МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Новосибирский национальный исследовательский государственный университет»
(Новосибирский государственный университет, НГУ)
Структурное подразделение Новосибирского государственного университета – Высший колледж информатики Университета (ВКИ НГУ)
КАФЕДРА ИНФОРМАТИКИ

ОТЧЕТ КУРСОВОГО ПРОЕКТА

Разработка программных модулей

# РАЗРАБОТКА ДЕСКТОПНОГО ПРИЛОЖЕНИЯ ДЛЯ ОС WINDOWS «Калькулятор Матриц»

Руководитель Пауль С.А.
Ассистент ВКИ НГУ

«___»_____2023 г.

Студент 3 курса Кризо Д.В.
гр. 107в2 «____»_____2023 г.

Новосибирск

2023

# СОДЕРЖАНИЕ

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫХ ОБОЗНАЧЕНИЙ И ТЕРМИНОВ

UI (Интерфейс пользователя) — это совокупность элементов, методов и средств, обеспечивающих взаимодействие человека с программным или аппаратным обеспечением, с целью выполнения определенных задач с максимальным комфортом и эффективностью.

Минор матрицы — это определитель, полученный путем выбора определенного поднабора строк и столбцов из исходной матрицы.

Стековый калькулятор — это тип программы или алгоритма, который использует структуру данных стек для выполнения математических операций. В таком калькуляторе числа и операции сохраняются на стеке, и операции выполняются в порядке Last In, First Out (LIFO), то есть последняя операция, добавленная в стек, выполняется первой.

DataGrid — представляет элемент управления, отображающий данные в настраиваемой сетке.

ComboBox — представляет элемент управления для выбора с раскрывающимся списком, который можно отображать и скрывать, щелкая стрелку в элементе управления.

LINQ (Language Integrated Query) — это технология запросов, встроенная в языки программирования, такие как C# и VB.NET. Она обеспечивает унифицированный и выразительный способ работы с данными из различных источников с использованием стандартных языковых конструкций.

MVVM (Model-View-ViewModel) — это архитектурный шаблон разработки программного обеспечения, который разделяет пользовательский интерфейс на три основных компонента: Model (модель данных), View (представление) и ViewModel (модель представления). MVVM помогает улучшить структуру кода, облегчает тестирование и поддерживает разделение логики пользовательского интерфейса от бизнес-логики.

# ВВЕДЕНИЕ

В современной образовательной практике важное место занимает изучение математики, и, в частности, темы матриц. Ученики, только начинающие свой путь в изучении высшей математики, часто сталкиваются с трудностями при выполнении операций над матрицами. Для облегчения этого процесса и создано приложение – простой и доступный калькулятор матриц.

Актуальность данной задачи обусловлена необходимостью предоставления учащимся инструмента для освоения и проверки базовых операций над матрицами. Целью данного исследования является разработка простого и понятного программного продукта, ориентированного на школьников и студентов, только начинающих свой путь в изучении матриц.

В контексте образовательной среды, где ключевым является облегчение процесса обучения, создание такого приложения приобретает практическую ценность и значимость. Продукт ориентирован на широкий круг пользователей, обладая простым интерфейсом и базовым набором функций, необходимых для успешного выполнения задач по матричной алгебре, так как на данный момент приложения для работы с матрицами существуют преимущественно на онлайн-сервисах, а встроенный калькулятор Windows не обладает функциями для работы с матрицами.

В ходе исследования уделяется внимание не только программной части приложения, но и его адаптации к уровню знаний и потребностям начинающих учеников. Основные методы исследования включают в себя анализ образовательных потребностей, проектирование простого и интуитивно понятного интерфейса, а также тестирование приложения с участием целевой аудитории.

Таким образом, данное исследование направлено на создание образовательного программного продукта, предоставляющего новичкам в изучении матриц простой, но эффективный инструмент для выполнения базовых операций и улучшения понимания материала.

# 1 ПОСТАНОВКА ЗАДАЧИ КП

Данное программное приложение представляет собой инструмент, созданный для учеников, только начинающих изучение матриц в рамках курса высшей математики. Основной целью приложения является предоставление простого и понятного средства для выполнения базовых операций над матрицами.

Задачи по обеспечению условий разработки и функционирования приложения:

1. Разработка функционала операций над матрицами: Программирование основных операций, таких как сложение, умножение, нахождение обратной матрицы и других, обеспечивая точность вычислений и интуитивную понятность.

2. Тестирование производительности: Проведение тестирования для проверки стабильности и эффективности работы приложения при выполнении различных матричных операций.

Задачи по проектированию и созданию приложения:

1. Проектирование простого пользовательского интерфейса (UI): Разработка интуитивно понятного интерфейса, обеспечивающего легкость в использовании, даже для тех, кто только начал изучение матриц.

2. Оптимизация процесса ввода данных: Создание удобного механизма ввода матриц.

3. Добавление функционала сохранения результатов: Реализация возможности сохранения результатов вычислений для обеспечения удобства работы с приложением.

## 1.1 Пользовательские требования

Приложение должно удовлетворять следующим пользовательским требованиям:

- Пользователи должны иметь возможность выполнять основные операции над матрицами, включая сложение, вычитание, умножение, умножение на число, нахождение обратной матрицы, транспонирование матрицы;
- Интерфейс приложения должен быть простым и понятным даже для пользователей, только начинающих изучение матриц. Элементы управления и ввод данных должны быть легкими в использовании, не требующими специальных навыков;
- Создание механизма ввода матриц при помощи клавиатуры. Возможность быстрого копирования и вставки матриц из других источников;
- Возможность сохранения результатов вычислений для последующего использования;
- Приложение должно быть автономным и не требовать постоянного соединения с интернетом для выполнения основных функций.

### 1.2 Системные требования

Для полноценной работы приложения необходим компьютер с операционной системой Windows 8 и выше.

### 1.3 Требования к графическому пользовательскому интерфейсу

Приложение должно удовлетворять следующим требованиям к графическому пользовательскому интерфейсу:

- Интерфейс должен быть интуитивно понятным для большинства пользователей. Простые и понятные элементы управления ускоряют процесс использования;
- Поддержка различных разрешений для обеспечения удобного использования приложения на различных экранах;
- Возможность удобного ввода матриц с использованием текстовых полей и обработка ошибок ввода пользователя;

- Предоставление ясных и легко доступных кнопок для выполнения основных операций, таких как сложение, умножение, транспонирование и др.;
- Явное отображение результатов операций и информационных сообщений, чтобы пользователь всегда был в курсе текущего состояния приложения;
- Возможность просмотра истории выполненных операций, что обеспечивает удобный способ вернуться к предыдущим результатам;
- Реализация функционала сохранения результатов вычислений в буфер обмена для удобства работы.

### 1.4 План-график выполнения КП

Разработка приложения состояла из следующих этапов:

Таблица 1 – План-график выполнения КП

| **Задача** | **Предполагаемое время на выполнение задачи** | **Затраченное время на выполнение задачи** |
|---|---|---|
| Продумывание идеи приложения | 1 день | 1 день |
| Определение функций приложения | 1 час | 3 часа |
| Определение дизайна приложения | 2 часа | 2 часа |
| Создание окна расширенного калькулятора | 1 день | 2 дня |
| Обработка возможных ошибок в расширенном калькуляторе | 3 часа | 7 часов |
| Создание окна матричного калькулятора | 1 день | 2 дня |
| Обработка ввода, частных случаев и возможных ошибок при использовании калькулятора матриц | 4 часа | 8 часов |
| Создание перемещения между калькуляторами в одном окне | 3 часа | 5 часов |

# 2 АНАЛИЗ ТРЕБОВАНИЙ И ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ

## 2.1 Описание предметной области задачи КП

Предметная область данной выпускной квалификационной работы связана с областью линейной алгебры и математическими операциями над матрицами. В рамках разработки калькулятора матриц рассматриваются ключевые аспекты, обуславливающие необходимость создания программного решения в данной тематике.

1. Линейная алгебра и матрицы:

a. Вводится концепция матриц как математического объекта, представляющего собой двумерный массив чисел, имеющих различные математические свойства [1].

b. Рассматриваются основные операции, такие как сложение, вычитание, умножение, нахождение обратной матрицы.

2. Проблемы в вычислениях с матрицами:

a. Сложные операции над матрицами могут быть подвержены ошибкам при ручном выполнении, что подчеркивает необходимость автоматизированных решений.

b. Некоторые калькуляторы матриц ограничены функциональностью или не предоставляют удобного интерфейса для пользователя.

3. Роль калькулятора матриц:

a. Калькулятор матриц может быть эффективным инструментом для обучения линейной алгебре и практического применения математических концепций.

b. В различных областях, таких как физика, экономика, информатика, калькулятор матриц может использоваться для решения реальных задач.

4. Существующие наработки и их ограничения:

a. Некоторые существующие приложения предоставляют ограниченные возможности в обработке матриц и не удовлетворяют всем потребностям пользователя.

b. Некоторые калькуляторы матриц могут иметь сложный интерфейс, что затрудняет их использование для начинающих пользователей.

### 2.1.1 Информационные объекты предметной области и взаимосвязи между ними

В приложении присутствуют следующие информационные объекты, выполняющие определенные функции и связанные между собой:

1. Расширенный калькулятор – объект, позволяющий пользователю посчитать необходимые операции, включающие в себя алгебраические и тригонометрические функции. Содержит в себе запись результатов и выражений, которые могут быть использованы в работе с матрицами.

2. Калькулятор матриц – объект, позволяющий пользователю проводить различные операции подсчета и работы с матрицами. Обладает функцией копирования результирующей матрицы, которая впоследствии может быть использована пользователем в его дальнейших целях.

### 2.1.2 Информационные и функциональные потребности пользователей разрабатываемой ПС (ПМ)

Приложение должно выполнять следующие задачи:

- Интуитивно понятный интерфейс и удобный ввод матриц;
- Расчет и выполнение основных операций над матрицами;
- Расчет выражений, включающих в себя алгебраические и тригонометрические функции;
- Реализация перехода между калькуляторами с сохранением введенных данных и расчетов;
- Реализация копирования результатов в буфер обмена.

### 2.1.3 Методы работы с информационными объектами предметной области

#### *2.1.3.1 Используемые математические модели*

- Суммой двух матриц $A = (a_{ij})$ и $B = (b_{ij})$ одинакового порядка называют матрицу $C = (c_{ij})$ такого же порядка, элементы которой равны сумме соответствующих элементов матриц A и B, то есть $c_{ij} = a_{ij} + b_{ij}$ [2];
- Разностью двух матриц $A = (a_{ij})$ и $B = (b_{ij})$ одинакового порядка называют матрицу $C = (c_{ij})$ такого же порядка, элементы которой равны разности соответствующих элементов матриц A и B, то есть $c_{ij} = a_{ij} - b_{ij}$;
- Умножение матриц — операция, при которой элемент $c_{ij}$ матрицы-результата C равен сумме произведений элементов A и B в соответствующей строке и столбце: $c_{ij} = \sum_k a_{ik} * b_{kj}$;
- Умножение матрицы на число заключается в умножении каждого элемента матрицы на данное число. Если $A = (a_{ij})$ – матрица, а $k$ – число, то результатом будет новая матрица $B = (b_{ij})$где $b_{ij} = k * a_{ij}$;
- Определитель матрицы – числовое значение, связанное с квадратной матрицей. Для матрицы $A = (a_{ij})$ порядка n, определитель $\det(A)$ рассчитывается по определенному правилу, зависящему от порядка матрицы [3]. Определитель для матрицы второго порядка $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, определитель вычисляется по формуле $\det(A) = a_{11} * a_{22} - a_{12} * a_{21}$. Определитель для матрицы третьего порядка $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$, определитель вычисляется по формуле $\det(A) = a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31})$. Определитель для матрицы A порядка n (n больше 3) $A = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mm} \end{bmatrix}$ вычисляется следующим

образом: $\det(A) = \sum_{i=1}^{m}(-1)^{1+i} * a_{1i} * M_{1i}$, где $M_{ij}$ — дополнительный минор квадратной матрицы — определитель матрицы, полученной из исходной вычеркиванием i-ой строки и j-ого столбца;

- Транспонирование матрицы заключается в замене строк матрицы ее столбцами и наоборот. Если $A = (a_{ij})$ - матрица, то транспонированная матрица $B = A^T$ определяется как $b_{ij} = a_{ji}$;
- Обратная матрица для квадратной матрицы $A$, обозначаемая $A^{-1}$, существует, если определитель матрицы не равен нулю. Обратная матрица обладает свойством: $A * A^{-1} = A^{-1} * A = I$, где $I$ – единичная матрица.

