# Цель

Сформировать практические навыки разработки и модульного тестирования классов.

# Задание

Разработайте класс Матрица (Matrix) для операций матричной алгебры в соответствии с предложенной ниже спецификацией требований.

Разработайте тестовые наборы для тестирования методов класса на основе заданной спецификации.

Выполните модульное тестирование класса средствами модульного тестирования Visual Studio.

Выполните анализ покрытия кода методов тестами.

## Спецификация типа данных Матрица

**ADT** Matrix

**Данные**

Матрица (тип Matrix) - это двумерная матрица со значениями целого типа.

Объект типа Матрица - изменяемый.

**Операции**

|  |  |
| --- | --- |
| ***Конструктор (Matrix)*** | |
| Вход: | Число строк i и столбцов j |
| Предусловия: | Число строк и столбцов должно быть больше 0. В противном случае выбрасывается исключение. |
| Процесс: | Создаёт объект типа Matrix с заданным числом строк и столбцов. Заносит число строк и столбцов в соответствующие свойства I и J. |
| Выход | Нет. |
| Постусловия | Создаётся массив I\*J, элементы которого инициированы 0. |
| ***Конструктор (Matrix)*** | |
| Вход: | (b) – объект тип Matrix. |
| Предусловия: | Число строк и столбцов в суммируемых матрицах должны совпадать. В противном случае выбрасывается исключение. |
| Процесс: | Создаёт новый объект типа Matrix, элементы которого, получены путём сложения элементов объектов this и b с одинаковыми индексами. |
| Выход | Объект типа Matrix. |
| Постусловия | Нет. |
| ***Вычесть (operator-)*** | |
| Вход: | (b) – объект тип Matrix. |
| Предусловия: | Число строк и столбцов в матрицах, участвующих в вычитании, должны совпадать. В противном случае  выбрасывается исключение. |
| Процесс: | Создаёт новый объект типа Matrix, элементы которого, получены путём вычитания элементов объектов this и b с одинаковыми индексами. |
| Выход | Объект типа Matrix. |
| Постусловия | Нет. |
| ***Умножить (operator\*)*** | |
| Вход: | (b) – объект тип Matrix. |
| Предусловия: | Матрицы, участвующие в умножении, должны быть согласованы для этой операции по числу строк и столбцов. В противном случае выбрасывается исключение. |
| Процесс: | Создаёт новый объект типа Matrix, элементы которого, получены путём умножения элементов объектов this и b в соответствии с правилами перемножения матриц |
| Выход | Объект типа Matrix. |
| Постусловия | Нет. |
| ***Равно (operator==)*** | |
| Вход: | (b) – объект тип Matrix. |
| Предусловия: | Число строк и столбцов в матрицах, участвующих в вычитании, должны совпадать В противном случае выбрасывается исключение. |
| Процесс: | Возвращает значение true, если элементы объектов this и b в на одинаковых позициях равны. |
| Выход | Объект типа bool. |
| Постусловия | Нет. |
| ***Транспонировать (Transp)*** | |
| Вход: | Нет. |
| Предусловия: | Матрица, подвергаемая транспонированию, должна иметь одинаковое число строк и столбцов. В противном случае выбрасывается исключение. |
| Процесс: | Создаёт новый объект типа Matrix, элементы которого, получены путём транспонирования элементов объекта this. |
| Выход | Объект типа Matrix. |
| Постусловия | Нет. |
| ***Минимальный элемент(Min)*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Отыскивает и возвращает минимальный среди элементов объекта this. |
| Выход | Объект типа int. |
| Постусловия | Нет. |
| ***ПреобразоватьВстроку(ToString)*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Преобразует элементы матрицы this в строковое представление построчно.  Напрмер: {{1,2,3},{4,5,6},{7,8,9}} |
| Выход | Нет. |
| Постусловия | Нет. |
| ***Взять элемент с индексами i,j (this [i,j])*** | |
| Вход: | Значения i, j типа int. |
| Предусловия: | Значения i, j должны находиться в допустимых диапазонах. |
| Процесс: | Возвращает элемент матрицы с индексами i,j. |
| Выход | Значение типа int. |
| Постусловия | Нет. |
| ***Записать элемент с индексами i,j (this [i,j])*** | |
| Вход: | Значения i, j типа int, n – новое значение элемента типа int. |
| Предусловия: | Значения i, j должны находиться в допустимых диапазонах. |
| Процесс: | Заменяет элемент матрицы с индексами i,j на значение n. |
| Выход | Нет. |
| Постусловия | Элемент матрицы с индексами i,j получает значение n. |

