**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное   
учреждение высшего образования   
«Санкт-Петербургский политехнический университет Петра Великого»  
ИНСТИТУТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

|  |  |
| --- | --- |
|  | **Допустить к защите** Заместитель директора  по УМР \_\_\_\_\_\_\_\_\_\_\_\_ Е.Г. Конакина\_  (Подпись) (ФИО)  «\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2023 г. |
|  |

**курсовой проект**Тема: «Программа вычисление определителя матрицы N \* N и обратной матрицы»

специальность 09.02.07 группа 32919/6

Студент (ка) \_\_\_\_\_\_\_\_\_\_\_\_  
 (подпись) (ФИО)  
Преподаватель Девятко Н.С.

(подпись) (ФИО)

Санкт-Петербург  
2023

Содержание

[ВВЕДЕНИЕ 3](#_Toc127174262)

[1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ 4](#_Toc127174263)

[1.1. Описание предметной области 4](#_Toc127174264)

[1.2. Анализ методов решения 6](#_Toc127174265)

[1.3. Обзор средств программирования 6](#_Toc127174266)

[1.4. Описание языка C# 7](#_Toc127174267)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 11](#_Toc127174268)

[2.1. Постановка задачи 11](#_Toc127174269)

[2.1.1. Основания для разработки 11](#_Toc127174270)

[2.1.2. Назначение программы 11](#_Toc127174271)

[2.2. Проектирование приложения 11](#_Toc127174272)

[2.2.1. Диаграмма прецедентов 11](#_Toc127174273)

[2.2.2. Диаграмма последовательностей 12](#_Toc127174274)

[2.2.3. Диаграмма классов 14](#_Toc127174275)

[2.2.4. Диаграмма активности 14](#_Toc127174276)

[2.3. Текст программы 16](#_Toc127174277)

[2.4. Описание программы 16](#_Toc127174278)

[2.4.1. Общие сведения 16](#_Toc127174279)

[2.4.2. Функциональное назначение 16](#_Toc127174280)

[2.4.3. Описание логической структуры системы 16](#_Toc127174281)

[2.4.4. Используемые технические и программные средства 18](#_Toc127174282)

[2.4.5. Вызов и загрузка 19](#_Toc127174283)

[2.5. Руководство оператора 19](#_Toc127174284)

[2.5.1. Назначение программы 19](#_Toc127174285)

[2.5.2. Выполнение программы и сообщения оператору 19](#_Toc127174286)

[2.6. Программа и методика испытаний 20](#_Toc127174287)

[2.6.1. Объект испытаний 20](#_Toc127174288)

[2.6.2. Цель испытаний 20](#_Toc127174289)

[2.6.3. Требования к программе 20](#_Toc127174290)

[2.6.4. Требования к программной документации 22](#_Toc127174291)

[2.6.5. Средства и порядок испытаний 22](#_Toc127174292)

[2.6.6. Методы испытаний 23](#_Toc127174293)

[2.7. Протокол испытаний 26](#_Toc127174294)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 29](#_Toc127174295)

[ПРИЛОЖЕНИЕ А 28](#_Toc127174296)

[ПРИЛОЖЕНИЕ Б 34](#_Toc127174300)

[ПРИЛОЖЕНИЕ В 38](#_Toc127174301)

# ВВЕДЕНИЕ

В современном мире математические вычисления и анализ данных являются неотъемлемой частью многих областей, таких как наука, финансы, инженерия и технологии. Вычисление определителя матрицы и обратной матрицы - это один из важных аспектов математических вычислений, который может быть применен в различных областях.

Разработанная в данном курсовом проекте программа предназначена для вычисления определителя и обратной матрицы размером N \* N. Она может быть использована как в учебных целях, так и в практических задачах, связанных с анализом данных и решением математических задач. Программа предоставляет быстрый и точный расчет определителя и обратной матрицы, что делает ее незаменимой для любого, кто работает с матрицами и математическими вычислениями.

# 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ

## 1.1. Описание предметной области

Предметная область данного курсового проекта - математические вычисления и анализ данных с использованием матриц. Задача заключается в разработке программы для вычисления определителя матрицы и обратной матрицы размером N \* N. Решением задачи является создание программного продукта, который позволит быстро и точно рассчитывать определитель и обратную матрицу.

Программа может быть применена в различных областях, таких как наука, финансы, инженерия и технологии. Она может использоваться в учебных целях для изучения матричных операций и решения математических задач, а также в практических задачах, связанных с анализом данных.

Преимущества использования данной программы заключаются в ее быстроте и точности расчетов, что позволяет сократить время на выполнение задач и повысить эффективность работы. Например, при работе с большими объемами данных, программа для вычисления определителя и обратной матрицы может значительно ускорить процесс анализа данных и сделать его более точным.

## 1.2. Анализ методов решения

1. Метод Гаусса-Жордана

Этот метод заключается в приведении матрицы к ступенчатому виду с помощью элементарных преобразований строк и столбцов, а затем обратным ходом вычисления элементов обратной матрицы. Шаги метода:

1. Привести матрицу к ступенчатому виду с помощью элементарных преобразований строк и столбцов.

2. Обратным ходом вычислить элементы обратной матрицы.

Пример:

Дана матрица A:

Шаг 1: Приводим матрицу к ступенчатому виду:

Шаг 2: Обратным ходом вычисляем элементы обратной матрицы:

Достоинства: метод прост в реализации и не требует большого объема памяти. Недостатки: метод может быть неэффективным для больших матриц, так как требует много операций.

2. Метод LU-разложения

Этот метод заключается в разложении матрицы на произведение нижней треугольной матрицы L и верхней треугольной матрицы U. Затем с помощью обратного хода вычисляются элементы обратной матрицы. Шаги метода:

1. Разложить матрицу на произведение нижней треугольной матрицы L и верхней треугольной матрицы U.

2. Решить систему уравнений Ly=b, где y=Ux.

3. Решить систему уравнений Ux=y.

Пример:

Дана матрица A:

Шаг 1: Разложим матрицу на произведение L и U:

L = ;

U= ;

Шаг 2: Решим систему уравнений Ly=b:

y1 = b1 = 1

y2 = b2 - 4y1 = -3

y3 = b3 - 7y1 - 2y2 = 2

Шаг 3: Решим систему уравнений Ux=y:

x3 = y3 / 0 = неопределено

x2 = (y2 + 6x3) / (-3) = -1

x1 = y1 - 2x2 - 3x3 = 0

Обратная матрица:

0 1/3 -2/9

-1/3 -2/9 5/18

2/3 1/9 -1/18

Достоинства: метод более эффективен, чем метод Гаусса-Жордана для больших матриц. Недостатки: метод требует большего объема памяти, чем метод Гаусса-Жордана. Выбранный метод: Метод LU-разложения.

Достоинства выбранного метода: более эффективен для больших матриц.

Недостатки выбранного метода: требует большего объема памяти.

Способы реализации выбранного метода:

1. Ручной способ - вычисление матрицы вручную с помощью калькулятора или на бумаге. 2. Автоматизированный способ - написание программы на языке программирования для вычисления матрицы.

3. Автоматический способ - использование готовых программных средств, таких как Excel или Matlab.

Достоинства и недостатки способов реализации:

1. Ручной способ: Достоинства: не требует дополнительных затрат на программирование или приобретение программных средств. Недостатки: может быть трудоемким и ошибочным при вычислении больших матриц.

2. Автоматизированный способ: Достоинства: более быстрый и точный, чем ручной способ. Недостатки: требует знаний языка программирования и времени на написание программы.

3. Автоматический способ: Достоинства: не требует знаний языка программирования или времени на написание программы. Недостатки: может быть ограничен функциональностью программных средств.

Выбранный способ: Автоматизированный способ.

Обоснование выбора: Автоматизированный способ более быстрый и точный, чем ручной способ, и не требует дополнительных затрат на приобретение программных средств, как в случае с автоматическим способом.Конец формы

## 1.3. Обзор средств программирования

Для решения задачи вычисления определителя матрицы и обратной матрицы можно использовать различные языки программирования и среды разработки. Одним из наиболее подходящих вариантов может быть язык программирования C# и среда разработки Visual Studio.

Программист для написания решения задачи вычисления определителя матрицы и обратной матрицы можно использовать различные языки программирования и среды разработки. Например, Python с библиотеками NumPy и SciPy предоставляют широкие возможности для работы с матрицами и выполнения вычислений. Однако, Python может быть менее производительным, чем C# или Java, и не иметь такой же удобной интегрированной среды разработки.

MATLAB является мощной средой для численных вычислений и имеет богатый набор инструментов для работы с матрицами. Однако, стоимость лицензии MATLAB может быть довольно высокой, и он может быть менее удобным для создания пользовательского интерфейса.

Java с использованием библиотеки Apache Commons Math также предоставляет возможности для работы с матрицами и выполнения вычислений. Однако, Java может быть менее удобным для создания пользовательского интерфейса, чем C# или Python.

При выборе языка программирования и среды разработки необходимо учитывать требования по функциональности, удобству использования, производительности и доступности. Кроме того, важно учитывать опыт и знания разработчиков, чтобы выбрать наиболее подходящий инструмент для решения задачи

Основными достоинствами C# являются его простота и удобство использования, а также возможность создания пользовательского интерфейса с помощью Windows Forms или WPF. Кроме того, C# имеет хорошую производительность и поддерживает многопоточность, что может ускорить выполнение вычислений.

Visual Studio предоставляет широкие возможности для разработки приложений на C#, включая интегрированную среду разработки, отладчик, компилятор и другие инструменты. Однако, стоимость лицензии Visual Studio может быть довольно высокой.

Другими вариантами языков программирования и сред разработки, которые можно использовать для решения задачи, могут быть Python с использованием библиотек NumPy и SciPy, MATLAB или Java с использованием библиотеки Apache Commons Math. Однако, каждый из этих вариантов имеет свои особенности и требует более детального анализа для выбора наиболее подходящего.

В целом, выбор языка программирования и среды разработки зависит от конкретных требований и условий проекта. Необходимо проанализировать все доступные варианты и выбрать наиболее подходящий, который будет соответствовать требованиям по функциональности, удобству использования и производительности.

Таким образом, средством программирования данного курсового проекта был выбран язык С#, среда разработки Visual Studio 2017.

