

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности электронных
вычислительных систем (КИБЭВС)

СЛОЖЕНИЕ МАТРИЦ

Курсовая работа по дисциплине «Основы программирования»

Пояснительная записка

Студент гр. 712-2:

_____ В.Д. Маркушин

«__» _____ 2024 г.

Руководитель

Преподаватель кафедры

КИБЭВС

_____ Б.С. Лодонова

Оценка «__» _____ 2024 г.

РЕФЕРАТ

Курсовая работа содержит 56 страниц пояснительной записки, 31 рисунок, 8 источников, 4 приложения.

ЯЗЫК ПРОГРАММИРОВАНИЯ C#, WINDOWS FORMS, VISUAL STUDIO, SQL SERVER MANAGEMENT STUDIO, БАЗЫ ДАННЫХ, ПРОЕКТИРОВАНИЕ.

Программа: «Сложение матриц.».

Цель работы: необходимо предоставить пользовательский интерфейс для авторизации и регистрации пользователя, спроектировать и разработать программу «Сложение матриц», реализующую взаимодействие пользователя с калькулятором матриц.

Разработка программы проводилась на языке программирования C#.

Курсовая работа выполнена в текстовом редакторе Microsoft Word 2013.

Пояснительная записка оформлена согласно ОС ТУСУР 01-2021 [1].

the abstract

The course work contains 56 pages of explanatory notes, 31 drawings, 8 sources, 4 appendices.

C# PROGRAMMING LANGUAGE, WINDOWS FORMS, VISUAL STUDIO, SQL SERVER MANAGEMENT STUDIO, DATABASES, DESIGN.

Program: "Matrix Addition."

The purpose of the work: it is necessary to provide a user interface for user authorization and registration, design and develop a program "Matrix Addition", which implements the user's interaction with a matrix calculator.

The program was developed in the C# programming language.

The course work was done in the Microsoft Word 2013 text editor.

The explanatory note is issued in accordance with OS TUSUR 01-2021 [1].

Министерство науки и высшего образования РФ
ФГБОУ ВО «Томский государственный университет систем управления и
радиоэлектроники»

Кафедра комплексной информационной безопасности электронных
вычислительных систем (КИБЭВС)

УТВЕРЖДАЮ

Зав.кафедрой КИБЭВС

_____ А.А. Шелупанов

«__» _____ 2023 г.

Задание

на курсовую работу по дисциплине «Основы программирования»
студенту группы 712-2 факультета безопасности В.Д. Маркушину.

Тема работы: «Сложение матриц».

Цель работы: Получение навыков программирования на языке высокого
уровня и применения подхода объектно-ориентированного программирования
на практике.

Срок сдачи студентом законченной работы: «__» февраля 2024 г.

Исходные данные к работе: сложение матриц — это математическая
операция, позволяющая складывать две матрицы одинакового размера путём
поэлементного суммирования их значений. Эта процедура требует, чтобы
матрицы имели одинаковое количество строк и столбцов, и в результате
сложения каждый элемент одной матрицы складывается с соответствующим
элементом другой. Такое сложение обладает свойствами коммутативности и
ассоциативности, а также возможностью сложения с нулевой матрицей,
которая не влияет на результат. Сложение матриц широко применяется в
различных областях, включая линейную алгебру, информатику и инженерию,
и является основой для многих более сложных математических операций.

Задание: разработать программу, позволяющую пользователю с помощью клавиатуры и мыши взаимодействовать с калькулятором матриц на примере команд: сложить, вычесть, умножить.

Требования к используемым технологиям: язык программирования C#, .Net Framework, СУБД SQL Server Management Studio 19.

СОГЛАСОВАНО

Преподаватель кафедры КИБЭВС

_____ Б.С. Лодонова

«__» _____ 2023 г.

ПРИНЯЛ К ИСПОЛНЕНИЮ

Студент гр. 712-2

_____ В.Д. Маркушин

«__» _____ 2023 г

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности электронно-
вычислительных систем (КИБЭВС)

Сложение матриц
ТЕХНИЧЕСКОЕ ЗАДАНИЕ
На 11 листах

СОГЛАСОВАНО

Преподаватель кафедры КИБЭВС

_____ Б.С. Лодонова

« ____ » _____ 2023 г.

РАЗРАБОТЧИК

Студент гр. 712-2

_____ В.Д. Маркушин

« ____ » _____ 2023 г.

Томск 2023

1 Общие сведения

1.1 Полное наименование системы и её условное обозначение

Полное наименование системы: “Сложение матриц”.

1.2 Заказчик

Заказчиком является Томский государственный университет систем управления и радиоэлектроники, кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС).

1.3 Исполнитель

Исполнителем является студент группы 712-2 Маркушин Владислав Дмитриевич.

1.4 Основания для разработки

Основанием для разработки является задание на выполнение курсовой работы по дисциплине «Основы программирования» для студентов направления 10.03.01 «Информационная безопасность».

2 Назначение и цель создания системы

2.1 Назначение системы

Система предназначена для реализации калькулятора матриц, названий «Сложение матриц».

2.2 Цели создания системы

Целью разработки является создание пользовательского приложения «Сложение матриц».

3 Характеристика объектов автоматизации

3.1 Объект автоматизации

Объектом автоматизации является процесс последовательных действий необходимых для калькулятора «Сложение матриц».

4 Требования к системе

4.1 Перечень подсистемы, их назначение и основные характеристики

В системе предлагается выделить следующие функциональные подсистемы:

- 1) Подсистема графического интерфейса, для более комфортного использования;
- 2) Подсистема ввода данных для ввода матриц и выполнения последующих операций;
- 3) Подсистема обработки данных для выполнения операций с матрицами;
- 4) Подсистема вывода результатов;
- 5) Подсистема регистрации;
- 6) Подсистема авторизации.

4.2 Требования к надежности

При возникновении сбоев в аппаратном обеспечении, включая разряд аккумулятора устройства, информационная система восстанавливает свою работоспособность после устранения сбоев и корректного перезапуска аппаратного обеспечения (за исключением случаев повреждения рабочих носителей информации с исполняемым программным кодом).

4.3 Требования к безопасности

Все технические решения, использованные при создании системы, а также при определении требований к аппаратному обеспечению, соответствует действующим нормам и правилам техники безопасности,

пожарной безопасности, а также охраны окружающей среды, при эксплуатации или утилизации.

4.4 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению

Для эксплуатации разрабатываемой информационной системы необходимы следующие условия:

- 1) Компьютер под управлением операционной системы Windows 7 и новее, и MacOS;
- 2) Предустановленный .Net Framework 4.7.2
- 3) Питание компьютера от сети или батареи;
- 4) Наличие периферийных устройств – мышь, клавиатура.

4.5 Требования к защите информации от несанкционированного доступа

Доступ к работе с интерфейсом системы имеют только авторизованные пользователи.

4.6 Требования к функциям разработчика

Роль разработчика заключается в обновлении и пополнении системы новыми функциями, а также исправление возможных ошибок в функционировании системы.

4.7 Требования к функциям пользователя

Пользователь может использовать все функции, которыми обладает система.

4.8 Описание процессов и функций работы с системой

Процессы и функции, выполняемые при эксплуатации системы, приведены в разбивке по подсистемам: подсистема графического интерфейса, подсистема ввода данных для ввода матриц и выполнения последующих операций, подсистема обработки данных для выполнения операций с матрицами, подсистема вывода результата. Процессы, реализованные под управлением различных подсистем, реализуются на основе системных процедур, которые являются составной частью функции системы. Системные процедуры группируются в соответствии с их назначением:

- 1) Графический интерфейс пользователя;
- 2) Проверка столбцов/строк;
- 3) Ввод данных;
- 4) Обработка данных;
- 5) Вывод данных;
- 6) Регистрация пользователя;
- 7) Авторизация пользователя.

4.9 Требования к информационному обеспечению системы

Компоненты системы должны активно взаимодействовать с системой управления базой данных (СУБД). Обмен информацией с СУБД должен происходить автоматически. Уровень хранения данных в системе должен быть построен на основе реляционных или объектно-реляционных СУБД. Доступ к данным должен быть предоставлен только авторизованным пользователям.

4.10 Требования к программному обеспечению

- 1) ОС Windows 7, 8, 10 и 11 или MacOS;
- 2) Язык программирования C#;
- 3) .Net Framework 4.7.2;
- 4) Установлено ПО.
- 5) СУБД MySQL.

5 Порядок контроля и приемки системы

5.1 Перечень этапов испытаний и проверок

Этапы испытаний подразделяются на предварительные и приемочные. Предварительные испытания проводятся на стадии тестирования разработчиком. Приемочные испытания проводятся во время сдачи проекта разработчиком совместно с заказчиком. Все подсистемы испытываются одновременно на корректность взаимодействия подсистем, влияние подсистем друг на друга, то есть испытания проводятся комплексно. Во время приемочных испытаний оценивается:

- 1) Полнота и качество реализации функций, указанных в настоящем техническом задании;
- 2) Демонстрация объектно-ориентированного подхода при реализации функций, указанных в настоящем техническом задании;
- 3) Выполнение каждого требования, относящегося к интерфейсу системы;
- 4) Полнота действий доступных пользователю:
 - Ввод данных пользователя;
 - Возможность начала пользования калькулятором;
 - Выход авторизованного пользователя из приложения.

Приемка результатов должна осуществляться в сроки, установленные заказчиком. Результаты проектирования системы и её тестирования предоставляются в электронном виде с помощью ЭИОС sdo.tusur.ru.

5.2 Общие требования к приемке работы

Приемка осуществляется представителями Заказчика и Исполнителя. Все создаваемые в рамках настоящей работы программные изделия

передаются Заказчику, как в виде готовых модулей, так и в виде исходных кодов, представляемых в сохраненных программах С#.

6 Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие

Для обеспечения готовности объекта к вводу системы в действие провести комплекс мероприятий:

- 1) Загрузка файлов приложения;
- 2) Проведение предварительных испытаний;
- 3) Проверка приемочных испытаний.