# Исходные тексты программ на языке C#

Тестируемый класс

Matrix.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace ConsoleApp1

{

public class MyException : ArgumentException

{

public MyException(string mes) : base(mes)

{ }

}

public class Matrix

{

int[,] matrix;

private int I, J;

public int[] Property

{

get

{

int[] property = { I, J };

return property;

}

}

// Конструктор

public Matrix(int i, int j)

{

if (i <= 0 || j <= 0)

throw new MyException("Число стобцов или строк матрицы должно быть больше 0");

this.matrix = new int[i, j];

this.I = i;

this.J = j;

for (int r = 0; r < this.I; r++)

for (int c = 0; c < this.J; c++)

matrix[r, c] = 0;

}

// Сложение

public static Matrix operator +(Matrix A, Matrix B)

{

if (A.I != B.I)

throw new MyException("Число строк в матрицах должно совпадать!");

if (A.J != B.J)

throw new MyException("Число столбцов в матрицах должно совпадать!");

Matrix C = new Matrix(A.I, A.J);

for (int i = 0; i < A.I; i++)

for (int j = 0; j < A.J; j++)

C.matrix[i, j] = A.matrix[i, j] + B.matrix[i, j];

return C;

}

// Вычитание

public static Matrix operator -(Matrix A, Matrix B)

{

if (A.I != B.I)

throw new MyException("Число строк в матрицах должно совпадать!");

if (A.J != B.J)

throw new MyException("Число столбцов в матрицах должно совпадать!");

Matrix C = new Matrix(A.I, A.J);

for (int i = 0; i < A.I; i++)

for (int j = 0; j < A.J; j++)

C.matrix[i, j] = A.matrix[i, j] - B.matrix[i, j];

return C;

}

// Умножение

public static Matrix operator \*(Matrix A, Matrix B)

{

if (A.J != B.I)

throw new MyException("Число столбоцов в первой матрице должно быть равно числу строк во второй матрице!");

Matrix C = new Matrix(A.I, B.J);

for (int i = 0; i < A.I; i++)

for (int j = 0; j < B.J; j++)

for (int k = 0; k < B.I; k++)

C.matrix[i, j] += A.matrix[i, k] \* B.matrix[k, j];

return C;

}

// равно

public static bool operator ==(Matrix A, Matrix B)

{

if (A.I != B.I || A.J != B.J)

throw new MyException("Число строк и/или столбцов не совпадает!");

bool flag = true;

for (int i = 0; i < A.I; i++)

for (int j = 0; j < A.J; j++)

if (A.matrix[i, j] != B.matrix[i, j])

{

flag = false; break;

}

return flag;

}

public static bool operator !=(Matrix a, Matrix b) => !(a == b);

// Транспонировать

public Matrix Transp()

{

if (I != J)

throw new MyException("Матрица не квадратная!");

Matrix B = new Matrix(I, J);

for (int j = 0; j < J; j++)

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

B.matrix[j, i] = matrix[i, j];

return B;

}

// минимальный элемент

public int Min()

{

int min = matrix[0, 0];

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

for (int j = 0; j < J; j++)

if (matrix[i, j] < min)

min = matrix[i, j];

return min;

}

// Преобразование матрицы в строку

public override string ToString()

{

string[] matrixToString = new string[I];

for (int i = 0; i < this.I; i++)

{

int[] vector = new int[J];

for (int j = 0; j < J; j++)

vector[j] = this.matrix[i, j];

matrixToString[i] = "{" + string.Join(",", vector) + "}";

}

return "{" + String.Join(",", matrixToString) + "}";

}

// Взятие элемента с позиции i и j

// Запись элемента на позицию i и j

public int this[int i, int j]

{

get

{

if (i < 0 || i >= I || j < 0 || j >= J)

throw new MyException("Индексы вышли за диапазон допустимых значений");

return matrix[i, j];

}

set

{

if (i < 0 || i >= I || j < 0 || j >= J)

throw new MyException("Индексы вышли за диапазон допустимых значений");

matrix[i, j] = value;

