**Лабораторная работа №7: автоматизированные тесты**

**Цель**

Создать автоматизированные тесты (юнит-тесты) для проверки программного кода своей библиотеки.

**Задание**

**Подготовительная часть**

Создать библиотеку, реализующую функционал сжатия массива байт RLE. Библиотека должна содержать следующие методы

* byte[] Pack ( byte[] data ) – сжимает набор байт алгоритмом RLE, возвращая упакованные данные;
* byte[] Unpack ( byte[] data ) – распаковывает набор данных.

Пример вызова Pack: если на входе 0xaa 0xaa 0xaa, то на выходе 0x03 0xaa

Пример вызова Unpack: если на входе 0x02 0xaa 0x01 0xbb то на выходе 0xaa 0xaa 0xbb.

Допускается любая реализация сжатия и распаковки, эффективность в данном случае не проверяется.

**Основная часть задания**

Написать юнит-тесты, запускаемые в Visual Studio для проверки функционала библиотеки. Набор тестов и конкретный проверяемый функционал остаётся на усмотрение студента.

Пример тестов:

* Передать недопустимые параметры, которые программа должна уметь обрабатывать без выброса исключения. Проверить, что при этом не будет выдано исключения.
* Передать параметры, ответ программы на которые заведомо известен. Проверить, что возвращаемый ответ равен ожидаемому.
* Проверить корректность работы программы при специально подобранных данных. Например, передать больше 255 одинаковых символов на сжатие, и проверить, что они упаковались корректно.
* Передать недопустимые параметры, которые программа умеет обрабатывать с выбросом исключения. Проверить, что выброшено действительно то исключение, которое ожидалось.

Необходимо разработать как минимум 5 тестов.

**Демонстрация работы**

Продемонстрировать программный код библиотеки и тестов. Запустить тесты.

**Вспомогательные материалы**

Visual Studio содержит отдельный тип проекта для тестов, который доступен в окне создания проекта по пути Visual C# \ Тест \ Проект модульного теста. Проект является обычной библиотекой классов, в которых каждый метод помечен особым атрибутом [TestMethod], который используется Visual Studio как индикатор того, что метод надо запускать в качестве теста. Для запуска тестов можно воспользоваться пунктом меню Тест \ Выполнить \ Все тесты. При выборе этого пункта Visual Studio загружает сборку с тестами и выполняет все методы, помеченные атрибутом [TestMethod]. Результат запуска тестов отображается в отдельном окне. Методам, реализующим тесты, необходимо выдавать названия, говорящие о проверяемом функционале, чтобы можно было быстро выявить проверяемый функционал.

Пример теста:

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace UnitTestProject1 {

[TestClass]

public class UnitTest1 {

[TestMethod]

[Timeout(1000)]

public void CheckStringLength() {

Assert.AreEqual(new HuffmanTest().GetStringLength("abc"), 3);

}

[TestMethod]

public void CheckMaxThreads() {

PrivateObject po = new PrivateObject(new HuffmanTest());

Assert.AreEqual(9, po.Invoke("GetMaximumThreads"));

}

}

}

Если тестовый метод выполнился без исключения, то считается, что тест успешен. Если произошло исключение, то Visual Studio сообщит об ошибке теста. Пространство имён Microsoft.VisualStudio.TestTools.UnitTesting содержит ряд удобных функций, выбрасывающих исключение в зависимости от своих параметров. Например:

* Assert.AreEqual
* Assert.IsTrue
* Assert.Fail

Как правило, тестирование выполняется только для открытых методов классов. Однако возможно тестировать и закрытые методы с помощью рефлексии или специального класса PrivateObject.