**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

по дисциплине: «ООПиП»

на тему: **«**Особенности разработки и верификации в объектно-ориентированном языке**»**

Выполнил: студент гр. ИТИ-21

Готченя А. А.

Принял: преподаватель

Башаримов Ю. С.

Гомель 2023

**Цель работы:** изучить, что такое перегрузка методов и операций, узнать, как создаются модульные тесты

**Задание:**

1. Необходимо разработать тип для работы с матрицами.
2. Реализовать методы, позволяющие выполнять операции умножения матриц, умножения матрицы на вектор, предусмотрев возможность их выполнения.
3. Тип должен быть размещён в библиотеке классов.
4. Весь код должен быть снабжён элементами документирования
5. Разработать не менее 15 модульных тестов для тестирования созданного класса
6. Создать консольное приложение с интерфейсом пользователя
7. Подключить созданную библиотеку в консольное приложение
8. Сгенерировать XML-файл с документацией по проекту
9. Верифицировать консольное приложение.

**Ход работы:**

На рисунке 1 изображено диалоговое окно с пользователем.

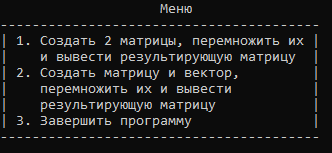


Рисунок 1 – Диалоговое окно с пользователем

При выборе пункта 1 пользователь вводит размерность обеих матриц, а затем инициализирует их. После этого происходит проверка на возможность умножения этих матриц, если это возможно, то пользователю выводится получившаяся матрица, а если нет, то выводится исключение. На рисунке 2 представлено успешное выполнение 1-ого пункта меню.

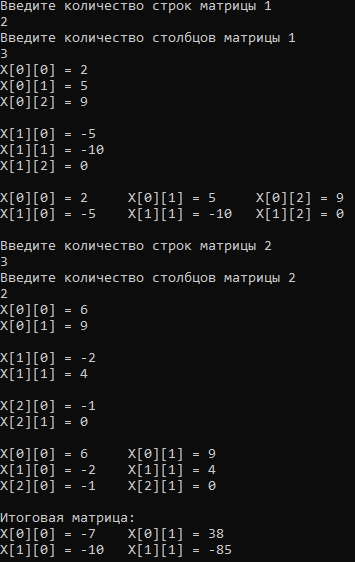


Рисунок 2 – Успешное выполнение 1-ого пункта меню

На рисунке 3 представлен вывод исключения при попытке перемножения матриц несоответствующих размерностей.

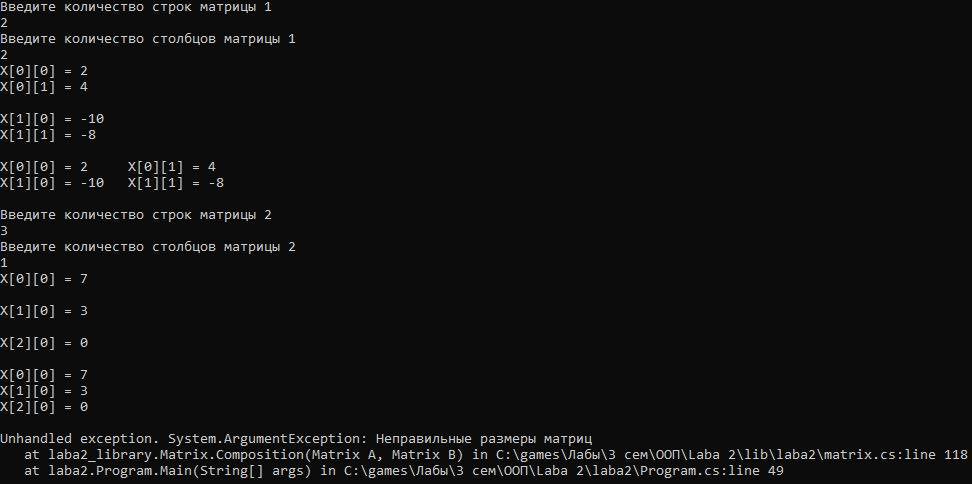


Рисунок 3 – Вывод исключения при попытке перемножения матриц несоответствующих размерностей

При выборе пункта 2 пользователь вводит размерности матрицы и вектора, а затем инициализирует их. После этого происходит проверка на возможность умножения матрицы на вектор, если это возможно, то пользователю выводится получившаяся матрица, а если нет, то выводится исключение. Данный метод является перегрузкой метода перемножения двух матриц (код находится в приложении А). На рисунке 4 представлено успешное выполнение 2-ого пункта меню.

