Министерство образования и науки Российской Федерации

ФГАОУ ВО «КФУ им. В.И. Вернадского»

физико-технический институт (структурное подразделение)

|  |
| --- |
|  |

Кафедра компьютерной инженерии и моделирования

Порозов Кирилл Сергеевич

отчет по практической работе №1  
по дисциплине **«ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»**

09.03.04 "Программная инженерия"

Оценка



Симферополь, 2023

**Практическая работа №1.   
Тема: Использование программных конструкций C#**

**Цель работы:** Научиться создавать простейшие консольные и WPF приложения на языке C# в среде Visual Studio, изучить возможность создания самодокументируемых приложений.

Научиться преобразовывать различные типы данных в C#, познакомиться с типом данных Decimal, научиться грамотно использовать циклы для итерационных вычислений с контролем погрешности, обрабатывать события нажатия клавиш, научиться использовать классы String, StringBuilder, научиться создавать самодокументируемые XML справочные файлы.

**Описание ключевых понятий:**   
**IDE** -интегрированная среда разработки IDE (Integrated Development Envirionment) Visual Studio,   
**CLR** -общеязыковая исполнительная среда CLR (Common Langvidge Runtime, виртуальная машина), обеспечивает выполнение сборки и является основным компонентом .NET Framework. Под Виртуальной Машиной понимают абстракцию операционной системы высокого уровня, которая обеспечивает выполнение (управляемого) программного кода. Управляемый код – программный код, который при своём выполнении способен использовать службы, предоставляемые CLR. Сама CLR состоит из двух главных компонентов: ядра (mscore.dll) и библиотеки базовых классов FCL (mscorlib.dll) .   
**FCL** - .NET Framework class library - библиотека классов,   
**CLS** - общеязыковые спецификации CLS, решение (solution), проект (project),   
**namespace** - пространство имен , способ организации системы типов в единую группу. Концепция пространства имён обеспечивает эффективную организацию и навигацию по этой библиотеке. Вне зависимости от языка программирования доступ к определённым классам обеспечивается за счёт их группировки в рамках общих пространств имён, сборка (assembly,  
**IL -**  MSIL (Microsoft Intermediate Language, IL), управляемый код, двухэтапная компиляция,  
**WPF** - **Windows Presentation Foundation** — аналог WinForms, система для построения клиентских приложений [Windows](https://ru.wikipedia.org/wiki/Windows) с визуально привлекательными возможностями взаимодействия с пользователем, графическая подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML. В основе WPF лежит векторная система визуализации, основанная на DirectX.

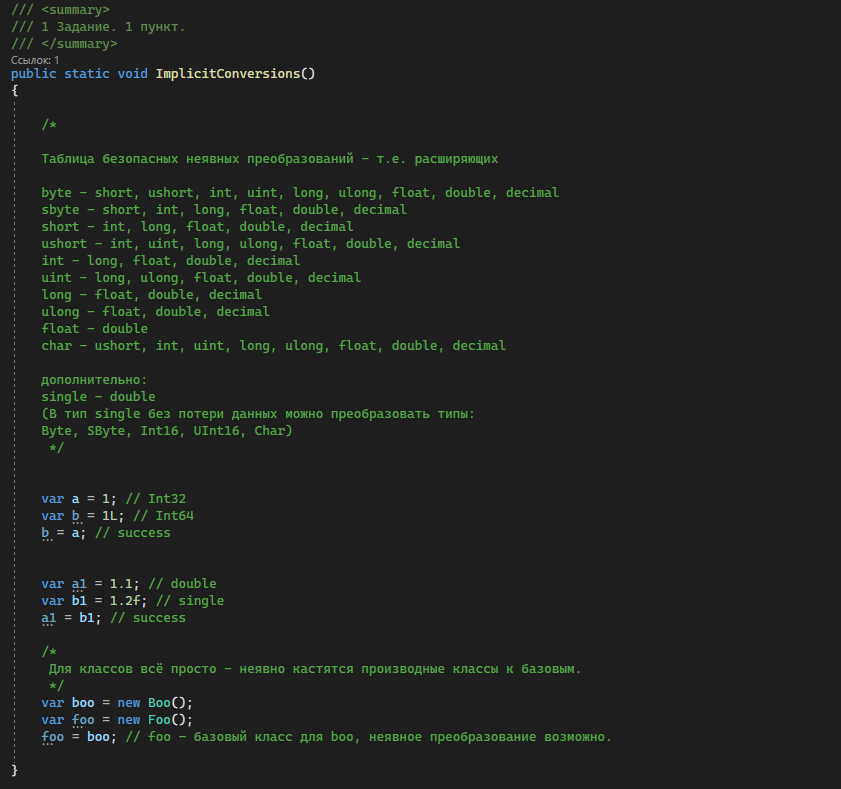
**Перед выполнением лабораторной работы изучена следующая литература:**

1. Презентация лектора курса: «Основы Net Framework»
2. Прослушана видеолекция Использование конструкций языка С#.
3. Прочитаны 1-2 лекции (Использование конструкций языка С# и Использование конструкций языка С#)
4. Получены начальные сведения о спецификации языка C#.
5. Изучены “Глава 1. Введение в C#” сайта Metanit.com.

**Выполнены 4 задания, описанных в методических указания к выполнению лабораторных работ.   
Задание 1: Приведение и преобразование типов**

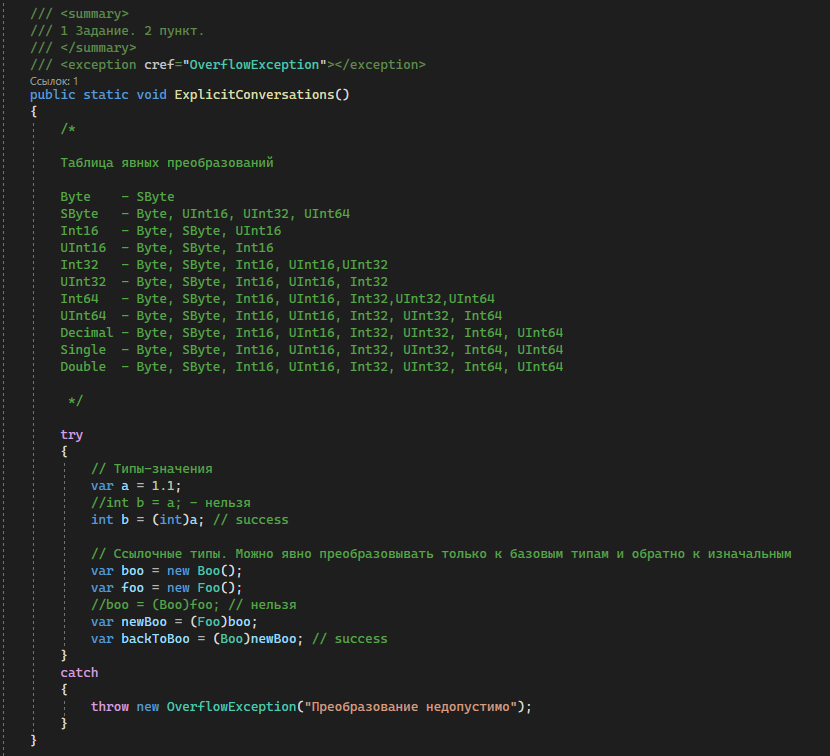
1. Неявное преобразование простых и ссылочных типов, в виде комментариев внести в программу таблицу неявных преобразований;

Неявное приведение ссылочных типов

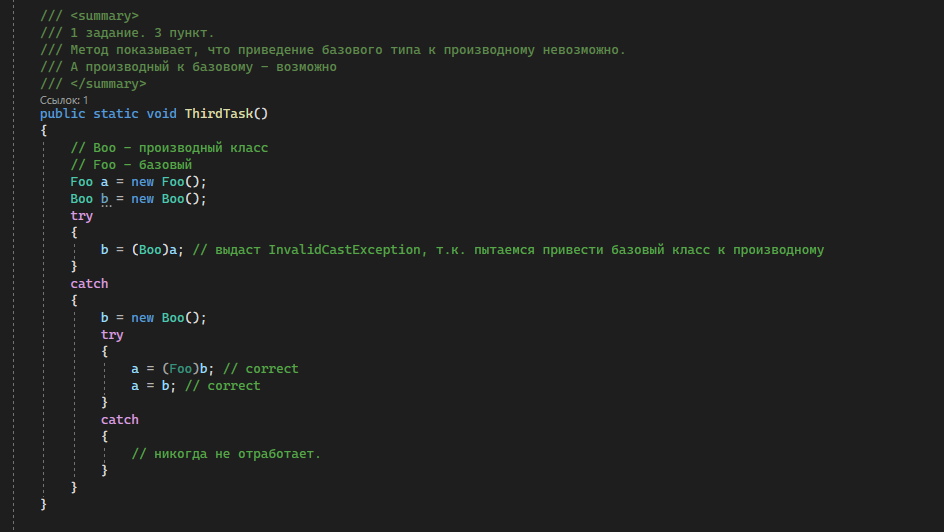


Неявное преобразование простых типов

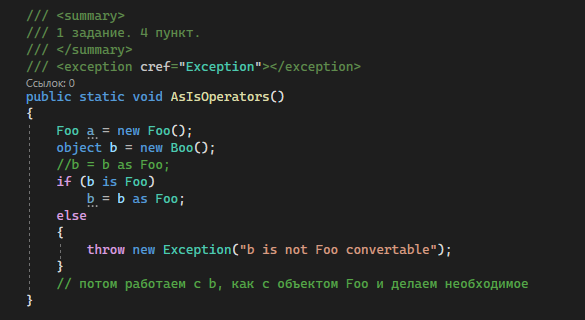
1. Явное преобразование простых и ссылочных типов, в виде комментариев внести в программу таблицу явных преобразований;



1. Вызвать и обработать исключение преобразования типов;

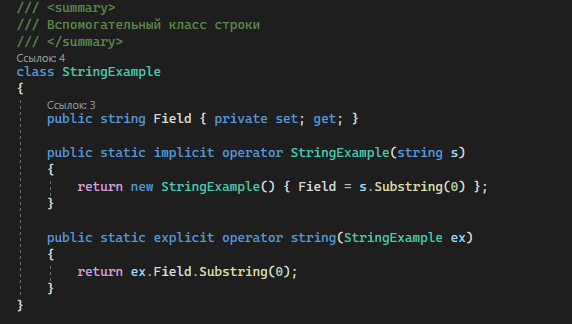


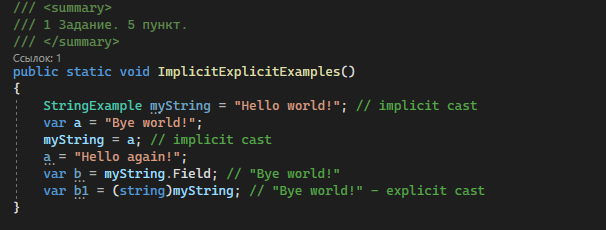
1. Безопасное приведение ссылочных типов с помощью операторов as и is;



В данной примере класс Boo – **ПРОИЗВОДНЫЙ** от Foo => ошибки не возникает и при использовании оператора is к базовому классу – он выдает **True,** при использовании оператора as преобразование к базовому типу возможно.

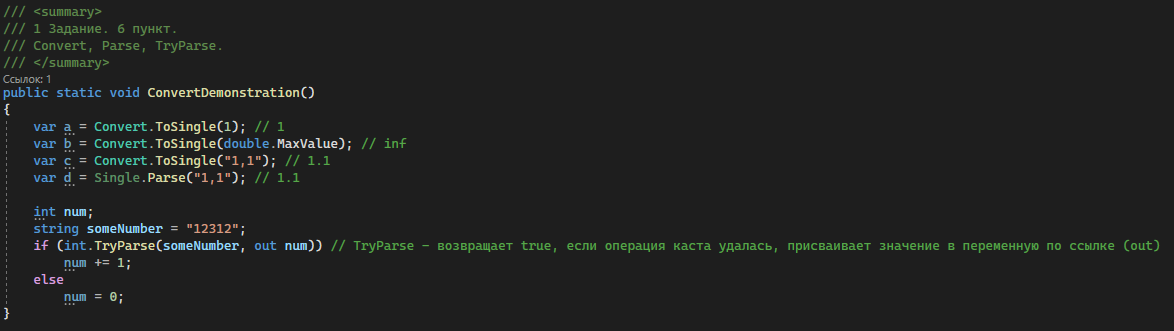
1. Пользовательское преобразование типов Implicit, Explicit;





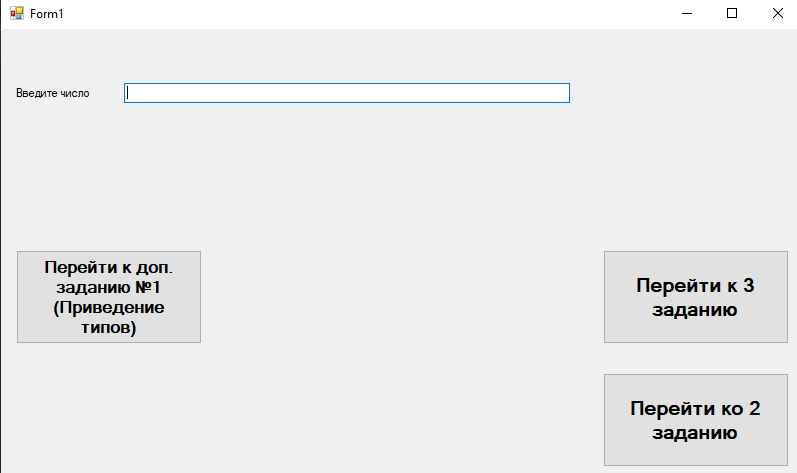
Присваеваем созданному нами классу StringExample значение строки, переопределив оператор implicit, принимающий строку, ошибки не получаем. Приведя наш произвольный тип явным образом к строке – получаем значение свойства Field.

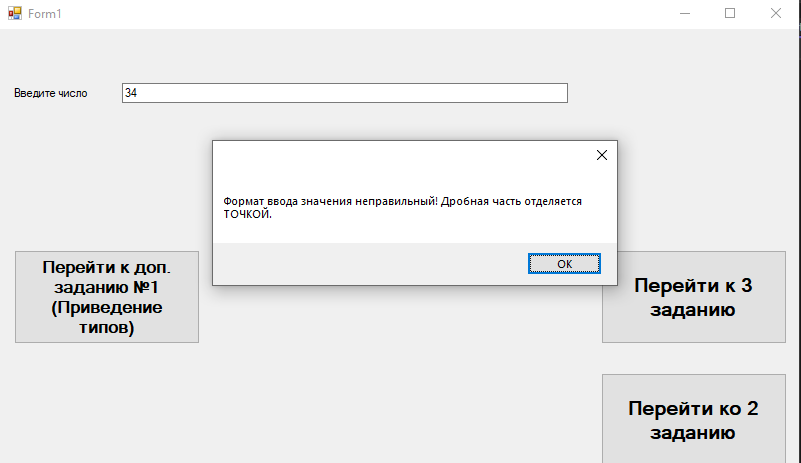
1. Преобразование с помощью класса Convert и преобразование строки в число с помощью методов Parse, TryParse класса System.Int32.



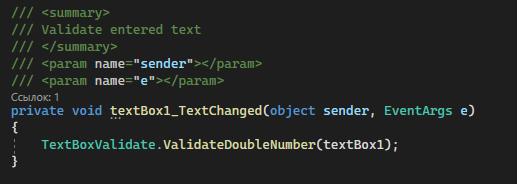
1. Написать программу, позволяющую ввод в текстовое поле TextBox только символов, задающих правильный формат вещественного числа со знаком.

Для TextBox переопределен метод TextChanged – при каждом изменении строки, если она не может быть приведена к типу float – следует открытие диалогового окна с предупреждением и автоматическим стиранием написанного символа

