**Отчёт**

**«Класс для работы с матрицами»**

**Задание**

Разработайте класс Матрица (Matrix) для операций матричной алгебры и протестируйте методы этого класса в режиме консольного приложения.

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

Описание операций класса Matrix представлены в таблице 1.

Таблица 1. Операции класса Matrix

|  |  |
| --- | --- |
| **Конструктор** (Matrix) | |
| Вход | Число строк i и столбцов j. |
| Предусловия | Число строк и столбцов должно быть больше 0. |
| Процесс | Создаёт объект типа Matrix с заданным числом строк и столбцов. Заносит число строк и столбцов в соответствующие свойства I и J. |
| **Сложить** (operator +) | |
| Вход | (b) – объект тип Matrix. |
| Предусловия | Число строк и столбцов в суммируемых матрицах должны совпадать |
| 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. |
| 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 типа Matrix. |
| Предусловия | Матрицы, участвующие в умножении, должны быть согласованы для этой операции по числу строк и столбцов. |
| 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 типа Matrix. |
| Предусловия | Число строк и столбцов в матрицах, участвующих в вычитании, должны совпадать. |
| Процесс | Возвращает значение true, если элементы объектов this и b в на одинаковых позициях равны. |
| Выход | Значение типа bool. |
| Постусловия | Нет |
| **Транспонировать** (Transp) | |
| Вход | Нет |
| Предусловия | Матрица, подвергаемая транспонированию, должна иметь одинаковое число строк и столбцов. |
| 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 | Значение типа int. |
| Постусловия | Нет |
| **Преобразовать в строку** (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 | Значение типа string |
| Постусловия | Нет |
| **Взять элемент с индексами** (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** (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. |

**Исходный текст приложения**

**Проект ConsoleApplicationMatrix**

Matrix.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplicationMatrix

{

public class MyException : Exception

{

public MyException(string s) : base(s)

{

}

}

public class Matrix

{

int[,] m; // Свойство для работы с числом строк.

public int I { get; set; } // Свойство для работы с числом столбцов.

public int J { get; set; } // Конструктор.

// Конструктор (Число строк i и столбцов j)

public Matrix(int i, int j)

{

if (i <= 0)

throw new MyException(string.Format("недопустимое значение i = {0}", i));

if (j <= 0)

throw new MyException(string.Format("недопустимое значение j = {0}", j));

I = i;

J = j;

m = new int[i, j];

}

// Оператор сложения

public static Matrix operator +(Matrix a, Matrix b)

{

if (a.I != b.I || a.J != b.J)

throw new MyException(string.Format("размерности не совпадают: {0}x{1} != {2}x{3}", a.I, a.J, b.I, b.J));

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[i, j] = a.m[i, j] + b.m[i, j];

}

return c;

}

// Оператор вычитания

public static Matrix operator -(Matrix a, Matrix b)

{

if (a.I != b.I || a.J != b.J)

throw new MyException(string.Format("размерности не совпадают: {0}x{1} != {2}x{3}", a.I, a.J, b.I, b.J));

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[i, j] = a.m[i, j] - b.m[i, j];

}

return c;

}

// Оператор умножения

public static Matrix operator \*(Matrix a, Matrix b)

{

if (a.J != b.I)

throw new MyException(string.Format("размерности не согласованы: {0}x{1} и {2}x{3}", a.I, a.J, b.I, b.J));

Matrix c = new Matrix(a.I, b.J);

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

{

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

{

int value = 0;

for (int k = 0; k < a.J; k++)

{

value += a.m[i, k] \* b.m[k, j];

}

c[i, j] = value;

}

}

return c;

}

public override bool Equals(object obj)

{

return (this as Matrix) == (obj as Matrix);

}

// Оператор равенства

public static bool operator ==(Matrix a, Matrix b)

{

if (a.I != b.I || a.J != b.J)

throw new MyException(string.Format("размерности не совпадают: {0}x{1} != {2}x{3}", a.I, a.J, b.I, b.J));

bool q = true;

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

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

{

if (a[i, j] != b[i, j])

{

q = false;

break;

}

}

return q;

}

// Оператор неравенства

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

{

return !(a == b);

}

// Транспонировать копию матрицы

public Matrix Transp()

{

if (I != J)

throw new MyException(string.Format("невозможно транспонировать не квадратную матрицу: {0}x{1}", this.I, this.J));

Matrix c = new Matrix(I, J);

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

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

{

c[i, j] = this[j, i];

}

return c;

}

// Минимальный элемент

public int Min()

{

int iMin = 0;

int jMin = 0;

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

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

{

if (this[i, j] < this[iMin, jMin])

{

iMin = i;

jMin = j;

}

}

return this[iMin, jMin];

}

// Отображение элементов в виде строки

public override string ToString()

{

string s = "{";

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

{

if (i > 0)

s += ",";

s += "{";

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

{

if (j > 0)

s += ",";

s += string.Format("{0}", this[i, j]);

}

s += "}";

}

s += "}";

return s;

}

// Индексатор для доступа к значениям компонентов матрицы.