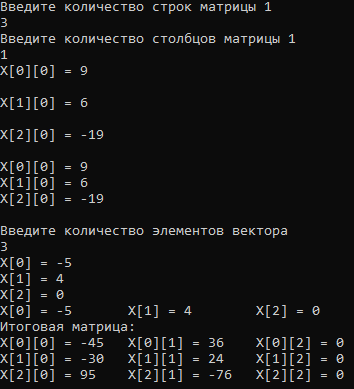


Рисунок 4 – Успешное выполнение 2-ого пункта меню

Для проверки работы некоторых методов были созданы модульные тесты, код которых находится в приложении А. В приложении А представлен код программы. В приложении Б представлены сгенерированные *XML*-файлы документации.

**Вывод:** в результате выполнения лабораторной работы изучили перегрузки методов и операций, а также научились создавать модульные тесты.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст программы**

**Program.cs**

using laba2\_library;

namespace laba2

{

class Program

{

static void Main(string[] args)

{

Matrix matrix1, matrix2, matrix3;

Vector;

int choice, row, column, length;

while (true)

{

Console.Clear();

Console.WriteLine(" Меню");

Console.WriteLine("----------------------------------------");

Console.WriteLine("| 1. Создать 2 матрицы, перемножить их |");

Console.WriteLine("| и вывести результирующую матрицу |");

Console.WriteLine("| 2. Создать матрицу и вектор, |");

Console.WriteLine("| перемножить их и вывести |");

Console.WriteLine("| результирующую матрицу |");

Console.WriteLine("| 3. Завершить программу |");

Console.WriteLine("----------------------------------------");

choice = int.Parse(Console.ReadLine());

switch (choice)

{

case 1:

Console.WriteLine("Введите количество строк матрицы 1");

row = int.Parse(Console.ReadLine());

Console.WriteLine("Введите количество столбцов матрицы 1");

column = int.Parse(Console.ReadLine());

matrix1 = new Matrix(row, column);

matrix1.Init();

matrix1.GetOutput();

Console.WriteLine("Введите количество строк матрицы 2");

row = int.Parse(Console.ReadLine());

Console.WriteLine("Введите количество столбцов матрицы 2");

column = int.Parse(Console.ReadLine());

matrix2 = new Matrix(row, column);

matrix2.Init();

matrix2.GetOutput();

matrix3 = Matrix.Composition(matrix1, matrix2);

Console.WriteLine("Итоговая матрица:");

matrix3.GetOutput();

break;

case 2:

Console.WriteLine("Введите количество строк матрицы 1");

row = int.Parse(Console.ReadLine());

Console.WriteLine("Введите количество столбцов матрицы 1");

column = int.Parse(Console.ReadLine());

matrix1 = new Matrix(row, column);

matrix1.Init();

matrix1.GetOutput();

Console.WriteLine("Введите количество элементов вектора");

length = int.Parse(Console.ReadLine());

vector = new Vector(length);

vector.Init();

vector.GetOutput();

matrix3 = Matrix.Composition(matrix1, vector);

Console.WriteLine("Итоговая матрица:");

matrix3.GetOutput();

break;

case 3:

return;

default:

Console.WriteLine("Нет такого пункта меню");

Console.ReadLine();

break;

}

}

}

}

}

**Matrix.cs**

using System;

using System.Runtime.CompilerServices;

namespace laba2\_library

{

public class Matrix

{

/// <summary>

/// Кол-во столбиков.

/// </summary>

public int Column { get; }

/// <summary>

/// Кол-во строк.

/// </summary>

public int Row { get; }

/// <summary>

/// Матрица

/// </summary>

public int[,] matrix;

/// <summary>

/// Базовый конструктор.

/// </summary>

/// <param name="row">Кол-во строк.</param>

/// <param name="column">Кол-во столбиков.</param>

public Matrix(int row, int column)

{

this.Column = column;

this.Row = row;

matrix = new int[row, column];

}

/// <summary>

/// Конструктор матрицы, в который передаётся вектор

/// </summary>

/// <param name="vector"></param>

public Matrix(Vector vector)

{

this.Column = vector.Length;

this.Row = 1;

this.matrix = new int[this.Row, this.Column];

for (int i = 0; i < vector.Length; i++)

matrix[0, i] = vector.vect[i];

}

/// <summary>

/// Инициализация элементов матрицы

/// </summary>

public void Init()

{

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

{

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

{

do

{

Console.Write($"X[{i}][{j}] = ");

}

while (!int.TryParse(Console.ReadLine(), out matrix[i, j]));

}

Console.WriteLine();

}

}

/// <summary>

/// Вывести элементы матрицы

/// </summary>

public void GetOutput()

{

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

{

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

{

Console.Write($"X[{i}][{j}] = {matrix[i, j]}\t");

}

Console.WriteLine();

}

Console.ReadLine();

}

/// <summary>

/// Перемножение матриц.

