МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
НАБЕРЕЖНОЧЕЛНИНСКИЙ ИНСТИТУТ (ФИЛИАЛ)
КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ (ИС)

Утверждаю
Заведующий кафедрой ИС

_____Р.А.Валиев

_____г.

КУРСОВАЯ РАБОТА

по дисциплине:
«Объектно-ориентированное программирование»
на тему:
«Разработка прикладной программы с использованием объектно-ориентированной технологии»

Автор:
студент группы 2161117

_____ А.А. Пугачёв

Оценка: _____

Руководитель:
доцент кафедры ИС

_____ Е.В. Зубков

Дата защиты: _____г. .

Набережные Челны
2018

КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

НАБЕРЕЖНОЧЕЛНИНСКИЙ ИНСТИТУТ (ФИЛИАЛ)

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ (ИС)

Направление подготовки 09.03.01

«Информатика и вычислительная техника»

Утверждаю
Заведующий кафедрой ИС

_____Р.А.Валиев

_____г.

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент: Пугачёв Алексей Алексеевич

1 Тема

«Разработка прикладной программы с использованием объектно-ориентированной технологии»

2 Содержание

«Разработка программы «Функциональный калькулятор» »

3 Срок представления к защите

11.06.2018 г.

4 Исходные данные
- информация по работе с Windows Form;
- информация о простых функциях и константах применяемых при вычислениях

5 Перечень подлежащих разработке вопросов
- анализ предметной области;
- проектирование системы с помощью методологи UML.

Задание выдано _____ г. _____ Е.В. Зубков

Задание принято _____ г. _____ А.А. Пугачёв

СОДЕРЖАНИЕ

1. Проектирование программного продукта

1.1 UML диаграмма Прецедентов

Данная диаграмма описывает возможные сценарии работы пользователя(актера) с приложением для калькулятора, где в качестве овалов(прецедентов) выступает требуемая цель со стороны пользователя, а связь между ними и пользователем осуществлена пунктирными линиями(зависимость), которые показывают, что все представленные прецеденты зависят от действий актёра, и в то же время, возвращают ему результат своей работы.

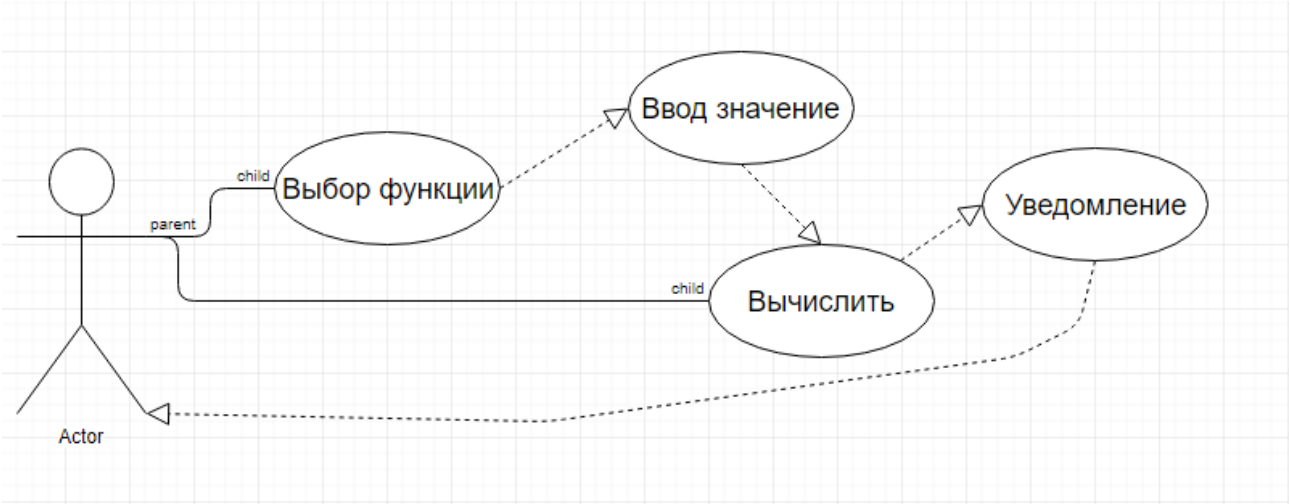

Рисунок 1.1 UML диаграмма Прецедентов

1.2 UML диаграмма Классов

Данный вид диаграммы определяет типы объектов системы и различного рода связи, которые существуют между ними.

Наша программа FunctionalCalc имеет точку входа в методе главного класса Form1. В виде стрелки с закрашенным острием в виде ромба изображена связь композиции, которая показывает, что класс Form1 является неделимой частью пространства имен программы FunctionalCalc. А связь ассоциация, между классами Form1 и FunctionalCalculator, показывает, что объекты одного класса связанны с объектами другого, и между этими классами происходит “полезное” взаимодействие.