}

}

}

}

MatrixConstructorTests.cs:

using Microsoft.VisualStudio.TestTools.UnitTesting;

using ConsoleApp1;

namespace MatrixTests

{

[TestClass]

public class MatrixConstructorTests

{

[TestMethod]

[ExpectedException(typeof(MyException))]

public void iLessThen0\_Expected\_MyException()

{

//arrange

int I = -1;

int J = 1;

//act

new Matrix(I, J);

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void JLessThen0\_Expected\_MyException()

{

//arrange

int I = 1;

int J = -1;

//act

new Matrix(I, J);

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void iEqual0\_Expected\_MyException()

{

//arrange

int I = 0;

int J = 1;

//act

new Matrix(I, J);

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void jEqual0\_Expected\_MyException()

{

//arrange

int I = 1;

int J = 0;

//act

new Matrix(I, J);

}

[TestMethod]

public void returnZeroMatrix()

{

//arrange

int I = 1;

int J = 1;

int[,] expected = new int[I, J];

//act

Matrix result = new Matrix(I, J);

//assert

CollectionAssert.Equals(expected, result);

}

[TestMethod]

public void setNumberRowsAndColumns()

{

//arrange

int I = 1;

int J = 2;

int[] expected = { I, J };

//act

Matrix result = new Matrix(I, J);

//assert

CollectionAssert.Equals(expected, result.Property);

}

}

}

MatrixDiffTests.cs:

using ConsoleApp1;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace MatrixTests

{

[TestClass]

public class MatrixDiffTests

{

[TestMethod]

[ExpectedException(typeof(MyException))]

public void numberRowsNotEqual\_Expected\_MyException()

{

//arrange

Matrix A = new Matrix(1, 1);

Matrix B = new Matrix(2, 1);

//act

Matrix C = A - B;

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void numberColumnsNotEqual\_Expected\_MyException()

{

//arrange

Matrix A = new Matrix(1, 2);

Matrix B = new Matrix(1, 1);

//act

Matrix C = A - B;

}

[TestMethod]

public void returnDiffMatrix()

{

//arrange

// Заполнение единицами

Matrix A = new Matrix(2, 2);

A[0, 0] = 1; A[1, 0] = 1; A[0, 1] = 1; A[1, 1] = 1;

// Заполнение двойками

Matrix B = new Matrix(2, 2);

B[0, 0] = 2; B[1, 0] = 2; B[0, 1] = 2; B[1, 1] = 2;

Matrix expected = new Matrix(2, 2);

expected[0, 0] = -1; expected[1, 0] = -1; expected[0, 1] = -1; expected[1, 1] = -1;

//act

Matrix C = A - B;

// assert

Assert.IsTrue(C == expected);

}

}

}

MatrixEqualTests.cs:

using ConsoleApp1;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace MatrixTests

{

[TestClass]

public class MatrixEqualTests

{

[TestMethod]

[ExpectedException(typeof(MyException))]

public void numberRowsNotEqual\_Expected\_MyException()

{

//arrange

Matrix A = new Matrix(1, 1);

Matrix B = new Matrix(2, 1);

//act

bool equal = A == B;

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void numberColumnsNotEqual\_Expected\_MyException()

{

//arrange

Matrix A = new Matrix(1, 2);

Matrix B = new Matrix(1, 1);

//act

bool equal = A == B;

}

[TestMethod]

public void returnEqualMatrix()

{

//arrange

Matrix A = new Matrix(2, 2);

A[0, 0] = 1; A[1, 0] = 2; A[0, 1] = 3; A[1, 1] = 4;

Matrix B = new Matrix(2, 2);

B[0, 0] = 1; B[1, 0] = 2; B[0, 1] = 3; B[1, 1] = 4;

//act

bool result = A == B;

// assert

Assert.IsTrue(result);

}

}

}

MatrixMinTests.cs:

using ConsoleApp1;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace MatrixTests

{

[TestClass]

public class MatrixMinTests

{

[TestMethod]

public void returnMinMatrixElement()

{

//arrange

Matrix A = new Matrix(2, 2);

A[0, 0] = 1; A[1, 0] = 2; A[0, 1] = 3; A[1, 1] = 4;

int expected = 1;

//act

int result = A.Min();

// assert

Assert.AreEqual(expected, result);

}

}

}

MatrixMultTests.cs:

using ConsoleApp1;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace MatrixTests

