Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Российский университет транспорта» (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа №3

по дисциплине

**«**Методы программирования**»**

Выполнили: студенты гр.ТКИ-342

Макаров Г.Б.

Лазухин М.С.

Вариант №17

Проверил: доцент кафедры УиЗИ, к.т.н. Сафронов А.И.

Москва ­– 2024 г.

1. **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 задания**

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.

Таблица 1 – Вариант расположения расчётных функций в комбинированном или обычном списке

|  |  |
| --- | --- |
| **№вар** | **Порядок индексов функций из списка** |
| 17 | 11, 7, 10, 6, 9, 3, 4, 8, 5, 2, 1. |

**Требования 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].

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

3. Нечётные варианты заменяют стандартный курсор экранной формы со «стрелки» на «руку».

4. Справочная информация должна быть вызвана в дочерней экранной форме и считана в статическое по размеру текстовое поле с выставленным запретом на редактирование текстовой информации (нечётные варианты).

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

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

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

8. Названия функций должны быть элементами обыкновенного списка (нечётные варианты).

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

Таблица 2 – Вариант разделителей, записываемых в файл

|  |  |
| --- | --- |
| **№вар** | **Разделитель** |
| 17 | ||| |

10. Предусмотреть кнопку, нажатие на которую инициирует заполнение двух списков (один список под значения аргумента, другой список – под значения функции, округлённых по правилу *(17 mod 5) + 1* для знаков после плавающей запятой данными о функциональной зависимости из файла. Обучающиеся с нечётными номерами вариантов выводят значения, считанные из файла, в дополнительные списки.

11. В качестве исходного состояния функциональных компонентов главной экранной формы принимается невидимое состояние (*.Visible = false*) – нечётные варианты.

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

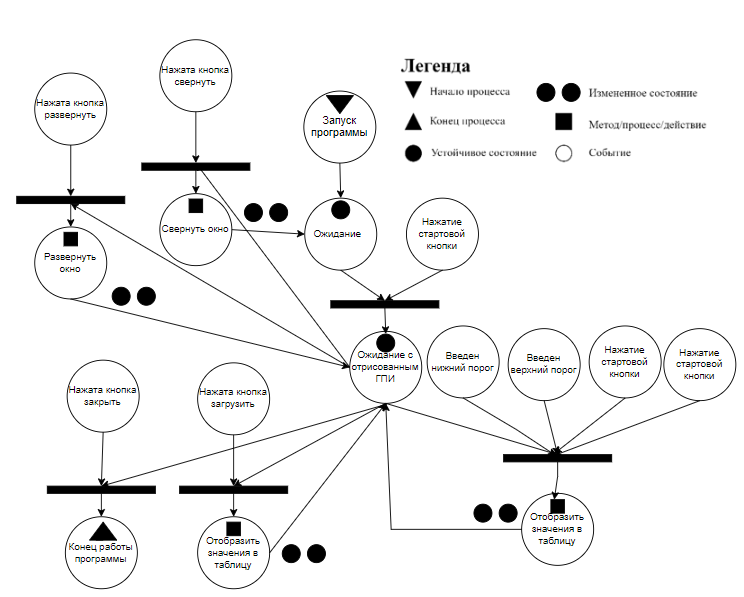
13. Все элементы программы должны носить значащие имена переменных, в которых отражено существо этих элементов, например, экранная форма – *frmMain*, ярлык – *lblHelp*, комбинированный список – *cmbFigures* и т.д.

14. 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).

**Вариант индивидуального задания:**

Таблица 3 – Вариант дополнительных функциональных зависимостей, рассчитываемых программой

|  |  |  |
| --- | --- | --- |
| **№вар** | **Функция** | **Позиция** |
| 17 | log2((x + 1) / (x - 1)) | 4 |

1. **Сеть Петри запрограммированного процесса**Рисунок 1 – Сеть Петри запрограммированного процесса
2. **Тестовые примеры**

* При нажатии кнопки «Справочная информация» происходит открытие окна со справочной информацией;
* Рассчитываются функции из списка;
* Округление значений по правилу *(17 mod 5) + 1*.

