Лабораторная работа №9

**Модульное тестирование**

Задание 1.

Написать тесты к каждой функции калькулятора, включая целые и дробные числа (8 тестов, 1й уже готов качестве примера).

В функции деления проверить, вызывается ли исключение «DivideByZeroException» при делении на 0. (ещё один тест)

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using LB9\_UnitTests.TasksModules;

namespace LB9\_Tests

{

[TestClass]

public class Task1

{

//готовый тест в качестве примера (сумма целых чисел)

[TestMethod] //указывает, что метод ниже - тестовый

public void sumTestInteger()

{

// arrange

var calc = new Calc(); //записываем в переменную модуль с функциями калькулятора

double value1 = 2; //число 1

double value2 = 3; //число 2

double expected = 5; //ожидаемый результат (правильный)

// act

double result = calc.sum(value1,value2); // отправляем число 1 и число 2 в функцию и записываем ответ

// assert

Assert.AreEqual(expected, result); //сравниваем ожидаемый результат с полученным ответом

}

[TestMethod]

public void sumTestDouble()

{

// arrange

var calc = new Calc(); //записываем в переменную модуль с функциями калькулятора

double value1 = 2.2; //число 1

double value2 = 3.3; //число 2

double expected = 5.5; //ожидаемый результат (правильный)

// act

double result = calc.sum(value1, value2); // отправляем число 1 и число 2 в функцию и записываем ответ

// assert

Assert.AreEqual(expected, result); //сравниваем ожидаемый результат с полученным ответом

}

[TestMethod]

public void diffTestInteger()

{

// arrange

var calc = new Calc(); //записываем в переменную модуль с функциями калькулятора

double value1 = 2; //число 1

double value2 = 3; //число 2

double expected = -1; //ожидаемый результат (правильный)

// act

double result = calc.diff(value1, value2); // отправляем число 1 и число 2 в функцию и записываем ответ

// assert

Assert.AreEqual(expected, result); //сравниваем ожидаемый результат с полученным ответом

}

[TestMethod]

public void diffTestDouble()

{

// arrange

var calc = new Calc(); //записываем в переменную модуль с функциями калькулятора

double value1 = 2.84; //число 1

double value2 = 3.8; //число 2

double expected = -0.96; //ожидаемый результат (правильный)

// act

double result = calc.diff(value1, value2); // отправляем число 1 и число 2 в функцию и записываем ответ

// assert

Assert.AreEqual(expected, result); //сравниваем ожидаемый результат с полученным ответом

}

[TestMethod]

public void multiInteger()

{

// arrange

var calc = new Calc(); //записываем в переменную модуль с функциями калькулятора

double value1 = 2; //число 1

double value2 = 3; //число 2

double expected = 6; //ожидаемый результат (правильный)

// act

double result = calc.multi(value1, value2); // отправляем число 1 и число 2 в функцию и записываем ответ

// assert

Assert.AreEqual(expected, result); //сравниваем ожидаемый результат с полученным ответом

}

[TestMethod]

public void multiDouble()

{

// arrange

var calc = new Calc(); //записываем в переменную модуль с функциями калькулятора

double value1 = 2.2; //число 1

double value2 = 3.3; //число 2

double expected = 7.26; //ожидаемый результат (правильный)

// act

double result = calc.multi(value1, value2); // отправляем число 1 и число 2 в функцию и записываем ответ

// assert

Assert.AreEqual(expected, result); //сравниваем ожидаемый результат с полученным ответом

}

[TestMethod]

public void divInteger()

{

// arrange

var calc = new Calc(); //записываем в переменную модуль с функциями калькулятора

double value1 = 6; //число 1

double value2 = 3; //число 2

double expected = 2; //ожидаемый результат (правильный)

// act

double result = calc.div(value1, value2); // отправляем число 1 и число 2 в функцию и записываем ответ

// assert

Assert.AreEqual(expected, result); //сравниваем ожидаемый результат с полученным ответом

}

[TestMethod]

public void divDouble()

{

// arrange

var calc = new Calc(); //записываем в переменную модуль с функциями калькулятора

double value1 = 6.6; //число 1

double value2 = 1; //число 2

double expected = 6.6; //ожидаемый результат (правильный)

// act

double result = calc.div(value1, value2); // отправляем число 1 и число 2 в функцию и записываем ответ

// assert

Assert.AreEqual(expected, result); //сравниваем ожидаемый результат с полученным ответом

}

[TestMethod]

[ExpectedException(typeof(DivideByZeroException), "Divide by zero exeption")]

public void divideByZeroException()

{

var calc = new Calc();

double value1 = 2;

double value2 = 0.0;

calc.div(value1, value2);

}

}

}

Задание 2.

