Минобрнауки России

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт

(технический университет)»

Дисциплина «Разработка ПС»

Отчёт по лабораторной работе №3.

Знакомство с WinForms

Разработка программной системы для построения графика функции

Преподаватели: Корниенко Иван Григорьевич

Федин Алексей Константинович

Исполнил студент 405 группы: Вишняков Виталий

Санкт-Петербург

2022

Постановка задачи

Необходимо написать приложение с использованием технологии WinForms для построения графика функции и вывода таблицы значений функции. Пользователь задает правую и левую границу, шаг, коэффициенты . При невозможности построить график функции в заданном интервале пользователю выдается предупреждение об этом с предложением сменить границы построения. Если график функции из-за коэффициентов вырождается в точку или не может быть построен пользователь также видит предупреждение.

В программе должны быть предусмотрены два варианта ввода данных: пользователем с клавиатуры или из файла. В работе должна присутствовать возможность сохранения исходных данных и сохранения результата работы программы, а также модульного тестирования.

Исходные данные

В качестве исходных данных программа использует: пользовательский ввод значений в специальные поля для ввода; текстовые файлы формата «txt», в которых хранится определённое число цифр в строго определённом порядке.

Значения, который ввел пользователь, является числом с плавающей запятой (тип double в C#). Результат представляет собой набор чисел с плавающей запятой (тип double в C#).

Особые ситуации

Необходимо рассмотреть следующие особые ситуации:

а) Невозможность построить график на заданном пользователем интервале
б) Невозможность построить график при заданных коэффициентах
в) Отсутствие ожидаемых программой файлов на чтение или содержание некорректных данных внутри существующих файлов.
г) Запись работы программы в уже существующий файл или создание недопустимого файла, а также проверка атрибутов существующего файла (Атрибут «Только для чтения»).

Математические методы и алгоритмы решения задач

Поставленная задача требует использования некоторых математических методов.

Для построения графика функция была задана математической формулой:

$$y = \pm\sqrt{\sqrt{a^4 + 4c^2x^2} - x^2 - c^2}$$

Рисунок 1 - Формула для графика Овалы Кассини

Форматы представления данных

Формат внешних файлов, из которых производится ввод данных – строго текстовый формат txt. Внутри читаемого txt файла для корректной работы должна содержаться только 7 чисел в строго определённом порядке: левая граница, правая граница, верхняя граница, нижняя граница, шаг, коэффициент C, коэффициент A.

Формат файлов для сохранения результатов работы программы - текстовый «.txt» или в виде таблицы MS Office Excel «.xls».

Структура программы

Программа разбита на 5 классов, также отдельно подключен проект для тестирования.

Основная последовательность работы программы – ожидания решения пользователя. Программа ожидает пользовательские нажатия на доступные в тот или иной момент элементы управления. После ввода корректный данных и нажатии кнопки «Построить график» программа запускает главный алгоритм. После чего результат появляется в специальном поле для отображения графика. Затем пользователь может просмотреть таблицу значений функции, сохранить исходные данные и результат работы. Кнопка «Настройка» позволяет включить или отключить отображение справки перед запуском основной программы. Кнопка «Справка» открывает информацию о программе. Комплекс продолжает свою работу до тех пор, пока его не закроет пользователь в правом верхнем углу или с помощью средств операционной системы.

Таблица 1 – Основные переменные программы

Имя	Тип	Описание
`leftBorder`	`double`	Левая граница графика
`rightBorder`	`double`	Правая граница графика
`topBorder`	`double`	Верхняя граница графика
`bottomBorder`	`double`	Нижняя граница графика
`step`	`double`	Значение шага
`coeffC`	`double`	Коэффициент C
`coeffA`	`double`	Коэффициент A
`x`	`double`	Координата X
`y`	`double`	Координата Y
`valuesX`	`List<double>`	Список координат X функции
`valuesY`	`List<double>`	Список координат Y функции

