**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №2**

**Тема:** Тестирование «Черным ящиком».

**Цель:** Сформировать практические навыки проведения тестирования черным ящиком.

**Ход работы**

1. Составить подробный список требований к приложению «калькулятор»;
2. Написать калькулятор, приближенный по визуальному представлению, к примеру, изначально предусмотрев набор багов;
3. Для дальнейшего выполнения практического задания обменяйтесь требованиями и написанным приложение с соседом.
4. Провести тестирование черного ящика и написать отчет в соответствии с примером.

Перед разработкой приложения «калькулятор» был составлен подробный список требований, разделенных по категориям – «функциональные требования» и «нефункциональные требования».

1. Функциональные требования
2. Основные арифметические операции:

* Сложение (+)
* Вычитание (-)
* Умножение (\* или ×)
* Деление (/ или ÷)

Должна быть предусмотрена обработка деления на ноль (вывод сообщения об ошибке или возврат особого значения).

* Возведение в степень (^ или )

1. Дополнительные операции:

* Извлечение квадратного корня (√)
* Вычисление процента (%)
* Изменение знака (+/-)
* Тригонометрические функции (sin, cos, tan)

1. Работа с числами:

* Ввод целых чисел.
* Ввод десятичных чисел (с плавающей точкой).

1. Пользовательский интерфейс:

* Цифровые кнопки (0-9).
* Кнопки для основных арифметических операций (+, -, \*, /).
* Кнопка для десятичной точки (.).
* Кнопки для дополнительных операций (√, %, +/-, !, sin, cos и т.д.).
* Кнопка «равно» (=) для вывода результата.
* Кнопка «очистить» для сброса ввода.
* Отображение введенного выражения и результата.
* Индикация ошибок (деление на ноль, неверный ввод).

1. Порядок выполнения операций:

Должен следовать стандартному порядку выполнения математических операций (сначала умножение и деление, затем сложение и вычитание).

Поддержка использования скобок для изменения порядка выполнения операций.

1. Нефункциональные требования:
2. Производительность:

Должен быть осуществлен быстрый отклик на действия пользователя, а также эффективное использование ресурсов устройства.

1. Надежность:

* Стабильная работа без ошибок и сбоев.
* Корректные результаты вычислений.
* Устойчивость к некорректному вводу.

1. Удобство использования:

* Интуитивно понятный интерфейс.
* Легкость ввода выражений и чисел.
* Удобное расположение кнопок.
* Хорошая читаемость отображаемой информации.
* Возможность использования на разных устройствах (мобильные, планшеты, ПК).

1. Безопасность:

Должны быть устранены уязвимости, которые могут привести к сбоям или утечке данных.

1. Совместимость:

* Работа на различных операционных системах.
* Корректное отображение на разных разрешениях экрана.

Далее при помощи *MS Visual Studio* было написано приложение с использованием инструментов *Windows Forms*, отвечающее всем вышеперечисленным требованиям и функциям калькулятора. Для проведения тестирования методом «черного ящика» были намеренно добавлены ошибки в программу. Код созданного калькулятора представлен ниже на листинге 1.

Листинг 1. Калькулятор с ошибками

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 thebestcalculator

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

float a, b;

int count;

bool znak = true;

private void calculate()

{

switch (count)

{

case 1:

b = a + float.Parse(textBox1.Text);

textBox1.Text = b.ToString();

break;

case 2:

b = a + float.Parse(textBox1.Text); //ошибка 1: "вместо + должен быть - "

textBox1.Text = b.ToString();

break;

case 3:

b = a \* float.Parse(textBox1.Text);

textBox1.Text = b.ToString();

break;

case 4:

b = a / float.Parse(textBox1.Text);

textBox1.Text = b.ToString();

break;

case 5:

b = (float)Math.Sin(a);

textBox1.Text = b.ToString();

break;

case 6:

b = (float)Math.Cos(a);

textBox1.Text = b.ToString();

break;

case 7:

b = (float)Math.Tan(a);

textBox1.Text = b.ToString();

break;

case 8:

b = (float)(1/Math.Tan(a));

textBox1.Text = b.ToString();

break;

case 9:

b = (float)a/10;//ошибка 3: "вместо 10 должно быть 100"

textBox1.Text = b.ToString();

break;

case 10:

b = (float)Math.Sqrt(b);//ошибка 2: "вместо a должен быть b "

textBox1.Text = b.ToString();

break;

default:

break;

}

// Сбрасываем count, чтобы подготовиться к следующему расчету

count = 0;

a = b; // Переносим результат в a для последующих операций

}

private void button2\_Click(object sender, EventArgs e)