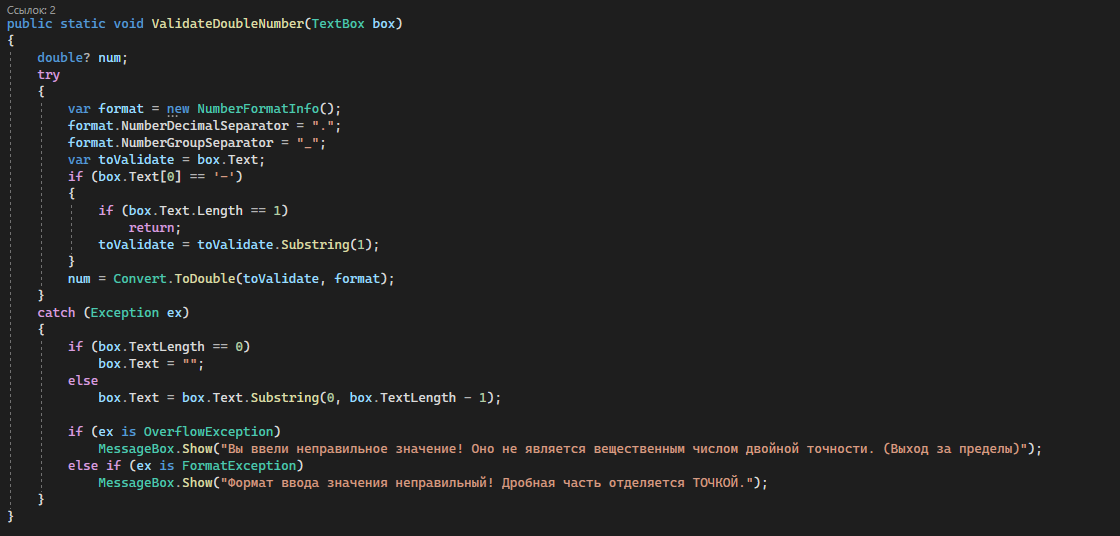




Метод, осуществляющий задачу



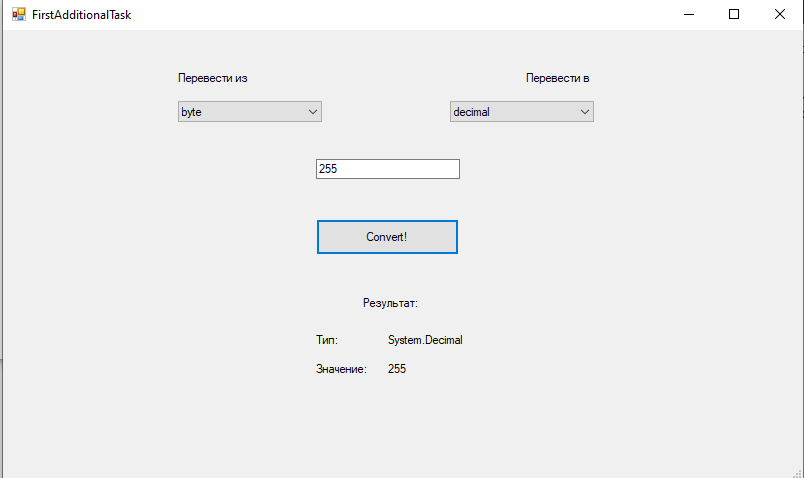
Сам метод

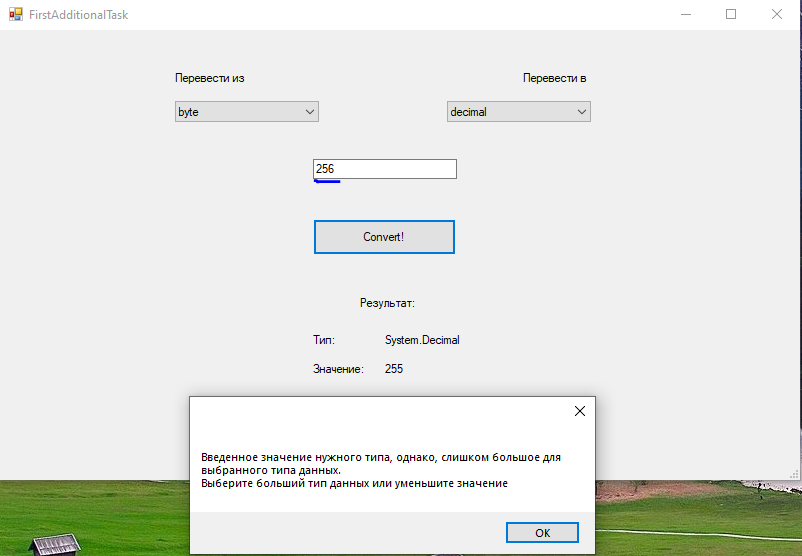


С работой данной программы можно ознакомиться, а также проверить её на github, ссылка в конце отчета.

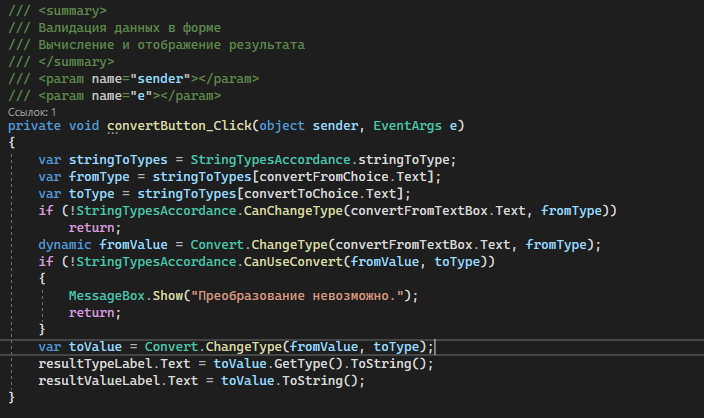
**Задание 1. Дополнительное**

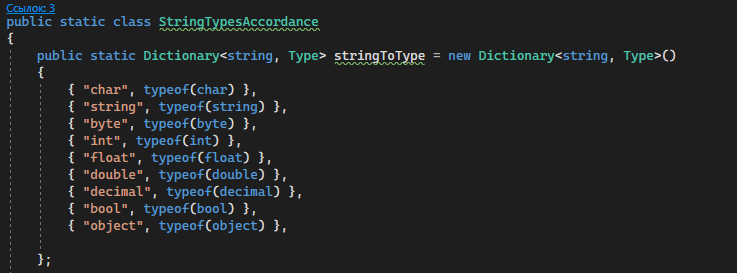
Создана программа, переводящая любой тип данных в любой другой. С соответствующей валидацией. Пример работы:

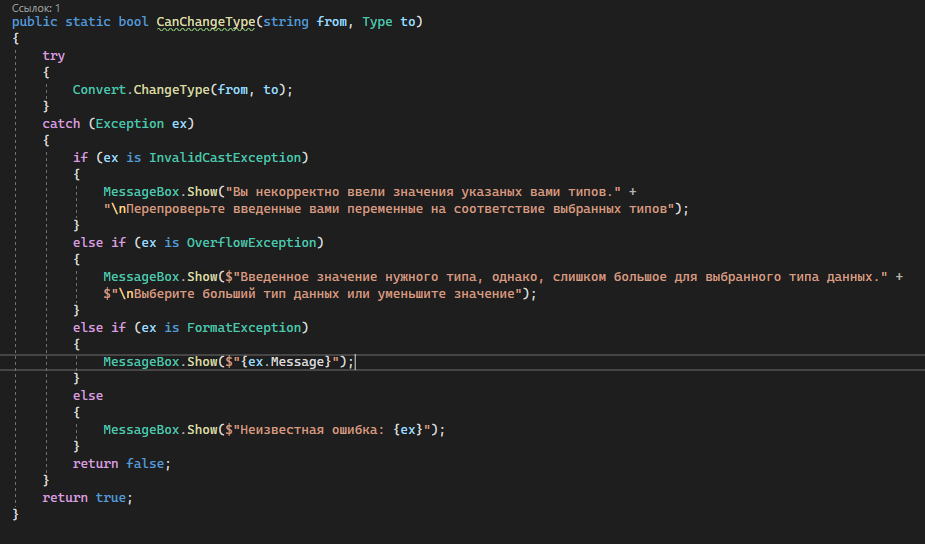


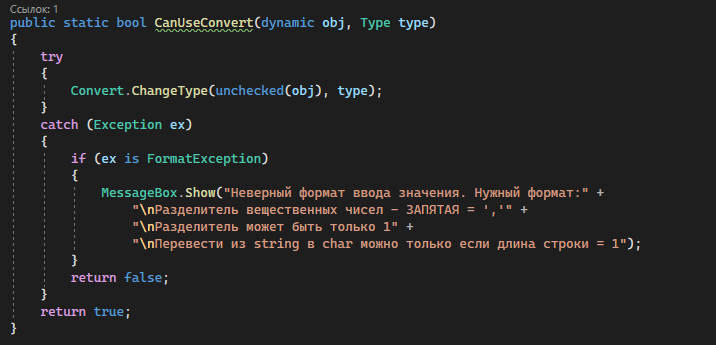


Код:

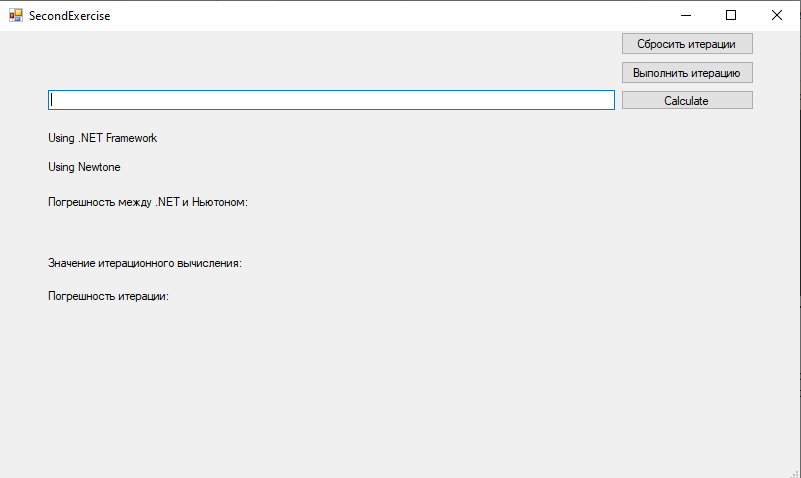








**Задание 2: Вычислить квадратный корень с контролем точности**

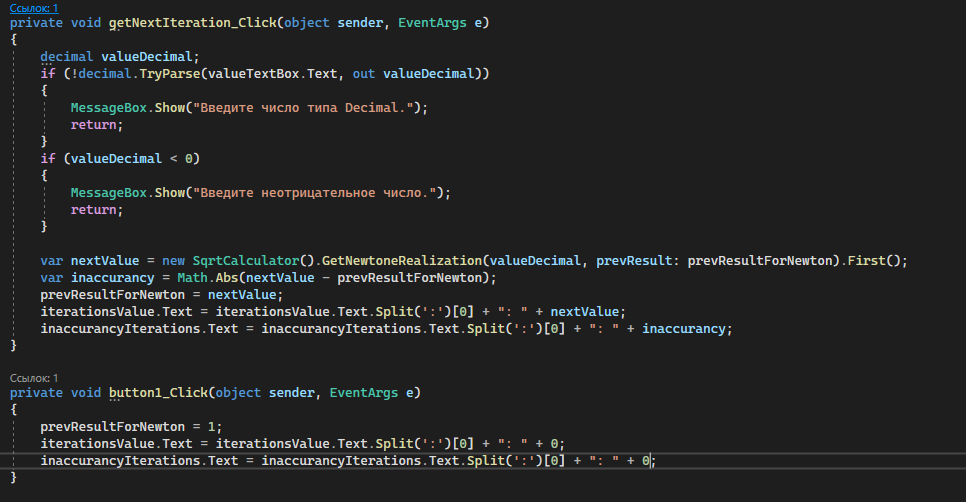


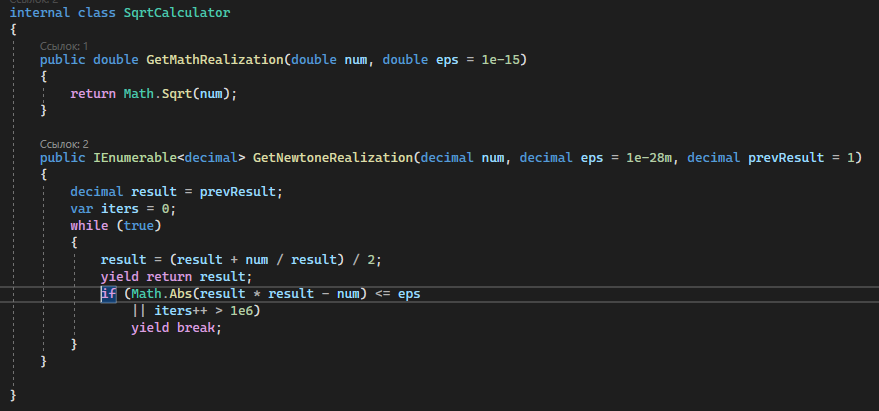
Итерационный алгоритм работает независимо от основного. Это сделано для удобного сравнения алгоритмов.