7 Требования к документированию

Состав программной документации:

- 1) Задание на курсовую работу
- 2) Техническое задание (ТЗ);
- 3) Пояснительная записка (ПЗ);
- 4) Документация к системе в электронном виде;

Документация должна быть оформлена с использованием:

- 1) ГОСТ 34.602-89;
- 2) ОС ТУСУР 01-2021 для технического задания;
- 3) ОС ТУСУР 01-2021 для пояснительной записки

Содержание

| | |
|---|----|
| Введение..... | 19 |
| 1 ОБЗОР ТЕМЫ..... | 20 |
| 2 ПРОЕКТИРОВАНИЕ РАЗРАБАТЫВАЕМОГО ПРИЛОЖЕНИЯ..... | 21 |
| 2.1 Обоснование выбранных технологий..... | 21 |
| 2.2 Описание алгоритмов приложения..... | 22 |
| 2.3 Определение временной сложности..... | 27 |
| 2.4 Проектирование структуры программы..... | 27 |
| 3 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ..... | 38 |
| 4 ТЕСТИРОВАНИЕ..... | 39 |
| 4.1 Функциональное тестирование..... | 43 |
| 4.2 Ручное тестирование..... | 44 |
| Заключение..... | 51 |
| Список использованных источников..... | 52 |
| Приложение А (Обязательное)..... | 53 |
| Приложение Б (Обязательное)..... | 54 |
| Приложение В (Обязательное)..... | 55 |
| Приложение Г (Обязательное)..... | 56 |

Введение

Целью данной курсовой работы является получение навыков разработки алгоритмов и их реализации на языке программирования высокого уровня в виде программного обеспечения для персонального компьютера.

1 ОБЗОР ТЕМЫ

Операции над матрицами, включающие сложение, вычитание и умножение, являются фундаментальными в линейной алгебре и находят широкое применение в различных областях науки и инженерии. Эти операции позволяют выполнять комплексные вычисления в таких сферах, как компьютерное моделирование, статистика, физика и экономика.

Сложение матриц выполняется путём поэлементного суммирования соответствующих элементов двух матриц одинакового размера. Это означает, что матрицы должны иметь одинаковое количество строк и столбцов. Эта операция обладает свойствами коммутативности и ассоциативности.

Вычитание матриц аналогично сложению и также требует, чтобы обе матрицы были одного размера. При вычитании соответствующие элементы одной матрицы вычитаются из соответствующих элементов другой.

Умножение матриц, в отличие от сложения и вычитания, более сложная операция. Оно не всегда возможно и требует, чтобы количество столбцов в первой матрице было равно количеству строк во второй.

Эти математические операции над матрицами не только позволяют решать комплексные системы уравнений, но и лежат в основе многих алгоритмов машинного обучения, обработки изображений и компьютерной графики.

ПРОЕКТИРОВАНИЕ РАЗРАБАТЫВАЕМОГО ПРИЛОЖЕНИЯ

2.1 Обоснование выбранных технологий

Перед тем, как начать реализовывать выбранную тему, необходимо определиться с набором технологий, с помощью которых будет написано приложение. Необходимо определиться со следующим набором: язык программирования, среда разработки, СУБД.

Приложение «Сложение матриц» будет выполнено на объектно-ориентированном языке программирования C# в интегрированной среде разработки Visual Studio.

Объектно-ориентированных языков программирования на сегодняшний момент достаточно много. Но остановимся на двух языках программирования C# и Python. Python – универсальный язык и используется во многих сферах. Python пригодится в создании компьютерных и мобильных приложений, его применяют в работе с большим объемом информации, при разработке web-сайтов и других разнообразных проектов, используют в машинном обучении. Язык C# используют для разработки самых разнообразных программ, включая desktop - и веб-приложения, мобильные приложения, к тому же, данный язык программирования уже изучался во втором семестре первого курса на дисциплине «Основы программирования».

В качестве среды разработки была выбрана программа «Visual Studio». Visual Studio предоставляет разработчикам широкие возможности среды разработки для эффективной и совместной разработки высококачественного кода [3].

При выборе базы данных внимание больше всего привлекла СУБД (система управления базами данных) SQL Server Management Studio (далее – SSMS). Среда SSMS предоставляет собой единую служебную программу, которая сочетает в себе обширную группу графических инструментов с

разными многофункциональными редакторами скриптов для доступа к SQL Server для разработчиков и администраторов баз данных всех уровней [7].

2.2 Описание алгоритмов приложения

Алгоритм А «Основное окно игры»:

A0. Начало.

A1. Создать класс Form1 в пространстве имен MatrixC_09.01.

A2. Установить Form1 как частичную форму (public partial class Form1: Form).

A3. Создать обработчики событий для кнопок (button1_Click_1, button2_Click, button3_Click, button4_Click, button5_Click, button6_Click).

A4. В обработчике button1_Click_1, считать размер матрицы из textBox1.

A5. Инициализировать dataGridView1 с указанным количеством строк и столбцов.

A6. В обработчике button2_Click, считать размер матрицы из textBox2.

A7. Инициализировать dataGridView2 с указанным количеством строк и столбцов.

A8. В обработчике button3_Click, проверить, равны ли тексты в textBox1 и textBox2.

A9. Если тексты в textBox1 и textBox2 равны, перейти к шагу A10. Если нет, показать сообщение об ошибке и перейти к шагу A24.

A10. Выполнить вычитание матриц: считать данные из dataGridView1 и dataGridView2.

A11. Вычесть данные матриц и отобразить результат в dataGridView3.

A12. В обработчике button4_Click, проверить, равны ли тексты в textBox1 и textBox2.

A13. Если тексты в textBox1 и textBox2 равны, перейти к шагу A14. Если нет, показать сообщение об ошибке и перейти к шагу A24.

A14. Выполнить сложение матриц: считать данные из dataGridView1 и dataGridView2.

- A15. Сложить данные матриц и отобразить результат в dataGridView3.
- A16. В обработчике button5_Click, проверить, равны ли тексты в textBox1 и textBox2.
- A17. Если тексты в textBox1 и textBox2 равны, перейти к шагу A18. Если нет, показать сообщение об ошибке и перейти к шагу A24.
- A18. Выполнить умножение матриц: считать данные из dataGridView1 и dataGridView2.
- A19. Умножить данные матриц и отобразить результат в dataGridView3.
- A20. В обработчике button6_Click, выполнить закрытие приложения.
- A21. Обработка Исключений: - A21.1. В каждом из обработчиков событий кнопок для операций над матрицами, использовать блок try-catch для обработки исключений.
- A22. Показать сообщение об ошибке при исключении в процессе выполнения операций над матрицами.
- A23. Показать сообщение об ошибке размерности матриц, если они не совпадают.
- A24. Конец.

Блок-схема к алгоритму А представлена в приложении А.

Алгоритм В «Регистрация нового пользователя»:

- B0. Начало.
- B1. Создать класс Form3 в пространстве имен MatrixC_09.01.
- B2. Установить Form3 как частичную форму (public partial class Form3 : Form).
- B3. Создать обработчики событий для компонентов формы (label2_Click, textBox1_TextChanged, buttonRegister_Click, button6_Click, button1_Click).
- B4. В обработчике buttonRegister_Click, начать регистрацию пользователя.
- B5. Проверить, пустое ли поле логина (loginField.Text == ""). Если да, показать сообщение "Введите логин" и вернуться к шагу A4.
- B6. Проверить, пустое ли поле пароля (passField.Text == ""). Если да, показать сообщение "Введите пароль" и вернуться к шагу A4.

- B7. Вызвать метод `checkUser()` для проверки наличия пользователя с таким же логином.
- B8. Если `checkUser()` возвращает `true` (пользователь существует), вернуться к шагу A4.
- B9. Создать экземпляр класса `DB` для работы с базой данных.
- B10. Подготовить SQL-команду для вставки нового пользователя в таблицу `users`.
- B11. Добавить параметры (`@login` и `@pass`) в SQL-команду.
- B12. Открыть соединение с базой данных.
- B13. Выполнить SQL-команду (вставка пользователя).
- B14. Если выполнение команды успешно (возвращает 1), показать сообщение "Аккаунт был создан". В противном случае показать сообщение "Аккаунт не был создан".
- B15. Закрыть соединение с базой данных.
- B17. В обработчике `button6_Click`, выполнить закрытие приложения.
- B18. В обработчике `button1_Click`, скрыть текущую форму (`Form3`) и открыть новую форму (`Form2`).
- B19. Конец.

Блок-схема к алгоритму В представлена в приложении Б.

Алгоритм С к функции "`checkuser()`":

- C0. Начало.
- C1. Создать экземпляр класса `DB` для работы с базой данных.
- C2. Создать объект `DataTable` для хранения результатов запроса.
- C3. Создать экземпляр `MySqlDataAdapter` для выполнения запроса к базе данных.
- C4. Подготовить SQL-команду для выборки пользователя с данным логином: "`SELECT * FROM users WHERE login = @uL`".
- C5. Добавить параметр `@uL` в SQL-команду, присвоив ему значение `loginField.Text`.

- C6. Установить подготовленную SQL-команду как SelectCommand для MySqlConnection.
- C7. Выполнить команду, используя MySqlConnection, и заполнить полученными данными объект DataTable.
- C8. Проверить количество строк в DataTable: Если количество строк больше нуля (пользователь с таким логином уже существует), перейти к шагу C9. Если количество строк равно нулю (пользователь не найден), перейти к шагу C10.
- C9. Показать сообщение "Такой логин уже существует" и вернуть true.
- C10. Вернуть false.
- C11. Конец.

Блок-схема к алгоритму C представлена в приложении В.

Алгоритм D «Авторизация пользователя»:

- D0. Начало.
- D1. Создать класс Form2 в пространстве имен MatrixC_09.01.
- D2. Установить Form2 как частичную форму (public partial class Form2 : Form).
- D3. В конструкторе класса Form2, инициализировать компоненты и настроить размер поля passField.
- D4. Создать обработчики событий для кнопок (buttonLogin_Click, button1_Click, button6_Click) и других компонентов (label3_Click).
- D5. В обработчике buttonLogin_Click, начать процедуру входа в систему.
- D6. Проверить, пустое ли поле логина (loginField.Text == ""). Если да, показать сообщение "Введите логин" и вернуться к шагу A5.
- D7. Проверить, пустое ли поле пароля (passField.Text == ""). Если да, показать сообщение "Введите пароль" и вернуться к шагу A5.
- D8. Присвоить переменным loginUser и passUser значения из полей ввода логина и пароля.
- D9. Создать экземпляр класса DB для работы с базой данных.
- D10. Создать объект DataTable для хранения результатов запроса.
- D11. Создать экземпляр MySqlConnection для выполнения запроса к базе данных.

- D12. Подготовить SQL-команду для выборки пользователя с данным логином и паролем: "SELECT * FROM users WHERE login = @uL AND pass = @uP".
- D13. Добавить параметры @uL и @uP в SQL-команду, присвоив им значения переменных loginUser и passUser.
- D14. Установить подготовленную SQL-команду как SelectCommand для MySqlDataAdapter.
- D15. Выполнить команду и заполнить полученными данными объект DataTable.
- D16. Проверить количество строк в DataTable: Если количество строк больше нуля (пользователь найден), перейти к шагу D17. Если количество строк равно нулю (пользователь не найден), показать сообщение "Неправильный логин или пароль" и вернуться к шагу D5.
- D17. Скрыть текущую форму (Form2) и открыть главную форму (Form1).
- D18. В обработчике button1_Click, скрыть текущую форму (Form2) и открыть форму регистрации (Form3).
- D19. В обработчике button6_Click, выполнить закрытие приложения.
- D20. Конец.

