ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА СЕВАСТОПОЛЯ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ ГОРОДА СЕВАСТОПОЛЯ «СЕВАСТОПОЛЬСКИЙ ТОРГОВО-ЭКОНОМИЧЕСКИЙ ТЕХНИКУМ»

**ОТЧЕТ**

**по учебной практике**

студента 3 курса, группы ИСП 9-2,

очной формы обучения

Протасевич Александр Александрович *(Ф.И.О. обучающегося)*

специальность 09.02.07 Информационные системы и программирование

Руководитель практики Кучеренко Я. А.

Начало практики «12» мая 2025 г.

Окончание практики «16» мая 2025 г.

Севастополь, 2025 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Учебная практика ПМ 02 «Ревьюирование программных модулей» занимает важное место в подготовке специалистов в области информационных технологий и программирования, обеспечивая формирование профессиональных компетенций, необходимых для разработки, оптимизации и оценки качества программного обеспечения. В ходе практики обучающиеся применяют теоретические знания на практике, развивают навыки самостоятельной работы и осваивают современные подходы к ревьюированию и тестированию программного кода. Основными целями практики являются систематизация, закрепление и расширение теоретических и практических знаний по выбранной специальности, развитие навыков самостоятельной работы и овладение методикой исследования при решении практических задач, а также определение уровня подготовки обучающегося и его способности применять знания для выполнения конкретных профессиональных задач.

Для достижения этих целей в рамках практики решаются задачи, связанные с изучением и работой с проектной документацией, разработанной с использованием графических языков спецификаций, выполнением оптимизации программного кода с применением специализированных инструментов, использованием методов и технологий тестирования и ревьюирования кода и документации, применением стандартных метрик для прогнозирования затрат, сроков и качества разработки, изучением принципов планирования и контроля развития программного проекта, а также современных стандартов качества программных продуктов. Кроме того, важной задачей является оформление отчета по учебной практике в соответствии с установленными нормативными требованиями.

Ревьюирование программного кода представляет собой процесс систематической проверки исходного кода, направленный на выявление ошибок, повышение качества программного продукта и обеспечение его соответствия стандартам разработки. Этот процесс включает анализ кода на предмет корректности, читаемости, производительности и безопасности, который может выполняться как индивидуально, так и в команде, с использованием автоматизированных инструментов или вручную. Ревьюирование играет ключевую роль в разработке программного обеспечения, поскольку позволяет выявлять и устранять ошибки на ранних стадиях, снижая затраты на исправление дефектов в дальнейшем. Оно способствует улучшению читаемости и поддерживаемости кода, упрощая его сопровождение, а также обеспечивает обмен знаниями между участниками команды, что способствует их профессиональному росту. Кроме того, ревьюирование гарантирует соблюдение корпоративных и отраслевых стандартов качества и помогает снизить риски, связанные с уязвимостями и потенциальными проблемами, повышая надежность и безопасность программного обеспечения.

Таким образом, учебная практика ПМ 02 «Ревьюирование программных модулей» не только закрепляет теоретические знания, но и формирует практические навыки, необходимые для профессиональной деятельности в области разработки программного обеспечения. В рамках данной практики будет выполнена задача по разработке класса «Дробное число со знаком» (Fractions) с реализацией арифметических операций и операций сравнения, а также проведено тестирование разработанных методов. Отчет по практике содержит описание выполненной работы, анализ результатов и выводы, подтверждающие достижение поставленных целей и задач.

1. МОДЕЛИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
   1. Анализ предметной области

Изучение предметной области включает рассмотрение основных аспектов, необходимых для эффективного выполнения задачи. В первую очередь это освоение интегрированной среды разработки (IDE), которая является ключевым инструментом для создания, отладки и тестирования программного кода. Для выполнения работы применяется Visual Studio (см. Таблицу 1).