/// </summary>

/// <param name="A">Матрица 1</param>

/// <param name="B">Матрица 2</param>

/// <param name="result"></param>

public static Matrix Composition(Matrix A, Matrix B)

{

if (A.Column == B.Row)

{

Matrix result = new Matrix(A.Row, B.Column);

int temp;

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

{

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

{

temp = 0;

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

{

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

}

result.matrix[i, j] = temp;

}

}

return result;

}

else

{

throw new ArgumentException("Неправильные размеры матриц");

}

}

/// <summary>

/// Умножение матрицы на вектор

/// </summary>

/// <param name="A">Матрица</param>

/// <param name="B">Вектор</param>

/// <param name="result"></param>

public static Matrix Composition(Matrix A, Vector vector)

{

Matrix B1 = new Matrix(vector);

return Composition(A, B1);

}

public static Matrix Composition(Vector, Matrix A)

{

Matrix B1 = new Matrix(vector);

return Composition(B1, A);

}

}

}

**Vector.cs**

namespace laba2\_library

{

/// <summary>

/// Класс, описывающий вектор

/// </summary>

public class Vector

{

/// <summary>

/// Кол-во элементов в векторе

/// </summary>

public int Length;

/// <summary>

/// Массив элементов вектора

/// </summary>

public int[] vect;

public Vector(int length)

{

this.Length = length;

vect = new int[length];

}

/// <summary>

/// Инициализация элементов вектора

/// </summary>

public void Init()

{

for (int i = 0; i < vect.Length; i++)

{

do

Console.Write($"X[{i}] = ");

while (!int.TryParse(Console.ReadLine(), out vect[i]));

}

}

/// <summary>

/// Вывод всех элементов вектора

/// </summary>

public void GetOutput()

{

for (int i = 0; i < vect.Length; i++)

{

Console.Write($"X[{i}] = {vect[i]}\t");

}

Console.ReadLine();

}

}

}

**UnitTest1.cs**

using laba2\_library;

namespace Tests1

{

[TestClass]

public class Tests

{

/// <summary>

/// Метод для проверки равности двух двумерных массивов

/// </summary>

/// <param name="array1"></param>

/// <param name="array2"></param>

/// <returns>True, если массивы равно поэлментно, иначе False.</returns>

public static bool AreTwoDimensionalArraysEqual(int[,] array1, int[,] array2)

{

if (array1 == null || array2 == null)

{

return false;

}

if (array1.GetLength(0) != array2.GetLength(0) || array1.GetLength(1) != array2.GetLength(1))

{

return false;

}

for (int i = 0; i < array1.GetLength(0); i++)

{

for (int j = 0; j < array1.GetLength(1); j++)

{

if (array1[i, j] != array2[i, j])

{

return false;

}

}

}

return true;

}

/// <summary>

/// Тестирование матрицы 3x1 и вектора 1x3

/// </summary>

[TestMethod]

public void TestComposition1()

{

Matrix result = new(1, 1);

result.matrix[0, 0] = 38;

Vector = new Vector(3);

vector.vect = new int[3] { 5, 6, 7 };

Matrix matrix2 = new(3, 1);

matrix2.matrix = new int[3, 1] { { 1 }, { 2 }, { 3 } };

Matrix matrix3 = Matrix.Composition( vector, matrix2);

Assert.AreEqual(result.Column, matrix3.Column);

Assert.AreEqual(result.Row, matrix3.Row);

Assert.IsTrue(AreTwoDimensionalArraysEqual(result.matrix, matrix3.matrix));

}

/// <summary>

/// Тестирование умножения 2 матриц размерностью 2x2.

/// </summary>

[TestMethod]

public void TestComposition2()

{

Matrix result = new(2, 2);

result.matrix = new int[2, 2] { {-75, 13 }, { 135, -81} };

Matrix matrix1 = new(2, 2);

matrix1.matrix = new int[2, 2] { { -5, 16}, { 9, 0} };

Matrix matrix2 = new(2, 2);

matrix2.matrix = new int[2, 2] { {15, -9 }, { 0, -2} };

Matrix matrix3 = Matrix.Composition(matrix1, matrix2);

Assert.AreEqual(result.Column, matrix3.Column);

Assert.AreEqual(result.Row, matrix3.Row);

Assert.IsTrue(AreTwoDimensionalArraysEqual(result.matrix, matrix3.matrix));

}

/// <summary>

/// Тестирование умножения матрицы 3x3 и матрицы 3x1

/// </summary>

[TestMethod]

public void TestComposition3()

{

Matrix result = new(3, 1);

result.matrix = new int[3, 1] { { -43 }, { 84 }, { 3 } };

Matrix matrix1 = new(3, 3);

matrix1.matrix = new int[3, 3] { { -5, 16, 2 }, { 9, -1, 5}, { 0, -3, 0} };

Matrix matrix2 = new(3, 1);

matrix2.matrix = new int[3, 1] { { 7 }, { -1 }, { 4 } };

Matrix matrix3 = Matrix.Composition(matrix1, matrix2);

Assert.AreEqual(result.Column, matrix3.Column);

Assert.AreEqual(result.Row, matrix3.Row);

Assert.IsTrue(AreTwoDimensionalArraysEqual(result.matrix, matrix3.matrix));

}

/// <summary>

/// Проверка на вызов исключения при попытке перемножения матрицы 3x3 и матрицы 1x3.