Блок-схема к алгоритму D представлена в приложении Д.

2.3 Определение временной сложности

Алгоритмическая сложность (Вычислительная сложность) – понятие, обозначающее функцию зависимости объема работы алгоритма от размера обрабатываемых данных [5].

Алгоритмическая сложность приложения "Сложение матриц" зависит от различных факторов, таких как: количество, тип действий, способ хранения данных и их обработки.

В целом, алгоритмическая сложность методов варьируется от $O(n^2)$ для сложения и вычитания матриц до $O(n^3)$ для умножения матриц.

2.4 Проектирование структуры программы приложения

Перед написанием программного кода, была составлена UML-диаграмма (диаграмма классов). С помощью UML-диаграммы можно отразить визуальное представление всех классов и их взаимосвязь друг с другом. Ниже представлена диаграмма классов (рисунок 2.4.1) для написания программы к приложению «Сложение матриц».

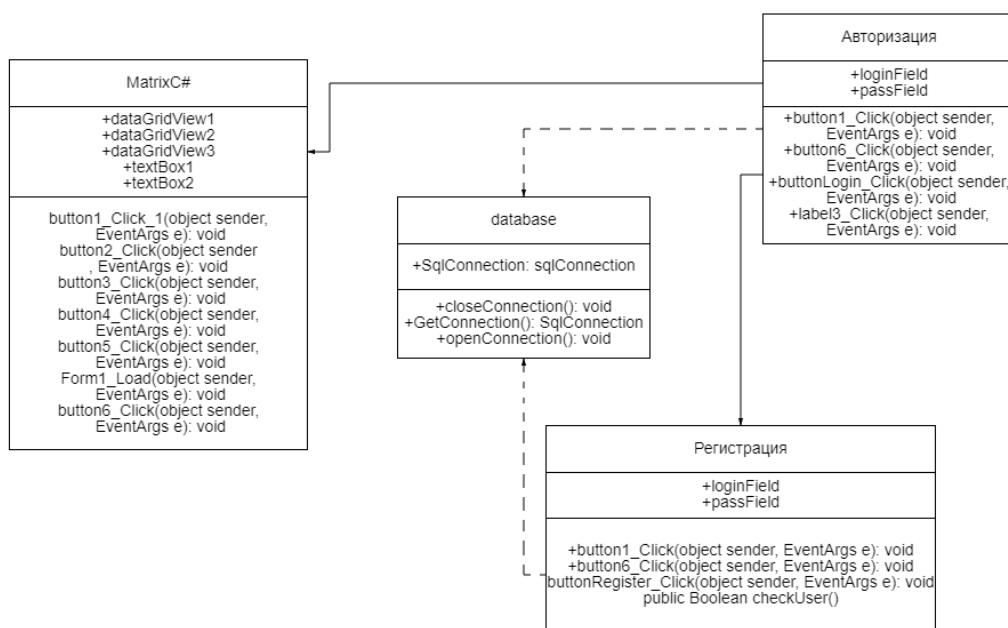


Рисунок 2.4.1 – UML-диаграмма

2 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

Для выполнения поставленной задачи, отраженной в техническом задании, были созданы следующие формы:

- Форма «Авторизация» – оконная Windows-форма, которая отражает и обеспечивает интерфейс пользователя с программой. Данная форма предназначена для авторизации уже существующего пользователя или же для перехода к регистрации ещё не зарегистрированного пользователя.
- Форма «Регистрация» – оконная Windows-форма, которая обеспечивает интерфейс пользователя с программой. Форма предназначена для регистрации нового пользователя.
- Форма основного окна «Сложение матриц» – оконная Windows-форма, предоставляющая интерфейс взаимодействия пользователя с программой. Предназначена для взаимодействия пользователя с калькулятором матриц с помощью мыши и клавиатуры.

Кроме того, был разработан класс «База данных» – класс, обеспечивающий инициализацию базы данных, создающий таблицы для хранения данных обо всех пользователях приложения, имеющий методы проверки уже существующего пользователя и исполняющий функцию регистрации нового пользователя.

Полный код программы представлен в репозитории.

Рассмотрим основной функционал программы.

При запуске программы открывается форма «Авторизация» (рисунок 3.1).

Авторизация

Введите логин

Введите пароль

Войти

Регистрация

Выйти

Рисунок 3.1 – Форма «Авторизация»

Заметим, что на форме присутствуют два текстовых поля для ввода логина и пароля. Ниже расположена кнопка «Войти», при нажатии которой, будет выполнен следующий фрагмент кода программы:

```
private void buttonLogin_Click(object sender, EventArgs e)
{
    if (loginField.Text == "")
    {
        MessageBox.Show("Введите логин");
        return;
    }

    if (passField.Text == "")
    {
        MessageBox.Show("Введите пароль");
        return;
    }

    String loginUser = loginField.Text;
    String passUser = passField.Text;

    DB db = new DB();

    DataTable table = new DataTable();

    MySqlDataAdapter adapter = new MySqlDataAdapter();
```

```

        MySqlCommand command = new MySqlCommand("SELECT * FROM
`users` WHERE `login` = @uL AND `pass` = @uP", db.GetConnection());
        command.Parameters.Add("@uL", MySqlDbType.VarChar).Value =
loginUser;
        command.Parameters.Add("@uP", MySqlDbType.VarChar).Value =
passUser;

        adapter.SelectCommand = command;
        adapter.Fill(table);

        if (table.Rows.Count > 0)
        {
            this.Hide();
            Form1 mainForm = new Form1();
            mainForm.Show();
        }

        else
            MessageBox.Show("Неправильный логин или пароль");
    }

```

Данная обработка кода обращается к базе данных, и, в случае нахождения такого пользователя логина и пароля в таблице, форма «Авторизация» закрывается и открывается форма основного окна «Сложение матриц» (рисунок 3.2).

Рисунок 3.2 – Основное окно «Сложение матриц»

Если же пользователь еще не зарегистрирован, в форме «Авторизация» (рисунок 3.1) есть кнопка «Регистрация», при нажатии на которую будет отработан следующий фрагмент кода:

```
private void button1_Click(object sender, EventArgs e)
{
    this.Hide();
    Form3 registerForm = new Form3();
    registerForm.Show();
}
```

Данный код позволяет открыть форму «Регистрация» (рисунок 3.3).

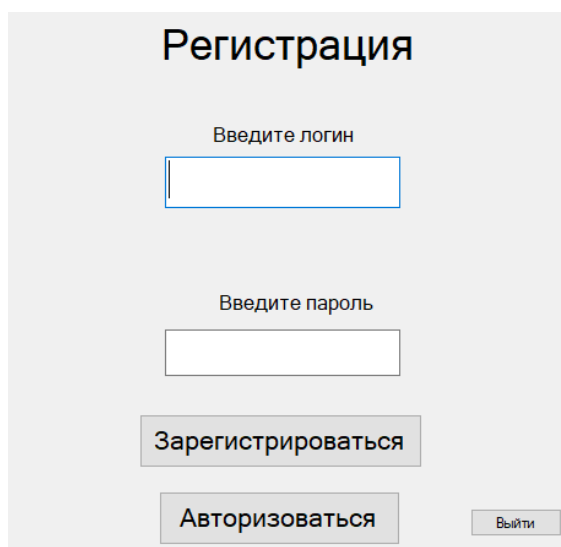


Рисунок 3.3 – Форма «Регистрация»

Высветившаяся форма «Регистрации» предлагает придумать логин и пароль и ввести их в текстовые поля соответственно. Ниже располагается кнопка «Зарегистрироваться», при нажатии на которую будет отработан следующий ряд событий:

```
private void buttonRegister_Click(object sender, EventArgs e)
{
    if (loginField.Text == "")
    {
        MessageBox.Show("Введите логин");
    }
}
```

```

        return;
    }
    if (passField.Text == "")
    {
        MessageBox.Show("Введите пароль");
        return;
    }
    if (checkUser())
        return;

    DB db = new DB();
    MySqlCommand command = new MySqlCommand("INSERT
    INTO`users` (`login`, `pass`) VALUES (@login, @pass)", db.GetConnection());
    command.Parameters.Add("@login", MySqlDbType.VarChar).Value
    = loginField.Text;
    command.Parameters.Add("@pass", MySqlDbType.VarChar).Value
    = passField.Text;
    db.openConnection();
    if (command.ExecuteNonQuery() == 1)
        MessageBox.Show("Аккаунт был создан");
    else
        MessageBox.Show("Аккаунт не был создан");
    db.closeConnection();
}

```

После успешной регистрации необходимо перейти на форму «Авторизация», с помощью кнопки «Авторизоваться». Теперь новый пользователь может ввести свои логин и пароль, и форма пропустит его к калькулятору.

Вернемся к основному окну (рисунок 3.2). Здесь появляется сам калькулятор. На форме расположены две таблицы, в которые необходимо вводить сами матрицы, два текстовых окна для ввода размерностей матриц, а

также выбор действий. При нажатии на кнопку «Задать» будет отработывать следующее событие:

```
private void button1_Click_1(object sender, EventArgs e)
{
    int n;
    int.TryParse(textBox1.Text, out n);
    dataGridView1.RowCount = n;
    dataGridView1.ColumnCount = n;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            dataGridView1.Columns[j].Width = 60;
        }
    }
}
```

После нажатия на кнопку «Задать», на главном окне появляются таблицы для ввода матриц (рисунок 3.4).