Таблица 2 – Классы, используемые в программе

Имя	Описание	Методы	Описание
CassiniOval	Построение овалов Кассини	public static double CalculatePointOnTheGraph(double coeffA, double coeffC, double x)	Расчет значения функции в точке x
Table	Отображение таблцы значений функции	private void Table_Load(object sender, EventArgs e)	Построение и заполнение таблицы
		private void ExcelButton_Click(object sender, EventArgs e)	Вывод таблицы в книгу Excel
WorkWithFiles	Сохранение данных	public static void SaveToFile(string fileOutputPath, string text)	Сохранение исходных данных
		public static string MakeResult(string leftBorder, string rightBorder, string topBorder, string bottomBorder, string step, string coeffC, string coeffA, List<double> valuesX, List<double> valuesY) {	Формирование результата
		public static List<decimal> FromFileInput(string fileInputPath)	Считывание данных из файла

Блок-схемы алгоритмов программы

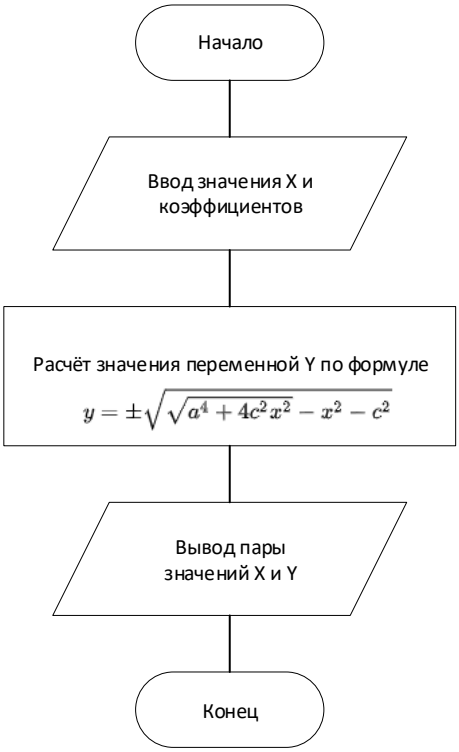

Рисунок 2 - Блок-схема основного алгоритма программы

Описание хода выполнения лабораторной работы

В ходе лабораторной работы было создано решение (Solution) интегрированной среде разработки Microsoft Visual Studio C# 2019. В нём был создан проект.

После написания основного цикла работы программы, были созданы функции пользовательского ввода, чтения данных из файла. Далее были созданы методы класса работы с полученными значениями. После был создан алгоритм построения графика, печати полученных данных на экран и сохранения во внешний файл как исходных данных, так и результатов.

В ходе работы над проектом были учтены и обработаны ошибки ввода некорректных данных, некорректное чтение файлов, а также обработаны возникающие исключения.

Также в код программы были включены модульные тесты, предназначенные для тестирования основного модуля программы. Данные для тестирования берутся из заранее подготовленных файлов. В случае удачного прохождения тестов на экран выводится сообщение об успешном их выполнении, иначе - сообщение о неудаче в ходе тестирования.