#### *2.1.3.2 Применяемые программные технологии, основанные на математических моделях*

Для выполнения сложных математических операций был использован внутренний класс System.Math (System.Runtime Version 4.1.2.0), который предоставляет константы и статические методы для тригонометрических, логарифмических и иных общих математических функций.

### 2.1.4 Обзор существующих программных реализаций решения задачи

Встроенные программные средства Windows (Калькулятор Windows) не позволяют вести расчеты матриц, а онлайн-сервисы требуют постоянного подключения к Интернету. Это влечет за собой идею разработки такого калькулятора матриц, который бы мог работать без подключения к Интернету в объединении с обычным калькулятором в одном приложении.

### 2.1.5 Концептуальное обоснование разработки

Таким образом, приложение «Калькулятор Матриц» решает проблемы, приведенные в пункте 2.1.4, то есть, совмещает в себе калькулятор матриц и расширенный калькулятор для расчетов алгебраических и

тригонометрических выражений в одном приложении без подключения к Интернету.

### 2.2 Классы и характеристики пользователей

Приложение "Калькулятор Матриц" ориентировано на один основной класс пользователей — студентов начальных курсов и предполагает работу с одним пользователем соответственно. Помимо студентов, приложение также открыто для других пользователей, интересующихся математикой и матричными вычислениями. Это может включать широкий спектр людей, начиная от студентов школы до любознательных людей, желающих углубить свои знания в области линейной алгебры.

Таким образом, "Калькулятор Матриц" разработан с учетом разнообразных потребностей пользователей, предоставляя инструмент для успешного выполнения матричных операций как в образовательных, так и профессиональных целях.

### 2.3 Функциональные требования

Функциональные требования определяют функциональность программного обеспечения, то есть описывают, какое поведение должна предоставлять разрабатываемая система. Для реализации поставленной цели были составлены функциональные требования, которые необходимо реализовать:

- Создание удобного перехода между двумя калькуляторами с возможностью переносить данные из одного в другой;
- Создание интуитивно понятного калькулятора матриц с различным набором операций и удобным вводом и обычного стекового калькулятора с алгебраическими и тригонометрическими операторами;
- Создание возможности копировать результаты в буфер обмена после обработки того или иного выражения или матрицы.

Диаграмма прецедентов - диаграмма поведения приложения (Рисунок 1), на которой показано множество прецедентов, а также отношения между ними [4].

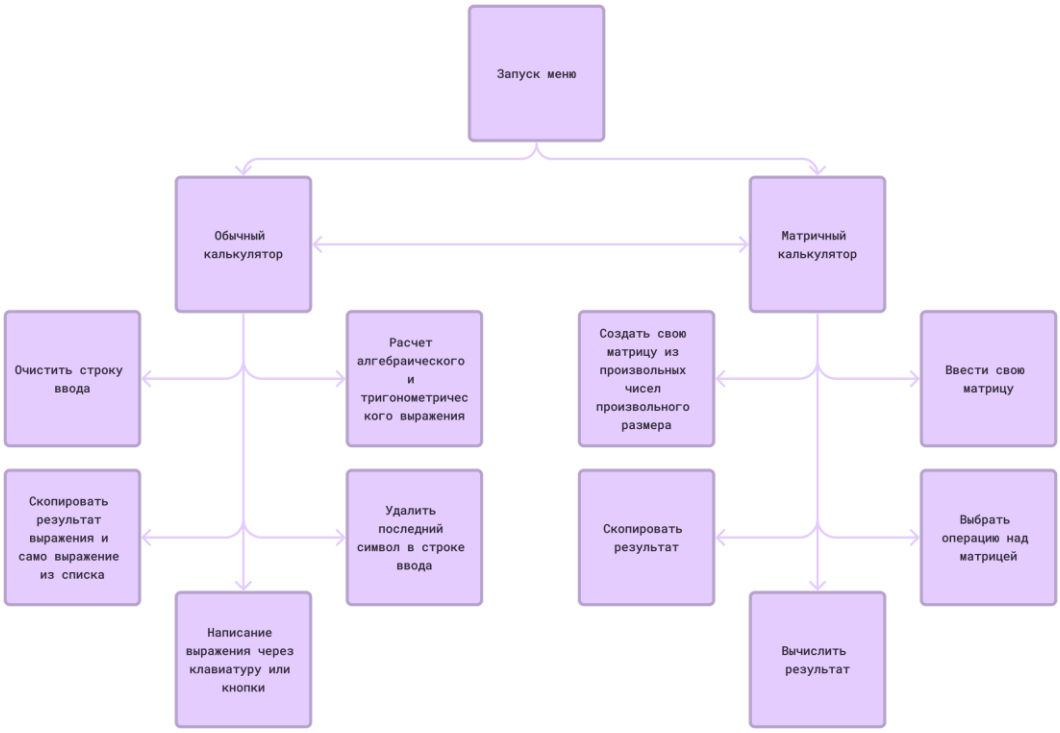


Рисунок 1 - Диаграмма прецедентов для приложения «Калькулятор Матриц»

### 2.4 Нефункциональные требования

Программа обладает функционалом, который дает пользователю обратную связь при попытке работы с неправильными данными:

- Предусмотрена валидация ввода данных матрицы, когда пользователь хочет написать несколько пробелов или переходов на новую строку, то программа просто не даст этого сделать, а будет убирать лишние знаки. Также, о том, правильно введена матрица или в ней присутствуют ошибки, при которых невозможно выполнить операции над матрицей, сообщает подпись под окном ввода, обладающая тремя состояниями:

NoMatrix – отсутствие матрицы; Error – присутствуют ошибки при вводе матрицы; Valid – матрица введена верно, можно выполнять операции.

- Если пользователь попробует выполнить операции с неправильно введённой матрицей, пустой матрицей или с матрицей, имеющей математические ошибки (например, несоответствие количества символов в строках, попытка найти обратную матрицу, чей определитель равен нулю и т.д.), то программа сообщит ему об этом всплывающем окном с описанной ошибкой.
- При работе с обычным калькулятором пользователь будет оповещён сообщением об ошибке, если его выражение содержит в себе ошибки, которые не позволяют вычислить результат выражения (например, лишние скобки и операторы, факториал слишком большого числа, наличие лишних символов в выражении, деление на 0).

# 3 ВЫБОР ПРОГРАММНЫХ СРЕД И СРЕДСТВ РАЗРАБОТКИ

В качестве основных критериев при выборе языков программирования и среды разработки были выбраны: простота реализации и удобство. Исходя из этого, была выбрана платформа Windows Presentation Foundation (WPF), язык разметки XAML, а также язык программирования C#.

WPF - это платформа для создания клиентских приложений с гибким пользовательским интерфейсом. Она позволяет разрабатывать программы с разнообразными интерфейсами, функциональностью и работой с базой данных [5]. Преимущества WPF:

- WPF использует веб-подобную модель компоновки, что обеспечивает гибкость в размещении элементов управления и адаптации интерфейса под различные языки или динамичное содержимое.
- WPF позволяет отображать стилизованный текст с использованием расширенных средств отображения документов.
- Использование стилей и шаблонов позволяет стандартизировать форматирование и легко изменять отображение элементов управления.
- WPF предоставляет абстракцию для определения команд приложения, которые можно использовать для множества элементов управления.
- Интерфейс создается с использованием XAML, отделяя его от кода и позволяя дизайнерам графики работать с ним в профессиональных инструментах.

XAML (eXtensible Application Markup Language) — это язык разметки, основанный на XML и разработанный Microsoft для декларативного программирования приложений. Он используется для описания пользовательского интерфейса и структуры приложений. В XAML выделяют четыре основные категории элементов: панели, элементы управления, элементы, связанные с документом и графические фигуры [6].

Преимущества использования XAML:

- Описывая разметку в XAML, можно легко создавать адаптивные интерфейсы с использованием различных панелей и выравнивания по горизонтали и вертикали.
- XAML позволяет использовать стили для задания свойств элементов управления, что обеспечивает удобство и единообразие в дизайне. Стили могут наследоваться и применяться выборочно.
- Используя шаблоны, можно выводить элементы по заданному формату, управлять отображением коллекций объектов и изменять их дизайн без необходимости повторного кодирования.
- XAML поддерживает привязку данных, что позволяет связывать значения свойств элементов интерфейса с данными приложения. Это делает взаимодействие с данными более гибким и динамичным.

C# - объектно-ориентированный язык программирования, созданный для разработки приложений на платформе Microsoft .NET Framework и .NET Core. Он имеет синтаксис, близкий к C++ и Java, обладает статической типизацией и поддерживает различные конструкции, такие как полиморфизм, делегаты, атрибуты, LINQ, исключения и другие [7].

Преимущества использования C#:

- C# поддерживает объектно-ориентированное программирование, что облегчает создание многофункциональных приложений с использованием классов и объектов.
- C# тесно интегрирован с продуктами Microsoft, что обеспечивает хорошую совместимость и поддержку со стороны различных продуктов компании.
- Размеры данных, такие как 32-битный int и 64-битный long, обеспечивают "мобильность" языка и упрощают программирование.
- Среда выполнения CLR автоматически управляет сборкой мусора, что освобождает разработчиков от необходимости явно освобождать память.

- C# обладает обширной библиотекой и множеством шаблонов, что упрощает разработку приложений за счет повторного использования кода.

Microsoft Visual Studio 2022 — это интегрированная среда разработки (IDE), предназначенная для создания различных типов приложений, включая те, которые используют технологии WPF, XAML, и C#. Вот несколько достоинств Visual Studio 2022:

Преимущества использования Visual Studio 2022:

- Visual Studio 2022 обеспечивает полную интеграцию с WPF и XAML, что делает разработку приложений с использованием этих технологий более удобной и эффективной. Визуальные редакторы, инструменты для работы с XAML, и отладчик помогают разработчикам создавать и отлаживать интерфейсы пользовательского интерфейса.
- Visual Studio 2022 обеспечивает мощную среду для разработки на языке C#. Она включает интеллектуальное автодополнение, отладчик, поддержку LINQ и другие функции, упрощающие процесс написания и отладки кода на C#.
- Visual Studio 2022 обеспечивает интегрированные инструменты для работы с .NET Framework и .NET Core, включая управление пакетами NuGet, создание проектов, и компиляцию. Это облегчает создание, сборку и управление .NET-приложениями.
- Visual Studio 2022 предоставляет мощные инструменты отладки, включая шаг за шагом выполнение кода, точки останова, анализ переменных и многое другое. Также предусмотрены средства профилирования для оптимизации производительности приложений.
- Visual Studio 2022 интегрирован с популярными системами контроля версий, такими как Git. Это обеспечивает командную работу над проектами, отслеживание изменений и управление версиями кода.
- Visual Studio 2022 позволяет использовать широкий спектр расширений и плагинов, что позволяет разработчикам индивидуализировать среду под свои потребности и интегрировать сторонние инструменты [8].