Таблица 1 – Характеристика Visual Studio 2022

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Возможности | Производительность | Простота использования | Документация |
| Средства тестирования | Встроенные утилиты для проведения модульных и автоматических тестов, совместимость с фреймворками xUnit, NUnit, MSTest, опция запуска и проверки кода в среде IDE. | Высокая — ускоренный процесс выполнения и запуска тестов. | Высокая — эргономичный и понятный интерфейс. | Детализированные материалы от Microsoft доступны на docs.microsoft.com. |
| Разработка баз данных | Средства для работы с SQL Server, Azure SQL, Entity Framework, создание схематических представлений баз данных, редактор SQL-запросов, управление обновлениями и анализ различий структур. | Высокая — надёжная интеграция с SQL Server. | Средняя — требуется базовое понимание SQL. | Полный обзор функций представлен на MSDN и Learn.microsoft.com |
| Отладка и диагностика | Расширенные инструменты для устранения ошибок: точки остановки, мониторинг переменных, трассировка вызовов, оценка производительности, контроль памяти и потоков. | Высокая — точное и оперативное выполнение. | Высокая — интуитивно удобный дизайн интерфейса. | Подробные инструкции по отладке и анализу доступны на Learn.microsoft.com. |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поддержка платформы разработки | Поддерживает широкий спектр платформ: .NET Framework, .NET Core, ASP.NET, WPF, WinForms, Xamarin, Unity, UWP и облачные сервисы Azure. | Высокая — отличная совместимость | Высокая — шаблоны проектов для разных платформ | Полная информация на сайте Microsoft Dev Center |
| Управление жизненным циклом приложений | Интеграция с Azure DevOps (бывший VSTS), Team Foundation Server (TFS), Git, CI/CD, управление задачами, контроль изменений, автоматизация сборок и тестирования. | Высокая — мощные инструменты управления | Средняя — требует обучения | Подробная документация по ALM и DevOps на learn.microsoft.com и azure.com |
| Интегрированная среда разработки | Единая платформа для разработки приложений под Windows, Web, Mobile, Cloud и т. д., с поддержкой C#, VB.NET, F#, C++,  Python, JavaScript и других языков. | Высокая — оптимальная загрузка проектов | Высокая — удобный UI, настраиваемые окна | Подробные руководства и туториалы на официальном сайте |

Для эффективной работы над проектом необходимо освоить основные понятия и термины, связанные с данной областью. Интегрированная среда разработки (IDE) представляет собой программную платформу, которая объединяет набор инструментов для создания программного обеспечения. Она включает текстовый редактор, компилятор или интерпретатор, отладчик, средства автоматизации сборки и, как правило, встроенные функции управления версиями.

Объектно-ориентированное программирование (ООП) — это подход к написанию кода, основанный на использовании объектов, объединяющих данные (атрибуты или свойства) и процедуры (методы). Этот подход опирается на ключевые принципы: инкапсуляция, позволяющая защитить внутреннюю логику объекта и ограничивать доступ через заданный интерфейс; наследование, которое позволяет создавать новые классы, расширяя возможности существующих; полиморфизм, обеспечивающий гибкость при использовании одного интерфейса для разных реализаций; и абстракция, выделяющая основные черты объекта, отвлекаясь от второстепенных деталей.

Конструктор — это особый метод класса, активируемый при создании экземпляра объекта, который служит для установки начального состояния. Он может принимать аргументы, задавать значения атрибутов и проверять правильность введённых данных. Деструктор, напротив, — метод, выполняемый перед удалением объекта, например, при завершении программы или освобождении памяти. В C# деструкторы применяются нечасто из-за автоматического управления памятью через Garbage Collector, но могут быть полезны для вывода диагностики или освобождения внешних ресурсов.

Ревью кода — это процедура анализа исходного кода другими разработчиками с целью выявления ошибок, повышения читаемости, соблюдения стандартов и улучшения общего качества продукта. Процесс может осуществляться вручную или с использованием специализированных платформ, таких как GitHub, поддерживающих системы контроля версий. Нотация языка — это совокупность символов и правил, используемых для формального описания структуры и поведения элементов программ. Например, UML (Unified Modeling Language) — это стандартный язык моделирования, применяемый для визуального представления, описания, проектирования и документирования программных систем.

* 1. Проектирование алгоритма

В ходе практической работы была поставлена цель разработать класс «Дробное число со знаком» (Fractions), представляющий число с использованием двух полей: целая часть, представленная длинным целым числом со знаком, и дробная часть, заданная беззнаковым коротким целым числом. Задача включала реализацию арифметических операций — сложения, вычитания и умножения, а также операций сравнения. Кроме того, в функции main было предусмотрено тестирование всех разработанных методов.

