МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ордена Трудового Красного Знамени
федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра «Информатика»

Лабораторные работы

по дисциплине «Алгоритмы и алгоритмические языки»

Вариант №21

Выполнил: студент группы БФИ №2202

Сидорук Д. В.

Принял: старший преподаватель Загвоздкина А. В.

Москва, 2023 г.

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ордена Трудового Красного Знамени
федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра «Информатика»

ОТЧЕТ

по дисциплине «Алгоритмы и алгоритмические языки»

Лабораторная работа № 1

Программирование арифметических выражений на языке Visual C# с использованием методов.

Выполнил: студент группы БФИ №2202

Сидорук Д. В.

Принял: старший преподаватель Загвоздкина А. В.

Москва, 2023 г.

РАЗРАБОТКА ПРОГРАММЫ ДЛЯ ВЫЧИСЛЕНИЯ АРИФМЕТИЧЕСКОГО ВЫРАЖЕНИЯ

Разработать программу для расчета арифметического выражения

$$z=\left(1+\frac{1}{x^2}\right)^x-12x^2y$$

Для того, чтобы выполнить задание, необходимо разработать следующие методы, которые должны быть размещены в внешней dll-библиотеке:

1. Метод public static double GetDouble(TextBox t), предназначенный для получения текста из текстового поля ввода и преобразования его в значение с плавающей точкой.

2. Метод public static double OutputDouble(TextBox t, double value), предназначенный для конвертации переданного значения в текст и установки его в качестве текста переданного текстового поля.

3. Метод public static double Evaluate(double x, double y), предназначенный для вычисления значения арифметического вычисления с возвратом результата по значению.

4. Метод public static double Evaluate(double x, double y, out double z), предназначенный для вычисления значения арифметического вычисления с возвратом результата через аргумент с out модификатором.

5. Метод public static double ReferenceEvaluate(double x, double y, ref double z), предназначенный для вычисления значения арифметического вычисления с возвратом результата через аргумент с ref модификатором.

Перечень блок-схем

На рисунке ниже приведена блок-схема алгоритма метода double GetDouble(TextBox t), предназначенного для получения текста из текстового поля ввода и преобразования его в значение с плавающей точкой.

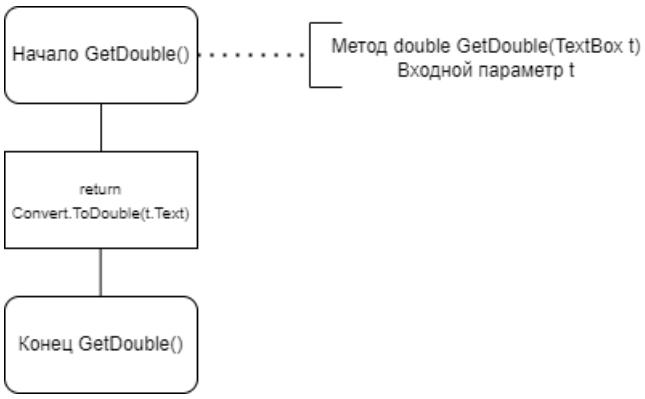

Рисунок 1 — блок-схема алгоритма метода double GetDouble(TextBox t), предназначенного для получения текста из текстового поля ввода и преобразования его в значение с плавающей точкой.

На рисунке ниже приведена блок-схема алгоритма метода void OutputDouble(TextBox t, double value), предназначенного для конвертации переданного значения в текст и установки его в качестве текста переданного текстового поля.

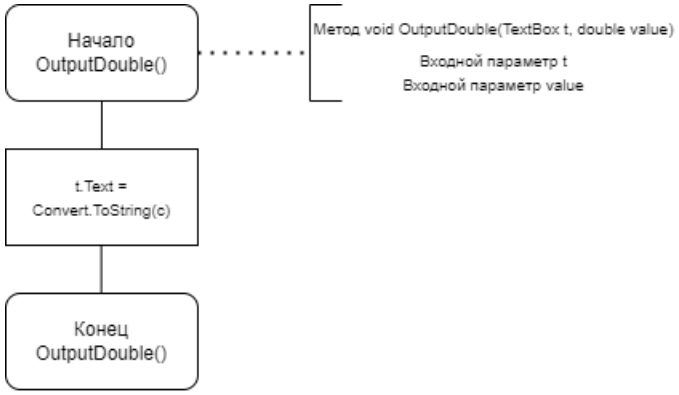

Рисунок 2 — блок-схема алгоритма метода double GetDouble(TextBox t), предназначенного для конвертации переданного значения в текст и установки его в качестве текста переданного текстового поля.

На рисунке ниже представлена блок-схема алгоритма метода double Evaluate(double x, double y), предназначенного для вычисления значения арифметического вычисления с возвратом результата по значению.

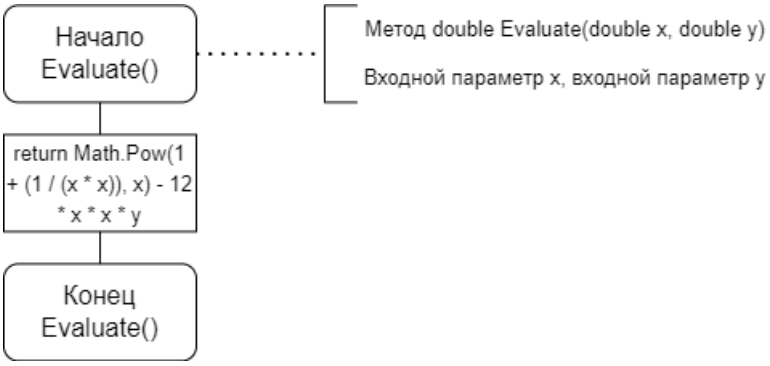

Рисунок 3 — блок-схема алгоритма метода double Evaluate(double x, double y), предназначенного для вычисления значения арифметического вычисления с возвратом результата по значению.

На рисунке ниже представлена блок-схема алгоритма метода double Evaluate(double x, double y, out double z), предназначенного для вычисления значения арифметического вычисления с возвратом результата через аргумент с out модификатором.

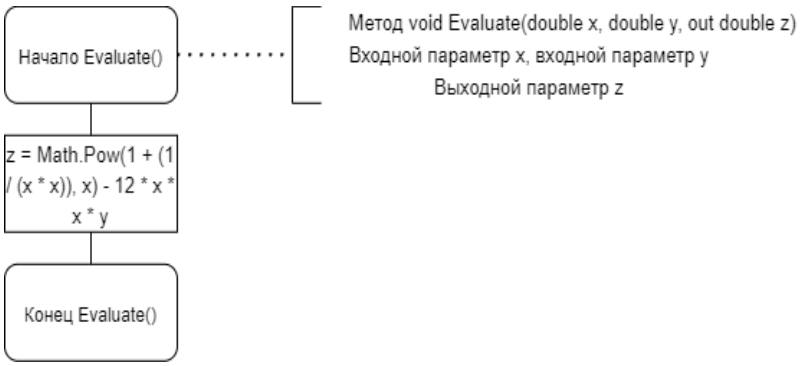

Рисунок 4 — блок-схема алгоритма метода double Evaluate(double x, double y, out double z), предназначенного для вычисления значения арифметического вычисления с возвратом результата через аргумент с out модификатором.

На рисунке ниже представлена блок-схема алгоритма метода double ReferenceEvaluate(double x, double y, ref double z), предназначенного для вычисления значения арифметического вычисления с возвратом результата через аргумент с ref модификатором.

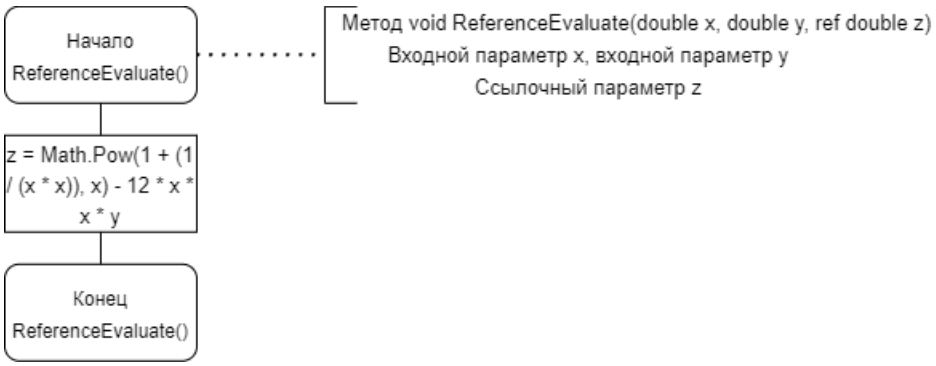

Рисунок 5 — блок-схема алгоритма метода double ReferenceEvaluate(double x, double y, ref double z), предназначенного для вычисления значения арифметического вычисления с возвратом результата через аргумент с ref модификатором.

На рисунке ниже представлена блок-схема алгоритма событийной кнопки.

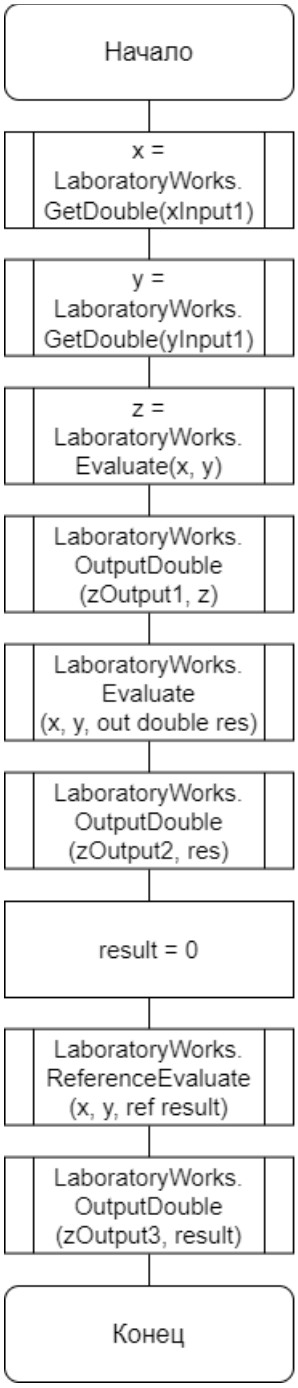

Рисунок 6 — блок-схема алгоритма событийной кнопки.