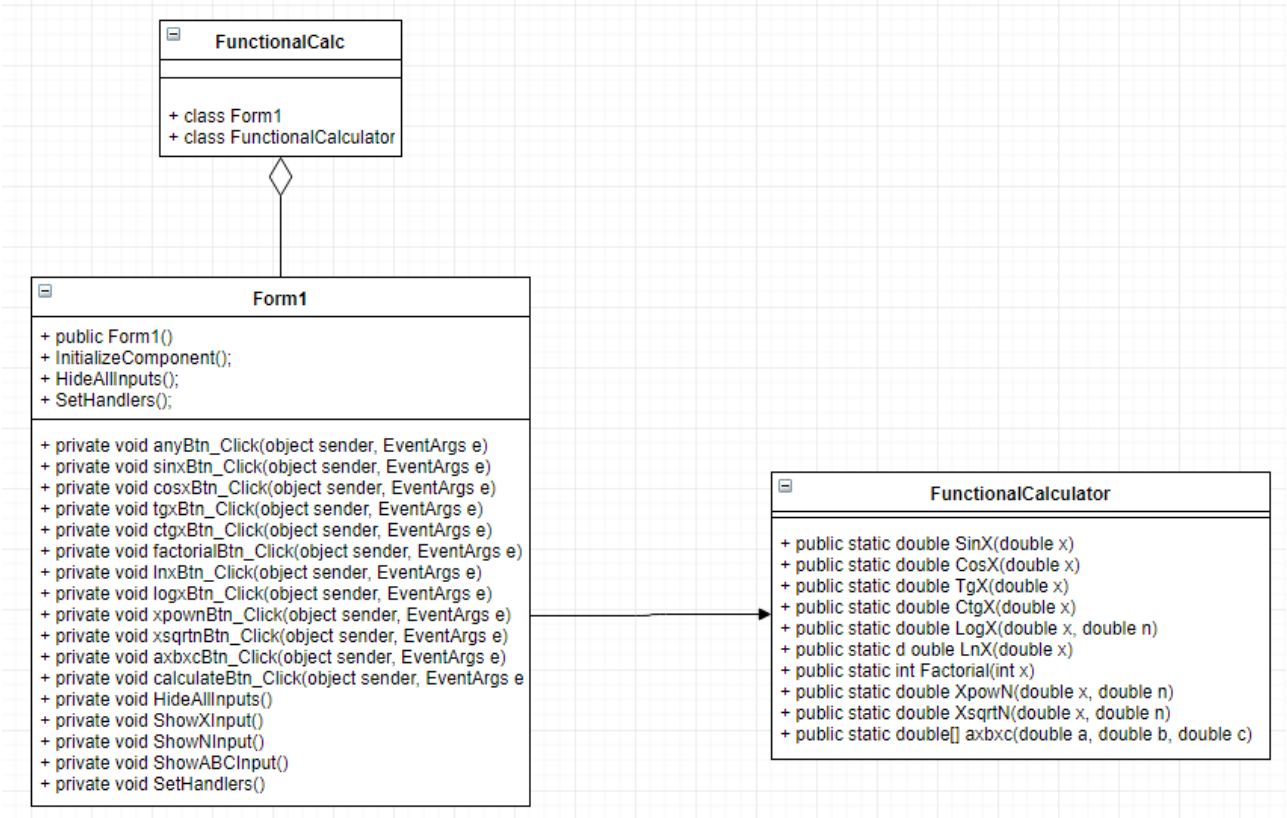

Рисунок 1.2 UML диаграмма Классов

1.3 UML диаграмма Состояний

Диаграмма Состояний отражает динамический алгоритм работы прикладной программы в зависимости от условий.

Вход в программу осуществляется с помощью черного закрашенного круга, а выход- с помощью такого же закрашенного круга, на фоне с белым внутренним ободом. В качестве прямоугольника с закругленными краями выступает поведение нашего главного объекта(формы), и в зависимости от направления стрелки, изменяется и состояние объекта. Закрашенные прямоугольники описывают параллельные процессы.

Рисунок 1.3 UML диаграмма Состояний

1.4 UML диаграмма Последовательностей

Диаграмма Последовательностей служит для описания более детального взаимодействия частей системы, нежели диаграмма Прецедентов. В верхней части диаграммы изображены “действующие лица” программы. Стрелка слева-направо отражают очередность вызова какого-то действия со стороны пользователя и получения сообщения от исполняющего эту функцию метода. Горизонтальный прямоугольник под “действующим лицом” программы показывает его время действия в контексте выполнения всей системы.

Вся работа приложения начинается с запуска приложения пользователем, а все взаимодействия с функционалом программы производятся внутри системы исходя из запросов пользователя. Тем самым программа получив сигнал переходит уже к выполнению определенных действий, основанных на готовых алгоритмах самой программы.

