Министерство науки и высшего образования Российской Федерации Рязанский государственный радиотехнический университет

имени В.Ф. Уткина

Кафедра «Электронные вычислительные машины»

**Отчёт**

о лабораторной работе №3

«Переопределение методов и операторов»

по дисциплине:

«Визуальное программирование»

Выполнили:

Студенты группы 140

Бригады №1

Волкова Е.В.

Егорова Е.М.

Поверил:

ст. преп. Бастрычкин А.С.

ст. преп. Хизриева Н. И.

Рязань 2024

**Цель работы**: изучить механизм перегрузки на языке C#.

**Задание**:

1) Создать класс Matrix со свойствами Rows и Columns типа uint (неотрицательные целые, количество строк и столбцов соответственно), доступными только для чтения и с закрытым полем:

double [,] elements;

2) Создать конструктор, который принимает в качестве аргументов размерность матрицы и затем создает матрицу заданного размера, заполненную нулями:

public Matrix(uint rows, uint columns)

3) Создать копирующий конструктор:

public Matrix(Matrix matrix)

4) Создать индексатор, который позволяет получить и задать значение элемента матрицы:

public double this[uint row, uint column]

В тело индексатора нужно добавить обработку исключений в случае попытки обращения по индексам, выходящим за границы матрицы.

5) Переопределить метод ToString(), который будет выводить количество строк, столбцов и элементов матрицы.

6) Перегрузить метод ToString(), который будет выводить в строку n первых элементов матрицы.

7) Перегрузить операторы +, -, \*

public static Matrix operator +(Matrix left, Matrix right)

{

// TODO: реализация

}

8) Реализовать в классе Matrix интерфейс ICloneable.

Поскольку классы представляют ссылочные типы, то это накладывает некоторые ограничения на их использование. Например, если мы создадим две переменные

var a = new Matrix(2,2);

var b = a;

то a и b будут указывать на один и тот же объект в памяти, поэтому изменения элементов a затронут также и переменную b.

Чтобы переменная b указывала на новый объект, но со значениями из a, мы можем применить клонирование с помощью реализации интерфейса ICloneable.

9) Реализовать в классе Matrix интерфейс IComparable.

**Практическая часть.**

Код программы с описанием класса Matrix:

using System.Text;

namespace VP\_Lab3

{

public class Matrix : ICloneable, IComparable

{

private double[,] elements;

public uint Rows { get; }

public uint Columns { get; }

public Matrix(uint rows, uint columns)

{

Rows = rows;

Columns = columns;

elements = new double[rows, columns];

}

public Matrix(Matrix matrix) : this(matrix.Rows, matrix.Columns)

{

Array.Copy(matrix.elements, elements, matrix.elements.Length);

}

public double this[uint row, uint column]

{

get

{

if (row >= Rows || column >= Columns)

{

throw new IndexOutOfRangeException();

}

return elements[row, column];

}

set

{

if (row >= Rows || column >= Columns)

{

throw new IndexOutOfRangeException();

}

elements[row, column] = value;

}

}

public void FillWithRandomValues()

{

Random random = new Random();

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

{

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

{

elements[i, j] = random.Next(1, 10);

}

}

}

public override string ToString()

{

return $"Matrix: Rows={Rows}, Columns={Columns}, Elements={Rows \* Columns}";

}

public string ToString(int n)

{

StringBuilder sb = new StringBuilder();

for (int i = 0; i < Math.Min(n, Rows \* Columns); i++)

{

sb.Append(elements[i / Columns, i % Columns]).Append(" ");

}

return sb.ToString();

}

public string ShowMatrix()

{

string result = "";

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

{

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

{

result += elements[i, j] + " ";

}

result += "\n";

}

return result;

}

public static Matrix operator +(Matrix left, Matrix right)

{

Matrix result = new Matrix(left.Rows, left.Columns);

for (uint i = 0; i < left.Rows; i++)

{

for (uint j = 0; j < left.Columns; j++)

{

result[i, j] = left[i, j] + right[i, j];

}

}

return result;

}

public static Matrix operator -(Matrix left, Matrix right)

{

Matrix result = new Matrix(left.Rows, left.Columns);

for (uint i = 0; i < left.Rows; i++)

{

for (uint j = 0; j < left.Columns; j++)

{

result[i, j] = left[i, j] - right[i, j];

}

}

return result;

}

public static Matrix operator \*(Matrix left, Matrix right)

{

Matrix result = new Matrix(left.Rows, right.Columns);

for (uint i = 0; i < left.Rows; i++)

{

for (uint j = 0; j < right.Columns; j++)

{

for (uint k = 0; k < left.Columns; k++)

{

result[i, j] += left[i, k] \* right[k, j];

}

}

}

return result;

}

public object Clone()

{

return new Matrix(this);

}

public int CompareTo(object obj)

{

if (obj == null) return 1;

Matrix otherMatrix = (Matrix)obj;

double sum0 = 0;

double sumOther = 0;

foreach (double element in elements)

{

sum0 += element;

}

foreach (double element in otherMatrix.elements)

{

sumOther += element;

}

return sum0.CompareTo(sumOther);

}

}

}

Код программы с описанием интерфейса:

namespace VP\_Lab3

{

class Program

{

static void Main()

{

Matrix matrix1 = new Matrix(2, 2);

matrix1.FillWithRandomValues();

Matrix matrix2 = new Matrix(2, 2);

matrix2.FillWithRandomValues();

Matrix resultMatrix = matrix1 + matrix2;

Console.WriteLine("Matrix 1:");

Console.WriteLine(matrix1.ShowMatrix());

Console.WriteLine(matrix1.ToString(3));

Console.WriteLine("Matrix 2:");

Console.WriteLine(matrix2.ShowMatrix());

Console.WriteLine(matrix2.ToString(2));

//сумма

Console.WriteLine("Сумма матриц:");

Console.WriteLine(resultMatrix.ShowMatrix());

//разность

resultMatrix = matrix1 - matrix2;

Console.WriteLine("Разность матриц:");

Console.WriteLine(resultMatrix.ShowMatrix());

//произведение

Console.WriteLine("Произведение мотриц:");

resultMatrix = matrix1 \* matrix2;

Console.WriteLine(resultMatrix.ShowMatrix());

// клонируем матрицу

Matrix clonedMatrix = (Matrix)matrix1.Clone();

// сравниваем матрицы

int comparisonResult = matrix1.CompareTo(matrix2);

if (comparisonResult < 0)

{

Console.WriteLine("Matrix 1 меньше Matrix 2");

}

else if (comparisonResult > 0)

{

Console.WriteLine("Matrix 1 больше Matrix 2");

}

else

{

Console.WriteLine("Matrix 1 = Matrix 2");

}

}

}

}

Результат работы программы на рисунке 1.

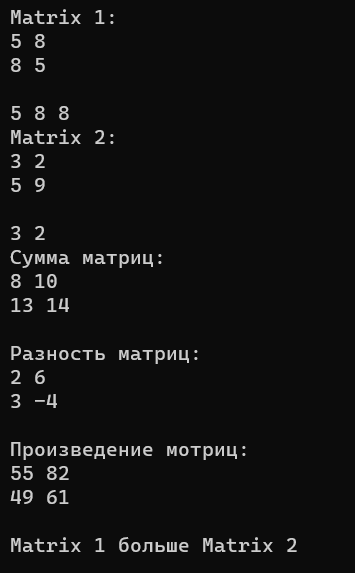
****

Рисунок 1 – Результат работы программы

**Заключение**

В ходе выполнения лабораторной работы был изучен механизм перегрузки на языке C#.