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-113

Шпилев Д. И.

Работу проверил:

старший преподаватель кафедры ПМиК

Агалаков А.А.

Новосибирск 2024 г.

**Содержание**

[1. Задание 3](#_Toc84522514)

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

[2.1. Код программы 6](#_Toc84522516)

[2.2. Код тестов 9](#_Toc84522517)

[3. Результаты модульных тестов 13](#_Toc84522518)

[4. Вывод 14](#_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**

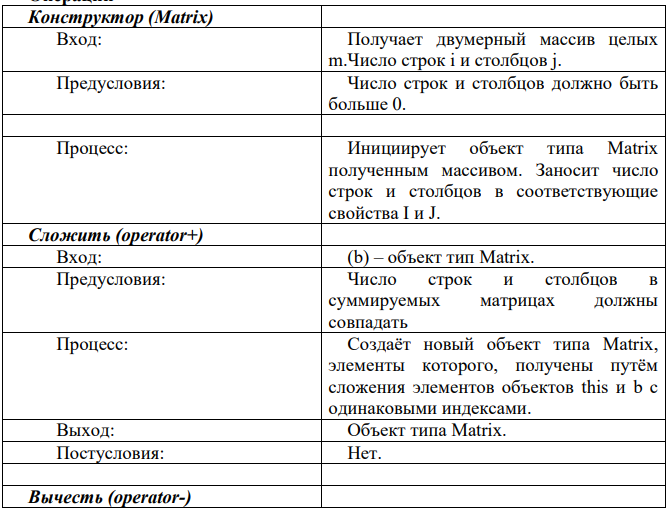
**ADT** Matrix

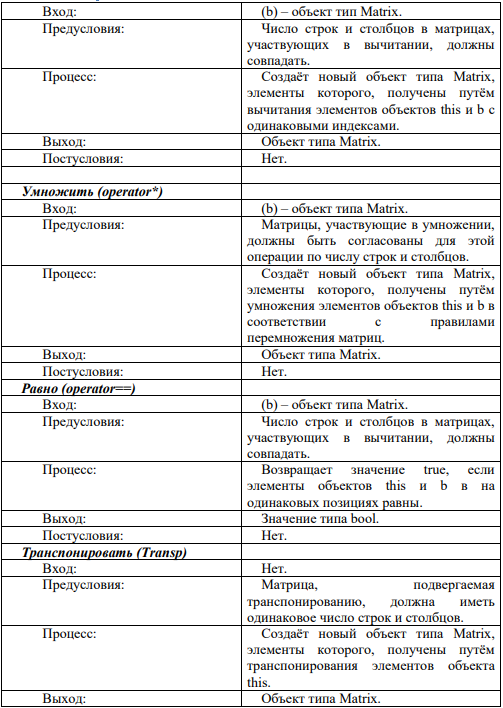
**Данные**

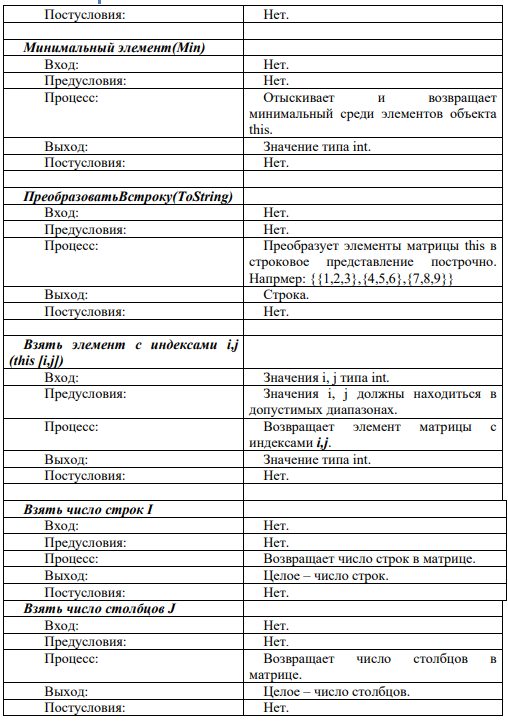
Матрица (тип Matrix) - это двумерная матрица со значениями целого типа.

Объект типа Матрица – не изменяемый.

**Операции**







**end** Matrix

1. **Исходный код программы**
   1. **Код программы**

***Matrix.cs***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace Lab4

{

public class Matrix

{

private readonly int[,] matrix;

public int I { get; }

public int J { get; }

public Matrix(int[,] matrix)

{

if (matrix.GetLength(0) < 1 || matrix.GetLength(1) < 1)

{

throw new ArgumentException("The number of I and J must be greater than 0");

}

this.matrix = (int[,])matrix.Clone();

I = matrix.GetLength(0);

J = matrix.GetLength(1);

}

public int this[int i, int j]

{

get

{

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

throw new IndexOutOfRangeException("Indexes are out of range of acceptable values");

return matrix[i, j];

}

}

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

{

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

{

throw new InvalidOperationException("Matrices must be of the same sizes");

}

int[,] result = new int[a.I, a.J];