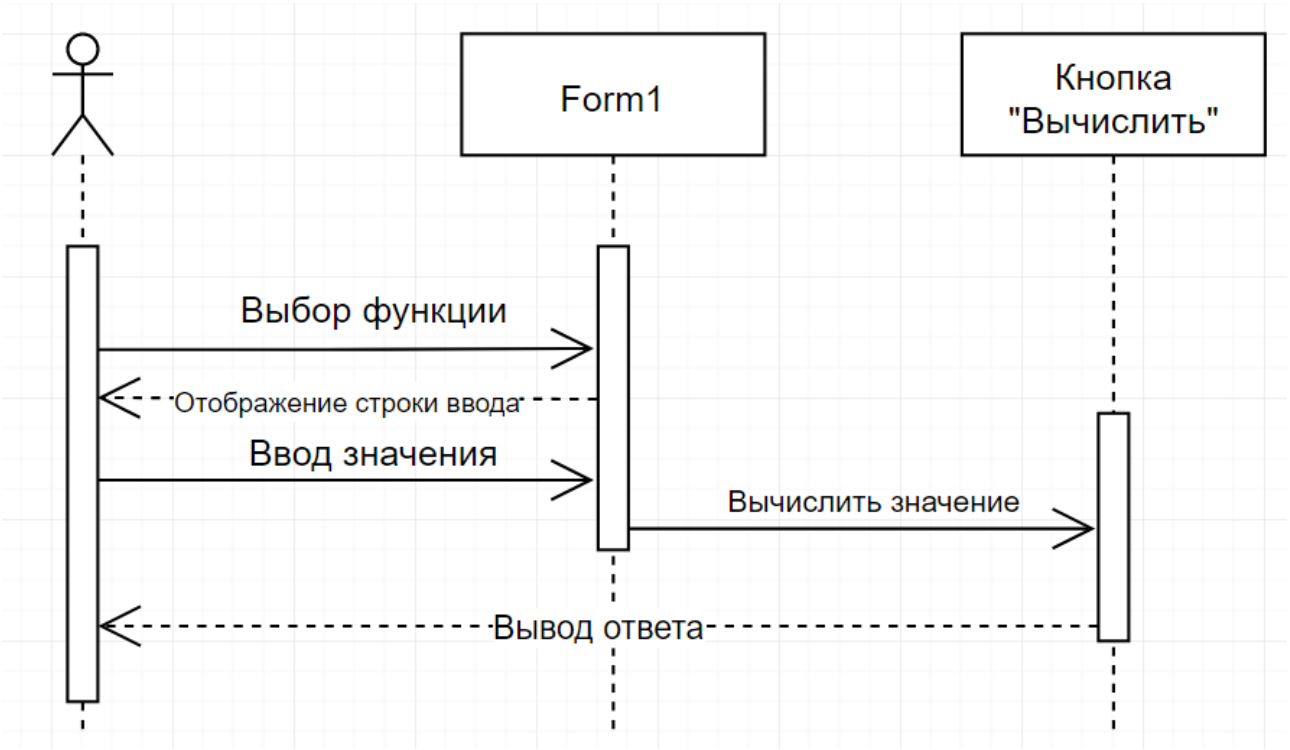

Рисунок 1.4 UML диаграмма Последовательностей

2. Листинг кода прикладной программы

2.1 Form1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FunctionalCalc
{
    public partial class Form1 : Form
    {
        Button choosenBtn = null; //показывает, какая кнопка была выбрана.
        //конструктор формы
        public Form1()
        {
            InitializeComponent(); //инициализация объектов формы
            HideAllInputs();
            SetHandlers();
        }

        //обработчик, который вызывается при нажатии на любую из кнопок функций. Нужен для
перекраски.
        private void anyBtn_Click(object sender, EventArgs e)
        {
            //обнуляем цвета фона всех кнопок
            sinxBtn.BackColor = SystemColors.ControlLightLight;
            cosxBtn.BackColor = SystemColors.ControlLightLight;
            tgxBtn.BackColor = SystemColors.ControlLightLight;
            ctgxBtn.BackColor = SystemColors.ControlLightLight;
            factorialBtn.BackColor = SystemColors.ControlLightLight;
            lnxBtn.BackColor = SystemColors.ControlLightLight;
            logxBtn.BackColor = SystemColors.ControlLightLight;
            xpownBtn.BackColor = SystemColors.ControlLightLight;
            xsqrtnBtn.BackColor = SystemColors.ControlLightLight;
            axbxcBtn.BackColor = SystemColors.ControlLightLight;

            //используем явное приведение типов для обращения к кнопке и задаём ей новый
цвет.
            (sender as Button).BackColor = SystemColors.GradientActiveCaption;

            //запоминаем кнопку функции, на которую нажали.
            choosenBtn = (sender as Button);

            HideAllInputs();
        }

        //обработчик нажатия на кнопку Синус
        private void sinxBtn_Click(object sender, EventArgs e)
        {
            ShowXInput();
        }
        //обработчик нажатия на кнопку косинус
        private void cosxBtn_Click(object sender, EventArgs e)
        {
```

```
    ShowXInput();
}
//обработчик нажатия на кнопку тангенс
private void tgxBtn_Click(object sender, EventArgs e)
{
    ShowXInput();
}
//обработчик нажатия на кнопку котангенс
private void ctgxBtn_Click(object sender, EventArgs e)
{
    ShowXInput();
}
//обработчик нажатия на кнопку факториал
private void factorialBtn_Click(object sender, EventArgs e)
{
    ShowXInput();
}
//обработчик нажатия на кнопку нат. логарифм
private void lnxBtn_Click(object sender, EventArgs e)
{
    ShowXInput();
}
//обработчик нажатия на кнопку логарифм по основанию
private void logxBtn_Click(object sender, EventArgs e)
{
    ShowXInput();
    ShowNInput();
}
//обработчик нажатия на кнопку x в степени
private void xpownBtn_Click(object sender, EventArgs e)
{
    ShowXInput();
    ShowNInput();
}
//обработчик нажатия на кнопку корень из x
private void xsqrtnBtn_Click(object sender, EventArgs e)
{
    ShowXInput();
    ShowNInput();
}
//обработчик нажатия на кнопку Квадратное уравнение с целыми коэф-тами
private void axbxcBtn_Click(object sender, EventArgs e)
{
    ShowABCInput();
}