## 1.4. Описание языка C#

C# (C Sharp) - это объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он используется для создания приложений для Windows, веб-приложений, игр и многих других целей. C# является частью платформы .NET и поддерживает множество функций, включая управляемый код, автоматическую сборку мусора и проверку типов во время выполнения.

C# имеет синтаксис, похожий на языки программирования Java и C++, но имеет более простую и понятную структуру. Он также поддерживает асинхронное программирование и функциональное программирование.

Источник: https://docs.microsoft.com/en-us/dotnet/csharp/ [5].

**Ключевые слова C#:**

|  |  |  |  |
| --- | --- | --- | --- |
| abstract | extern | null | struct |
| as | false | object | switch |
| base | finally | operator | this |
| bool | fixed | out | throw |
| break | float | override | true |
| byte | for | params | try |
| case | foreach | private | typeof |
| catch | goto | protected | uint |
| const | if | public | ulong |
| continue | implicit | readonly | unchecked |
| decimal | in | ref | unsafe |
| default | int | return | ushort |
| delegate | interface | sbyte | using |
| do | internal | sealed | virtual |
| double | is | short | void |
| else | lock | sizeof | while |
| enum | long | stackalloc |  |
| event | namespace | static |  |
| explicit | new | string |  |

**Контекстные ключевые слова C#:**

|  |  |  |  |
| --- | --- | --- | --- |
| add | equals | join | set |
| ascending | from | let | value |
| async | get | on | var |
| await | global | orderby | where |
| by | group | partial | yield |
| descending | in | remove |  |
| dynamic | into | select |  |

**Основные алгоритмические конструкции C#:**

|  |  |
| --- | --- |
| Оператор присваивания | string hello = "hello " + "world"; //результат равен "hello world"  int x1 = 2 + 4; // результат равен 6  int x2 = 10 - 6; //результат равен 4  int x3 = 10 \* 6; //результат равен 60  double x4 = 10.0 / 4.0; //результат равен 2.5  double x5 = 10.0 % 4.0; //результат равен 2  int y1 = 5;  int z1 = ++y1; // z1=6; y1=6  int y2 = 5;  int z2 = y2++; // z2=5; y2=6  int y3 = 5;  int z3 = --y3; // z3=4; y3=4  int y4 = 5;  int z4 = y4--; // z4=5; y4=4 |
| Условный оператор | *if (условие) {(действие) } else {(альтернатива)} ;*  int num1 = 8;  int num2 = 6;  if(num1 > num2)  {      Console.WriteLine("Число {0} больше числа {1}", num1, num2);  }  else  {      Console.WriteLine("Число {0} меньше числа {1}", num1, num2);  } |
| Оператор множественного ветвления | *Конструкция switch/case аналогична конструкции if/else, так как позволяет обработать сразу несколько условий:*  Console.WriteLine("Нажмите Y или N");  string selection = Console.ReadLine();  switch (selection)  {      case "Y":          Console.WriteLine("Вы нажали букву Y");          break;      case "N":          Console.WriteLine("Вы нажали букву N");          break;      default:          Console.WriteLine("Вы нажали неизвестную букву");          break;  }  *После ключевого слова switch в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после оператора сase. И если совпадение будет найдено, то будет выполняться определенный блок сase.*  *В конце блока сase ставится оператор break, чтобы избежать выполнения других блоков.*  *Если мы хотим также обработать ситуацию, когда совпадения не будет найдено, то можно добавить блок default,.* |
| Арифметический цикл  **(**применяется, когда известно количество повторений цикла) | for *([инициализация счетчика]; [условие]; [изменение счетчика])*  for (int i = 0; i < 9; i++)  {      Console.WriteLine("Квадрат числа {0} равен {1}", i, i \* i);  } |
| Цикл с предусловием  **(**применяется, когда неизвестно количество повторений цикла) | *while (условие) { тело цикла };*  Этот цикл будет выполняться до тех пор, пока истинно *условие* (логическое выражение, возвращающее значение типа **Boolean**). При этом если это выражение сразу равно **false**, *тело цикла* не будет выполнено ни разу. Нужно очень внимательно следить за написанием *условия* и контролем завершения цикла, так как в результате ошибки цикл **while** будет повторяться бесконечное количество раз, что приведёт к "зацикливанию" и "зависанию" программы. |
| Цикл с постусловием  **(**применяется, когда неизвестно количество повторений цикла) | *do { тело цикла } while (условие);*  Повторения сначала выполняет *тело цикла*, а затем уже проверяет выполнение *условия*: Таким образом, этот вариант цикла гарантирует, что *тело цикла* будет выполнен по крайней мере один раз. И будет выполняться до тех пор, пока *условие* не станет истинным (т.е. **true**). |

# 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1. Постановка задачи

Основание для разработки:

Заказчиком является учебное учереждение, занимающаяся обучением студентов программированию и требующая выполнить курсовую которая включает программу для вычисления определителя матрицы N \* N и обратной матрицы.

Назначение разработки:

Разработать приложение на языке C#, которое будет вычислять определитель матрицы N \* N и обратную матрицу. Приложение должно быть удобным в использовании и иметь интуитивно понятный интерфейс.

Требования к программе:

- Приложение должно быть написано на языке C# с использованием среды разработки Visual Studio.

- Приложение должно иметь возможность ввода матрицы N \* N с клавиатуры или из файла.

- Приложение должно вычислять определитель матрицы N \* N и обратную матрицу.

- Приложение должно выводить результаты вычислений на экран или сохранять в файл.

- Приложение должно иметь проверку на корректность введенных данных.

Требования к программной документации:

- Разработать техническое задание на приложение.

- Разработать руководство пользователя на приложение.

- Разработать документацию для программиста.

Стадии и сроки разработки:

1. Анализ требований - 1 неделя.

2. Проектирование приложения - 1 неделя.

3. Разработка приложения - 2 недели.

4. Тестирование и отладка - 1 неделя.

Итого: 5 недель.

Виды испытаний:

- Функциональное тестирование - проверка на соответствие требованиям.

### 2.1.1. Основания для разработки

Разработка ведётся на основании задания к курсовому проекту по профессиональному модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем» МДК 01.01 «Разработка программных модулей» и утверждена Институтом среднего профессионального образования.

### 2.1.2. Назначение программы

Программа для вычисления определителя матрицы N \* N и обратной матрицы, предназначена для решения математических задач и позволяет упростить процесс вычисления для пользователя.

Полный текст технического задания приведен в приложении А.

## 2.2. Проектирование приложения

На этапе проектирования были разработаны диаграмма прецедентов, диаграмма последовательностей, диаграмма классов и диаграмма активности.

### 2.2.1. Диаграмма прецедентов

Диаграммы прецедентов частично описывает use case – прецедент использования проектируемой системы, давая частичное описание частичного применения системы с точки зрения условного внешнего обозревателя. При этом описание фокусируется на том, что должна делать система по отношению к своему внешнему окружению (периферии), а не то на том, как она эта делает, то есть диаграмма прецедентов есть частичная спецификация (рисунок 1).



Рисунок 1 Диаграмма прецедентов

### 2.2.2. Диаграмма последовательностей

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования.

Все действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций. На диаграмме последовательности объект изображается в виде прямоугольника, от которого вниз проведена пунктирная вертикальная линия. Эта линия называется линией жизни (lifeline) объекта. Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия.



Рисунок 2 Диаграмма последовательностей

### 2.2.3. Диаграмма классов

На рисунке 3 показана диаграмма классов проекта. Программа содержит 2 класса: Form1-класс формы игры, класс Login.cs содержит методы работы с бинарным файлом логинов и результатов. Этот класс использует модуль Form1 во время авторизации и при отображении таблицы рекордов.



Рисунок 3 Диаграмма классов

### 2.2.4. Диаграмма активности

На рисунке 4 показана диаграмма активности (деятельности) во время сеанса игры.

Диаграмма активности – это UML-диаграмма, на которой показаны действия, состояния. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов – вложенных видов деятельности и отдельных действий action, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.



Рисунок 4 Диаграмма активности сеанса игры

## 2.3. Текст программы

Текст программы в соответствии с ГОСТ 19.101-77 (СТ СЭВ 1626-79) и ГОСТ 19.401-79 (СТ СЭВ 3746-82) представляет собой запись программы на исходном языке программирования с необходимыми комментариями. Текст программы представляет собой документ, выполненный машинным способом, и приведен в приложении В.

## 2.4. Описание программы

1. Общие сведения:

Данное приложение создано на языке программирования C# в среде разработки Visual Studio и предназначено для вычисления определителя и обратной матрицы матрицы N x N. Для работы программы необходима операционная система Windows.

2. Назначение программы:

Программа для вычисления определителя матрицы N \* N и обратной матрицы предназначена для решения математических задач в учебных и профессиональных целях. Она облегчает процесс вычисления определителя матрицы и обратной матрицы для пользователя.

3. Описание логической структуры:

Программа состоит из двух основных модулей: вычисление определителя матрицы и вычисление обратной матрицы. Каждый модуль включает в себя несколько подмодулей и использует математические алгоритмы для вычислений.

4. Используемые технические и программные средства:

Для создания данного приложения использовались следующие технические и программные средства:

- Язык программирования C#;

- Среда разработки Visual Studio;

- Библиотека математических функций MathNet.Numerics.

5. Вызов и загрузка:

Для запуска программы необходимо открыть исполняемый файл и воспользоваться соответствующими функциями для вычисления определителя и обратной матрицы матрицы N x N.

6. Входные и выходные данные:

Входные данные для программы – матрица N x N.

Выходные данные – определитель матрицы и ее обратная матрица. Данные могут быть выведены на экран или сохранены в файл.

### 2.4.1. Общие сведения

Программа для вычисления определителя матрицы N \* N и обратной матрицы была создана на языке программирования C# в среде разработки Visual Studio и предназначена для работы под управлением операционной системы Windows.

### 2.4.2. Функциональное назначение

Программа для вычисления определителя матрицы N \* N и обратной матрицы предназначена для решения математических задач в учебных и профессиональных целях. Она позволяет увеличить эффективность работы с матрицами и упрощает процесс вычисления определителя и обратной матрицы для пользователя. Данная программа может использоваться для решения задач в математических науках, при проектировании и анализе систем, а также в инженерных, экономических и финансовых областях.