Для выполнения задачи был разработан алгоритм, включающий создание класса Fractions с указанными полями. Целая часть хранит знаковое значение, а дробная часть — положительное число с ограниченной разрядностью. Арифметические операции реализованы с учётом корректного учёта знака и обработки переполнения. Сравнение чисел выполняется на основе их числового представления, включая целую и дробную части.

Процесс вычислений для арифметических операций основывается на преобразовании дробных чисел в единый формат. Например, при сложении учитывается выравнивание разрядов дробной части, а при умножении результат округляется с сохранением заданной точности. Для проверки корректности работы алгоритма в функции main создаются экземпляры класса с различными значениями, и на них тестируются все реализованные операции.

Описание работы программы представлено в виде блок-схемы (Рисунок 1), которая иллюстрирует логику выполнения для двух ключевых сценариев. После запуска программы пользователь выбирает одно из действий. Если выбрано действие "1", программа запрашивает значения двух дробных чисел (каждое состоит из целой и дробной части), выполняет выбранную арифметическую операцию (сложение, вычитание или умножение) и выводит результат. Если выбрано действие "2", программа запрашивает значения двух дробных чисел и проводит их сравнение, отображая, какое число больше, меньше или равны ли они. При некорректном вводе данных пользователю выводится сообщение об ошибке. После выполнения выбранного действия программа завершает свою работу.

