**Практическая работа**

**«Модульное тестирование по вариантам»**

**Цель работы:** Научиться разрабатывать модульные тесты для кода, написанного на языке C#

**Порядок выполнения работы:**

1. Создать с помощью Visual Studio консольный проект C# (Console Application).

2. Создать класс (согласно своему варианту), который будет имитировать тестируемую логику.

3. Создать в проекте тестовый класс.

4. Разработать не менее пяти тестирующих функций. При разработке этих функций следует активно применять функции класса «Assert»

5. Разработать тестовые методы со спецификацией ожидаемых исключений.

6. Запустить проверку разработанного тестового класса. Скомпилировать и запустить проект. Посмотреть на результат теста.

7. Внести в тестируемые классы изменения, приводящие к ошибкам. Скомпилировать и запустить проект.

8. Посмотреть, пойманы ли ошибки модульным тестом.

**1. Краткие теоретические сведения**

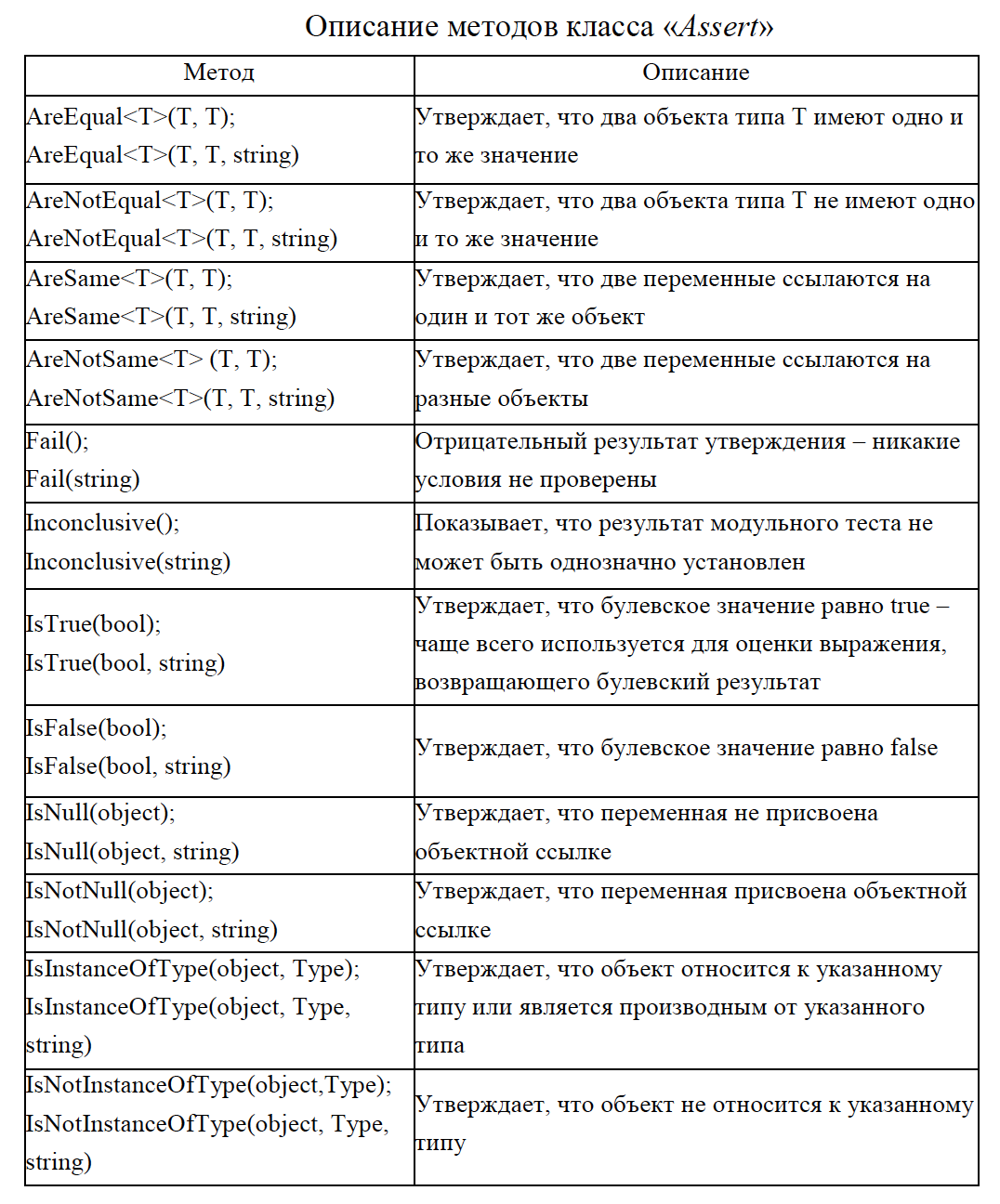
Пространство имен Microsoft.VisualStudio.TestTools.UnitTesting содержит классы для работы модульных тестов.

В классе «*Assert*» определен набор статических методов, которые можно использовать в тестах. Класс «*Assert*» является одним из самых часто применяемых. Описание методов представлено в таблице ниже.

**Каждый статический метод в классе «*Assert*» позволяет проверить какой-то аспект модульного теста, и если проверка не проходит, эти методы генерируют исключение. Чтобы модульный тест прошел, все утверждения должны завершиться успешно.**

***Каждый метод в классе «Assert» имеет перегруженную версию, которая принимает параметр string. В случае отрицательного результата утверждения эта строка помещается в элемент сообщения внутри объекта исключения.***

Методы *AreEqual()* и *AreNotEqual()* имеют несколько перегруженных версий, предназначенных для сравнения специфических типов. Например, существует версия, которая позволяет сравнивать строки без учета регистра символов.