### 3.1 Характеристика выбранных программных сред и средств

Microsoft Visual Studio 2022 версия 17.1.0, C# версия 10, WPF (входит в состав .NET 6.0), XAML (входит в состав .NET 6.0).

# 4 АЛГОРИТМ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

## 4.1 Этапы реализации ПС (ПМ)

Разработка приложения включала в себя следующие этапы:

1. Продумывание общего функционала приложения, его цели, задачи, функции, поиск уже существующих разработок.

2. Разработка удобного и понятного интерфейса и навигации между окнами.

3. Написание общего кода программы и математических функций для калькулятора матриц и обычного калькулятора.

4. Тестирование приложения и параллельное добавление валидации и обработка ошибок ввода пользователя.

## 4.2 Пользовательский интерфейс ПС (ПМ)

### 4.2.1 Взаимодействие пользователей с ПС (ПМ)

При открытии приложения перед пользователем появляется калькулятор матриц, в котором расположено два поля для ввода матриц и одно поле для вывода результата. Пользователь вводит данные матрицы, выбирает операцию, которую он хочет совершить с введенной или введенными матрицами и нажимает на кнопку “Вычислить”, получая результат, если введенная матрица имеет корректный вид или предупреждение об ошибке, если матрица введена не корректно. Есть возможность создать матрицу из определенных элементов определенного размера при помощи функционального блока внизу окна. Переход на обычный калькулятор осуществляется с помощью кнопки с иконкой калькулятора, где перед пользователем появится окно с полем ввода и вывода, история выражений и результатов, которые пользователь вычислил и блок с кнопками для ввода чисел и математических операций над этими числами. Дополнительно присутствуют кнопки для удаления последнего

символа, очистки всего введенного выражения, копирование выбранного в списке выражения или его результата.

### 4.2.2 Проектирование пользовательских сценариев

Переход между калькуляторами осуществляется через кнопки Button с соответствующими иконками.

В окне с обычным калькулятором пользователь может ввести своё выражение как через отдельный блок элементов, состоящий из Button, либо введя с клавиатуры в левый TextBox. Результат после вычисления получается в правом TextBox, изменение содержимого которого запрещено программно. Полученные выражения с результатами отображаются в ListView, состоящем из колонок “Выражение” и “Результат”. Скопировать выражение или результат определенной записи можно, выбрав кнопкой мыши строку нужного вычисления, нажав на соответствующую кнопку Button под элементом ListView “Скопировать Выражение” или “Скопировать Результат”.

В окне с калькулятором матриц ввод матриц происходит в левый и центральный блок TextBox, правый TextBox невозможно изменить пользователю, так как он будет содержать в себе результат вычислений. Выбор операции происходит при помощи элемента ComboBox, содержащим в себе выпадающий список действий с матрицей или матрицами. Заполенние поля для ввода матриц заранее выбранным элементов с заранее выбранным размером матрицы осуществляется при помощи 3 TextBox и одной кнопки Button. Вычисление матрицы производится после нажатия на кнопку Button “Вычислить”, копирование результата матрицы производится при нажатии на кнопку Button “Скопировать матрицу”. О состоянии и правильности введенной матрицы сообщает обновляющийся в режиме реального времени элемент Label, расположенный под соответствующими TextBox.

### 4.2.3 Определение операций пользователей

Пользователь может перемещаться между двумя калькуляторами при помощи кнопок с соответствующими иконками, при использовании обычного калькулятора пользователь может нажимать на кнопки с числами и операторами, вводя их в поле для вычисления, также пользователь может очищать последний введенный элемент либо все введенное выражение и получить результат введенного выражения в списке, откуда может скопировать результат или выражение выбранной записи, нажав на соответствующие кнопки. Перейдя на окно с матрицами, пользователь способен выбрать операцию, производимую над матрицей или матрицами, создать заготовленную матрицу по нужным параметрам, выполнить операцию над матрицами нажав на кнопку “Вычислить”, и скопировать получившуюся матрицу после вычисления, нажав на соответствующую кнопку.

### 4.2.4 Составление функциональных блоков

Программа содержит в себе функциональные блоки, которые обладают следующими функциями:

CalculatorModel.cs – основной класс обычного калькулятора:

1. GetNumberFromString(string str, ref int ind):

- Извлекает число из строки, начиная с указанной позиции.
- Использует цикл, чтобы собрать последовательность цифр, включая десятичные точки.
- Возвращает извлеченное число в формате `double`.

2. evaluate(double b, double a, char oper):

- Выполняет бинарную операцию между двумя числами (`a` и `b`) в соответствии с заданным оператором.
- Поддерживает операции сложения, вычитания, умножения, деления, возведения в степень и взятия остатка от деления.
- Возвращает результат операции.

3. evaluateFunc(double a, char oper):

• Выполняет математическую функцию с одним аргументом (`a`) в соответствии с заданным оператором.

• Поддерживает функции синуса, косинуса, тангенса, котангенса, квадратного корня и факториала.

4. Factorial(double f):

• Рекурсивно вычисляет факториал числа.

• Проверяет ограничения на входное число (должно быть от 0 до 777).

5. Calculate(string expression):

• Вычисляет значение математического выражения, представленного в виде строки.

• Использует стеки для хранения чисел и операторов, а также словарь для определения приоритетов операторов.

• Поддерживает числа, операторы и функции, а также константы π и e.

• Обрабатывает унарный минус, скобки и обеспечивает правильный порядок операций.

• Возвращает результат вычисления выражения в виде `double`.

• Выбрасывает исключения в случае ошибок, таких как несоответствие скобок или неверное выражение.

Matrix.cs – основной класс матричного калькулятора:

1. IsSquare:

• Свойство, возвращающее true, если матрица квадратная (количество строк равно количеству столбцов).

2. FromStr(string str):

• Статический метод, преобразующий строку в объект Matrix.

• Удаляет лишние пробелы и символ новой строки, разделяет строки и элементы матрицы, преобразует их в числа.

3. ValidateMatrixString(string str):

• Статический метод для валидации строки матрицы.

• Проверяет, не является ли строка null или пустой. Затем удаляет лишние пробелы и символ новой строки.

• Разделяет строки и элементы матрицы, проверяя их количество и правильность формата (числа).

• Возвращает ValidationResult: Valid в случае успешной валидации, Error в случае ошибки, No_Matrix, если строка пуста.

4. CalculateDeterminant():

• Метод для вычисления определителя матрицы.

5. Size:

• Свойство, возвращающее кортеж с числом строк и столбцов матрицы.

6. Minor(int a, int b, double[,] arr):

• Вспомогательный метод для вычисления минора матрицы.

7. CalculateDeterminant(double[,] arr):

• Вспомогательный метод для рекурсивного вычисления определителя матрицы.

• Использует правило знакочередования и миноры.

8. ToString():

• Переопределенный метод для преобразования объекта Matrix в строку.

• Возвращает строковое представление матрицы с округлением чисел до 3 знаков после запятой.

9. operator +(Matrix m1, Matrix m2):

• Перегруженный оператор сложения матриц, который проверяет совпадение размеров матриц, затем складывает соответствующие элементы.

10. operator -(Matrix m1, Matrix m2):

- Перегруженный оператор вычитания матриц, который проверяет совпадение размеров матриц, затем вычитает соответствующие элементы.

11. operator *(Matrix m1, Matrix m2):

- Перегруженный оператор умножения матриц, который проверяет совпадение размеров матриц, затем выполняет умножение.

12. Transpose():

- Метод для транспонирования матрицы, который меняет местами строки и столбцы.

13. operator *(Matrix m1, double nubmer):

- Перегруженный оператор умножения матрицы на число, который умножает каждый элемент матрицы на заданное число.

14. Inverse():

- Метод для вычисления обратной матрицы, который проверяет вырожденность матрицы и возвращает новую матрицу - обратную исходной.

15. enum ValidationResult:

- Перечисление для возвращаемых значений метода ValidateMatrixString. Содержит значения Valid (валидация успешна), Error (ошибка валидации), No_Matrix (пустая строка).

CalculatorHistory.cs – класс, отвечающий за запись результатов вычислений в окне с обычным калькулятором.

Locator.cs – класс, отвечающий за сохранение данных, введенных пользователем при переходе из одного калькулятора в другой.

PropChange.cs – класс, предоставляющий механизм для уведомления об изменении свойств объекта. Этот класс является базовым классом для других классов моделей, которые хотят поддерживать уведомление об изменении свойств.

RelayCommand.cs – класс, который позволяет создавать экземпляры команд, которые могут быть привязаны к элементам управления в пользовательском интерфейсе (например, кнопкам) и автоматически

обновлять их доступность (активность) на основе результата выполнения метода canExecute.

### 4.2.5 Проектирование структуры экранов ПС (ПМ) и схемы навигации

В программе присутствует два рабочих окна, между которыми пользователь может перемещаться при помощи соответствующих теме экрана кнопок сбоку экрана приложения и одно общее окно, которое содержит в себе данные этих двух окон и кнопки для перемещения между ними. При нажатии на кнопку с матрицами, выделенное место для отрисовки окна калькулятора показывает рабочее окно калькулятора матриц с соответствующими элементами ввода, вывода, выбора операций и копирования результатов. При нажатии на кнопку с калькулятором, перед пользователем в выделенном месте экрана приложения отображается окно калькулятора с соответствующими элементами для ввода, вывода и хранения результатов вычислений. При переходе между окнами, введенные пользователем данные сохраняются через внутренний функционал программы. Основное окно программы поддерживает его растягивание вплоть до полноэкранного режима, все элементы при этом будут растянуты симметрично изначальному отображению.

### 4.2.6 Низкоуровневое проектирование

На этапе разработки приложения, была придумана следующая визуальная составляющая окон, отображенная на рисунке 2, которая далее была преобразована в более удобное и понятное представление, отображенное на рисунке 3, позволившее избавиться от одного окна и работать параллельно сразу в двух окнах благодаря сайд-бару, содержащем в себе стилизованные кнопки для перехода между калькуляторами.


Рисунок 2 – Набросок дизайна окон приложения

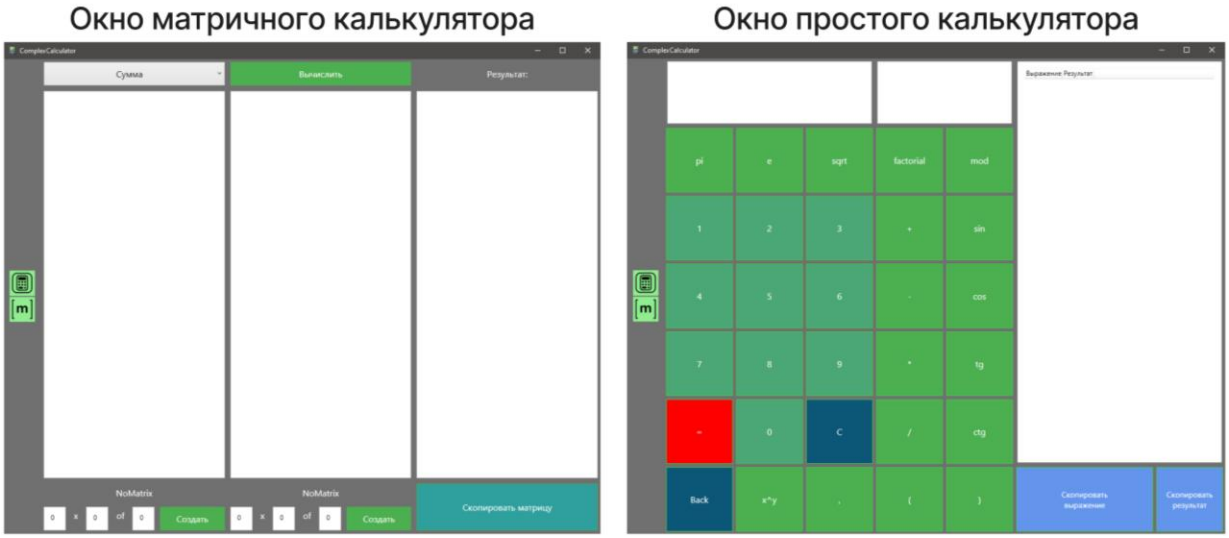


Рисунок 3 – Дизайн окон готового приложения

### 4.3 Входные, выходные и промежуточные данные

В качестве входных данных обычного калькулятора является строка, состоящая из операндов и операторов, которые впоследствии переводятся в нужные типы переменных и заполняют два разных стека – стек для чисел и

стек для операторов, и на выходе получается новая строка, состоящая из целого числа или числа с плавающей точкой – результат вычисления, введённого пользователем выражения.