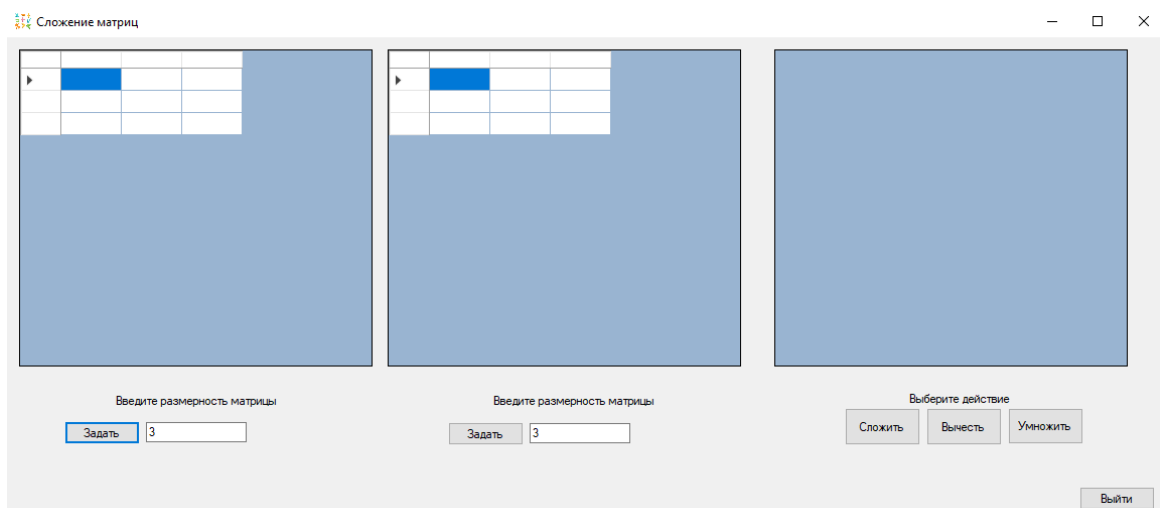


Рисунок 3.4 – Ситуация после нажатия на кнопку «Задать»

Если нажать на кнопку «Сложить» будет обработан следующий фрагмент кода:

```
private void button4_Click(object sender, EventArgs e)
{
    try
    {
        if (textBox1.Text == textBox2.Text)
        {
            int n;
            int.TryParse(textBox2.Text, out n);
            dataGridView3.RowCount = n;
            dataGridView3.ColumnCount = n;
            int[,] a = new int[n, n];
            int[,] b = new int[n, n];
            int[,] c = new int[n, n];
            for (int i = 0; i < n; i++)
                for (int j = 0; j < n; j++)
                    a[i, j] = Convert.ToInt32(dataGridView1[i, j].Value);
            for (int i = 0; i < n; i++)
                for (int j = 0; j < n; j++)
                    b[i, j] = Convert.ToInt32(dataGridView2[i, j].Value);
            for (int i = 0; i < n; i++)
                for (int j = 0; j < n; j++)
                {
                    c[i, j] = a[i, j] + b[i, j];
                    dataGridView3[i, j].Value = c[i, j];
                }
        }
    }
    else
```

```

        MessageBox.Show("Складывать матрицы можно только
одинаковой размерности.", "Ошибка!");
    }
    catch (Exception)
    {
        MessageBox.Show("Непредвиденная ошибка");
    }
}

```

После отработки кода основное окно будет выглядеть следующим образом (рисунок 3.5). В таблице результата отобразилась конечная матрица.

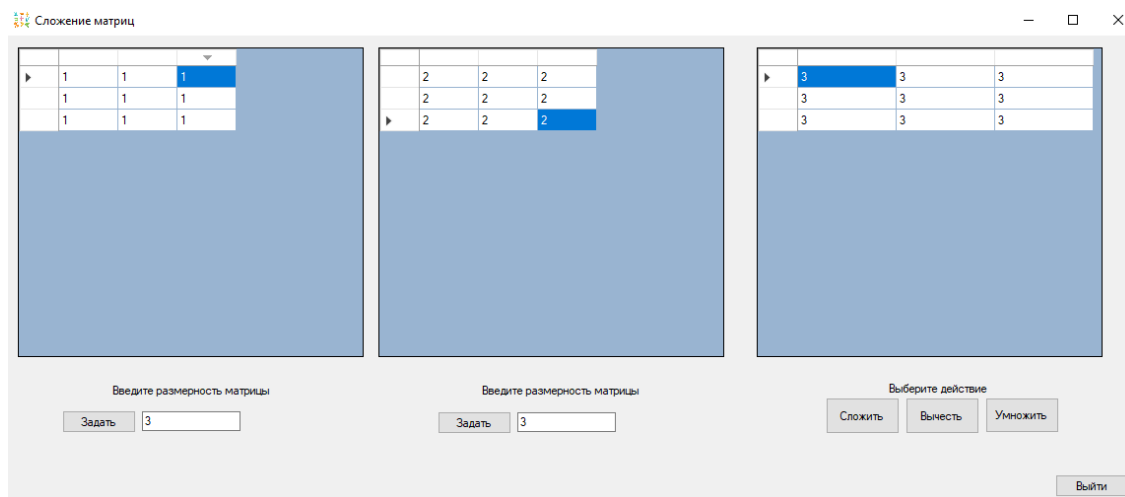


Рисунок 3.5 – Окно игры после нажатия на кнопку «Сложить»

Следующая кнопка «Вычесть». Когда пользователь нажмет на нее, программа приведет в действие фрагмент кода:

```

private void button3_Click(object sender, EventArgs e)
{
    try
    {
        if (textBox1.Text == textBox2.Text)
        {
            int n;
            int.TryParse(textBox2.Text, out n);

```

```

dataGridView3.RowCount = n;
dataGridView3.ColumnCount = n;
int[,] a = new int[n, n];
int[,] b = new int[n, n];
int[,] c = new int[n, n];
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        a[i, j] = Convert.ToInt32(dataGridView1[i, j].Value);
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        b[i, j] = Convert.ToInt32(dataGridView2[i, j].Value);
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
    {
        c[i, j] = a[i, j] - b[i, j];
        dataGridView3[i, j].Value = c[i, j];
    }
}
else
    MessageBox.Show("Вычитать матрицы можно только
одинаковой размерности.", "Ошибка!");
}
catch (Exception)
{
    MessageBox.Show("Непредвиденная ошибка");
}

}

```

После отработки кода основное окно будет выглядеть следующим образом (рисунок 3.6). В таблице результата отобразилась конечная матрица.

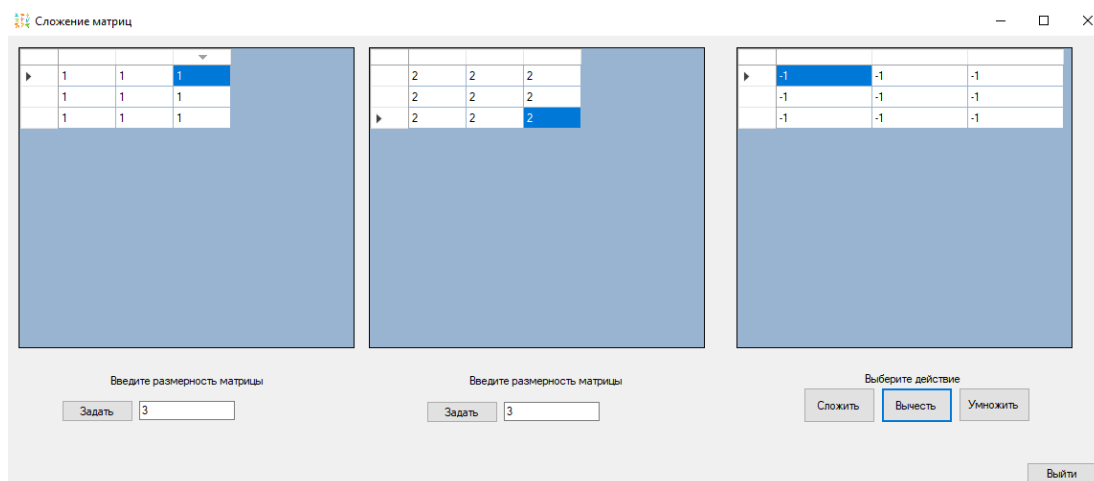


Рисунок 3.6 – Окно игры после нажатия на кнопку «Вычислить»

Когда будет нажата кнопка «Умножить», будет отработан следующий фрагмент кода:

```
private void button5_Click(object sender, EventArgs e)
{
    if (textBox1.Text == textBox2.Text)
    {
        int n, v;
        int.TryParse(textBox2.Text, out n);
        dataGridView3.RowCount = n;
        dataGridView3.ColumnCount = n;
        int[,] a = new int[n, n];
        int[,] b = new int[n, n];
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                a[j, i] = Convert.ToInt32(dataGridView1[i, j].Value);
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                b[j, i] = Convert.ToInt32(dataGridView2[i, j].Value);
        for (int i = 0; i < n; i++)
        {

```

```

        for (int j = 0; j < n; j++)
        {
            v = 0;
            for (int r = 0; r < n; r++)
                v += a[i, r] * b[r, j];
            dataGridView3[i, j].Value = v;
        }
    }
}
else
    MessageBox.Show("Умножать матрицы можно только
одинаковой размерности.", "Ошибка!");
}

```

После отработки кода основное окно будет выглядеть следующим образом (рисунок 3.7). В таблице результата отобразилась конечная матрица.

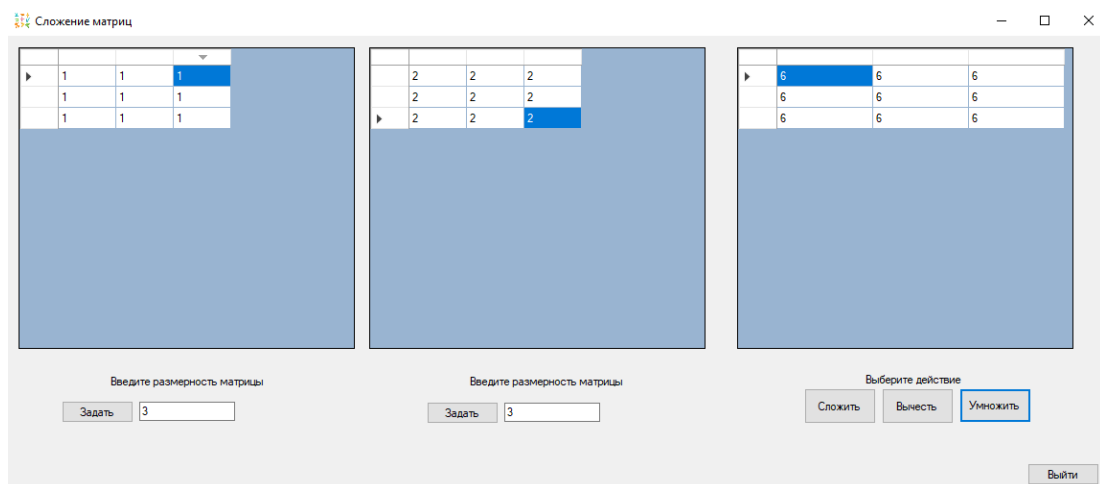


Рисунок 3.7 – Окно игры после нажатия на кнопку «Умножить»

При нажатии на кнопку «Выйти» программа закрывается.

4 ТЕСТИРОВАНИЕ

В этом разделе будет представлено ручное и функциональное (NUnit) тестирование пользовательского интерфейса. Перед тем, как выполнять ручное тестирование, были составлены: тест план, чек-лист и Mind-Map.