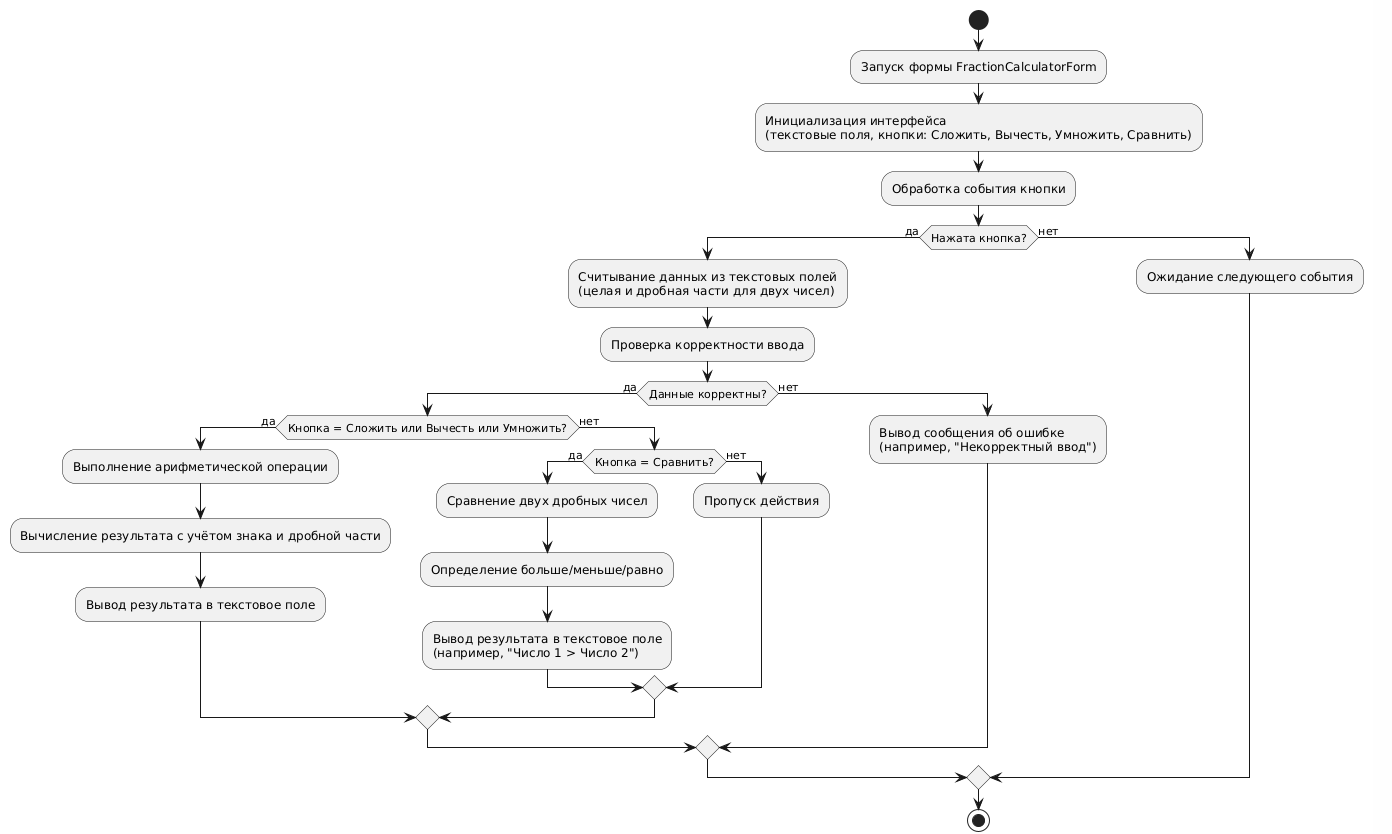


Рисунок 1 – Блок-схема

* 1. Разработка кода решения задачи

После создания алгоритма решения была начата реализация программы. В процессе разработки были соблюдены принципы ООП, эффективности и повторного использования имеющихся модулей. Совокупность этих принципов позволяет сделать качественный и понятный код.

Разработанный код представлен в приложении А.

1. АНАЛИЗ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Работа с GitHub

Для совместной работы над проектом и ревьюирования кода другими участниками был использован GitHub. Для начала его работы был создан репозиторий на сайте, затем при помощи программы GitHub декстоп он был клонирован локально на пк. Далее в репозитории был создан проект (см. рисунок 2), который в дальнейшем будет загружен на сайт для совместной разработки.