Написать 3 теста к функции timeStrToNumerical.

Написать 10 тестов к функции timeConverter. В работе данной функции имеется минимум два бага – найти их и написать к ним по тесту.

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using LB9\_UnitTests.TasksModules;

namespace LB9\_Tests

{

[TestClass]

public class Task2

{

[TestMethod]

public void meme()

{

string str = "34:29";

TimeConverter memes = new TimeConverter();

memes.timeStrToNumerical(str, out int a, out int b);

int c = 34, d = 29;

Assert.AreEqual(a, c);

Assert.AreEqual(b, d);

}

[TestMethod]

public void meme1()

{

string str = "3.2:4.2";

TimeConverter memes = new TimeConverter();

memes.timeStrToNumerical(str, out int a, out int b);

string s = "System.FormatException";

Assert.AreEqual(a, s);

Assert.AreEqual(b, s);

}

[TestMethod]

public void meme2()

{

string str = "3:4";

TimeConverter memes = new TimeConverter();

memes.timeStrToNumerical(str, out int a, out int b);

int c = 3, d = 4;

Assert.AreEqual(a, c);

Assert.AreEqual(b, d);

}

[TestMethod]

public void mem()

{

int a = 30, b = 30;

TimeConverter memes = new TimeConverter();

string exit = memes.timeConverter(a, b);

string c = "half past half past";

Assert.AreEqual(exit, c);

}

[TestMethod]

public void mem1()

{

int a = 0, b = 20;

TimeConverter memes = new TimeConverter();

string exit = memes.timeConverter(a, b);

string c = "zero o'clock";

Assert.AreEqual(exit, c);

}

[TestMethod]

public void mem2()

{

int a = 20, b = 25;

TimeConverter memes = new TimeConverter();

string exit = memes.timeConverter(a, b);

string c = "twenty two minutes to twenty";

Assert.AreEqual(exit, c);

}

[TestMethod]

public void mem3()

{

int a = 20, b = 22;

TimeConverter memes = new TimeConverter();

string exit = memes.timeConverter(a, b);

string c = "twenty two minutes to twenty";

Assert.AreEqual(exit, c);

}

[TestMethod]

public void mem4()

{

int a = 20, b = 23;

TimeConverter memes = new TimeConverter();

string exit = memes.timeConverter(a, b);

string c = "twenty three minutes to twenty";

Assert.AreEqual(exit, c);

}

[TestMethod]

public void mem5()

{

int a = 20, b = 24;

TimeConverter memes = new TimeConverter();

string exit = memes.timeConverter(a, b);

string c = "twenty four minutes to twenty";

Assert.AreEqual(exit, c);

}

[TestMethod]

public void mem6()

{

int a = 20, b = 29;

TimeConverter memes = new TimeConverter();

string exit = memes.timeConverter(a, b);

string c = "twenty nine minutes to twenty";

Assert.AreEqual(exit, c);

}

[TestMethod]

public void mem7()

{

int a = 20, b = 28;

TimeConverter memes = new TimeConverter();

string exit = memes.timeConverter(a, b);

string c = "twenty eight minutes to twenty";

Assert.AreEqual(exit, c);

}

[TestMethod]

public void mem8()

{

int a = 20, b = 27;

TimeConverter memes = new TimeConverter();

string exit = memes.timeConverter(a, b);

string c = "twenty seven minutes to twenty";

Assert.AreEqual(exit, c);

}

[TestMethod]

public void mem9()

{

int a = 20, b = 26;

TimeConverter memes = new TimeConverter();

string exit = memes.timeConverter(a, b);

string c = "twenty six minutes to twenty";

Assert.AreEqual(exit, c);

}

}

}

В тестах, которые выдают ошибку, ожидалась ошибка