{

[TestClass]

public class MatrixMultTests

{

[TestMethod]

[ExpectedException(typeof(MyException))]

public void numberAColumnsNotEqualNumberBRows\_Expected\_MyException()

{

//arrange

Matrix A = new Matrix(1, 1);

Matrix B = new Matrix(2, 1);

//act

Matrix C = A \* B;

}

[TestMethod]

public void returnMultMatrix()

{

//arrange

Matrix A = new Matrix(2, 2);

A[0, 0] = 1; A[1, 0] = 2; A[0, 1] = 3; A[1, 1] = 4;

Matrix B = new Matrix(2, 2);

B[0, 0] = 1; B[1, 0] = 2; B[0, 1] = 3; B[1, 1] = 4;

Matrix expected = new Matrix(2, 2);

expected[0, 0] = 7; expected[1, 0] = 10; expected[0, 1] = 15; expected[1, 1] = 22;

//act

Matrix C = A \* B;

// assert

Assert.IsTrue(C == expected);

}

}

}

MatrixSumTests.cs:

using ConsoleApp1;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace MatrixTests

{

[TestClass]

public class MatrixSumTests

{

[TestMethod]

[ExpectedException(typeof(MyException))]

public void numberRowsNotEqual\_Expected\_MyException()

{

//arrange

Matrix A = new Matrix(1, 1);

Matrix B = new Matrix(2, 1);

//act

Matrix C = A + B;

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void numberColumnsNotEqual\_Expected\_MyException()

{

//arrange

Matrix A = new Matrix(1, 2);

Matrix B = new Matrix(1, 1);

//act

Matrix C = A + B;

}

[TestMethod]

public void returnSumMatrix()

{

//arrange

// Заполнение единицами

Matrix A = new Matrix(2, 2);

A[0, 0] = 1; A[1, 0] = 1; A[0, 1] = 1; A[1, 1] = 1;

// Заполнение двойками

Matrix B = new Matrix(2, 2);

B[0, 0] = 2; B[1, 0] = 2; B[0, 1] = 2; B[1, 1] = 2;

Matrix expected = new Matrix(2, 2);

expected[0,0] = 3; expected[1,0] = 3; expected[0,1] = 3; expected[1,1] = 3;

//act

Matrix C = A + B;

// assert

Assert.IsTrue(C == expected);

}

}

}

MatrixTestElement.cs:

using ConsoleApp1;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace MatrixTests

{

[TestClass]

public class MatrixTestElement

{

[TestMethod]

[ExpectedException(typeof(MyException))]

public void IOutOfBoundsIsLessZero\_Expected\_MyException()

{

//arrange

Matrix A = new Matrix(2, 2);

int i = -1;

//act

int el = A[i, 0];

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void JOutOfBoundsIsLessZero\_Expected\_MyException()

{

//arrange

Matrix A = new Matrix(2, 2);

int j = -1;

//act

int el = A[0, j];

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void IOutOfBoundsIsEqualLength\_Expected\_MyException()

{

//arrange

Matrix A = new Matrix(2, 2);

int i = 2;

//act

int el = A[i, 0];

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void JOutOfBoundsIsEqualZero\_Expected\_MyException()

{

//arrange

Matrix A = new Matrix(2, 2);

int j = -1;

//act

int el = A[0, j];

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void IOutOfBoundsIsHightLength\_Expected\_MyException()

{

//arrange

Matrix A = new Matrix(2, 2);

int i = 3;

//act

int el = A[i, 0];

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void JOutOfBoundsIsHightZero\_Expected\_MyException()

{

//arrange

Matrix A = new Matrix(2, 2);

int j = 3;

//act

int el = A[0, j];

}

[TestMethod]

public void getMatrixElement()

{

//arrange

Matrix A = new Matrix(2, 2);

A[0, 0] = 1; A[1, 0] = 2; A[0, 1] = 3; A[1, 1] = 4;

int expected = 3;

//act

int result = A[0, 1];

// assert

Assert.AreEqual(expected, result);

}

[TestMethod]

public void setMatrixElement()

{

//arrange

Matrix A = new Matrix(2, 2);

A[0, 0] = 1; A[1, 0] = 2; A[0, 1] = 3; A[1, 1] = 4;

int expected = 10;

//act

A[0, 1] = 10;

// assert

Assert.AreEqual(expected, A[0, 1]);

