Оглавление

[Цель работы 3](#_Toc512170465)

[Задание 3](#_Toc512170466)

[Краткая теория: 4](#_Toc512170467)

[WPF 4](#_Toc512170468)

[Шаблон проектирования 4](#_Toc512170469)

[Матрицы: основные определения и понятия 5](#_Toc512170470)

[Результаты работы программы: 6](#_Toc512170471)

[Выводы: 8](#_Toc512170472)

[Список использованных литературных источников 9](#_Toc512170473)

[Приложение с листингом исходных файлов 10](#_Toc512170474)

Цель работы**:** В рамках курса «Мультимедиа-технологии» необходимо выполнить одну лабораторную работу. В процессе обучения по направлению «Специалист по информационным системам и технологиям» предполагается, что студенты постепенно реализуют проект в одной из выбранных предметных областей, добавляя к нему новые модули по мере изучения новых дисциплин. Поэтому задание на лабораторную работу является творческим и заключается в реализации какой-либо части проекта с использованием мультимедиа-технологий.

Задание**:** Реализовать приложение для работы с матрицами на WPF. Реализовать основные функции для работы с матрицами:

1. Умножение
2. Сложение
3. Вычитание
4. Проверка вводимых данных

# Краткая теория:

## WPF

Данная программа будет написана на языке C#, с использованием технологий WPF (Windows Presentation Foundation). Технология WPF является частью экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов.

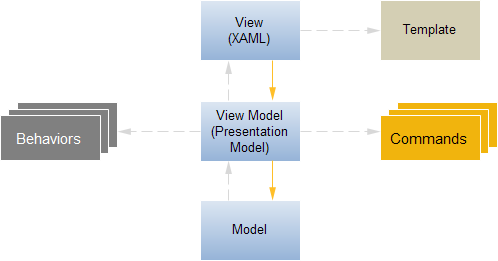
Одной из важных особенностей является использование языка декларативной разметки интерфейса XAML, основанного на XML, что позволяет создавать насыщенный графический интерфейс, используя декларативное объявление интерфейса, или код на управляемых языках C# и VB.NET, либо совмещать и то, и другое

## Шаблон проектирования

Программная система будет разработана с использование шаблона(паттерна) MVVM(Model-View-ViewModel).

MVVM — паттерн (шаблон) проектирования приложения, ориентированный на современные платформы разработки, такие как Windows Presentation Foundation, Silverlight от компании Microsoft, ZK framework. MVVM удобно использовать вместо классического MVC и ему подобных в тех случаях, когда в платформе, на которой ведётся разработка, присутствует «связывание данных». В шаблонах проектирования MVC/MVP изменения в пользовательском интерфейсе не влияют непосредственно на Mодель, а предварительно идут через Контроллер (англ. Controller) или Presenter. В таких технологиях как WPF и Silverlight есть концепция «связывания данных», позволяющая связывать данные с визуальными элементами в обе стороны.

Общая реализация шаблона MVVM:



Где:

* **Model** - есть полноценный набор объектов, которые отображают («моделируют», отсюда и название) объекты внешнего мира различной степени абстракции.
* **ViewModel** – уровень бизнес-логики.
* **View** – представление ViewModel

## Матрицы: основные определения и понятия

Матрицей размера http://www.webmath.ru/poleznoe/images/matrix/formules_625.png называется прямоугольная таблица, содержащая http://www.webmath.ru/poleznoe/images/matrix/formules_626.png чисел, состоящая из http://www.webmath.ru/poleznoe/images/formules_127.png строк и http://www.webmath.ru/poleznoe/images/formules_128.png столбцов.

Если матрица содержит http://www.webmath.ru/poleznoe/images/formules_127.png строк и http://www.webmath.ru/poleznoe/images/formules_128.png столбцов, то матрица называется **матрицей размера** http://www.webmath.ru/poleznoe/images/matrix/formules_625.png или http://www.webmath.ru/poleznoe/images/matrix/formules_625.png-матрицей. Размер матрицы указывается справа внизу возле ее имени, либо таблицы с обозначением элементов.