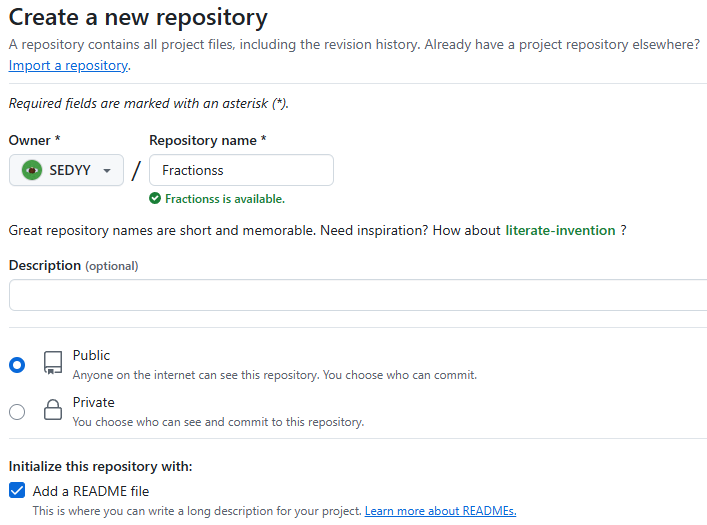


Рисунок 2 – Создание репозитория

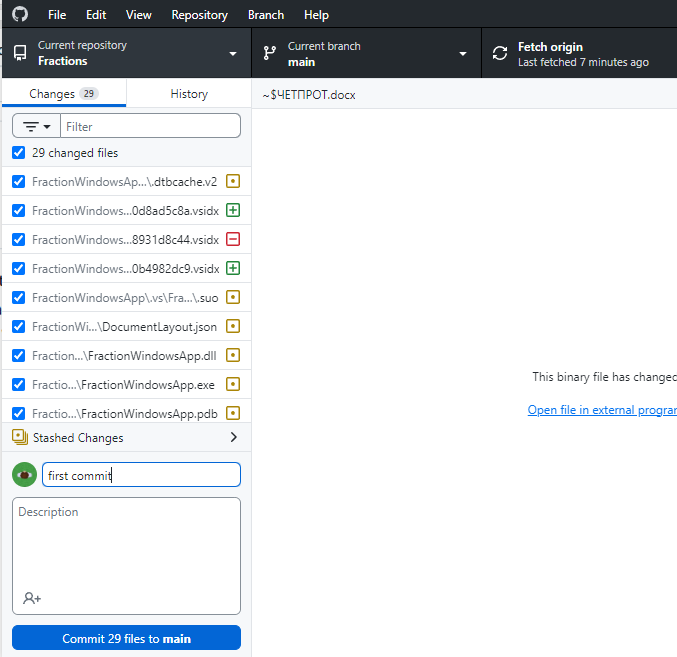


Рисунок 3 – коммит в GitHub Desktop

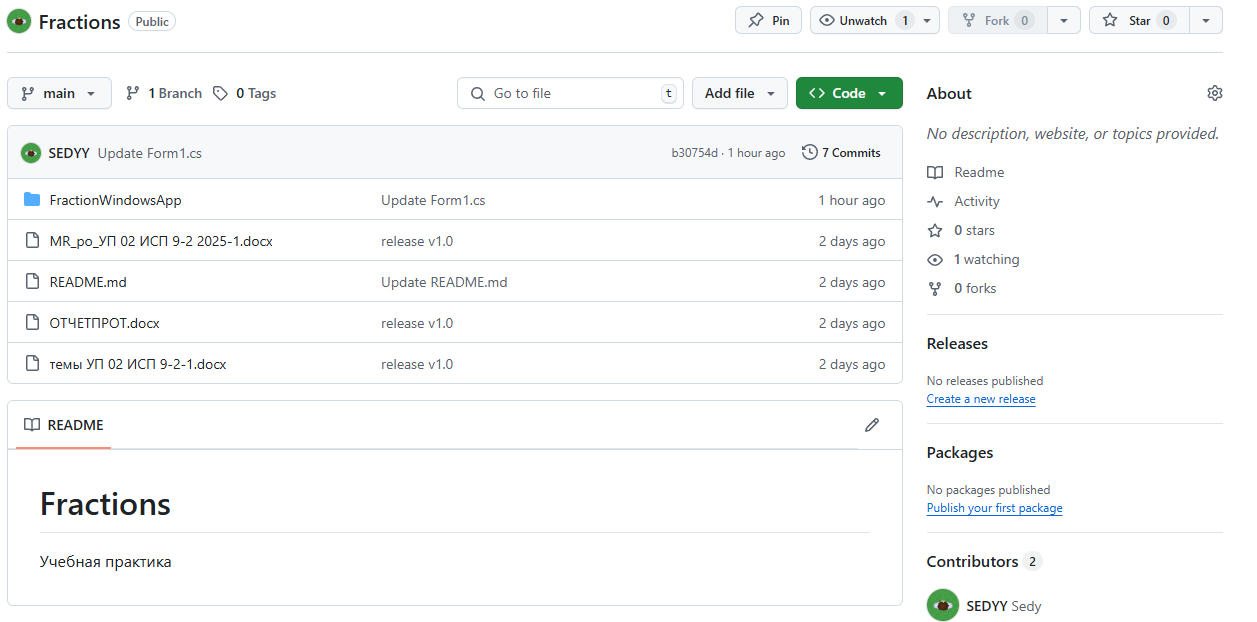


Рисунок 4 – созданный коммит на сайте

Далее был добавлен соавтор репозитория для совместной разработки (см. Рисунок 5), а точнее для ревьюирования.