Содержание DLL-библиотеки

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

namespace LaboratoryWorksLibrary
{
    public class LaboratoryWorks
    {
        /* Все лабораторные работы */
        public static double GetDouble(TextBox t)
        {
            return Convert.ToDouble(t.Text);
        }
        public static void OutputDouble(TextBox t, double value)
        {
            t.Text = Convert.ToString(value);
        }

        /* Первая лабораторная работа */
        public static double Evaluate(double x, double y)
        {
            return Math.Pow(1 + (1 / (x * x)), x) - 12 * x * x * y;
        }
        public static void Evaluate(double x, double y, out double z)
        {
            z = Math.Pow(1 + (1 / (x * x)), x) - 12 * x * x * y;
        }
        public static void ReferenceEvaluate(double x, double y, ref double z)
        {
            z = Math.Pow(1 + (1 / (x * x)), x) - 12 * x * x * y;
        }
    }
}
```

Содержание основной части программы

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

using LaboratoryWorksLibrary;

namespace LaboratoryWorksGUI
{
    public partial class Laboratory1Form : Form
    {
        public Laboratory1Form()
        {
            InitializeComponent();
        }

        private void evaluateButton_Click(object sender, EventArgs e)
        {
            double x = LaboratoryWorks.GetDouble(xInput);
            double y = LaboratoryWorks.GetDouble(yInput);

            double z = LaboratoryWorks.Evaluate(x, y);
            LaboratoryWorks.OutputDouble(zOutput1, z);

            LaboratoryWorks.Evaluate(x, y, out double res);
            LaboratoryWorks.OutputDouble(zOutput2, res);

            double result = 0;
            LaboratoryWorks.ReferenceEvaluate(x, y, ref result);
            LaboratoryWorks.OutputDouble(zOutput3, result);
        }
    }
}
```

Результаты выполнения программы

На рисунке ниже приведен результат выполнения программы при входных данных x = 1,005; y = 3,01

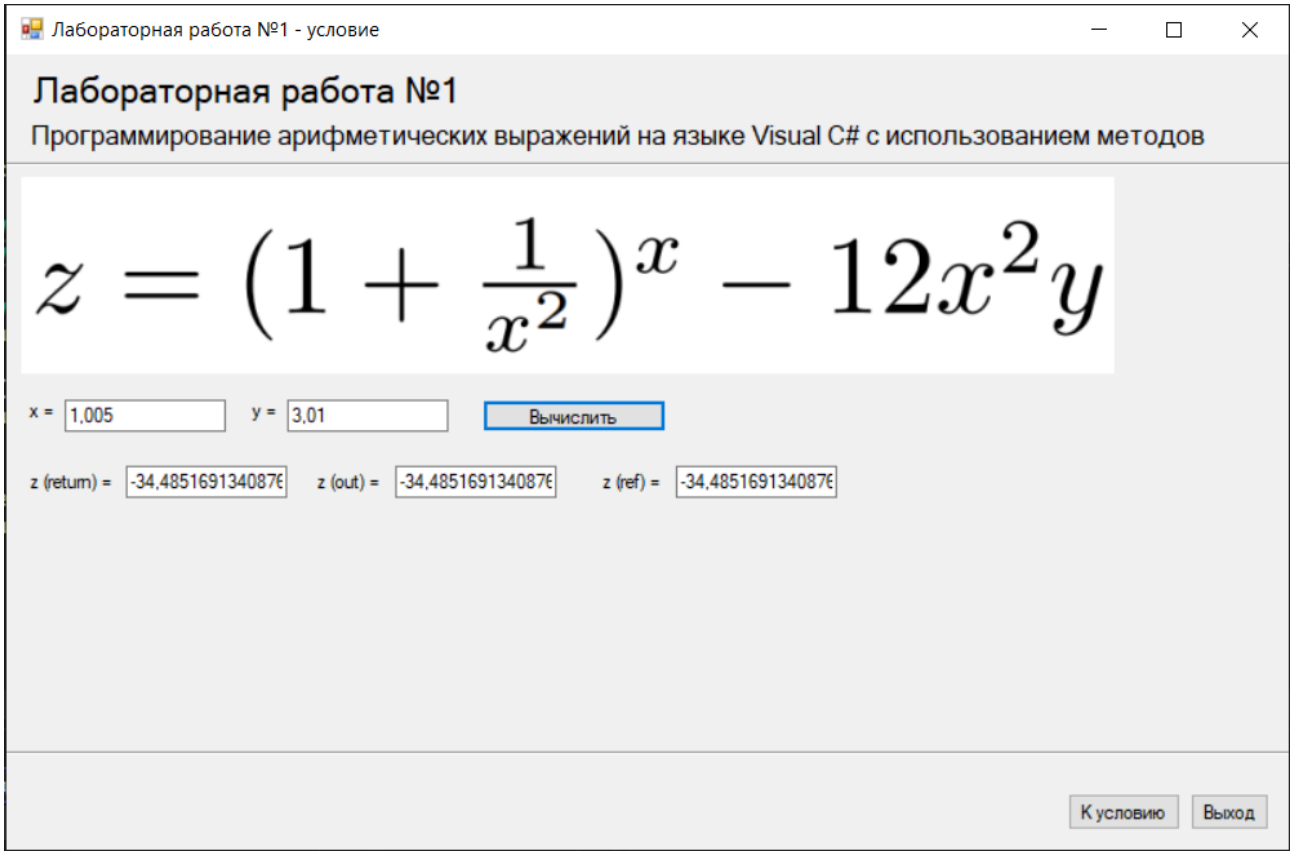

Рисунок 7 — результат выполнения программы при входных данных x = 1,005; y = 3,01

Проверим правильность работы программы:

$$x=1.005,\ y=3.01,\ z=\left(1+\frac{1}{(1.005)^2}\right)^{1.005}-12*(1.005)^2*3.01=\left(1+\frac{1}{1.010025}\right)^{1.005}-12*1.010025*3.01\approx-34.485$$

Видим, что результат был посчитан правильно во всех трех случаях.

Список используемых источников

1. Гуриков С. Р. Введение в программирование на языке Visual C#: учебное пособие / С. Р. Гуриков. — Москва : ФОРУМ : ИНФРА-М, 2020. — 447 с.

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ордена Трудового Красного Знамени
федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра «Информатика»

ОТЧЕТ

по дисциплине «Алгоритмы и алгоритмические языки»

Лабораторная работа № 2

Программирование алгоритмов разветвляющихся структур

Выполнил: студент группы БФИ №2202

Сидорук Д. В.

Принял: старший преподаватель Загвоздкина А. В.

Москва, 2023 г.

Задание

Разработать программу для расчета функции

$$c = \begin{cases} min\{max\{x^2 + y^2; a^x + y^a\}; \sqrt{x} + \sqrt{a^x}\} & x > a; y > 0 \\ \frac{min\{a+b; x^{a+b}\}}{1+a+b^a} & x \leq a; 0 \leq y \leq 3x \\ \mathrm{tg}^2 |x + y| & \text{в противном случае} \end{cases}$$

Для того, чтобы выполнить задание, необходимо разработать следующие методы:

1. Метод public static double GetDouble(TextBox t), предназначенный для получения текста из текстового поля ввода и преобразования его в значение с плавающей точкой.

2. Метод public static void OutputDouble(TextBox t, double value), предназначенный для конвертации переданного значения в текст и установки его в качестве текста переданного текстового поля.

3. Метод public static double Evaluate(double a, double b, double x, double y), предназначенный для расчета данной функции.

Вышеперечисленные методы должны быть размещены в dll-библиотеке.

Перечень блок-схем

На рисунке ниже приведена блок-схема алгоритма метода double GetDouble(TextBox t), предназначенного для получения текста из текстового поля ввода и преобразования его в значение с плавающей точкой.

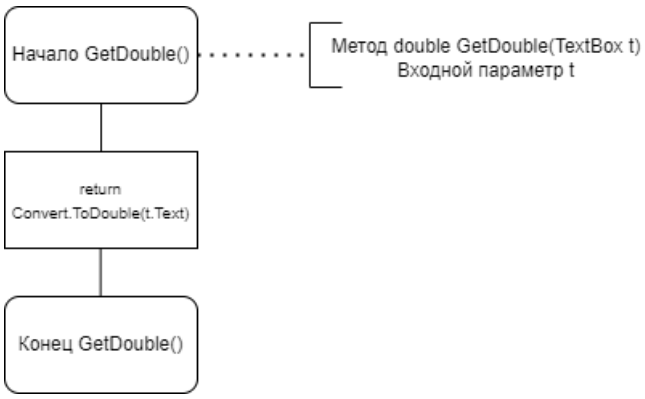

Рисунок 1 — блок-схема алгоритма метода double GetDouble(TextBox t), предназначенного для получения текста из текстового поля ввода и преобразования его в значение с плавающей точкой.

На рисунке ниже приведена блок-схема метода void OutputDouble(TextBox t, double value), предназначенного для конвертации переданного значения в текст и установки его в качестве текста переданного текстового поля.

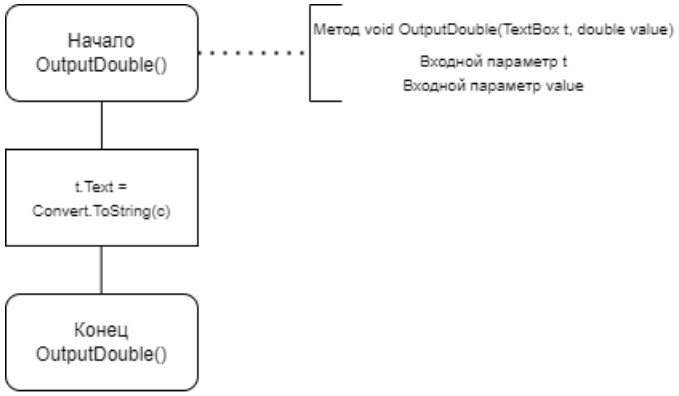

Рисунок 2 — блок-схема алгоритма метода void OutputDouble(TextBox t, double value), предназначенного для конвертации переданного значения в текст и установки его в качестве текста переданного текстового поля.

На рисунке ниже приведена блок-схема алгоритма метода double Evaluate(double a, double b, double x, double y) предназначенного для расчета данной функции.