{

int lenght = textBox1.Text.Length - 1;

string text = textBox1.Text;

textBox1.Clear();

for (int i = 0; i < lenght; i++)

{

textBox1.Text = textBox1.Text + text[i];

}

}

private void button11\_Click(object sender, EventArgs e)

{

textBox1.Text = textBox1.Text + 3;

}

private void button10\_Click(object sender, EventArgs e)

{

textBox1.Text = textBox1.Text + 2;

}

private void button9\_Click(object sender, EventArgs e)

{

textBox1.Text = textBox1.Text + 1;

}

private void button8\_Click(object sender, EventArgs e)

{

a = float.Parse(textBox1.Text);

textBox1.Clear();

count = 4;

label2.Text = a.ToString() + ":";

znak = true;

}

private void button7\_Click(object sender, EventArgs e)

{

textBox1.Text = textBox1.Text + 9;

}

private void button6\_Click(object sender, EventArgs e)

{

textBox1.Text = textBox1.Text + 8;

}

private void button5\_Click(object sender, EventArgs e)

{

textBox1.Text = textBox1.Text + 7;

}

private void button4\_Click(object sender, EventArgs e)

{

a = float.Parse(textBox1.Text);

textBox1.Clear();

count = 9;

label2.Text = a.ToString() + "%";

znak = true;

}

private void button3\_Click(object sender, EventArgs e)

{

a = float.Parse(textBox1.Text);

textBox1.Clear();

count = 10;

label2.Text = "√"+a.ToString();

znak = true;

}

private void button1\_Click(object sender, EventArgs e)

{

textBox1.Text = "";

label1.Text = "";

}

private void button12\_Click(object sender, EventArgs e)

{

a = float.Parse(textBox1.Text);

textBox1.Clear();

count = 2;

label2.Text = a.ToString() + "-";

znak = true;

}

private void button16\_Click(object sender, EventArgs e)

{

textBox1.Text = textBox1.Text + - 4;//ошибка 4: "вместо -4 должно быть 0"

}

private void button19\_Click(object sender, EventArgs e)

{

textBox1.Text = textBox1.Text + 5;

}

private void button18\_Click(object sender, EventArgs e)

{

textBox1.Text = textBox1.Text + 6;

}

private void button13\_Click(object sender, EventArgs e)

{

a = float.Parse(textBox1.Text);

textBox1.Clear();

count = 1;

label2.Text = a.ToString() + "+";

znak = true;

}

private void button17\_Click(object sender, EventArgs e)

{

a = float.Parse(textBox1.Text);

textBox1.Clear();

count = 3;

label2.Text = a.ToString() + "\*";

znak = true;

}

private void button14\_Click(object sender, EventArgs e)

{

// Вычисление результата

calculate();

label2.Text = "";

}

private void button24\_Click(object sender, EventArgs e)

{

if (znak == true)

{

textBox1.Text = "-" + textBox1.Text;

znak = false;

}

else if (znak == false)

{

textBox1.Text = textBox1.Text.Replace("-", "");

znak = true;

}

}

private void button20\_Click(object sender, EventArgs e)

{

textBox1.Text = textBox1.Text + "4";

}

private void button23\_Click(object sender, EventArgs e)

{

a = float.Parse(textBox1.Text);

textBox1.Clear();

count = 5; // Устанавливаем count для синуса

znak = true;

label2.Text = "sin"+ a.ToString();

znak = true;

}

private void button22\_Click(object sender, EventArgs e)

{

a = float.Parse(textBox1.Text);

textBox1.Clear();

count = 6; // Устанавливаем count для синуса

znak = true;

label2.Text = "cos" + a.ToString();

znak = true;

}

private void button21\_Click(object sender, EventArgs e)

{

a = float.Parse(textBox1.Text);

textBox1.Clear();

count = 7; // Устанавливаем count для синуса

znak = true;

label2.Text = "tg" + a.ToString();

znak = true;

}

private void button29\_Click(object sender, EventArgs e)

{

a = float.Parse(textBox1.Text);

textBox1.Clear();

count = 8; // Устанавливаем count для синуса

znak = false; //ошибка 5: "должно быть true вместо false"

label2.Text = "ctg" + a.ToString();

znak = false;//ошибка 6: "должно быть true вместо false"

}

private void button15\_Click(object sender, EventArgs e)

{

textBox1.Text = textBox1.Text + ",";

}

}

}

Внешний вид получившегося калькулятора представлен на рисунке 1.