1. **Листинг кода**

namespace Laba3

{

    public partial class ParentForm : Form

    {

        public ParentForm()

        {

            InitializeComponent();

        }

        private Button btnGo;

        private Button btnInfo;

        private Button btnCheck;

        private TextBox tbMin;

        private TextBox tbMax;

        private ListBox lby;

        private ListBox lbx;

        private ListBox lbx1;

        private ListBox lby1;

        private ListBox lbMain;

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

        {

            buttons();

            textboxes();

            lists();

            btnGo.Visible = true;

        }

        private void buttons()

        {

            BackColor = SystemColors.MenuHighlight;

            Cursor = Cursors.Hand;

            Button Buttongenerator(int x, int y, string label, bool anchor)

            {

                Button example = new Button();

                example.Location = new Point(this.Width - x, this.Height - y);

                example.Text = label;

                example.AutoSize = true;

                example.BackColor = Color.Black;

                example.Padding = new Padding(6);

                example.Font = new Font("Times New Roman", 18);

                example.Visible = false;

                example.ForeColor = Color.White;

                if (anchor == true) example.Anchor = (AnchorStyles.Top | AnchorStyles.Right);

                return example;

            }

            Button btnInfo = Buttongenerator(100, 485, "Справка", true);

            btnInfo.Click += BtnInfo\_Click;

            Controls.Add(btnInfo);

            btnGo = Buttongenerator(800, 100, "Нажмите, чтобы начать работу", false);

            btnGo.Click += btnGo\_Click;

            Controls.Add(btnGo);

            Button btnCheck = Buttongenerator(100, 100, "Выгрузить", true);

            btnCheck.Click += btnCheck\_Click;

            Controls.Add(btnCheck);

        }

        private void textboxes()

        {

            TextBox TextBoxgenerator(int x, int y)

            {

                TextBox example = new TextBox();

                example.Location = new Point(x, y);

                example.AutoSize = true;

                example.Visible = false;

                return example;

            }

            tbMin = TextBoxgenerator(btnGo.Left + btnGo.Width / 3, this.Height - 470);

            Controls.Add(tbMin);

            tbMax = TextBoxgenerator(btnGo.Left + 2 \* btnGo.Width / 3, this.Height - 470);

            Controls.Add(tbMax);

        }

        private void lists()

        {

            ListBox ListBoxgenerator(int x, int y)

            {

                ListBox example = new ListBox();

                example.Location = new Point(x, y);

                example.Size = new Size(btnGo.Width / 3, 200);

                example.Visible = false;

                return example;

            }

            lbx = ListBoxgenerator(btnGo.Left + btnGo.Width / 3, 65);

            lbx.SelectedIndexChanged += lbx\_SelectedIndexChanged;

            Controls.Add(lbx);

            lby = ListBoxgenerator(lbx.Right, 65);

            lby.SelectedIndexChanged += lby\_SelectedIndexChanged;

            Controls.Add(lby);

            lbx1 = ListBoxgenerator(lby.Right, 65);

            lbx1.SelectedIndexChanged += lbx1\_SelectedIndexChanged;

            Controls.Add(lbx1);

            lby1 = ListBoxgenerator(lbx1.Right, 65);

            lby1.SelectedIndexChanged += lby1\_SelectedIndexChanged;

            Controls.Add(lby1);

            lbMain = ListBoxgenerator(btnGo.Left, 65);

            lbMain.Items.Add("ArcSin x");

            lbMain.Items.Add("Sqrt x");

            lbMain.Items.Add("Log\_10 x");

            lbMain.Items.Add("Log\_2 ((x+1)/(x-1))");

            lbMain.Items.Add("ArcCos x");

            lbMain.Items.Add("Sin x");

            lbMain.Items.Add("x^2");

            lbMain.Items.Add("Cos x");

            lbMain.Items.Add("ArcTg x");

            lbMain.Items.Add("Log\_e x");

            lbMain.Items.Add("Tan x");

            lbMain.Items.Add("Log\_2 x");