Результаты работы программы

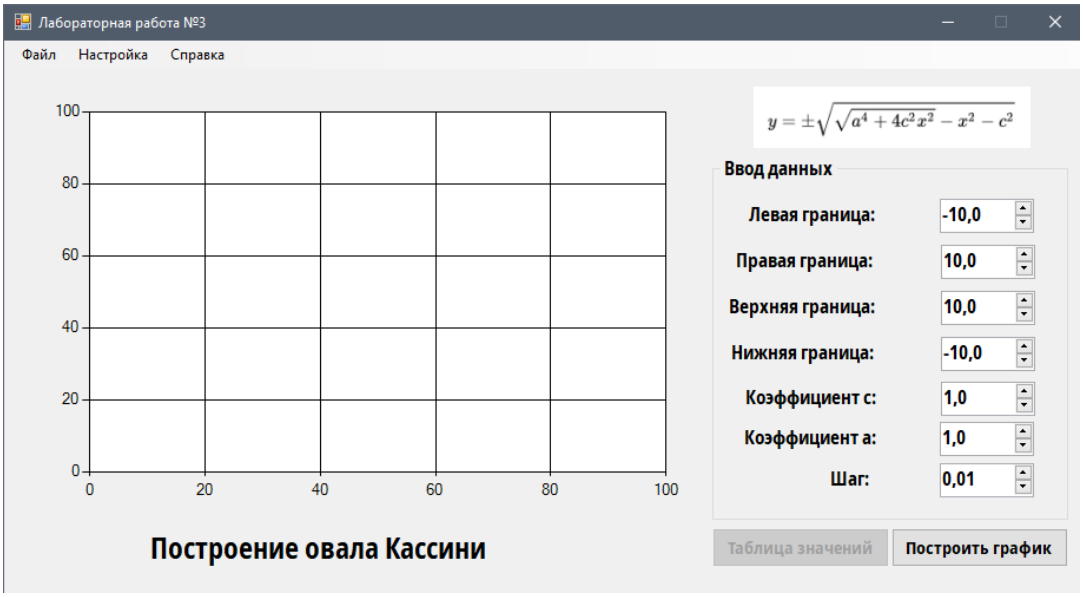

Рисунок 3 – Первый запуск программы

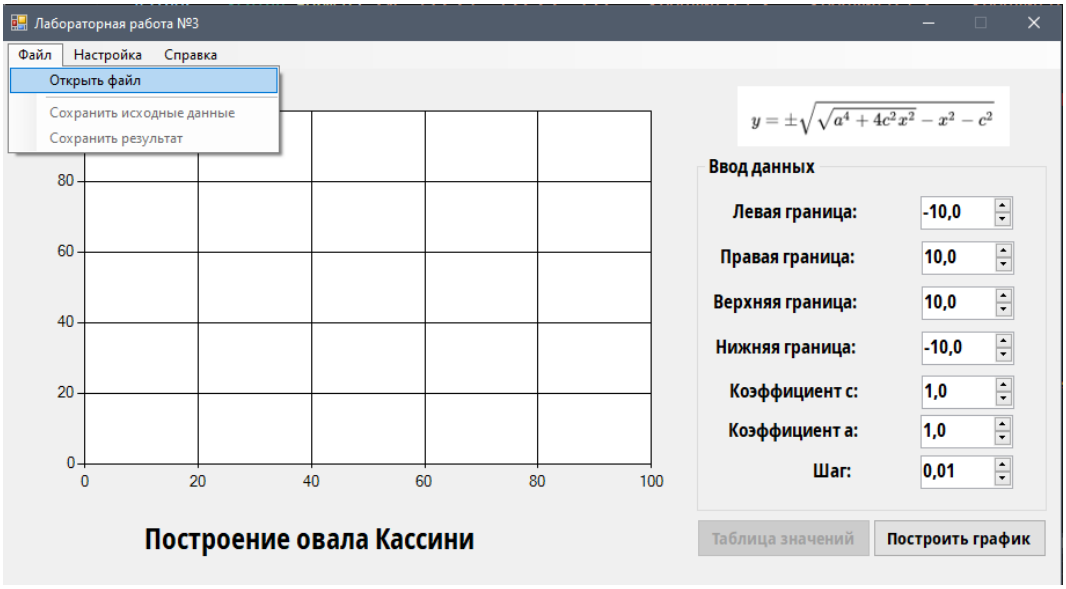

Рисунок 4 – Пример работы программы при вызове меню «Файл»