**Операции над матрицами:**

**Сложение**: Суммой матриц http://www.webmath.ru/poleznoe/images/vector/formules_422.png и http://www.webmath.ru/poleznoe/images/vector/formules_423.png одного размера называется матрица http://www.webmath.ru/poleznoe/images/matrix/formules_684.png такого же размера, получаемая из исходных путем сложения соответствующих элементов:

Сложение матриц

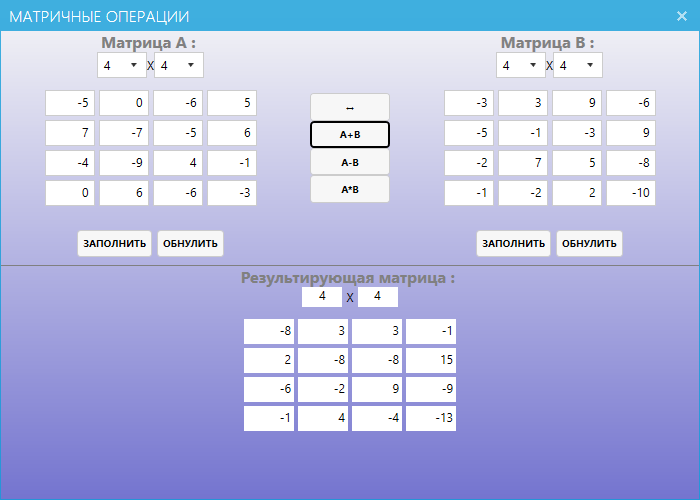
**Вычитание:** Разностью матриц http://www.webmath.ru/poleznoe/images/vector/formules_422.png и http://www.webmath.ru/poleznoe/images/vector/formules_423.png одного и того же размера называется матрица http://www.webmath.ru/poleznoe/images/matrix/formules_744.png такого же размера, получаемая из исходных путем прибавления к матрице http://www.webmath.ru/poleznoe/images/vector/formules_422.png матрицы http://www.webmath.ru/poleznoe/images/vector/formules_423.png, умноженной на (-1).

**Умножение:** Произведением матрицы http://www.webmath.ru/poleznoe/images/matrix/formules_698.png на матрицу http://www.webmath.ru/poleznoe/images/matrix/formules_699.png называется матрица http://www.webmath.ru/poleznoe/images/matrix/formules_700.png такая, что элемент матрицы http://www.webmath.ru/poleznoe/images/matrix/formules_690.png, стоящий в http://www.webmath.ru/poleznoe/images/matrix/formules_639.png-ой строке и http://www.webmath.ru/poleznoe/images/matrix/formules_640.png-ом столбце, т.е. элемент http://www.webmath.ru/poleznoe/images/matrix/formules_701.png, равен сумме произведений элементов http://www.webmath.ru/poleznoe/images/matrix/formules_639.png-ой строки матрицы http://www.webmath.ru/poleznoe/images/vector/formules_422.png на соответствующие элементы http://www.webmath.ru/poleznoe/images/matrix/formules_640.png-ого столбца матрицы http://www.webmath.ru/poleznoe/images/vector/formules_423.png.

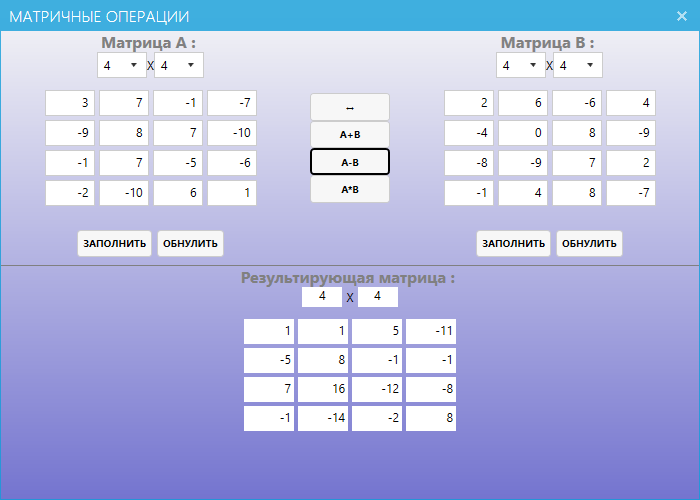
**Деление:** В теории матриц нет понятия «деления матрицы», матрицы можно только умножать. Если нужно разделить матрицу на некоторое число *k*, то используется термин умножить матрицу на дробь 1/*k*. А вместо «разделить матрицу *A* на матрицу *B*» говорят, что нужно [умножить матрицу](http://ru.solverbook.com/spravochnik/matricy/umnozhenie-matric/) *A* на матрицу *B-1*, где *B-1* – [обратная матрица](http://ru.solverbook.com/spravochnik/matricy/obratnaya-matrica/) к матрице *B*.