Напишем тест для метода, который вычисляет квадратный корень для заданного числа.

namespace Class Ass

{

public class MyClass

{

public static double GetSgrt(double value)

{

return Math.Sqrt(value);

}

}

}

При тестировании данного метода следует учесть два варианта решений, когда при вычислении квадратного корня мы получаем целое значение ‒ метод *IsSqrtTest*(), и когда проверка значений на равенство выполняется с учетом погрешности, в случае если решение содержит дробную часть, метод *DeltaTest()*.

namespace ClassAssTest

{

[TestClass]

public class MyClassTest

{

[TestMethod]

public void IsSqrtTest()

{

// arrange

const double input = 4;

const double expected = 2;

//act

double actual = ClassAss.MyClass.GetSgrt(input);

//assert

Assert.AreEqual(expected, actual, "Sqrt of {0} should have been {1}!", input, expected);

}

[TestMethod]

public voidDeltaTest()

{

// arrange

const double expected = 3.1;

const double delta = 0.07;

//act

double actual = ClassAss.MyClass.GetSgrt(10);

//assert

//проверка значений на равенство с учетом погрешности

Assert.AreEqual(expected, actual, delta, "fail mtssege!");

}

}

}

Класс «*TestContext*» используется для хранения информации, передаваемой для модульных тестов. Рассмотрим работу данного класса на примере.

namespace ContexUnitTest

{

[TestClass]

public classUnitTest1

{

public TestContextTestContext { get; set; } // свойство

[TestMethod]

public void TestMethod1()

{

TestContext.WriteLine("TestContext.TestRunDirectory{0}",TestContext.TestRunDirectory);

TestContext.WriteLine("TestName{0}", TestContext.TestName);

TestContext.WriteLine("CurrentTestOutcomey{0}", TestContext.CurrentTestOutcome);// результат работы теста

}

[TestCleanup]

public void TestCleanup()

{

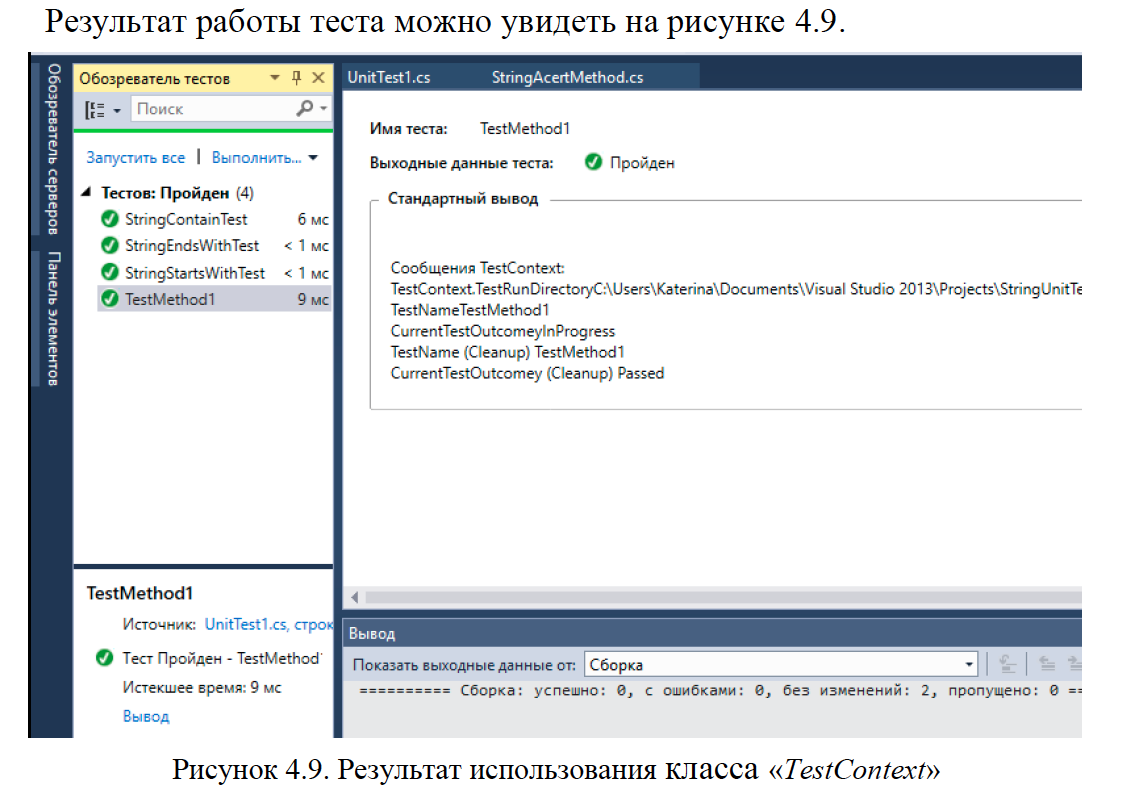
TestContext.WriteLine("TestName (Cleanup) {0}", TestContext.TestName);

TestContext.WriteLine("CurrentTestOutcomey (Cleanup) {0}", TestContext.CurrentTestOutcome);

}

}

}



Дополнительные атрибуты теста:

[ClassInitialize()] используется для выполнения кода до выполнения первого теста в классе.

[ClassCleanUp()] используется для выполнения кода после завершения выполнения всех тестов в классе.

[TestInitialize()] используется для выполнения кода до выполнения каждого теста.

[TestCleanUp()] используется для выполнения кода после завершения выполнения каждого теста.

**Варианты заданий:**

Вычислить и вывести на экран значения функции F в соответствии с вариантом задания. Через *a, b, c, x* – обозначены действительные числа.