/// </summary>

[TestMethod]

public void TestCompositionException1()

{

Matrix matrix1 = new(3, 3);

matrix1.matrix = new int[3, 3] { { -5, 16, 2 }, { 9, -1, 5 }, { 0, -3, 0 } };

Matrix matrix2 = new(1, 3);

matrix2.matrix = new int[1, 3] { { 7 , -1, 4 } };

Assert.ThrowsException<ArgumentException>(() => Matrix.Composition(matrix1, matrix2));

}

}

}

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**XML-файлы документации**

**Laba2\_library.xml**

<?xml version="1.0"?>

<doc>

<assembly>

<name>laba2\_library</name>

</assembly>

<members>

<member name="P:laba2\_library.Matrix.Column">

<summary>

Кол-во столбиков.

</summary>

</member>

<member name="P:laba2\_library.Matrix.Row">

<summary>

Кол-во строк.

</summary>

</member>

<member name="F:laba2\_library.Matrix.matrix">

<summary>

Матрица

</summary>

</member>

<member name="M:laba2\_library.Matrix.#ctor(System.Int32,System.Int32)">

<summary>

Базовый конструктор.

</summary>

<param name="row">Кол-во строк.</param>

<param name="column">Кол-во столбиков.</param>

</member>

<member name="M:laba2\_library.Matrix.#ctor(laba2\_library.Vector)">

<summary>

Конструктор матрицы, в который передаётся вектор

</summary>

<param name="vector"></param>

</member>

<member name="M:laba2\_library.Matrix.Init">

<summary>

Инициализация элементов матрицы

</summary>

</member>

<member name="M:laba2\_library.Matrix.GetOutput">

<summary>

Вывести элементы матрицы

</summary>

</member>

<member name="M:laba2\_library.Matrix.Composition(laba2\_library.Matrix,laba2\_library.Matrix)">

<summary>

Перемножение матриц.

</summary>

<param name="A">Матрица 1</param>

<param name="B">Матрица 2</param>

<param name="result"></param>

</member>

<member name="M:laba2\_library.Matrix.Composition(laba2\_library.Matrix,laba2\_library.Vector)">

<summary>

Умножение матрицы на вектор

</summary>

<param name="A">Матрица</param>

<param name="B">Вектор</param>

<param name="result"></param>

</member>

<member name="T:laba2\_library.Vector">

<summary>

Класс, описывающий вектор

</summary>

</member>

<member name="F:laba2\_library.Vector.Length">

<summary>

Кол-во элементов в векторе

</summary>

</member>

<member name="F:laba2\_library.Vector.vect">

<summary>

Массив элементов вектора

</summary>

</member>

<member name="M:laba2\_library.Vector.Init">

<summary>

Инициализация элементов вектора

</summary>

</member>

<member name="M:laba2\_library.Vector.GetOutput">

<summary>

Вывод всех элементов вектора

</summary>

</member>

</members>

</doc>

**Tests1.xml**

<?xml version="1.0"?>

<doc>

<assembly>

<name>Tests1</name>

</assembly>

<members>

<member name="M:Tests1.Tests.AreTwoDimensionalArraysEqual(System.Int32[0:,0:],System.Int32[0:,0:])">

<summary>

Метод для проверки равности двух двумерных массивов

</summary>

<param name="array1"></param>

<param name="array2"></param>

<returns>True, если массивы равно поэлментно, иначе False.</returns>

</member>

<member name="M:Tests1.Tests.TestComposition1">

<summary>

Тестирование матрицы 3x1 и вектора 1x3

</summary>

</member>

<member name="M:Tests1.Tests.TestComposition2">

<summary>

Тестирование умножения 2 матриц размерностью 2x2.

</summary>

</member>

<member name="M:Tests1.Tests.TestComposition3">

<summary>

Тестирование умножения матрицы 3x3 и матрицы 3x1

</summary>

</member>

<member name="M:Tests1.Tests.TestCompositionException1">

<summary>

Проверка на вызов исключения при попытке перемножения матрицы 3x3 и матрицы 1x3.

</summary>

</member>

</members>

</doc>