFigures* и т.д.

см. Диаграмму классов

16. Выбор значения абсциссы в списке должен инициировать выбор соответствующего ему значения ординаты и наоборот – выбор значения ординаты должен инициировать выбор соответствующего ему значения абсциссы (работа с параметрами *SelectedIndex* или *SelectedItem* списков).

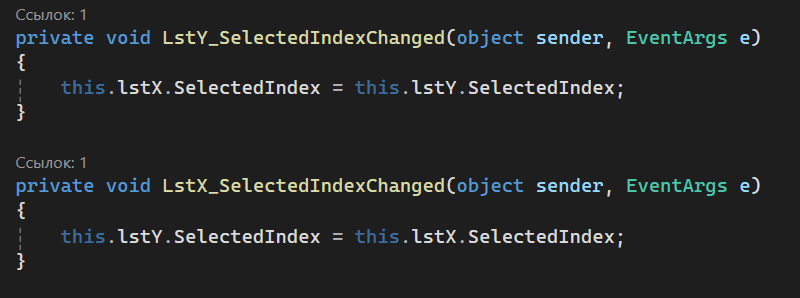


Рисунок 19 - Фрагмент кода с функциями обработки выбора значения в списках абсцисс или ординат

# 10. Расчёт тестовых примеров с использованием составленного программного обеспечения.