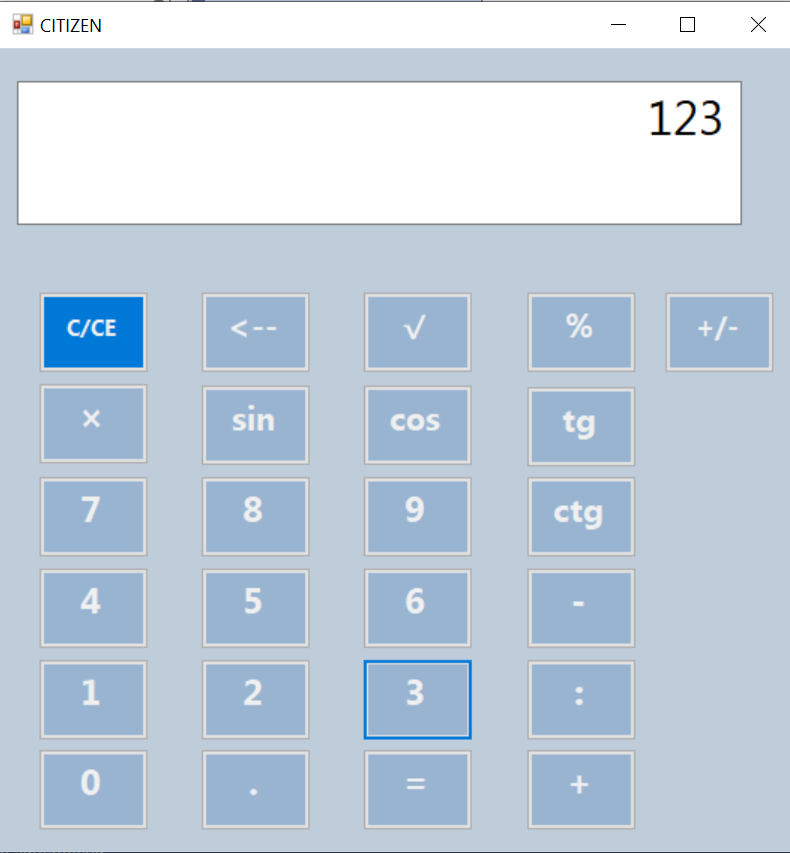


Рисунок 1 – Калькулятор

Затем был выполнен обмен требованиями и написанным приложением с соседом. После этого для дальнейшего выполнения работы необходимо было выполнить тестирование методом черного ящика с полученным приложением.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Название**  **теста** | **Описание сценария** | **Входные данные** | **Выходные данные** | **Удачное/**  **Неудачное тестирование** | **Предложения по исправлению найденных ошибок.** | **Пожелания пользователей** |
| Функция суммы | Сложение двух положительных чисел; Проверка результата | Первая переменная=3  Вторая переменная=8 | Результат=11 | Неудачное | - | Поле для ввода значений и вывода, объединить |

**Вывод:** Были сформированы практические навыки проведения тестирования черным ящиком.

**Ответы на контрольные вопросы:**

1. Что такое «тестирование черного ящика»?

Ответ: Тестирование черного ящика – это метод тестирования программного обеспечения без доступа к коду, при котором тестировщик проверяет приложение как пользователь.

1. Назовите основные особенности тестирования «черного ящика».

Ответ: Основные особенности тестирования «черного ящика» заключаются в следующем:

* фокусировка на функциональных требованиях и поведении системы;
* тестирование проводится с точки зрения конечного пользователя;
* основано на спецификациях, требованиях и документации.

1. Каковы распространенные техники тестирования «черного ящика»?

Ответ: К распространенным техникам тестирования «черного ящика» относят:

* разделение на классы эквивалентности;
* анализ граничных значений;
* таблицы принятия решений;
* граф причинно-следственных связей;
* тестирование переходов состояний;
* тестирование на основе вариантов использования.

1. Как эффективно провести *BlackBox Testing*?

Ответ: Для эффективного проведения тестирования «черного ящика» необходимо пройти несколько этапов:

* + - 1. Четко изучить функциональные и нефункциональные требования к ПО.
      2. Создать исчерпывающие тестовые наборы, покрывающие все сценарии и требования.
      3. Применять различные техники тестирования (эквивалентное разделение, граничные значения, таблицы решений и т.д.).
      4. Смотреть на систему как пользователь, не зная её внутреннего устройства.
      5. Тщательно записать все найденные дефекты, шаги для их воспроизведения и результаты тестов.
      6. Проводить регрессионное тестирование после исправления дефектов.
      7. Если возможно, автоматизировать рутинные тесты для повышения эффективности и снижения затрат времени.
      8. Регулярно анализировать тестовое покрытие, чтобы определить, какие области системы еще не протестированы.

1. Каковы преимущества *Black box testing*?

Ответ:

1. В чем заключаются недостатки *Black box testing*?

Ответ:

1. Каковы парадигмы тестирования методом черного ящика?

Ответ: