

Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет телекоммуникаций и
информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Лабораторная работа №4
«Разработка и модульное тестирование класса Матрица на C#»

Выполнил:
Студент группы ИП-013
Копытина Т.А.
Работу проверил:
ассистент кафедры ПМиК
Агалаков А.А.

Новосибирск 2023 г.

Содержание

1. Задание	3
2. Исходный код программы.....	6
2.1. Код программы	6
2.2. Код тестов.....	14
3. Результаты модульных тестов	18
4. Вывод	18

1. 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 по критерию С2 (путей). 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 – не изменяемый.

Операции

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.
<i>Умножить (operator*)</i>	
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.
<i>Равно (operator==)</i>	
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.
<i>Транспонировать (Transp)</i>	
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.

<i>Минимальный элемент(Min)</i>	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Отыскивает и возвращает минимальный среди элементов объекта this.
Выход:	Значение типа int.
Постусловия:	Нет.
<i>ПреобразоватьВстроку(ToString)</i>	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Преобразует элементы матрицы this в строковое представление построчно. Напрмер: {{1,2,3},{4,5,6},{7,8,9}}
Выход:	Строка.
Постусловия:	Нет.
<i>Взять элемент с индексами i,j (this [i,j])</i>	
Вход:	Значения i, j типа int.
Предусловия:	Значения i, j должны находиться в допустимых диапазонах.
Процесс:	Возвращает элемент матрицы с индексами <i>i,j</i> .
Выход:	Значение типа int.
Постусловия:	Нет.
<i>Взять число строк I</i>	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает число строк в матрице.
Выход:	Целое – число строк.
Постусловия:	Нет.
<i>Взять число столбцов J</i>	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает число столбцов в матрице.
Выход:	Целое – число столбцов.
Постусловия:	Нет.

end Matrix

2. Исходный код программы

2.1. Код программы

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace lab4
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                Matrix a = new Matrix(3, 3);
                Matrix b = new Matrix(3, 3);
                Matrix c;
                for (int i = 0; i < a.I; i++)
                {
                    for (int j = 0; j < a.J; j++)
                    {
                        a[i, j] = a.J * i + j;
                    }
                }
                Console.WriteLine("Matrix a:\n");
                a.Show();
                for (int i = 0; i < a.I; i++)
                {
                    for (int j = 0; j < a.J; j++)
                    {
                        b[i, j] = a.J * i + j;
                    }
                }
                Console.WriteLine("Matrix b:\n");
                b.Show();

                Console.WriteLine("a + b:\n");
                c = a + b;
                c.Show();
                Console.WriteLine("\n");

                Console.WriteLine("a - b:\n");
                c = a - b;
                c.Show();
                Console.WriteLine("\n");

                Console.WriteLine("a * b:\n");
                c = a * b;
```

```

        c.Show();
        Console.WriteLine("\n\n");

        Console.WriteLine("a == b:\n");
        bool res = a == b;
        Console.WriteLine(res + "\n\n");

        Console.WriteLine("Transpose a:\n");
        a.Show();
        c = a;
        c.Transp();
        c.Show();
        Console.WriteLine("\n");

        Console.WriteLine("Minimum a:\n");
        a.Show();
        int m = a.Min();
        Console.WriteLine("Min a = " + m + "\n\n");

        Console.WriteLine("ToString a:\n");
        a.Show();
        string str = a.Matrix_str();
        Console.WriteLine("a = " + str + "\n\n");

        Console.WriteLine("Take elem a[1, 2]:\n");
        int elem = a.Take_elem(1, 2);
        Console.WriteLine("a[1, 2] = " + elem + "\n\n");

        Console.WriteLine("Write elem a[1, 1] = 7:\n");
        a.Show();
        c = a;
        c.Write_elem(3, 0, 7);
        c.Show();
        //Console.WriteLine(c + "\n\n");
    }
    catch (MyException e)
    {
        Console.WriteLine(e.Message);
    }
    Console.ReadKey();
}
}
}

```

Matrix.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace lab4
{
    // Обработка исключения
    public class MyException : Exception
    {
        public MyException(string str) : base(str) { }
    }
    public class Matrix
    {
        int[,] matrix;
        public int I { get; set; }
        public int J { get; set; }
        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;
            matrix = new int[i, j];
        }
        public int this[int i, int j]
        {
            get
            {
                if (i < 0 | i > I - 1)
                {
                    throw new
MyException(string.Format("Неверное значение i = {0}", i));
                }
                if (j < 0 | i > J - 1)
                {
                    throw new
MyException(string.Format("Неверное значение j = {0}", j));
                }
                return matrix[i, j];
            }
            set
            {
                if (i < 0 | i > I - 1)
                {

```



```

        throw new
MyException(string.Format("Неверное значение i = {0}", i));
    }
    if (j < 0 | i > J - 1)
    {
        throw new
MyException(string.Format("Неверное значение j = {0}", j));
    }
    matrix[i, j] = value;
    }
}
public static Matrix operator +(Matrix a, Matrix b)
{
    if (a.I != b.I | a.J != b.J)
    {
        throw new MyException(string.Format("Размерности
матриц a и b разные"));
    }
    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;
}
public static bool operator ==(Matrix a, Matrix b)
{
    if (a.I != b.I | a.J != b.J)
    {
        throw new MyException(string.Format("Размерности
матриц a и b разные"));
    }
    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)
    {
        return !(a.matrix == b.matrix);
    }
    public static Matrix operator -(Matrix a, Matrix b)
    {
        if (a.I != b.I | a.J != b.J)
        {
            throw new MyException(string.Format("Размерности
матриц а и b разные"));
        }
        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;
    }
    public static Matrix operator *(Matrix a, Matrix b)
    {
        if (a.I != b.I | a.J != b.J)
        {
            throw new MyException(string.Format("Размерности
матриц а и b разные"));
        }
        Matrix c = new Matrix(a.I, a.J);
        for (int i = 0; i < a.I; i++)
        {
            for (int j = 0; j < a.J; j++)
            {
                for (int k = 0; k < a.J; k++)
                {
                    c.matrix[i, j] += a.matrix[i, k] *
b.matrix[k, j];
                }
            }
        }
        return c;
    }
    public Matrix Transp()
    {
        if (I != J)
        {

```

```

        throw new MyException(string.Format("Число строк
и столбцов должно совпадать"));
    }
    int tmp = 0;
    for (int i = 0; i < I; i++)
    {
        for (int j = 0; j < i; j++)
        {
            tmp = this[i, j];
            this[i, j] = this[j, i];
            this[j, i] = tmp;
        }
    }
    return this;
}
public int Min()
{
    int minimum = this[0, 0];
    for (int i = 0; i < I; i++)
    {
        for (int j = 0; j < J; j++)
        {
            if (this[i, j] < minimum)
            {
                minimum = this[i, j];
            }
        }
    }
    return minimum;
}

public string Matrix_str()
{
    string str = "{";
    string str_tmp = "";
    for (int i = 0; i < I; i++)
    {
        str_tmp += "{ ";
        for (int j = 0; j < J; j++)
        {
            str_tmp = str_tmp + this[i, j].ToString() +
" ";

        }
        str_tmp = str_tmp + "}";
    }
    str = str + str_tmp + "}";
    return str;
}
public int Take_elem(int n, int m)

```

```

        {
            if (n < 0 | n > I - 1)
            {
                throw new MyException(string.Format("Неверное
значение n = {0}", n));
            }
            if (m < 0 | m > J - 1)
            {
                throw new MyException(string.Format("Неверное
значение m = {0}", m));
            }
            int num = 0;
            for (int i = 0; i < I; i++)
            {
                for (int j = 0; j < J; j++)
                {
                    if (i == n && j == m)
                    {
                        num = this[i, j];
                    }
                }
            }
            return num;
        }
        public void Write_elem(int n, int m, int new_num)
        {
            if (n < 0 | n > I - 1)
            {
                throw new MyException(string.Format("Неверное
значение n = {0}", n));
            }
            if (m < 0 | m > J - 1)
            {
                throw new MyException(string.Format("Неверное
значение m = {0}", m));
            }
            for (int i = 0; i < I; i++)
            {
                for (int j = 0; j < J; j++)
                {
                    if (i == n && j == m)
                    {
                        this[i, j] = new_num;
                    }
                }
            }
        }

        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();
    }
    public override bool Equals(object obj)
    {
        return (this as Matrix) == (obj as Matrix);
    }
}

```

2.2.

2.3. Код тестов

MatrixTest.cs

```
using System;
using lab4;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace TestProject1
{
    [TestClass]
    public class MatrixTests
    {
        [TestMethod]
        [ExpectedException(typeof(MyException))]

        public void OutOfBounds_i()
        {
            Matrix a = new Matrix(0, 4);
        }

        [TestMethod]
        [ExpectedException(typeof(MyException))]

        public void OutOfBounds_j()
        {
            Matrix a = new Matrix(4, -1);
        }

        [TestMethod]
        public void Sum()
        {
            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 expect = new Matrix(2, 2);
            expect[0, 0] = 3;
            expect[0, 1] = 3;
            expect[1, 0] = 3;
            expect[1, 1] = 3;
            Matrix actual = new Matrix(2, 2);
            actual = a + b;
        }
    }
}
```

```

        Assert.IsTrue(actual == expect);
    }

    [TestMethod]
    public void Diff()
    {
        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 expect = new Matrix(2, 2);
        expect[0, 0] = 1;
        expect[0, 1] = 1;
        expect[1, 0] = 1;
        expect[1, 1] = 1;
        Matrix actual = new Matrix(2, 2);
        actual = b - a;
        Assert.IsTrue(actual == expect);
    }

    [TestMethod]
    public void Mult()
    {
        Matrix a = new Matrix(2, 2);
        a[0, 0] = 3;
        a[0, 1] = 3;
        a[1, 0] = 3;
        a[1, 1] = 3;
        Matrix b = new Matrix(2, 2);
        b[0, 0] = 2;
        b[0, 1] = 2;
        b[1, 0] = 2;
        b[1, 1] = 2;
        Matrix expect = new Matrix(2, 2);
        expect[0, 0] = 12;
        expect[0, 1] = 12;
        expect[1, 0] = 12;
        expect[1, 1] = 12;
        Matrix actual = new Matrix(2, 2);
        actual = a * b;
        Assert.IsTrue(actual == expect);
    }