### 2.4.3. Описание логической структуры системы

Программа содержит 5 модуля:

Form1.cs - модуль определителя матрицы, Form2.cs – модуль обратной матрицы, Form3.cs – модуль главное меню, Form4.cs – модуль справочная инвормация, класс Matrix содержит методы работы с бинарным файлом логинов и результатов. Этот класс использует модуль Form1, Form2 для ввода данных в функции расчета матрицы.

Программа использует функции следующих библиотек среды С#:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

Исполняемый файл программы создан средствами среды C#, имеет имя   
Determinate.exe и размер 16 751 байт.

#### 2.4.3.1. Описание методов класса Matrix.cs

Класс Matrix предназначен для организации работы с расчетами матрицы.

Модуль содержит одно поле N и двумерный массив Data размером N x N, а также соответствующие им свойства для чтения и установки значений полей.

Также в модуле реализовано: public double this[int x, int y] свойство для доступа к элементам массива;

Matrix Exclude(int row, int col) метод для исключения из матрицы строки row и столбца col;

public double CalculateDeterminant() метод для вычисления определителя матрицы.

public string CalculateDeterminantSteps() метод для вычисления определителя матрицы с выводом промежуточных шагов.

Всего в модуле 3 поля и 3 метода.

#### 2.4.3.2. Описание методов класса Form1.cs

Класс формы Form1.cs предназначен для определителя матрицы, данная форма является одной из главных. Связность модуля информационная (последовательная) (СС=9), так как действия внутри модуля связаны с данными, а также важна последовательность действий.

Модуль содержит 10 методов: Модуль содержит форму Form1, которая включает в себя DataGridView initial\_DGV для ввода матрицы, NumericUpDown numericUpDown1 для выбора размерности матрицы, RadioButton random\_radioButton и manual\_radioButton для выбора способа ввода матрицы, а также Button button1 и Label label2 для вывода результата вычисления определителя. label3 (тип Label) - компонент для вывода значения определителя; Также реализованы следующие методы и обработчики событий: Form1() конструктор формы, инициализирующий фильтры для saveFileDialog1 и openFileDialog1; numericUpDown1\_ValueChanged() обработчик события изменения размерности матрицы, обновляющий размерность и инициализирующий значения ячеек DataGridView; random\_radioButton\_CheckedChanged() обработчик события изменения способа ввода матрицы, генерирующий случайные значения ячеек DataGridView в случае выбора random\_radioButton и очищающий ячейки в случае выбора manual\_radioButton; button1\_Click() обработчик нажатия на кнопку "Вычислить", вычисляющий определитель матрицы Matrix на основе значений в DataGridView initial\_DGV и выводящий результат в Label label2; initial\_DGV\_EditingControlShowing() обработчик события начала редактирования ячейки DataGridView, устанавливающий обработчик initial\_DGV\_KeyPress() для фильтрации ввода; initial\_DGV\_KeyPress() метод для фильтрации ввода по условию (ввод цифр, знака "-", знака "," и backspace); initial\_DGV\_CellValueChanged() обработчик изменения значения ячейки DataGridView, очищающий Label label2. richTextBox1 (тип RichTextBox) - компонент для вывода шагов вычисления определителя матрицы.- RecordToFile(string filename) - метод для записи рекорда в файл (Рис. В1); - RecordToTable(string filename, DataGridView dg) - метод для вывода рекордов в таблицу (Рис. В2); - FindLogin(string filename) - метод для поиска логина в файле при авторизации. Дополнительно класс содержит метод FormClosing, который сохраняет матрицу из таблицы initial\_DGV в текстовый файл на жестком диске при закрытии программы. Метод button2\_Click переносит данные из таблицы initial\_DGV в новый экземпляр Form2 и отображает его на экране. Метод button3\_Click извлекает матрицу из таблицы initial\_DGV, вычисляет значение определителя и выводит результат в label3 и richTextBox1. Если возникает ошибка, метод выводит сообщение с описанием ошибки. Метод radioButton1\_CheckedChanged открывает диалоговое окно для выбора файла, считывает матрицу из файла и выводит ее в таблицу initial\_DGV. Если матрица имеет неодинаковое количество столбцов и строк, метод выбрасывает исключение.

#### 2.4.3.3. Описание методов класса Form2.cs

Модуль содержит : - password (тип поля не указан) - поле, предположительно, используется для хранения пароля; - record (тип поля не указан) - поле, предположительно, используется для хранения записей; - login (тип поля не указан) - поле, предположительно, используется для хранения логина. Для каждого поля предоставлены соответствующие свойства для чтения и установки значений. Модуль имеет : - LoginToFile(string filename) - метод для записи нового логина в файл; - RecordToFile(string filename) - метод для записи рекорда в файл (Рис. В1); - RecordToTable(string filename, DataGridView dg) - метод для вывода рекордов в таблицу (Рис. В2); - FindLogin(string filename) - метод для поиска логина в файле при авторизации. Класс Form2 соответствует второй форме программы и содержит следующие поля: - double[,] a - массив для хранения значения матрицы; - saveFileDialog1 (тип SaveFileDialog) - диалоговое окно для сохранения файла; - openFileDialog1 (тип OpenFileDialog) - диалоговое окно для выбора файла; - N (тип int) - размерность матрицы, задаваемая пользователем через компонент numericUpDown1. Класс Form2 также имеет 4 метода: - конструктор по умолчанию, инициализирующий компоненты saveFileDialog1 и openFileDialog1; - конструктор, принимающий двумерный массив объектов data, отображающий его содержимое в таблице dataGridView1; - метод Form2\_Load, вызывающий метод numericUpDown1\_ValueChanged; - метод numericUpDown1\_ValueChanged, изменяющий размерность матрицы и сбрасывающий значения таблицы dataGridView2. Обработчик событий radioButton1\_CheckedChanged генерирует случайные значения матрицы и заполняет ими таблицу dataGridView1. Если выбран режим radioButton2, метод обнуляет значения таблицы dataGridView1. В методе button1\_Click происходит чтение значений из DataGridView, создание объекта матрицы mx, вычисление ее обратной матрицы inv с помощью метода Invert и вывод результатов в dataGridView2. В форме Form2 определены 4 метода для работы с DataGridView: dataGridView1\_KeyPress для ввода только чисел и допустимых символов, dataGridView1\_CellValueChanged для очистки dataGridView2 при изменении значения в dataGridView1, dataGridView1\_EditingControlShowing для подключения обработчика событий клавиатуры dataGridView1\_KeyPress и Form2\_FormClosing для сохранения данных из dataGridView1 в файл при закрытии формы. Для записи данных в файл использован класс StreamWriter. radioButton3\_CheckedChanged для открытия файла с матрицей и отображения ее в DataGridView1, button2\_Click для очистки DataGridView1, dataGridView2\_CellContentClick, dataGridView1\_CellContentClick, button3\_Click, label1\_Click - без каких-либо действий внутри них. В radioButton3\_CheckedChanged происходит открытие диалогового окна выбора файла и считывание его содержимого построчно в массив строк fileText. Если количество строк и столбцов не совпадает, генерируется исключение. Создается двумерный массив a размера fileText.Length на fileText[0].Split(' ').Length и заполняется значениями из файла. Затем DataGridView1 заполняется значениями из массива a, а также устанавливается максимальный размер контрола по вертикали в зависимости от количества строк в матрице. Если произошла ошибка, выводится сообщение об ошибке с помощью MessageBox. В button2\_Click происходит очистка DataGridView1, которая происходит путем удаления всех строк и столбцов из DataGridView1.

#### 2.4.3.3. Описание методов класса Form3.cs

Класс Form3, который является дочерней формой от главной формы приложения. Класс Form3 содержит несколько методов: конструктор по умолчанию (public Form3()), методы button1\_Click, button2\_Click и button3\_Click, которые вызывают другие формы.

Метод button1\_Click открывает форму Form2, которая предназначена для работы с матрицами, путем вызова конструктора формы Form2 и вызова метода ShowDialog для отображения формы. Затем метод скрывает текущую форму, вызвавший ее объект (Form3).

Метод button2\_Click открывает форму Form1, которая предназначена для авторизации пользователей, путем вызова конструктора формы Form1 и вызова метода ShowDialog для отображения формы. Затем метод скрывает текущую форму, вызвавший ее объект (Form3).

Метод button3\_Click открывает форму Form5, которая предназначена для работы с игрой, путем вызова конструктора формы Form5 и вызова метода ShowDialog для отображения формы. Затем метод скрывает текущую форму, вызвавший ее объект (Form3).

### 2.4.4. Используемые технические и программные средства

Для нормального функционирования данной информационной системы необходим компьютер, клавиатура, мышь и следующие технические средства:

- процессор Intel или другой совместимый;

- объем свободной оперативной памяти ~500 Кб;

- объем необходимой памяти на жестком диске ~3Мб;

- стандартный VGA-монитор или совместимый;

- стандартная клавиатура;

- манипулятор «мышь».

### 2.4.5. Вызов и загрузка

Программа может быть загружена как с диска, так и с жесткого диска. В последнем случае требуется предварительно переписать программу с диска на жесткий диск.

Исполняемым файлом программы является файл Determinate.exe. Для его запуска необходимо дважды щелкнуть по исполняемому файлу левой кнопкой мышки.

## 2.5. Руководство оператора

### 2.5.1. Назначение программы

Данная программа предназначена для решения матричных задач, а именно для вычисления определителя матрицы и обратной матрицы. Она позволяет оператору быстро и эффективно решать матричные задачи с помощью ввода матриц и получения результатов.

### 2.5.2. Условия выполнения программы

Для успешного выполнения программы необходимо выполнение следующих условий:

- Наличие Windows 7,8 или 10 операционных систем - Установленный на компьютере .NET Framework 4.0 или выше

- Свободное место на жестком диске для установки программы

### 2.5.3. Выполнение программы и сообщение оператору

После запуска программы оператору будет предложено выбрать одну из двух опций - «Вычисление определителя матрицы» или «Вычисление обратной матрицы». После выбора опции оператору необходимо ввести данные матрицы в соответствующие поля. Программа поддерживает матрицы размерности N\*N.

После ввода матрицы оператору будет предложено нажать кнопку «Вычислить». В результате программный алгоритм вычислит определитель матрицы или обратную матрицу и выведет ответ на экран. В случае ошибки введенных данных оператору будет выведено сообщение об ошибке с указанием причины. В случае, если оператор захочет вычислить определитель или обратную матрицу другой матрицы, ему необходимо нажать кнопку «Очистить» и повторить процедуру ввода данных.

