Автономная некоммерческая организация профессиональная образовательная организация Московский Международный Колледж Цифровых технологий "Академия ТОП"

ОТЧЕТ ПО ПРОХОЖДЕНИЮ УЧЕБНОЙ ПРАКТИКИ

Преподаватель Гвоздев С.М.

личная подпись, дата

Студент Борисова Д.А.

личная подпись, дата

Группа 9/3-РПО-23/1

г. Новомосковск

2025г.

Оглавление

[ВВЕДЕНИЕ 3](#_Toc217302531)

[ГЛАВА 1. СОЗДАНИЕ ПРОЕКТА 5](#_Toc217302532)

[1.1. Разметка MainWindow 5](#_Toc217302533)

[ГЛАВА 2. РЕАЛИЗАЦИЯ «MAINPAGE» И «TASKPAGE» 10](#_Toc217302534)

[ГЛАВА 3. РЕАЛИЗАЦИЯ ДИРЕКТОРИИ «PageTask» 15](#_Toc217302535)

[ЗАКЛЮЧЕНИЕ 16](#_Toc217302536)

### ВВЕДЕНИЕ

Работа посвящена разработке комплексного учебного приложения на платформе Windows Presentation Foundation (WPF) в среде Visual Studio с использованием языка программирования C#.

Приложение разработано с целью практического закрепления знаний в области:

* Объектно-ориентированного программирования (ООП)
* Разработки графических пользовательских интерфейсов (GUI)
* Работы с коллекциями и алгоритмами обработки данных
* Анализа временной и пространственной сложности алгоритмов
* Обработки исключений и валидации данных

Цель работы

Основная цель: Разработать многофункциональное WPF приложение, которое демонстрирует решение пяти различных задач математического и программистского характера с интуитивным и современным пользовательским интерфейсом.

Задачи:

Реализовать решение систем линейных уравнений методом Крамера с анализом особых случаев

Реализовать конвертацию десятичных чисел в различные системы счисления (двоичная, восьмеричная, шестнадцатеричная)

Разработать класс Student с методами расчёта успеваемости и определения статуса обучения

Реализовать алгоритм удаления дубликатов из массива с сохранением порядка

Реализовать алгоритм поиска анаграмм в текстовом фрагменте

Требования к приложению

Функциональные требования:

* Приложение должно содержать минимум 5 отдельных модулей (вкладок) для решения различных задач
* Каждый модуль должен иметь собственный интерфейс с полями ввода, кнопками и полями для вывода результатов
* Все вычисления должны проводиться корректно и проверяться на достоверность
* Приложение должно обрабатывать ошибки пользовательского ввода и выводить соответствующие сообщения

Нефункциональные требования:

* Интерфейс должен быть современным и привлекательным (использование Material Design)
* Приложение должно быть отзывчивым и не зависать при обработке данных
* Код должен быть хорошо структурирован, документирован и следовать лучшим практикам C#

# ГЛАВА 1. СОЗДАНИЕ ПРОЕКТА

## 1.1. Разметка MainWindow

Сначала создание нового проекта: в списке шаблонов выбирать «Приложение WPF (Microsoft)» и нажать кнопку «Далее».

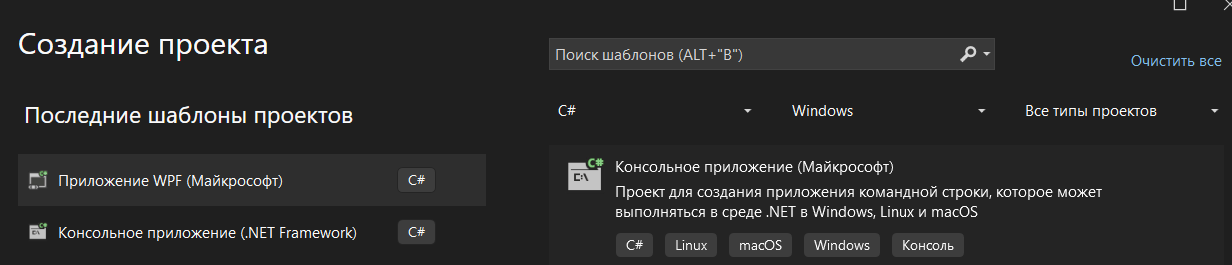


Рисунок 1.1 – Создание проекта

Далее в настройках нового проекта заполнить поля с именем проекта и расположением.

После заполнения нажимаем «Далее»

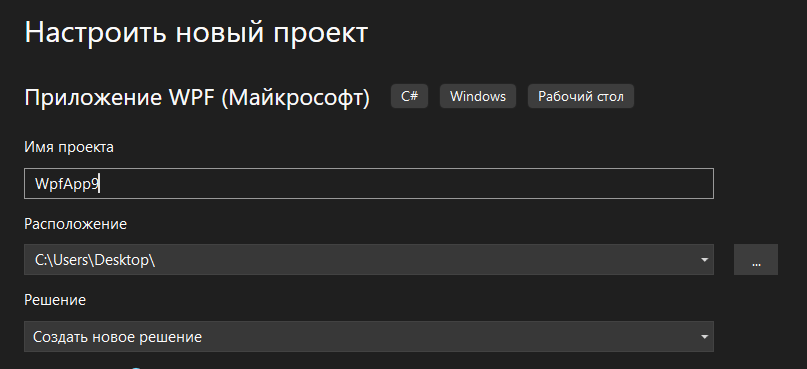


Рисунок 1.2 – Настройка нового проекта

Выбрать платформу «.NET 8» и нажать создать.

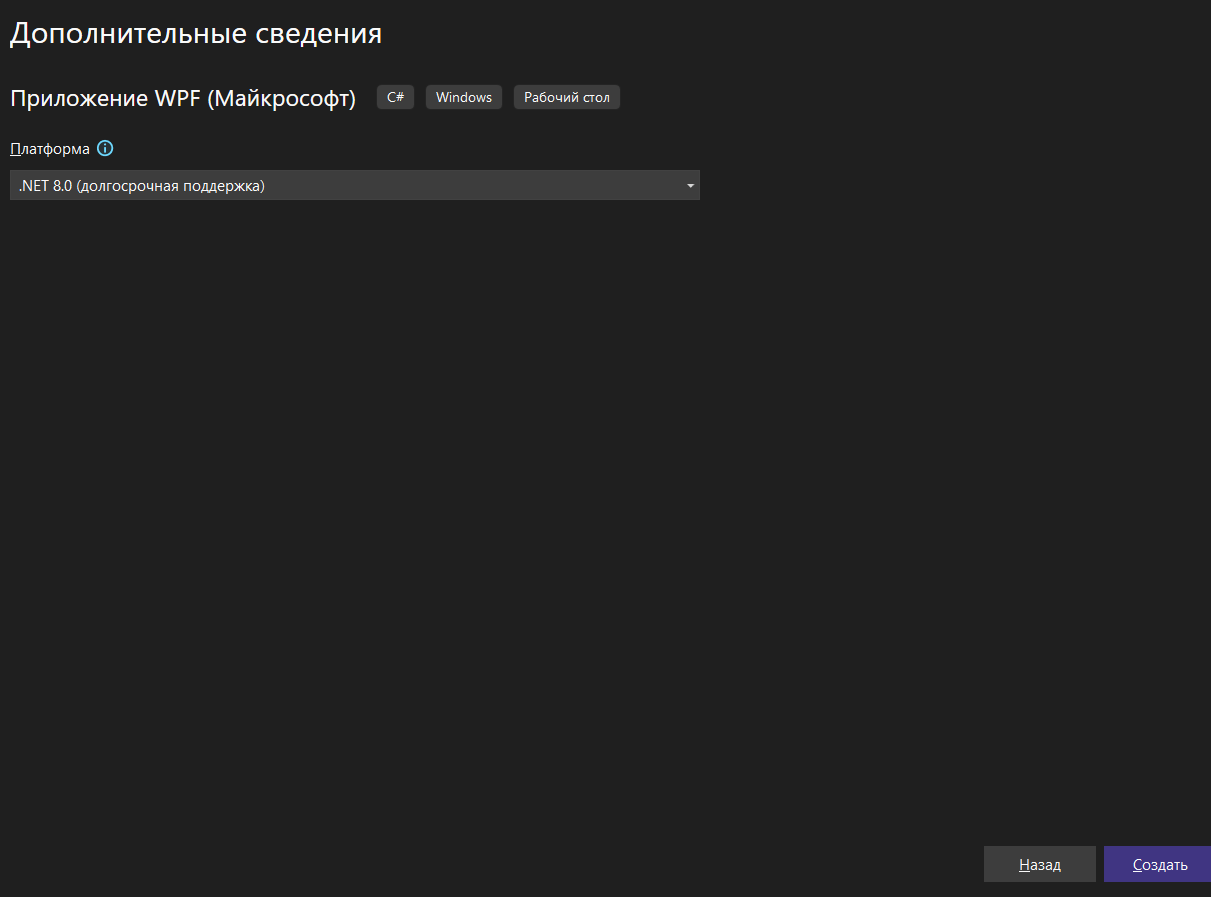


Рисунок 1.3 – Выбор платформы проекта

Формируем базовую структуру проекта. Она будет включать следующие папки:

- Core – папка, в которой размещаются служебные и вспомогательные классы;

- Images – папка для хранения всех графических ресурсов приложения;

- AppStyle – папка, содержащая файлы со стилями оформления интерфейса;

- View – папка, где находятся окна, страницы и пользовательские элементы, используемые в приложении.

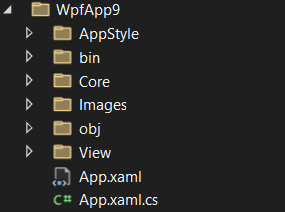


Рисунок 1.4 – Структура проекта

Далее нужно подключить библиотеку оформления Material Design.

Для этого щёлкаем правой кнопкой мыши по проекту, в открывшемся меню выбираем пункт «Управление пакетами NuGet…», переходим на вкладку «Обзор», в строку поиска вводим `Material Design` и устанавливаем два пакета: MaterialDesignThemes и MaterialDesignColors.

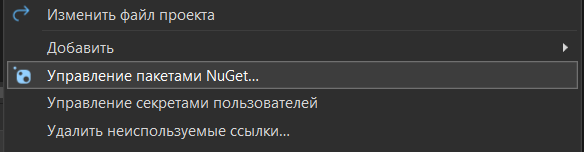


Рисунок 1.5 – Контекстное меню «Управление пакетами NuGet…»



Рисунок 1.6 – Установка «MaterialDesignThemes» и «MaterialDesignColors».

Далее перемещаем файл «MainWindow» в папку «View» и вносим корректировки в файл «App.xaml». Обновляем путь к «MainWindow» в атрибуте «StartupUri», подключаем словарь ресурсов, объявляем пространство имён «materialDesign» и добавляем ссылку на файл с общими стилями приложения.



Рисунок 1.8 – Редактирование файла «App.xaml»

Далее переходим к настройке основных стилей оформления. Для этого открываем папку «AppStyle» и вносим изменения в ресурсный словарь «MainStyle».

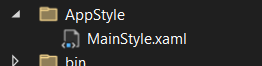


Рисунок 1.9 – Редактирование словаря ресурсов «MainStyle.xaml»

Затем переходим в папку «Core» и создаём класс «ConnectionCore», предназначенный для управления навигацией в приложении.



Рисунок 1.10 – Класс «ConnectionCore»

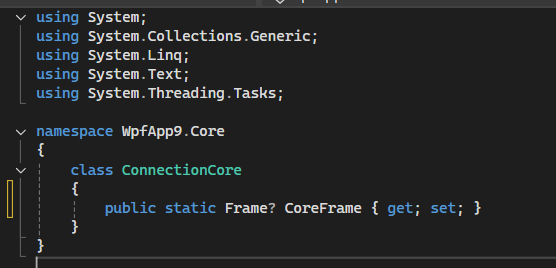


Рисунок 1.11 – Реализация класса «ConnectionCore»

Далее открываем папку «View» и создаём в ней две новые подпапки: «Pages» и «PageTask». Затем в них размещаем пять страниц для заданий, главную страницу и отдельную страницу с обзором всех пяти заданий.

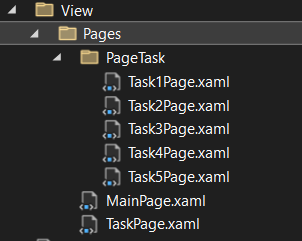


Рисунок 1.12 – Директория «View»

Далее переходим к созданию разметки файла «MainWindow.xaml» и разработке соответствующей логики в «MainWindow.xaml.cs».

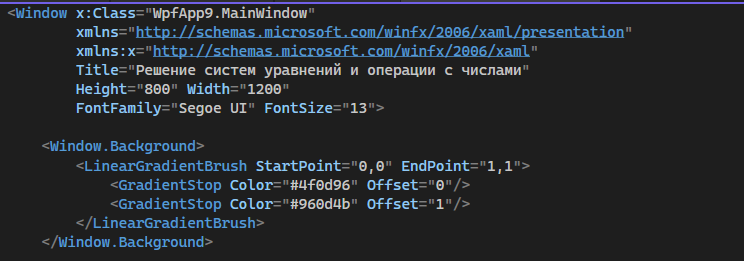


Рисунок 1.13 – Разметка «MainWindow.xaml»

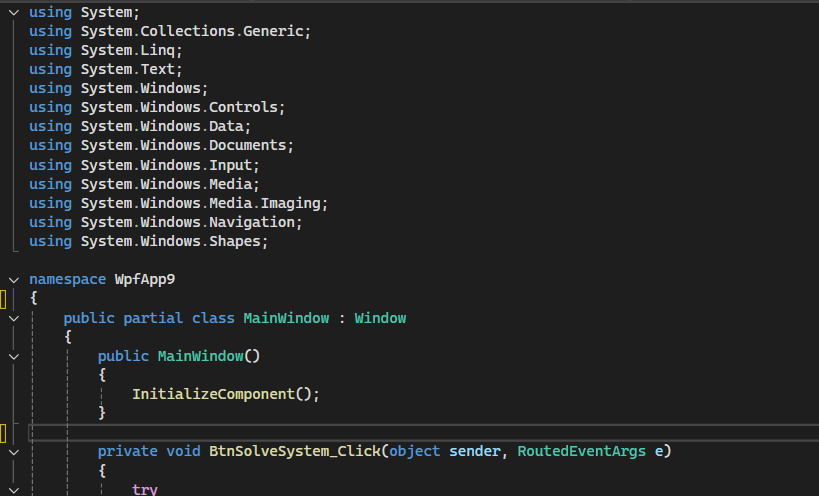


Рисунок 1.14 – Реализации логики «MainWindow.xaml.cs»

На этом этапе завершается создание проекта и настройка интерфейса главного окна MainWindow.

# ГЛАВА 2. РЕАЛИЗАЦИЯ «MAINPAGE» И «TASKPAGE»

Перейдем к разметке «MainPage.xaml».



Рисунок 2.1 – Разметка «MainPage.xaml»

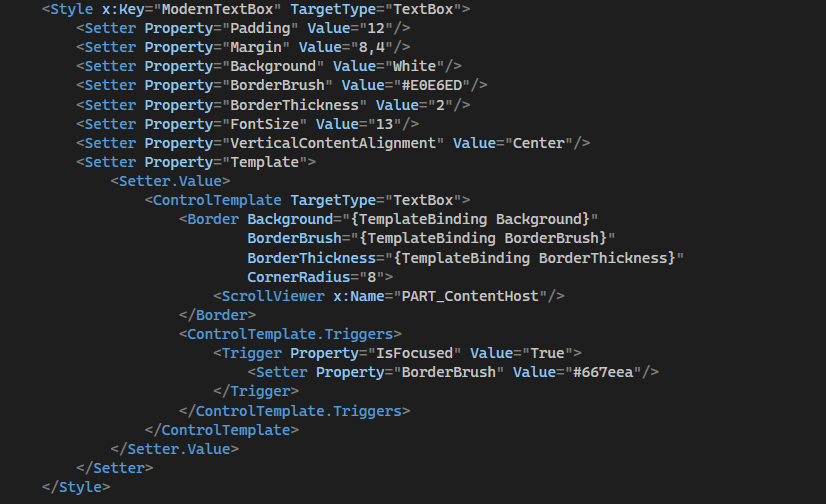


Рисунок 2.2 – Разметка «MainPage.xaml»

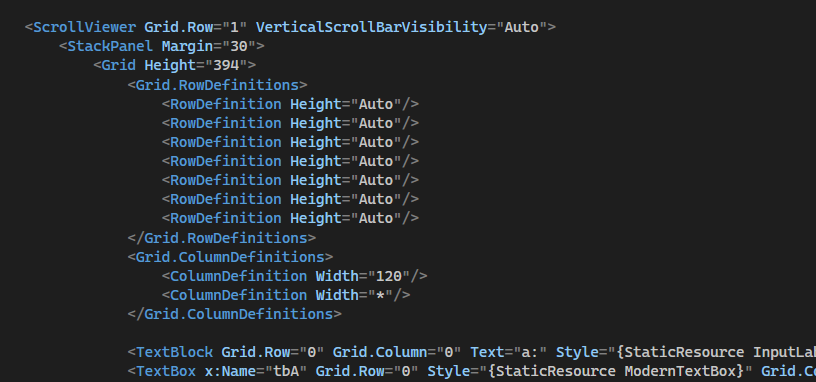


Рисунок 2.3 – Разметка «MainPage.xaml»

После разметки перейдем к реализации метода «BtnTask\_Click».

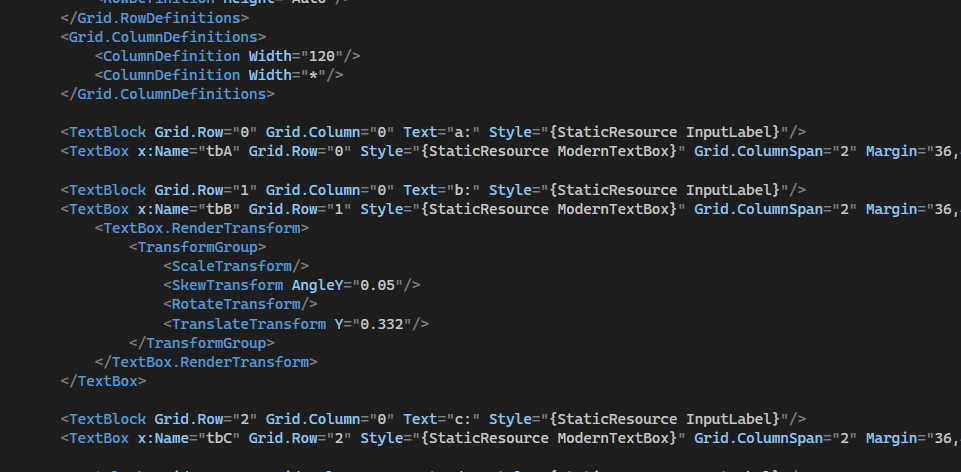


Рисунок 2.5 – Разметка «TaskPage.xaml»

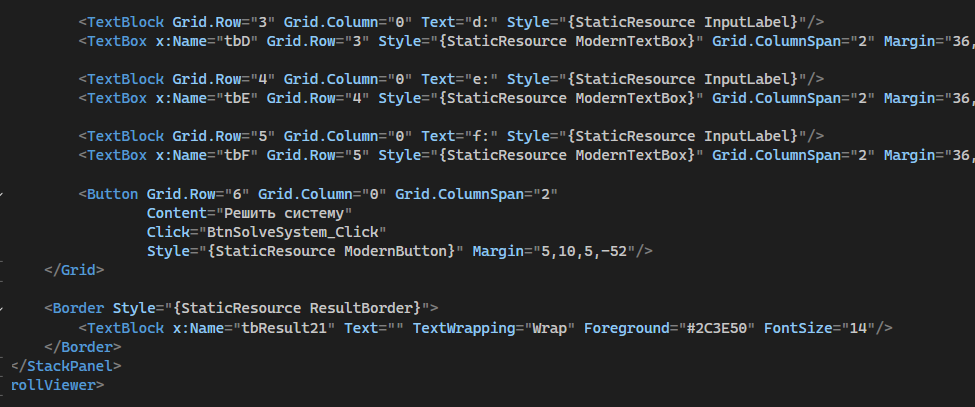


Рисунок 2.6 – Разметка «TaskPage.xaml»

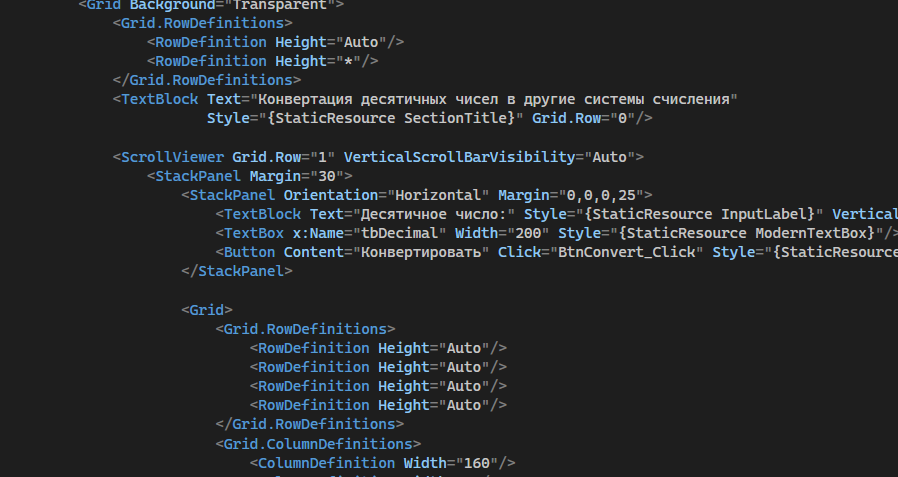


Рисунок 2.7 – Разметка «TaskPage.xaml»

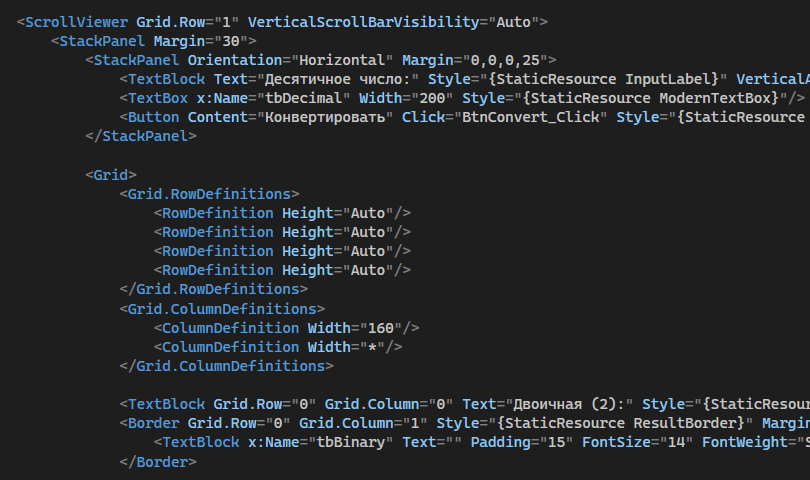


Рисунок 2.8 – Разметка «TaskPage.xaml»

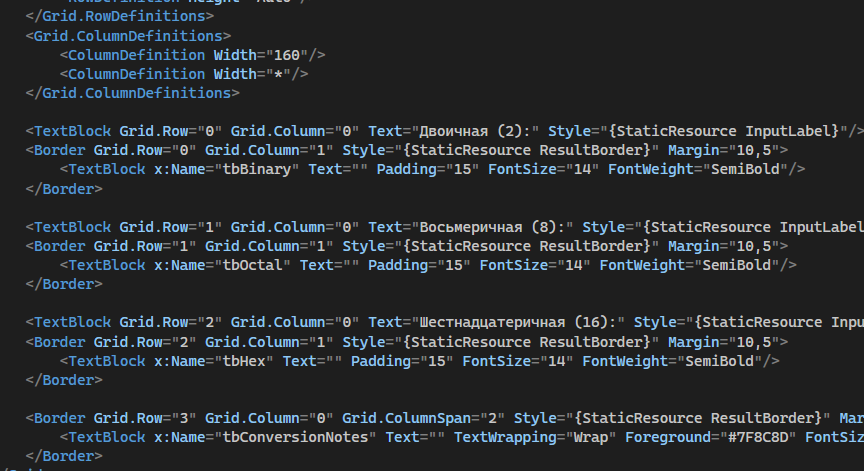


Рисунок 2.9 – Разметка «TaskPage.xaml»

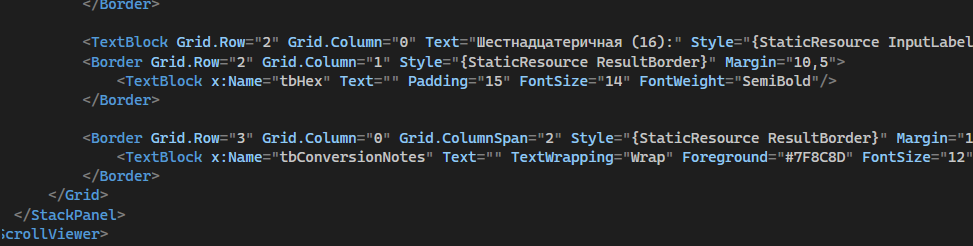


Рисунок 2.10 – Разметка «TaskPage.xaml»

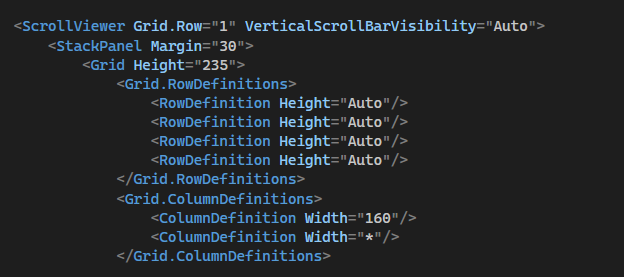


Рисунок 2.10 – Разметка «TaskPage.xaml»

Далее открываем файл «TaskPage.xaml.cs» и реализуем методы, обеспечивающие навигацию между страницами.

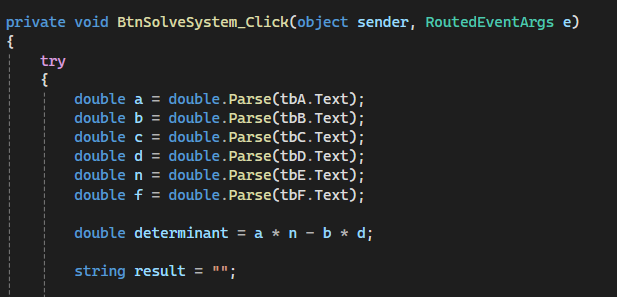


Рисунок 2.12 – Реализация «TaskPage.xaml.cs»

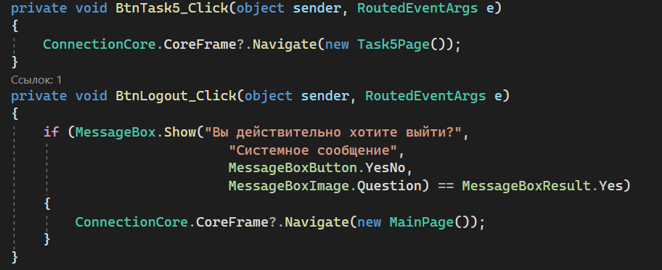


Рисунок 2.13 – Реализация «TaskPage.xaml.cs»

Запустим приложение и проверим навигацию.

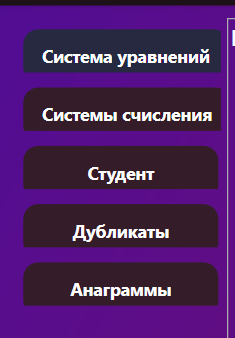


Рисунок 2.14 – MainPage

На этом реализация «MainPage» и «TaskPage» закончена.

# ГЛАВА 3. РЕАЛИЗАЦИЯ ДИРЕКТОРИИ «PageTask»

Директория «PageTask» содержит пять страниц с заданиями. Реализацию начинаем с разметки «Task1Page».

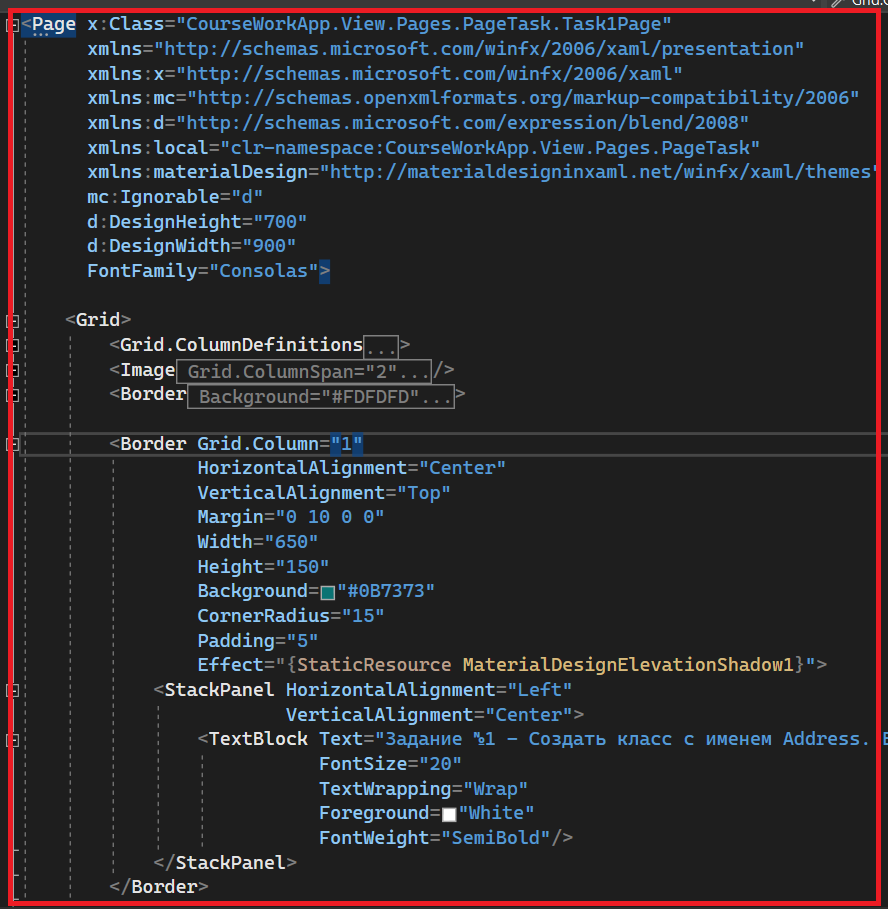


Рисунок 3.1 – Разметка «Task1Page»

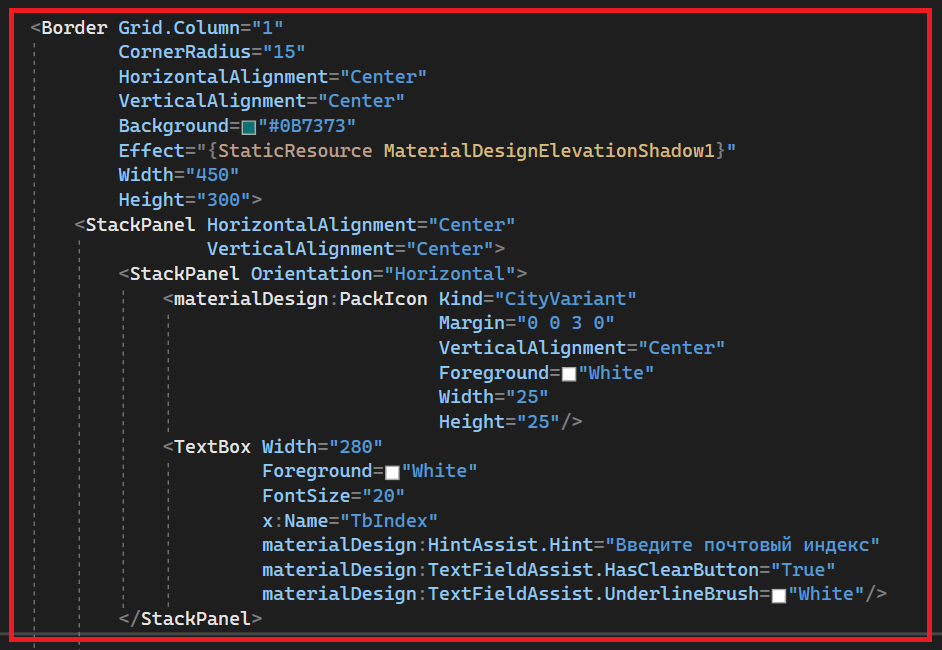


Рисунок 3.2 – Разметка «Task1Page»

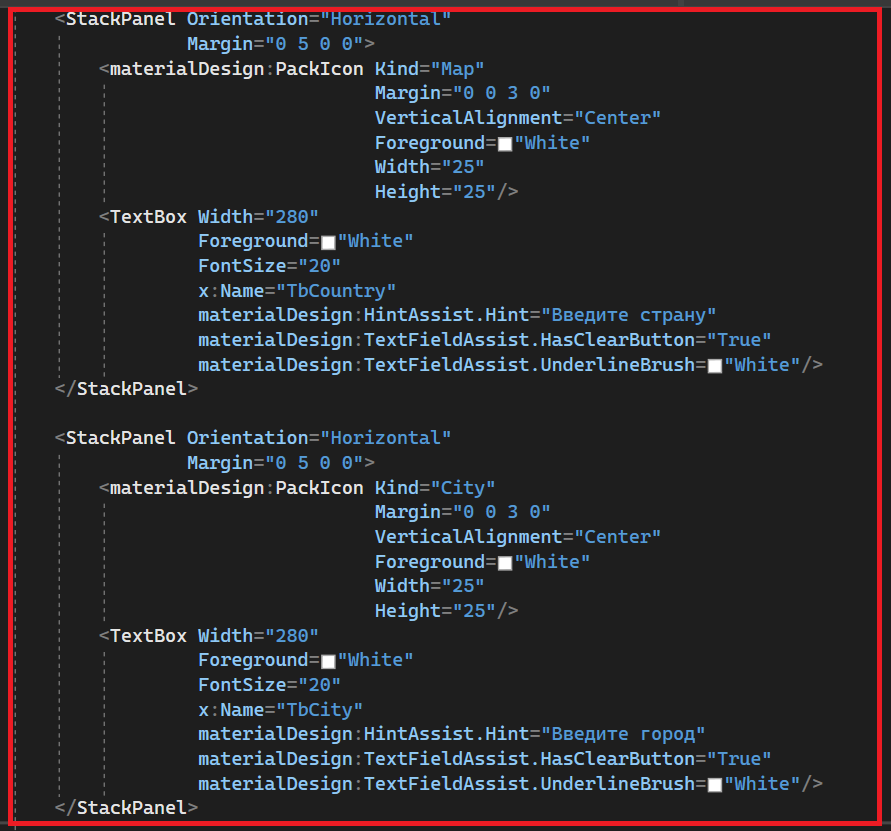


Рисунок 3.3 – Разметка «Task1Page»



Рисунок 3.4 – Разметка «Task1Page»

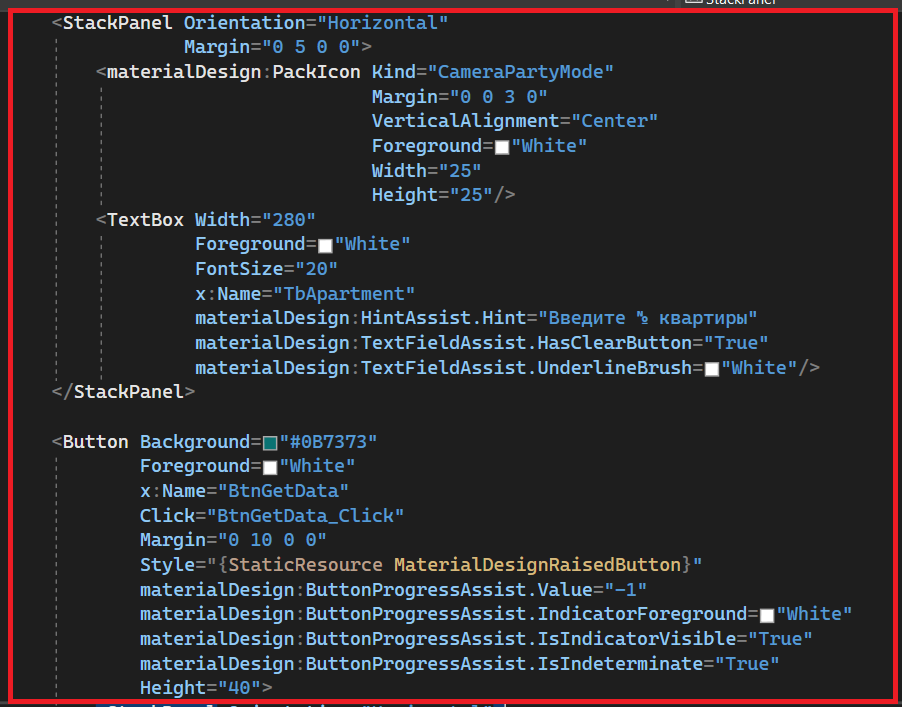


Рисунок 3.5 – Разметка «Task1Page»

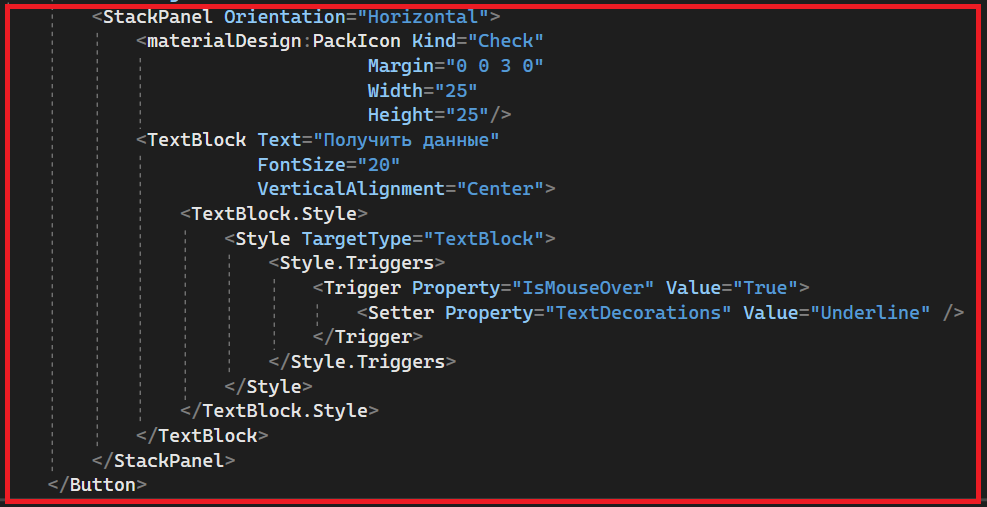


Рисунок 3.6 – Разметка «Task1Page»

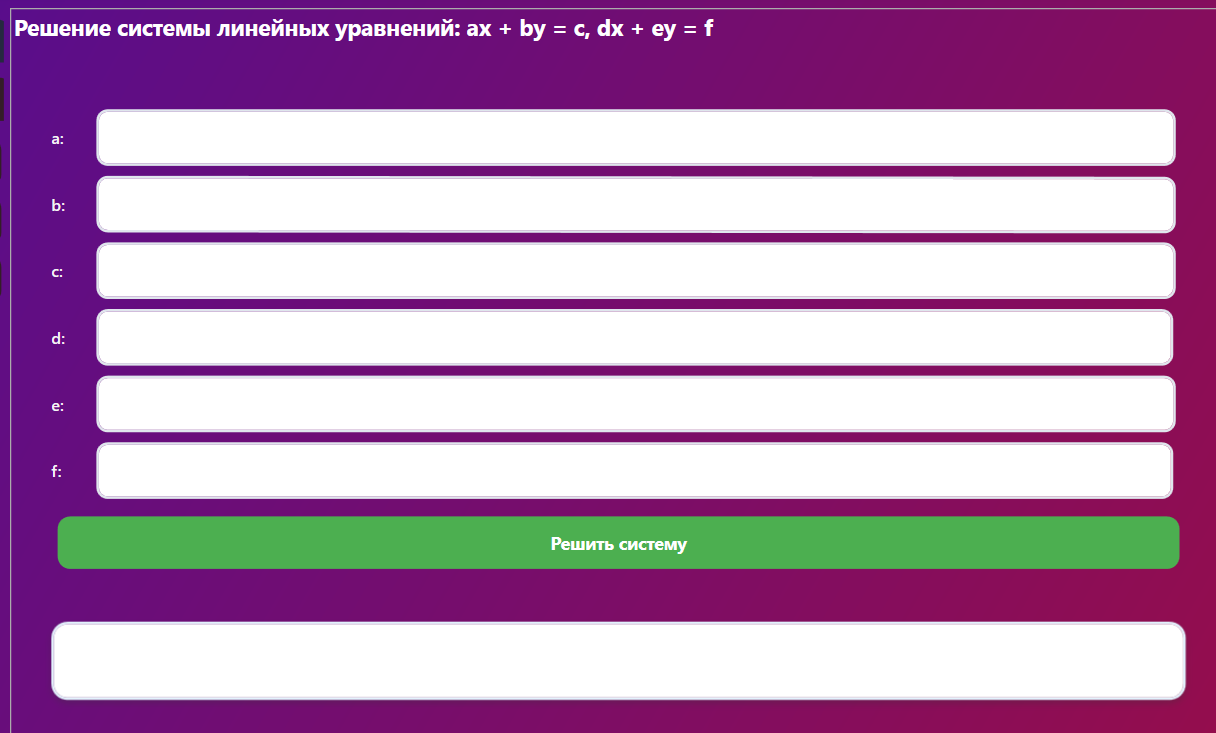


Рисунок 3.7 – Результат разметки «Task1Page»

Для разработки алгоритма решения задачи переходим к файлу «Task1Page.xaml.cs». Согласно условию задания создаём класс «BtnSolveSystem\_Click» с требуемыми свойствами, считываем данные из элементов «TextBox» и присваиваем их соответствующим полям класса, одновременно выполняя проверку на корректность заполнения полей ввода.