В качестве входных данных матричного калькулятора является строка или строки, состоящие из целых чисел или чисел с плавающей точкой, отформатированные в формате алгебраической матрицы, то есть, элементы в строке разделены одним пробелом, а строки разделены между собой символом перевода строки. Далее эти данные переводятся в одномерный или двумерный массив чисел, над которым впоследствии происходят выбранные пользователем операции вычисления, на выходе получается новая матрица, формат которой идентичен формату входных данных (строка, разделенная пробелами и символами перевода строки). Если пользователь выбрал операцию вычисления определителя матрицы, то на выходе он получит строку, состоящую из одного числа.

### 4.4 Алгоритмы реализации используемых математических моделей

Примеры реализации математических алгоритмов вычисления, используемых в приложении, представлены следующим кодом:

Реализация вычисления определителя матрицы представлена на рисунке 4, сложение матриц на рисунке 5, транспонирование матрицы на рисунке 6.

```
private double CalculateDeterminant(double[,] arr)
{
    int n = arr.GetLength(0);
    double det = 0;
    if (n == 1) return arr[0, 0];
    else if (n == 2)
    {
        det = arr[0, 0] * arr[1, 1] - arr[0, 1] * arr[1, 0];
        return det;
    }
    else
    {
        for (int k = 0; k < n; k++) det += Math.Pow(-1, k) * arr[0, k] * CalculateDeterminant(Minor(0, k, arr));
        return det;
    }
}
```

Рисунок 4 – Реализация поиска определителя матрицы

```
public static Matrix operator +(Matrix m1, Matrix m2)
{
    if (m1.Size != m2.Size) return new(new double[0, 0]);
    double[,] arr = new double[m1.Size.rows, m1.Size.cols];
    for (int i = 0; i < m1.Size.rows; i++)
    {
        for (int j = 0; j < m2.Size.cols; j++)
        {
            arr[i, j] = m1.arr[i, j] + m2.arr[i, j];
        }
    }
    return new(arr);
}
```

Рисунок 5 – Реализация сложения двух матриц

```
public Matrix Transpose()
{
    int rows = arr.GetLength(0);
    int cols = arr.GetLength(1);

    double[,] result = new double[cols, rows];

    for (int i = 0; i < cols; i++)
    {
        for (int j = 0; j < rows; j++)
        {
            result[i, j] = arr[j, i];
        }
    }

    return new Matrix(result);
}
```

Рисунок 6 – Реализация транспонирования матрицы

### 4.5 Алгоритмы использования применяемых программных технологий

При написании приложения была использована библиотека System.Math, которая предоставляет константы и статические методы для тригонометрических, логарифмических и иных общих математических функций [9].

Описание используемых встроенных функций библиотеки:

- Math.Cos(Double) - Возвращает косинус указанного угла.
- Math.Pow(Double, Double) - Возвращает указанное число, возведенное в указанную степень.
- Math.Round(Double) - Округляет значение с плавающей запятой двойной точности до ближайшего целого значения; значения посередине округляются до ближайшего четного числа.
- Math.Sin(Double) - Возвращает синус указанного угла.
- Math.Sqrt(Double) - Возвращает квадратный корень из указанного числа.
- Math.Tan(Double) - Возвращает тангенс указанного угла.

### 4.6 Архитектура и схема функционирования ПС (ПМ)

Приложение реализовано на основе паттерна MVVM. Паттерн MVVM (Model-View-ViewModel) позволяет отделить логику приложения от визуальной части (представления). Данный паттерн является архитектурным, то есть он задает общую архитектуру приложения. MVVM состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View). Модель реализует интерфейс INotifyPropertyChanged который позволяет уведомлять систему об изменениях свойств модели. Благодаря этому облегчается привязка к представлению, хотя опять же прямое взаимодействие между моделью и представлением отсутствует. Для взаимодействия пользователя и приложения в MVVM используются команды. В WPF команды представлены интерфейсом ICommand. [10]

Диаграмма классов приложения представлена на диаграмме 1.

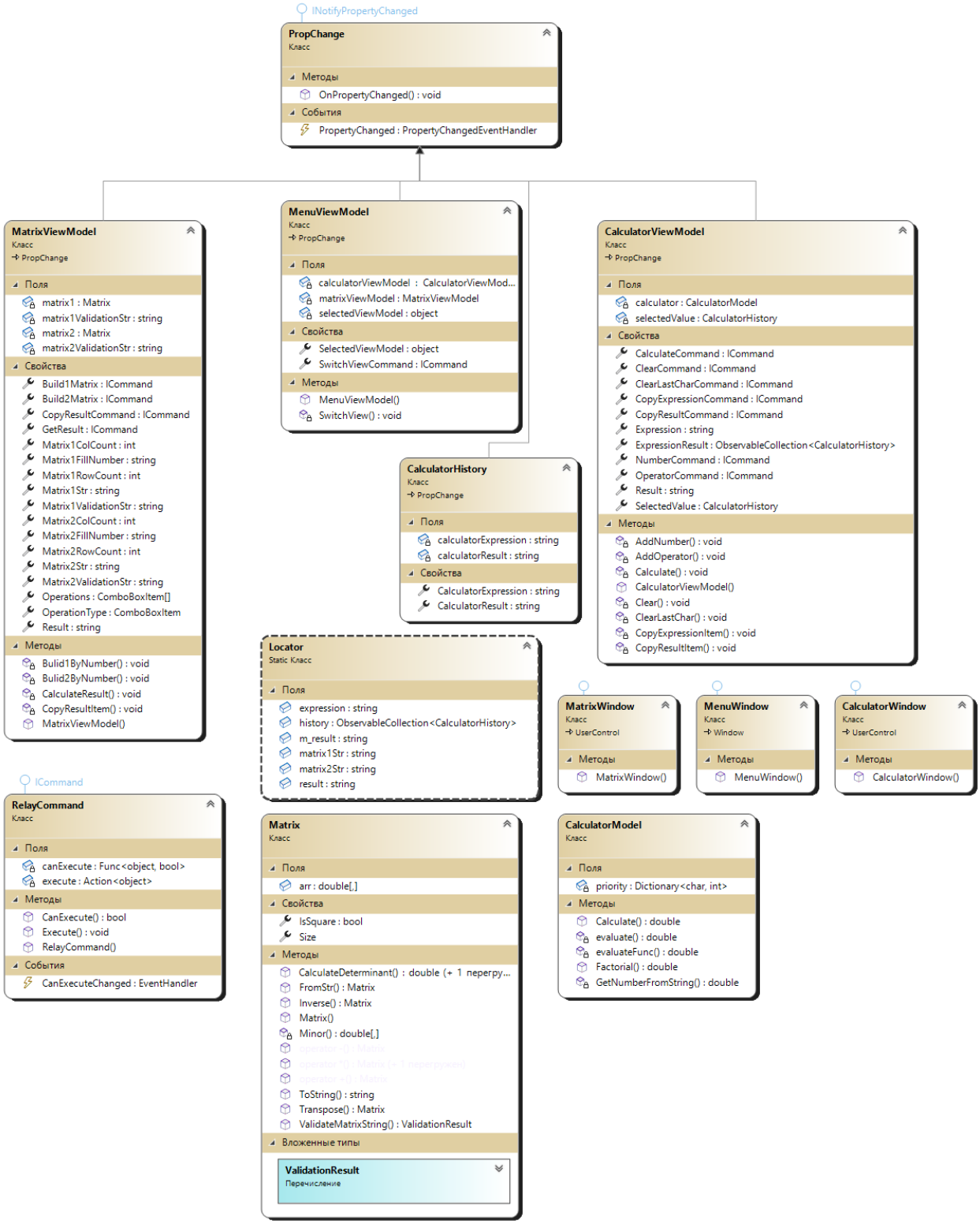


Диаграмма 1 – Диаграмма классов приложения

Класс PropChange представляет собой базовый класс, реализующий интерфейс INotifyPropertyChanged, который используется для поддержки уведомлений об изменении свойств объекта. Классы MatrixViewModel,

MenuViewModel, CalculatorViewModel, CalculatorHistrory наследуются от класса PropChange.

Класс RelayCommand представляет собой реализацию интерфейса ICommand и используется для связывания логики выполнения команды с элементами управления в пользовательском интерфейсе. Команды, реализованные с помощью интерфейса ICommand находятся в классах MatrixViewModel, CalculatorViewModel.

Также на диаграмме можно увидеть статический класс Locator, предназначенный для сохранения результатов вычислений в каждом калькуляторе, классы Matrix и CalculatorModel, содержащие в себе лоигку каждого калькулятора и классы MatrixWindow, MenuWindow, CalculatorWindow, которые отвечают за инициализацию объектов пользовательского интерфейса XAML окон [11].

# 5 ТЕСТИРОВАНИЕ И ОПТИМИЗАЦИЯ

## 5.1 План тестирования

Тестирование приложения заключалось в том, чтобы найти такие входные данные, при которых приложение уходило в бесконечный цикл или прекращало свою работу вследствие необработанной ошибки. Так же был произведен тест растягивания окна для того, чтобы посмотреть, как будут растягиваться элементы интерфейса при изменении размеров главного окна приложения.

## 5.2 Результаты тестирования

Были найдены следующие недочеты и способы их решения:

1. Проблема: Высчитывание факториала большого числа, из-за которого приложение переставало отвечать.

- Решение: Ограничение диапазона чисел для вычисления факториала.

2. Проблема: Высчитывание тригонометрических функций приводило не к целому числу, а к “почти целому” числу, всё из-за особенностей типов с плавающей точкой языка C#.

- Решение: Округление и отбрасывание слишком маленькой дробной части.

3. Проблема: неправильный ввод матрицы пользователем, после которого программа либо не могла вычислить результат, либо уходила в бесконечный цикл.

- Решение: создание валидации матриц, ограничение и обработка вводимых значений до максимально допустимых.

4. Проблем с растягиванием окна приложения не нашлось, все элементы растягиваются и отображаются корректно.

# 6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

При открытии приложение перед пользователем открывается окно приложения, на котором изначально показан калькулятор матриц, содержащий в себе поля для ввода и вывода данных, выбора операции (Рисунок 4).