## 2.6. Программа и методика испытаний

### 2.6.1. Объект испытаний

Объектом испытаний является игровая программа Determinate.exe. Данная программа предназначена для решения матричных задач, а именно для вычисления определителя матрицы и обратной матрицы.

### 2.6.2. Цель испытаний

Целью испытаний является проверка соответствия программы требованиям Технического Задания.

### 2.6.3. Требования к программе

В процессе испытаний подлежат проверке следующие требования к программе:

#### 2.6.3.1. Требования к функциональным характеристикам

**1 версия:**

* программа должна иметь возможность ввода матрицы целых чисел размером не больше 20\*20, размерность матрицы задается вручную с клавиатуры;
* матрица должна содержать числа в интервале (-100;100);
* если все введенные значения верны, программа должна вычислять определитель матрицы N\*N, где n≤20;
* программа должна иметь функцию нахождения обратной матрицы;
* программа должна осуществлять вывод результатов на экран.

2 версия:

* программа должна иметь возможность ввода матрицы вручную, случайным образом в заданном интервале или из текстового файла специального формата;
* матрицу можно загрузить из текстового файла специального формата, где числа будут идти через пробел, и каждая строка матрицы будет начинаться с новой строки файла;
* должна быть предусмотрена возможность вывода на экран хранящейся в файле матрицы, ее корректировки и повторной записи в файл;
* программа должна выводить сообщение, если были введены неверные значения;
* программа должна предоставлять справку о математическом методе вычисления определителя и обратной матрицы;
* программа должна иметь возможность вывода подробного решения;
* программа должна обеспечивать возможность вывода результатов на экран, а также на принтер или в текстовый файл.

#### 2.6.3.2. Требования к информационной и программной совместимости

Для полноценного функционирования данной системы необходимо наличие операционной системы выше Microsoft Windows 7 или совместимой. Язык интерфейса – русский.

#### 2.6.3.3. Требования к маркировке и упаковке

Программа должна поставляться в виде проекта, исполняемого (еxе) файла, установщика и документации.

#### 2.6.3.4. Требования к транспортировке и хранению

Программа распространяется в электронном виде. Требования к транспортировке и хранению не предъявляются.

### 2.6.4. Требования к программной документации

На испытания должны быть представлены следующие программные документы:

* техническое задание
* текст программы
* описание программы
* руководство оператора
* описание языка

### 2.6.5. Средства и порядок испытаний

Для проведения испытаний необходимы следующие технические средства:

* процессор Intel или другой совместимый;
* объем свободной оперативной памяти ~500 Кб;
* объем необходимой памяти на жестком диске ~3Мб;
* стандартный VGA-монитор или совместимый;
* стандартная клавиатура;
* манипулятор «мышь».

Для проведения испытаний необходимы следующие программные средства:

Операционная система Windows 7.

Испытания проводятся в следующем порядке:

1) проверяется наличие и комплектность программной документации (п.2.6.4)

2) проверяется соответствие требованиям к маркировке и упаковке (п.2.6.3.3)

3) проверяется соответствие требованиям к функциональным характеристикам (п.2.6.3.1)

4) проверяется соответствие требованиям к информационной и программной совместимости (п.2.6.3.2)

### 2.6.6. Методы испытаний

Методы испытаний: для проверки каждого требования к программе, указанного в подпункте «Средства и порядок испытаний», необходимо выполнить следующие действия:

1. Для проверки наличия и комплектности программной документации (п.2.6.4) необходимо открыть файл с руководством пользователя и убедиться в наличии всех необходимых разделов и информации.

2. Для проверки соответствия требованиям к маркировке и упаковке (п.[2.6.3.3](https://2.6.3.3/)) необходимо проверить наличие у программы соответствующей маркировки и упаковки.

3. Для проверки соответствия требованиям к функциональным характеристикам (п.[2.6.3.1](https://2.6.3.1/)) необходимо запустить программу, ввести исходные данные (матрицу) и вычислить ее определитель и/или обратную матрицу. Результаты вычислений должны соответствовать математически верным решениям при введенных исходных данных.

4. Для проверки соответствия требованиям к информационной и программной совместимости (п.[2.6.3.2](https://2.6.3.2/)) необходимо быть уверенным, что программа совместима с операционной системой Windows 7 и не вызывает непредвиденных ошибок при ее использовании.

Для проведения испытаний необходимы следующие технические средства: процессор Intel или другой совместимый; объем свободной оперативной памяти ~500 Кб; объем необходимой памяти на жестком диске ~3Мб; стандартный VGA-монитор или совместимый; стандартная клавиатура; манипулятор «мышь».

Для проведения испытаний необходимо иметь операционную систему Windows 7.

#### 2.6.6.1. Для проверки способности программы осуществлять вычисление определителя, необходимо:

* запустить программу
* выбрать определитель
* заполнить строки и столбцы матрицы
* нажать вычислить определитель

#### 2.6.6.2. Для проверки способности программы осуществлять вычисление определителя, с пошаговым решение, необходимо:

* запустить программу
* выбрать определитель
* заполнить строки и столбцы матрицы
* нажать подробное решение

#### 2.6.6.3. Для проверки способности программы реализовывать заполнение строк и столбцов матрицы вручную, необходимо:

* запустить программу
* выбрать определитель или обратная матрица
* выбрать ввод вручную
* убедиться, что вы можете вставлять числа в поля и строки.

#### 2.6.6.4. Для проверки способности программы реализовывать заполнение строк и столбцов матрицы случайным образом, необходимо:

* запустить программу
* выбрать определитель или обратная матрица
* выбрать ввод случайно
* убедиться, что матрица заполнена.

#### 2.6.6.5. Для проверки способности программы реализовывать заполнение строк и столбцов матрицы из файла, необходимо:

* запустить программу
* выбрать определитель или обратная матрица
* выбрать ввод из файла
* выбрать файл с компьютера
* убедиться, что матрица заполнена.

#### 2.6.6.6. Для проверки способности программы осуществлять вычисление обратной матрицы, необходимо:

* запустить программу
* обратной матрицы
* заполнить строки и столбцы матрицы
* нажать обратная матрица

#### 2.6.6.7. Для проверки способности программы изменять размерность матрицы:

* запустить программу
* выбрать определитель или обратная матрица
* изменить размерность
* убедиться, что изменилось кол-во строк и столбцов

#### 2.6.6.8. Для проверки наличия в программе справки:

* запустить программу
* выбрать справка
* убедиться, что в содержимом форме находиться справка

#### 2.6.6.9. Для проверки способности программы очищать поле матрицы:

* запустить программу
* обратной матрицы
* заполнить строки и столбцы матрицы
* нажать обратная матрица
* нажать очистить

## 2.7. Протокол испытаний

Результаты испытаний программы представлены в таблице 1, рисунки приведены в приложении Б.

**Результаты испытаний программы**

*Таблица 1*

| **№** | **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Результат** |
| --- | --- | --- | --- | --- |
| 1 | Способность программы осуществлять вычисление определителя | * запустить программу * нажать «определитель матрицы» * заполнить строки и столбцы матрицы * нажать вычислить определитель | Открывается окно, в котором можно решить определитель, после нажатия кнопки вычислить определитель  Пользователь увидит определить матрицы, он будет выведен в label2 | Пройден  Рис. Б1 |
| 2 | Способность программы осуществлять вычисление определителя, с пошаговым решение, необходимо | * запустить программу * нажать «определитель матрицы» * заполнить строки и столбцы матрицы * нажать подробное решение | Открывается окно, в котором можно решить определитель, после нажатия кнопки вычислить определитель  Пользователь увидит в RichTextBox пошаговое решение | Пройден  Рис. Б2 |
| 3 | Способность программы реализовывать заполнение строк и столбцов матрицы вручную | * запустить программу * нажать на определитель или обратная матрица * В GroupBox выбрать вид ввода вручную * Убедиться что вы можете заполнять столбцы и строки матрицы в DataGridView | При выборе ввода вручную пользователь может изменять строки и столбцы для заполнения матрицы вручную, например используя клавиатуру | Пройден  Рис. Б3 |
| 4 | Способность программы реализовывать заполнение строк и столбцов матрицы случайным образом | * запустить программу * нажать на определитель или обратная матрица * В GroupBox выбрать вид ввода случайно * Убедиться, что столбцы и строки матрицы заполнены случайным образом в DataGridView | При выборе ввода случайно строки и столбцы для заполнения матрицы будут заполнены случайным образом | Пройден  Рис. Б4 - Б5 |
| 5 | Способность программы реализовывать заполнение строк и столбцов матрицы из файла | * запустить программу * нажать на определитель или обратная матрица * В GroupBox выбрать вид ввода из файла * Открыть в проводнике файл из которого вы хотите взять данные * Убедиться, что вы столбцы и строки матрицы в DataGridView заполнены идентично по соотношению данным из файла | При выборе ввода из файла пользователю будет предложено выбрать файл из которого будет производиться импорт данных, строки и столбцы для заполнения матрицы будут заполнены соответственно данным хранящимся в файле | Пройден  Рис. Б6 |
| 6 | Способность программы осуществлять вычисление обратной матрицы | * запустить программу * нажать обратная матрица * заполнить строки и столбцы матрицы * нажать обратная матрица * заполнить строки и столбцы матрицы * нажать обратная матрица | После перехода в форму для вычисления обратной матрицы пользователь заполняет столбцы и строки матрицы, после этого нажимает обратная матрица, выводится результат вычисление в dataGriedView2 | Пройден  Рис. Б7 |
| 7 | Способность программы изменять размерность | * запустить программу * выбрать определитель или обратная матрица * в numericupdown изменить размер для матрицы * Убедиться в изменения посмотрев в dataGriedView | Пользователь может менять размерность матрицы в окнах для решения определителя или обратной матрицы в numericupdown, результат изменений он может увидеть в dataGriedView | Пройден  Рис. Б7 |
| 8 | Способность программы демонстрации информационной справки | * запустить программу * выбрать справку * убедиться в наличии справки | Пользователь может прочесть информационную справку программы, нажав в главном меню на кнопку Справка | Пройден  Рис. Б8 |
| 9 | Способность программы очищать поле матрицы | * запустить программу * выбрать обратную матрицу * заполнить поля * нажать очистить | Пользователь может очищать поля с помощью кнопки очистить в форме обратной матрицы | Пройден  Рис. Б8 |

ЗАКЛЮЧЕНИЕ

Разработанная в ходе выполнения курсового проекта программа удовлетворяет всем требованиям технического задания, что подтверждается протоколом испытаний.