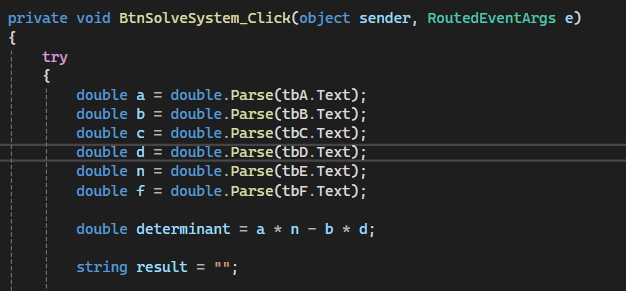


Рисунок 3.8 – Результат класса

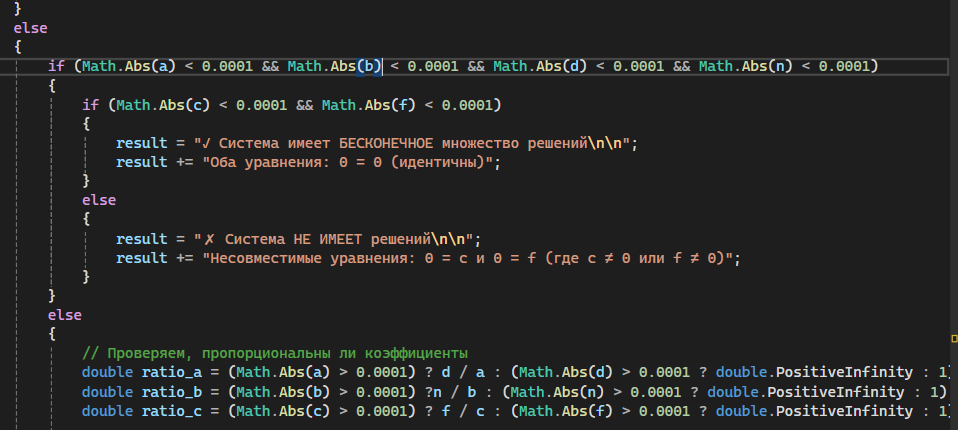


Рисунок 3.9 – Решение задания №1

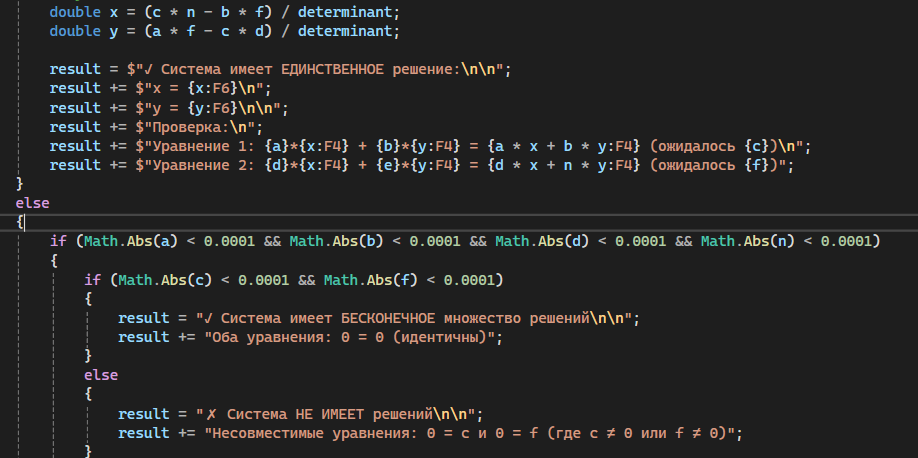


Рисунок 3.10 – Решение задания №1

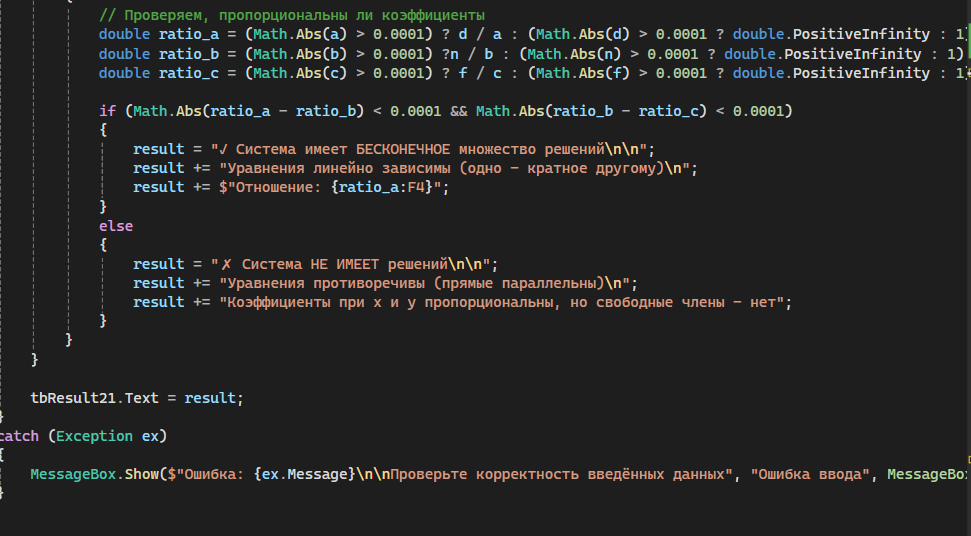


Рисунок 3.11 – Решение задания №1

Запустим приложение и проверим логику решения первого задания.

****

Рисунок 3.12 – Реализация «Task1Page»