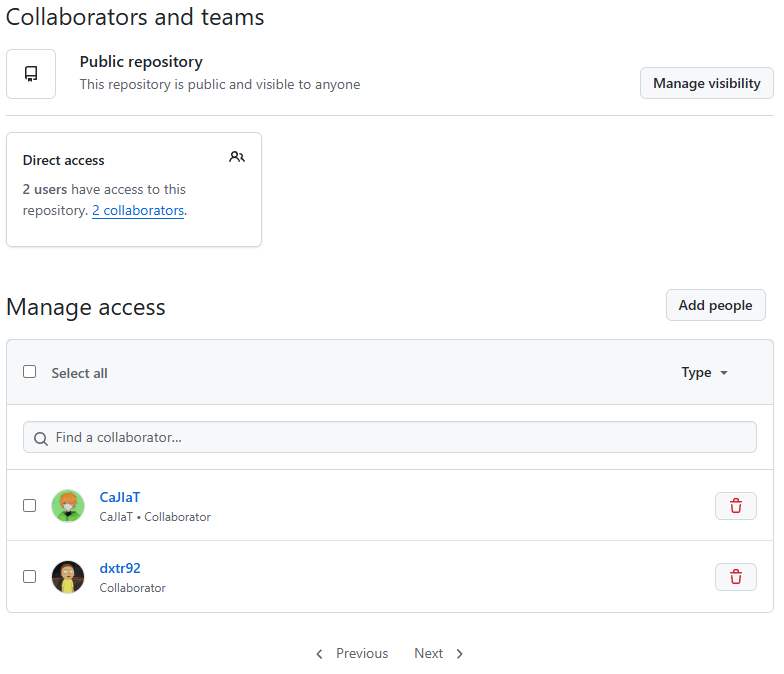


Рисунок 5 – Соавторы репозитория

2.2 Ревьюирование кода

В рамках выполнения учебной практики был получен доступ проекта другого студента. После анализа кода был проведен тест функциональности программы методом белого ящика (см. Рисунок 6).

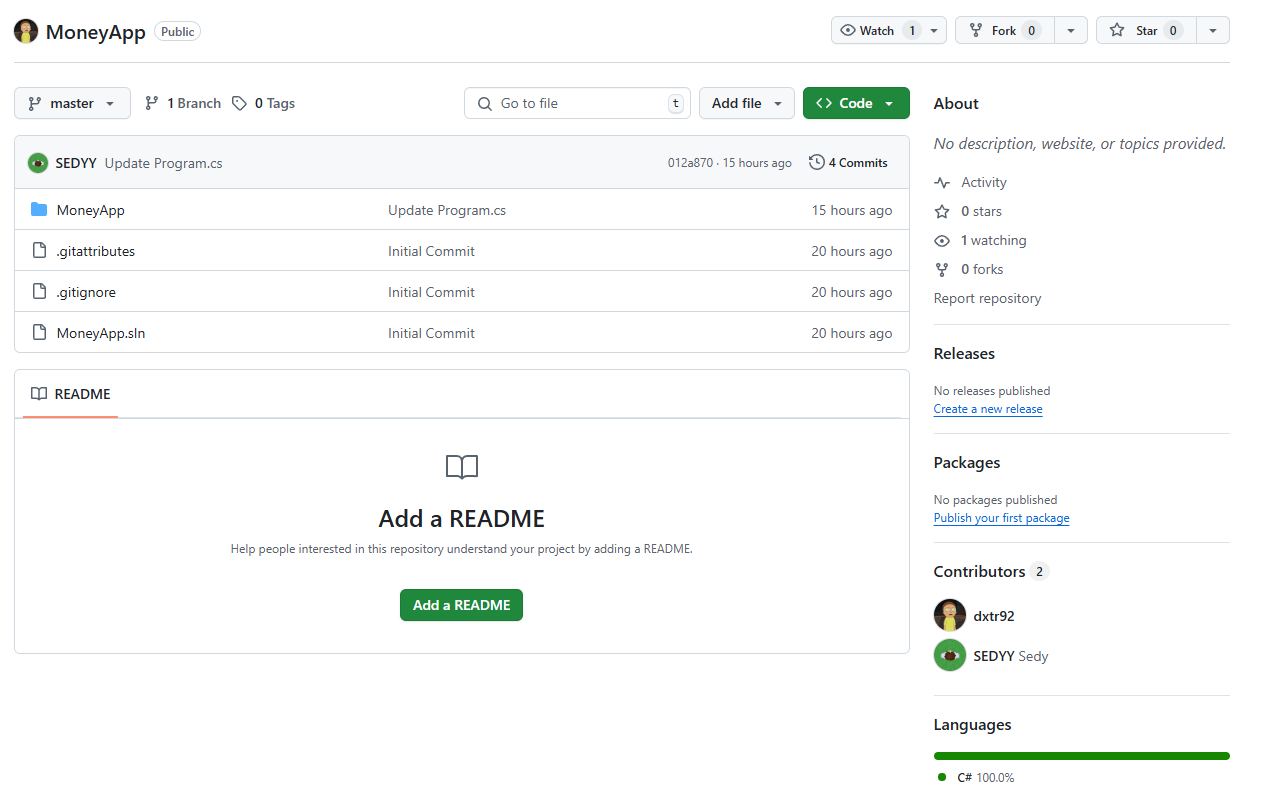


Рисунок 6 – Репозиторий другого студента

Ревьюирование кода – процесс проверки и тестирования кода с целью выявления и исправления ошибок, а также для улучшения качества программы и соответствия требованиям. Первым делом разработчик программы делает репозиторий с исходным кодом, затем другой разработчик производит ревью – анализирует код, оценивает его логику, читаемость, эффективность, а также корректность решений. В процессе ревью был открыт issue, (специальная функция для описания ошибок в коде проекта) в котором указаны замечания либо предложения по улучшению. После доработки автором код может быть одобрен и интегрирован в основную ветку проекта.