public int this[int i, int j]

{

get // Взять элемент с индексами i, j

{

if (i < 0 | i > I - 1)

throw new MyException(string.Format("неверное значение i = {0}", i));

if (j < 0 | j > J - 1)

throw new MyException(string.Format("неверное значение j = {0}", j));

return m[i, j];

}

set // Записать элемент с индексами i, j

{

if (i < 0 | i > I - 1)

throw new MyException(string.Format("неверное значение i = {0}", i));

if (j < 0 | j > J - 1)

throw new MyException(string.Format("неверное значение j = {0}", j));

m[i, j] = value;

}

}

//Вывод значений компонентов на консоль.

public void Show()

{

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

{

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

{

Console.Write("\t" + this[i,j] );

}

Console.WriteLine();

}

Console.WriteLine();

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplicationMatrix

{

class Program

{

static void Main(string[] args)

{

try

{

//Создаём матрицу a.

Matrix a = new Matrix(3, 3);

//Создаём матрицу b.

Matrix b = new Matrix(3, 3);

//Объявляем матрицу c.

Matrix c;

//Заполняем матрицу a.

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

{

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

{

a[i, j] = a.J \* i + j;

}

}

//Выводим матрицу a.

a.Show();

//Заполняем матрицу b.

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

{

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

{

b[i, j] = a.J \* i + j + 1;

}

}

//Выводим матрицу a.

b.Show();

//Складываем матрицы a и b.

c = a + b;

//Выводим матрицу c.

c.Show();

}

catch (MyException e)

{

Console.WriteLine(e.Message);

}

}

}

}

**Проект MatrixTests**

MainTest.cs

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using ConsoleApplicationMatrix;

namespace MatrixTests

{

[TestClass]

public class MainTest

{

[TestMethod]

[ExpectedException(typeof(MyException))]

public void Matrix\_Expected\_MyException\_i()

{

//act (выполнить)

Matrix a = new Matrix(0, 2);

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void Matrix\_Expected\_MyException\_j()

{

//act (выполнить)

Matrix a = new Matrix(2, -1);

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void Sum\_Expected\_MyException()

{

//arrange(обеспечить)

Matrix a = new Matrix(2, 1);

a[0, 0] = 1;

a[1, 0] = 1;

Matrix b = new Matrix(2, 2);

b[0, 0] = 2;

b[0, 1] = 2;

b[1, 0] = 2;

b[1, 1] = 2;

//act (выполнить)

Matrix c = a + b;

}

[TestMethod]

public void Sum()

{

//arrange(обеспечить)

Matrix a = new Matrix(2, 2);

a[0, 0] = 1;

a[0, 1] = 1;

a[1, 0] = 1;

a[1, 1] = 1;

Matrix b = new Matrix(2, 2);

b[0, 0] = 2;

b[0, 1] = 2;

b[1, 0] = 2;

b[1, 1] = 2;

Matrix expected = new Matrix(2, 2);

expected[0, 0] = 3;

expected[0, 1] = 3;

expected[1, 0] = 3;

expected[1, 1] = 3;

//act (выполнить)

Matrix actual = a + b;

//assert(доказать)

Assert.IsTrue(actual == expected); //Оракул

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void Subtract\_Expected\_MyException()

{

//arrange(обеспечить)

Matrix a = new Matrix(2, 1);

a[0, 0] = 1;

a[1, 0] = 1;

Matrix b = new Matrix(2, 2);

b[0, 0] = 2;

b[0, 1] = 2;

b[1, 0] = 2;

b[1, 1] = 2;

//act (выполнить)

Matrix c = a + b;

}

[TestMethod]

public void Subtract()

{

//arrange(обеспечить)

Matrix a = new Matrix(2, 2);

a[0, 0] = 3;

a[0, 1] = 4;

a[1, 0] = 5;

a[1, 1] = 1;

Matrix b = new Matrix(2, 2);

b[0, 0] = 2;

b[0, 1] = 2;

b[1, 0] = 2;

b[1, 1] = 2;

Matrix expected = new Matrix(2, 2);

expected[0, 0] = 1;

expected[0, 1] = 2;

expected[1, 0] = 3;

expected[1, 1] = -1;

//act (выполнить)

Matrix actual = a - b;

//assert(доказать)

Assert.IsTrue(actual == expected); //Оракул

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void Multiply\_Expected\_MyException()

{

//arrange(обеспечить)

Matrix a = new Matrix(2, 1);

a[0, 0] = 1;

a[1, 0] = 1;

Matrix b = new Matrix(2, 1);

b[0, 0] = 2;

b[1, 0] = 2;

//act (выполнить)

Matrix c = a \* b;

}

[TestMethod]

public void Multiply()

{

//arrange(обеспечить)

Matrix a = new Matrix(2, 3);

a[0, 0] = 2;

a[0, 1] = 3;

a[0, 2] = 4;

a[1, 0] = 5;

a[1, 1] = 6;

a[1, 2] = 7;