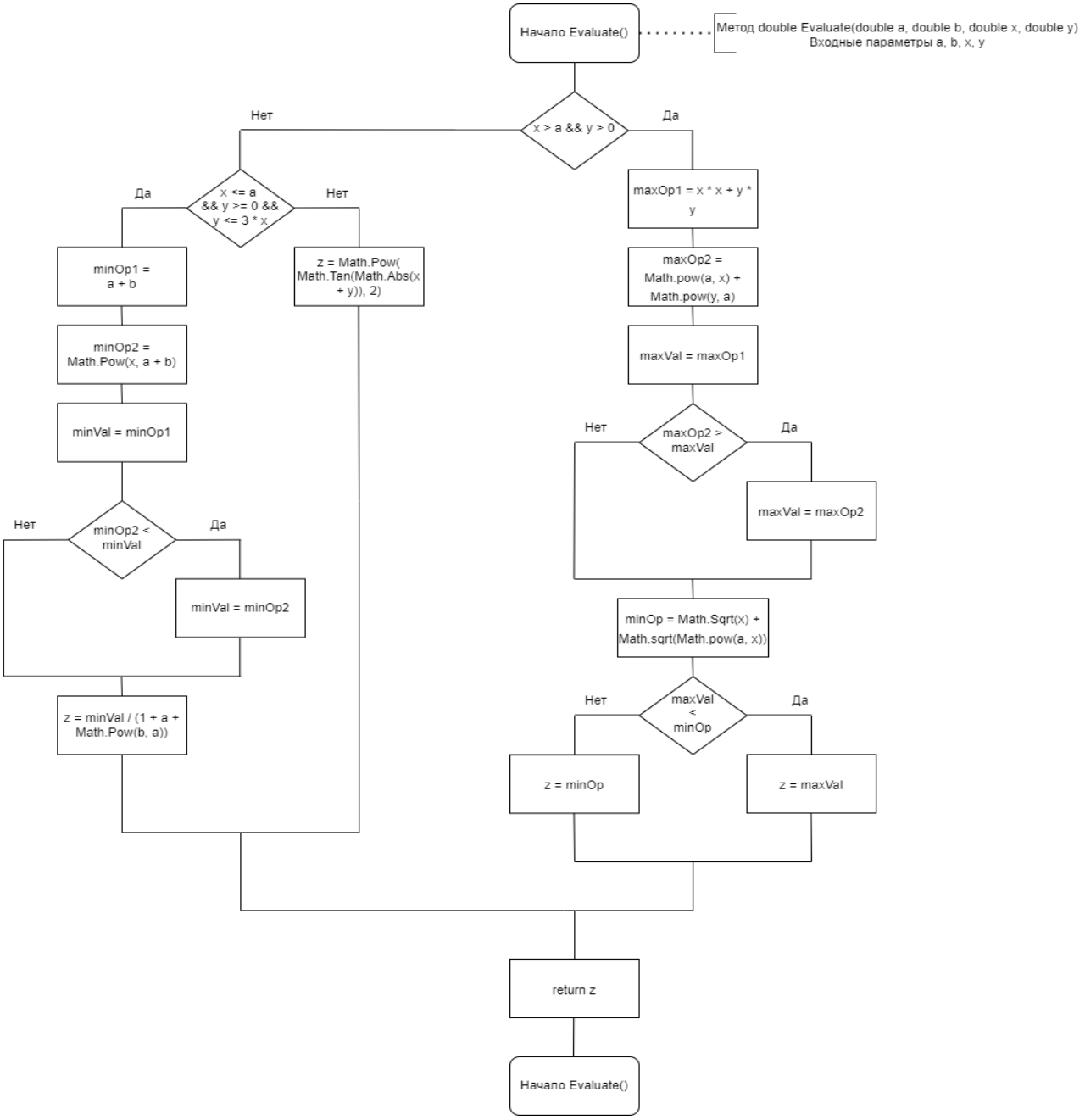

Рисунок 3 — блок-схема алгоритма метода double Evaluate(double a, double b, double x, double y), предназначенного для расчета данной функции.

Алгоритм событийной кнопки

На рисунке ниже приведена блок-схема алгоритма событийной кнопки.

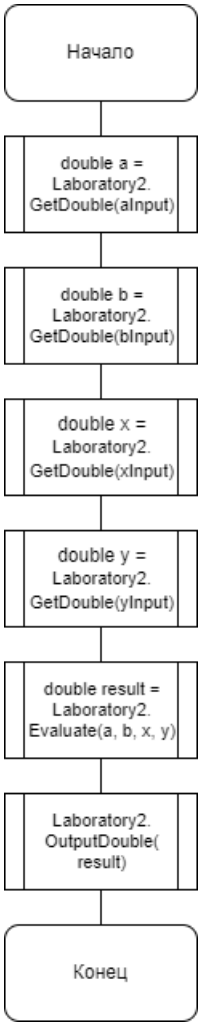

Рисунок 4 — блок-схема алгоритма событийной кнопки.

Содержание DLL-библиотеки

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

namespace Laboratory2Library
{
    public class Laboratory2
    {
        public static double GetDouble(TextBox t)
        {
            return Convert.ToDouble(t.Text);
        }
        public static void OutputDouble(TextBox t, double value)
        {
            t.Text = Convert.ToString(value);
        }
        public static double Evaluate(double a, double b, double x, double y)
        {
            double z;
            if (x > a && y > 0)
            {
                double maxOp1 = x * x + y * y;
                double maxOp2 = Math.Pow(a, x) + Math.Pow(y, a);

                double maxVal = maxOp1;
                if (maxOp2 > maxVal) maxVal = maxOp2;

                double minOp = Math.Sqrt(x) + Math.Sqrt(Math.Pow(a, x));
                if (maxVal < minOp) z = maxVal;
                else z = minOp;
            } else if (x <= a && y >= 0 && y <= 3 * x)
            {
                double minOp1 = a + b;
                double minOp2 = Math.Pow(x, a + b);

                double minVal = minOp1;
                if (minOp2 < minVal) minVal = minOp2;

                z = minVal / (1 + a + Math.Pow(b, a));
            } else z = Math.Pow(Math.Tan(Math.Abs(x + y)), 2);
            return z;
        }
    }
}
```

Содержание основной части программы

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

using Laboratory2Library;

namespace LaboratoryWorks
{
    public partial class Laboratory2Form : Form
    {
        public Laboratory2Form()
        {
            InitializeComponent();
        }

        private void evaluateButton_Click(object sender, EventArgs e)
        {
            double a = Laboratory2.GetDouble(aInput);
            double b = Laboratory2.GetDouble(bInput);
            double x = Laboratory2.GetDouble(xInput);
            double y = Laboratory2.GetDouble(yInput);

            double result = Laboratory2.Evaluate(a, b, x, y);
            Laboratory2.OutputDouble(cOutput, result);
        }
    }
}
```

Результаты выполнения работы

На рисунке ниже приведен результат выполнения программы при входных данных a = 1; b = 1; x = 2; y = 2, при которых алгоритм вычисляет значения выражения первой ветви

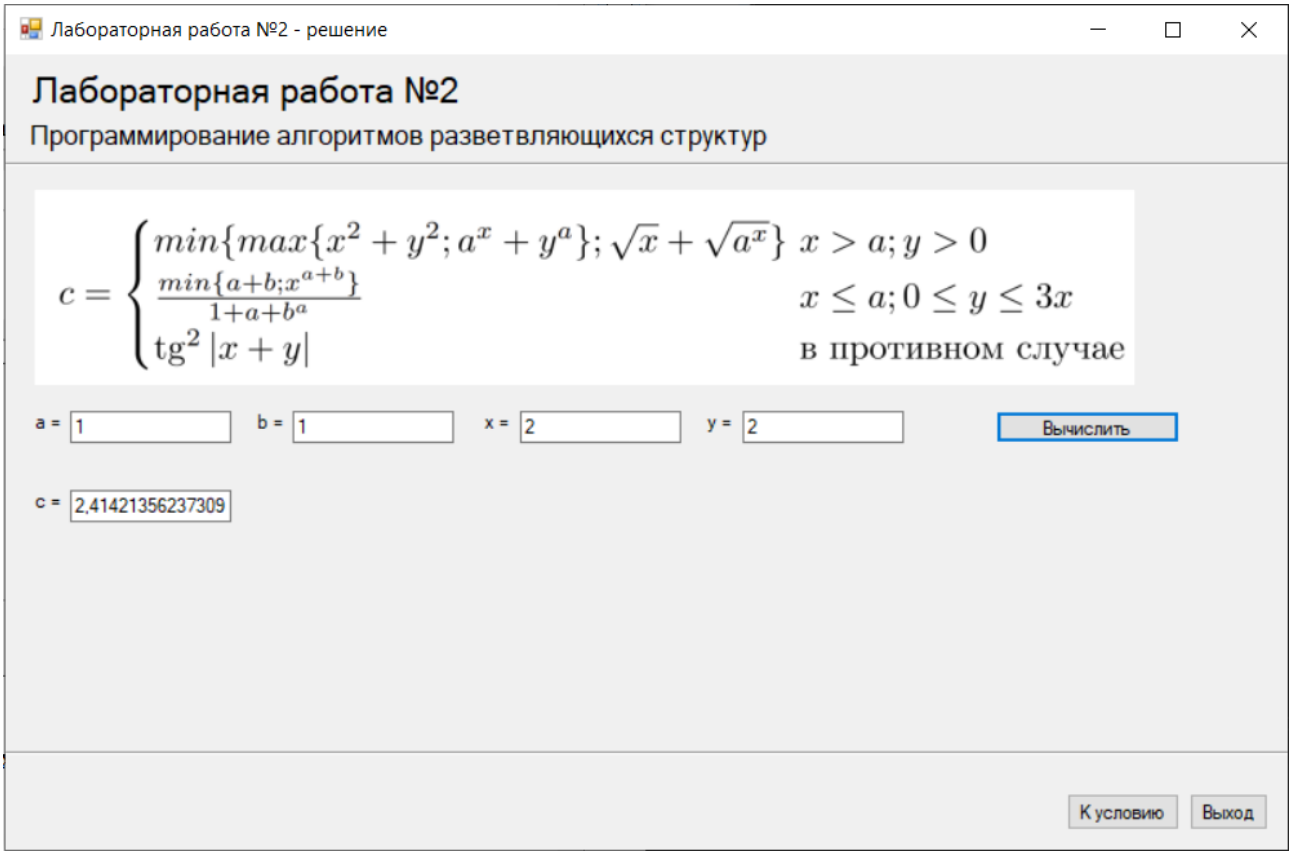

Рисунок 5 — результат выполнения программы при входных данных a = 1; b = 1; x = 2; y = 2

$x>a; y>0 \Leftrightarrow 2>1; 2>0$, следовательно алгоритм вычисляет значение выражения первой ветви.

Выполним проверку: $x^2+y^2=2^2+2^2=8$, $a^x+y^a=1^2+2^1=3$, максимальное из них — восемь; $\sqrt{x}+\sqrt{a^x}=\sqrt{2}+\sqrt{1}=2.414...$; минимальное значение из 3 и 2.414... — 2.414..., поэтому оно и является результатом выполнения программы.