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

{

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

{

result[i, j] = a.matrix[i, j] + b.matrix[i, j];

}

}

return new Matrix(result);

}

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

{

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

{

throw new InvalidOperationException("Matrices must be of the same sizes");

}

int[,] result = new int[a.I, a.J];

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

{

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

{

result[i, j] = a.matrix[i, j] - b.matrix[i, j];

}

}

return new Matrix(result);

}

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

{

if (a.J != b.I)

throw new InvalidOperationException("Matrices are not consistent for multiplication");

int[,] result = new int[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 < a.J; k++)

{

result[i, j] += a[i, k] \* b[k, j];

}

}

}

return new Matrix(result);

}

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

{

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

throw new InvalidOperationException("Matrices are not consistent for multiplication");

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

{

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

{

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

}

}

return true;

}

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

public Matrix Transp()

{

if (I != J)

{

throw new InvalidOperationException("Matrix must have the same number of rows and columns");

}

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

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

{

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

{

result[j, i] = matrix[i, j];

}

}

return new Matrix(result);

}

public int Min()

{

int min = matrix[0, 0];

foreach (int num in matrix)

{

min = Math.Min(min, num);

}

return min;

}

public override string ToString()

{

string str = "{";

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

{

str += "{";

for (int j = 0; j < J - 1; ++j)

{

str += matrix[i, j].ToString() + ",";

}

str += matrix[i, J - 1].ToString() + "}";

if (i != I - 1) str += ",";

}

str += "}";

return str;

}

public override bool Equals(object obj)

{

if (obj == null || GetType() != obj.GetType())

return false;

Matrix other = obj as Matrix;

return this == other;

}

public override int GetHashCode()

{

return base.GetHashCode();

}

}

}

* 1. **Код тестов**

***MatrixTest.cs***

using Lab4;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

namespace MatrixTest

{

[TestClass]

public class MatrixTest

{

[TestMethod]

public void Constructor\_ValidInput\_CreatesMatrix()

{

int[,] data = { { 1, 2 }, { 3, 4 } };

Matrix matrix = new Matrix(data);

Assert.AreEqual(2, matrix.I);

Assert.AreEqual(2, matrix.J);

Assert.AreEqual(1, matrix[0, 0]);

Assert.AreEqual(4, matrix[1, 1]);

}

[TestMethod]

public void Constructor\_InvalidInput\_ThrowsException\_EmptyMatrix()

{

int[,] data = { };

Assert.ThrowsException<ArgumentException>(() => new Matrix(data));

}

[TestMethod]

public void Equals\_TwoEqualMatrices\_ReturnsTrue()

{

Matrix matrix1 = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

Matrix matrix2 = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

Assert.IsTrue(matrix1.Equals(matrix2));

}

[TestMethod]

public void Equals\_TwoDifferentMatrices\_ReturnsFalse()

{

Matrix matrix1 = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

Matrix matrix2 = new Matrix(new int[,] { { 5, 6 }, { 7, 8 } });

Assert.IsFalse(matrix1.Equals(matrix2));

}

[TestMethod]

public void Equals\_TwoDifferentSizes()

{

Matrix matrix1 = new Matrix(new int[,] { { 1, 2 }, { 3, 4 }, { 6, 2 } });

Matrix matrix2 = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

Assert.ThrowsException<InvalidOperationException>(() => matrix1 == matrix2);

}

[TestMethod]

public void Equals\_TwoDifferentTypes()

{

Matrix matrix1 = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

Assert.IsFalse(matrix1.Equals(new int[,] { { 1, 2 }, { 3, 4 } }));

}

[TestMethod]

public void Equals\_Null()

{

Matrix matrix1 = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

Assert.IsFalse(matrix1.Equals(null));

}

[TestMethod]

public void Inequals\_TwoDifferentMatrices\_ReturnsTrue()

{

Matrix matrix1 = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

Matrix matrix2 = new Matrix(new int[,] { { 1, 2 }, { 3, 5 } });

Assert.IsTrue(matrix1 != matrix2);

}

[TestMethod]

public void Add\_TwoMatrices\_ReturnsCorrectResult()

{

Matrix matrix1 = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

Matrix matrix2 = new Matrix(new int[,] { { 5, 6 }, { 7, 8 } });

Matrix result = matrix1 + matrix2;

Assert.AreEqual(new Matrix(new int[,] { { 6, 8 }, { 10, 12 } }), result);

}

[TestMethod]

public void Add\_TwoMatrices\_DifferentI()

{

Matrix matrix1 = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

Matrix matrix2 = new Matrix(new int[,] { { 5, 6 }, { 7, 8 }, { 6, 2} });

Assert.ThrowsException<InvalidOperationException>(() => matrix1 + matrix2);

}

[TestMethod]

public void Subtract\_TwoMatrices\_ReturnsCorrectResult()

{

Matrix matrix1 = new Matrix(new int[,] { { 5, 6 }, { 7, 8 } });

Matrix matrix2 = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

Matrix result = matrix1 - matrix2;