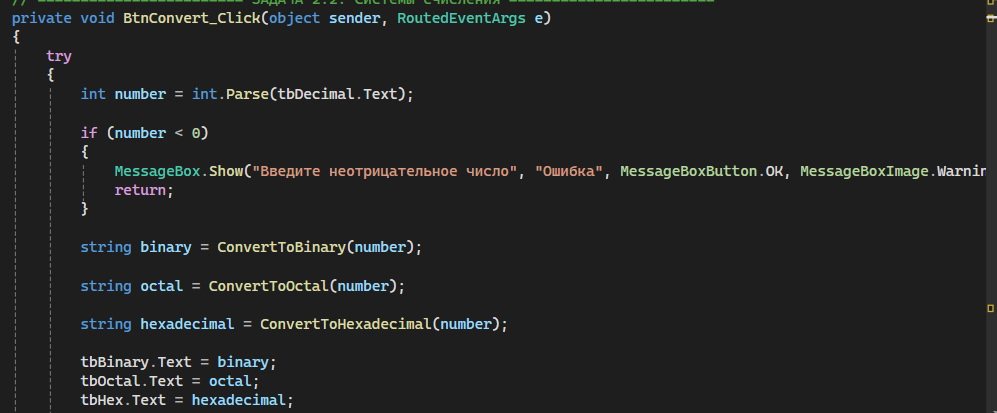
****

Рисунок 3.13 – Решение задания №2

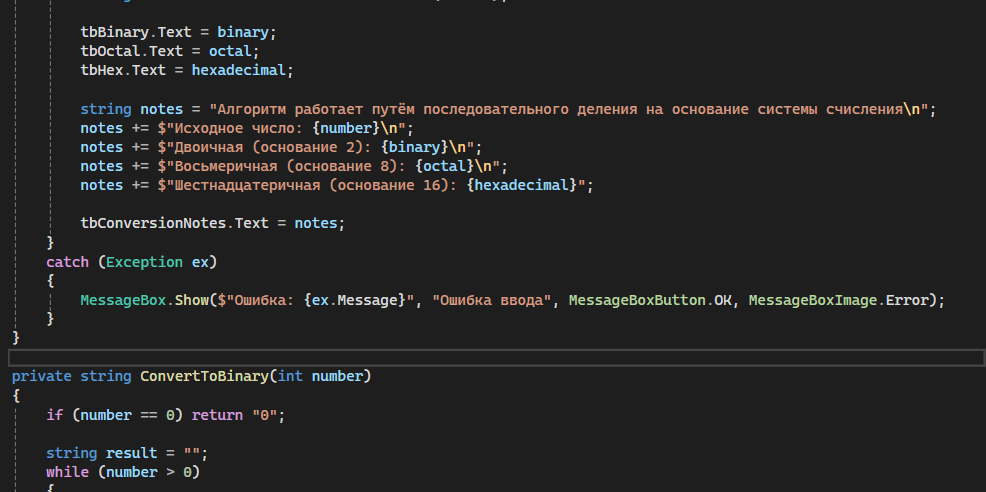
****

Рисунок 3.14 – Решение задания №2

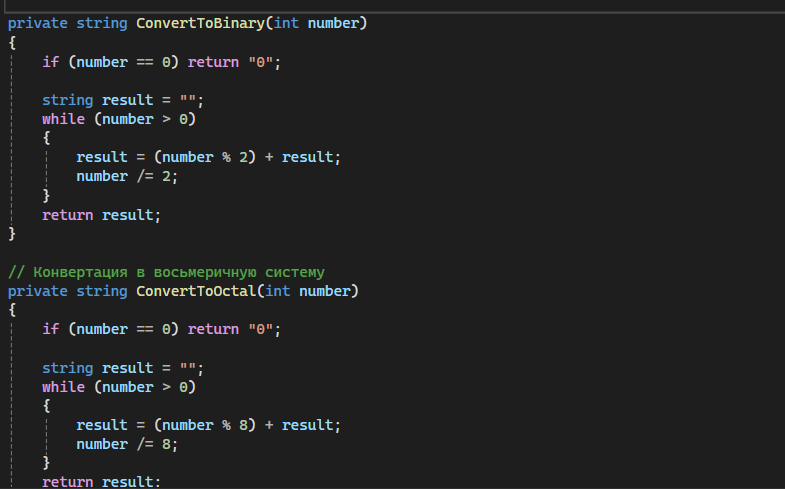
****

Рисунок 3.15 – Решение задания №2

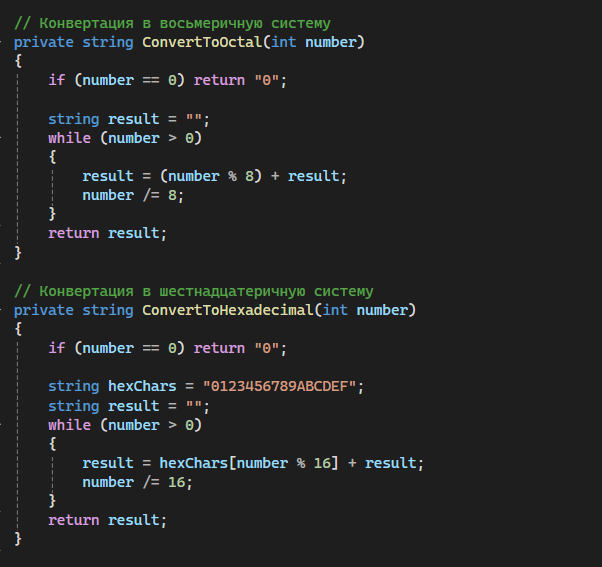
****

Рисунок 3.16 – Решение задания №2

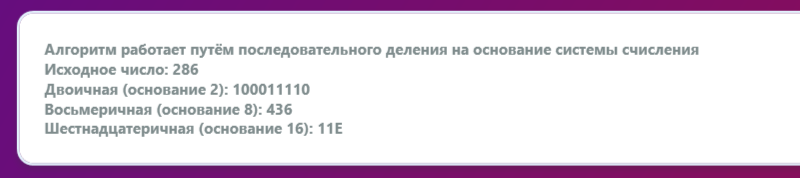
****

Рисунок 3.17 – Реализация «Task2Page»

Далее переходим к реализации разметки «Task3Page».

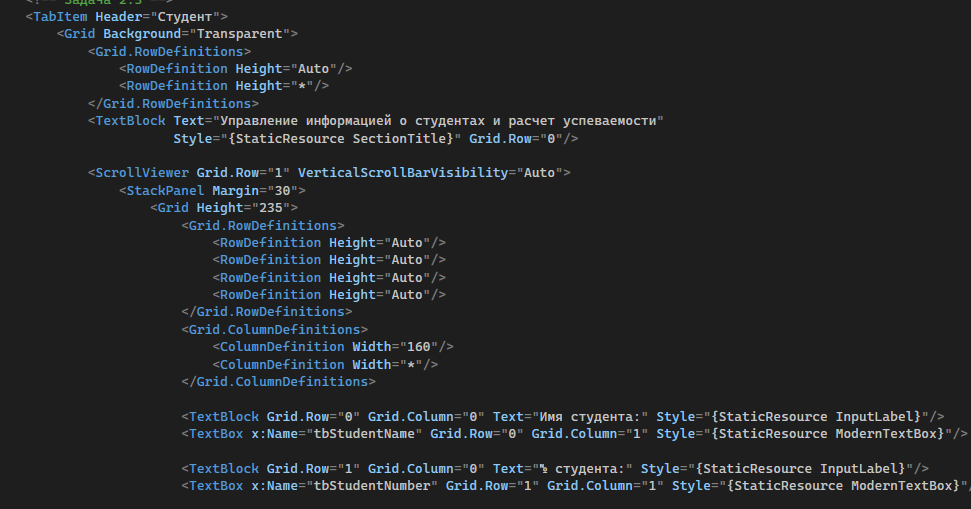


Рисунок 3.18 – Разметка «Task3Page»

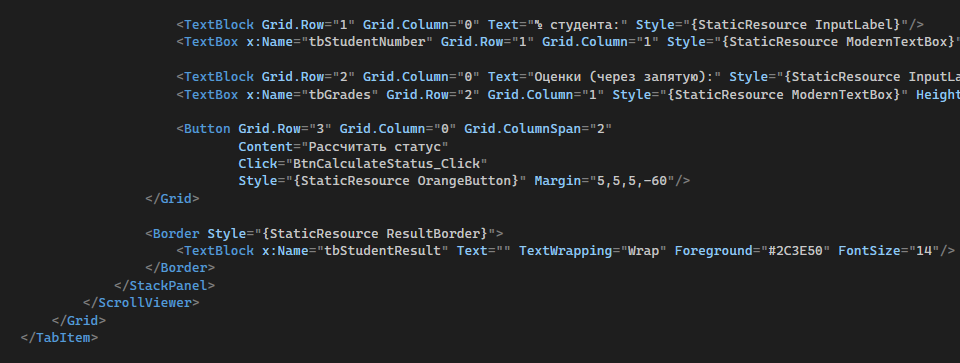


Рисунок 3.19 – Разметка «Task3Page»

Для реализации логики решения задания переходим в файл «Task3Page.xaml.cs».