На рисунке ниже приведен результат выполнения программы при входных данных a = 2; b = 1; x = 1; y = 2, при которых алгоритм вычисляет значения выражения второй ветви

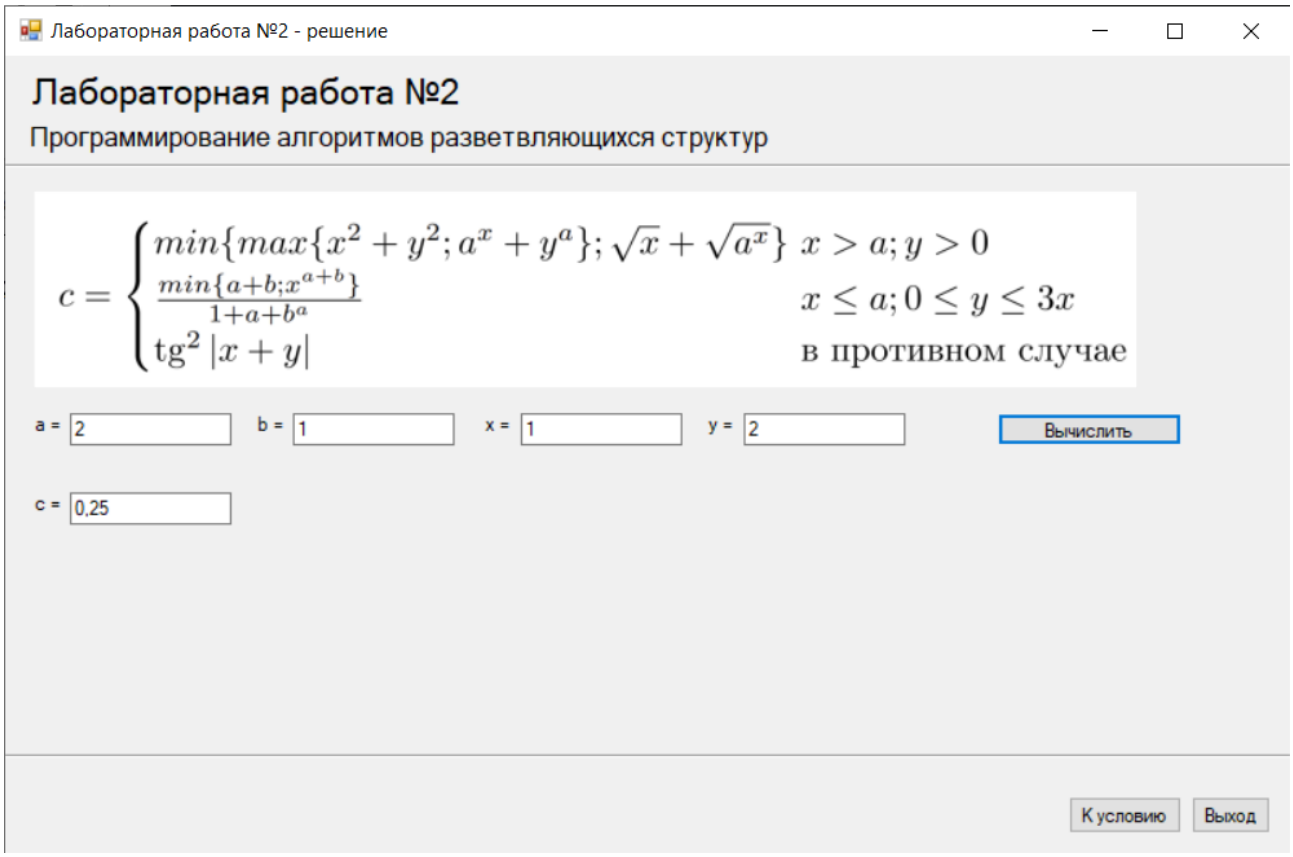

Рисунок 6 — результат выполнения программы при входных данных a = 2; b = 1; x = 1; y = 2

$x \le a; 0 \le y \le 3\, x \Leftrightarrow 1 \le 2; 0 \le 2 \le 3$, следовательно алгоритм вычисляет значение выражения второй ветви.

Выполним проверку: $a+b=2+1=3, x^{a+b}=1^3=1$, минимальное из них — 1; $1+a+b^a=1+2+1^2=4, \frac{min\{a+b; x^{a+b}\}}{1+a+b^a}=\frac{1}{4}=0.25$, данное значение является результатом выполнения программы.

На рисунке ниже приведен результат выполнения программы при входных данных a = 2; b = 1; x = 1; y = 4, при которых алгоритм вычисляет значения выражения третьей ветви

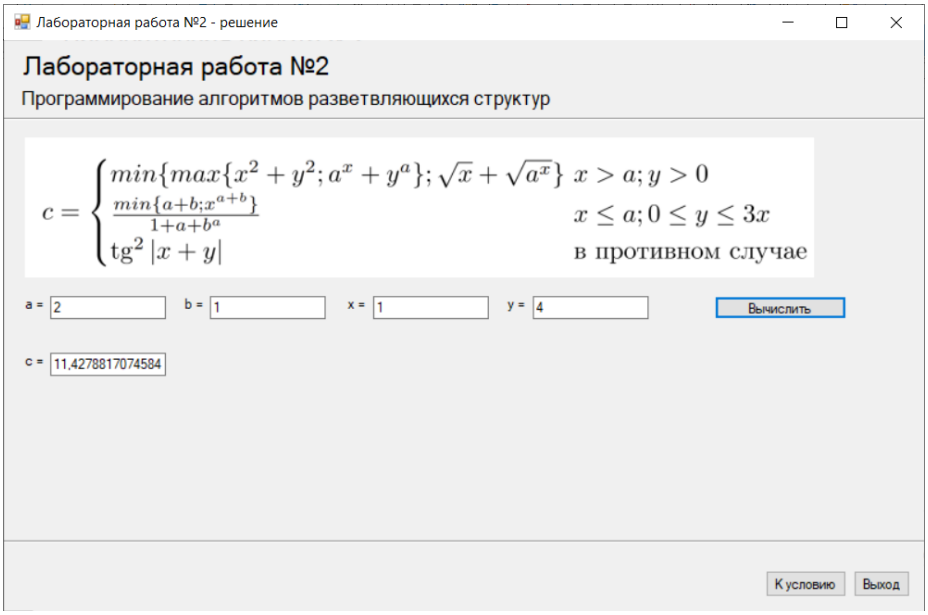

Рисунок 7 — результат выполнения программы при входных данных a = 2; b = 1; x = 1; y = 4

$x>a;\ y>0$ не справедливо, так как $x=1, a=2, y=4$; так же не справедливо $x\le a; 0\le y\le 3x$, следовательно алгоритм вычисляет значение выражения третьей ветви.

$tg\,5\approx -3.3805, tg^2 5\approx 11.4278\ldots$, данное значение и является результатом выполнения программы.

Список используемых источников

1. Гуриков С. Р. Введение в программирование на языке Visual C#: учебное пособие / С. Р. Гуриков. — Москва : ФОРУМ : ИНФРА-М, 2020. — 447 с.

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ордена Трудового Красного Знамени
федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра «Информатика»

ОТЧЕТ

по дисциплине «Алгоритмы и алгоритмические языки»

Лабораторная работа № 3

Табулирование функции

Выполнил: студент группы БФИ №2202

Сидорук Д. В.

Принял: старший преподаватель Загвоздкина А. В.

Москва, 2023 г.

Задание

Постройте таблицу и найдите наименьшее значение функции $y=f(x)$ при изменении x на отрезке $[a;b]$ с шагом h

$$y=\frac{\ln^2 x}{x}$$

Отрезок $[6;8]$, шаг $h=0{,}2$

Для того, чтобы выполнить задание, необходимо разработать следующие методы:

1. Метод public static double GetDouble(TextBox t), предназначенный для получения текста из текстового поля ввода и преобразования его в значение с плавающей точкой.

2. Метод public static void OutputDouble(TextBox t, double value), предназначенный для конвертации переданного значения в текст и установки его в качестве текста переданного текстового поля.

3. Метод public static double CalculateExpression(double x), предназначенный для нахождения значения данной функции.

4. Метод public static double TabulateFunction(double a, double b, double h, DataGridView view), предназначенный для табулирования данной функции и возвращения ее минимального значения.

5. Метод public static void OutputViewRow(DataGridView view, double x, double y), предназначенный для вывода результатов табулирования функции в табличный компонент DataGridView.

Вышеперечисленные методы должны быть размещены в dll-библиотеке.

Перечень блок-схем

На рисунке ниже приведена блок-схема алгоритма метода double GetDouble(TextBox t), предназначенного для получения текста из текстового поля ввода и преобразования его в значение с плавающей точкой.

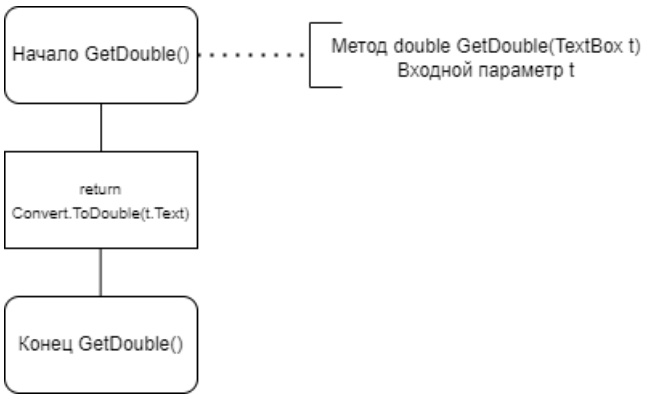

Рисунок 1 — блок-схема алгоритма метода double GetDouble(TextBox t), предназначенного для получения текста из текстового поля ввода и преобразования его в значение с плавающей точкой.

На рисунке ниже приведена блок-схема метода void OutputDouble(TextBox t, double value), предназначенного для конвертации переданного значения в текст и установки его в качестве текста переданного текстового поля.

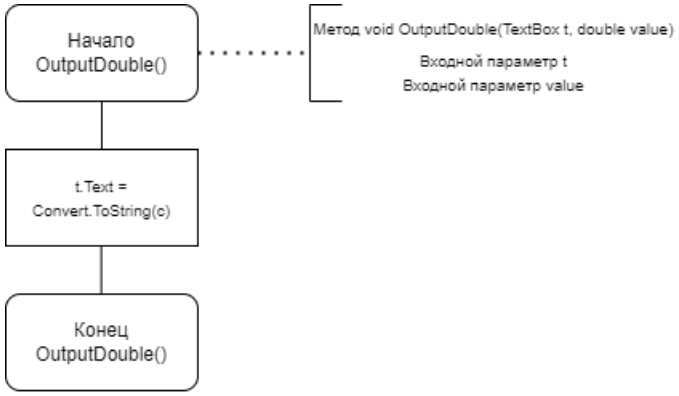

Рисунок 2 — блок-схема алгоритма метода void OutputDouble(TextBox t, double value), предназначенного для конвертации переданного значения в текст и установки его в качестве текста переданного текстового поля.

На рисунке ниже приведена блок-схема алгоритма метода public static double CalculateExpression(double x), предназначенного для нахождения значения данной функции.

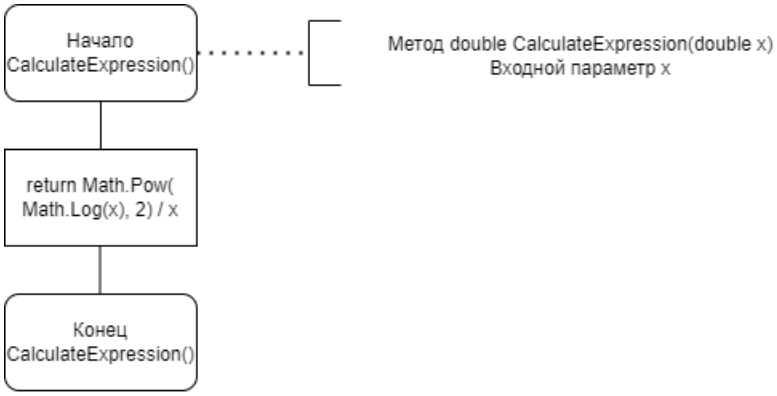

Рисунок 3 — блок-схема алгоритма метода double CalculateExpression(double x), предназначенного для нахождения значения данной функции.