Тест план:

1. Тестируется desktop приложение «Сложение матриц» в рамках написания курсовой работы по дисциплине «Основы программирования»;
2. Операционная система: Windows;
3. В приложении необходимо протестировать следующие функции:
 - Авторизация пользователя;
 - Регистрация пользователя;
 - Работа кнопок основного окна приложения.
4. Тестирование проводится в течение нескольких дней, по результатам тестирования предоставляется подробная отчетность и все тестовые документы, созданные в рамках проделанной работы;
5. Критерий начала тестирования – окончание составления нужной документации;
6. Критерий окончания тестирования – проведение всех основных действий пункта 7;
7. Необходимо провести ручное тестирование основного функционала согласно пункту 3. Такой тест позволит точно убедиться в работоспособности приложения. Для тестирования требуется написать тест-кейсы для функций, относящихся к основному функционалу. Также необходимо провести функциональное тестирование.

Тестовая стратегия – это то, как будет тестироваться продукт (рисунок 4.1.).

| | | | | | | | | |
|--------------------|---|---------------------------------------|--|--------|----------------------|--|--------------------|--------|
| Название проекта | | Introduction: | Функциональное решение для реализации приложения "Сложение матриц" | | | | | |
| Сложение матриц | | Objective: | Предоставление пользователям возможности управлять приложением при помощи мыши и клавиатуры | | | | | |
| Дата завершения | 12.01.2023 | Principles: | Выполнить работу в срок. Разобраться в реализации. Качественно выполнить проект и показать достойный результат | | | | | |
| Дата сдачи проекта | 03.03.2023 | Entry Conditions(for test execution): | | People | | | | |
| In Scope | | 1 | Поставлена задача в рамках обучения | | 1 | Менеджер проекта | Маркушин Владислав | |
| 1 | Функциональное тестирование | 2 | Собственное желание | | 2 | Бизнес-аналитик | Маркушин Владислав | |
| 2 | Ручное тестирование | Exit Conditions(for test execution): | | 3 | Тестировщик | Маркушин Владислав | | |
| Out of Score | | 1 | При тестировании не найдено крит.ошибок | | Test Environment | | | |
| 1 | Отсутствие прочих видов тестирования, т.к. представленных вполне достаточно | 2 | Приложение работает на ОС Windows 10 | | 1 | Предполагается, что большинство пользователей будут работать в Windows-системах | | |
| | | 3 | Срок на выполнение задания подошел к концу | | 2 | Предполагается, что большинство пользователей имеют компьютерную мышь и клавиатуру | | |
| | | 4 | Все, включенное в план тестирования, завершено | | Timescales | | | |
| | | Risks | | 1 | Планирование проекта | | 3 дня | |
| | | 1 | Отключение ПК по техническим причинам | | 2 | Написание тест-кейсов, чек-листов | | 1 день |
| | | 2 | Выход из строя компьютерной мыши или клавиатуры | | 3 | Выполнение тестов | | 2 дня |
| | | | | 4 | Подведение итогов | | 2 дня | |

Рисунок 4.1 – Тестовая стратегия

Тест-кейс — это форма записи проверки, которую проводит тестировщик. По сути, это алгоритм действий, по которому предполагается тестировать уже написанную программу. В нём подробно прописаны шаги, которые нужно сделать для подготовки к тесту, сама проверка и ожидаемый результат [10]. Тест-кейс представлен на рисунках 4.2-4.4. Интеллект-карта (Mind-Map) отображена на рисунке 4.5. Чек-лист представлен на рисунке 4.6.

| Проверка авторизации пользователя | | |
|-----------------------------------|---------------------------|---|
| № | Действие | Результат |
| 1 | Ввести логин | В поле ввода "Логин" отображается admin |
| 2 | Ввести пароль | В поле ввода "Пароль" отображается 123 |
| 3 | Войти через кнопку "Вход" | Открывается окно приложения |

Рисунок 4.2 – Тест-кейс «Проверка авторизации пользователя»

| Проверка регистрации пользователя | | |
|-----------------------------------|--|---|
| № | Действие | Результат |
| 1 | Открыть регистрацию через кнопку "Регистрация" | Открывается окно регистрации |
| 2 | Ввести логин | В поле ввода "Логин" отображается admin |
| 3 | Ввести пароль | В поле ввода "Пароль" отображается 123 |
| 4 | Зарегистрироваться | Логин и пароль пользователя сохранены в базу данных |

Рисунок 4.3 – Тест-кейс «Проверка регистрации пользователя»

| Проверка работы приложения | | |
|----------------------------|--|--|
| № | Действие | Результат |
| 1 | Ввести в текстовые поля размерность матриц, после чего нажать на кнопку "Задать" | На таблицах появляются свободные окна для ввода элементов в матрицы |
| 2 | Нажать на кнопку "Сложить" | Матрицы из двух таблиц складываются, результат выводится в третью таблицу |
| 3 | Нажать на кнопку "Вычесть" | Матрицы из двух таблиц вычитаются, результат выводится в третью таблицу |
| 4 | Нажать на кнопку "Умножить" | Матрицы из двух таблиц перемножаются, результат выводится в третью таблицу |
| 5 | Закрыть программу через кнопку "Выйти" | Закрывается основное окно, выход из программы |

Рисунок 4.4 – Тест-кейс «Проверка работы игры»

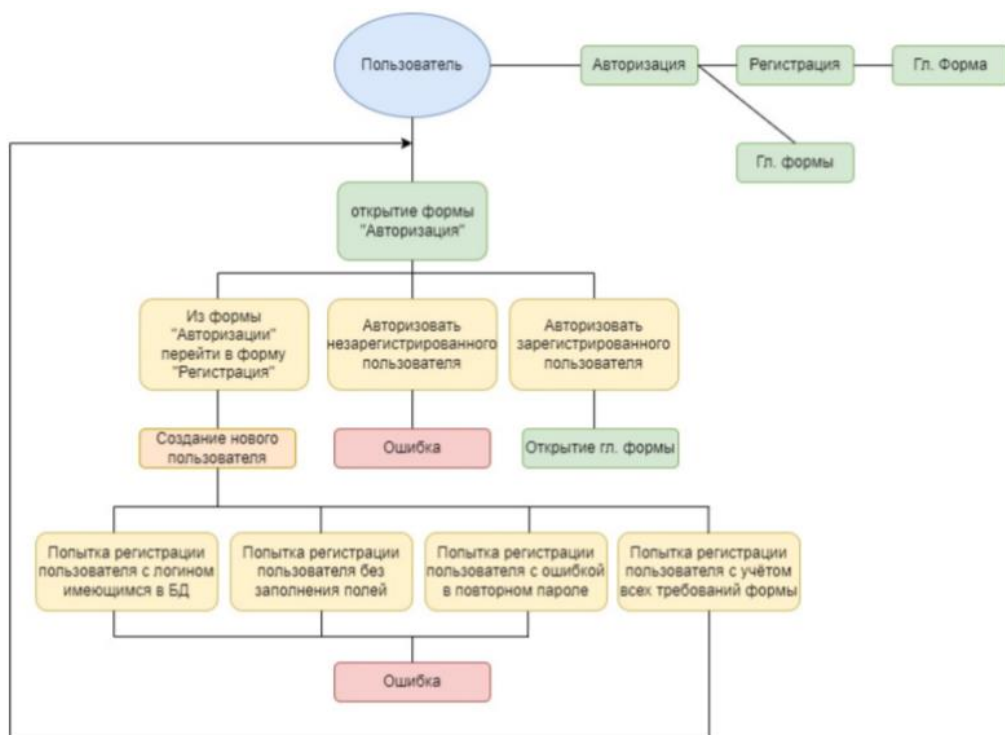


Рисунок 4.5 – Интеллект-карта

| Проверка | Результат | | |
|-----------------------------|------------|-----------|------------|
| | Windows 10 | Windows 7 | Windows 11 |
| функция | | | |
| Кнопка "Вход" | ОК | ОК | ОК |
| Ввод пароля в авторизации | ОК | ОК | ОК |
| Ввод логина в авторизации | ОК | ОК | ОК |
| Ввод пароля в регистрации | ОК | ОК | ОК |
| Ввод логина в авторизации | ОК | ОК | ОК |
| Кнопка "Зарегистрироваться" | ОК | ОК | ОК |
| Кнопка "Задать" | ОК | ОК | ОК |
| Кнопка "Сложить" | ОК | ОК | ОК |
| Кнопка "Умножить" | ОК | ОК | ОК |
| Кнопка "Выйти" | ОК | ОК | ОК |

Рисунок 4.6 – Чек-лист

4.1 Функциональное тестирование

Согласно заданию нужно протестировать функционал основной игры, а именно, правильно ли работают кнопки и свою ли задачу они выполняют. Для этого данный класс переводим в режим доступа «Public». Кнопки переводим также в общедоступный режим. На рисунке 4.4.1 представлено успешное прохождение автоматических тестов, написанных с использованием библиотека NUnit.

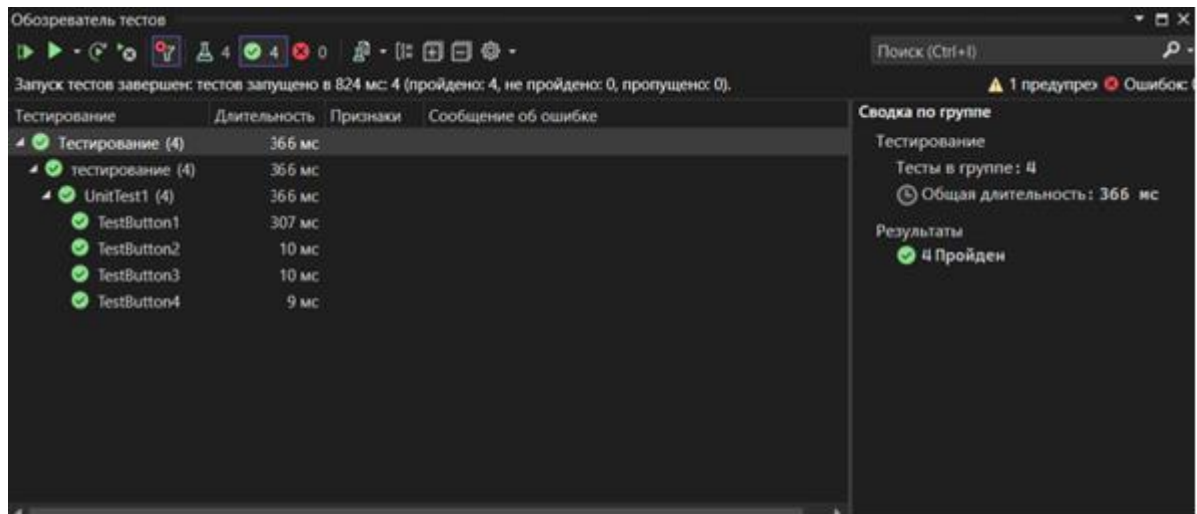


Рисунок 4.1.1 – Функциональное тестирование

4.2 Ручное тестирование

На данном этапе будем самостоятельно поэтапно проверять работоспособность программы с намеренным выполнением ошибок. На данном этапе будут проверяться формы «Авторизация» и «Регистрация», а также главное окно с калькулятором. Подробное описание каждой кнопки было рассмотрено в разделе «Реализация приложения» (стр.30).