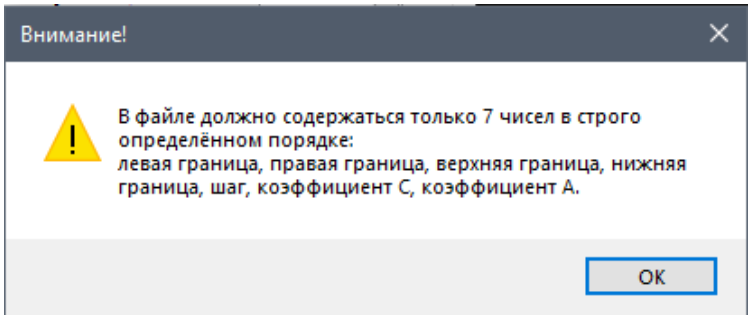

Рисунок 5 – Информация об ожидаемом содержимом открываемого файла

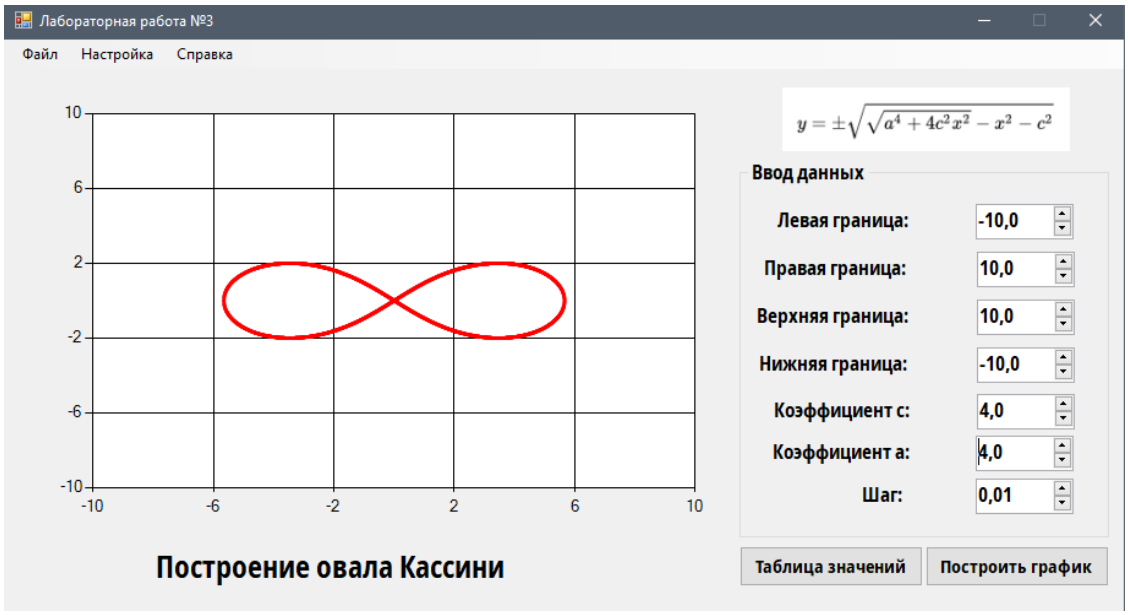

Рисунок 6 – Пример работы программы.

Исходный текст программы

[Начало программы ---]

[Начало Program.cs ---]

```
using System;
using System.Windows.Forms;

namespace Lab3
{
    static class Program
    {

        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainWindow());
        }
    }
```

[Конец Program.cs ---]

[Начало MainWindow.cs ---]

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace Lab3
{
    public partial class MainWindow : Form
    {
        private static double leftBorder; // Левая граница
        private static double rightBorder; // Правая граница
        private static double topBorder; // Верхняя граница
        private static double bottomBorder; // Нижняя граница
        private static double step; // Шаг
        private static double coeffC; // Коэффициент C
        private static double coeffA; // Коэффициент A
        private static double x; // Координата X
        private static double y; // Координата Y
        public static List<double> valuesX = new List<double>(); // Список координат X функции
        public static List<double> valuesY = new List<double>(); // Список координат Y функции

        public MainWindow()
        {
            InitializeComponent();
            saveFileDialog1.Filter = @"Text files(*.txt)|*.txt";
            MaximizeBox = false;
            chartCO.Show();
            chartCO.Series["CassiniOvalPos"].Points.AddXY(0, 0);
            chartCO.Series["CassiniOvalNeg"].Points.AddXY(0, 0);
            chartCO.ChartAreas[0].AxisX.Minimum = 0;
            chartCO.ChartAreas[0].AxisX.Maximum = 100;
            chartCO.ChartAreas[0].AxisY.Minimum = 0;
            chartCO.ChartAreas[0].AxisY.Maximum = 100;
            if (InfoShowing.Default.Show == true)
            {
                InfoToolStripMenuItem_Click(null, null);
                ShowInfoOnStartToolStripMenuItem.Checked = true;
            }
            else ShowInfoOnStartToolStripMenuItem.Checked = false;
        }


        private void CreateChartButton_Click(object sender, EventArgs e)
        {
```

```
            try
            {
                valuesX.Clear();
                valuesY.Clear();
                chartCO.Series["CassiniOvalPos"].Points.Clear();
                chartCO.Series["CassiniOvalNeg"].Points.Clear();

                leftBorder = (double)LeftBorderUpDown.Value;
                rightBorder = (double)RightBorderUpDown.Value;
                topBorder = (double)TopBorderUpDown.Value;
                bottomBorder = (double)BottomBorderUpDown.Value;
                step = (double)ScaleUpDown.Value;
                coeffC = (double)CUpDown.Value;
                coeffA = (double)AUpDown.Value;


                if (topBorder <= bottomBorder || leftBorder >= rightBorder)
                {
                    throw new ArgumentOutOfRangeException();
                }

