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 С#»**

Выполнил:

Студент группы ИП-013

Копытина Т.А.

Работу проверил:

ассистент кафедры ПМиК

Агалаков А.А.

Новосибирск 2023 г.

**Содержание**

[1. Задание 3](#_Toc84522514)

[2. Исходный код программы 6](#_Toc84522515)

[2.1. Код программы 6](#_Toc84522516)

[2.2. Код тестов 14](#_Toc84522517)

[3. Результаты модульных тестов 18](#_Toc84522518)

[4. Вывод 18](#_Toc84522519)

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**

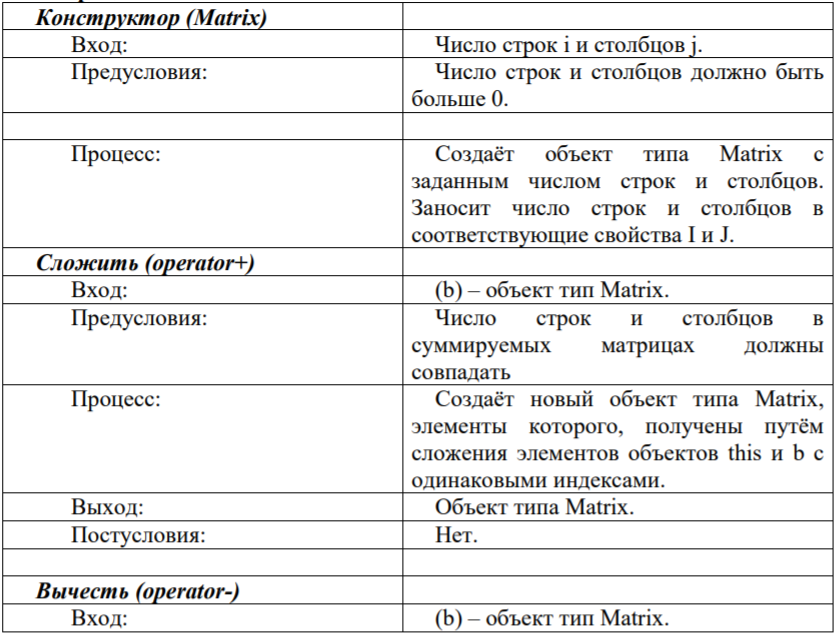
**ADT** Matrix

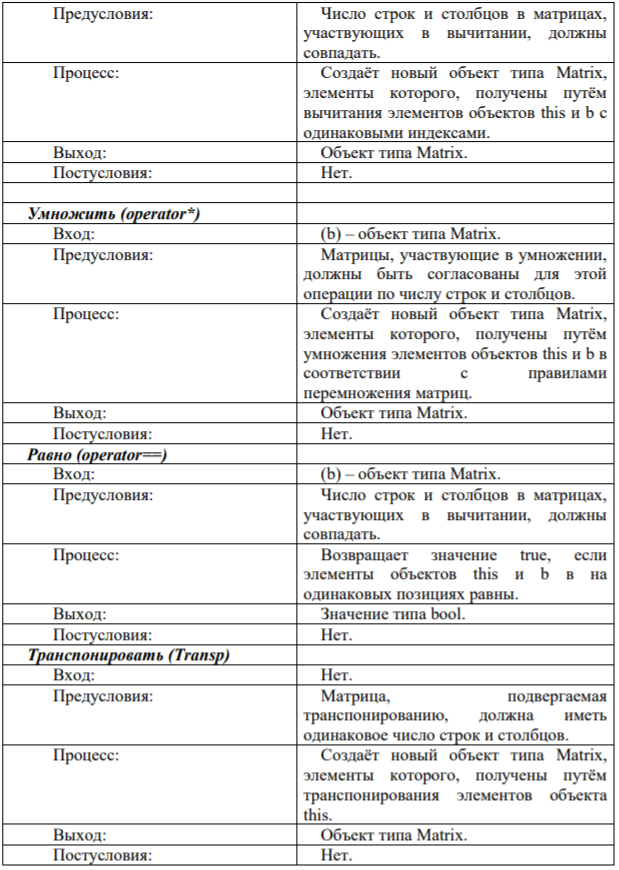
**Данные**

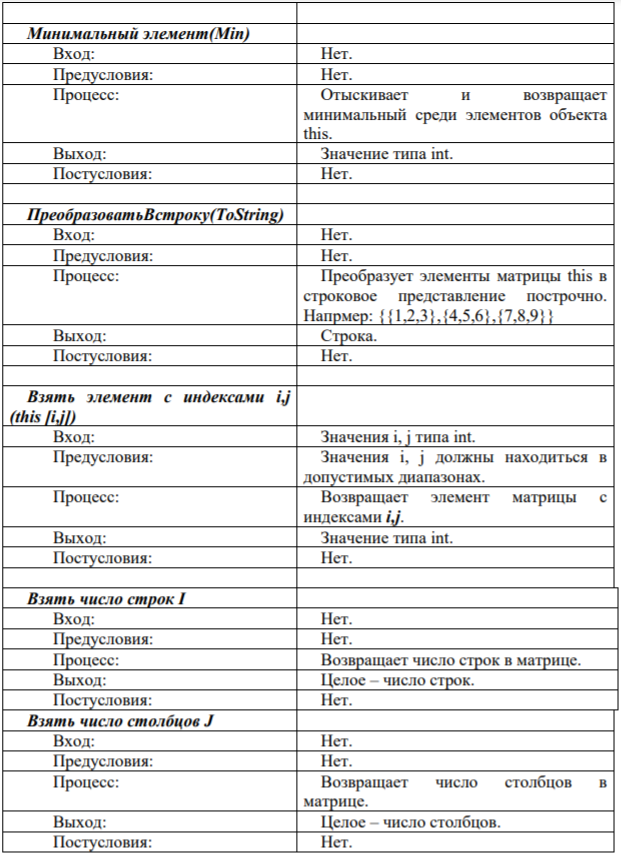
Матрица (тип Matrix) - это двумерная матрица со значениями целого типа.

Объект типа Матрица – не изменяемый.

**Операции**







**end** Matrix

1. **Исходный код программы**
   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 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 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++)

{

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. **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;

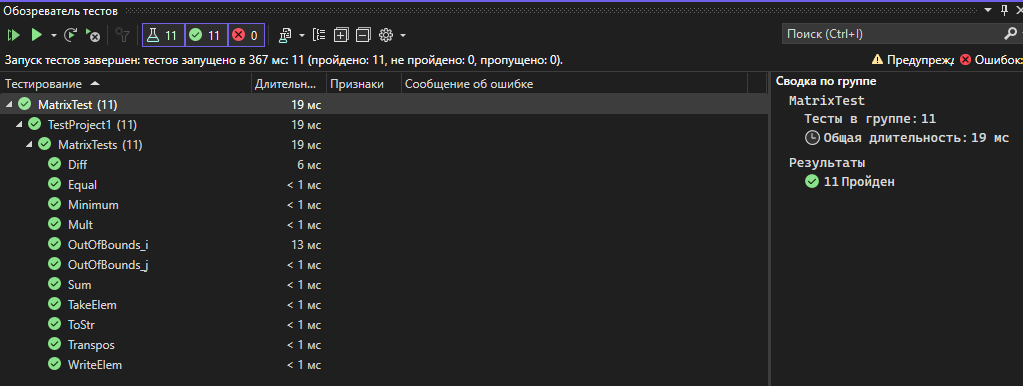
expect[1, 1] = 4;

Assert.IsTrue(actual == expect);

}

}

}

1. **Результаты модульных тестов**
2. **Вывод**

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