Matrix b = new Matrix(3, 4);

b[0, 0] = 2;

b[0, 1] = 3;

b[0, 2] = 4;

b[0, 3] = 5;

b[1, 0] = 6;

b[1, 1] = 7;

b[1, 2] = 8;

b[1, 3] = 9;

b[2, 0] = 10;

b[2, 1] = 11;

b[2, 2] = 12;

b[2, 3] = 13;

Matrix expected = new Matrix(2, 4);

expected[0, 0] = 62;

expected[0, 1] = 71;

expected[0, 2] = 80;

expected[0, 3] = 89;

expected[1, 0] = 116;

expected[1, 1] = 134;

expected[1, 2] = 152;

expected[1, 3] = 170;

//act (выполнить)

Matrix actual = a \* b;

//assert(доказать)

Assert.IsTrue(actual == expected); //Оракул

}

[TestMethod]

public void Equal()

{

//arrange(обеспечить)

Matrix a = new Matrix(2, 2);

a[0, 0] = 1;

a[0, 1] = 1;

a[1, 0] = 1;

a[1, 1] = 1;

Matrix b = new Matrix(2, 2);

b[0, 0] = 1;

b[0, 1] = 1;

b[1, 0] = 1;

b[1, 1] = 1;

//act (выполнить)

bool r = a == b;

//assert(доказать)

Assert.IsTrue(r);

Assert.AreEqual(a, b);

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void Transp\_Expected\_MyException()

{

//arrange(обеспечить)

Matrix a = new Matrix(2, 3);

a[0, 0] = 2;

a[0, 1] = 3;

a[0, 2] = 4;

a[1, 0] = 5;

a[1, 1] = 6;

a[1, 2] = 7;

//act (выполнить)

Matrix b = a.Transp();

}

[TestMethod]

public void Transp()

{

//arrange(обеспечить)

Matrix a = new Matrix(2, 2);

a[0, 0] = 2;

a[0, 1] = 3;

a[1, 0] = 4;

a[1, 1] = 5;

Matrix expected = new Matrix(2, 2);

expected[0, 0] = 2;

expected[0, 1] = 4;

expected[1, 0] = 3;

expected[1, 1] = 5;

//act (выполнить)

Matrix actual = a.Transp();

//assert(доказать)

Assert.IsTrue(actual == expected); //Оракул

}

[TestMethod]

public void Min()

{

//arrange(обеспечить)

Matrix a = new Matrix(2, 2);

a[0, 0] = 2;

a[0, 1] = 3;

a[1, 0] = -1;

a[1, 1] = 5;

//act (выполнить)

int min = a.Min();

//assert(доказать)

Assert.IsTrue(min == -1); //Оракул

}

[TestMethod]

public void Str()

{

//arrange(обеспечить)

Matrix a = new Matrix(2, 2);

a[0, 0] = 2;

a[0, 1] = 3;

a[1, 0] = -1;

a[1, 1] = 5;

string expected = "{{2,3},{-1,5}}";

//act (выполнить)

string actual = a.ToString();

//assert(доказать)

Assert.IsTrue(expected == actual);

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void This\_Expected\_MyException\_set\_j()

{

//act (выполнить)

Matrix a = new Matrix(2, 2);

a[1, 3] = 2;

}

[TestMethod]

[ExpectedException(typeof(MyException))]

public void This\_Expected\_MyException\_get\_i()

{

//act (выполнить)

Matrix a = new Matrix(2, 2);

int r = a[3, 1];

}

}

}

**Результаты тестирования методов класса Matrix**

Ниже перечислены имена модульных тестов для каждого из методов и операторов класса Matrix.

**Конструктор**

* тест Matrix\_Expected\_MyException\_i
* тест Matrix\_Expected\_MyException\_j

**Оператор «+»**

* тест Sum\_Expected\_MyException
* тест Sum

**Оператор «-»**

* тест Subtract\_Expected\_MyException
* тест Subtract

**Оператор «\*»**

* тест Multiply\_Expected\_MyException
* тест Multiply

**Оператор «==» и метод IsEqual**

* тест Equal

**Метод Transp**

* тест Transp\_Expected\_MyException
* тест Transp

**Метод Min**

* тест Min

**Метод ToString**

* тест Str

**Оператор индексации**

* тест This\_Expected\_MyException\_set\_j
* тест This\_Expected\_MyException\_get\_i

Все 15 тестов выполняются успешно, как показано на рисунке 1.

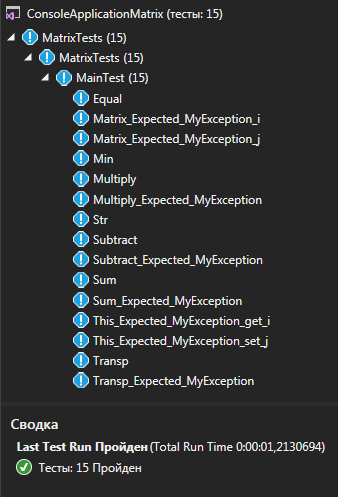


Рисунок 1 – результаты выполнения тестов.