            Controls.Add(lbMain);

        }

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

        {

            lbx.SelectedIndex = lby.SelectedIndex;

        }

        private void lby1\_SelectedIndexChanged(object sender, EventArgs e)

        {

            lbx1.SelectedIndex = lby1.SelectedIndex;

        }

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

        {

            lby.SelectedIndex = lbx.SelectedIndex;

        }

        private void lbx1\_SelectedIndexChanged(object sender, EventArgs e)

        {

            lby1.SelectedIndex = lbx1.SelectedIndex;

        }

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

        {

            string round(string rounding)

            {

                double temp = Double.Parse(rounding);

                temp = Math.Round(temp, 3);

                rounding = temp.ToString();

                return rounding;

            }

            lbx1.Items.Clear();

            lby1.Items.Clear();

            lbx1.Visible = true;

            lby1.Visible = true;

            string text = File.ReadAllText("Data.txt");

            char[] check = {'$', '|'};

            string xy = "$$$";

            foreach (var ch in text)

            {

                string temp = "";

                temp += ch;

                if (xy.Substring(xy.Length -  3, 3).Equals("|||"))

                {

                    xy = xy.Trim(check);

                    lbx1.Items.Add(xy);

                    xy = "$$$";

                }

                if (temp.Equals("\n"))

                {

                    xy = xy.Trim(check);

                    if (xy.IndexOf(",") > -1)

                    {

                        xy = round(xy);

                    }

                    lby1.Items.Add(xy);

                    xy = "$$$";

                }

                xy += ch;

            }

        }

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

        {

                foreach (Control cntrl in this.Controls)

                {

                    cntrl.Visible = true;

                }

                lbx1.Visible = false;

                lby1.Visible = false;

            if (lbMain.SelectedIndex > -1)

            {

                lbx.Items.Clear();

                lby.Items.Clear();

               if (string.IsNullOrEmpty(tbMin.Text) || string.IsNullOrEmpty(tbMax.Text))

                {

                    return;

                }

                StreamWriter sr = new StreamWriter("Data.txt");

                int counter = 0;

                for (int i = Int32.Parse(tbMin.Text); i <= Int32.Parse(tbMax.Text); i += 1)

                {

                    lbx.Items.Add(i);

                    switch (lbMain.SelectedIndex)

                    {

                        case 0:

                            lby.Items.Add(Math.Asin(i));

                            break;

                        case 1:

                            lby.Items.Add(Math.Sqrt(i));

                            break;

                        case 2:

                            lby.Items.Add(Math.Log10(i));

                            break;

                        case 3:

                            if (i == 1) lby.Items.Add("Деление на 0");

                            else

                                lby.Items.Add(Math.Log((i + 1) / (i - 1), 2));

                            break;

                        case 4:

                            lby.Items.Add(Math.Acos(i));

                            break;

                        case 5:

                            lby.Items.Add(Math.Sin(i));

                            break;

                        case 6:

                            lby.Items.Add(i \* i);

                            break;

                        case 7:

                            lby.Items.Add(Math.Cos(i));

                            break;

                        case 8:

                            lby.Items.Add(Math.Atan(i));

                            break;

                        case 9:

                            lby.Items.Add(Math.Log(i));

                            break;

                        case 10:

                            lby.Items.Add(Math.Tan(i));

                            break;

                        case 11:

                            lby.Items.Add(Math.Log(i, 2));

                            break;

                        default:

                            lby.Items.Add(0000000);

                            break;

                    }

                    string x = lbx.Items[counter].ToString();

                    string y = lby.Items[counter].ToString();

                    sr.Write(x);

                    sr.Write("|||");

                    sr.Write(y);

                    sr.Write("\n");

                    counter++;

                }

                sr.Close();

            }

        }

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

        {

            ChildForm Child = new ChildForm();

            Child.Owner = this;

            Child.ShowDialog();