//обработчик нажатия на кнопку "Вычислить"
private void calculateBtn_Click(object sender, EventArgs e)
{
    if (choosenBtn == sinxBtn)
    {
        if (tbX.Text == "")
            MessageBox.Show("Аргумент не указан.");
        else
        {
            var answer = FunctionalCalculator.SinX(double.Parse(tbX.Text));
            tbAnswer.Text = answer.ToString();
        }
    }
    if (choosenBtn == cosxBtn)
    {
        if (tbX.Text == "")
            MessageBox.Show("Аргумент не указан.");
        else
```

```
                {
                    var answer = FunctionalCalculator.CosX(double.Parse(tbX.Text));
                    tbAnswer.Text = answer.ToString();
                }
            }
            if (choosenBtn == tgxBtn)
            {
                if (tbX.Text == "")
                    MessageBox.Show("Аргумент не указан.");
                else
                {
                    var answer = FunctionalCalculator.TgX(double.Parse(tbX.Text));
                    tbAnswer.Text = answer.ToString();
                }
            }
            if (choosenBtn == ctgxBtn)
            {
                if (tbX.Text == "")
                    MessageBox.Show("Аргумент не указан.");
                else
                {
                    var answer = FunctionalCalculator.CtgX(double.Parse(tbX.Text));
                    tbAnswer.Text = answer.ToString();
                }
            }
            if (choosenBtn == factorialBtn)
            {
                if (tbX.Text == "")
                    MessageBox.Show("Аргумент не указан.");
                else
                {
                    var answer = FunctionalCalculator.Factorial(int.Parse(tbX.Text));
                    tbAnswer.Text = answer.ToString();
                }
            }
            if (choosenBtn == lnxBtn)
            {
                if (tbX.Text == "")
                    MessageBox.Show("Аргумент не указан.");
                else
                {
                    var answer = FunctionalCalculator.LnX(double.Parse(tbX.Text));
                    tbAnswer.Text = answer.ToString();
                }
            }
            if (choosenBtn == logxBtn)
            {
                if (tbX.Text == "")
                    MessageBox.Show("Аргументы не указаны.");
                else
                {
                    var answer = FunctionalCalculator.LogX(double.Parse(tbX.Text),
double.Parse(tbN.Text));
                    tbAnswer.Text = answer.ToString();
                }
            }
            if(choosenBtn == xpownBtn)
            {
                if (tbX.Text == "")
                    MessageBox.Show("Аргументы не указаны.");
                else
                {
                    var answer = FunctionalCalculator.XpowN(double.Parse(tbX.Text),
double.Parse(tbN.Text));
```

```
                    tbAnswer.Text = answer.ToString();
                }
            }
            if(choosenBtn == xsqrtnBtn)
            {
                if (tbX.Text == "" || tbN.Text == "")
                    MessageBox.Show("Аргументы не указаны.");
                else
                {
                    var answer = FunctionalCalculator.XsqrtN(double.Parse(tbX.Text),
double.Parse(tbN.Text));
                    tbAnswer.Text = answer.ToString();
                }
            }
            if (choosenBtn == axbxcBtn)
            {
                if (tbA.Text == "" || tbB.Text == "" || tbC.Text == "")
                    MessageBox.Show("Аргументы не указаны.");
                else
                {
                    var answer = FunctionalCalculator.axbxc(double.Parse(tbA.Text),
double.Parse(tbB.Text), double.Parse(tbC.Text));
                    tbAnswer.Text = "X1: " + Math.Round(answer[0],3).ToString() + ", X2: " +
Math.Round(answer[1],3).ToString();
                }
            }
        }

        //скрывает все поля ввода вместе с надписями.
        private void HideAllInputs()
        {
            xLabel.Visible = false;
            tbX.Visible = false;
            nLabel.Visible = false;
            tbN.Visible = false;
            tbA.Visible = false;
            tbB.Visible = false;
            tbC.Visible = false;
            xPlusLabel.Visible = false;
            yPlusLabel.Visible = false;
            nullLabel.Visible = false;
        }

        //показывает поле для ввода значения X
        private void ShowXInput()
        {
            xLabel.Visible = true;
            tbX.Visible = true;
            tbX.Text = "";
        }
        //показывает поле для ввода значения N
        private void ShowNInput()
        {
            nLabel.Visible = true;
            tbN.Visible = true;
            tbN.Text = "";
        }
        //показывает поля для ввода коэф-тов квадратного уравнения.
        private void ShowABCInput()
        {
            tbA.Visible = true;
            tbA.Text = "";
            tbB.Visible = true;
            tbB.Text = "";
```

```
            tbC.Visible = true;
            tbC.Text = "";
            xPlusLabel.Visible = true;
            yPlusLabel.Visible = true;
            nullLabel.Visible = true;
        }