Разработанная программа может быть использована в развлекательных целях, для отдыха и релаксации.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд ., Язык программирования C#, 4-е издание, СПб:-ПИТЕР, 2020
2. Фленов М., Библия C#, 3-е издание, СПб:-БХВ, 2019
3. Культин Н., Microsoft Visual C# в задачах и примерах, 2-е издание, СПб:-БХВ, 2015
4. Жанры игр и их особенности [Электронный ресурс] — URL: [https://sreda.temadnya.com/ 1043184354218215685/zhanry-igr-i-ih-osobennosti/](https://sreda.temadnya.com/%201043184354218215685/zhanry-igr-i-ih-osobennosti/) (дата обращения: 12.09.2020)
5. Описание языка C# [Электронный ресурс] — URL: https://techrocks.ru/2019/02/16/c-sharp-programming-language-overview/ (дата обращения: 03.12.2019)
6. Стандарты Единой Системы Программной Документации:

ГОСТ 19.105-78 Общие требования к программным документам

ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом

ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению

ГОСТ 19.301-78 Программа и методика испытаний. Требования к содержанию и оформлению

ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению

ГОСТ 19.402-78 Описание программы. Требования к содержанию и оформлению

ГОСТ 19.505-79 Руководство оператора. Требования к содержанию и оформлению

ГОСТ 19.701-90 Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения

ПРИЛОЖЕНИЕ А

Министерство ОБРАЗОВАНИЯ И науки Российской Федерации  
Федеральное автономное образовательное учреждение высшего профессионального образования  
«Санкт-Петербургский государственный политехнический университет Петра Великого»  
(ФГАОУ ВО «СПбПУ)  
**Институт среднего профессионального образования**

УТВЕРЖДАЮ  
Председатель ПЦК   
\_\_\_\_\_\_\_\_\_\_\_ Андреев В.А.  
\_\_ .\_\_. 2023

Игра с графическим интерфейсом «Вычисление определителя матрицы N \* N и обратной матрицы»  
**Техническое задание**  
Листов 7

ПРОВЕРИЛ  
Преподаватель   
\_\_\_\_\_\_\_\_\_\_\_ Девятко Н.С.  
27.02.2023

ВЫПОЛНИЛ  
Студент группы 32919/6   
\_\_\_\_\_\_\_\_\_\_\_ Степанов К. О.  
27.02.2023

2023

# ВВЕДЕНИЕ

* 1. Полное наименование программной разработки: «Программа для вычисления определителя (детерминанта) матрицы.»
  2. Вычислительная программа с графическим интерфейсом: человек может в любое время проводить вычисления, требуемые для нахождения определителя (детерминанта) матрицы, а также легко модифицировать ее (менять размер, записывать данные вручную или случайно).
  3. В соответствии с заданием программный продукт состоит из теоретической и практической частей. Теоретическая часть включает описание предметной области задачи, анализ методов её решения, обзор и обоснование выбора средств программирования. Практическая часть включает подробное описание проектирования программного продукта, разработку и реализацию на языке программирования C# с использованием среды программирования Visual Studio программных модулей программного продукта.

# ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

* 1. Разработка ведётся на основании задания к курсовому проекту по профессиональному модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем» МДК 01.01 «Разработка программных модулей» и утверждена Институтом среднего профессионального образования.

# НАЗНАЧЕНИЕ РАЗРАБОТКИ

* 1. Основное назначение программного продукта: Программа будет применяться для вычисления определителя (детерминанта) матрицы.
  2. Эксплуатационное назначение программного продукта: программа предназначена для широкого круга пользователей, предназначена для облегчения вычисления определителя матрицы.

4. ТРЕБОВАНИЯ К РАЗРАБОТКЕ

4.1. Требования к функциональным характеристикам:

**1 версия:**

* программа должна иметь возможность ввода целых чисел вручную в матрицу, матрица может содержать числа (-100;100);

- матрица размером не больше 20\*20, размерность матрицы задается вручную с клавиатуры;

- если все введенные значения верны, программа должна вычислять определить матрицы N\*N, n≤20;

* программа должна иметь функцию нахождения обратной матрицы;
* программа должна осуществлять вывод результатов на экран;

**2 версия:**

* программа должна иметь возможность ввода вручную, случайным образом в заданном интервале или из текстового файла специального формата;
* матрицу можно загрузить из текстового файла специального формата: числа будут идти через пробел и каждая строка матрицы с новой строки файла;
* должна быть предусмотрена возможность вывода на экран хранящейся в файле матрицы, ее корректировки и повторной записи в файл;
* программа должна выводить сообщение если были введены неверные значения;

- программа должна предоставлять справку, о математическом методе вычисления определителя и обратной матрицы

- возможность вывода подробного решения

- вывод результатов на экран, а по желанию пользователя – в текстовый файл.

4.2. Требования к надежности:

* использование лицензированного программного обеспечения;
* проверка программы на наличие вирусов;
* организация бесперебойного питания.

4.3. Требования к составу и параметрам технических средств

Для нормального функционирования данной информационной системы необходим компьютер, клавиатура, мышь и следующие технические средства:

- процессор Intel или другой совместимый;

- объем свободной оперативной памяти ~500 Кб;

- объем необходимой памяти на жестком диске ~20Мб;

- стандартный VGA-монитор или совместимый;

- стандартная клавиатура;

- манипулятор «мышь».

4.4. Требования к информационной и программной совместимости

Для полноценного функционирования данной системы необходимо наличие операционной системы выше Microsoft Windows XP или совместимой. Язык интерфейса – русский.

4.5. Требования к маркировке и упаковке

Программа должна поставляться в виде проекта, исполняемого (еxе) файла, установщика и документации.

4.6. Требования к транспортировке и хранению

Программа распространяется в электронном виде. Требования к транспортировке и хранению не предъявляются.

4.7. Специальные требования

Теоретическая часть включает описание предметной области задачи, анализ методов её решения, обзор и обоснование выбора средств программирования. Практическая часть включает подробное описание проектирования программного продукта, разработку и реализацию на языке программирования C# с использованием среды программирования Visual Studio программных модулей программного продукта.

5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

5.1. Предварительный состав программной документации:

- «Техническое задание»;

- разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии;

- разрабатываемое программное обеспечение должно включать справочную систему.

5.2. Перечень материалов пояснительной записки



Рисунок 1 – Перечень материалов пояснительной записки

6. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Технико-экономические показатели не рассчитываются.

7. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Таблица 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Содержание стадии** | **Содержание этапа** | **Срок 2023 г.** | | **Форма  отчетности** |
| **начало** | **конец** |
| Техническое задание | Составление технического задания | 20.02 | 25.02 | Техническое задание |
| Эскизный проект | Проектирование программы | 27.02 | 11.03 | UML-диаграммы |
| Рабочий проект | Разработка спецификаций | 13.03 | 18.03 | Спецификации отдельных компонентов |
| Составление программы | 20.03 | 08.04 | Программная документация |
| Приёмо-сдаточные испытания | 10.04 | 15.04 | Протокол испытаний (п. 2.7 пояснительной записки) |
| Приёмка | Защита курсового проекта | 17.04 | 22.04 | Оценка за курсовой проект |

8. ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

8.1. Порядок контроля

Контроль выполнения должен осуществляться руководителем курсового проекта (преподавателем) в соответствие с п.7.

8.2. Порядок приемки

Приемка должна осуществляться с участием руководителя после проведения приемо-сдаточных испытаний. В результате защиты курсового проекта должна быть выставлена оценка за курсовой проект.

ПРИЛОЖЕНИЕ Б

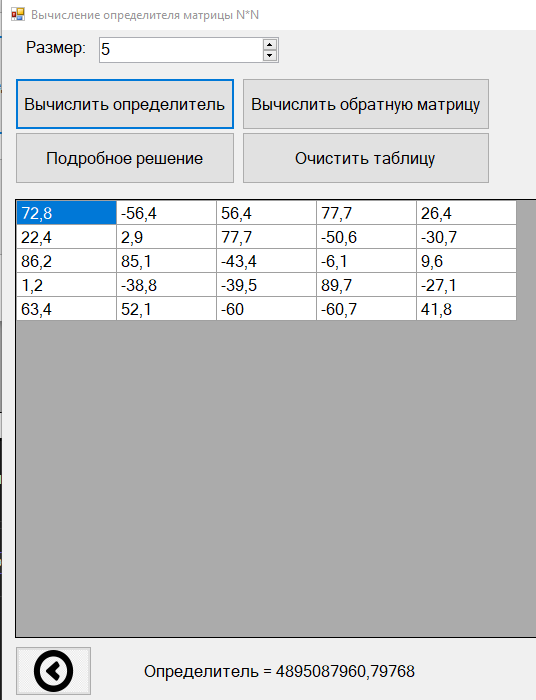


Рисунок Б1 Способность программы осуществлять вычисление определителя

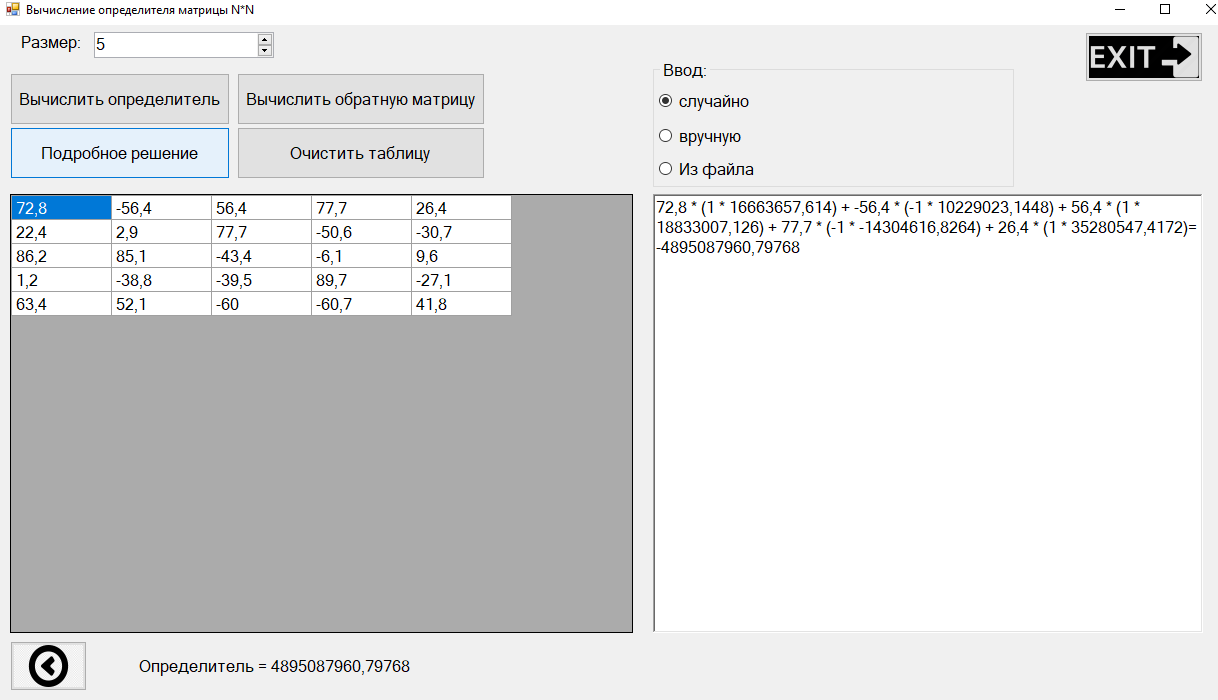


Рисунок Б2 Способность программы осуществлять вычисление определителя, с пошаговым решение, необходимо