            Child.Dispose();

        }

     }

    public partial class ChildForm : Form

    {

        private TextBox msg;

        public ChildForm()

        {

            msg = new TextBox();

            msg.Location = new Point(0, 0);

            msg.Size = new Size(this.Width, this.Height);

            msg.Multiline = true;

            msg.Font = new Font("Times New Roman", 18);

            msg.ReadOnly = true;

            msg.Text = "Данная программа реализует автоматическое построение значений абсцисы и ординаты функции\n" +

            "на промежутке между двумя задаваемыми значениями с шагом 1. Пожалуйста, введите значение\n" +

            "нижней границы промежутка в левое заполняемое поле, а значение верхней границы промежутка\n" +

            "в правое, после чего выберите одну из доступных функций и нажмите на кнопку (Нажмите, чтобы\n" +

            "начать работу). В результате левый столбец будет заполнен значениями абсциссы а правый\n" +

            "значениями ординаты. Результаты автоматически выгружаются в файл  Data.txt откуда их можно\n" +

            "выгрузить обратно в программу с округлением по нажатию кнопки выгрузить. Результаты будут в отдельной таблице\n";

            Controls.Add(msg);

            Resize += msg\_Resize;

        }

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

        {

            msg.Size = new Size(this.Width, this.Height);

        }

    }

}

1. **Графический пользовательский интерфейс**

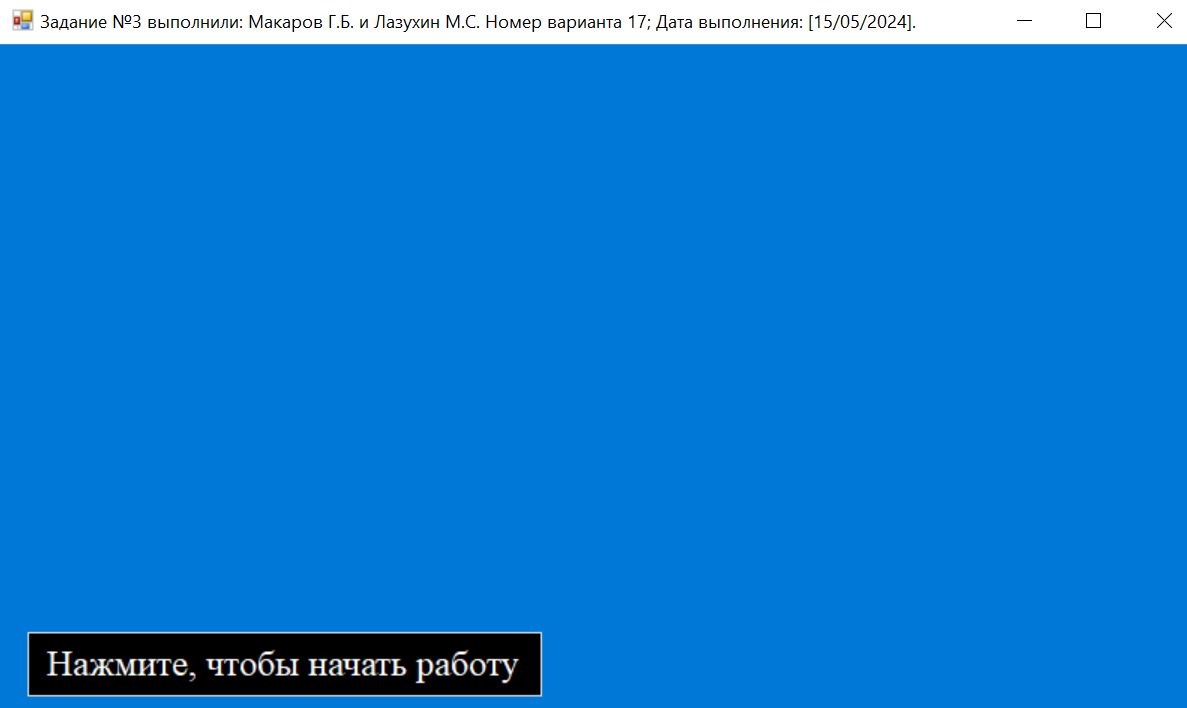
****

Рисунок 2 – ГПИ стартовой страницы

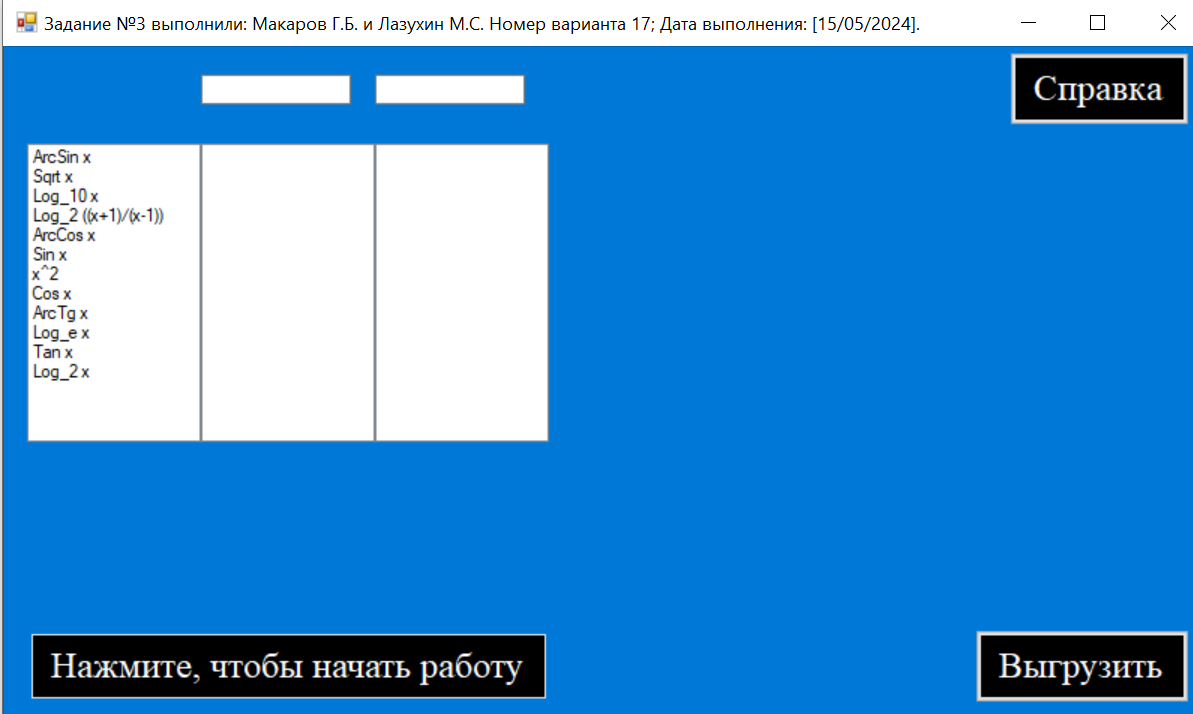
****

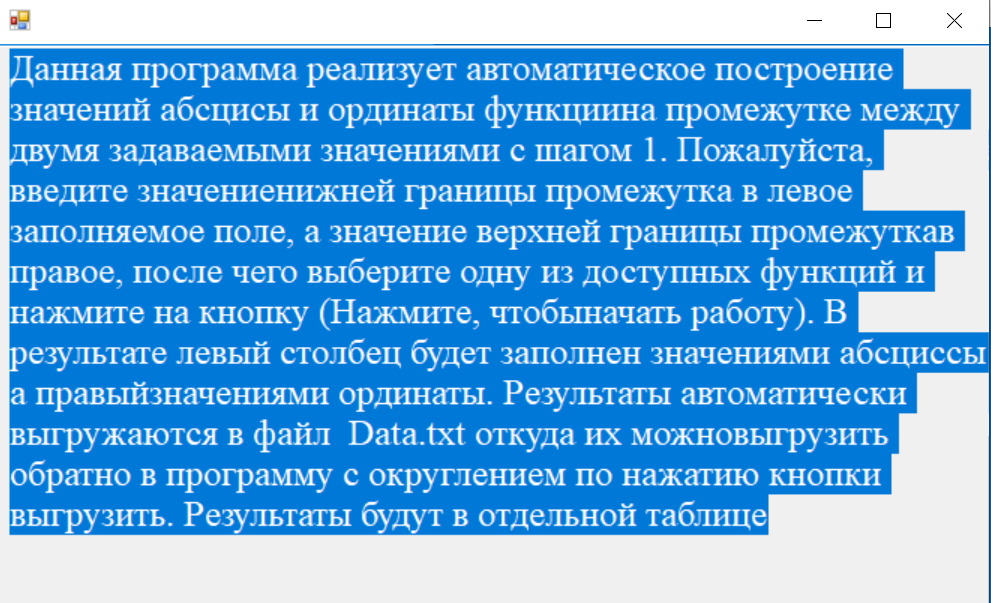
Рисунок 3 – ГПИ основной страницы после начала работы с программой****

Рисунок 4 – Справка, вызываемая нажатием на кнопку «Справка»