        //устанавливает всем кнопкам обработчики событий.
        private void SetHandlers()
        {
            sinxBtn.Click += sinxBtn_Click;
            cosxBtn.Click += cosxBtn_Click;
            tgxBtn.Click += tgxBtn_Click;
            ctgxBtn.Click += ctgxBtn_Click;
            factorialBtn.Click += factorialBtn_Click;
            lnxBtn.Click += lnxBtn_Click;
            logxBtn.Click += logxBtn_Click;
            xpownBtn.Click += xpownBtn_Click;
            xsqrtnBtn.Click += xsqrtnBtn_Click;
            axbxcBtn.Click += axbxcBtn_Click;
        }
    }
}
```

2.2 FunctionalCalculator

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FunctionalCalc
{
    //класс абстрактный, поскольку нам не нужно создавать его экземпляры.
    //все методы и поля класса - статические
    abstract class FunctionalCalculator
    {
        public static double SinX(double x)
        {
            x = Math.PI * x / 180;
            return Math.Sin(x);
        }

        public static double CosX(double x)
        {
            x = Math.PI * x / 180;
            return Math.Cos(x);
        }

        public static double TgX(double x)
        {
            x = Math.PI * x / 180;
            return Math.Tan(x);
        }

        public static double CtgX(double x)
        {
            x = Math.PI * x / 180;
            return (1.0 / Math.Tan(x));
        }

        public static double LogX(double x, double n)
        {
            return Math.Log(x, n);
        }

        public static double LnX(double x)
        {
            return Math.Log(x);
        }

        public static int Factorial(int x)
        {
            int retVal = 1;

            for (int i = 2; i <= x; i++)
                retVal *= i;
            return retVal;
        }

        public static double XpowN(double x, double n)
        {
            return Math.Pow(x, n);
        }
```

```
        public static double XsqrtN(double x, double n)
        {
            return Math.Pow(x, 1 / n);
        }

        public static double[] axbxc(double a, double b, double c)
        {
            var disc = b * b - 4 * a * c;
            var sqrtDisc = Math.Sqrt(disc);

            var x1 = (-1 * b + sqrtDisc) / (2 * a);
            var x2 = (-1 * b - sqrtDisc) / (2 * a);

            return new double[] { x1, x2 };
        }

    }
}
```

2.3 Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FunctionalCalc
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

3. Результат выполнения программы

При запуске программы перед нами открывается главное окно, изображенный на рисунке 4.1. на котором находятся:

- кнопки с функциями

- кнопка «Вычислить»

- несколько строков для ввода значения (в зависимости от выбора функции появляется нужное количество значений)

- строка для вывода ответа при вычисления функции

- информация «Значения» для пользователя

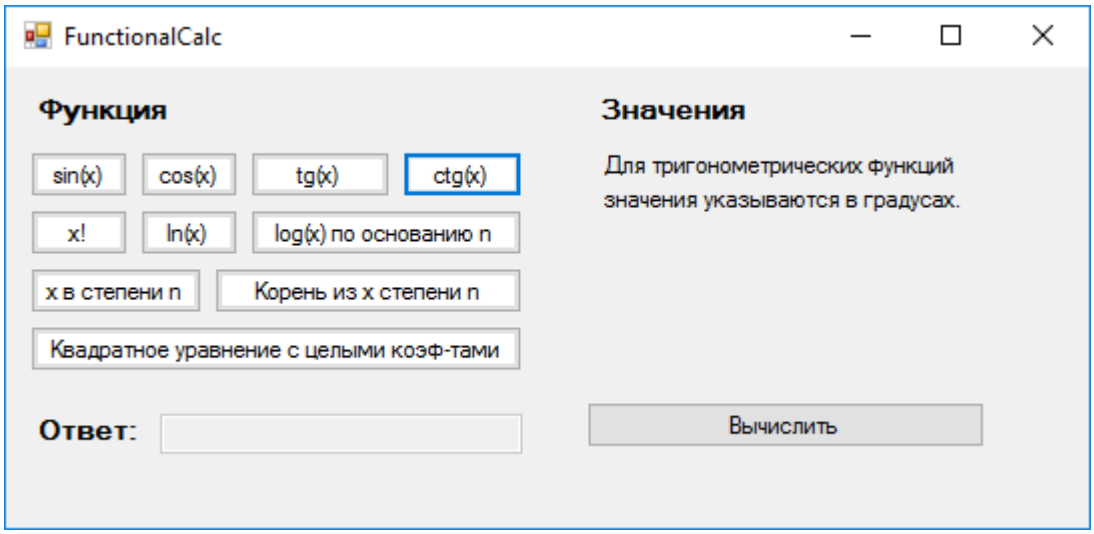

Рис. 4.1

При нажатии, к примеру, на кнопку с функцией sin(x), загорается кнопка синим цветом и появляется строка для ввода значения. В случае, если пользователь не вспомнит, какую функцию он выбирал для вычисления, то для этого и есть «светящиеся» кнопка, и она останется такой до сих пор, пока пользователь не перейдет к другой функции или не закроет программу. Пример на рисунке 4.2