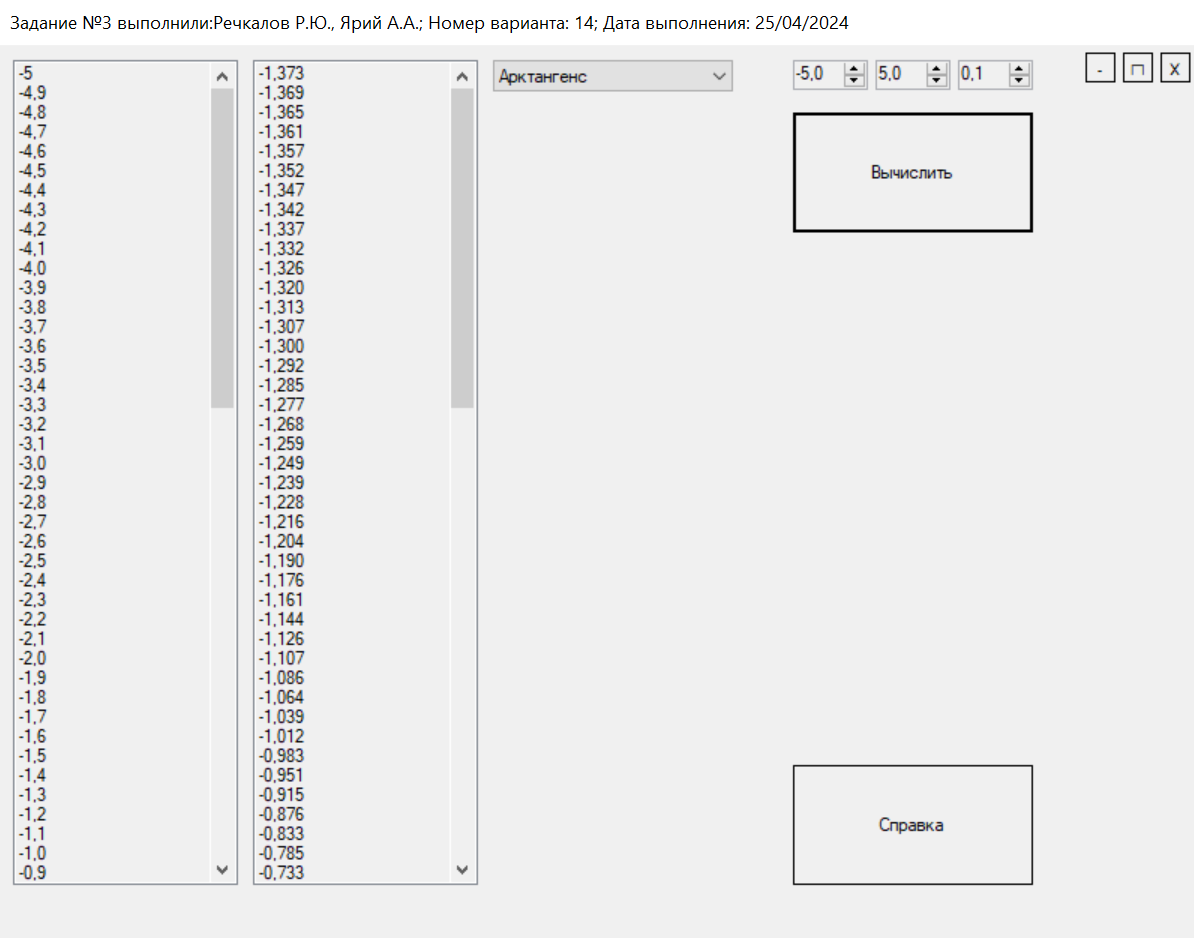


Рисунок 20 – Расчёт функции

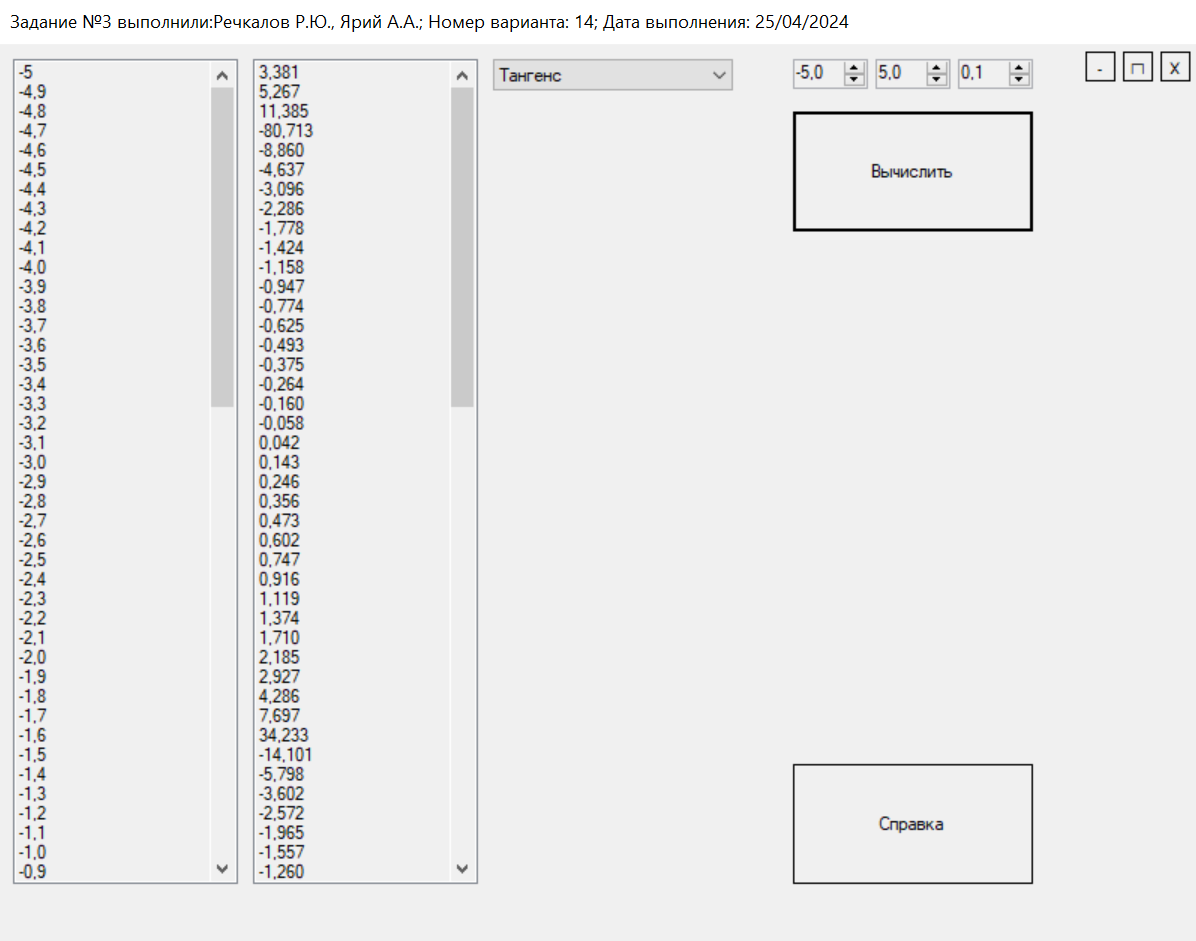


Рисунок 21 – Расчёт функции