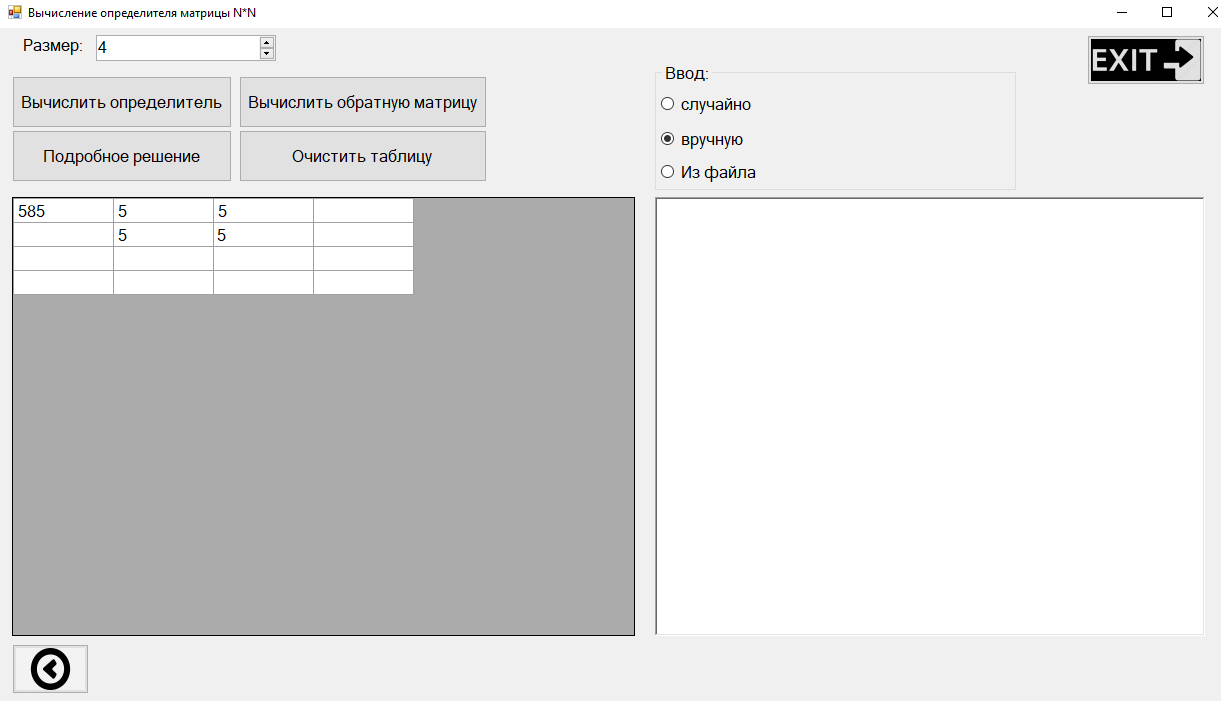


Рисунок Б3 Способность программы реализовывать заполнение строк и столбцов матрицы вручную

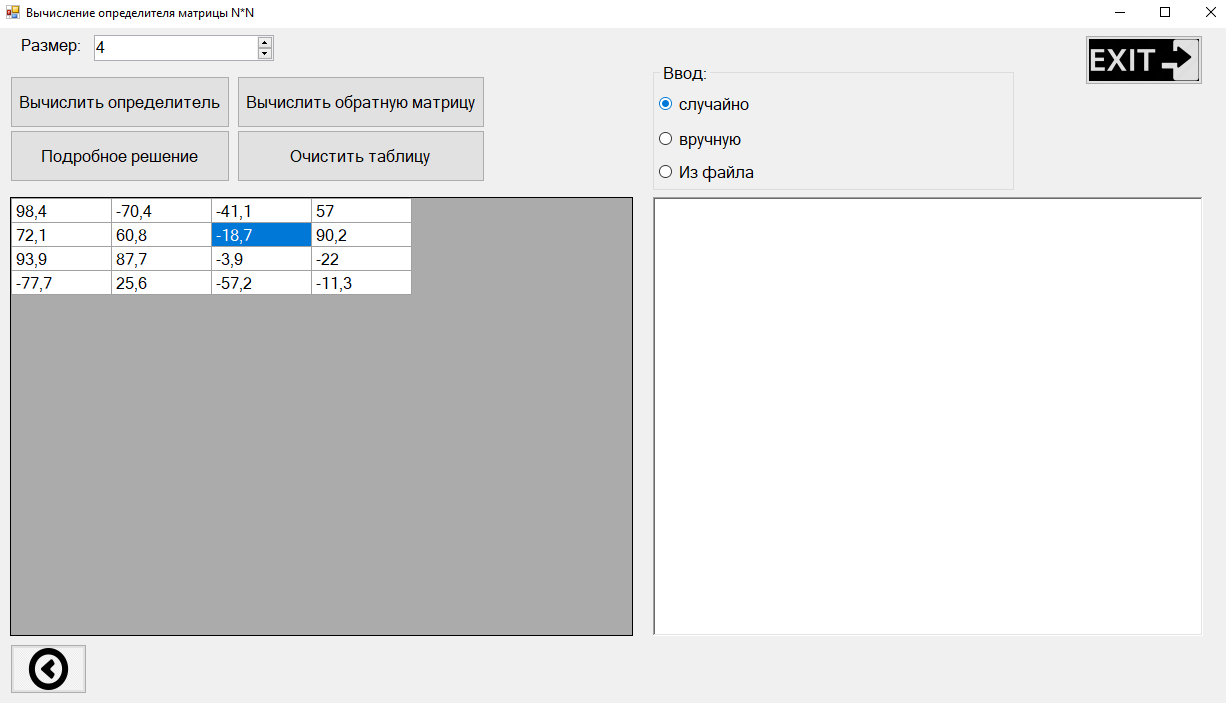


Рисунок Б4 Способность программы реализовывать заполнение строк и столбцов матрицы случайным образом

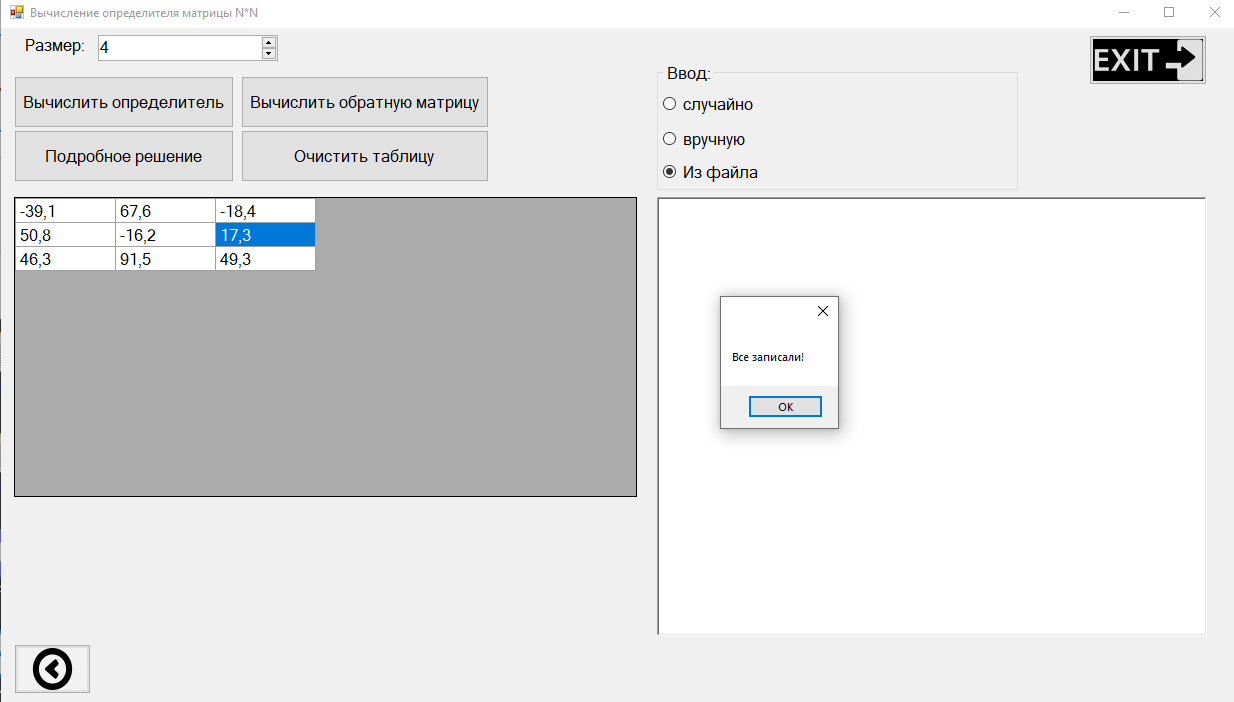


Рисунок Б5 Способность программы реализовывать заполнение строк и столбцов матрицы из файла