На рисунке ниже приведена блок-схема алгоритма метода public static void OutputViewRow(DataGridView view, double x, double y), предназначенного для вывода результатов табулирования функции в табличный компонент DataGridView.

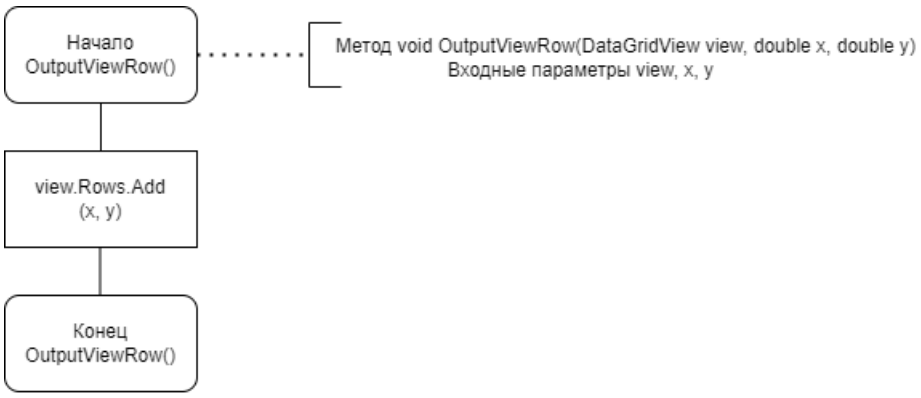

Рисунок 4 — блок-схема алгоритма void OutputViewRow(DataGridView view, double x, double y), предназначенного для вывода результатов табулирования функции в табличный компонент DataGridView.

На рисунке ниже приведена блок-схема алгоритма метода public static double TabulateFunction(double a, double b, double h, DataGridView view), предназначенного для табулирования данной функции и возвращения ее минимального значения.

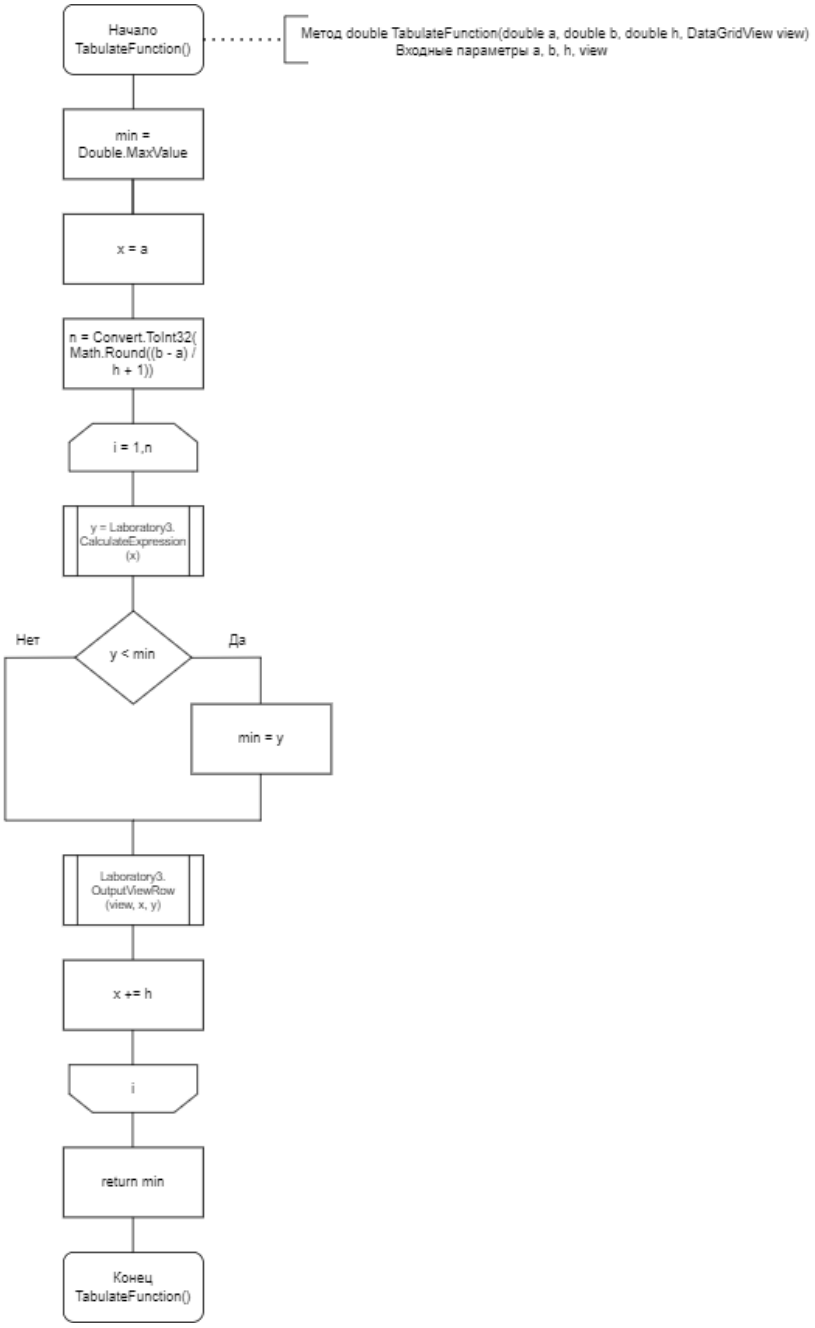

Рисунок 5 — блок-схема алгоритма double TabulateFunction(double a, double y, double h, DataGridView view), предназначенного для табулирования данной функции и возвращения ее минимального значения.

Алгоритм событийной кнопки

На рисунке ниже приведена блок-схема алгоритма событийной кнопки.

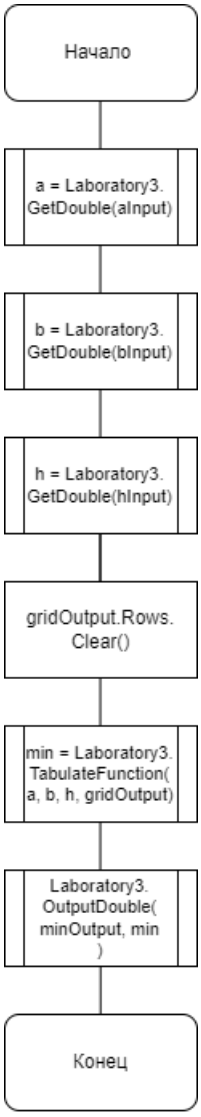

Рисунок 6 — блок-схема алгоритма событийной кнопки.

Содержание DLL-библиотеки

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

namespace Laboratory3Library
{
    public class Laboratory3
    {
        public static double GetDouble(TextBox t)
        {
            return Convert.ToDouble(t.Text);
        }
        public static void OutputDouble(TextBox t, double value)
        {
            t.Text = Convert.ToString(value);
        }
        public static void OutputViewRow(DataGridView view, double x, double y)
        {
            view.Rows.Add(x, y);
        }
        public static double CalculateExpression(double x)
        {
            return Math.Pow(Math.Log(x), 2) / x;
        }
        public static double TabulateFunction(double a, double b, double h,
DataGridView view)
        {
            double min = Double.MaxValue;
            double x = a;
            int n = Convert.ToInt32(Math.Round((b - a) / h + 1));

            for (int i = 1; i <= n; ++i)
            {
                double y = Laboratory3.CalculateExpression(x);
                if (y < min) min = y;
                Laboratory3.OutputViewRow(view, x, y);
                x += h;
            }

            return min;
        }
    }
}
```

Содержание основной части программы

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

using Laboratory3Library;

namespace LaboratoryWorks
{
    public partial class Laboratory3Form : Form
    {
        public Laboratory3Form()
        {
            InitializeComponent();
        }

        private void evaluateButton_Click(object sender, EventArgs e)
        {
            double a = Laboratory3.GetDouble(aInput);
            double b = Laboratory3.GetDouble(bInput);
            double h = Laboratory3.GetDouble(hInput);

            gridOutput.Rows.Clear();
            double min = Laboratory3.TabulateFunction(a, b, h, gridOutput);
            Laboratory3.OutputDouble(minOutput, min);
        }
    }
}
```

Результаты выполнения работы

На рисунке ниже приведен результат выполнения программы при входных данных a = 6; b = 8; h = 0.2.

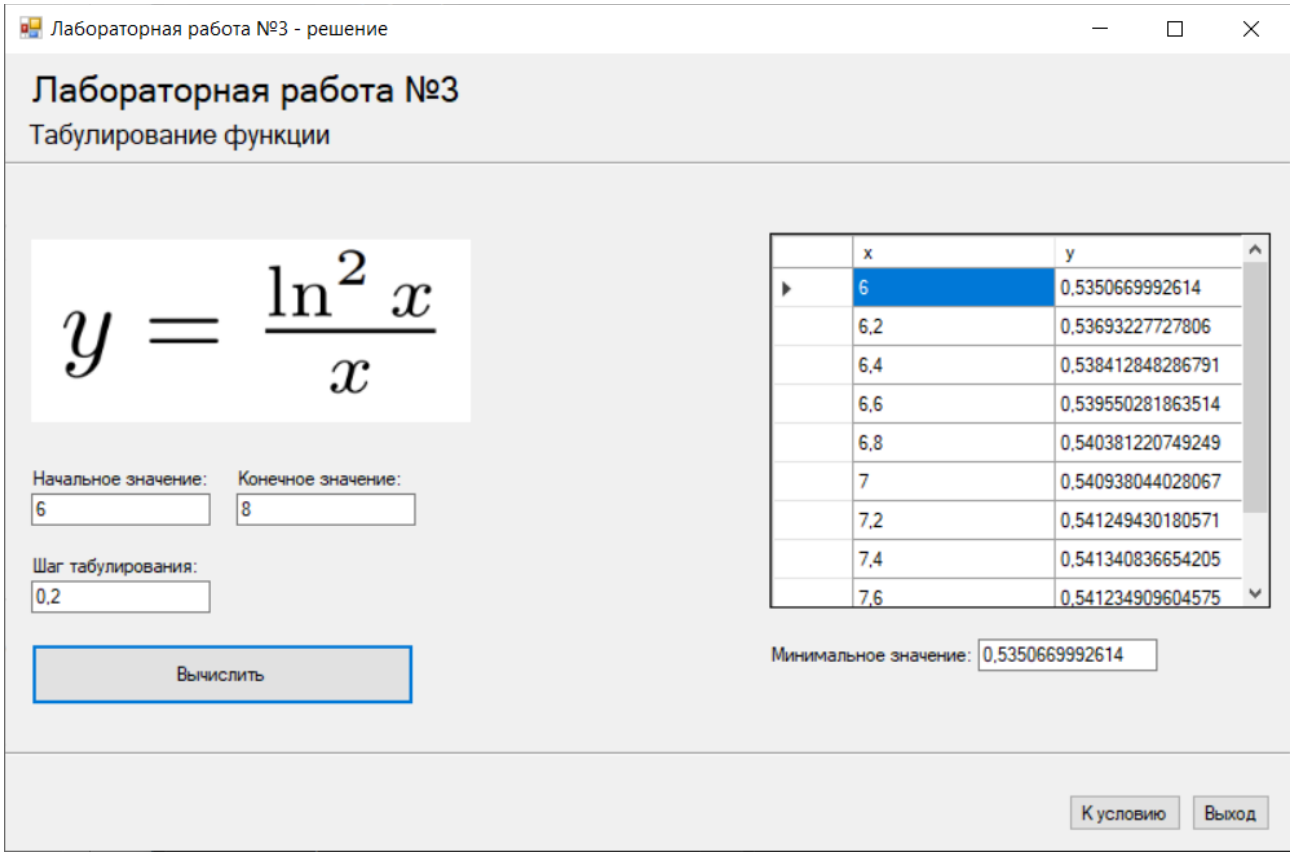

Рисунок 7 — результат выполнения программы при a = 6, b = 8, h = 0.2, значения от x = 6 до x = 7,6