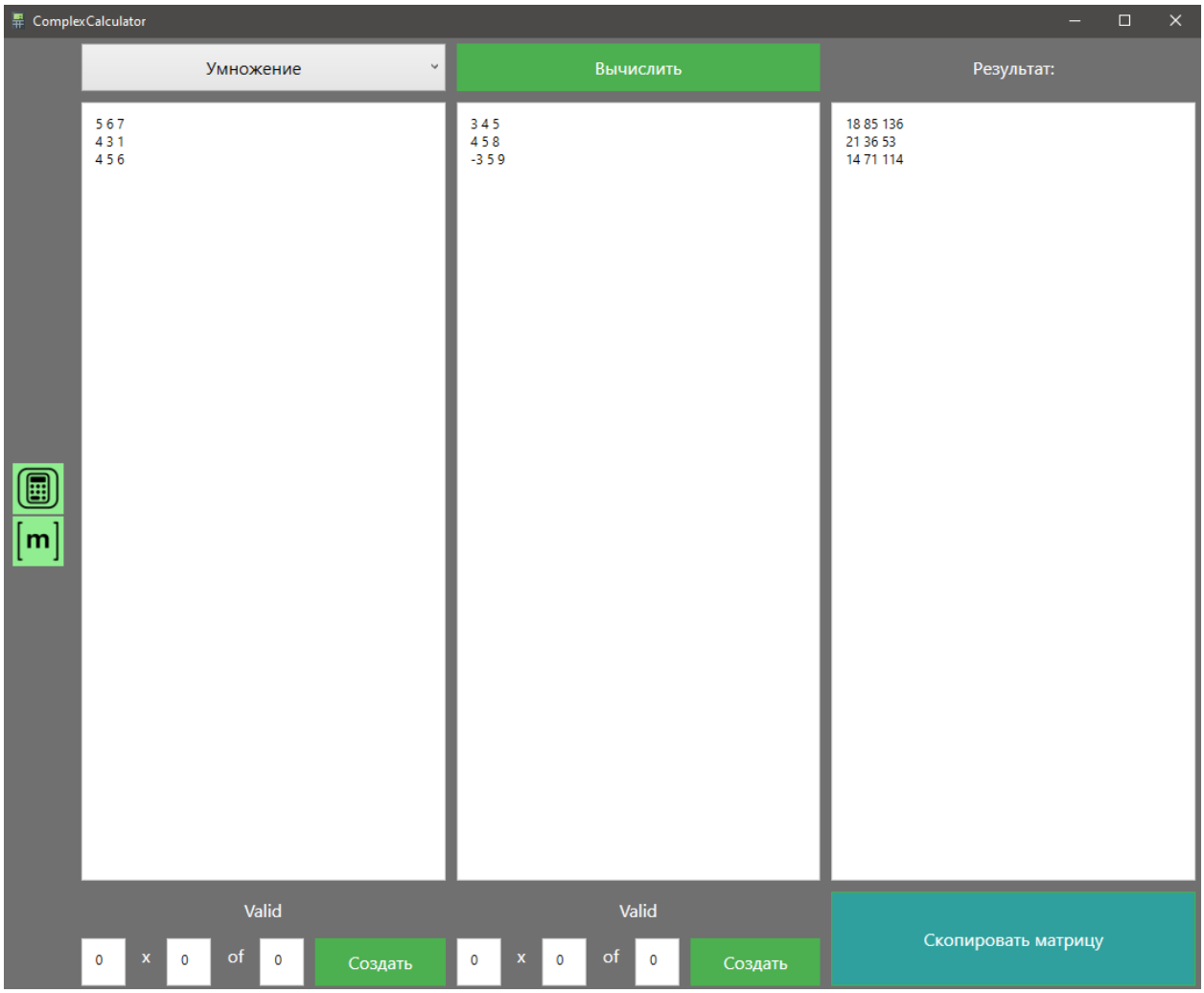


Рисунок 4 – Окно калькулятора матриц

При нажатии на выпадающий список, можно увидеть и выбрать нужную операцию, которую необходимо произвести с матрицей или матрицами (Рисунок 5).

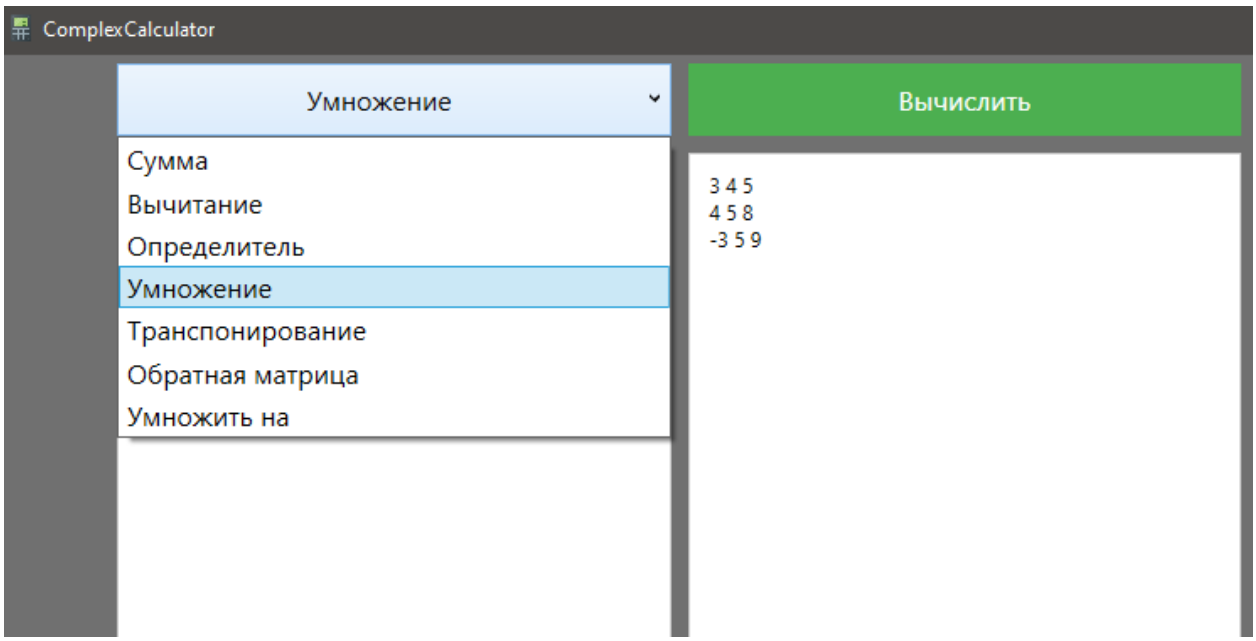


Рисунок 5 – Выбор операции над матрицами

При помощи полей снизу возможно заполнить матрицу необходимым размером и элементами, которые выберет пользователь (Рисунок 6).

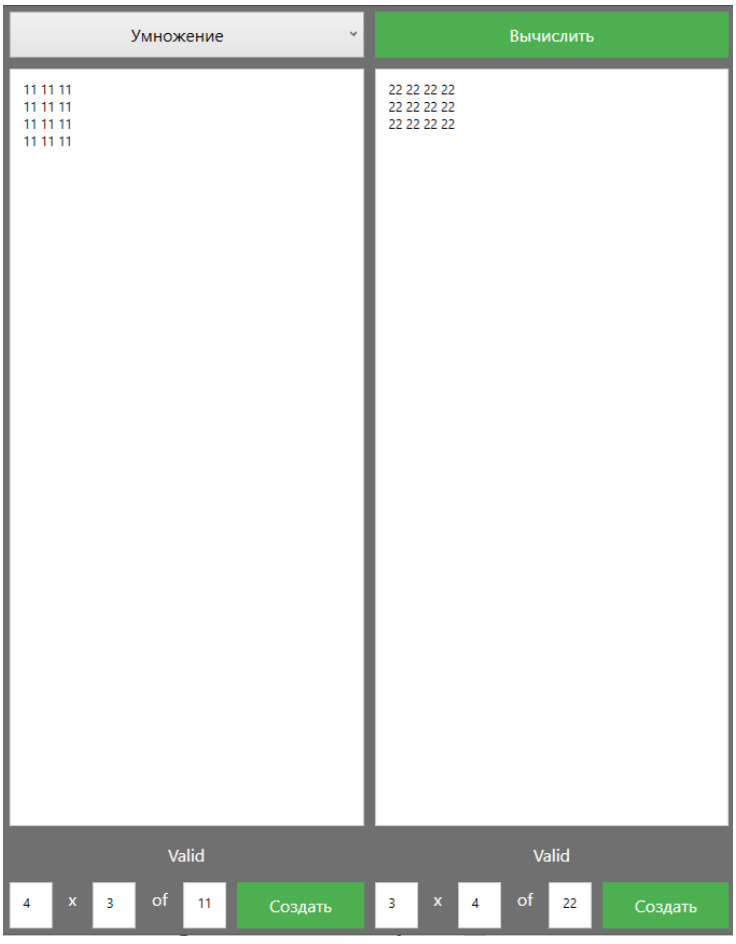


Рисунок 6 – Заполнение полей матрицы произвольными данными

При нажатии в боковом меню на кнопку с калькулятором, откроется новое окно обычного калькулятора, ввод выражения можно осуществлять как через кнопки на форме, так и через клавиатуру (Рисунок 7).


Рисунок 7 – Окно обычного калькулятора

Когда пользователь ввёл свое выражение и нажал на красную кнопку со знаком “Равно”, то результат введенного выражения отобразится в соседнем поле и само выражение и его результат запишется в список справа, откуда, выбрав нужную запись, можно скопировать выражение и его результат (Рисунок 8).


Рисунок 8 – Запись выражений пользователя в список

# ЗАКЛЮЧЕНИЕ

В рамках данной работы был спроектирован и реализован комплексный калькулятор, состоящий из простого калькулятора и калькулятора матриц. Приложение реализовано средствами платформы WPF (Windows Presentation Foundation), языка разметки XAML и языка программирования C# на основе паттерна MVVM. Для достижения этой цели были выполнены следующие задачи:

1. Был проведен анализ предметной области.
2. Было произведено проектирование, позволившее определить окончательную структуру приложения.
3. Была собрана необходимая математическая теория для реализации приложения.
4. Был спроектирован простой и удобный пользовательский интерфейс.
5. Было проведено функциональное тестирование и обработка ошибок на этапе контроля качества программного обеспечения.

В результате курсовой работы было создано удобное приложение для пользователей, которым необходимо проводить расчеты матриц и алгебраических выражений.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1 Линейная алгебра и матрицы [Электронный ресурс] – URL: https://mipt.ru/education/chair/mathematics/study/uchebniki/Ершов_Л_ЛА_2022(5).pdf (Дата обращ. 15.12.23).

2 Матрицы и операции над ними [Электронный ресурс] – URL: https://www.e-biblio.ru/book/bib/02_estestv_nauki/algebra_i_teoriya_chisel/alferova_balukevich_praktikum/docs/piece002.htm (Дата обращ. 15.12.23).

3 Определитель квадратной матрицы [Электронный ресурс] – URL: https://prog-cpp.ru/matrix-determinant/ (Дата обращ. 16.12.23).

4 Диаграмма прецедентов [Электронный ресурс] – URL: https://ciu.nstu.ru/kaf/persons/1914/page47048/diagramm_precedentov (Дата обращ. 15.12.23).

5 Определение WPF [Электронный ресурс] – URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-8.0 (Дата обращ. 16.12.23).

6 Определение XAML [Электронный ресурс] – URL: https://learn.microsoft.com/ru-ru/xamarin/xamarin-forms/xaml/xaml-basics/ (Дата обращ. 16.12.23).

7 Определение C# [Электронный ресурс] – URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/fundamentals/tutorials/oop (Дата обращ. 16.12.23).

8 Определение Visual Studio [Электронный ресурс] – URL: https://learn.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2022 (Дата обращ. 16.12.23).

9 Модуль System.Math [Электронный ресурс] – URL: https://learn.microsoft.com/ru-ru/dotnet/api/system.math?view=net-8.0 (Дата обращ. 16.12.23).

10 Определение MVVM [Электронный ресурс] – URL: https://metanit.com/sharp/wpf/22.1.php (Дата обращ. 15.12.23).

11 Определение ICommand [Электронный ресурс] – URL: https://metanit.com/sharp/wpf/22.3.php (Дата обращ. 15.12.23).