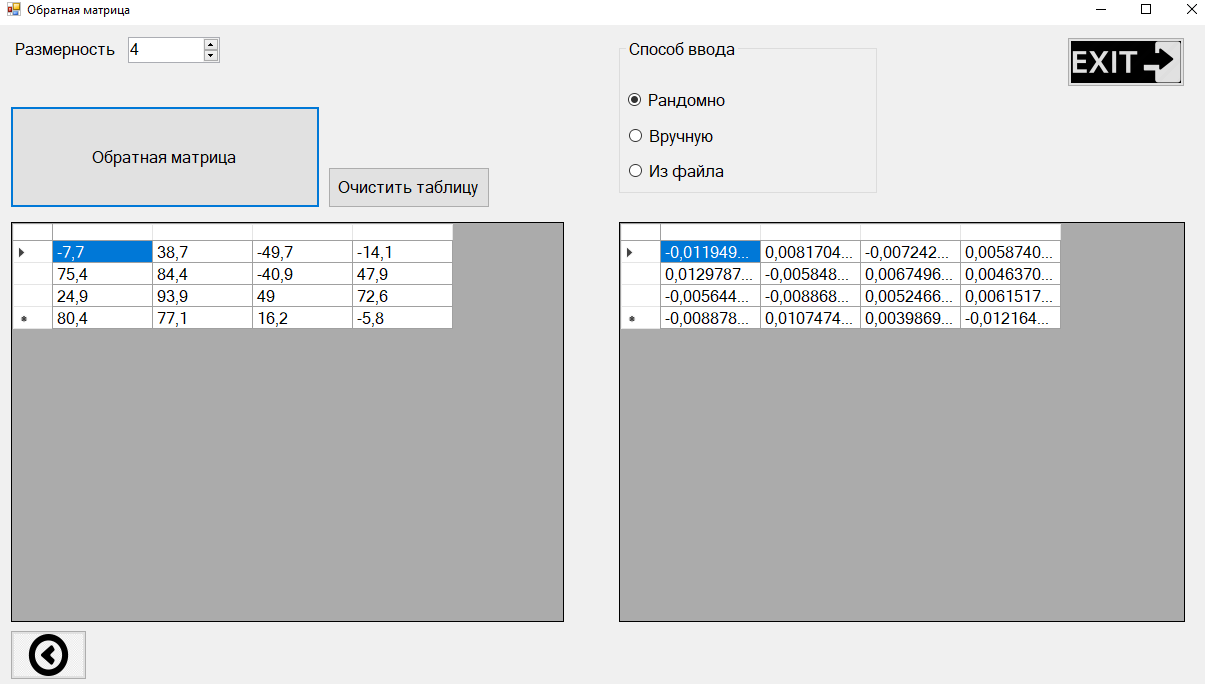


Рисунок Б6 Способность программы осуществлять вычисление обратной матрицы

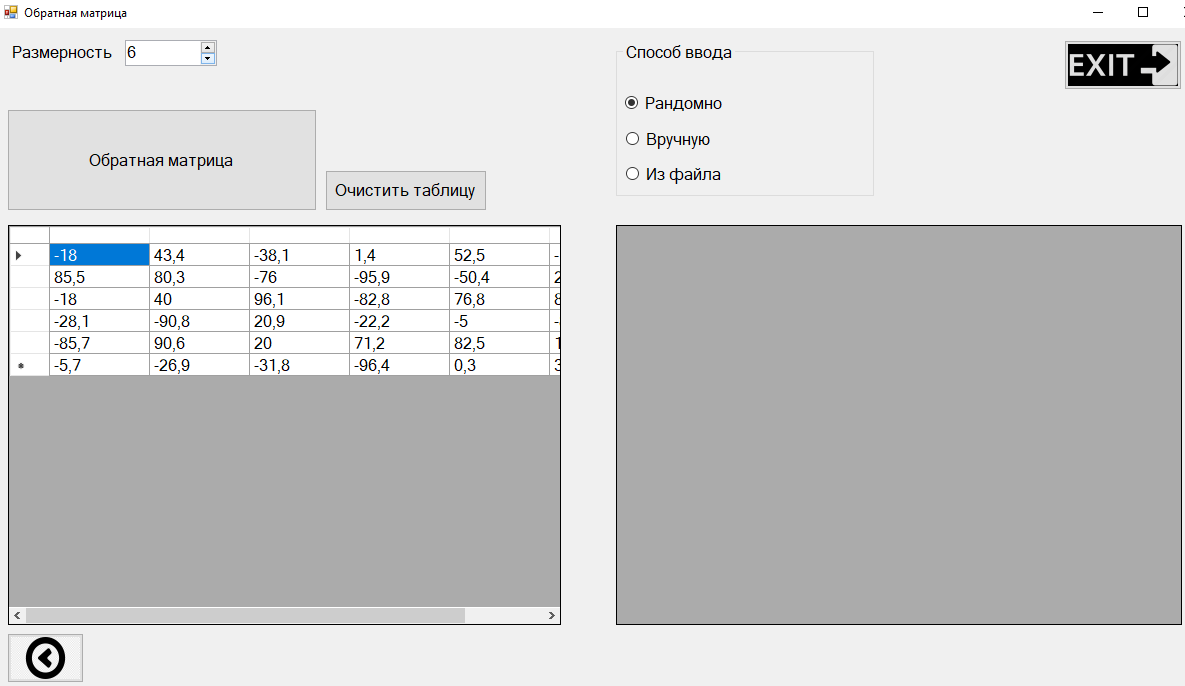


Рисунок Б7 Способность программы изменять размерность

ПРИЛОЖЕНИЕ В

**Текст программы**

**Модуль Determinant.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Determinant

{

public partial class Determinant : Form

{

double[,] a;

public Determinant()

{

InitializeComponent();

saveFileDialog1.Filter = "Text files(\*.txt)|\*.txt|All files(\*.\*)|\*.\*";

openFileDialog1.Filter = "Text files(\*.txt)|\*.txt|All files(\*.\*)|\*.\*";

}

int N;

private void Form1\_Load(object sender, EventArgs e)

{

numericUpDown1\_ValueChanged(sender, e);

}

private void numericUpDown1\_ValueChanged(object sender, EventArgs e)

{

N = initial\_DGV.RowCount = initial\_DGV.ColumnCount = (int)numericUpDown1.Value;

if (random\_radioButton.Checked) random\_radioButton\_CheckedChanged(sender, e);

label2.Text = string.Empty;

}

private void random\_radioButton\_CheckedChanged(object sender, EventArgs e)

{

if (random\_radioButton.Checked)

{

Random ran = new Random();

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

{

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

{

initial\_DGV[j, i].Value = Math.Round(ran.NextDouble() \* (99 + 99) - 99, 1);

}

}

initial\_DGV.ReadOnly = true;

}

if (manual\_radioButton.Checked)

{

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

{

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

{

initial\_DGV[j, i].Value = null;

}

}

initial\_DGV.ReadOnly = false;

}

label2.Text = string.Empty;

}

private void button1\_Click(object sender, EventArgs e)

{

Matrix mx = new Matrix(N);

try

{

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

{

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

{

if (initial\_DGV[j, i].Value == null) initial\_DGV[j, i].Value = 0;

mx[i, j] = Convert.ToDouble(initial\_DGV[j, i].Value);

}

}

label2.Text = $"Определитель = {mx.CalculateDeterminant()}";

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void initial\_DGV\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

e.Control.KeyPress += new KeyPressEventHandler(initial\_DGV\_KeyPress);

}

private void initial\_DGV\_KeyPress(object sender, KeyPressEventArgs e)

{

string t = ((TextBox)sender).Text;

char k = e.KeyChar;

e.Handled = !(char.IsDigit(k) || k == ',' && !t.Contains(",") || k == '-' && !t.Contains("-") || k == (char)Keys.Back);

}

private void initial\_DGV\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

label2.Text = string.Empty;

}

private void button2\_Click(object sender, EventArgs e)

{

// сохраняем значения ячеек DataGridView в массив

var data = new object[initial\_DGV.Rows.Count, initial\_DGV.Columns.Count];

for (int i = 0; i < initial\_DGV.Rows.Count; i++)

{

for (int j = 0; j < initial\_DGV.Columns.Count; j++)

{

data[i, j] = initial\_DGV.Rows[i].Cells[j].Value;

}

}

// открываем вторую форму и передаем значения через конструктор

var form2 = new InversMatrix(data);

form2.Show();

}

private void button3\_Click(object sender, EventArgs e)

{

Matrix mx = new Matrix(N);

try

{

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

{

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

{

if (initial\_DGV[j, i].Value == null) initial\_DGV[j, i].Value = 0;

mx[i, j] = Convert.ToDouble(initial\_DGV[j, i].Value);

}

}

richTextBox1.Text = mx.CalculateDeterminantSteps();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void groupBox1\_Enter(object sender, EventArgs e)

{

}

private void manual\_radioButton\_CheckedChanged(object sender, EventArgs e)

{

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

try

{

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

// получаем выбранный файл

string filename = openFileDialog1.FileName;

// читаем файл в строку

string[] fileText = System.IO.File.ReadAllLines(filename);

if (fileText.Length != fileText[0].Split(' ').Length) throw new Exception("Надо что б столбики равнялись строчечкам");

a = new double[fileText.Length, fileText[0].Split(' ').Length];

initial\_DGV.RowCount = a.GetLength(0);

initial\_DGV.ColumnCount = a.GetLength(1);

initial\_DGV.MaximumSize = new Size(initial\_DGV.Width, initial\_DGV.RowCount \* 100);

for (int i = 0; i < initial\_DGV.RowCount; i++)

{

for (int j = 0; j < initial\_DGV.ColumnCount; j++)

{

initial\_DGV.Rows[i].Cells[j].Value = double.Parse(fileText[i].Split(' ')[j]);

}

}

MessageBox.Show("Все записали!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void label3\_Click(object sender, EventArgs e)

{

}

private void button4\_Click(object sender, EventArgs e)

{

initial\_DGV.Rows.Clear();

initial\_DGV.Columns.Clear();

}

private void button5\_Click(object sender, EventArgs e)

{

this.Hide();

MainMenu MainMenu = new MainMenu();

MainMenu.ShowDialog();

}

private void button7\_Click(object sender, EventArgs e)

{

// Создаем диалоговое окно для сохранения файла

SaveFileDialog saveFileDialog = new SaveFileDialog();

// Устанавливаем начальную директорию и фильтр файлов

saveFileDialog.InitialDirectory = @"C:\";

saveFileDialog.Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

// Открываем диалоговое окно и проверяем, что пользователь нажал "ОК"