Начнем с формы «Авторизация». Попробуем авторизоваться и войти в игру без каких-либо данных (рисунок 4.2.1). Следом заполним только одно из двух полей (рисунок 4.2.2) и потом попробуем авторизовать не зарегистрированного пользователя (рисунок 4.2.3).

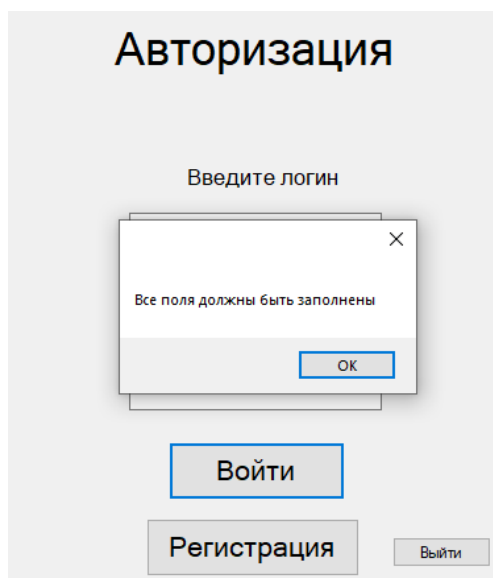


Рисунок 4.2.1 – Попытка авторизоваться с пустыми полями

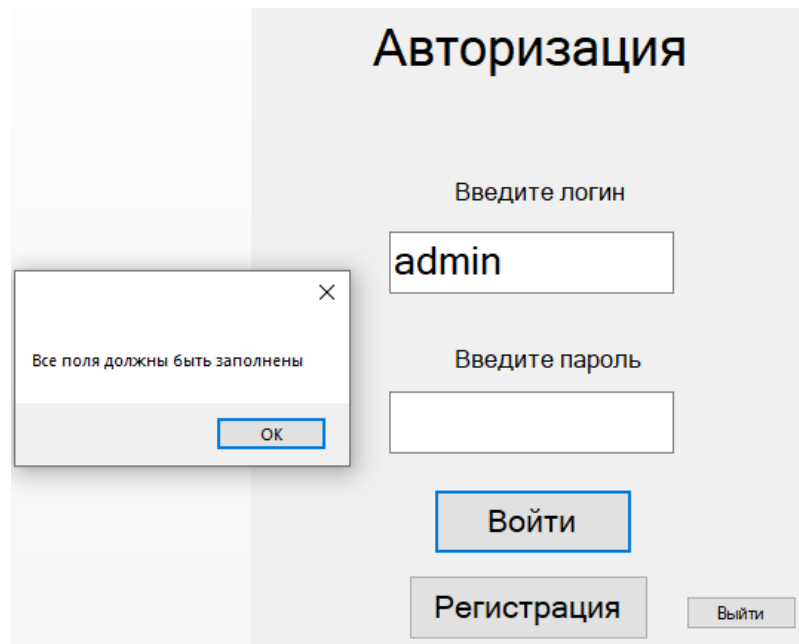


Рисунок 4.2.2 – Попытка авторизоваться с одним пустым полем

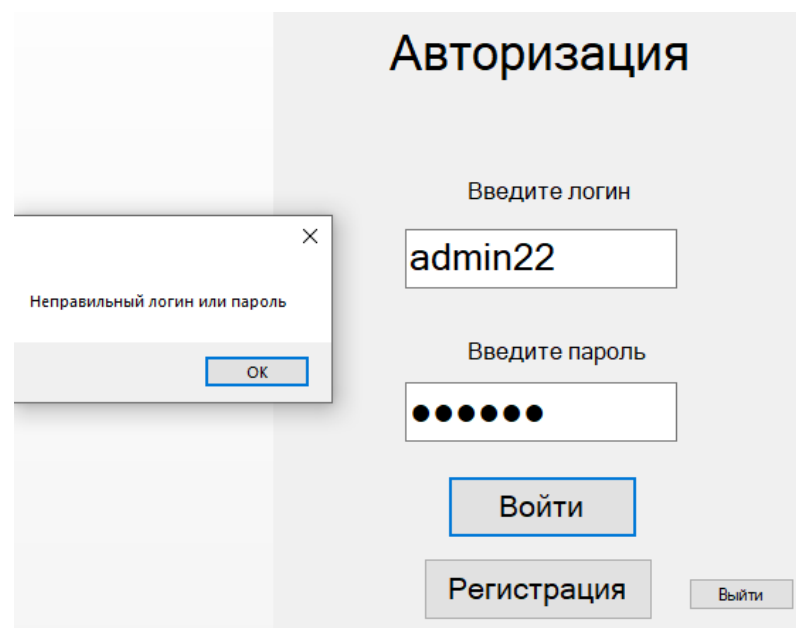
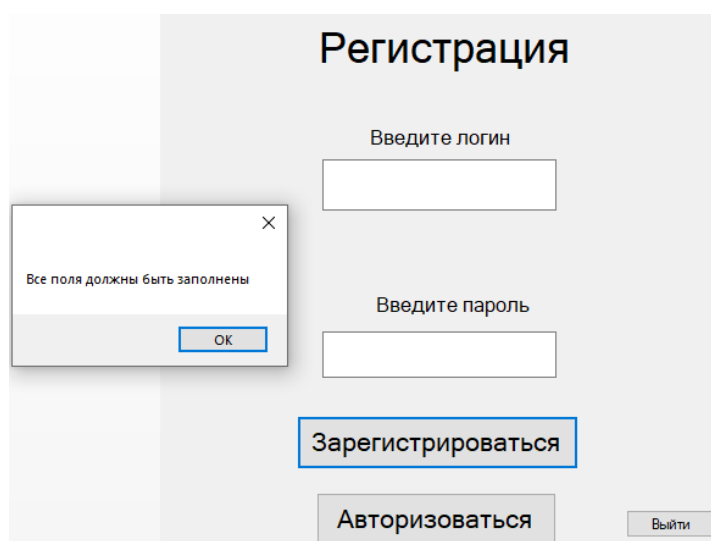


Рисунок 4.2.3 – Попытка авторизовать не зарегистрированного пользователя

Таким образом, можно заметить, что попасть в игру не зарегистрированный пользователь не может, все части кода программы работают корректно.

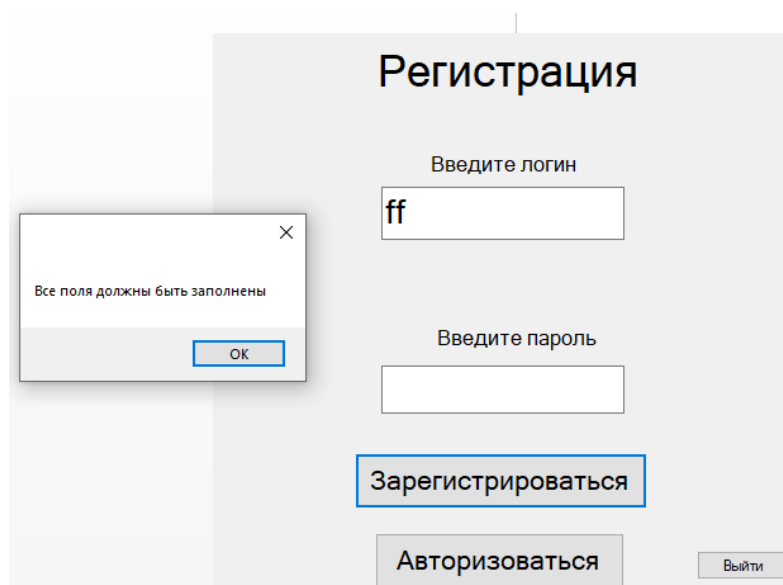
Пробуем провести тесты над формой «Регистрация», переключившись на данную форму через кнопку «Регистрация».

Необходимо будет протестировать следующие случаи: создание аккаунта с пустым логином и паролем, только с введенным логином, только с введенным паролем, а также попробуем повторно зарегистрировать уже зарегистрированного пользователя. Результаты данных тестов представлены на рисунках 4.2.4 – 4.2.7 соответственно.



The screenshot shows a web form titled "Регистрация" (Registration). It contains two input fields: "Введите логин" (Enter login) and "Введите пароль" (Enter password). Below these fields are three buttons: "Зарегистрироваться" (Register), "Авторизоваться" (Log in), and "Выйти" (Logout). A modal dialog box is displayed over the form, containing the text "Все поля должны быть заполнены" (All fields must be filled) and an "ОК" button. The "Зарегистрироваться" button is highlighted with a blue border.

Рисунок 4.2.4 – Попытка зарегистрироваться с пустыми полями



The screenshot shows the same "Регистрация" form. The "Введите логин" field now contains the text "ff", while the "Введите пароль" field remains empty. The modal dialog box with the message "Все поля должны быть заполнены" is still present. The "Зарегистрироваться" button is highlighted with a blue border.

