Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Программирование»

**ОТЧЕТ**

к лабораторной работе №7

на тему:

**«Перегрузка операторов»**

БГУИР 6-05-0612-02 67

|  |
| --- |
| Выполнил студент группы 353503  КОХАН Артём Игоревич |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

**Задание 1. Вариант 7.** Класс квадратная матрица 2\*2 типа int. К полям обращаться через индексатор. Перегрузить +, -, ++, --, \*, \* на число, / на число. Сравнить на == и !=. (d) если определитель = 0, матрица = false. Преобразовать в число (определитель) и назад (a, 0, 0, a) – в обоих случаях явно.

Для заданной задачи реализовать следующее:

– спроектировать класс согласно варианту индивидуального задания. Для класса использовать отдельный модуль;

– спроектировать конструкторы и свойства с контролем корректности

вводимых значений;

– перегрузить метод toString();

– перегрузить математические операции (имеющие смысл для объектов класса), инкремент и декремент (изменить поля на 1), отношения (==, !=, <, >), true и false, преобразования типа;

– создать несколько объектов класса. Продемонстрировать использование конструкторов и свойств;

– продемонстрировать работу всех методов и операций.

# 2 Выполнение работы

Для выполнения задания был создан класс Matrix, где реализована логика программы. В классе имеется двумерный массив 2x2, обращение к этому поле происходит через индексатор, причём если обращение произошло вне диапазона двумерного массива, то throw бросит исключение. Далее реализована перегрузка всех требуемых операций и методов, стоит акцентировать внимание на перегрузке преобразовании типов, где используется explicit, что позволяет предотвратить неявные преобразования типов.

namespace Task1.Matrix2x2

{

internal class Matrix

{

private int[,] matrix = new int[2, 2];

public int this[int i, int j]

{

get

{

if (i < 0 || i > 1 || j < 0 || j > 1)

{

throw new IndexOutOfRangeException("Индекс вне диапазона матрицы 2х2\n");

}

return matrix[i, j];

}

set

{

if (i < 0 || i > 1 || j < 0 || j > 1)

{

throw new IndexOutOfRangeException("Индекс вне диапазона матрицы 2х2\n");

}

matrix[i, j] = value;

}

}

public Matrix(){}

public Matrix(int a11, int a12, int a21, int a22)

{

matrix[0, 0] = a11;

matrix[0, 1] = a12;

matrix[1, 0] = a21;

matrix[1, 1] = a22;

}

public static Matrix operator +(Matrix firstMatrix, Matrix secondMatrix)

{

Matrix result = new Matrix();

for (byte i = 0; i < 2; ++i)

{

for (byte j = 0; j < 2; ++j)

{

result[i, j] = firstMatrix[i, j] + secondMatrix[i, j];

}

}

return result;

}

public static Matrix operator ++(Matrix value)

{

Matrix result = new Matrix();

for (byte i = 0; i < 2; ++i)

{

for (byte j = 0; j < 2; ++j)

{

result[i, j] = ++value[i, j];

}

}

return result;

}

public static Matrix operator -(Matrix firstMatrix, Matrix secondMatrix)

{

Matrix result = new Matrix();

for (byte i = 0; i < 2; ++i)

{

for (byte j = 0; j < 2; ++j)

{

result[i, j] = firstMatrix[i, j] - secondMatrix[i, j];

}

}

return result;

}

public static Matrix operator --(Matrix valueMatrix)

{

for (byte i = 0; i < 2; ++i)

{

for (byte j = 0; j < 2; ++j)

{

--valueMatrix[i, j];

}

}

return valueMatrix;

}

public static Matrix operator \*(Matrix firstMatrix, Matrix secondMatrix)

{

Matrix result = new Matrix();

result[0, 0] = firstMatrix[0, 0] \* secondMatrix[0, 0] + firstMatrix[0, 1] \* secondMatrix[1, 0];

result[0, 1] = firstMatrix[0, 0] \* secondMatrix[0, 1] + firstMatrix[0, 1] \* secondMatrix[1, 1];

result[1, 0] = firstMatrix[1, 0] \* secondMatrix[0, 0] + firstMatrix[1, 1] \* secondMatrix[1, 0];

result[1, 1] = firstMatrix[1, 0] \* secondMatrix[0, 1] + firstMatrix[1, 1] \* secondMatrix[1, 1];

return result;