}

}

}

MatrixToStringTests.cs:

using ConsoleApp1;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace MatrixTests

{

[TestClass]

public class MatrixToStringTests

{

[TestMethod]

public void returnMatrixInStringFormat()

{

//arrange

Matrix A = new Matrix(2, 2);

A[0, 0] = 1; A[1, 0] = 2; A[0, 1] = 3; A[1, 1] = 4;

string expected = "{{1,3},{2,4}}";

//act

string result = A.ToString();

// assert

Assert.AreEqual(expected, result);

}

}

}

MatrixTranspTests.cs:

using ConsoleApp1;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace MatrixTests

{

[TestClass]

public class MatrixTranspTests

{

[TestMethod]

[ExpectedException(typeof(MyException))]

public void numberRowsNotEqualNumberColumns\_Expected\_MyException()

{

//arrange

Matrix A = new Matrix(1, 2);

//act

A.Transp();

}

[TestMethod]

public void returnTranspMatrix()

{

//arrange

Matrix A = new Matrix(2, 2);

A[0, 0] = 1; A[1, 0] = 2; A[0, 1] = 3; A[1, 1] = 4;

Matrix expected = new Matrix(2, 2);

expected[0, 0] = 1; expected[1, 0] = 3; expected[0, 1] = 2; expected[1, 1] = 4;

//act

Matrix result = A.Transp();

// assert

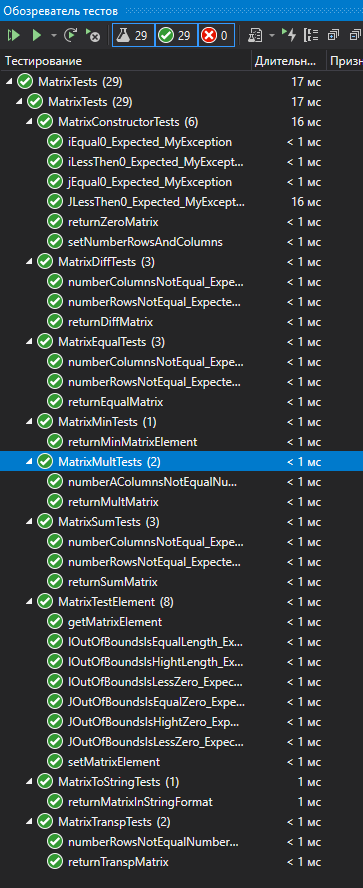
CollectionAssert.Equals(expected, result);

}

}

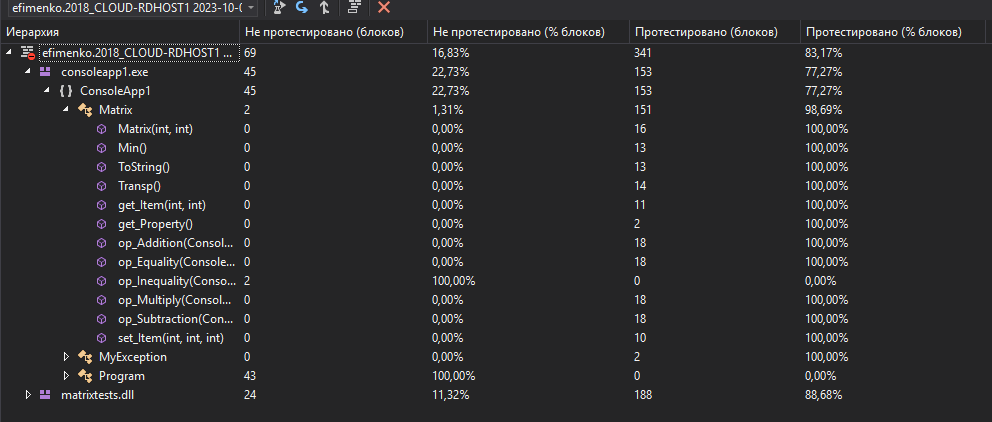
}

# Результаты выполнения модульных тестов



Все тесты выполнились успешно.

# Результаты покрытия разработанного кода тестами



Таким образом, по рисунку выше видно, что написанные тесты покрыли весь код (100%).

# Вывод

В результате выполнения практической работы были получены практические навыки модульного тестирования. Был разработан класс Matrix на языке программирования C#.

Тестирование выполнилось успешно и покрыло весь код.