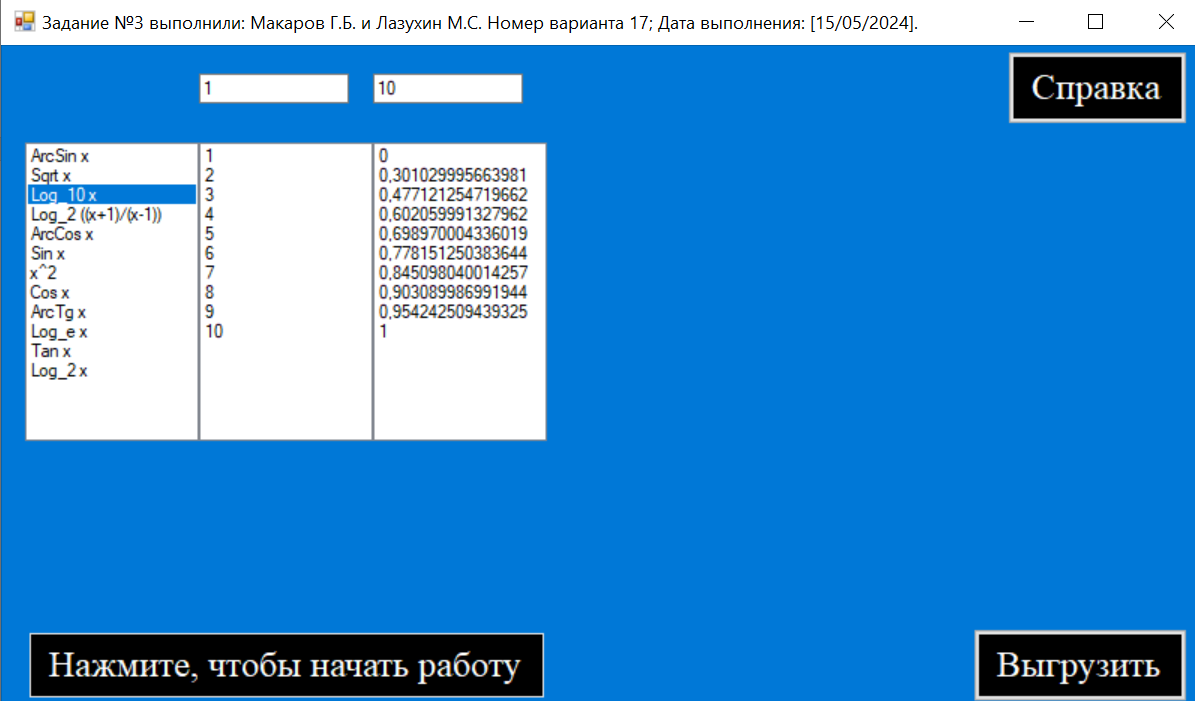


Рисунок 5 – Результат работы программы с указанными функцией и границами

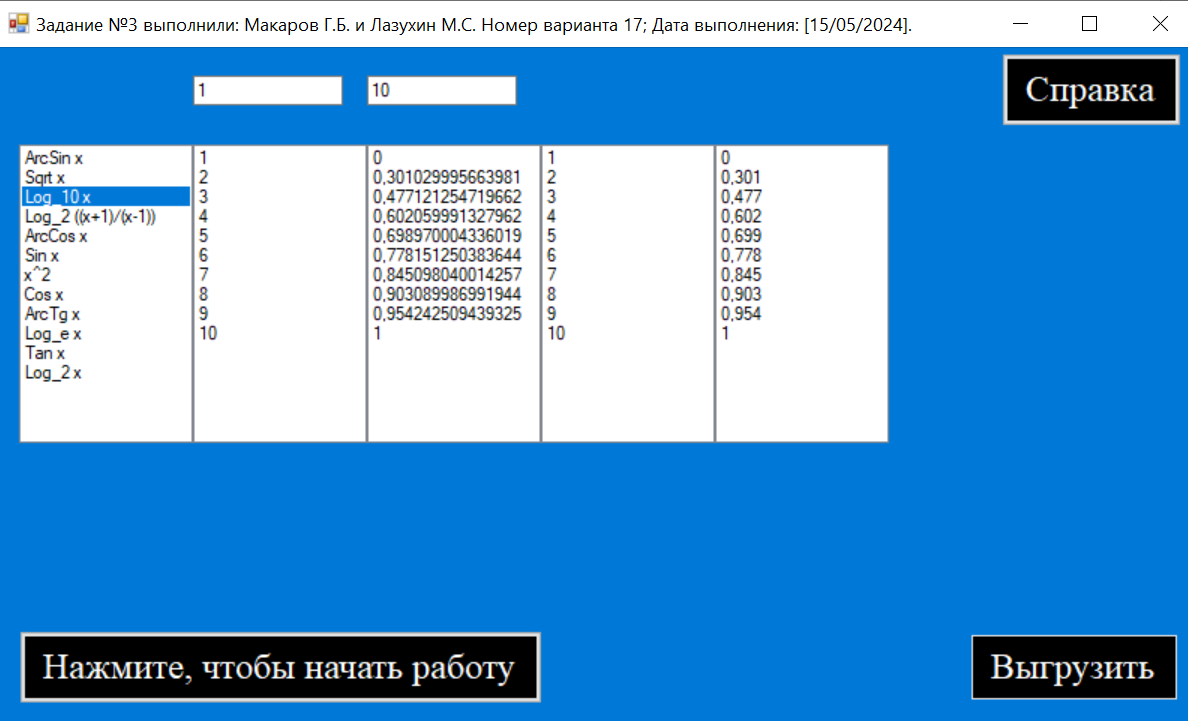


Рисунок 6 – Результат работы программы с указанными функцией и границами после нажатия кнопки Выгрузить

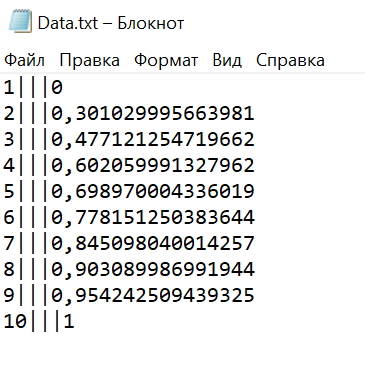


Рисунок 7 – Содержимое файла Data.txt, в который производится запись значений.

1. **Вывод**

В ходе выполнения лабораторной работы мы закрепили навыки разработки визуального пользовательского интерфейса, освоили работу с текстовыми файлами в среде *Microsoft Visual Studio*, научились взаимно увязывать одни элементы управления с другими, а также получили представление о перерисовке и перемасштабировании.