Assert.AreEqual(new Matrix(new int[,] { { 4, 4 }, { 4, 4 } }), result);

}

[TestMethod]

public void Subtract\_TwoMatrices\_DifferentI()

{

Matrix matrix1 = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

Matrix matrix2 = new Matrix(new int[,] { { 5, 6 }, { 7, 8 }, { 1, 2} });

Assert.ThrowsException<InvalidOperationException>(() => matrix1 - matrix2);

}

[TestMethod]

public void Multiply\_TwoMatrices\_ReturnsCorrectResult()

{

Matrix matrix1 = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

Matrix matrix2 = new Matrix(new int[,] { { 5, 6 }, { 7, 8 } });

Matrix result = matrix1 \* matrix2;

Assert.AreEqual(new Matrix(new int[,] { { 19, 22 }, { 43, 50 } }), result);

}

[TestMethod]

public void Multiply\_TwoMatrices\_WrongSizes()

{

Matrix matrix1 = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

Matrix matrix2 = new Matrix(new int[,] { { 5, 6 }, { 7, 8 }, { 1, 2 } });

Assert.ThrowsException<InvalidOperationException>(() => matrix1 \* matrix2);

}

[TestMethod]

public void Min\_ReturnsMinimumElement()

{

Matrix matrix = new Matrix(new int[,] { { 5, 3 }, { 7, 1 } });

int minElement = matrix.Min();

Assert.AreEqual(1, minElement);

}

[TestMethod]

public void Transp\_ReturnsTransposedMatrix()

{

Matrix matrix = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

Matrix result = matrix.Transp();

Assert.AreEqual(new Matrix(new int[,] { { 1, 3 }, { 2, 4 } }), result);

}

[TestMethod]

public void Transp\_ErrorSize()

{

Matrix matrix = new Matrix(new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 } });

Assert.ThrowsException<InvalidOperationException>(() => matrix.Transp());

}

[TestMethod]

public void ToString\_ReturnsCorrectStringRepresentation()

{

Matrix matrix = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

string result = matrix.ToString();

Assert.AreEqual("{{1,2},{3,4}}", result);

}

[TestMethod]

public void Indexer\_ValidIndexes\_ReturnsCorrectElement()

{

Matrix matrix = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

Assert.AreEqual(1, matrix[0, 0]);

Assert.AreEqual(4, matrix[1, 1]);

}

[TestMethod]

public void Indexer\_InvalidIndexes\_ThrowsException()

{

Matrix matrix = new Matrix(new int[,] { { 1, 2 }, { 3, 4 } });

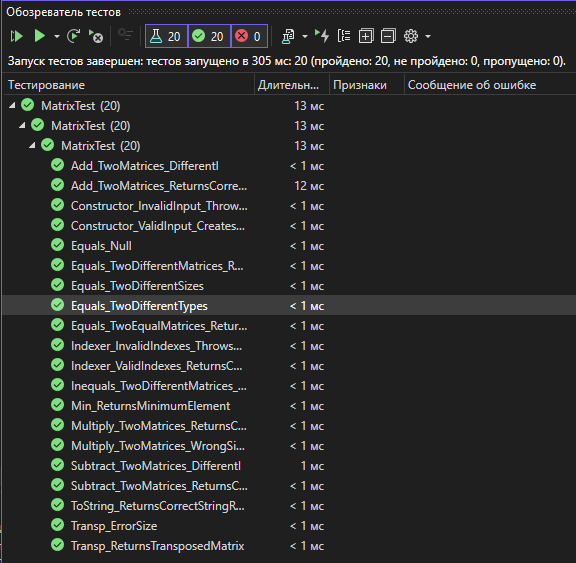
Assert.ThrowsException<IndexOutOfRangeException>(() => { var element = matrix[2, 2]; });

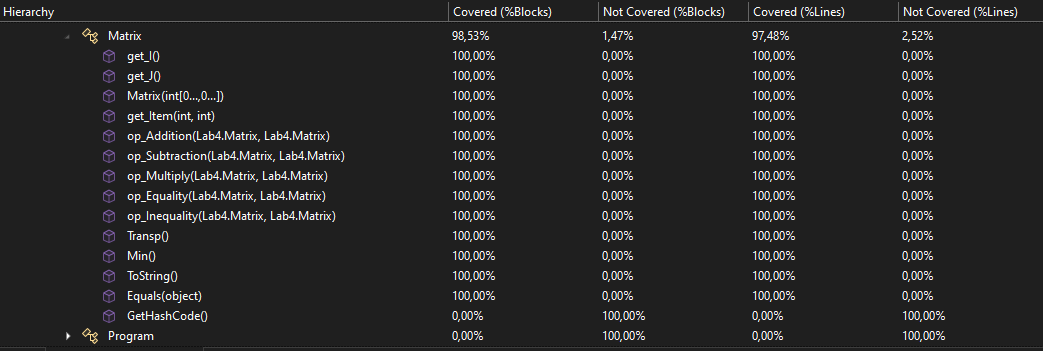
}

}

}

1. **Результаты модульных тестов**

****

****

1. **Вывод**

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