Рисунок 4.2.5 – Попытка зарегистрироваться с пустым паролем

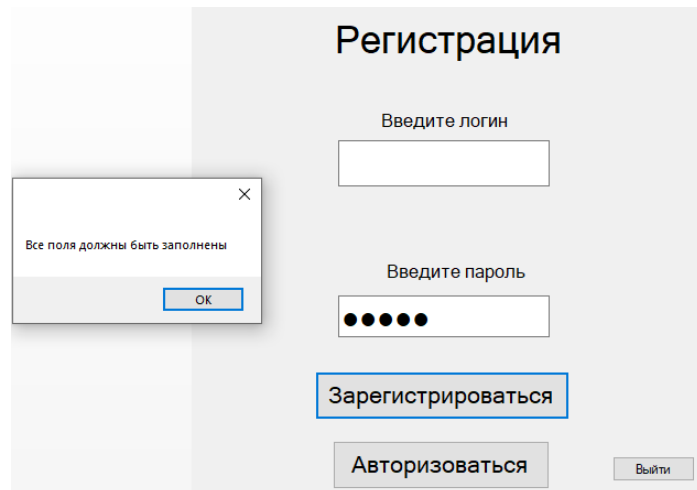


Рисунок 4.2.6 – Попытка зарегистрироваться с пустым логином

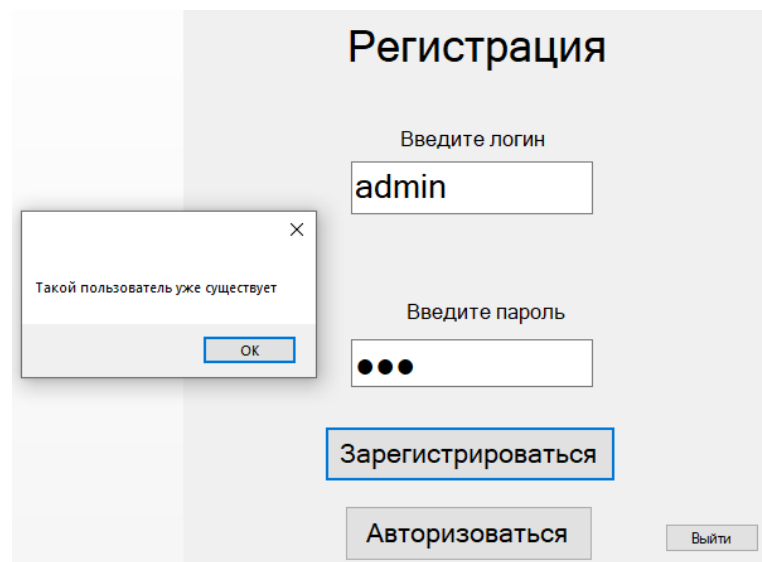
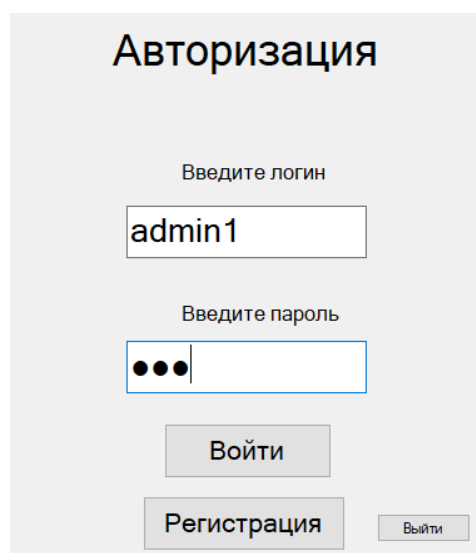


Рисунок 4.2.7 – Попытка зарегистрировать уже зарегистрированного пользователя

Видим, что все фрагменты кода в форме регистрации написаны верно, и создают нового пользователя только при условии ввода корректных данных.

Попробуем авторизовать только что зарегистрированного пользователя (рисунки 4.2.8-4.2.9).



Авторизация

Введите логин

admin1

Введите пароль

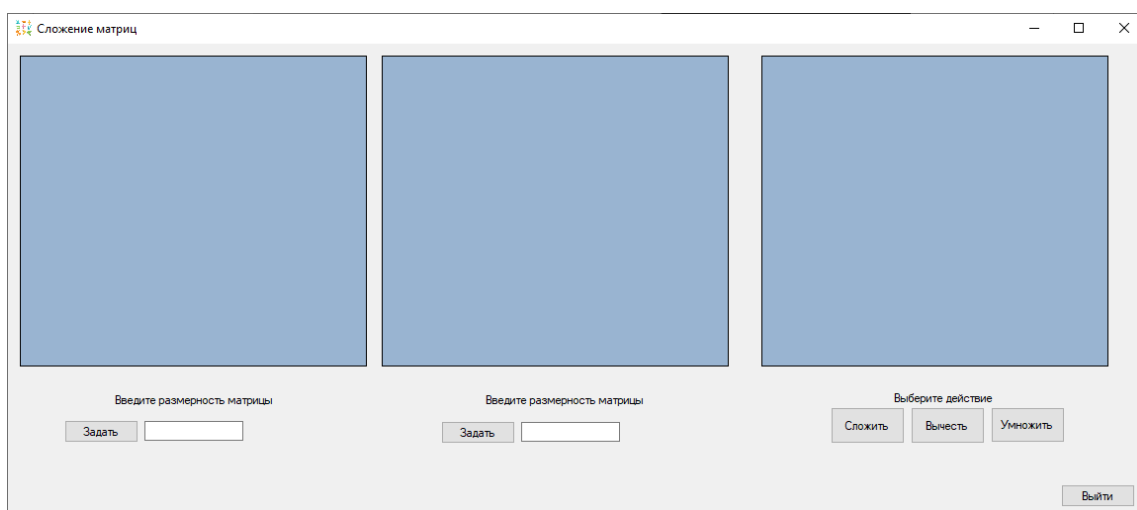
•••

Войти

Регистрация

Выйти

Рисунок 4.2.8 – Авторизация существующего пользователя



Сложение матриц

Введите размерность матрицы

Задать

Введите размерность матрицы

Задать

Выберите действие

Сложить

Вычесть

Умножить

Выйти

Рисунок 4.2.9 – Переход на калькулятора

Пробуем провести тесты над главной формой «Сложение матриц», переключившись на данную форму через кнопку «Войти».

Необходимо будет протестировать случаи проведения операций с матрицами разных размерностей. Результаты данных тестов представлены на рисунках 4.2.10 – 4.2.12 соответственно.

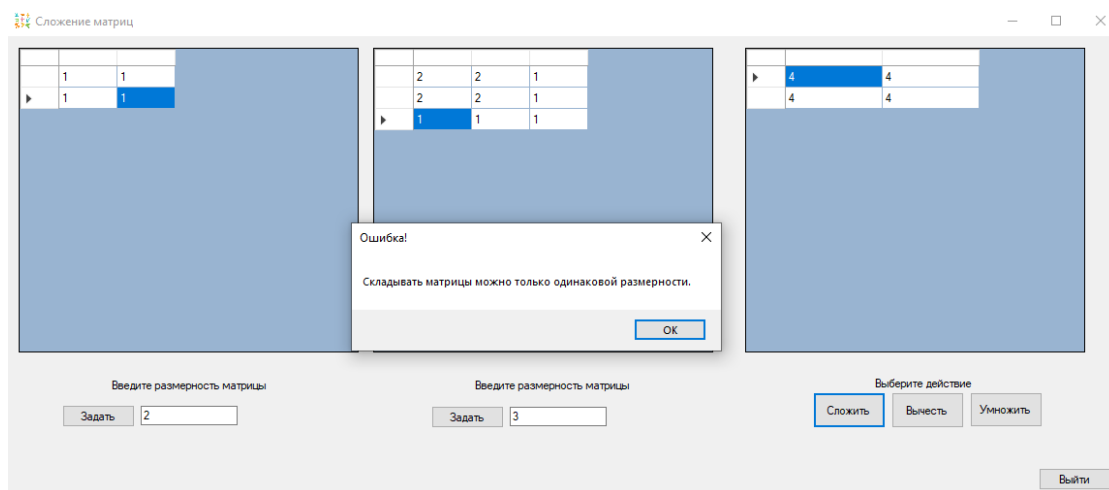


Рисунок 4.2.10 – Попытка сложить матрицы разной размерности

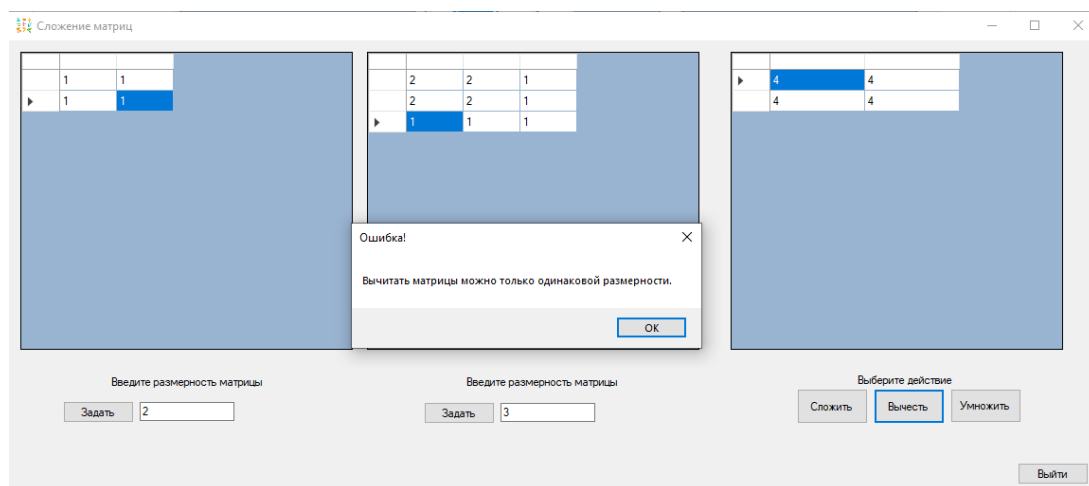


Рисунок 4.2.11 – Попытка вычесть матрицы разной размерности

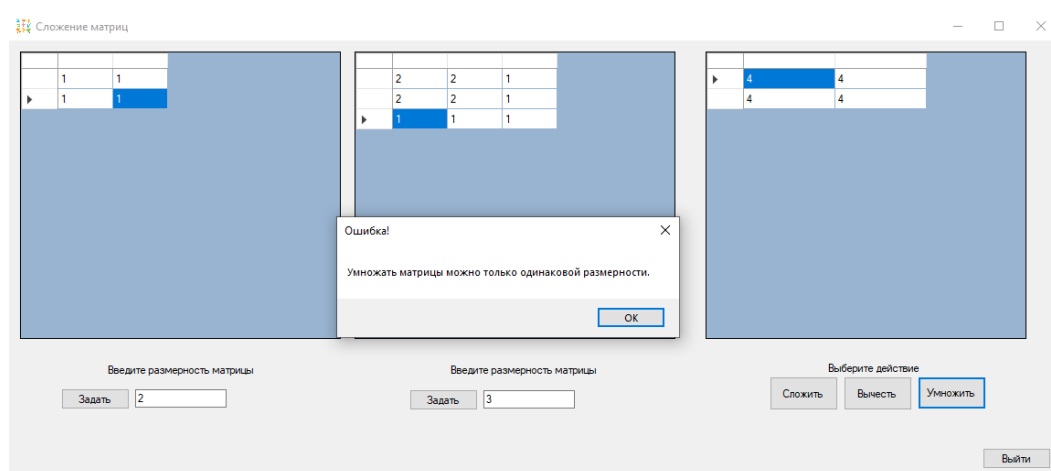


Рисунок 4.2.12 – Попытка перемножить матрицы разной размерности

Заметим, что при соблюдении всех правил регистрации и авторизации можно успешно попасть на форму игры. При попытках сложить матрицы разной размерности программа выдает ошибку. Таким образом, программа проходит ручное тестирование.

Заключение

В ходе выполнения данной курсовой работы было разработано приложение, реализующее приложение «Сложение матриц» для взаимодействия пользователя с калькулятором матриц. Данное приложение написано на языке программирования C# в среде разработки Microsoft Visual Studio Community 2022 (64-разрядная версия) – Current Версия 17.5.5 на платформе .NET Framework. Хранение данных было организовано с помощью подключения СУБД SQL Server Management Studio.