                if (coeffA == coeffC && coeffC == 0)
                {
                    MessageBox.Show("График вырождается в точку." + Environment.NewLine +
                              "Измените значение коэффициентов.", "Предупреждение!",
                              MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    return;
                }
                chartCO.ChartAreas[0].AxisX.Minimum = leftBorder;
                chartCO.ChartAreas[0].AxisX.Maximum = rightBorder;
                chartCO.ChartAreas[0].AxisY.Minimum = bottomBorder;
                chartCO.ChartAreas[0].AxisY.Maximum = topBorder;


                x = -Math.Sqrt(Math.Pow(coeffC, 2) + Math.Pow(coeffA, 2));
                chartCO.Series["CassiniOvalPos"].Points.AddXY(x, 0);
                chartCO.Series["CassiniOvalNeg"].Points.AddXY(x, 0);
                valuesX.Add(x);
                valuesY.Add(0);

                for (x = -Math.Sqrt(Math.Pow(coeffC, 2) + Math.Pow(coeffA, 2)) + step; x <
Math.Sqrt(Math.Pow(coeffC, 2) + Math.Pow(coeffA, 2)); x += step)
                {
                    y = CassiniOval.CalculatePointOnTheGraph(coeffA, coeffC, x); // Рассчёт
координаты Y
                    // Проверка на построения графика в заданном интервале
                    if (x - step > rightBorder || x - step < leftBorder || y > topBorder || y <
bottomBorder)
                    {
                        throw new IndexOutOfRangeException();
                    }
                    chartCO.Series["CassiniOvalPos"].Points.AddXY(x, y); // Добавление точки на
график
                    chartCO.Series["CassiniOvalNeg"].Points.AddXY(x, -y);
                    valuesX.Add(x); // Добавление точки в таблицу
                    valuesY.Add(y);
                }

                x = Math.Sqrt(Math.Pow(coeffC, 2) + Math.Pow(coeffA, 2));
                chartCO.Series["CassiniOvalPos"].Points.AddXY(x, 0);
                chartCO.Series["CassiniOvalNeg"].Points.AddXY(x, 0);
                valuesX.Add(x);
                valuesY.Add(0);

                TableButton.Enabled = true;
                SaveDataToolStripMenuItem.Enabled = true;
                SaveResultToolStripMenuItem.Enabled = true;

            }
```

```
            catch (IndexOutOfRangeException)
            {
                MessageBox.Show("График не может быть построен при указанных данных." +
Environment.NewLine +
                                "Измените значение коэффициентов, шага или границ.", "Ошибка!",
                                MessageBoxButtons.OK, MessageBoxIcon.Error);
                valuesX.Clear();
                valuesY.Clear();
                chartCO.Series["CassiniOvalPos"].Points.Clear();
                chartCO.Series["CassiniOvalNeg"].Points.Clear();
                TableButton.Enabled = false;
                SaveDataToolStripMenuItem.Enabled = false;
                SaveResultToolStripMenuItem.Enabled = false;

            }
            catch (ArgumentOutOfRangeException)
            {
                MessageBox.Show("Ошибка!" + Environment.NewLine +
                                "Нижняя граница должна быть меньше верхней." + Environment.NewLine +
                                "Левая граница должна быть меньше правой.", "Ошибка!",
                                 MessageBoxButtons.OK, MessageBoxIcon.Error);
                valuesX.Clear();
                valuesY.Clear();
                chartCO.Series["CassiniOvalPos"].Points.Clear();
                chartCO.Series["CassiniOvalNeg"].Points.Clear();
                TableButton.Enabled = false;
                SaveDataToolStripMenuItem.Enabled = false;
                SaveResultToolStripMenuItem.Enabled = false;
            }
            catch (OverflowException)
            {
                MessageBox.Show("Одно из значений было недопустимо малым или недопустимо большим.",
                                "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                valuesX.Clear();
                valuesY.Clear();
                chartCO.Series["CassiniOvalPos"].Points.Clear();
                chartCO.Series["CassiniOvalNeg"].Points.Clear();
                TableButton.Enabled = false;
                SaveDataToolStripMenuItem.Enabled = false;
                SaveResultToolStripMenuItem.Enabled = false;
            }
        }