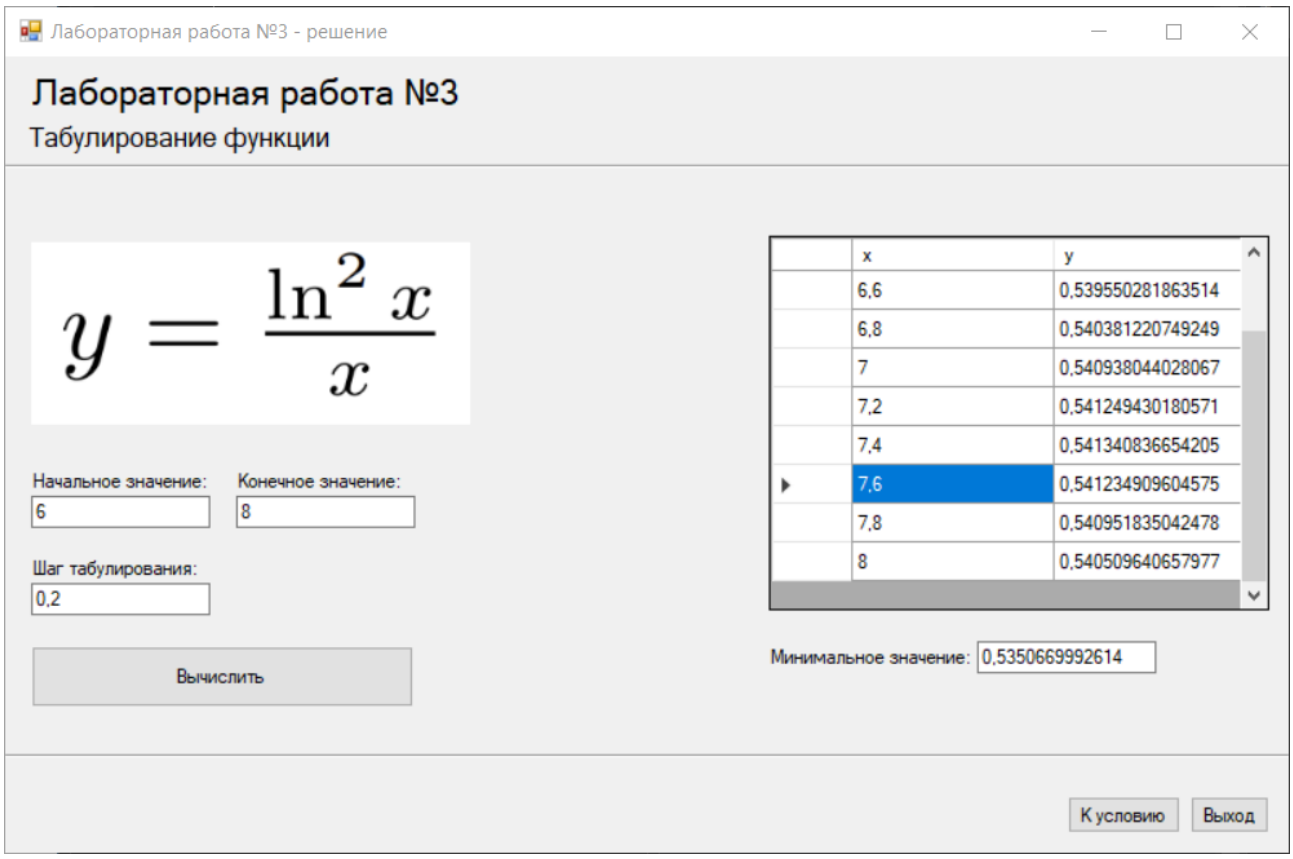

Рисунок 8 — результат выполнения программы при a = 6, b = 8, h = 0.2, значения от x = 6 до x = 8

Мы видим, что функция изменяется от начального значения до конечного с определенным шагом.

Выполним проверку, проверив корректность подсчета значений функции для начального и конечного x: $y(6)=\frac{\ln^2 6}{6}\approx\frac{3.2104}{6}\approx 0.535$ — корректное значение, $y(8)=\frac{\ln^2 8}{8}\approx\frac{4.324}{8}\approx 0.5405$ — корректное значение.

Также видим, что минимальное значение было найдено правильно.

На рисунке ниже приведен результат выполнения программы при входных данных a = 1; b = 3; h = 0.5.

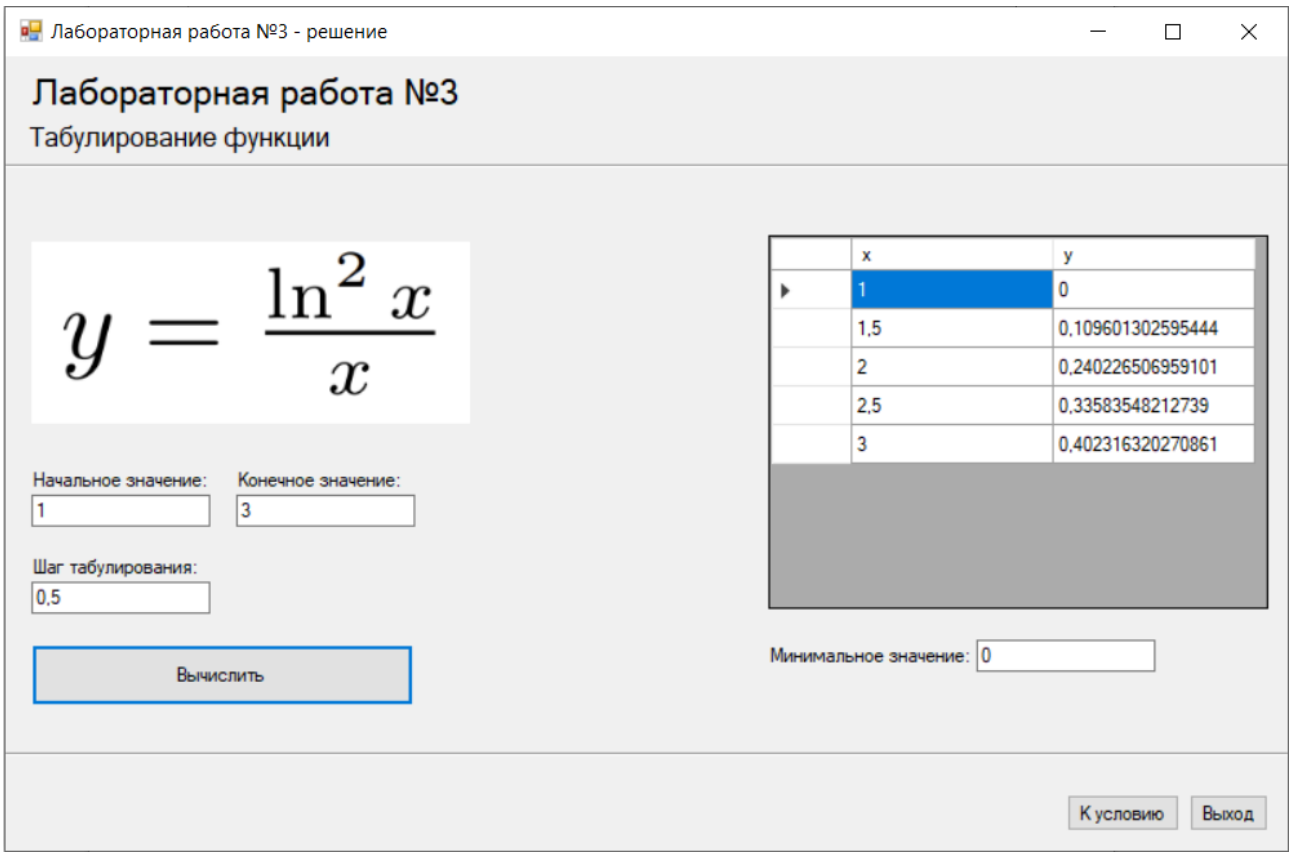

Рисунок 9 — результат выполнения программы при a = 1, b = 3, h = 0.5, значения от x = 1 до x = 3

Мы видим, что функция изменяется от начального значения до конечного с определенным шагом.

Выполним проверку, проверив корректность подсчета значений функции для начального и конечного x: $y(1)=\frac{\ln^2 1}{1}=\frac{0}{1}=0$ — корректное значение.

$y(3)=\frac{\ln^2 3}{3}\approx\frac{1.207}{3}\approx 0.4023$ — корректное значение.

Также видим, что минимальное значение было найдено правильно.

Список используемых источников

1. Гуриков С. Р. Введение в программирование на языке Visual C#: учебное пособие / С. Р. Гуриков. — Москва : ФОРУМ : ИНФРА-М, 2020. — 447 с.

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И
МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ордена Трудового Красного Знамени
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра «Информатика»

ОТЧЕТ

по дисциплине «Алгоритмы и алгоритмические языки»

Лабораторная работа № 4

Создание приложений, использующих итеративные циклические структуры

Выполнил: студент группы БФИ №2202

Сидорук Д. В.

Принял: старший преподаватель Загвоздкина А. В.

Москва, 2023 г.

Задание

Пусть $y_0=0; y_k=\frac{y_{k-1}+1}{y_{k-1}+2}; k=1,2,...$

Дано действительное число $\varepsilon>0$. Найдите первый член y_n, для которого выполнено условие $y_n-y_{n-1}<\varepsilon$.

