Практическая работа №2. Разработка класса на C# и модульное тестирование.

Задание. Реализуйте пользовательский тип `Matrix` для работы с матрицами произвольного размера. Проверку функциональности необходимо реализовать с помощью модульных тестов.

Функциональность по вариантам представлена ниже в таблице, в тексте выделена желтым цветом.

Вариант: V = (F + N) % 7, где F – первая буква фамилии в верхнем регистре на английском языке, N – первая буква имени в верхнем регистре на английском языке.

Вариант	Хранение данных	Свойства	Операции	Методы	Модульные тесты
0	double[][]	IsSymmetric	m1 + m2	Trace, Parse	MSUnit
1	double[,]	IsDiagonal	m1 – m2	Transpose, Parse	NUnit
2	double[]	IsSymmetric	m1 * m2	Trace, TryParse,	MSUnit
3	double[][]	IsDiagonal	m1 * K	Transpose, TryParse	NUnit
4	double[,]	IsSymmetric	m1 + m2	Transpose, Parse	MSUnit
5	double[]	IsDiagonal	m1 – m2	Trace, Parse	NUnit
6	double[][]	IsSymmetric	m1 * m2	Transpose, TryParse	MSUnit

Рекомендации

Данные. Структура для хранения данных (элементов матрицы) может быть реализована в виде прямоугольного массива (`double[,] data`), массива массивов (`double[][] data`) или одномерного массива (`double[] data`).

Внешний пользователь типа Matrix не должен напрямую работать с внутренней структурой данных.

Конструкторы. Реализуйте несколько конструкторов для инициализации нового объекта:

```
public Matrix(int nRows, int nCols) { .. }
public Matrix(double[,] initData) { ..}
```

Свойства. Реализуйте доступ к элементам матрицы через свойство-индексатор:

```
public double this[int i, int j] {..}
```

Размер матрицы доступен только для чтения через свойства:

```
public int Rows {.. }
public int Columns { .. }
// размер квадратной матрицы
public int? Size { .. }
```

Реализуйте несколько булевых свойств:

```
// Является ли матрица квадратной
public bool IsSquared { .. }
```

```
// Является ли матрица нулевой
public bool IsEmpty { .. }
// Является ли матрица единичной
public bool IsUnity { .. }
// Является ли матрица диагональной
public bool IsDiagonal { .. }
// Является ли матрица симметричной
public bool IsSymmetric { .. }
```

Операторы. Реализуйте стандартные матричные операции через перегрузку операторов:

```
public static Matrix operator+(Matrix m1, Matrix m2)
public static Matrix operator-(Matrix m1, Matrix m2)
public static Matrix operator*(Matrix m1, double d)
public static Matrix operator*(Matrix m1, Matrix m2)
```

Реализуйте операторы преобразования типов:

```
public static explicit operator Matrix(double[,] arr)
```

Методы. Реализуйте экземплярные методы для транспонирования матрицы и для вычисления следа матрицы:

```
public Matrix Transpose() {.. }
public double Trace() { ..}
```

Реализуйте переопределение метода `ToString` для преобразования матрицы в строку:

```
public override string ToString() { .. }
```

Статические методы. Реализуйте статические методы для порождения единичной и нулевой матрицы определенного размера:

```
public static Matrix GetUnity(int Size) { .. }
public static Matrix GetEmpty(int Size) { .. }
```

Реализуйте статические методы для создания матрицы по строчке в определенном формате.

```
public static Matrix Parse(string s) { .. }
public static bool TryParse(string s, out Matrix m){ ..}
```

Формат строки может быть таким: элементы строчки матрицы разделены пробелами, после запятой начинается новая строчка. Например,

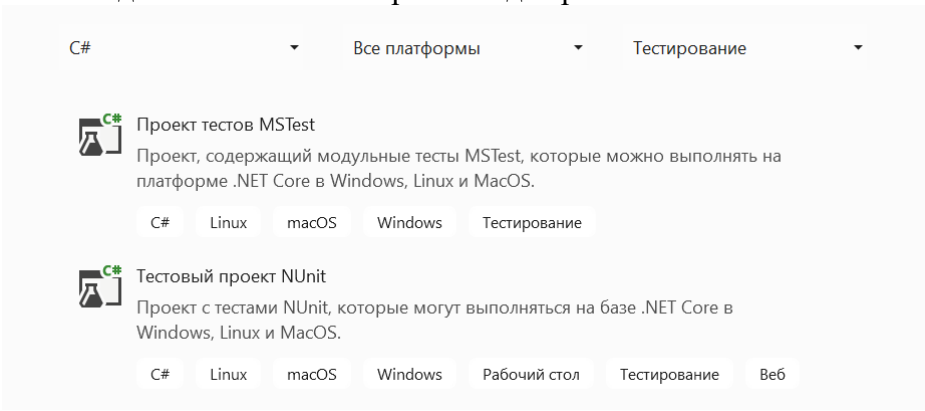
«1 2 3, 4 5 6, 7 8 9»

Для разбиения строки на подстроки используйте метод `Split` класса `String`.

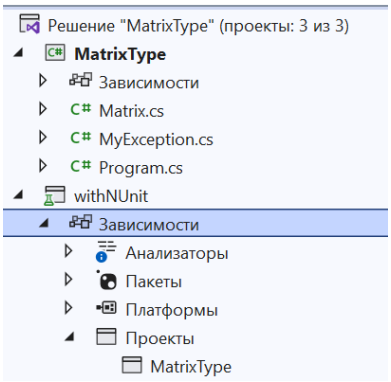
Метод `Parse` должен генерировать исключение `FormatException`, если формат некорректный.

Модульное тестирование класса Matrix

Для проверки реализованной функциональности предлагается ввести отдельный проект с модульными тестами. Можно использовать библиотеку MSTest или NUnit. В Visual Studio есть отдельные шаблоны проектов для работы с этими библиотеками:



Библиотеки предполагают подключение проекта с классом Matrix как зависимости:



В тестовом проекте вводится класс, содержащий тестовые методы. Каждый метод предназначен для тестирования отдельной функциональности класса Matrix. Например, в следующем методе проверим, что при создании матрицы происходит корректная установка свойств матрицы:

```
[TestClass]
Ссылок: 0
public class UnitTest1
{
    [TestMethod]
    Ссылок: 0
    public void CreateCertainMatrix()
    {
        Matrix m = new Matrix(3, 5);
        Assert.AreEqual(3, m.Rows);
        Assert.AreEqual(5, m.Columns);
    }
```

С помощью дополнительных атрибутов можно настроить модульный тест на серию экспериментов с определенными параметрами метода.

В практической работе необходимо ввести модульные тесты, которые покрывают всю реализованную функциональность класса Matrix.