        private void SaveDataToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (saveFileDialog1.ShowDialog() == DialogResult.Cancel) // Обработка закрытия окна
сохранения введенных данных
                return;
            string fileOutputPath = saveFileDialog1.FileName; // Получение имени файла
            saveFileDialog1.FileName = string.Empty;
            // Формирование результата
            string answer = LeftBorderUpDown.Text + " " +
                            RightBorderUpDown.Text + " " +
                            TopBorderUpDown.Text + " " +
                            BottomBorderUpDown.Text + " " +
                            ScaleUpDown.Text + " " +
                            CUpDown.Text + " " +
                            AUpDown.Text;
            // Сохранение результата
            WorkWithFiles.SaveToFile(fileOutputPath, answer);
        }


        private void AUpDown_ValueChanged(object sender, EventArgs e)
        {
            CreateChartButton_Click(null, null);
        }

        private void CUpDown_ValueChanged(object sender, EventArgs e)
        {
            CreateChartButton_Click(null, null);
```

```
        }

        private void ScaleUpDown_ValueChanged(object sender, EventArgs e)
        {
            CreateChartButton_Click(null, null);
        }

        private void TableButton_Click(object sender, EventArgs e)
        {
            var table = new Table(valuesX, valuesY, this);
            table.Show();
        }

        private void OpenFileToolStripMenuItem_Click(object sender, EventArgs e)
        {
            try
            {
                MessageBox.Show("В файле должно содержаться только 7 чисел в строго определённом
порядке:" + Environment.NewLine +
                                "левая граница, правая граница, верхняя граница, нижняя граница,
шаг, коэффициент C, коэффициент A." + Environment.NewLine, "Внимание!",
                                MessageBoxButtons.OK, MessageBoxIcon.Warning);
                if (openFileDialog1.ShowDialog() == DialogResult.Cancel)
                    return;
                string fileInputPath = openFileDialog1.FileName;
                List<decimal> initialData = WorkWithFiles.FromFileInput(fileInputPath);
                openFileDialog1.FileName = string.Empty;
                if (initialData.Count > 7 || initialData.Count < 7)
                {
                    throw new ArgumentOutOfRangeException();
                }
                LeftBorderUpDown.Value = initialData[0];
                RightBorderUpDown.Value = initialData[1];
                TopBorderUpDown.Value = initialData[2];
                BottomBorderUpDown.Value = initialData[3];
                ScaleUpDown.Value = initialData[4];
                CUpDown.Value = initialData[5];
                AUpDown.Value = initialData[6];


            }
            catch (FormatException)
            {
                MessageBox.Show("Файл содержит некорректные данные.\n" +
                                "Файл не должен содержать букв и спец. символов.", "Ошибка!",
                                MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            catch (ArgumentOutOfRangeException)
            {
                MessageBox.Show("В файле недостаточно данных или файл содержит больше данных, чем
нужно.", "Ошибка!",
                                MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void SaveResultToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (saveFileDialog1.ShowDialog() == DialogResult.Cancel)
                return;
            string fileOutputPath = saveFileDialog1.FileName;
            saveFileDialog1.FileName = string.Empty;
            string answer = WorkWithFiles.MakeResult(LeftBorderUpDown.Text, RightBorderUpDown.Text,
TopBorderUpDown.Text, BottomBorderUpDown.Text, ScaleUpDown.Text, CUpDown.Text, AUpDown.Text,
valuesX, valuesY);
            WorkWithFiles.SaveToFile(fileOutputPath, answer);
        }



        private void ShowInfoOnStartToolStripMenuItem_Click(object sender, EventArgs e)
        {
```

```
            if (ShowInfoOnStartToolStripMenuItem.Checked)
            {
                ShowInfoOnStartToolStripMenuItem.Checked = false;
                InfoShowing.Default.Show = false;
                InfoShowing.Default.Save();
            }
            else
            {
                ShowInfoOnStartToolStripMenuItem.Checked = true;
                InfoShowing.Default.Show = true;
                InfoShowing.Default.Save();
            }
        }
    }
}
```

[Конец MainWindow.cs ---]

[Начало CassiniOval.cs ---]