В ходе выполнения курсовой работы были освоены следующие навыки:

- навыки создания десктопных приложений на языке программирования C#;
- навыки работы с СУБД SQL Server Management Studio.

Для достижения поставленной цели были решены следующие задачи:

- анализ темы «Сложение матриц»;
- описание алгоритма через словесное описание и создание блок-схемы алгоритма;
- определение временной сложности алгоритма;
- построение диаграммы классов;
- реализация механизма авторизации и регистрации пользователя;
- реализация механизма работы с БД;
- реализация механизма добавления пользователей;
- функциональное тестирование программы;
- ручное тестирование программы.

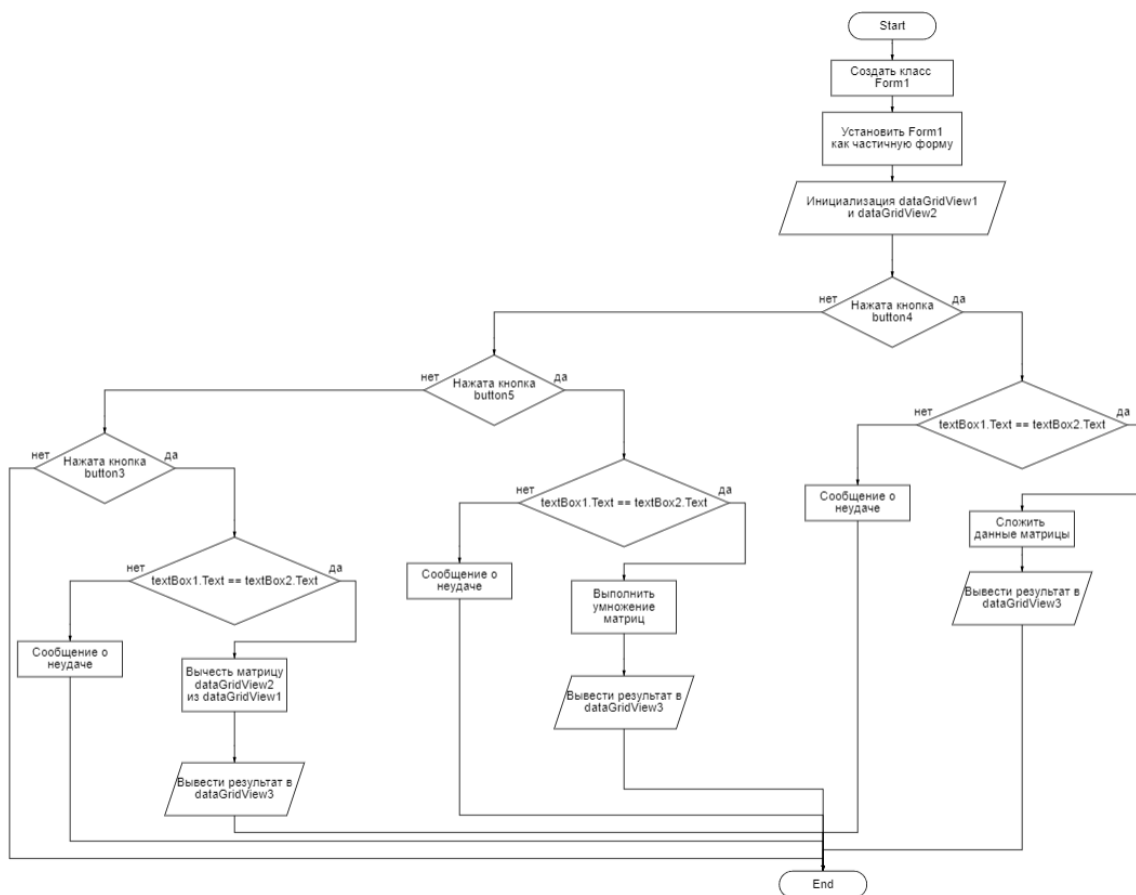
Список использованных источников

1. Образовательный стандарт вуза ОС ТУСУР 01-2021 [Электронный ресурс]: сайт ТУСУРа URL: <https://regulations.tusur.ru/storage/150499/>;
2. Харченко С.С. Основы программирования: учебно-методическое пособие по курсовой работе. – Томск: В-Спектр, 2019. – 48 с.;
3. <https://learn.microsoft.com/ru-ru/visualstudio/get-started/>;
4. <https://blog.fenix.help/podgotovka-k-testam-yekzamenam-zachetam/kak-skladyvat-matritsy-razlichnogo-poryadka>;
5. https://www.yuripetrov.ru/edu/python/ch_06_01.html;
6. <https://gb.ru/blog/luchshie-ide-dlya-razrabotki-na-c/>;
7. <https://learn.microsoft.com/ru-ru/sql/ssms/>;
8. <https://practicum.yandex.ru/blog/chto-takoe-test-keys-i-kak-ego-sostavit/>;

Приложение А

(Обязательное)

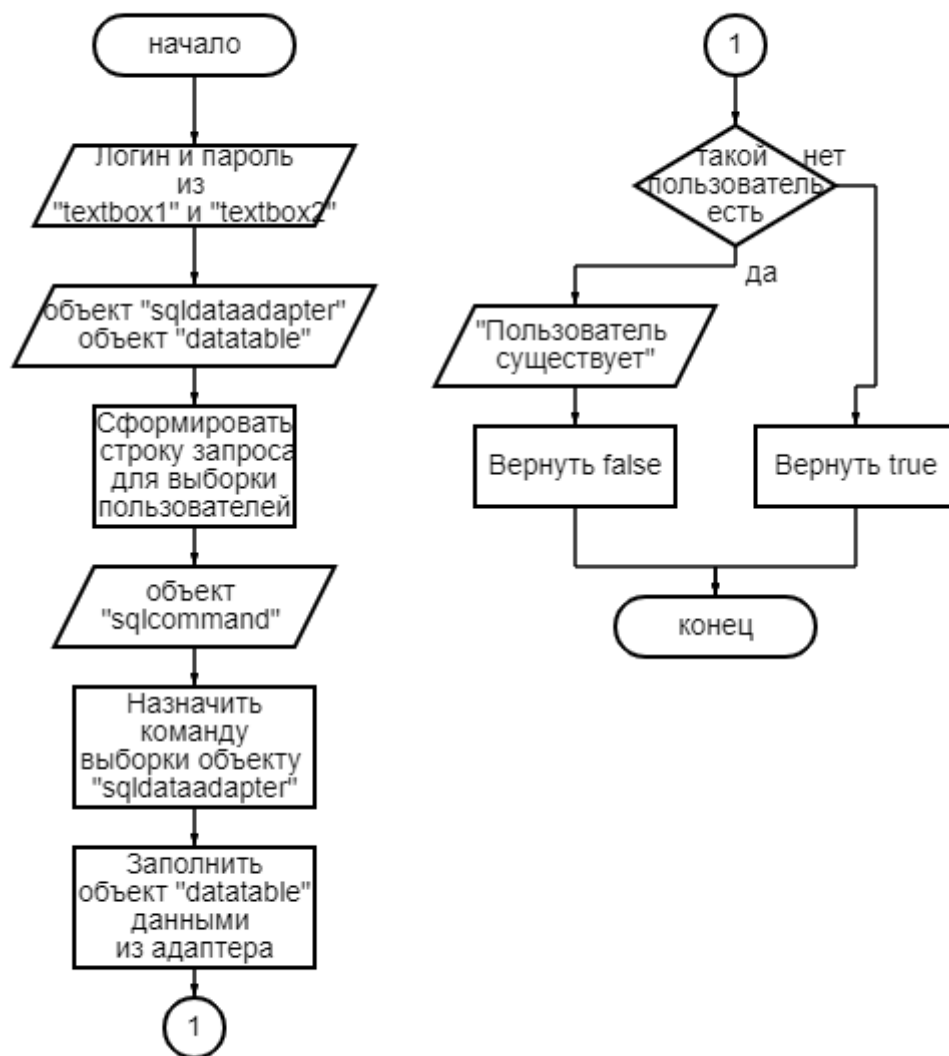
Блок-схема к алгоритму А



Приложение Б

(Обязательное)

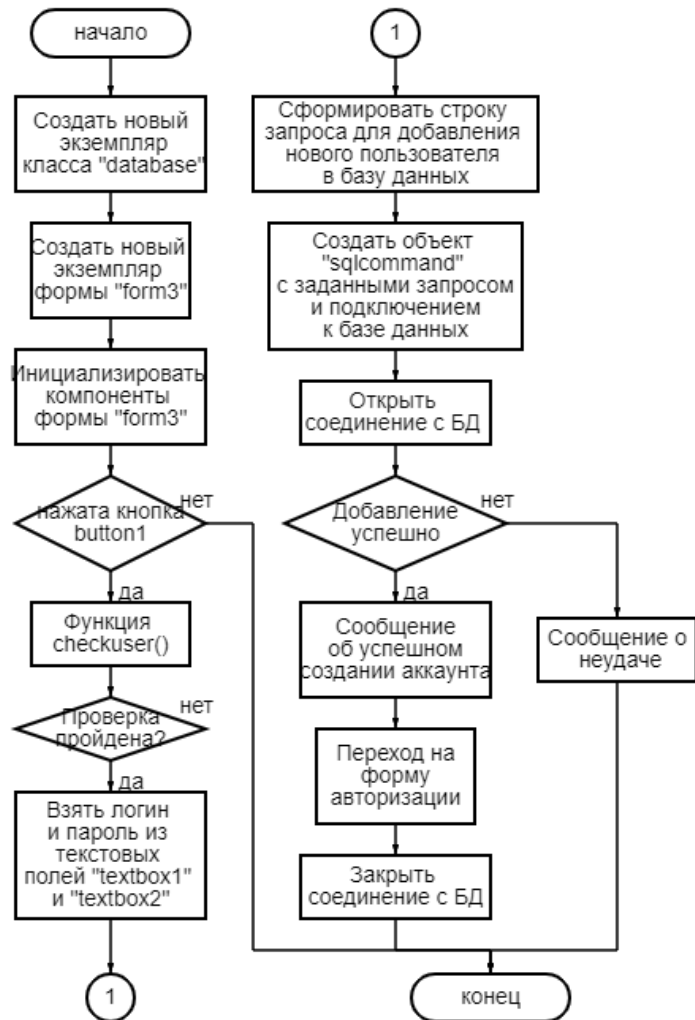
Блок-схема к алгоритму В



Приложение В

(Обязательное)

Блок-схема к алгоритму С



Приложение Г

(Обязательное)

Блок-схема к алгоритму D