12 Исходный код приложения «Калькулятор Матриц» [Электронный ресурс] – URL: https://github.com/staizy/ComplexCalculator

# ПРИЛОЖЕНИЯ

## Приложение А

### Окно MenuView.xaml:

```
<Window x:Class="ComplexCalculator.MenuWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:i="http://schemas.microsoft.com/expression/2010/interactivity"
        xmlns:local="clr-namespace:ComplexCalculator"
        mc:Ignorable="d"
        Title="ComplexCalculator" MinHeight="900" MinWidth="1100" Height="900"
Width="1100" Icon="/Resources/icon.png"
        Background="#707070">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="15*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <ContentControl Grid.RowSpan="7" Grid.ColumnSpan="8" Grid.Row="0"
Grid.Column="1" Content="{Binding SelectedViewModel}"/>
        <StackPanel VerticalAlignment="Center" Orientation="Vertical"
Grid.Column="0" Grid.Row="1" Margin="10px">
            <Button Command="{Binding SwitchViewCommand}"
CommandParameter="Calculator" Background="LightGreen">
                <Image Source="/Resources/calculator.png"/>
            </Button>
            <Button Grid.Column="1" Command="{Binding SwitchViewCommand}"
CommandParameter="Matrix" Background="LightGreen">
                <Image Source="/Resources/matrix.png"/>
            </Button>
        </StackPanel>
    </Grid>
</Window>
```

### Окно MatrixView.xaml:

```
<UserControl x:Class="ComplexCalculator.MatrixWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:ComplexCalculator"
        mc:Ignorable="d"
        MinHeight="800" MinWidth="900"
        FontSize="16">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="auto"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
```

```
        </Grid.ColumnDefinitions>
        <Label Foreground="white" HorizontalAlignment="Center"
VerticalAlignment="Center" Content="Результат:" Grid.Column="2"/>
        <TextBox AcceptsReturn="True" Grid.Row="1" Text="{Binding Matrix1Str,
UpdateSourceTrigger=PropertyChanged}"/>
        <TextBox AcceptsReturn="True" Grid.Row="1" Grid.Column="1" Text="{Binding
Matrix2Str, UpdateSourceTrigger=PropertyChanged}"/>
        <TextBox Grid.Row="1" Grid.Column="2" IsReadOnly="True" Text="{Binding
Result}"/>
        <DockPanel Grid.Row="2" >
            <Label VerticalContentAlignment="Center"
HorizontalContentAlignment="Center"
                   Foreground="white" Content="{Binding Matrix1ValidationStr}"
DockPanel.Dock="Top"/>
            <TextBox MaxLength="2" MinWidth="40" Text="{Binding Matrix1RowCount}"/>
            <Label Foreground="white" Content="x"/>
            <TextBox MaxLength="2" MinWidth="40" Text="{Binding Matrix1ColCount}"/>
            <Label Foreground="white" Content="of"/>
            <TextBox MaxLength="5" MinWidth="40" Text="{Binding
Matrix1FillNumber}"/>
            <Button Content="Создать" Command="{Binding Build1Matrix}"/>
        </DockPanel>
        <DockPanel Grid.Row="2" Grid.Column="1">
            <Label VerticalContentAlignment="Center"
HorizontalContentAlignment="Center"
                   Foreground="white" Content="{Binding Matrix2ValidationStr}"
DockPanel.Dock="Top"/>
            <TextBox MaxLength="2" MinWidth="40" Text="{Binding Matrix2RowCount}"/>
            <Label Foreground="white" Content="x"/>
            <TextBox MaxLength="2" MinWidth="40" Text="{Binding Matrix2ColCount}"/>
            <Label Foreground="white" Content="of"/>
            <TextBox MaxLength="5" MinWidth="40" Text="{Binding Matrix2FillNumber}"
/>
            <Button Content="Создать" Command="{Binding Build2Matrix}"/>
        </DockPanel>
        <Button Grid.Column="1" Content="Вычислить" Command="{Binding GetResult}"/>
        <Button Grid.Column="3" Grid.Row="3" Content="Скопировать матрицу"
                Command="{Binding CopyResultCommand}" Background="#FF2FA09D"/>
        <ComboBox HorizontalContentAlignment="Center"
VerticalContentAlignment="Center"
                  SelectedItem="{Binding OperationType}" ItemsSource="{Binding
Operations}"/>
    </Grid>
</UserControl>
```

## Окно CalculatorView.xaml:

```
<UserControl x:Class="ComplexCalculator.CalculatorWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:ComplexCalculator"
        mc:Ignorable="d"
        MinHeight="800" MinWidth="900">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
```

```
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <!--- Строка ввода -->
        <TextBox Grid.ColumnSpan="3" TextWrapping="Wrap" CharacterCasing="Lower"
Text="{Binding Expression}"/>
        <!--- Строка вывода -->
        <TextBox Grid.ColumnSpan="2" Grid.Column="3" TextWrapping="Wrap"
IsReadOnly="True" Text="{Binding Result}"/>
        <!--- Цифры -->
        <Button Background="#4CA777" Grid.Row="2" Content="1" Command="{Binding
NumberCommand}" CommandParameter="1"/>
        <Button Background="#4CA777" Grid.Row="2" Grid.Column="1" Content="2"
Command="{Binding NumberCommand}" CommandParameter="2"/>
        <Button Background="#4CA777" Grid.Row="2" Grid.Column="2" Content="3"
Command="{Binding NumberCommand}" CommandParameter="3"/>
        <Button Background="#4CA777" Grid.Row="3" Content="4" Command="{Binding
NumberCommand}" CommandParameter="4"/>
        <Button Background="#4CA777" Grid.Row="3" Grid.Column="1" Content="5"
Command="{Binding NumberCommand}" CommandParameter="5"/>
        <Button Background="#4CA777" Grid.Row="3" Grid.Column="2" Content="6"
Command="{Binding NumberCommand}" CommandParameter="6"/>
        <Button Background="#4CA777" Grid.Row="4" Content="7" Command="{Binding
NumberCommand}" CommandParameter="7"/>
        <Button Background="#4CA777" Grid.Row="4" Grid.Column="1" Content="8"
Command="{Binding NumberCommand}" CommandParameter="8"/>
        <Button Background="#4CA777" Grid.Row="4" Grid.Column="2" Content="9"
Command="{Binding NumberCommand}" CommandParameter="9"/>
        <Button Background="#4CA777" Grid.Row="5" Grid.Column="1" Content="0"
Command="{Binding NumberCommand}" CommandParameter="0"/>
        <!--- Операции -->
        <Button Grid.Row="2" Grid.Column="3" Content="+" Command="{Binding
OperatorCommand}" CommandParameter="+"/>
        <Button Grid.Row="3" Grid.Column="3" Content="-" Command="{Binding
OperatorCommand}" CommandParameter="-"/>
        <Button Grid.Row="4" Grid.Column="3" Content="*" Command="{Binding
OperatorCommand}" CommandParameter="*"/>
        <Button Grid.Row="5" Grid.Column="3" Content="/" Command="{Binding
OperatorCommand}" CommandParameter="/"/>
        <Button Grid.Row="2" Grid.Column="4" Content="sin" Command="{Binding
OperatorCommand}" CommandParameter="sin("/>
        <Button Grid.Row="3" Grid.Column="4" Content="cos" Command="{Binding
OperatorCommand}" CommandParameter="cos("/>
        <Button Grid.Row="4" Grid.Column="4" Content="tg" Command="{Binding
OperatorCommand}" CommandParameter="tan("/>
        <Button Grid.Row="5" Grid.Column="4" Content="ctg" Command="{Binding
OperatorCommand}" CommandParameter="ctg("/>
        <Button Grid.Row="6" Grid.Column="3" Content="(" Command="{Binding
OperatorCommand}" CommandParameter="("/>
        <Button Grid.Row="6" Grid.Column="4" Content=")" Command="{Binding
OperatorCommand}" CommandParameter=")"/>
        <Button Grid.Row="6" Grid.Column="2" Content="," Command="{Binding
OperatorCommand}" CommandParameter=","/>
        <Button Grid.Row="6" Grid.Column="1" Content="x^y" Command="{Binding
OperatorCommand}" CommandParameter="^"/>
        <Button Grid.Row="1" Grid.Column="0" Content="pi" Command="{Binding
OperatorCommand}" CommandParameter="pi"/>
```

```
        <Button Grid.Row="1" Grid.Column="1" Content="e" Command="{Binding
OperatorCommand}" CommandParameter="e"/>
        <Button Grid.Row="1" Grid.Column="2" Content="sqrt" Command="{Binding
OperatorCommand}" CommandParameter="sqrt("/>
        <Button Grid.Row="1" Grid.Column="3" Content="factorial" Command="{Binding
OperatorCommand}" CommandParameter="!"/>
        <Button Grid.Row="1" Grid.Column="4" Content="mod" Command="{Binding
OperatorCommand}" CommandParameter="%"/>
        <!--- Команды -->
        <Button Background="#0C5777" Grid.Row="5" Grid.Column="2" Content="C"
Command="{Binding ClearCommand}"/>
        <Button Background="#0C5777" Grid.Row="6" Grid.Column="0" Content="Back"
Command="{Binding ClearLastCharCommand}"/>
        <Button Background="red" Grid.Row="5" Grid.Column="0" Content="="
Command="{Binding CalculateCommand}"/>
        <!--- Запись результатов -->
        <Button Background="CornflowerBlue" FontSize="14" Grid.Row="6"
Grid.Column="7" Command="{Binding CopyResultCommand}">
            <TextBlock
TextAlignment="Center">Скопировать<LineBreak/>результат</TextBlock>
        </Button>
        <Button Grid.ColumnSpan="2" Background="CornflowerBlue" FontSize="14"
Grid.Row="6" Grid.Column="5"
                Command="{Binding CopyExpressionCommand}">
            <TextBlock
TextAlignment="Center">Скопировать<LineBreak/>выражение</TextBlock>
        </Button>
        <ListView Grid.Column="5" Grid.ColumnSpan="3" Grid.RowSpan="6"
ItemsSource="{Binding ExpressionResult}"
                  SelectedItem="{Binding SelectedValue}">
            <ListView.View>
                <GridView>
                    <GridViewColumn Width="auto" Header="Выражение"
DisplayMemberBinding="{Binding CalculatorExpression}" />
                    <GridViewColumn Width="auto" Header="Результат"
DisplayMemberBinding="{Binding CalculatorResult}" />
                </GridView>
            </ListView.View>
        </ListView>
    </Grid>
</UserControl>
```

## Приложение Б

### MenuViewModel.cs:

```
using System.Windows.Input;
using ComplexCalculator.Model;

namespace ComplexCalculator
{
    public class MenuViewModel : PropChange
    {
        private CalculatorViewModel calculatorViewModel = new CalculatorViewModel();
        private MatrixViewModel matrixViewModel = new MatrixViewModel();
        private object selectedViewModel;

        public object SelectedViewModel
        {
            get { return selectedViewModel; }
            set
            {
                selectedViewModel = value;
```

```
                OnPropertyChanged(nameof(SelectedViewModel));
            }
        }

        public MenuViewModel()
        {
            SwitchViewCommand = new RelayCommand(SwitchView);
            SelectedViewModel = matrixViewModel;
        }

        public ICommand SwitchViewCommand { get; set; }

        private void SwitchView(object parameter)
        {
            if (parameter.ToString() == "Calculator") SelectedViewModel =
calculatorViewModel;
            else if (parameter.ToString() == "Matrix") SelectedViewModel =
matrixViewModel;
        }
    }
}
```