```
using System;

namespace Lab3
{
    public static class CassiniOval
    {
        public static double CalculatePointOnTheGraph(double coeffA, double coeffC, double x)
        {
            return Math.Sqrt(Math.Sqrt(Math.Pow(coeffA, 4) + (4 * Math.Pow(x, 2) * Math.Pow(coeffC,
2))) - Math.Pow(x, 2) - Math.Pow(coeffC, 2));
        }
    }
```

[Конец CassiniOval.cs ---]

[Начало Table.cs ---]

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace Lab3
{

    public partial class Table : Form
    {
        public int rowCount;
        public static List<double> tempValuesX = new List<double> { };
        public static List<double> tempValuesY = new List<double> { };
        decimal left;
        decimal right;
        decimal top;
        decimal bottom;
        decimal coeffC;
        decimal coeffA;
        decimal step;

        public Table(List<double> valuesX, List<double> valuesY, MainWindow textBoxes)
        {
            left = textBoxes.LeftBorderUpDown.Value;
            right = textBoxes.RightBorderUpDown.Value;
            top = textBoxes.TopBorderUpDown.Value;
            bottom = textBoxes.BottomBorderUpDown.Value;
            coeffC = textBoxes.CUpDown.Value;
```

```
            coeffA = textBoxes.AUpDown.Value;
            step = textBoxes.ScaleUpDown.Value;

            tempValuesX.Clear();
            tempValuesY.Clear();
            rowCount = valuesX.Count;
            for (int i = 0; i < valuesX.Count; i++)
            {
                tempValuesX.Add(valuesX[i]);
                tempValuesY.Add(valuesY[i]);
            }
            InitializeComponent();
        }

        private void Table_Load(object sender, EventArgs e)
        {
            TableFunc.Rows.Clear();
            TableFunc.RowCount = rowCount;
            for (int i = 0; i < tempValuesX.Count; i++)
            {
                TableFunc[0, i].Value = Math.Round(tempValuesX[i], 2);
                TableFunc[1, i].Value = Math.Round(tempValuesY[i], 2);
                TableFunc[2, i].Value = -Math.Round(tempValuesY[i], 2);
            }
        }

        private void ExcelButton_Click(object sender, EventArgs e)
        {
            Microsoft.Office.Interop.Excel.Application ExcelApp = new
Microsoft.Office.Interop.Excel.Application();
            Microsoft.Office.Interop.Excel.Workbook ExcelWorkBook;
            Microsoft.Office.Interop.Excel.Worksheet ExcelWorkSheet;
            //Книга
            ExcelWorkBook = ExcelApp.Workbooks.Add(System.Reflection.Missing.Value);
            //Таблица
            ExcelWorkSheet =
(Microsoft.Office.Interop.Excel.Worksheet)ExcelWorkBook.Worksheets.get_Item(1);

            ExcelApp.Cells[1, 1] = "Результат построения Овалов Кассини";
            ExcelApp.Cells[2, 1] = "Левая граница: "; ExcelApp.Cells[2, 2] = left;
            ExcelApp.Cells[3, 1] = "Правая граница: "; ExcelApp.Cells[3, 2] = right;
            ExcelApp.Cells[4, 1] = "Верхняя граница: "; ExcelApp.Cells[4, 2] = top;
            ExcelApp.Cells[5, 1] = "Нижняя граница: "; ExcelApp.Cells[5, 2] = bottom;
            ExcelApp.Cells[6, 1] = "Коэффициент C: "; ExcelApp.Cells[6, 2] = coeffC;
            ExcelApp.Cells[7, 1] = "Коэффициент A: "; ExcelApp.Cells[7, 2] = coeffA;
            ExcelApp.Cells[8, 1] = "Шаг: "; ExcelApp.Cells[8, 2] = step;
            ExcelApp.Cells[9, 1] = "Таблица значений:";
            ExcelApp.Cells[10, 1] = "X"; ExcelApp.Cells[10, 2] = "Y"; ExcelApp.Cells[10, 3] = "-Y";
            for (int i = 0; i < TableFunc.Rows.Count; i++)
            {
                for (int j = 0; j < TableFunc.ColumnCount; j++)
                {
                    ExcelApp.Cells[i + 11, j + 1] = TableFunc.Rows[i].Cells[j].Value;
                }
            }
            ExcelApp.Visible = true;
            ExcelApp.UserControl = true;
        }
    }

}
```

[Конец Table.cs ---]

[Начало WorkWithFiles.cs ---]

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Lab3
{
    class WorkWithFiles
```