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

// Открываем файл для записи

using (StreamWriter writer = new StreamWriter(saveFileDialog.FileName))

{

// Перебираем строки таблицы

foreach (DataGridViewRow row in initial\_DGV.Rows)

{

// Перебираем ячейки в строке

foreach (DataGridViewCell cell in row.Cells)

{

// Записываем значение в файл без пробела в конце строки

writer.Write(cell.Value);

// Если это не последняя ячейка в строке, добавляем разделитель

if (cell.ColumnIndex < row.Cells.Count - 1)

{

writer.Write(" ");

}

}

// Переходим на следующую строку

writer.WriteLine();

}

}

}

Application.Exit();

}

}

}

**Модуль InversMatrix.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Determinant

{

public partial class InversMatrix : Form

{

double[,] a;

public InversMatrix()

{

InitializeComponent();

saveFileDialog1.Filter = "Text files(\*.txt)|\*.txt|All files(\*.\*)|\*.\*";

openFileDialog1.Filter = "Text files(\*.txt)|\*.txt|All files(\*.\*)|\*.\*";

}

int N;

public InversMatrix(object[,] data)

{

InitializeComponent();

numericUpDown1.Value = data.GetLength(0);

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

{

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

{

dataGridView1.Rows[i].Cells[j].Value = data[i, j];

}

}

}

private void Form2\_Load(object sender, EventArgs e)

{

numericUpDown1\_ValueChanged(sender, e);

}

private void numericUpDown1\_ValueChanged(object sender, EventArgs e)

{

N = dataGridView1.RowCount = dataGridView1.ColumnCount = (int)numericUpDown1.Value;

if (radioButton1.Checked) radioButton1\_CheckedChanged(sender, e);

dataGridView2.Rows.Clear();

dataGridView2.Columns.Clear();

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton1.Checked)

{

Random ran = new Random();

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

{

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

{

dataGridView1[j, i].Value = Math.Round(ran.NextDouble() \* (99 + 99) - 99, 1);

}

}

dataGridView1.ReadOnly = true;

}

if (radioButton2.Checked)

{

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

{

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

{

dataGridView1[j, i].Value = null;

}

}

dataGridView1.ReadOnly = false;

}

dataGridView2.Rows.Clear();

dataGridView2.Columns.Clear();

}

private void button1\_Click(object sender, EventArgs e)

{

Matrix mx = new Matrix(N);

try

{

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

{

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

{

if (dataGridView1[j, i].Value == null) dataGridView1[j, i].Value = 0;

mx[i, j] = Convert.ToDouble(dataGridView1[j, i].Value);

}

}

Matrix inv = mx.Invert(out double det);

if (inv != null)

{

dataGridView2.RowCount = dataGridView2.ColumnCount = N;

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

{

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

{

dataGridView2[j, i].Value = inv[i, j];

}

}

}

else MessageBox.Show("Для выроженной матрицы обратная не существует.", "Результат", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dataGridView1\_KeyPress(object sender, KeyPressEventArgs e)

{

string t = ((TextBox)sender).Text;

char k = e.KeyChar;

e.Handled = !(char.IsDigit(k) || k == ',' && !t.Contains(",") || k == '-' && !t.Contains("-") || k == (char)Keys.Back);

}

private void dataGridView1\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

dataGridView2.Rows.Clear();

dataGridView2.Columns.Clear();

}

private void dataGridView1\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

e.Control.KeyPress += new KeyPressEventHandler(dataGridView1\_KeyPress);

}

private void Form2\_FormClosing(object sender, FormClosingEventArgs e)

{

// Создаем диалоговое окно для сохранения файла

SaveFileDialog saveFileDialog = new SaveFileDialog();

// Устанавливаем начальную директорию и фильтр файлов

saveFileDialog.InitialDirectory = @"C:\";

saveFileDialog.Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

// Открываем диалоговое окно и проверяем, что пользователь нажал "ОК"

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

// Открываем файл для записи

using (StreamWriter writer = new StreamWriter(saveFileDialog.FileName))

{

// Перебираем строки таблицы

foreach (DataGridViewRow row in dataGridView1.Rows)

{

// Перебираем ячейки в строке

foreach (DataGridViewCell cell in row.Cells)

{

// Записываем значение в файл без пробела в конце строки

writer.Write(cell.Value);

// Если это не последняя ячейка в строке, добавляем разделитель

if (cell.ColumnIndex < row.Cells.Count - 1)

{

writer.Write(" ");

}

}

// Переходим на следующую строку

writer.WriteLine();

}

}

}

}

private void radioButton3\_CheckedChanged(object sender, EventArgs e)

{

try

{

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

// получаем выбранный файл

string filename = openFileDialog1.FileName;

// читаем файл в строку

string[] fileText = System.IO.File.ReadAllLines(filename);

if (fileText.Length != fileText[0].Split(' ').Length) throw new Exception("Надо что б столбики равнялись строчечкам");

a = new double[fileText.Length, fileText[0].Split(' ').Length];

dataGridView1.RowCount = a.GetLength(0);

dataGridView1.ColumnCount = a.GetLength(1);

dataGridView1.MaximumSize = new Size(dataGridView1.Width, dataGridView1.RowCount \* 100);

for (int i = 0; i < dataGridView1.RowCount; i++)

{

for (int j = 0; j < dataGridView1.ColumnCount; j++)

{

dataGridView1.Rows[i].Cells[j].Value = double.Parse(fileText[i].Split(' ')[j]);

}

}

MessageBox.Show("Все записали!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void button2\_Click(object sender, EventArgs e)

{

dataGridView1.Rows.Clear();

dataGridView1.Columns.Clear();

}

private void dataGridView2\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void button3\_Click(object sender, EventArgs e)

{

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void button5\_Click(object sender, EventArgs e)

{

this.Hide();

MainMenu MainMenu = new MainMenu();

MainMenu.ShowDialog();

}

private void button3\_Click\_1(object sender, EventArgs e)

{

// Создаем диалоговое окно для сохранения файла

SaveFileDialog saveFileDialog = new SaveFileDialog();

// Устанавливаем начальную директорию и фильтр файлов

saveFileDialog.InitialDirectory = @"C:\";

saveFileDialog.Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

// Открываем диалоговое окно и проверяем, что пользователь нажал "ОК"

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

// Открываем файл для записи

using (StreamWriter writer = new StreamWriter(saveFileDialog.FileName))

{

// Перебираем строки таблицы

foreach (DataGridViewRow row in dataGridView1.Rows)

{

// Перебираем ячейки в строке

foreach (DataGridViewCell cell in row.Cells)

{

// Записываем значение в файл без пробела в конце строки

writer.Write(cell.Value);

// Если это не последняя ячейка в строке, добавляем разделитель

if (cell.ColumnIndex < row.Cells.Count - 1)

{

writer.Write(" ");

}

}

// Переходим на следующую строку

writer.WriteLine();

}

}

}

Application.Exit();

}

}

}

**Модуль** **MainMenu.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Determinant

{

public partial class MainMenu : Form

{

public MainMenu()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

InversMatrix form2 = new InversMatrix();

form2.ShowDialog();

this.Hide();

}

private void button2\_Click(object sender, EventArgs e)

{

Determinant form3 = new Determinant();

form3.ShowDialog();

this.Hide();

}

private void Form3\_Load(object sender, EventArgs e)

{

}

private void button3\_Click(object sender, EventArgs e)

{

Reference form3 = new Reference();

form3.ShowDialog();

this.Hide();

}

}

}

**Модуль** **Reference.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Determinant

{

public partial class Reference : Form

{

public Reference()

{

InitializeComponent();

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void button5\_Click(object sender, EventArgs e)

{

this.Hide();

MainMenu MainMenu = new MainMenu();

MainMenu.ShowDialog();

}

}

}

**Класс Matrix.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Determinant

{

public class Matrix

{

readonly int N;

readonly double[,] Data;

public Matrix(int n)

{

N = n;

Data = new double[N, N];

}

public double this[int x, int y]

{

get => Data[x, y];

set => Data[x, y] = value;

}

Matrix Exclude(int row, int col)

{

Matrix res = new Matrix(N);

for (int i = 0; i < N - 1; i++)

{

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

{

res[i, j] = i < row ? this[i, j] : this[i + 1, j];

}

}

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

{

for (int j = 0; j < N - 1; j++)

{

res[i, j] = j < col ? res[i, j] : res[i, j + 1];

}

}

Matrix resized = new Matrix(N - 1);

for (int i = 0; i < N - 1; i++)

{

for (int j = 0; j < N - 1; j++)

{

resized[i, j] = res[i, j];

}

}

return resized;

}

public double CalculateDeterminant()

{

if (N == 1) return this[0, 0];

if (N == 2) return this[0, 0] \* this[1, 1] - this[0, 1] \* this[1, 0];

double res = 0;

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

{

res += (j % 2 == 1 ? 1 : -1) \* this[1, j] \* Exclude(1, j).CalculateDeterminant();

}

return res;

}

public string CalculateDeterminantSteps()

{

if (N == 1) return $"{this[0, 0]} = {this[0, 0]}";

if (N == 2)

{

double a = this[0, 0];

double b = this[0, 1];

double c = this[1, 0];

double d = this[1, 1];

double det = a \* d - b \* c;

return $"{a} \* {d} - {b} \* {c} = {det}";

}

string steps = "";

int sign = 1;

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

{

Matrix minor = Exclude(0, j);

double det = minor.CalculateDeterminant();

steps += $"{this[0, j]} \* ({sign} \* {det}) + ";

sign = -sign;

}

steps = steps.Substring(0, steps.Length - 3); // remove the last " + "

double result = 0;

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

{

Matrix minor = Exclude(0, j);

double det = minor.CalculateDeterminant();

result += this[0, j] \* sign \* det;

sign = -sign;

}

steps += $"= {result}";

return steps;

}

Matrix Transpose()

{

Matrix res = new Matrix(N);

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

{

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

{

res[j, i] = this[i, j];

}

}

return res;

}

public Matrix Invert(out double det)

{

det = CalculateDeterminant();

if (det == 0) return null;

Matrix res = new Matrix(N);

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

{

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

{

res[i, j] = ((i + j) % 2 == 1 ? -1 : 1) \* Exclude(i, j).CalculateDeterminant() / det;

}

}

return res.Transpose();

}

}

}