Для того, чтобы выполнить задание, необходимо разработать следующие методы:

1. Метод public static double GetDouble(TextBox t), предназначенный для получения текста из текстового поля ввода и преобразования его в значение с плавающей точкой.

2. Метод public static void OutputDouble(TextBox t, double value), предназначенный для конвертации переданного значения в текст и установки его в качестве текста переданного текстового поля.

3. Метод public static double Calculate(double previous) для нахождения следующего члена последовательности на основании значения предыдущего ее члена.

4. Метод public static double TabulateFunction(DataGridView view, double start, double e), предназначенного для табулирования данной функции и возвращения первого ее члена, для которого выполнено условие $y_n-y_{n-1}<\varepsilon$.

5. Метод public static void OutputViewRow(DataGridView view, int n, double y), предназначенный для вывода результатов табулирования функции в табличный компонент DataGridView.

Вышеперечисленные методы должны быть размещены в dll-библиотеке.

Формализация задачи

Выведение рекуррентной формулы не требуется, она дана в постановке задачи: $y_k=\frac{y_{k-1}+1}{y_{k-1}+2}$. Нахождение производной не требуется.

Перечень блок-схем

На рисунке ниже приведена блок-схема алгоритма метода double GetDouble(TextBox t), предназначенного для получения текста из текстового поля ввода и преобразования его в значение с плавающей точкой.

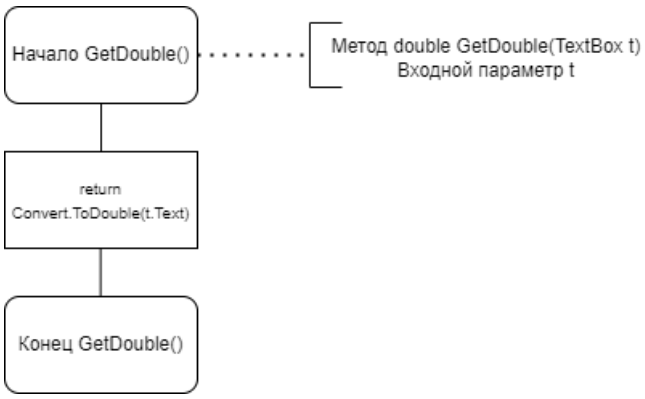

Рисунок 1 — блок-схема алгоритма метода double GetDouble(TextBox t), предназначенного для получения текста из текстового поля ввода и преобразования его в значение с плавающей точкой.

На рисунке ниже приведена блок-схема алгоритма метода void OutputDouble(TextBox t, double value), предназначенного для конвертации переданного значения в текст и установки его в качестве текста переданного текстового поля.

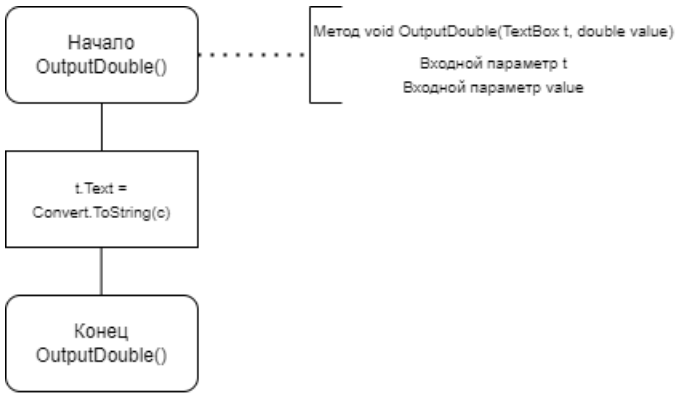

Рисунок 2 — блок-схема алгоритма метода void OutputDouble(TextBox t, double value), предназначенного для конвертации переданного значения в текст и установки его в качестве текста переданного текстового поля.

На рисунке ниже приведена блок-схема алгоритма метода double Calculate(double previous), предназначенного для нахождения следующего члена последовательности на основании значения предыдущего ее члена.

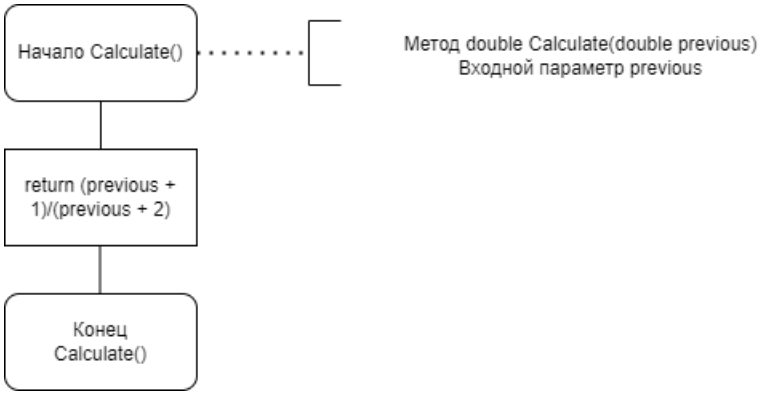

Рисунок 3 — блок-схема алгоритма метода double Calculate(double previous), предназначенного для нахождения следующего члена последовательности на основании значения предыдущего ее члена.

На рисунке ниже приведена блок-схема алгоритма метода public static void OutputViewRow(DataGridView view, double n, double y), предназначенного для вывода результатов табулирования функции в табличный компонент DataGridView.

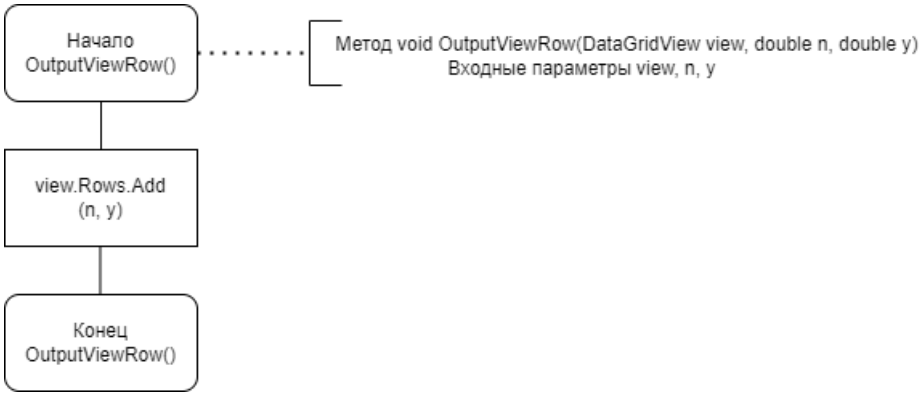

Рисунок 4 — блок-схема алгоритма void OutputViewRow(DataGridView view, double n, double y), предназначенного для вывода результатов табулирования функции в табличный компонент DataGridView.

На рисунке ниже приведена блок-схема алгоритма метода double TabulateFunction(DataGridView view, double start, double e), предназначенного для табулирования данной функции и возвращения первого ее члена, для которого выполнено условие $y_n - y_{n-1} < \varepsilon$.

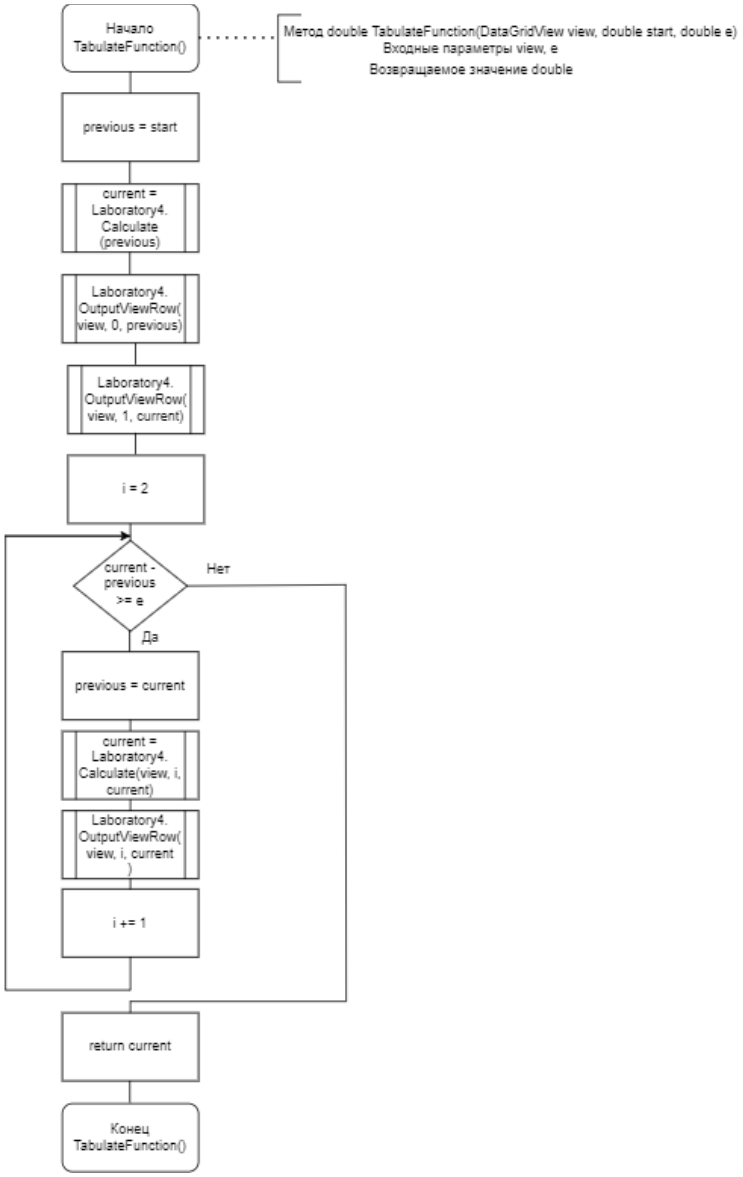

Рисунок 5 — блок-схема алгоритма метода double TabulateFunction(DataGridView view, double start, double e), предназначенного для табулирования данной функции и возвращения первого ее члена, для которого выполнено условие $y_n - y_{n-1} < \varepsilon$.