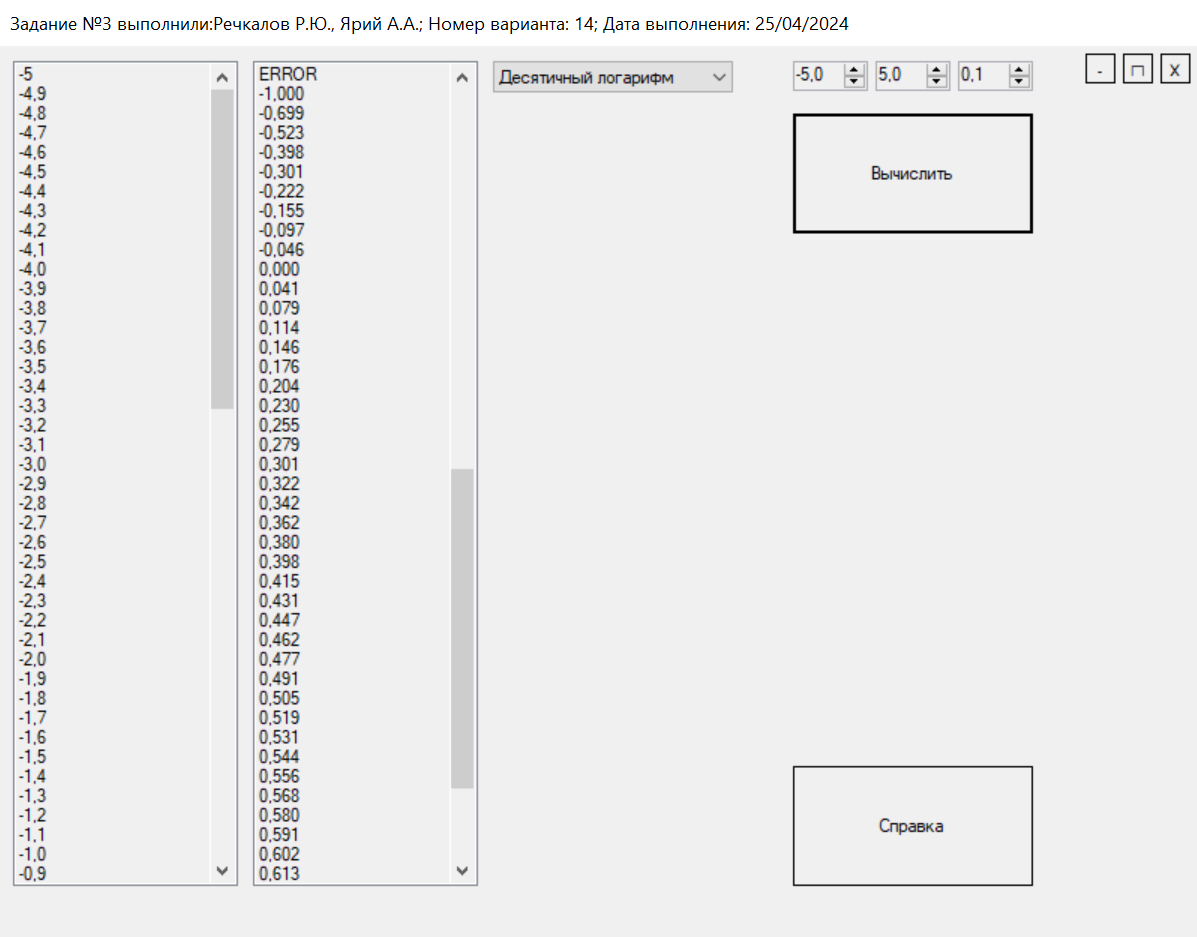


Рисунок 22 – Расчёт функции

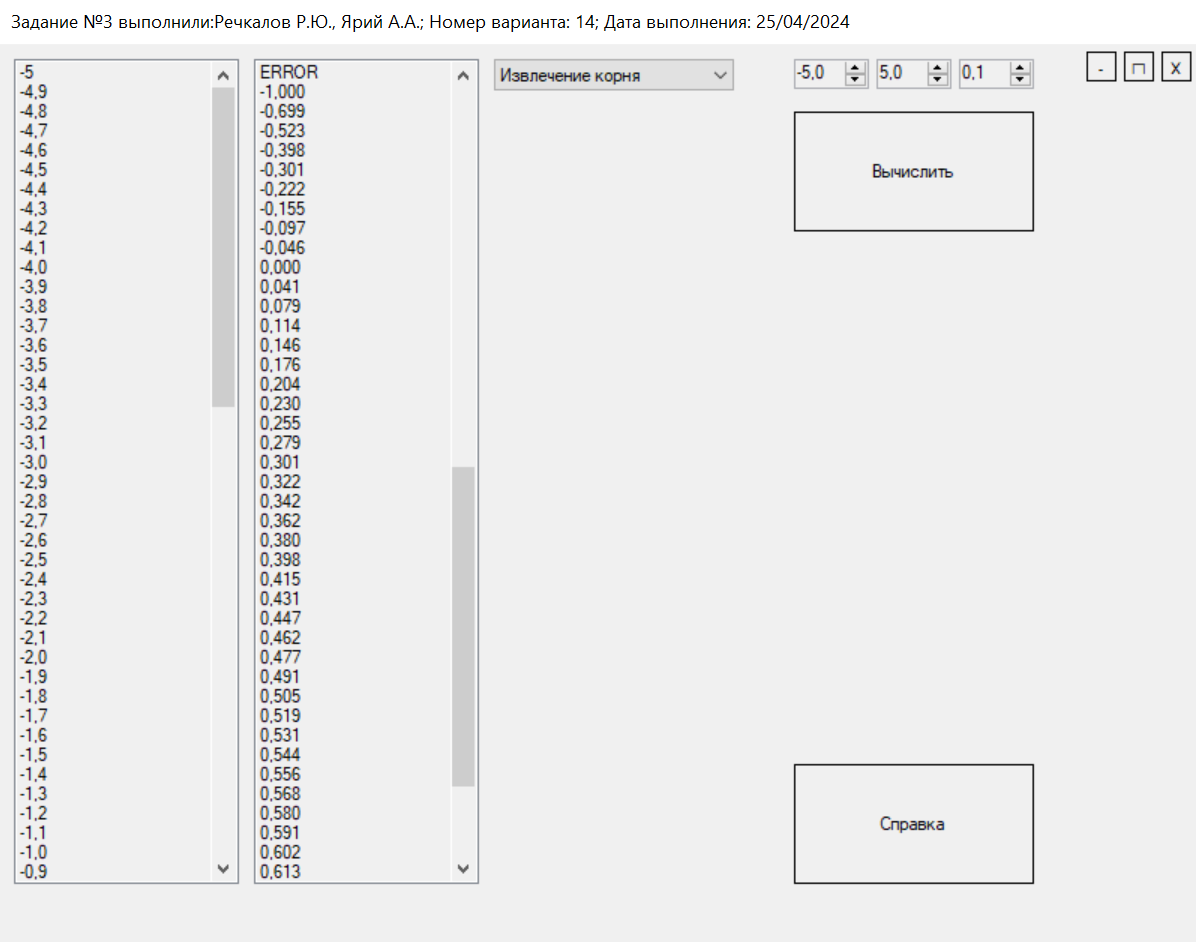


Рисунок 23 – Расчёт функции

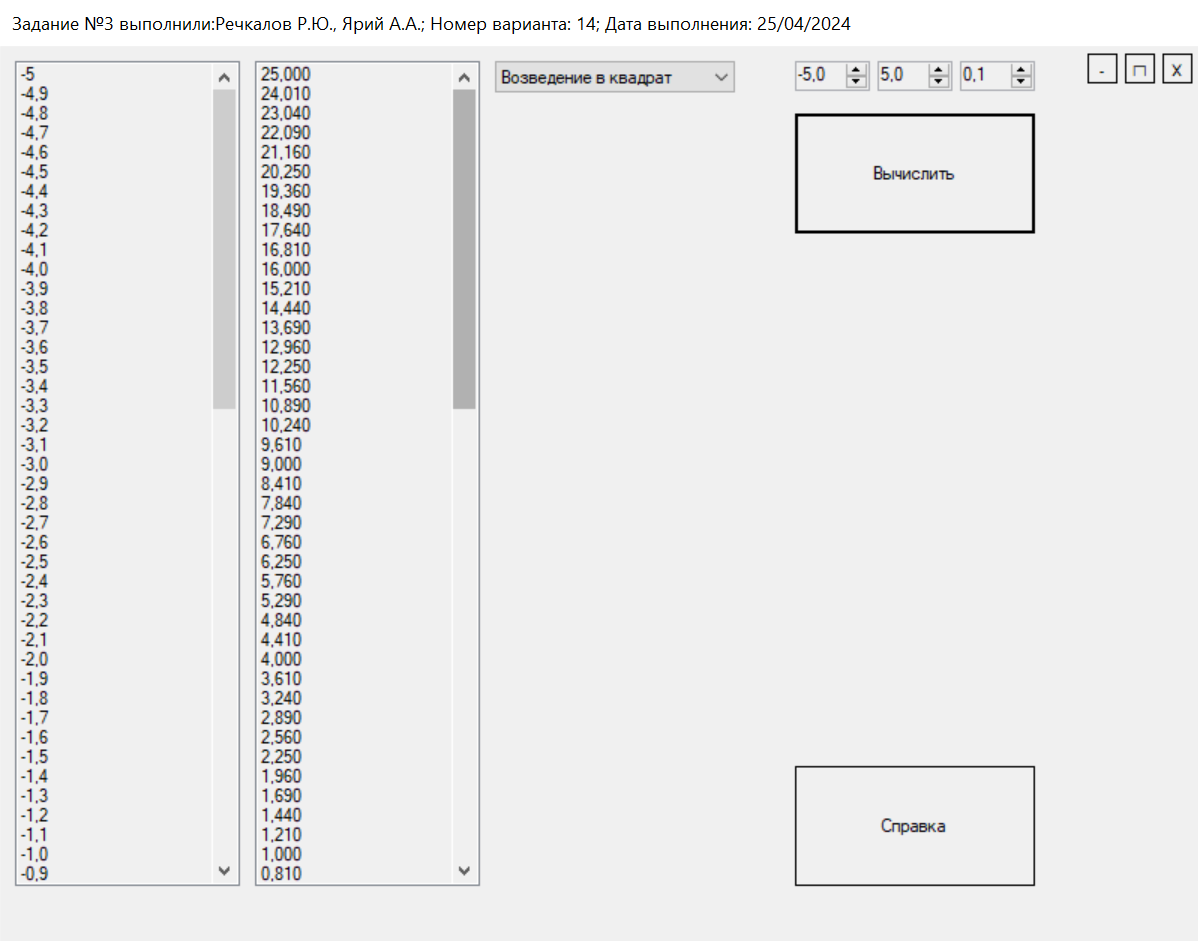