}

public static Matrix operator \*(Matrix valueMatrix, int value)

{

for (byte i = 0; i < 2; ++i)

{

for (byte j = 0; j < 2; ++j)

{

valueMatrix[i, j] \*= value;

}

}

return valueMatrix;

}

public static Matrix operator /(Matrix valueMatrix, int value)

{

for (byte i = 0; i < 2; ++i)

{

for (byte j = 0; j < 2; ++j)

{

valueMatrix[i, j] /= value;

}

}

return valueMatrix;

}

public static bool operator ==(Matrix firstMatrix, Matrix secondMatrix)

{

for (byte i = 0; i < 2; ++i)

{

for (byte j = 0; j < 2; ++j)

{

if (firstMatrix[i, j] != secondMatrix[i, j])

{

return false;

}

}

}

return true;

}

public static bool operator >(Matrix firstMatrix, Matrix secondMatrix)

{

if(firstMatrix[0, 0] \* firstMatrix[1, 1] - firstMatrix[0, 1] \* firstMatrix[1, 0] >

secondMatrix[0, 0] \* secondMatrix[1, 1] - secondMatrix[0, 1] \* secondMatrix[1, 0])

{

return true;

}

return false;

}

public static bool operator <(Matrix firstMatrix, Matrix secondMatrix)

{

if (firstMatrix[0, 0] \* firstMatrix[1, 1] - firstMatrix[0, 1] \* firstMatrix[1, 0] <

secondMatrix[0, 0] \* secondMatrix[1, 1] - secondMatrix[0, 1] \* secondMatrix[1, 0])

{

return true;

}

return false;

}

public static bool operator !=(Matrix firstMatrix, Matrix secondMatrix) =>

!(firstMatrix == secondMatrix);

public static bool operator true(Matrix valueMatrix) =>

valueMatrix[0, 0] \* valueMatrix[1, 1] - valueMatrix[0, 1] \* valueMatrix[1, 0] != 0;

public static bool operator false(Matrix valueMatrix) =>

valueMatrix[0, 0] \* valueMatrix[1, 1] - valueMatrix[0, 1] \* valueMatrix[1, 0] == 0;

public static explicit operator int(Matrix valueMatrix) =>

valueMatrix[0, 0] \* valueMatrix[1, 1] - valueMatrix[0, 1] \* valueMatrix[1, 0];

public static explicit operator Matrix(int a) =>

new Matrix { [0, 0] = a, [0, 1] = 0, [1, 0] = 0, [1, 1] = a };

public override string ToString() =>

$"{matrix[0, 0]} {matrix[0, 1]}\n{matrix[1, 0]} {matrix[1, 1]}\n";

}

}

Продемонстрируем работу класса Matrix в классе Program, используя все реализованные методы и операции. Создадим два объекта класса Matrix и проведём над ними все возможные операции.

namespace Task1

{

file class Program

{

static void Main(string[] args)

{

Matrix matrix1 = new Matrix(1, 2, 3, 4);

Matrix matrix2 = new Matrix(5, 6, 7, 8);

Console.WriteLine("Матрица 1:");

Console.WriteLine(matrix1.ToString());

Console.WriteLine("Матрица 2:");

Console.WriteLine(matrix2.ToString());

Matrix sum = matrix1 + matrix2;

Console.WriteLine("Сумма матриц 1 и 2:");

Console.WriteLine(sum.ToString());

Matrix diff = matrix1 - matrix2;

Console.WriteLine("Разность матриц 1 и 2:");

Console.WriteLine(diff.ToString());

Matrix product = matrix1 \* matrix2;

Console.WriteLine("Произведение матриц 1 и 2:");

Console.WriteLine(product.ToString());

Matrix increment = ++matrix1;