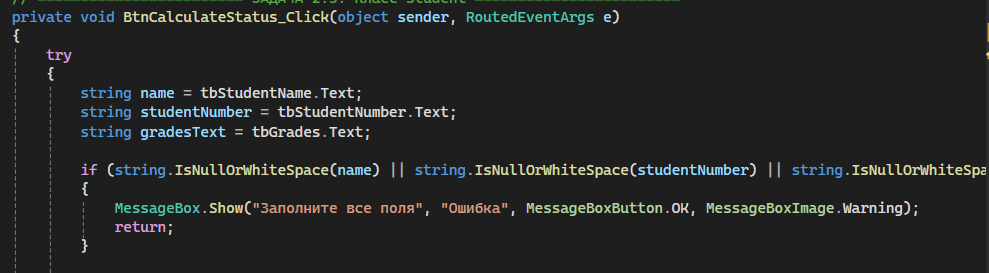


Рисунок 3.20 – Решение задания №3

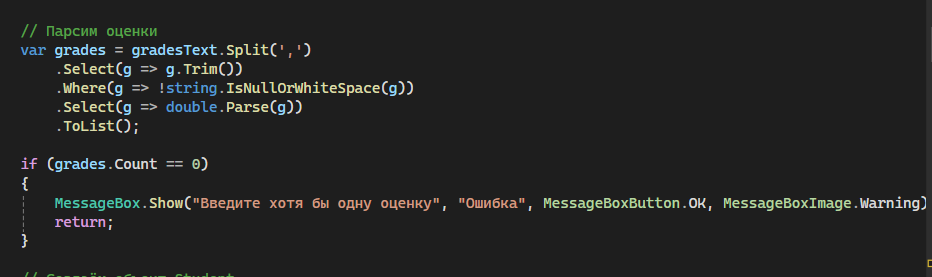


Рисунок 3.21 – Решение задания №3

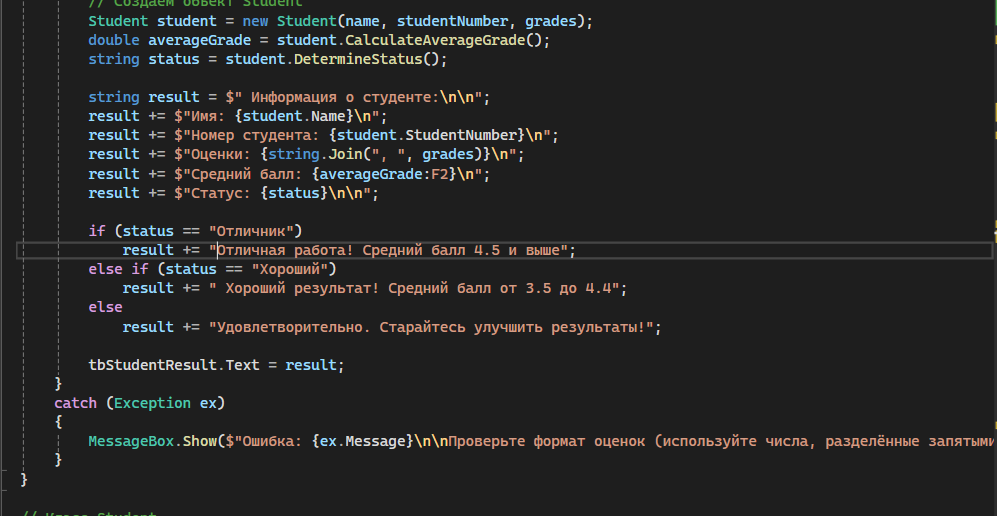


Рисунок 3.22 – Решение задания №3

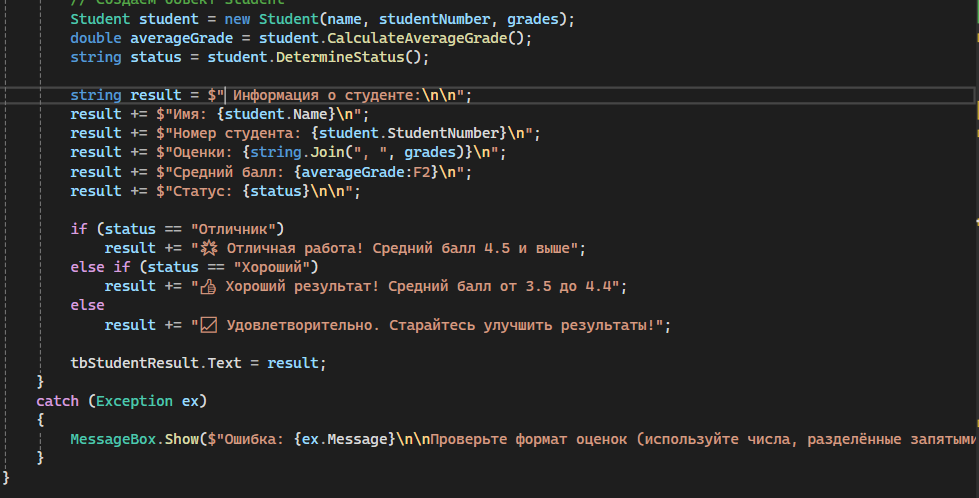


Рисунок 3.23 – Решение задания №3

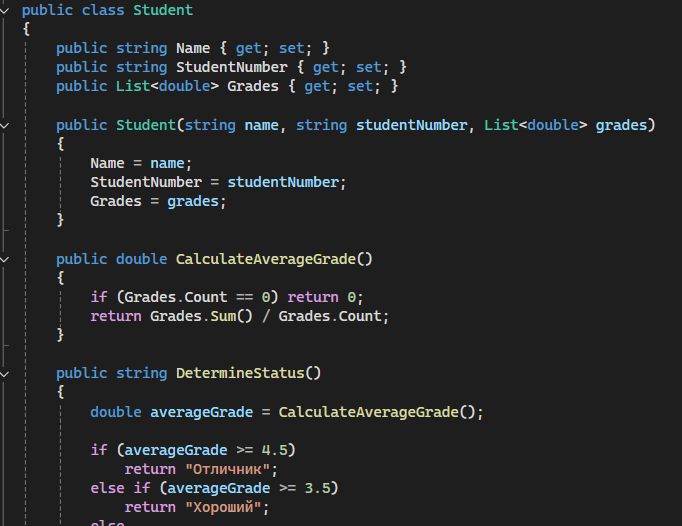
 Рисунок 3.24 – Решение задания №3



Рисунок 3.25 – Решение задания №3

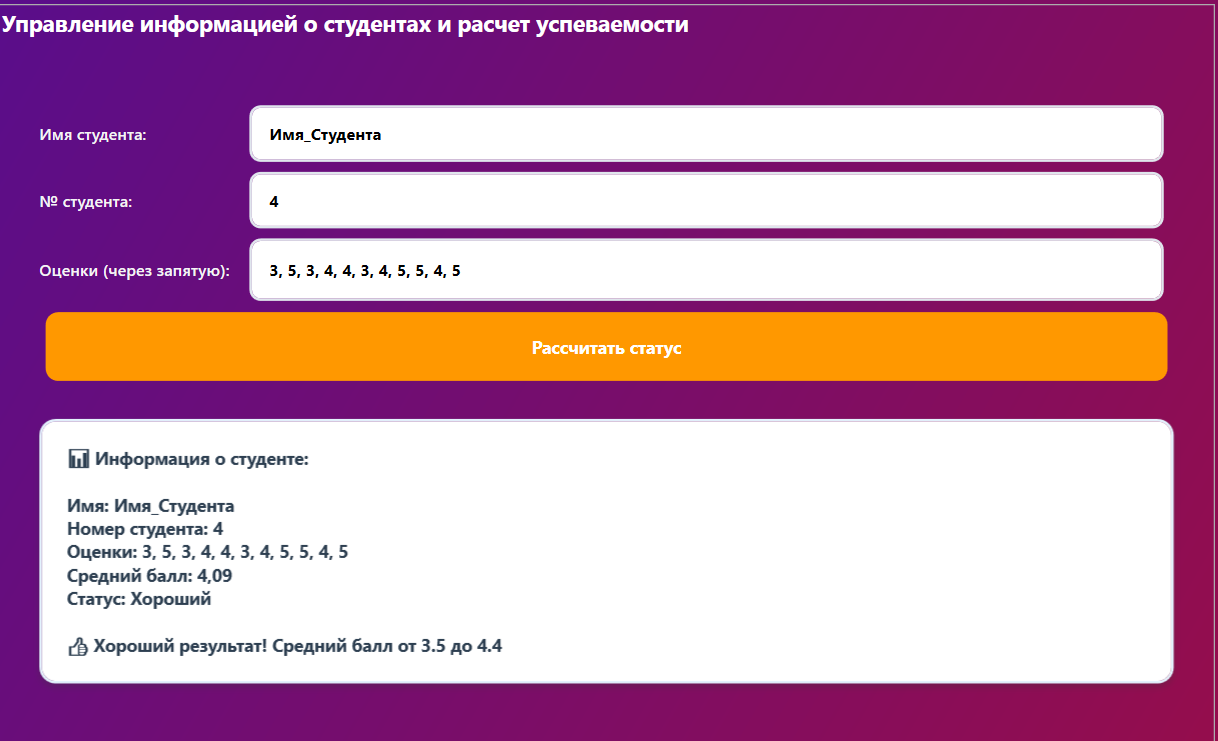


Рисунок 3.26 – Реализация «Task3Page»

Далее переходим к реализации разметки «Task4Page».

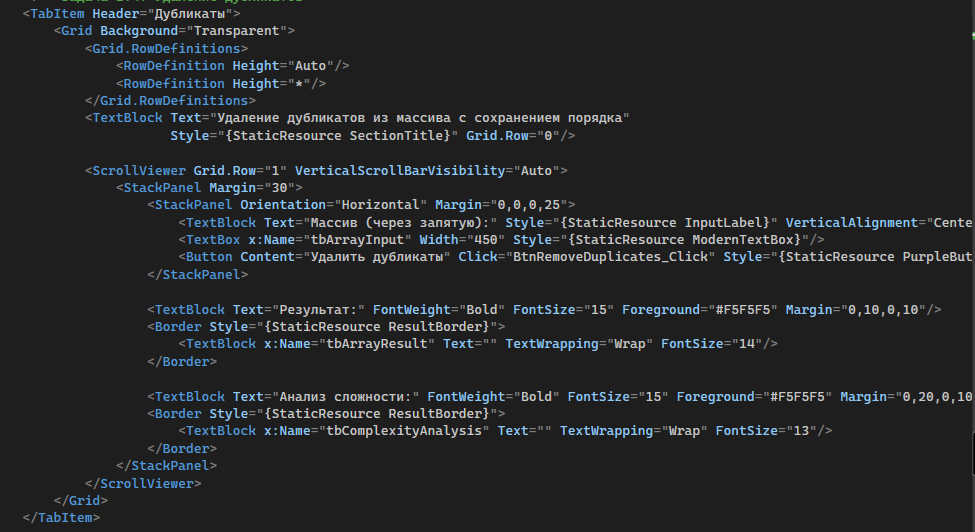


Рисунок 3.27 – Разметка «Task4Page»

Для реализации логики решения задания переходим в файл «Task4Page.xaml.cs».

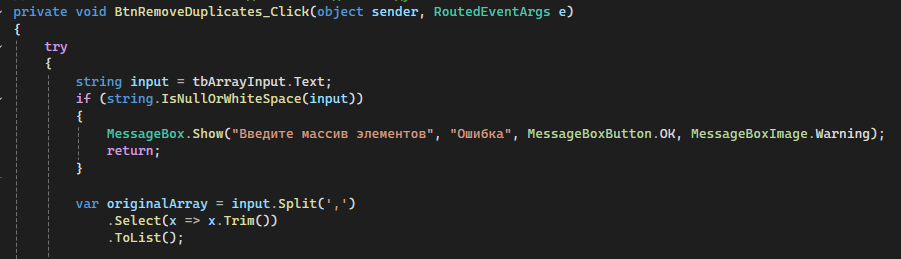


Рисунок 3.26 – Решение задания №4

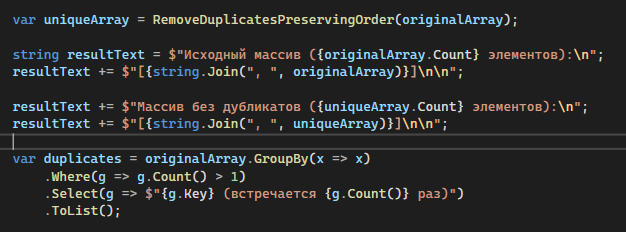


Рисунок 3.27 – Решение задания №4

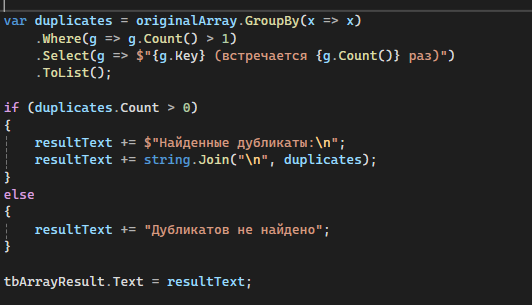


Рисунок 3.28 – Решение задания №4

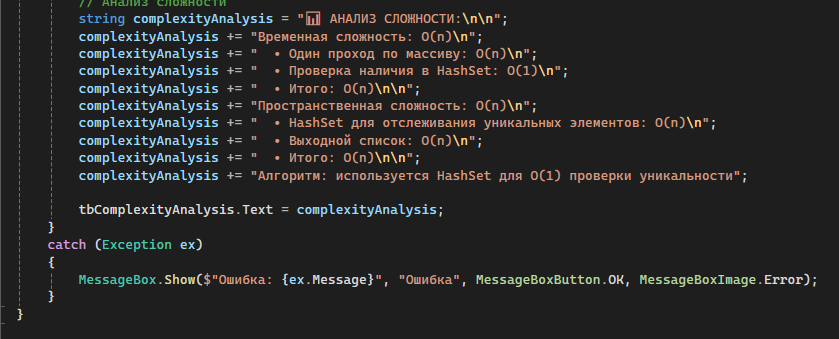


Рисунок 3.29 – Решение задания №4

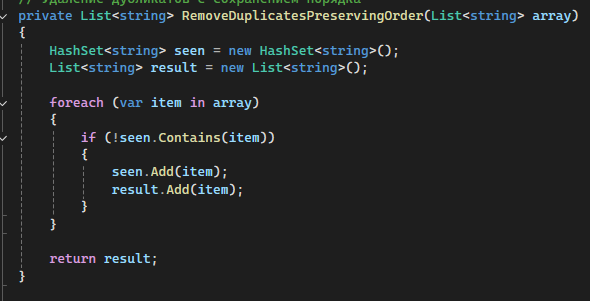


Рисунок 3.30 – Решение задания №4



Рисунок 3.31 – Реализация «Task4Page»

Далее переходим к реализации разметки «Task5Page».