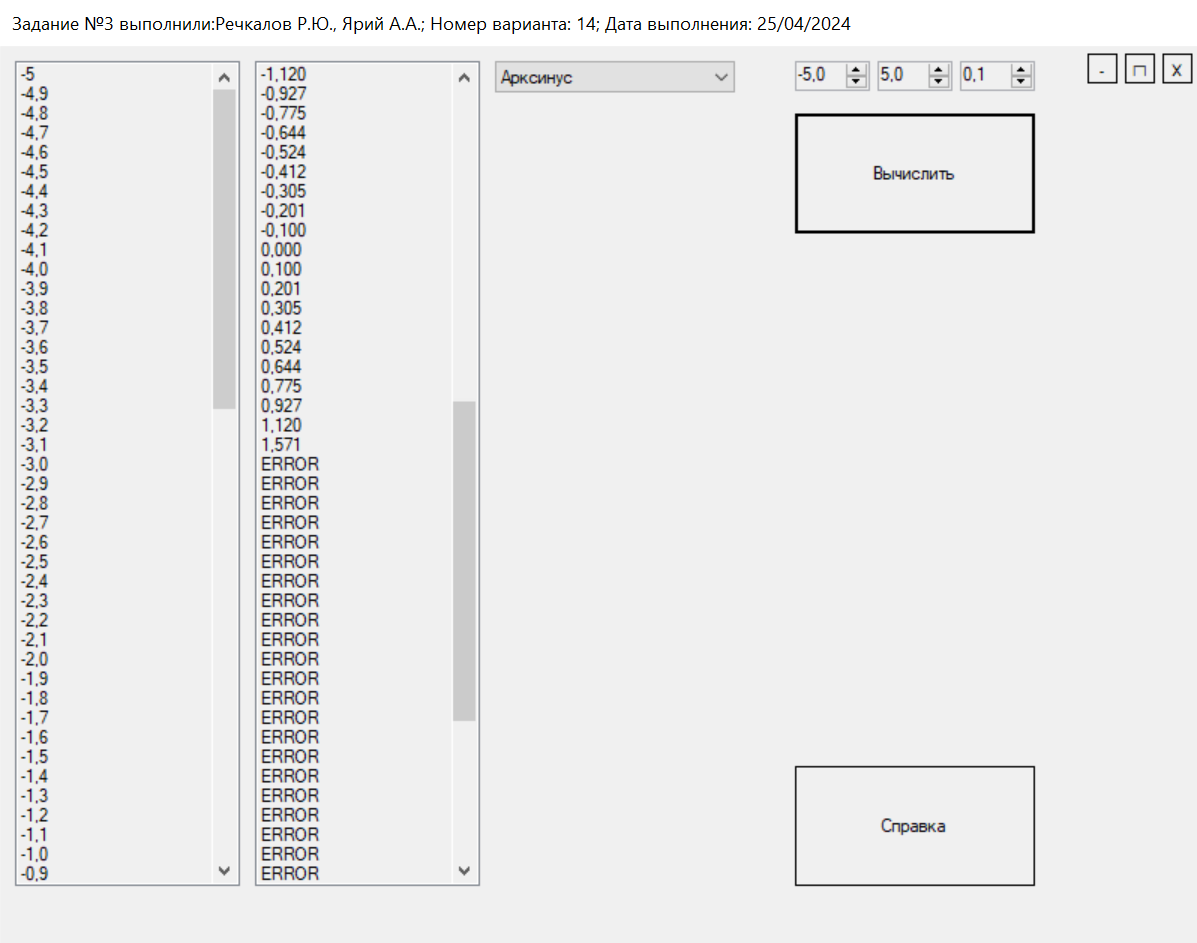
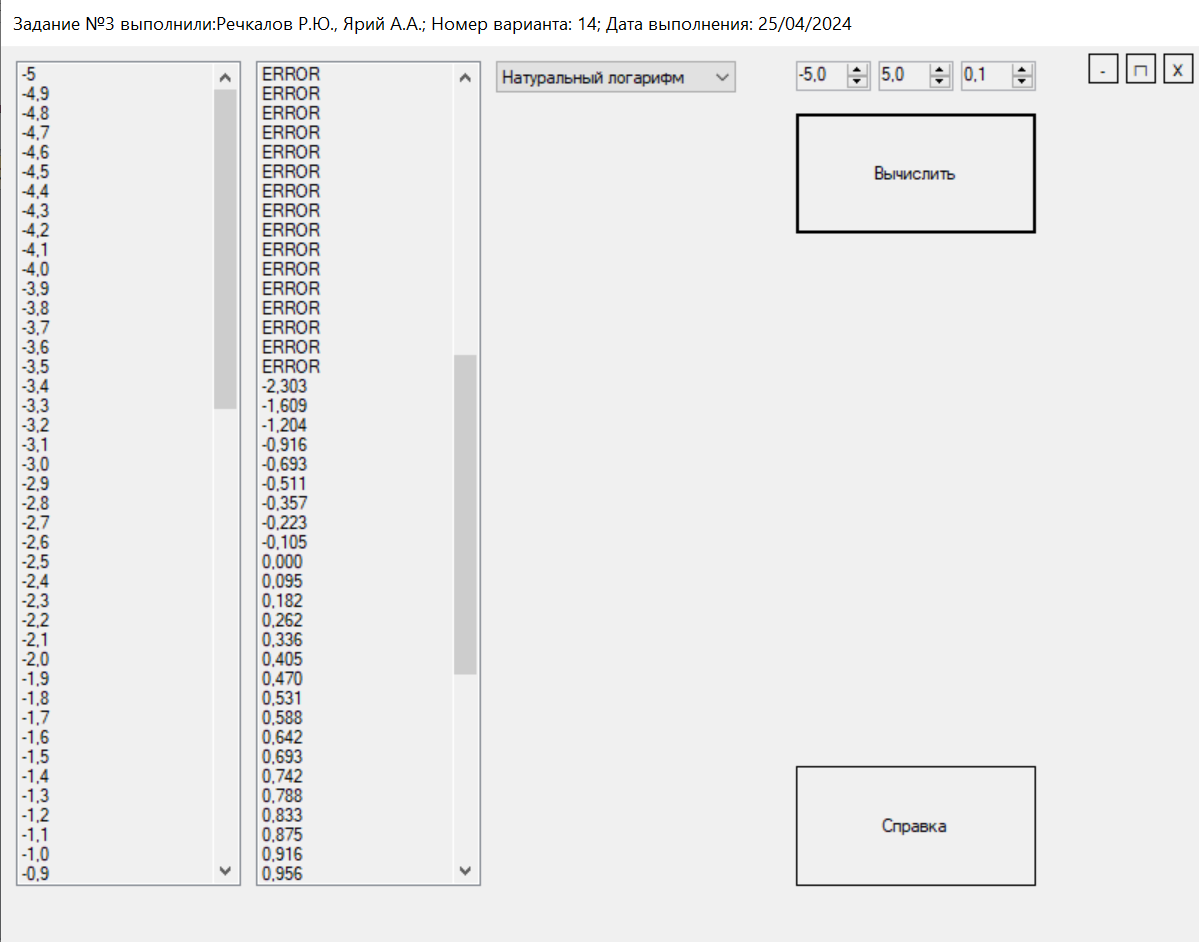
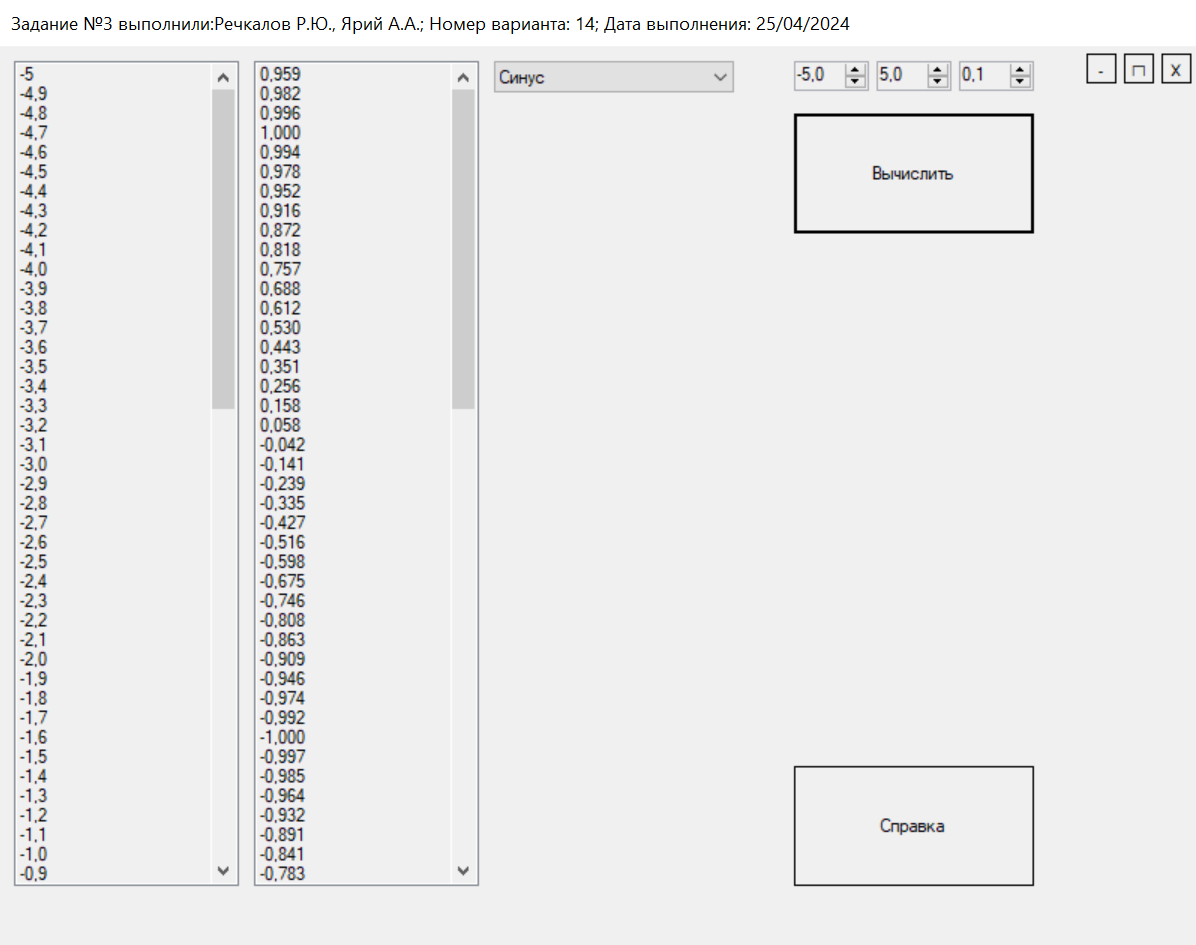
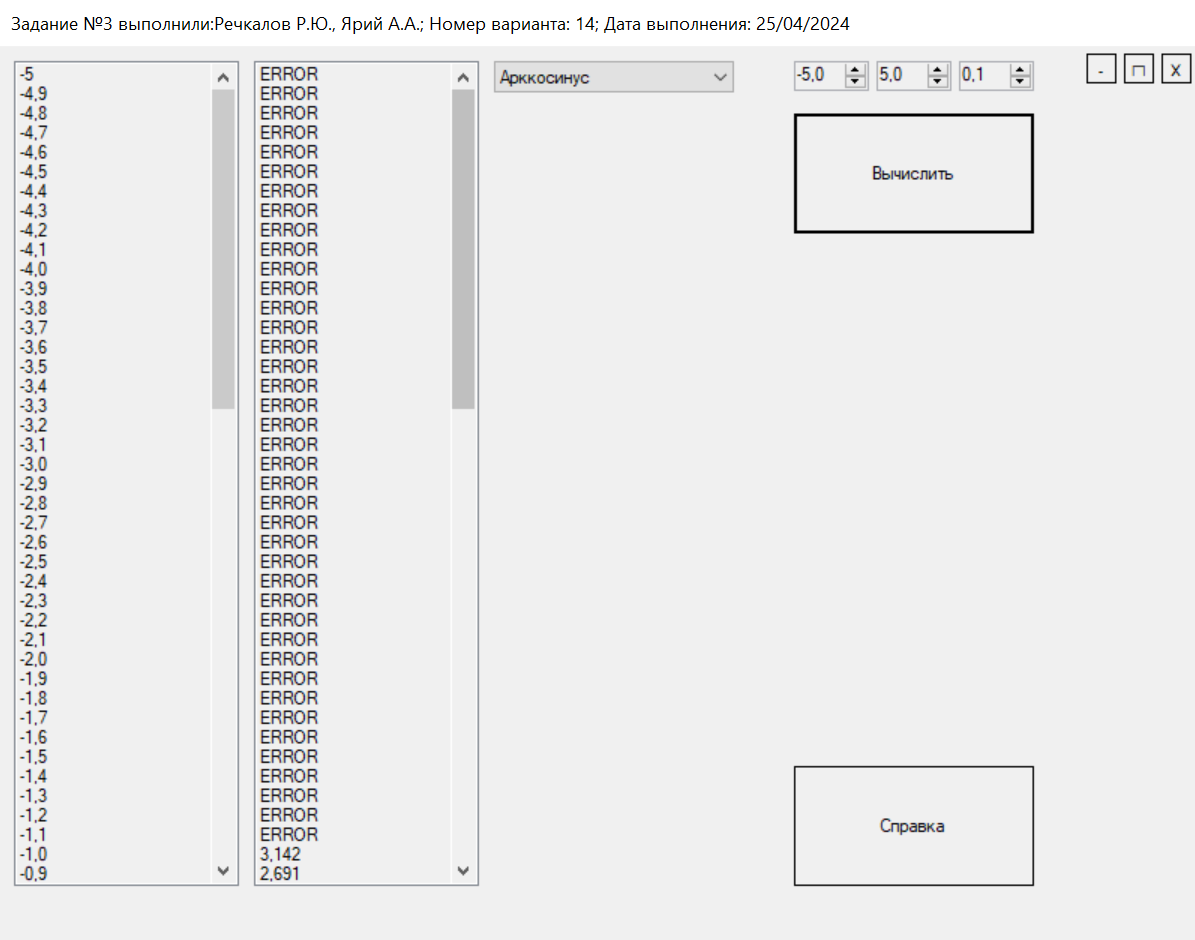