Console.WriteLine("Инкремент матрицы 1:");

Console.WriteLine(increment.ToString());

Matrix decrement = --matrix2;

Console.WriteLine("Декремент матрицы 2:");

Console.WriteLine(decrement.ToString());

Matrix multiplyByNumber = matrix1 \* 2;

Console.WriteLine("Умножение матрицы 1 на число 2:");

Console.WriteLine(multiplyByNumber.ToString());

Matrix divideByNumber = matrix1 / 2;

Console.WriteLine("Деление матрицы 1 на число 2:");

Console.WriteLine(divideByNumber.ToString());

bool areEqual = matrix1 == matrix2;

Console.WriteLine("Матрицы 1 и 2 равны: " + areEqual);

bool areNotEqual = matrix1 != matrix2;

Console.WriteLine("Матрицы 1 и 2 не равны: " + areNotEqual);

bool isFirstGreater = matrix1 > matrix2;

Console.WriteLine("Матрица 1 больше матрицы 2: " + isFirstGreater);

bool isFirstLess = matrix1 < matrix2;

Console.WriteLine("Матрица 1 меньше матрицы 2: " + isFirstLess);

if(matrix1)

{

Console.WriteLine("Определитель матрицы 1 НЕ равен 0");

}

else

{

Console.WriteLine("Определитель матрицы 1 равен 0");

}

int determinant1 = (int)matrix1;

Console.WriteLine("Определитель матрицы 1: " + determinant1);

int determinant2 = (int)matrix2;

Console.WriteLine("Определитель матрицы 2: " + determinant2);

Matrix fromNumber = (Matrix)determinant1;

Console.WriteLine("Матрица из числа (определителя матрицы 1):");

Console.WriteLine(fromNumber.ToString());

}

}

}

Результат работы программы продемонстрирован ниже (см. рисунок 1).

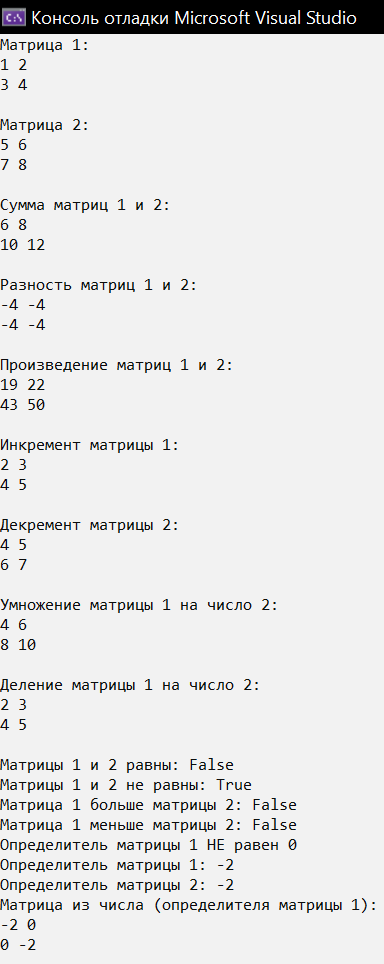


Рисунок 1 – Результат работы программы

Помимо этого стоит, в заключении, упомянуть про то, что если попробовать обратится вне диапазона двумерного массива, то throw бросит исключение (см. рисунок 2).

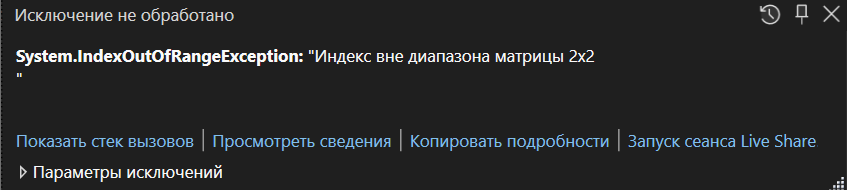


Рисунок 2 – Обращение вне диапазона двумерного массива

# Вывод

В ходе лабораторной работы были изучены принципы перегрузки операторов и методов, а также создавать индексирование получения для полей класса.