Алгоритм событийной кнопки

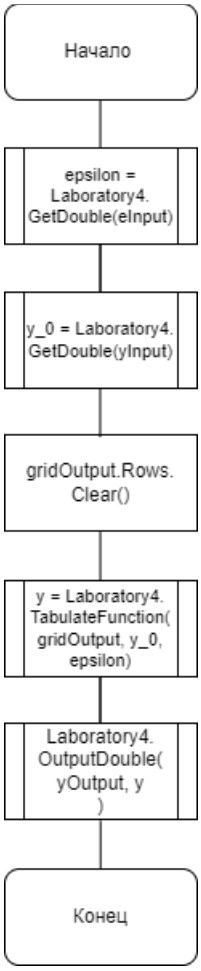

Рисунок 6 — блок-схема алгоритма событийной кнопки

Содержание dll-библиотеки

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

namespace Laboratory4Library
{
    public class Laboratory4
    {
        public static double GetDouble(TextBox t)
        {
            return Convert.ToDouble(t.Text);
        }
        public static void OutputDouble(TextBox t, double value)
        {
            t.Text = Convert.ToString(value);
        }
        public static void OutputViewRow(DataGridView view, int n, double y)
        {
            view.Rows.Add(n, y);
        }
        public static double Calculate(double previous)
        {
            return (previous + 1) / (previous + 2);
        }
        public static double TabulateFunction(DataGridView view, double start, double
e)
        {
            double previous = start;
            double current = Laboratory4.Calculate(previous);

            Laboratory4.OutputViewRow(view, 0, previous);
            Laboratory4.OutputViewRow(view, 1, current);

            int i = 2;
            while (current - previous >= e)
            {
                previous = current;
                current = Laboratory4.Calculate(previous);
                Laboratory4.OutputViewRow(view, i, current);
                i += 1;
            }

            return current;
        }

    }
}
```

Содержание основной части программы

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

using Laboratory4Library;

namespace LaboratoryWorks
{
    public partial class Laboratory4Form : Form
    {
        public Laboratory4Form()
        {
            InitializeComponent();
        }

        private void evaluateButton_Click(object sender, EventArgs e)
        {
            double epsilon = Laboratory4.GetDouble(eInput);
            double y_0 = Laboratory4.GetDouble(yInput);

            gridOutput.Rows.Clear();
            double y = Laboratory4.TabulateFunction(gridOutput, y_0, epsilon);
            Laboratory4.OutputDouble(yOutput, y);
        }
    }
}
```

Результаты выполнения программы

На рисунке ниже приведен результат выполнения программы при $\varepsilon = 0{,}01$, $y_0 = 0$

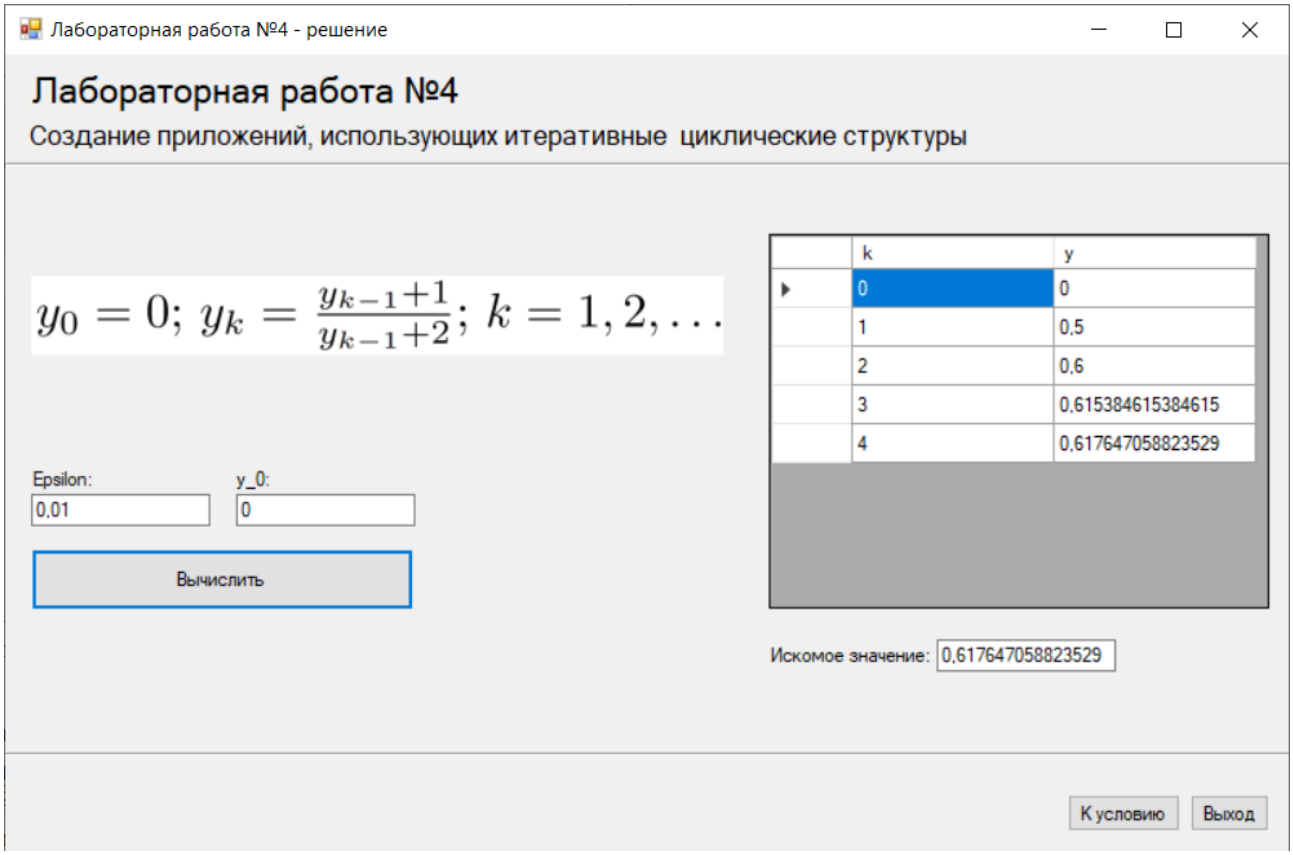

Рисунок 7 — результат выполнения программы при $\varepsilon = 0{,}01$, $y_0 = 0$

Для проверки убедимся, что все значения посчитаны верно: $y_0 = 0, y_1 = \frac{0+1}{0+2} = 0.5, y_2 = \frac{0.5+1}{0.5+2} = \frac{1.5}{2.5} = 0.6, y_3 = \frac{0.6+1}{0.6+2} = \frac{1.6}{2.6} \approx 0.615, y_4 \approx \frac{1.615}{2.615} \approx 0.617$

Теперь убедимся, что искомое значение найдено верно: $0.5 - 0$ не меньше, чем 0.01, поэтому y_1 не является ответом; $0.6 - 0.5 = 0.1$ не меньше, чем 0.01, поэтому y_2 не является ответом; $0.615 - 0.6 = 0.015$ не меньше, чем 0.01, поэтому y_3 не является ответом; $0.617 - 0.615 = 0.002$ меньше, чем 0.01, поэтому y_4 является ответом.

На рисунке ниже приведен результат выполнения программы при $\varepsilon = 0{,}01$, $y_0 = 0{,}2$

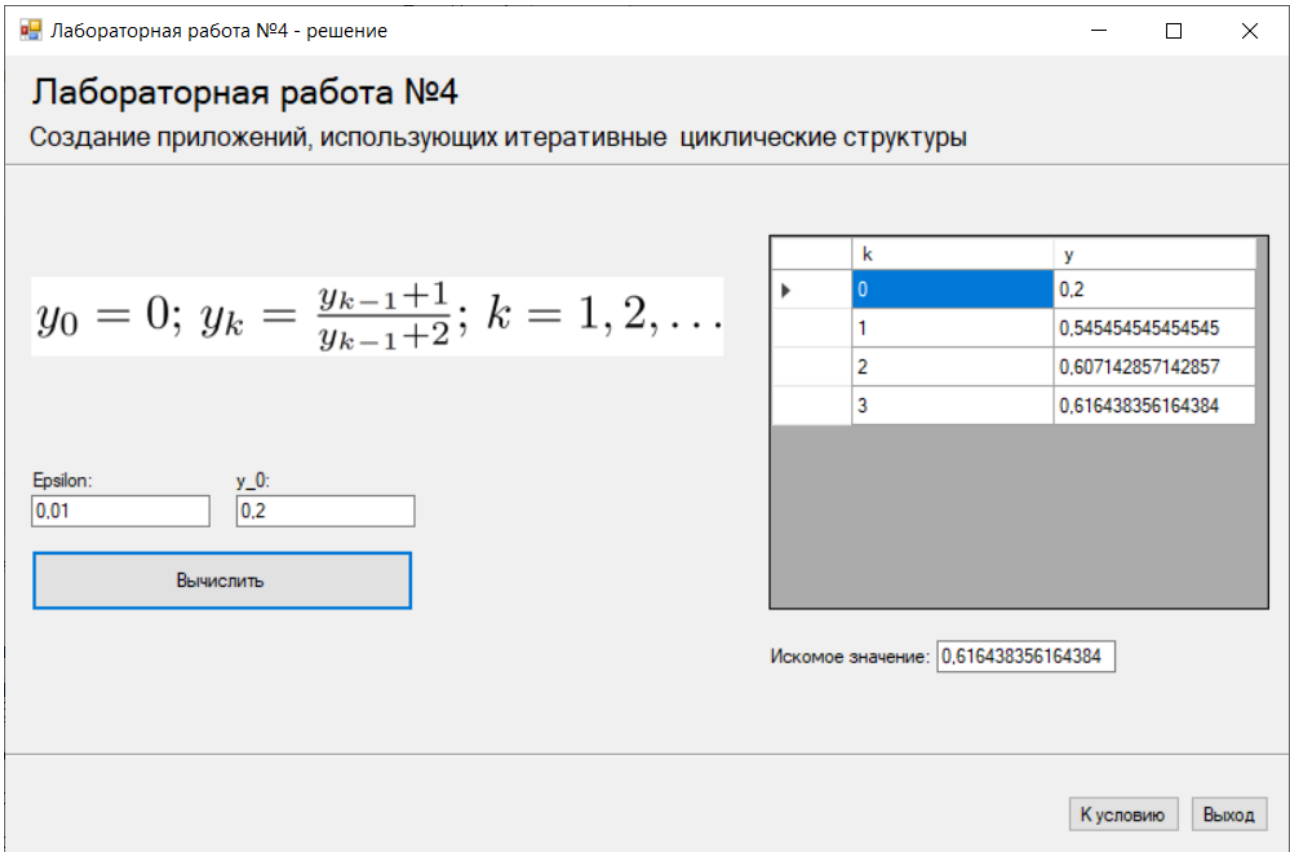

Рисунок 8 — результат выполнения программы при $\varepsilon = 0{,}01$, $y_0 = 0{,}2$

Для проверки убедимся, что все значения посчитаны верно: $y_0 = 0.2$, $y_1 = \frac{1.2}{2.2} = 0.545$, $y_2 = \frac{1.545}{2.545} \approx 0.607$, $y_3 = \frac{1.607}{2.607} \approx 0.616$

Теперь убедимся, что искомое значение найдено верно: $0.545 - 0.2 = 0.345$ не меньше, чем 0.01, поэтому y_1 не является ответом; $0.607 - 0.545 = 0.062$ не меньше, чем 0.01, поэтому y_2 не является ответом; $0.616 - 0.607 = 0.009$ меньше, чем 0.01, поэтому y_3 является ответом.

Список используемых источников

1. Гуриков С. Р. Введение в программирование на языке Visual C#: учебное пособие / С. Р. Гуриков. — Москва : ФОРУМ : ИНФРА-М, 2020. — 447 с.