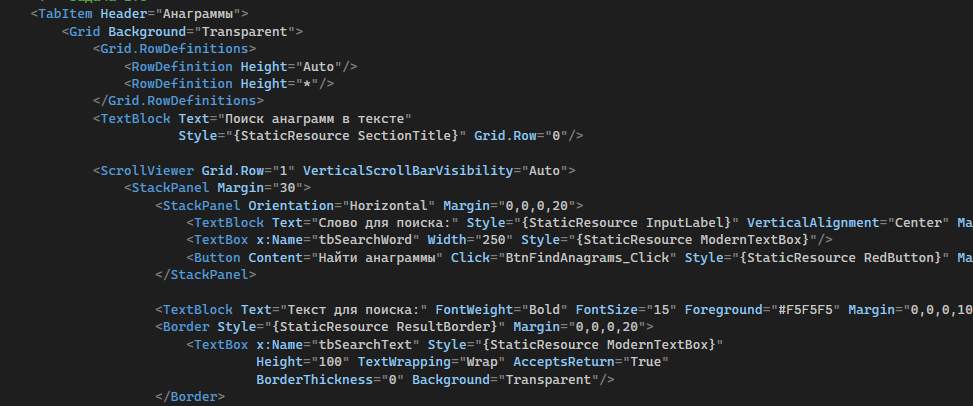
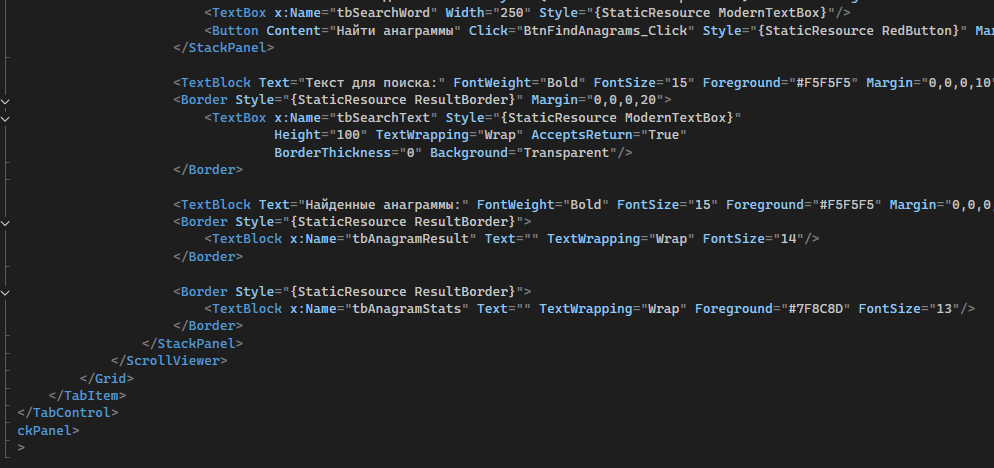


Рисунок 3.32 – Разметка «Task5Page»

 Рисунок 3.33 – Разметка «Task5Page»

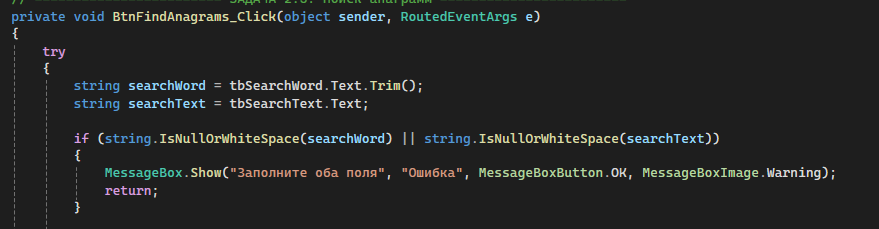


Рисунок 3.34 – Решение задания №5

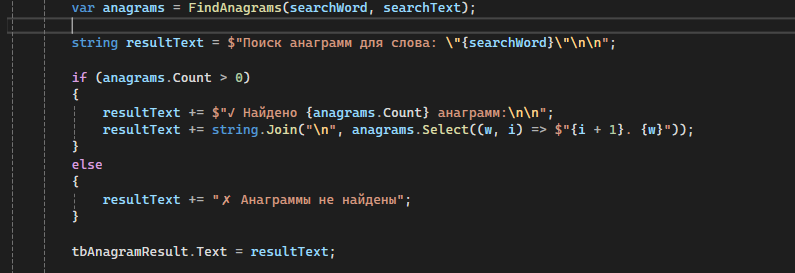


Рисунок 3.35 – Решение задания №5

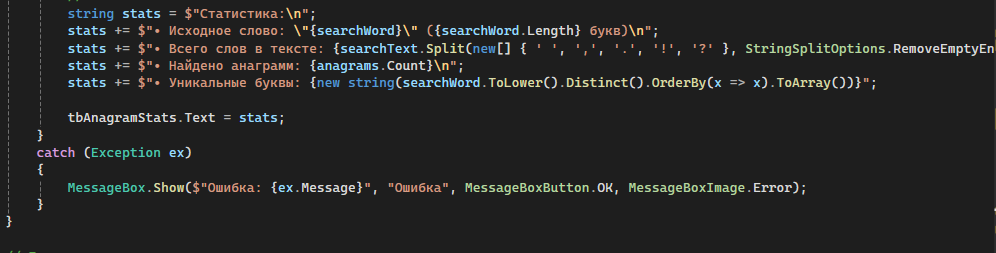


Рисунок 3.36 – Решение задания №5

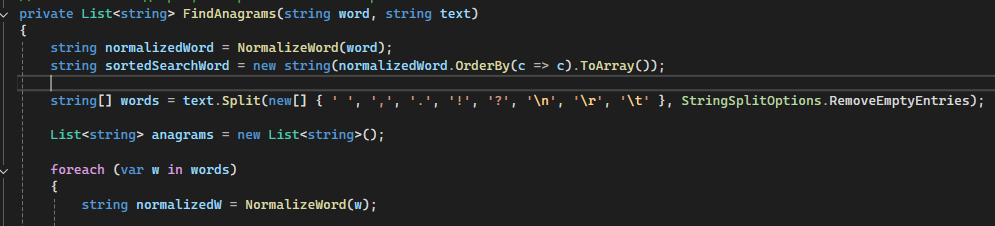


Рисунок 3.37 – Решение задания №5

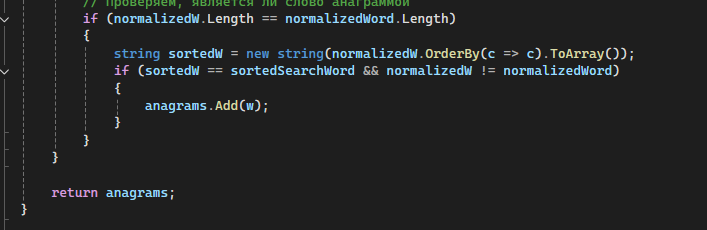


Рисунок 3.38 – Решение задания №5



Рисунок 3.39 – Решение задания №5

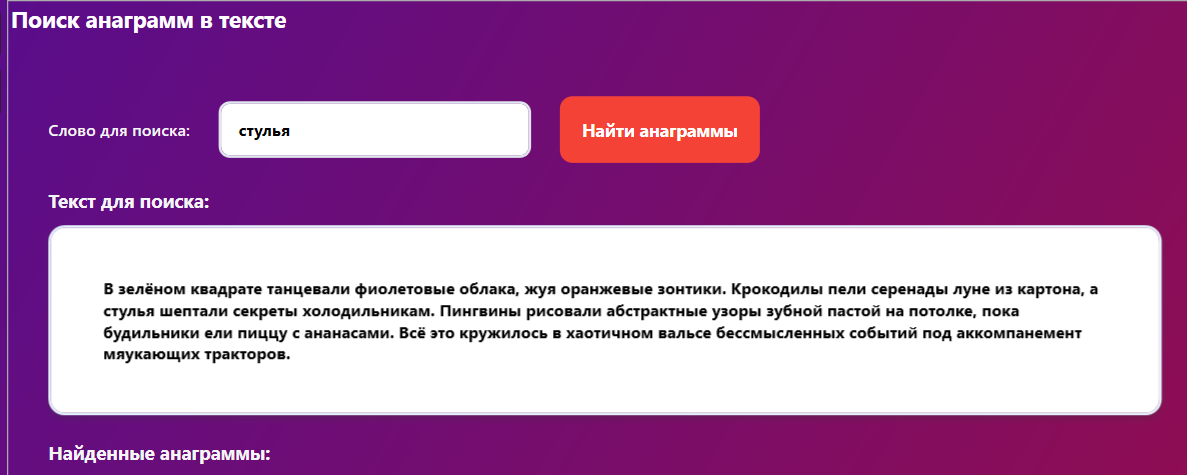


Рисунок 3.40 – Реализация «Task5Page»



Рисунок 3.41 – Реализация «Task5Page»

### ЗАКЛЮЧЕНИЕ

В результате выполнения данной учебной практики было успешно спроектировано и реализовано комплексное приложение на платформе Windows Presentation Foundation (WPF) с использованием языка программирования C#. Разработанное решение полностью соответствует поставленным целям и задачам, продемонстрировав практическое применение основных концепций современного программирования.

Практика позволила закрепить и углубить знания в области объектно-ориентированного программирования. При разработке приложения были успешно применены принципы инкапсуляции, полиморфизма и наследования. Разработанная архитектура проекта с логическим разделением кода на папки Core, View и AppStyle обеспечивает удобство сопровождения и возможность дальнейшего расширения функциональности. Особое внимание было уделено созданию класса Student, который демонстрирует правильное использование свойств, конструкторов и методов для работы с объектами.

Работа над интерфейсом приложения способствовала расширению навыков в области разработки графических пользовательских интерфейсов. Использование технологии XAML позволило эффективно разделить логику приложения от её визуального представления, что является одним из главных преимуществ WPF. Интеграция библиотеки Material Design придала приложению современный и профессиональный внешний вид, соответствующий актуальным стандартам дизайна. Разработанная система стилей в MainStyle.xaml обеспечивает единообразное оформление всех элементов интерфейса и позволяет быстро изменять внешний вид приложения без дублирования кода.

Особую ценность имеет реализация пяти различных функциональных модулей, каждый из которых решает отдельную прикладную задачу. Первый модуль, посвящённый решению систем линейных уравнений методом Крамера, продемонстрировал глубокое понимание математических алгоритмов и их программной реализации. Второй модуль с конвертацией чисел между различными системами счисления показал навыки работы с побитовыми операциями и числовыми алгоритмами. Третий модуль со структурой Student иллюстрирует способность разработчика создавать практические решения для обработки данных об учащихся. Четвёртый и пятый модули, посвящённые удалению дубликатов и поиску анаграмм, демонстрируют владение алгоритмами обработки строк и коллекций данных.

В процессе выполнения практики особенное внимание было уделено обработке ошибок и валидации пользовательского ввода. Приложение корректно обрабатывает различные типы исключений, выводя информативные сообщения об ошибках. Это гарантирует стабильную работу программы даже при некорректном вводе данных пользователем. Тестирование всех функциональных модулей проводилось систематически, что позволило убедиться в корректности реализованных алгоритмов.

Приобретённые в ходе практики умения работы с Visual Studio 2022, управления пакетами через NuGet и использования систем контроля версий явились важным дополнением к теоретическим знаниям. Практический опыт организации структуры проекта, создания многоуровневой архитектуры и взаимодействия различных компонентов приложения будет полезен при работе над более сложными проектами в будущем.

Результаты практики позволяют сделать вывод о достижении всех поставленных целей. Разработанное приложение может быть использовано как в учебных целях для демонстрации основных концепций программирования на платформе .NET, так и в качестве основы для дальнейшего расширения функциональности. Возможные направления развития включают добавление новых математических функций, расширение системы анализа данных, внедрение сохранения результатов в файлы различных форматов и создание дополнительных модулей для обработки информации.