Внешний вид программы. Кнопка Calculate рассчитывает значение из textBox по Math.Sqrt() и Ньютону, а кнопка Выполнить итерацию выводит

значения корня, а также погрешность на ней.

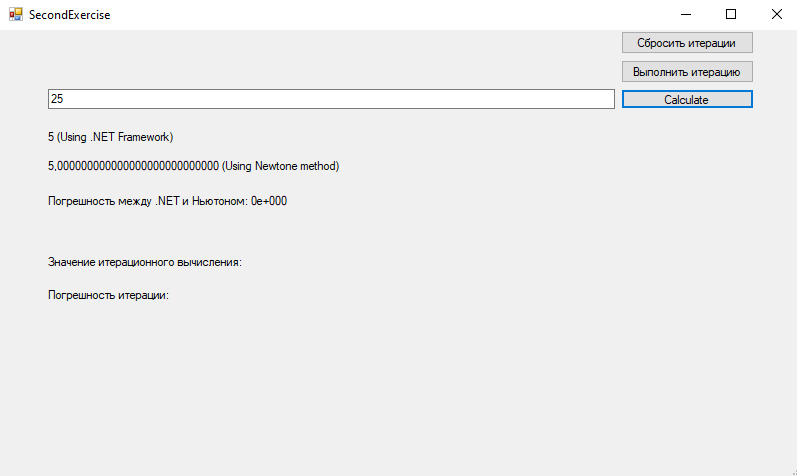


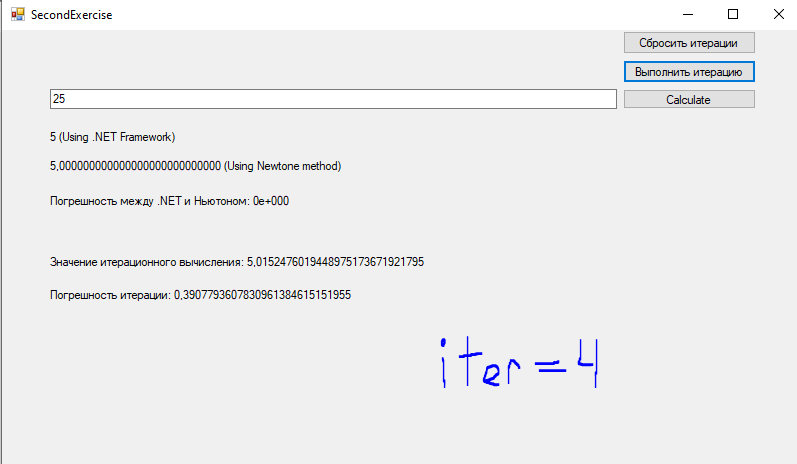
SecondExercise.cs

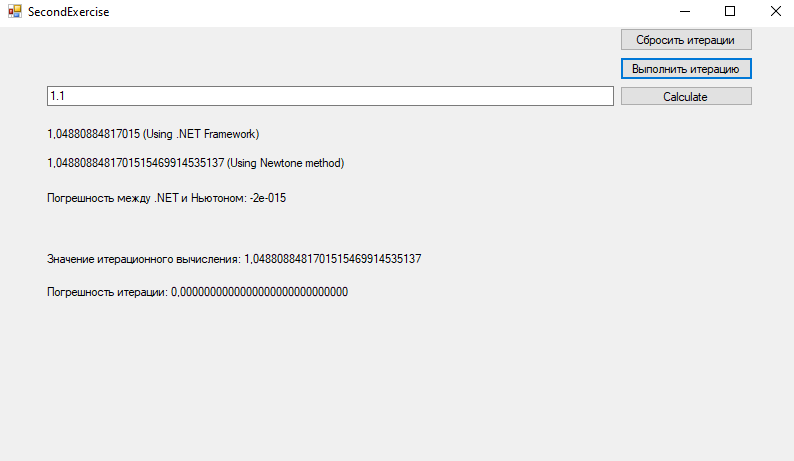


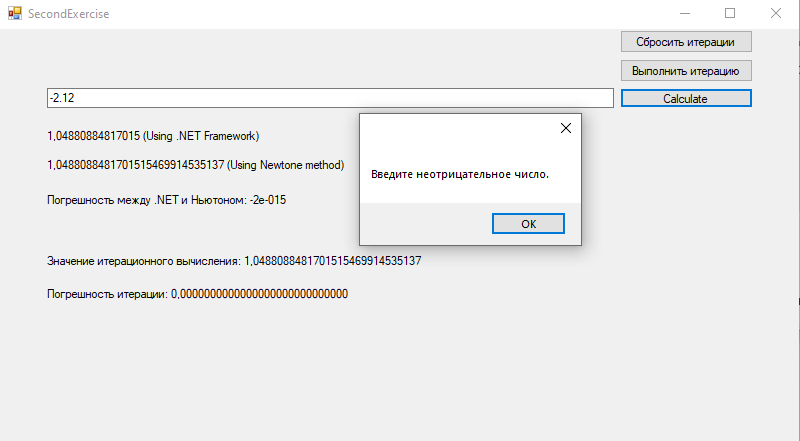
Метод GetMathRealization, срабатывающий при нажатии на кнопку Вычислить. Вычисляет корень квадратный введенного числа в textBox по MathSqrt. GetNewtoneRealization – ленивый метод, возвращающий IEnumerable для текущей итерации, зная предыдущую. Для проверки значений используются 2 условия – либо отличие вычисления более чем на 1е-28, либо превышение кол-ва итераций (>1e6).