```
    {
        public static void SaveToFile(string fileOutputPath, string text)
        {
            System.IO.File.WriteAllText(fileOutputPath, text);
        }

        public static string MakeResult(string leftBorder, string rightBorder, string topBorder,
string bottomBorder, string step, string coeffC, string coeffA, List<double> valuesX, List<double>
valuesY)
        {
            string values;
            string answer = "Левая граница: " + leftBorder + Environment.NewLine +
                            "Правая граница: " + rightBorder + Environment.NewLine +
                            "Верхняя граница: " + topBorder + Environment.NewLine +
                            "Нижняя граница: " + bottomBorder + Environment.NewLine +
                            "Шаг: " + step + Environment.NewLine +
                            "Коэффициент C: " + coeffC + Environment.NewLine +
                            "Коэффициент A: " + coeffA + Environment.NewLine + Environment.NewLine +
                            "Таблица значений." + Environment.NewLine + Environment.NewLine;
            values = String.Format("{0, 12} {1, 15} {2, 15}", "Координата X", "Координата Y",
"Координата -Y");
            for (int i = 0; i < valuesX.Count; i++)
            {
                values += String.Format("\n{0, 9} {1, 16} {2, 15}",
                                        Math.Round(valuesX[i], 2), Math.Round(valuesY[i], 2), -
Math.Round(valuesY[i], 2));

            }
            answer += values;
            return answer;
        }
        public static List<decimal> FromFileInput(string fileInputPath)
        {
            string arrStr = null;
            arrStr = System.IO.File.ReadAllText(fileInputPath);
            string[] stringSeparators = { "\r", "\n", " ", "\t" };
            List<decimal> tempList = arrStr.Split(stringSeparators,
StringSplitOptions.RemoveEmptyEntries)
                                    .Select(n => decimal.Parse(n))
                                    .ToList();
            fileInputPath = string.Empty;
            return tempList;
        }
    }
}
```

[Конец WorkWithFiles.cs ---]

[Начало UnitTest.cs ---]

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using Lab3;

namespace UnitTestProject
{
    [TestClass]
    public class UnitTest
    {
        [TestMethod]
        public void CalculatePointOnTheGraph1()
        {

            double coefficientC = 50.0;
            double coefficientA = 50.0;
            double x = 30.0;
            double expectedY = 22.47;

            double resultY = CassiniOval.CalculatePointOnTheGraph(coefficientC, coefficientA, x);
            resultY = Math.Round(resultY, 2);

            Assert.AreEqual(expectedY, resultY);
```

```
        }
        [TestMethod]
        public void CalculatePointOnTheGraph2()
        {

            double coefficientC = 70.0;
            double coefficientA = 70.0;
            double x = 60.0;
            double expectedY = 35.0;

            double resultY = CassiniOval.CalculatePointOnTheGraph(coefficientC, coefficientA, x);
            resultY = Math.Round(resultY, 2);

            Assert.AreEqual(expectedY, resultY);
        }
        [TestMethod]
        public void CalculatePointOnTheGraph3()
        {

            double coefficientC = 90.0;
            double coefficientA = 90.0;
            double x = 90.0;
            double expectedY = 43.73;

            double resultY = CassiniOval.CalculatePointOnTheGraph(coefficientC, coefficientA, x);
            resultY = Math.Round(resultY, 2);

            Assert.AreEqual(expectedY, resultY);
        }
        [TestMethod]
        public void CalculatePointOnTheGraph4()
        {

            double coefficientC = 60.0;
            double coefficientA = 60.0;
            double x = 30.0;
            double expectedY = 24.31;

            double resultY = CassiniOval.CalculatePointOnTheGraph(coefficientC, coefficientA, x);
            resultY = Math.Round(resultY, 2);

            Assert.AreEqual(expectedY, resultY);
        }
        [TestMethod]
        public void CalculatePointOnTheGraph5()
        {

            double coefficientC = 80.0;
            double coefficientA = 80.0;
            double x = 20.0;
            double expectedY = 18.85;

            double resultY = CassiniOval.CalculatePointOnTheGraph(coefficientC, coefficientA, x);
            resultY = Math.Round(resultY, 2);

            Assert.AreEqual(expectedY, resultY);
        }
    }
}
```

[Конец UnitTest.cs ---]

[Конец программы ---]