# Результаты работы программы:

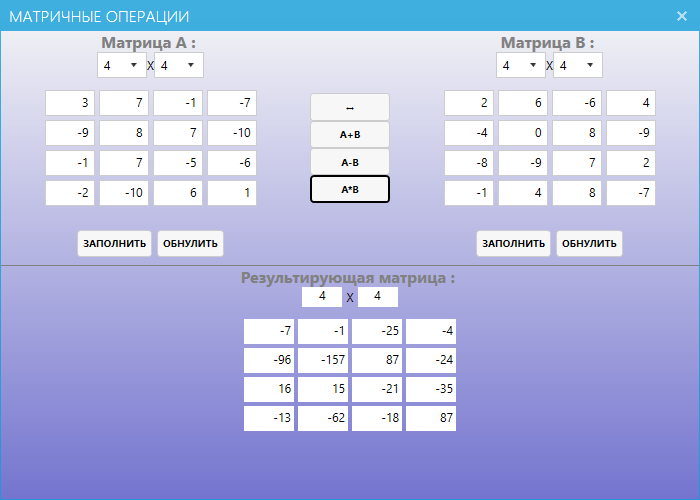
Интерфейс приложения (на котором представлен результат сложения):



Вычитание:



Умножение:



# Выводы:

В ходе выполнения лабораторной работе были получены навыки программирования на языке C#, с использованием графических возможностей WPF, а также изучен и применен паттерн MVVM. Изучены и реализованы основные методы для работы с матрицами (сложение, вычитание, умножение).

# Список использованных литературных источников

1. Р. Беллман, Введение в теорию матриц
2. C# 4.0: полное руководство, Герберт Шилдт.
3. WPF 4.0 Полное руководство, Адам Натан.

# Приложение с листингом исходных файлов

MatrixView.xaml

<Grid>

            <Grid.RowDefinitions>

                <RowDefinition Height="auto"/>

                <RowDefinition/>

                <RowDefinition Height="auto"/>

            </Grid.RowDefinitions>

            <StackPanel>

                <TextBlock VerticalAlignment="Center"

                   HorizontalAlignment="Center"

                   FontWeight="Bold" Foreground="Gray"

                   FontSize="16">

                    <Run Text="Матрица"/>

                    <Run Text="{Binding Name, Mode=OneWay}"/>

                    <Run Text=": "/>

                </TextBlock>

                <StackPanel Orientation="Horizontal" HorizontalAlignment="Center">

                    <ComboBox ItemsSource="{Binding RowSizes}" SelectedItem="{Binding M}">

                    </ComboBox>

                    <TextBlock Text="X" VerticalAlignment="Center"/>

                    <ComboBox ItemsSource="{Binding ColumnSizes}" SelectedItem="{Binding N}">

                    </ComboBox>

                </StackPanel>

            </StackPanel>

            <ItemsControl Grid.Row="1"

              ItemsSource="{Binding Elements}"

              HorizontalAlignment="Center"

              VerticalAlignment="Top"

              Padding="10">

                <ItemsControl.ItemsPanel>

                    <ItemsPanelTemplate>

                        <UniformGrid Rows="{Binding M}" Columns="{Binding N}"/>

                    </ItemsPanelTemplate>

                </ItemsControl.ItemsPanel>

                <ItemsControl.ItemTemplate>

                    <DataTemplate>

                        <TextBox Width="50" Height="25" HorizontalContentAlignment="Right"

                         Margin="2">

                            <TextBox.Text>

                                <Binding Path="Value" NotifyOnValidationError="True"