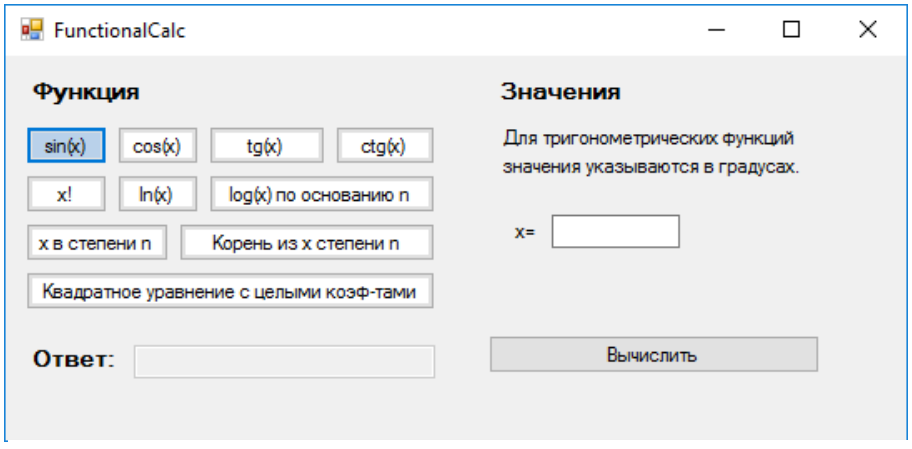

Рис 4.2

Если выберем функцию, где для вычисления нужны две значении, то для этого появляется вторая строка для ввода второй значении. Результат можно увидеть на рисунке 4.3. Так же, есть одна функция, где нужны три значения, и это показано на рисунке 4.4.

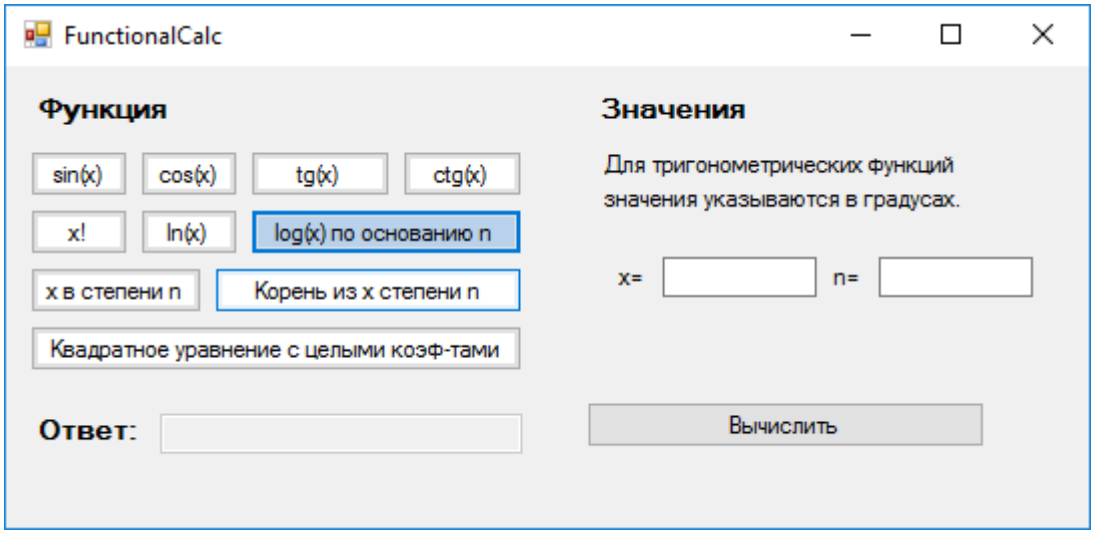

Рис 4.3.

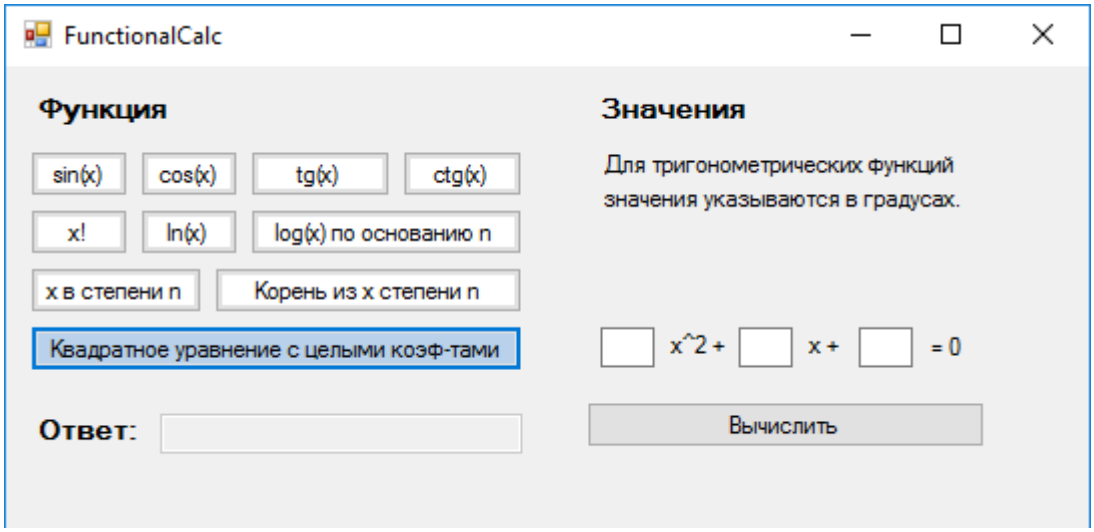

Рис 4.4.