Тест приложения:

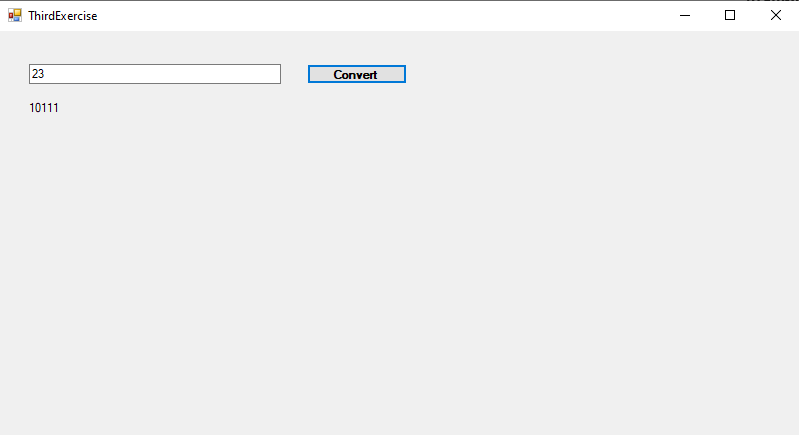








**Задание 3. Конвертация целочисленных данных в бинарные**



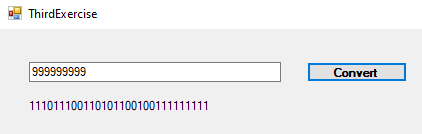
Внешний вид приложения



Программный код

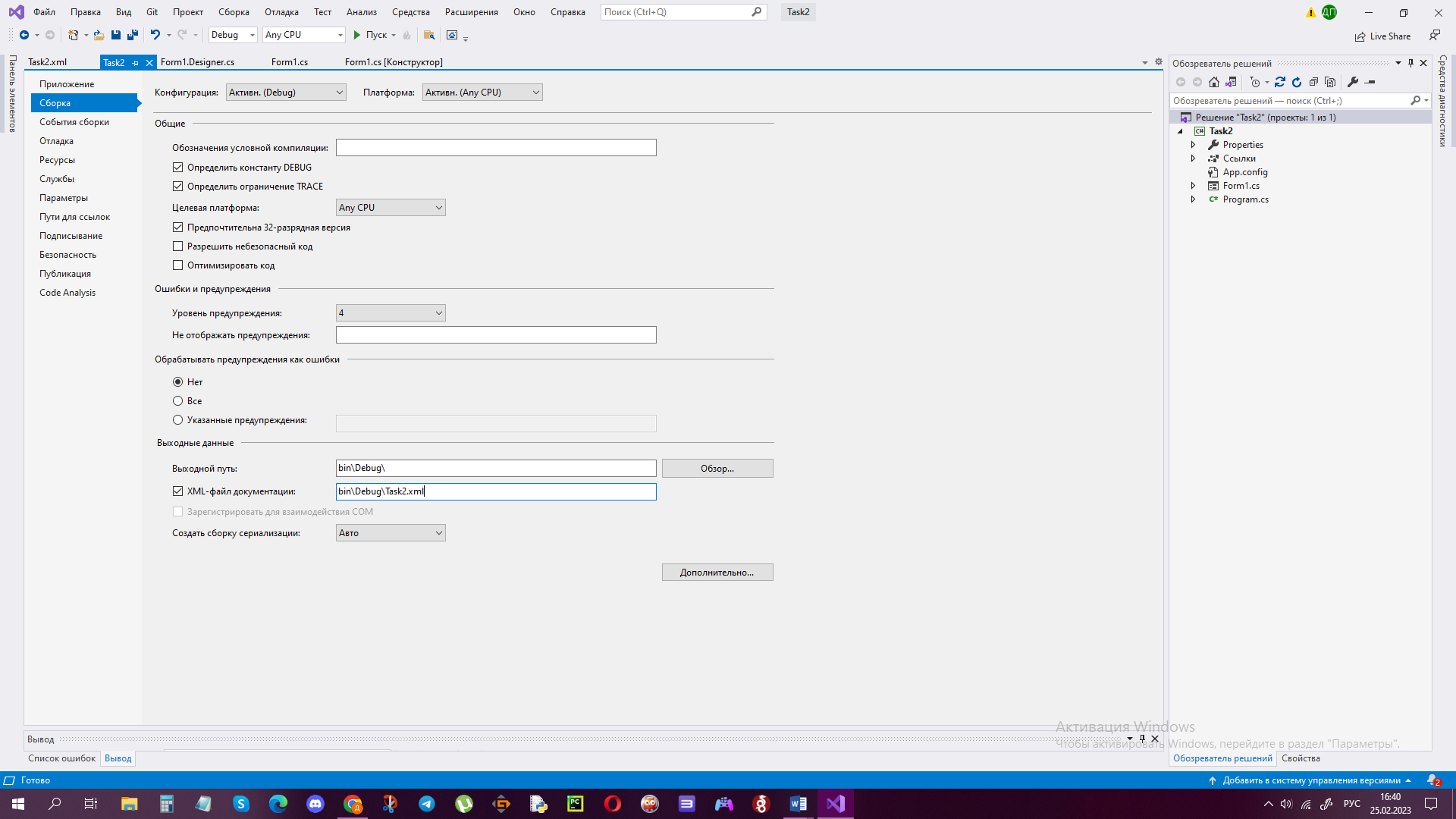
Тесты программы:



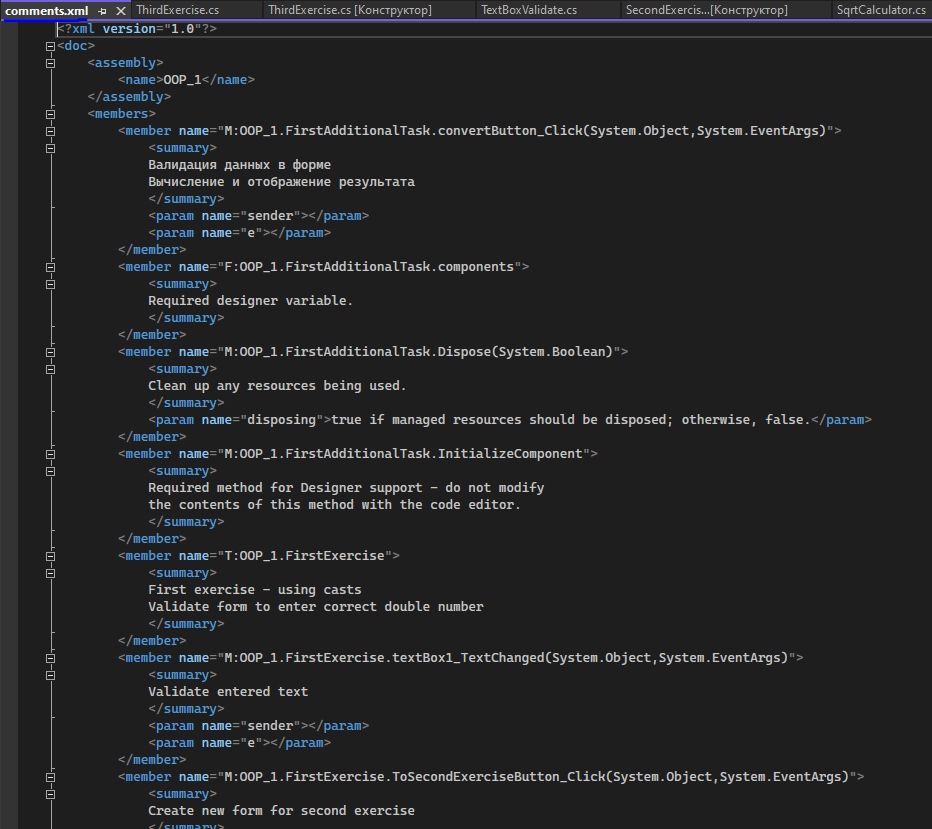


**Задание 4. Создание документации для приложения**

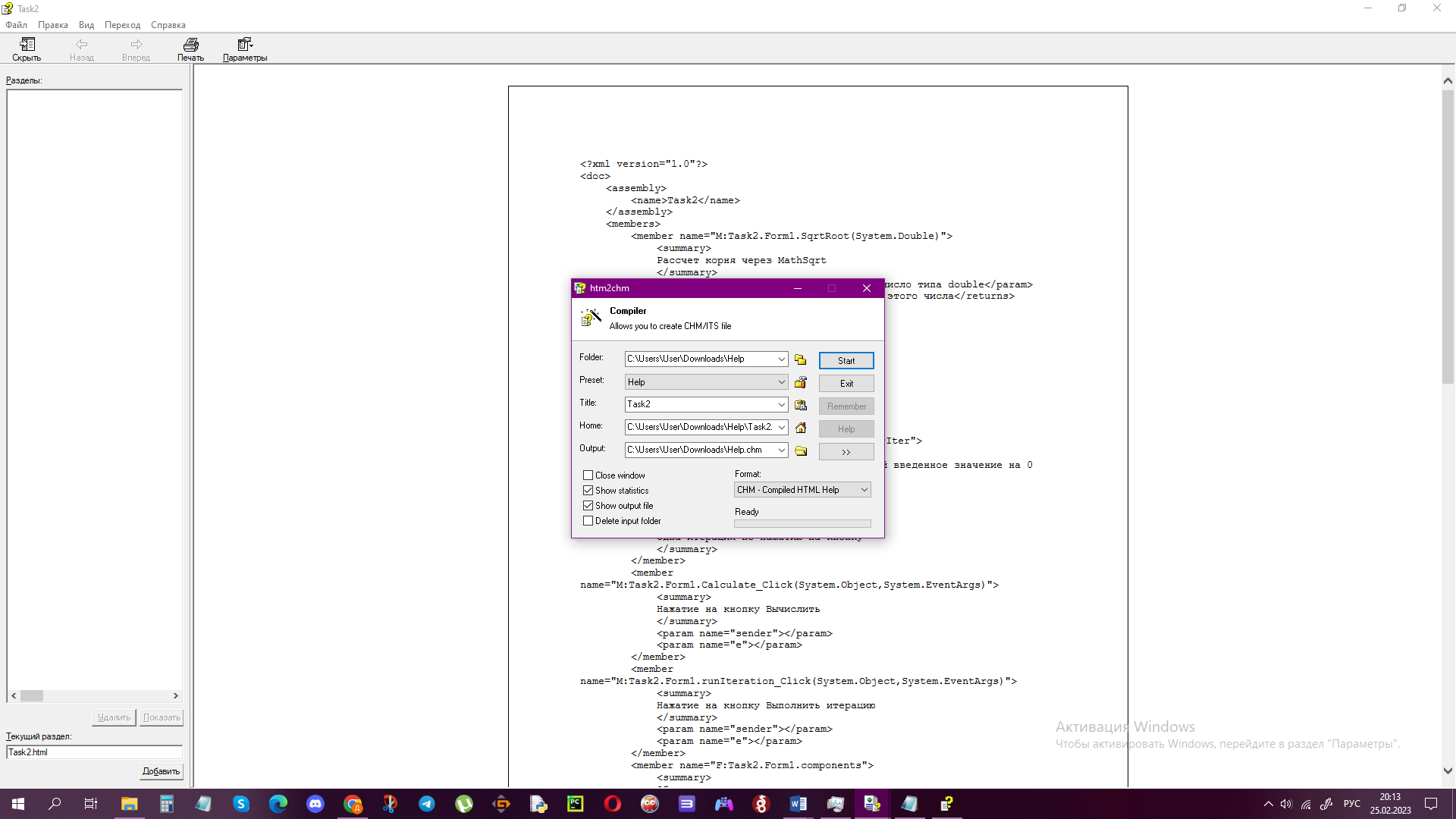
Добавление xml комментариев



Генерация xml файла



**Задание 4 (бонус 1 балл)**: Создать многостраничный .chm файл из справочного .xml файла с помощью любого доступного пакета со свободной лицензией, например: CHM Builder, PowerCHM, HTML Help Workshop, HTM2CHM.



**Представлены 4 проекта, реализованных в Visual Studio Community 2019.**

**Проекты представлены преподавателю в электронной форме, продемонстрирована их работоспособность, разъяснены детали программного кода.**

<https://github.com/JustBlood/OOP_1> - ссылка на решение