                                 UpdateSourceTrigger="PropertyChanged">

                                    <Binding.ValidationRules>

                                        <vm:NumericRule/>

                                    </Binding.ValidationRules>

                                </Binding>

                            </TextBox.Text>

                        </TextBox>

                    </DataTemplate>

                </ItemsControl.ItemTemplate>

            </ItemsControl>

            <StackPanel Grid.Row="2"

                Orientation="Horizontal"

                HorizontalAlignment="Center"

                Margin="0,0,0,8">

                <Button Command="{Binding RandomFillCommand}" Content="Заполнить"/>

                <Button Command="{Binding ClearCommand}" Content="Обнулить" Margin="5,0,0,0"/>

            </StackPanel>

        </Grid>

MainView(главное окно):

<Grid>

        <Grid.RowDefinitions>

            <RowDefinition/>

            <RowDefinition/>

        </Grid.RowDefinitions>

        <Grid Grid.Row="0" HorizontalAlignment="Stretch">

            <Grid.ColumnDefinitions>

                <ColumnDefinition/>

                <ColumnDefinition Width="100"/>

                <ColumnDefinition/>

            </Grid.ColumnDefinitions>

            <views:MatrixView Grid.Column="0" DataContext="{Binding MatrixA}"/>

            <StackPanel Grid.Column="1" Width="80" VerticalAlignment="Center">

                <Button Content="&#x2194;" Command="{Binding SwapMatrixCommand}" ToolTip="Перестановка матриц"/>

                <Button Content="A+B" Command="{Binding SumMatrixCommand}"  ToolTip="Сложение матриц"/>

                <Button Content="A-B" Command="{Binding SubMatrixCommand}"  ToolTip="Вычитание матриц"/>

                <Button Content="A\*B" Command="{Binding MultMatrixCommand}" ToolTip="Умножение матриц"/>

            </StackPanel>

            <views:MatrixView Grid.Column="2" DataContext="{Binding MatrixB}"/>

        </Grid>

        <Grid Grid.Row="1">

            <Grid.RowDefinitions>

                <RowDefinition Height="auto"/>

                <RowDefinition/>

            </Grid.RowDefinitions>

            <Border BorderBrush="Gray" BorderThickness="2" Height="1" VerticalAlignment="Top"/>

            <views:ResultMatrixView Grid.Row="1" DataContext="{Binding MatrixC}"/>

        </Grid>

    </Grid>

Matrix.cs

public class Matrix

{

public string Name { get; set; }

public int RowsCount { get; private set; }

public int ColumnsCount { get; private set; }

public double this[int i, int j]

{

get { return mass[i, j]; }

set { mass[i, j] = value; }

}

double[,] mass;

#region Constructors

public Matrix()

{

mass = new double[RowsCount, ColumnsCount];

}

public Matrix(int N)

{

RowsCount = N;

ColumnsCount = N;

mass = new double[RowsCount, ColumnsCount];

}

public Matrix(int rowsCount, int columnsCount)

{

RowsCount = rowsCount;

ColumnsCount = columnsCount;

mass = new double[RowsCount, ColumnsCount];

}

public Matrix(double[,] mass)

{

this.mass = mass;

RowsCount = mass.GetLength(0);

ColumnsCount = mass.GetLength(1);

}

public Matrix(Matrix matrix)

{

mass = new double[matrix.RowsCount, matrix.ColumnsCount];

RowsCount = matrix.RowsCount;

ColumnsCount = matrix.ColumnsCount;

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

{

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

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

}

}

#endregion

public static Matrix Mult(Matrix A, Matrix B)

{

if (!(A.ColumnsCount == B.RowsCount))

{

throw new ArgumentException("Произведение матрицы А на В не имеет смысла, т.к. число столбоц матрицы A не равно числу строк матрицы В");

}

Matrix res = new Matrix(A.RowsCount, B.RowsCount);

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

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

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

res[i, j] += A[i, k] \* B[k, j];

return res;

}

public static Matrix Mult(Matrix A, int scalar)

{

Matrix temp = new Matrix(A);

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

{

for (int j = 0; j < A.ColumnsCount; j++)

{

temp[i, j] = temp[i, j] \* scalar;

}

}

return temp;

}

public static Matrix Div(Matrix A, Matrix B)

{

return A \* B.Invert();