На рисунке 4.5, 4.6, 4.7 изображены вычисления некоторые функции на практике.

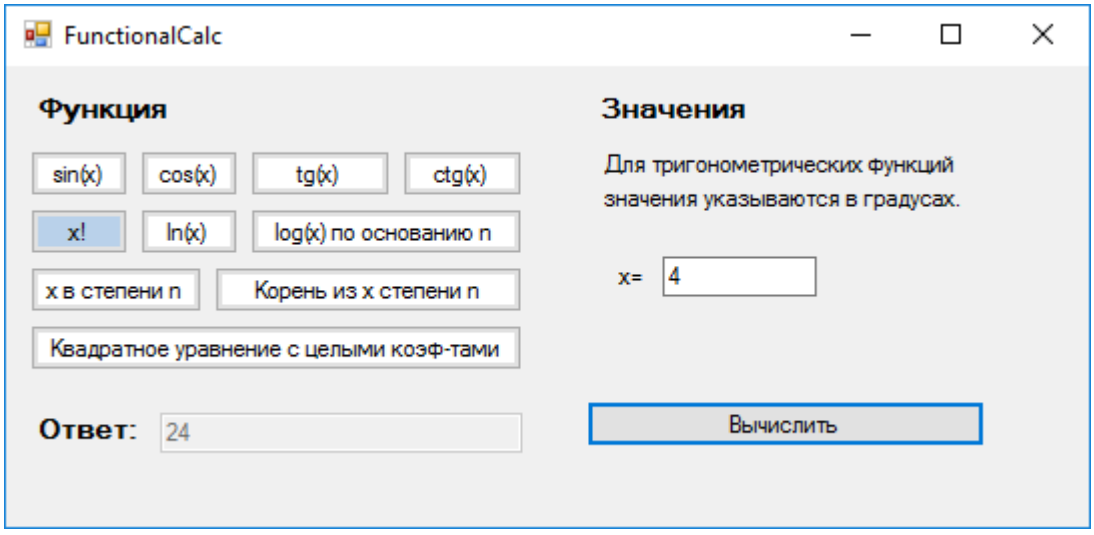

Рис 4.5

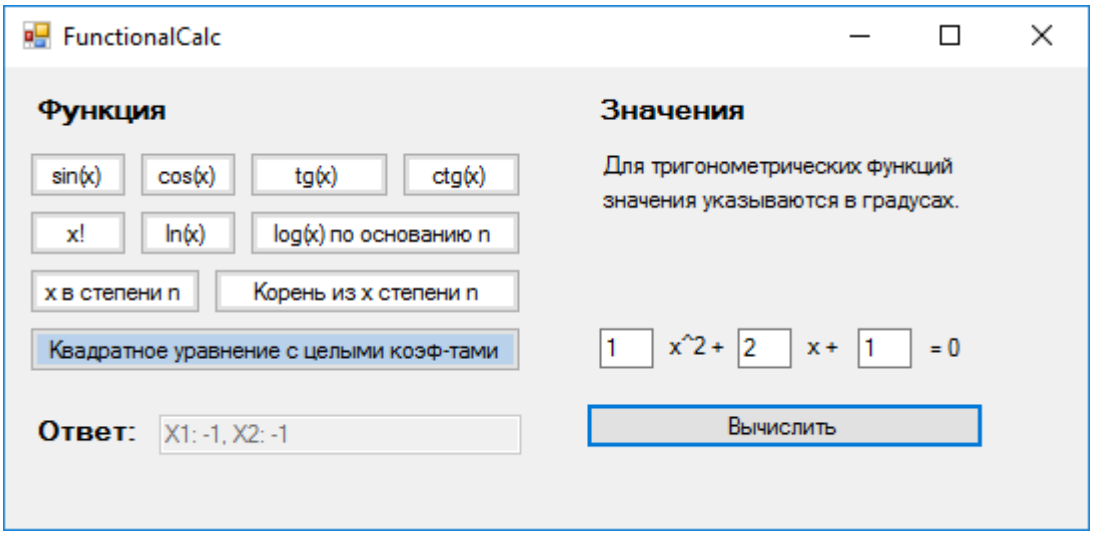

Рис 4.6

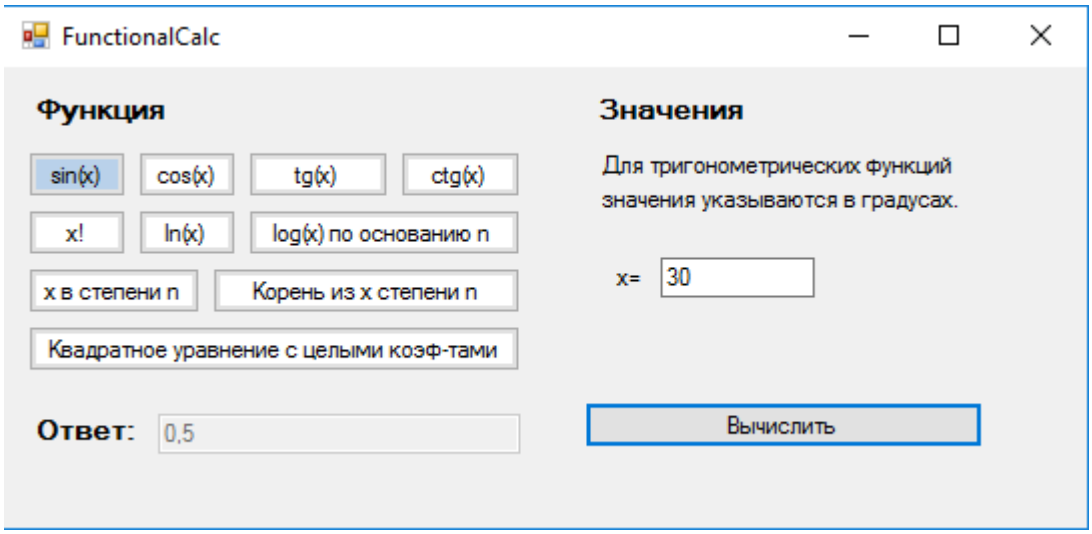

Рис 4.7

В противном случае, если пользователь не вводит значение, то появляется следующее сообщение (рис 4.8).

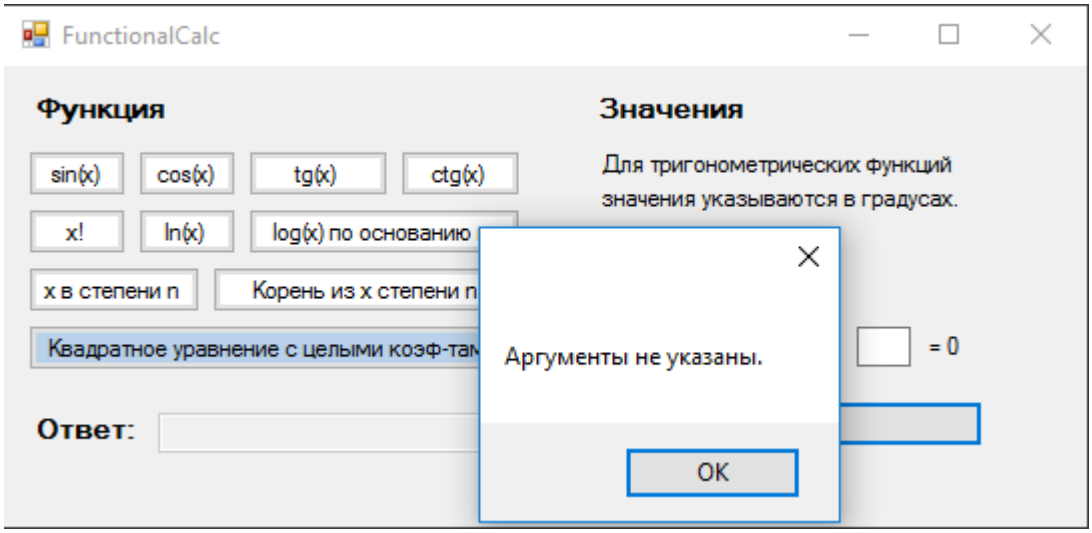

Рис 4.8

4. Заключение

При разработке был создан отдельный абстрактный класс FunctionalCalculator, который хранит все формулы простых функций. Математические функции были реализованы при помощи методов стандартного класса Math. Исключение составляют – факториал и поиск корней квадратного уравнения.

Факториал реализован методом перемножения всех чисел от 1 до указанного, а поиск корней квадратного уравнения – «школьным» способом. (Поиск корня из дискриминанта и применение формулы (-b +- d)/(2*a), где d – корень дискриминанта).

В классе Form1, содержит поле типа Button с названием choosenBtn и начальным значением null. Данное поле является указателем на то, какая кнопка была нажата последней на форме.

Значение в данном поле перезаписывается в обработчиках нажатия кнопок и в последующем используется при определении того, какая функция должна быть посчитана.