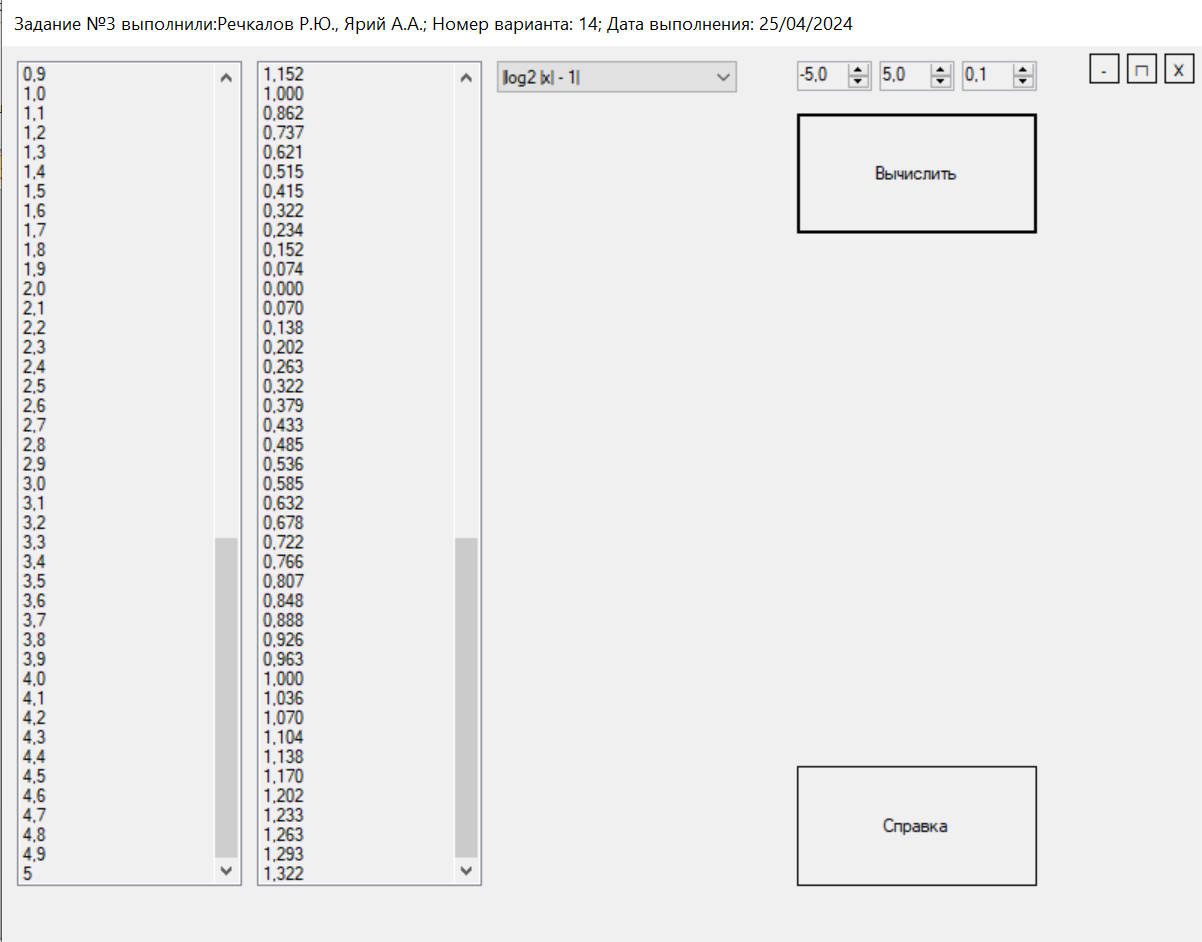
Рисунок 24 – Расчёт функции

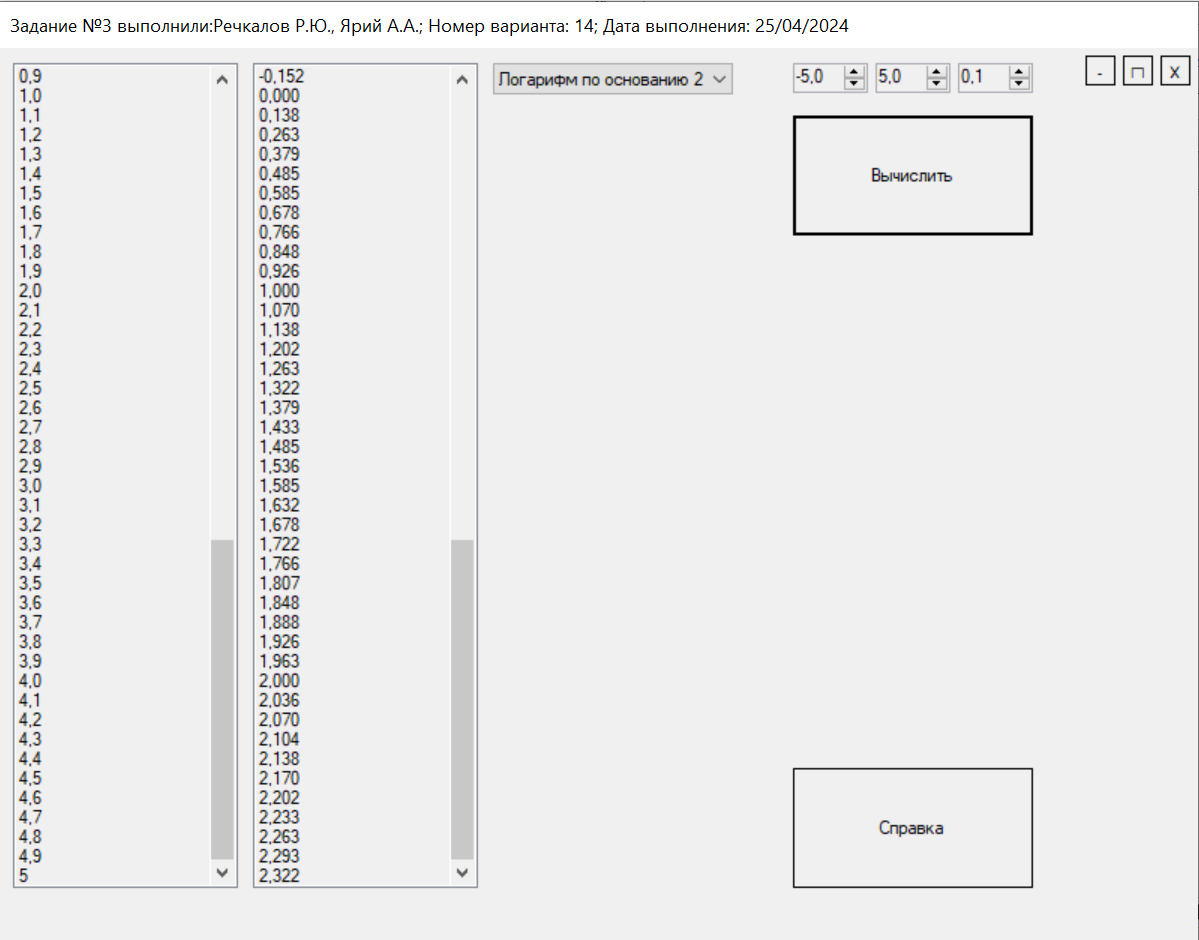
 Рисунок 25 – Расчёт функции

 Рисунок 26 – Расчёт функции

 Рисунок 27 – Расчёт функции

 Рисунок 28 – Расчёт функции

 Рисунок 29 – Расчёт функции

 Рисунок 30 – Расчёт функции

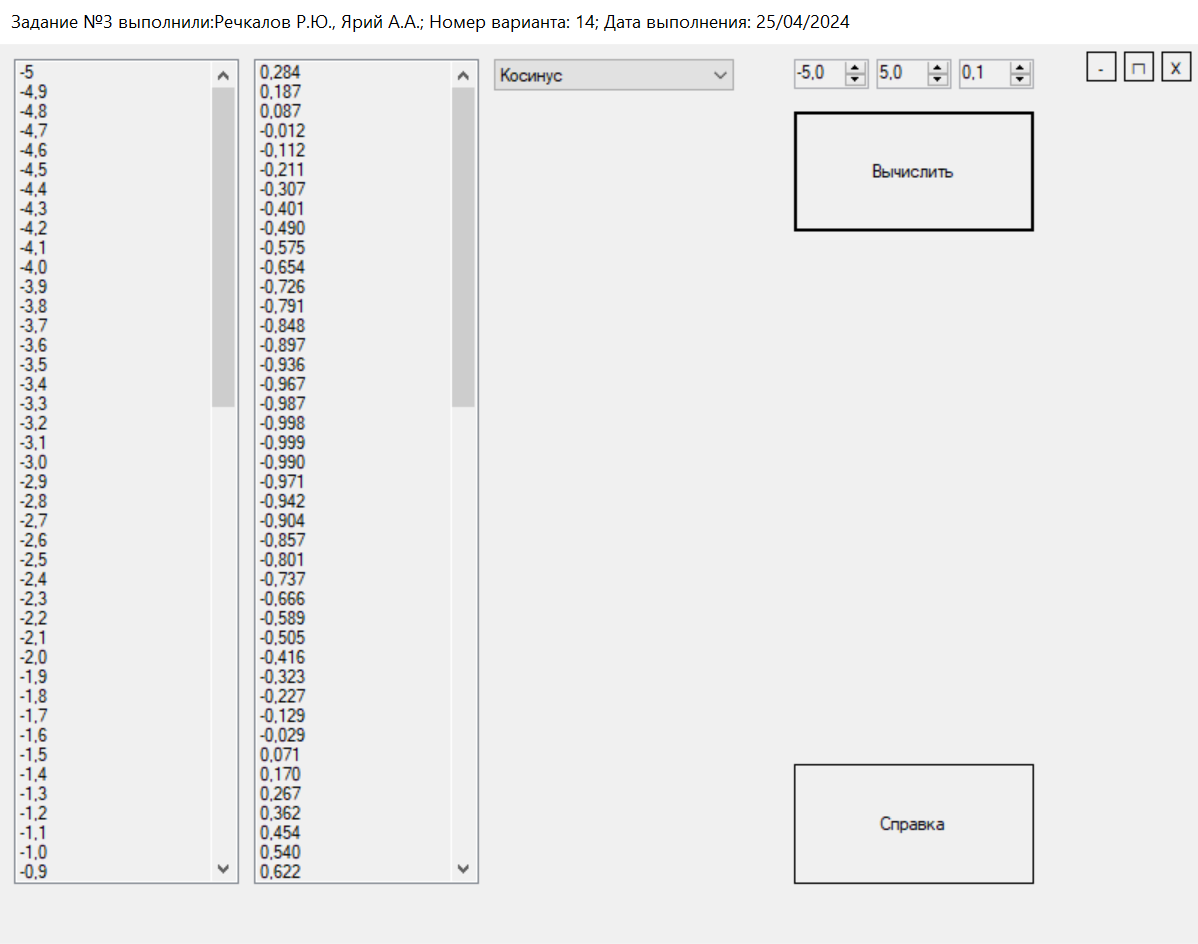


Рисунок 31 – Расчёт функции

# 11. Формулировка вывода о проделанной работе

В процессе выполнения данной работы мы существенно расширили наши знания и навыки в разработке приложений с графическим пользовательским интерфейсом на языке программирования C# в среде Microsoft Visual Studio. Подробно рассмотрим ключевые аспекты и результаты, достигнутые в ходе выполнения задания:

1.Разработка визуального интерфейса:

2.Работа с текстовыми файлами:

3.Управление элементами интерфейса:

4.Перерисовка и перемасштабирование элементов:

5.Реализация математических функций:

6.Обработка событий и взаимодействие с пользователем:

В целом, выполнение данной работы позволило нам глубже понять принципы разработки Windows Forms приложений на языке C# в Visual Studio, освоить множество практических навыков, необходимых для создания гибких, функциональных и удобных для пользователя программ. Мы научились эффективно использовать возможности C# для работы с графическим интерфейсом, текстовыми файлами и математическими расчетами, что существенно расширяет наши компетенции в области программирования.