}

public static Matrix Div(Matrix A, double num)

{

Matrix res = new Matrix(A.RowsCount, A.RowsCount);

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

for (int j = 0; j < A.RowsCount; j++)

res[i, j] += A[i, j] \* 1 / num;

return res;

}

public static Matrix Add(Matrix A, Matrix B)

{

if (!(A.RowsCount == B.RowsCount && A.ColumnsCount == B.ColumnsCount))

{

throw new ArgumentException("Сложение не имеет смысла, т.к. размеры матриц не одинаковы");

}

Matrix res = new Matrix(A.RowsCount, A.ColumnsCount);

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

{

for (int j = 0; j < A.ColumnsCount; j++)

{

res[i, j] = A[i, j] + B[i, j];

}

}

return res;

}

public static Matrix Sub(Matrix A, Matrix B)

{

if (!(A.RowsCount == B.RowsCount && A.ColumnsCount == B.ColumnsCount))

{

throw new ArgumentException("Вычитание не имеет смысла, т.к. размеры матриц не одинаковы");

}

Matrix res = new Matrix(A.RowsCount, A.ColumnsCount);

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

{

for (int j = 0; j < A.ColumnsCount; j++)

{

res[i, j] = A[i, j] - B[i, j];

}

}

return res;

}

public Matrix Invert()

{

if (RowsCount != ColumnsCount)

{

throw new ArgumentException("Обратная матрица существует только для квадратных");

}

double determinant = DetRec();

if (determinant == 0)

{

throw new ArgumentException("Детерминант матрицы равен нулю");

}

Matrix matrix = new Matrix(RowsCount);

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

{

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

{

Matrix tmp = Exclude(i, j);

matrix[j, i] = (1 / determinant) \* Math.Pow(-1, i + j) \* detRec(tmp.mass);

}

}

return matrix;

}

// Перегрузка оператора сложения

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

{

return Add(a, b);

}

// Перегрузка оператора вычитания

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

{

return Sub(a, b);

}

// Перегрузка оператора умножения

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

{

return Mult(a, b);

}

// Перегрузка оператора деления

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

{

return Div(a, b);

}

public double MaxElement()

{

double max = mass[0, 0];

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

{

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

{

if (mass[i,j] > max)

{

max = mass[i, j];

}

}

}

return max;

}

public double MinElement()

{

double min = mass[0, 0];

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

{

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

{

if (mass[i, j] < min)

{

min = mass[i, j];

}

}

}

return min;

}

public double DetRec()

{

return detRec(mass);

}

public override string ToString()

{

string res = "";

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

{

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

{

//res += string.Format("{0: 0.00; -0.00; 0.00}", mass[i,j]);

res += string.Format("{0,7:0.00}", mass[i, j]);

}

res += "\n";

}

return res;

}

double detRec(double[,] matrix)

{

if (matrix.Length == 4)

{

return matrix[0, 0] \* matrix[1, 1] - matrix[0, 1] \* matrix[1, 0];

}

double sign = 1, result = 0;

for (int i = 0; i < matrix.GetLength(1); i++)

{

double[,] minor = GetMinor(matrix, i);

result += sign \* matrix[0, i] \* detRec(minor);

sign = -sign;

}

return result;

}

double[,] GetMinor(double[,] matrix, int n)

{

double[,] result = new double[matrix.GetLength(0) - 1, matrix.GetLength(0) - 1];

for (int i = 1; i < matrix.GetLength(0); i++)

{

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

result[i - 1, j] = matrix[i, j];

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

result[i - 1, j - 1] = matrix[i, j];

}

return result;

}

Matrix Exclude(int row, int column)

{

if (row > RowsCount || column > ColumnsCount)

throw new IndexOutOfRangeException("Строка или столбец не принадлежат матрице.");

Matrix ret = new Matrix(RowsCount - 1, ColumnsCount - 1);

int offsetX = 0;

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

{

int offsetY = 0;

if (i == row)

{

offsetX++;

continue;

}

for (int t = 0; t < ColumnsCount; t++)

{

if (t == column)

{

offsetY++;

continue;

}

ret[i - offsetX, t - offsetY] = mass[i, t];

}

}

return ret;

}

}