В конструкторе формы, помимо стандартной инициализации, вызываются 2 метода HideAllInputs() и SetHandlers(). Первый используется для того, чтобы скрыть все поля для ввода с формы, так как они не понадобятся при инициализации формы.

Второй – присваивает всем кнопкам на форме обработчик anyBtn_Click(), в данном обработчике изменяется выделение кнопки, на которую был произведён клик, а также, производится запись в описанное выше поле choosenBtn.

Все обработчики с программы являются стандартными типа RoutedEventHandler. Каждый из них отображает те поля для ввода, которые необходимы для соответствующей кнопки.

Также, отдельно можно выделить обработчик: calculateBtn_Click

В нём производится определение того, какая кнопка была нажата последней, а также, были ли заполнены поля для ввода значений. Затем, производится считывание значений в полях ввода, а также приведение их к необходимому типу данных.(double или int)

Здесь же, после всех вычислений, производится вывод ответа на форму.

В данном обработчике также производится запрос к описанному выше абстрактному классу FunctionalCalculator для вычисления необходимых функций.

Методы:

ShowXInput

ShowNInput

ShowABCInput

SetHandlers

Манипулируют с видимостью полей для ввода.

Разработанная программа под названием «Функциональный калькулятор» проста и удобна для вычисление простых функций.