## CalculatorViewModel.cs:

```
using System;
using System.Collections.ObjectModel;
using System.Windows;
using System.Windows.Input;
using ComplexCalculator.Model;

namespace ComplexCalculator
{
    public class CalculatorViewModel : PropChange
    {
        private CalculatorModel calculator = new CalculatorModel();

        public string Expression
        {
            get { return Locator.expression; }
            set
            {
                Locator.expression = value;
                OnPropertyChanged("Expression");
            }
        }

        public string Result
        {
            get { return Locator.result; }
            set
            {
                Locator.result = value;
                OnPropertyChanged("Result");
            }
        }

        private CalculatorHistory selectedValue;
        public CalculatorHistory SelectedValue
        {
            get
            {
                return selectedValue;
            }
            set
            {
```

```
                selectedValue = value;
                OnPropertyChanged("SelectedValue");
                OnPropertyChanged("ExpressionResult");
            }
        }

        public ICommand NumberCommand { get; set; }
        public ICommand OperatorCommand { get; set; }
        public ICommand CalculateCommand { get; set; }
        public ICommand ClearCommand { get; set; }
        public ICommand ClearLastCharCommand { get; set; }
        public ICommand CopyResultCommand { get; set; }
        public ICommand CopyExpressionCommand  { get; set; }

        public CalculatorViewModel()
        {
            NumberCommand = new RelayCommand(AddNumber);
            OperatorCommand = new RelayCommand(AddOperator);
            CalculateCommand = new RelayCommand(Calculate);
            ClearCommand = new RelayCommand(Clear);
            ClearLastCharCommand = new RelayCommand(ClearLastChar);
            CopyResultCommand = new RelayCommand(CopyResultItem);
            CopyExpressionCommand = new RelayCommand(CopyExpressionItem);
        }

        private void CopyResultItem(object parameter)
        {
            if (SelectedValue != null)
Clipboard.SetText(SelectedValue.CalculatorResult);
            else MessageBox.Show("Вы не выбрали выражение из списка!");
        }

        private void CopyExpressionItem(object parameter)
        {
            if (SelectedValue != null)
Clipboard.SetText(SelectedValue.CalculatorExpression);
            else MessageBox.Show("Вы не выбрали выражение из списка!");
        }

        private void Clear(object parameter)
        {
            Expression = "";
        }

        private void ClearLastChar(object parameter)
        {
            if (Expression.Length > 0) Expression = Expression.Substring(0,
Expression.Length - 1);
        }

        private void AddNumber(object parameter)
        {
            string number = parameter as string;
            if (number != null)
            {
                Expression += number;
            }
        }

        private void AddOperator(object parameter)
        {
            string op = parameter as string;
            if (op != null)
            {
                Expression += op;
```

```
            }
        }

        public ObservableCollection<CalculatorHistory> ExpressionResult
        {
            get { return Locator.history; }
        }

        private void Calculate(object parameter)
        {
            try
            {
                Result = calculator.Calculate(Expression).ToString();
                Locator.history.Add(new CalculatorHistory { CalculatorExpression =
Expression, CalculatorResult = Result });
                OnPropertyChanged("ExpressionResult");
            }
            catch (Exception ex)
            {
                Result = "Error: " + ex.Message;
            }
        }
    }
}
```

## MatrixViewModel.cs:

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using ComplexCalculator.Model;

namespace ComplexCalculator
{
    public class MatrixViewModel : PropChange
    {
        private Matrix matrix1;
        private Matrix matrix2;
        private string matrix1ValidationStr;
        private string matrix2ValidationStr;
        public int Matrix1RowCount { get; set; }
        public int Matrix1ColCount { get; set; }
        public string Matrix1FillNumber { get; set; } = "0";
        public int Matrix2RowCount { get; set; }
        public int Matrix2ColCount { get; set; }
        public string Matrix2FillNumber { get; set; } = "0";
        public static ComboBoxItem[] Operations { get; set; } = new ComboBoxItem[] {
new ComboBoxItem { Content = "Сумма" }, new ComboBoxItem { Content = "Вычитание" },
                                                            new ComboBoxItem { Content
= "Определитель" },  new ComboBoxItem { Content = "Умножение" },
                                                           new ComboBoxItem { Content =
"Транспонирование" }, new ComboBoxItem { Content = "Обратная матрица" },
                                                           new ComboBoxItem { Content =
"Умножить на" } };
        public ComboBoxItem OperationType { get; set; } = Operations[0];


        public MatrixViewModel()
        {
            Build1Matrix = new RelayCommand(Bulid1ByNumber);
            Build2Matrix = new RelayCommand(Bulid2ByNumber);
            GetResult = new RelayCommand(CalculateResult);
            CopyResultCommand = new RelayCommand(CopyResultItem);
            Matrix1ValidationStr = "NoMatrix";
```

```
            Matrix2ValidationStr = "NoMatrix";
        }

        private void CopyResultItem(object parameter)
        {
            if (Locator.m_result != "")
Clipboard.SetText(Locator.m_result.ToString());
            else MessageBox.Show("Результирующая матрица пуста, копировать
нечего!");
        }

        public string Result
        {
            get { return Locator.m_result; }
            set
            {
                Locator.m_result = value;
                OnPropertyChanged(nameof(Result));
            }
        }

        public string Matrix1Str
        {
            get
            {
                var res = Matrix.ValidateMatrixString(Locator.matrix1Str);
                Matrix1ValidationStr = res.ToString();
                if (res == Matrix.ValidationResult.Valid) matrix1 =
Matrix.FromStr(Locator.matrix1Str);
                return Locator.matrix1Str;
            }
            set
            {
                Locator.matrix1Str = value.Replace("\r", "").Replace("  ", "
").Replace(" \n", "\n").Replace("\n\n", "\n");
                OnPropertyChanged(nameof(Matrix1Str));
                var res = Matrix.ValidateMatrixString(value);
                Matrix1ValidationStr = res.ToString();
                if (res == Matrix.ValidationResult.Valid) matrix1 =
Matrix.FromStr(value);
            }
        }
        public string Matrix2Str
        {
            get
            {
                var res = Matrix.ValidateMatrixString(Locator.matrix2Str);
                Matrix2ValidationStr = res.ToString();
                if (res == Matrix.ValidationResult.Valid) matrix2 =
Matrix.FromStr(Locator.matrix2Str);
                return Locator.matrix2Str;
            }
            set
            {
                Locator.matrix2Str = value.Replace("\r", "").Replace("  ", "
").Replace(" \n", "\n").Replace("\n\n", "\n");
                OnPropertyChanged(nameof(Matrix2Str));
                var res = Matrix.ValidateMatrixString(value);
                Matrix2ValidationStr = res.ToString();
                if (res == Matrix.ValidationResult.Valid) matrix2 =
Matrix.FromStr(value);
            }
        }

        public string Matrix1ValidationStr
```

```
{
    get { return matrix1ValidationStr; }
    set
    {
        matrix1ValidationStr = value;
        OnPropertyChanged(nameof(Matrix1ValidationStr));
    }
}
public string Matrix2ValidationStr
{
    get { return matrix2ValidationStr; }
    set
    {
        matrix2ValidationStr = value;
        OnPropertyChanged(nameof(Matrix2ValidationStr));
    }
}

public ICommand Build1Matrix { get; set; }
public ICommand Build2Matrix { get; set; }
public ICommand GetResult { get; set; }
public ICommand CopyResultCommand { get; set; }

private void Bulid1ByNumber(object parameter)
{
    string str = "";
    if (!double.TryParse(Matrix1FillNumber.ToString(), out _))
    {
        MessageBox.Show("Это не число");
    }
    else
    {
        for (int i = 0; i < Matrix1RowCount; i++)
        {
            for (int j = 0; j < Matrix1ColCount; j++)
            {
                str += Matrix1FillNumber.ToString() + " ";
            }
            str = str.Trim() + "\n";
        }
        Matrix1Str = str;
    }
}

private void Bulid2ByNumber(object parameter)
{
    string str = "";
    if (!double.TryParse(Matrix2FillNumber.ToString(), out _))
    {
        MessageBox.Show("Это не число");
    }
    else
    {
        for (int i = 0; i < Matrix2RowCount; i++)
        {
            for (int j = 0; j < Matrix2ColCount; j++)
            {
                str += Matrix2FillNumber.ToString() + " ";
            }
            str = str.Trim() + "\n";
        }
        Matrix2Str = str;
    }
}
```

```
        private void CalculateResult(object parameter)
        {
            if (Matrix1ValidationStr != "Valid") { MessageBox.Show("Матрица должна
иметь адекватный вид!"); return; }
            Matrix1Str = Matrix1Str.Trim();
            switch (OperationType.Content)
            {
                case "Определитель":
                    {
                        if (matrix1 == null) MessageBox.Show("Для вычисления
определителя матрицы она должна быть не пустой");
                        else
                        {
                            if (!matrix1.IsSquare) MessageBox.Show("Для вычисления
определителя матрицы она должна быть квадратной");
                            else if (matrix1.Size.rows > 10 || matrix1.Size.cols >
10) MessageBox.Show("Для вычисления определителя матрицы она должна быть размера не
более 10 на 10 элементов.");
                            else
                            {
                                Result = matrix1.CalculateDeterminant().ToString();
                            }
                        }
                    }
                    break;
                case "Сумма":
                    {
                        if (matrix1 == null || matrix2 == null) MessageBox.Show("Для
вычисления определителя матрицы она должна быть не пустой");
                        else if (matrix1.Size != matrix2.Size)
MessageBox.Show("Матрицы должны быть одного размера!");
                        else Result = (matrix1 + matrix2).ToString();
                    }
                    break;
                case "Вычитание":
                    {
                        if (matrix1 == null || matrix2 == null) MessageBox.Show("Для
вычисления определителя матрицы она должна быть не пустой");
                        else if (matrix1.Size != matrix2.Size)
MessageBox.Show("Матрицы должны быть одного размера!");
                        else Result = (matrix1 - matrix2).ToString();
                    }
                    break;
                case "Умножение":
                    {
                        if (matrix1 == null || matrix2 == null) MessageBox.Show("Для
вычисления определителя матрицы она должна быть не пустой");
                        else if (matrix1.Size.cols != matrix2.Size.rows)
MessageBox.Show("Количество строк одной матрицы должно соответствовать количеству
столбцов другой матрицы!");
                        else Result = (matrix1 * matrix2).ToString();
                    }
                    break;
                case "Транспонирование":
                    {
                        if (matrix1 == null) MessageBox.Show("Для вычисления
определителя матрицы она должна быть не пустой");
                        else Result = matrix1.Transpose().ToString();
                    }
                    break;
                case "Умножить на":
                    {
                        if (matrix1 == null || matrix2 == null) MessageBox.Show("Для
вычисления определителя матрицы она должна быть не пустой");
```

```
                        else if (matrix2.Size != (1, 1)) MessageBox.Show("Во втором
окне должно быть одно число!");
                        else Result = (matrix1 * matrix2.arr[0, 0]).ToString();
                    }
                    break;
                case "Обратная матрица":
                    {
                        if (matrix1 == null) MessageBox.Show("Для вычисления
определителя матрицы она должна быть не пустой");
                        else if (matrix1.Size.rows > 10 || matrix1.Size.cols > 10)
MessageBox.Show("Для вычисления обратной матрицы она должна быть размера не более 10
на 10 элементов.");
                        else
                        {
                            try
                            {
                                Result = matrix1.Inverse().ToString();
                            }
                            catch (Exception ex)
                            {
                                MessageBox.Show(ex.Message);
                            }
                        }
                    }
                    break;
            }
        }
    }
}
```

## Приложение В

## CalculatorModel.cs:

```
using System;
using System.Collections.Generic;

namespace ComplexCalculator.Model
{
    public class CalculatorModel
    {
        private Dictionary<char, int> priority = new Dictionary<char, int>()
        {
            {'(', 0},
            {')', 0},
            {'+', 1},
            {'-', 1},
            {'*', 2},
            {'/', 2},
            {'s', 3},
            {'c', 3},
            {'t', 3},
            {'g', 3},
            {'~', 3},
            {'^', 3},
            {'q', 3},
            {'!', 3},
            {'%', 2}
        };

        private double GetNumberFromString(string str, ref int ind)
```

```
        {
            string num = "";
            bool foundDecimalPoint = false;

            while (ind < str.Length)
            {
                char c = str[ind];
                if (Char.IsDigit(c) || c == '.' || (c == ',' && !foundDecimalPoint))
                {
                    num += c;
                    if (c == '.' || c == ',')
                    {
                        foundDecimalPoint = true;
                    }
                    ind++;
                }
                else
                {
                    ind--;
                    break;
                }
            }
            return Double.Parse(num);
        }

        private double evaluate(double b, double a, char oper)
        {
            switch (oper)
            {
                case '+': return a + b;
                case '-': return a - b;
                case '*': return a * b;
                case '/': return a / b;
                case '^': return Math.Pow(a,b);
                case '%': return a % b;
                default:
                    throw new Exception("Неправильный оператор " + oper);
            }
        }

        private double evaluateFunc(double a, char oper)
        {
            switch (oper)
            {
                case 's': return Math.Round(Math.Sin(a), 8);
                case 'c': return Math.Round(Math.Cos(a), 8);
                case 't': return Math.Round(Math.Tan(a), 8);
                case 'g': return Math.Round((1.0 / Math.Tan(a)), 8);
                case 'q': return Math.Sqrt(a);
                case '!': return Factorial(a);
                default:
                    throw new Exception("Неправильный оператор " + oper);
            }
        }

        public double Factorial(double f)
        {
            if (f > 777 || f < 0) throw new ArgumentException("Невозможно посчитать
факториал этого числа.");
            if (f == 0) return 1;
            else return f * Factorial(f - 1);
        }

        public double Calculate(string expression)
        {
```

```
            Stack<double> numbers = new Stack<double>();
            Stack<char> operators = new Stack<char>();
            for (int pos = 0; pos < expression.Length; pos++)
            {
                char c = expression[pos];
                if (Char.IsDigit(c))
                {
                    numbers.Push(GetNumberFromString(expression, ref pos));
                }
                else if (c == 'p' && expression[++pos] == 'i')
                {
                    numbers.Push(3.141592653589793);
                }
                else if (c == 'e')
                {
                    numbers.Push(2.718281828459045);
                }
                else if (c == '(')
                {
                    operators.Push('(');
                }
                else if (c == ')')
                {
                    while (operators.Count > 0 && operators.Peek() != '(')
                    {
                        if (operators.Peek() == 's' || operators.Peek() == 'c' ||
operators.Peek() == 't' ||
                            operators.Peek() == '!' || operators.Peek() == 'g' ||
operators.Peek() == 'q')
                        {
                            numbers.Push(evaluateFunc(numbers.Pop(),
operators.Pop()));
                        }
                        else
                        {
                            numbers.Push(evaluate(numbers.Pop(), numbers.Pop(),
operators.Pop()));
                        }
                    }
                    if (operators.Count > 0 && operators.Peek() == '(')
                    {
                        operators.Pop();
                    }
                    else
                    {
                        throw new InvalidOperationException("Несоответсвие
операторов");
                    }
                }
                else if (c == '-' && (pos == 0 || expression[pos - 1] == '('))
                {
                    operators.Push('~');
                }
                else if ((c == 's' && expression[pos+1] == 'i' && expression[pos +
2] == 'n') ||
                         (c == 'c' && expression[pos+1] == 'o' && expression[pos +
2] == 's') ||
                         (c == 't' && expression[pos+1] == 'a' && expression[pos +
2] == 'n'))
                {
                    operators.Push(c);
                    pos += 2;
                }

                else if (c == '!')
```

```
                {
                    operators.Push('!');
                }

                else if ((c == 'c' && expression[pos+1] == 't' && expression[pos +
2] == 'g'))
                {
                    operators.Push(expression[pos+2]);
                    pos += 2;
                }

                else if ((c == 's' && expression[pos + 1] == 'q' && expression[pos +
2] == 'r' && expression[pos + 3] == 't'))
                {
                    operators.Push(expression[pos + 1]);
                    pos += 3;
                }

                else
                {
                    while (operators.Count > 0 && priority[operators.Peek()] >=
priority[c])
                    {
                        if (operators.Peek() == 's' || operators.Peek() == 'c' ||
operators.Peek() == 't' || operators.Peek() == 'g' ||
                            operators.Peek() == '!' || operators.Peek() == 'g' ||
operators.Peek() == 'q')
                        {
                            numbers.Push(evaluateFunc(numbers.Pop(),
operators.Pop()));
                        }
                        else
                        {
                            numbers.Push(evaluate(numbers.Pop(), numbers.Pop(),
operators.Pop()));
                        }
                    }
                    operators.Push(c);
                }
            }

            while (operators.Count > 0)
            {
                if (operators.Peek() == '(' || operators.Peek() == ')')
                {
                    throw new InvalidOperationException("Несоответствие скобок.");
                }
                if (operators.Peek() == 's' || operators.Peek() == 'c' ||
operators.Peek() == 't' || operators.Peek() == 'g' ||
                    operators.Peek() == '!' || operators.Peek() == 'g' ||
operators.Peek() == 'q')
                {
                    numbers.Push(evaluateFunc(numbers.Pop(), operators.Pop()));
                }
                else
                {
                    numbers.Push(evaluate(numbers.Pop(), numbers.Pop(),
operators.Pop()));
                }
            }

            if (numbers.Count == 1)
            {
                return numbers.Pop();
            }
```

```
            else
            {
                throw new InvalidOperationException("Ошибка в выражении.");
            }
        }
    }
    }
```

### Matrix.cs:

```
using System;

namespace ComplexCalculator.Model
{
    public class Matrix
    {
        public double[,] arr;
        public bool IsSquare
        {
            get { return arr.GetLength(0) == arr.GetLength(1); }
        }

        public Matrix(double[,] arr)
        {
            this.arr = arr;
        }

        public static Matrix FromStr(string str)
        {
            str = str.Trim();
            if (str[str.Length - 1] == '\n') str = str.Substring(0, str.Length - 1);
            string[] rows = str.Split('\n');
            int rowCount = rows.Length;
            int colCount = rows[0].Split(' ').Length;

            double[,] matrixArr = new double[rowCount, colCount];

            for (int i = 0; i < rowCount; i++)
            {
                string[] elements = rows[i].Split(' ');

                for (int j = 0; j < colCount; j++)
                {
                    double.TryParse(elements[j], out double element);
                    matrixArr[i, j] = element;
                }
            }
            return new Matrix(matrixArr);
        }

        public static ValidationResult ValidateMatrixString(string str)
        {
            if (string.IsNullOrEmpty(str)) return ValidationResult.No_Matrix;
            str = str.Trim();
            if (str[str.Length - 1] == '\n') str = str.Substring(0, str.Length - 1);

            string[] rows = str.Split('\n');
            int rowCount = rows.Length;
            int colCount = rows[0].Split(' ').Length;

            for (int i = 0; i < rowCount; i++)
            {
                string[] elements = rows[i].Split(' ');

                if (elements.Length != colCount) return ValidationResult.Error;
                foreach (string element in elements)
```

```
                {
                    if (!double.TryParse(element, out _)) return
ValidationResult.Error;
                }
            }
            return ValidationResult.Valid;
        }

        public double CalculateDeterminant()
        {
            return CalculateDeterminant(arr);
        }

        public (int rows, int cols) Size
        {
            get => (arr.GetLength(0), arr.GetLength(1));
        }

        private double[,] Minor(int a, int b, double[,] arr)
        {
            double[,] result = new double[arr.GetLength(0) - 1, arr.GetLength(1) -
1];

            int row = 0, col = 0;
            for (int i = 0; i < arr.GetLength(0); i++)
            {
                if (i == a) continue;
                for (int j = 0; j < arr.GetLength(1); j++)
                {
                    if (j == b) continue;
                    result[row, col] = arr[i, j];
                    col++;
                }
                row++;
                col = 0;
            }
            return result;
        }

        private double CalculateDeterminant(double[,] arr)
        {
            int n = arr.GetLength(0);
            double det = 0;
            if (n == 1) return arr[0, 0];
            else if (n == 2)
            {
                det = arr[0, 0] * arr[1, 1] - arr[0, 1] * arr[1, 0];
                return det;
            }
            else
            {
                for (int k = 0; k < n; k++) det += Math.Pow(-1, k) * arr[0, k] *
CalculateDeterminant(Minor(0, k, arr));
                return det;
            }
        }

        public override string ToString()
        {
            var str = "";
            for (int i = 0; i < arr.GetLength(0); i++)
            {
                for (int j = 0; j < arr.GetLength(1); j++)
                {
                    str += arr[i, j].ToString("0.###") + " ";
```

```
        }
        str = str.Trim() + "\n";
    }
    return str;
}

public static Matrix operator +(Matrix m1, Matrix m2)
{
    if (m1.Size != m2.Size) return new(new double[0, 0]);
    double[,] arr = new double[m1.Size.rows, m1.Size.cols];
    for (int i = 0; i < m1.Size.rows; i++)
    {
        for (int j = 0; j < m2.Size.cols; j++)
        {
            arr[i, j] = m1.arr[i, j] + m2.arr[i, j];
        }
    }
    return new(arr);
}

public static Matrix operator -(Matrix m1, Matrix m2)
{
    if (m1.Size != m2.Size) return new(new double[0, 0]);
    double[,] arr = new double[m1.Size.rows, m1.Size.cols];
    for (int i = 0; i < m1.Size.rows; i++)
    {
        for (int j = 0; j < m2.Size.cols; j++)
        {
            arr[i, j] = m1.arr[i, j] - m2.arr[i, j];
        }
    }
    return new(arr);
}

public static Matrix operator *(Matrix m1, Matrix m2)
{
    int rows = m1.Size.rows;
    int cols = m2.Size.cols;
    int commonDimension = m1.Size.cols;

    double[,] result = new double[rows, cols];

    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            double sum = 0;
            for (int k = 0; k < commonDimension; k++)
            {
                sum += m1.arr[i, k] * m2.arr[k, j];
            }
            result[i, j] = sum;
        }
    }

    return new Matrix(result);
}

public Matrix Transpose()
{
    int rows = arr.GetLength(0);
    int cols = arr.GetLength(1);

    double[,] result = new double[cols, rows];
```

```
            for (int i = 0; i < cols; i++)
            {
                for (int j = 0; j < rows; j++)
                {
                    result[i, j] = arr[j, i];
                }
            }

            return new Matrix(result);
        }

        public static Matrix operator *(Matrix m1, double nubmer)
        {
            int rows = m1.Size.rows;
            int cols = m1.Size.cols;

            double[,] result = new double[rows, cols];

            for (int i = 0; i < rows; i++)
            {
                for (int j = 0; j < cols; j++)
                {
                    result[i, j] = m1.arr[i, j] * nubmer;
                }
            }

            return new Matrix(result);
        }

        public Matrix Inverse()
        {
            int size = arr.GetLength(0);
            double determinant = CalculateDeterminant(arr);

            if (determinant == 0)
                throw new InvalidOperationException("Матрица вырожденная, обратной
матрицы не существует.");

            double[,] inverse = new double[size, size];

            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
                {
                    double minorDeterminant = CalculateDeterminant(Minor(i, j,
arr));
                    double cofactor = (i + j) % 2 == 0 ? minorDeterminant : -
minorDeterminant;
                    inverse[j, i] = cofactor / determinant;
                }
            }
            return new Matrix(inverse);
        }

        public enum ValidationResult
        {
            Valid,
            Error,
            No_Matrix
        }
    }
      }
```