```

```

[TestMethod]
public void Equal()
{
    Matrix a = new Matrix(2, 2);
    a[0, 0] = 1;
    a[0, 1] = 2;
    a[1, 0] = 3;
    a[1, 1] = 4;
    Matrix b = new Matrix(2, 2);
    b[0, 0] = 1;
    b[0, 1] = 2;
    b[1, 0] = 3;
    b[1, 1] = 4;
    bool res = a == b;
    Assert.IsTrue(res);
}

[TestMethod]
public void Transpos()
{
    Matrix a = new Matrix(2, 2);
    a[0, 0] = 1;
    a[0, 1] = 2;
    a[1, 0] = 3;
    a[1, 1] = 4;
    Matrix actual = a.Transp();
    Matrix expect = new Matrix(2, 2);
    expect[0, 0] = 1;
    expect[0, 1] = 3;
    expect[1, 0] = 2;
    expect[1, 1] = 4;
    Assert.IsTrue(actual == expect);
}

[TestMethod]
public void Minimum()
{
    Matrix a = new Matrix(2, 2);
    a[0, 0] = 4;
    a[0, 1] = 3;
    a[1, 0] = 2;
    a[1, 1] = 1;
    int actual = a.Min();
    int expect = 1;
    Assert.IsTrue(actual == expect);
}

[TestMethod]
public void ToStr()

```



```

    {
        Matrix a = new Matrix(2, 2);
        a[0, 0] = 1;
        a[0, 1] = 2;
        a[1, 0] = 3;
        a[1, 1] = 4;
        string actual = a.Matrix_str();
        string expect = "{{ 1 2 } { 3 4 }}";
        Assert.IsTrue(actual == expect);
    }

[TestMethod]
public void TakeElem()
{
    Matrix a = new Matrix(2, 2);
    a[0, 0] = 4;
    a[0, 1] = 3;
    a[1, 0] = 2;
    a[1, 1] = 1;
    int actual = a.Take_elem(1, 0);
    int expect = 2;
    Assert.IsTrue(actual == expect);
}

[TestMethod]
public void WriteElem()
{
    Matrix actual = new Matrix(2, 2);
    actual[0, 0] = 1;
    actual[0, 1] = 2;
    actual[1, 0] = 3;
    actual[1, 1] = 4;
    actual.Write_elem(1, 0, 8);
    Matrix expect = new Matrix(2, 2);
    expect[0, 0] = 1;
    expect[0, 1] = 2;
    expect[1, 0] = 8;
    expect[1, 1] = 4;
    Assert.IsTrue(actual == expect);
}
}
}

```

3. Результаты модульных тестов

The screenshot shows the 'Обозреватель тестов' (Test Explorer) window in Visual Studio. The status bar at the top indicates 'Запуск тестов завершен: тестов запущено в 367 мс: 11 (пройдено: 11, не пройдено: 0, пропущено: 0)'. The main pane displays a tree view of test results:

Тестирование	Длительность...	Признаки	Сообщение об ошибке
MatrixTest (11)	19 мс		
TestProject1 (11)	19 мс		
MatrixTests (11)	19 мс		
Diff	6 мс		
Equal	< 1 мс		
Minimum	< 1 мс		
Mult	< 1 мс		
OutOfBounds_i	13 мс		
OutOfBounds_j	< 1 мс		
Sum	< 1 мс		
TakeElem	< 1 мс		
ToStr	< 1 мс		
Transpos	< 1 мс		
WriteElem	< 1 мс		

On the right, the 'Сводка по группе' (Group Summary) pane shows:

- MatrixTest
- Тесты в группе: 11
- Общая длительность: 19 мс
- Результаты: 11 Пройден

4. Вывод

По итогам данной лабораторной работе были сформированы практические навыки разработки на C# и модульного тестирования классов средствами Visual Studio.