На основании ревью были выявлены ошибки, которые указаны в issue (см. Рисунок 7,8).

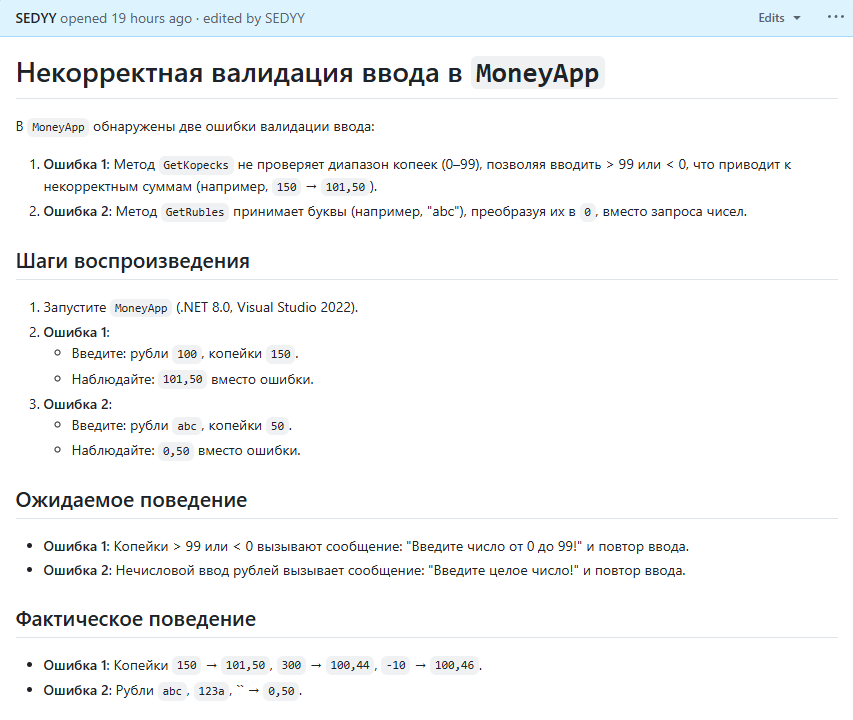


Рисунок 7 – Issue

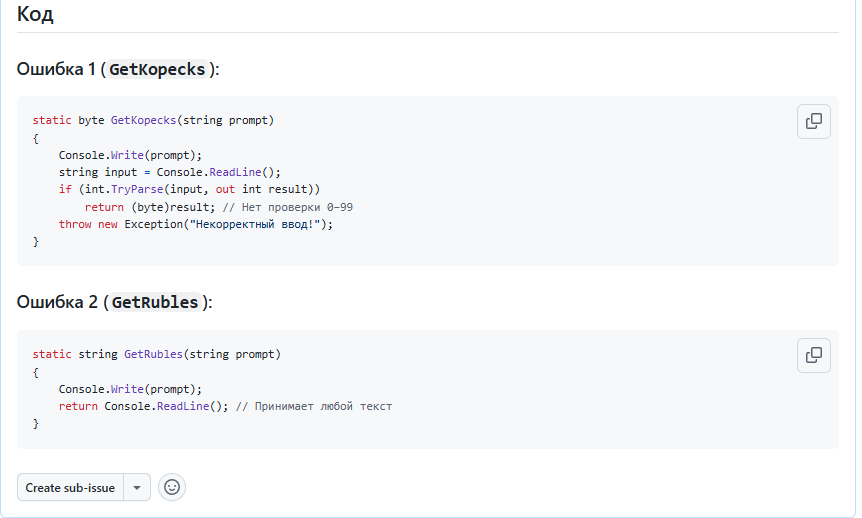


Рисунок 8 – Issue

2.3 Тестирование программы

Тестирование программы является важным этапом разработки. Чаще всего оно проводится с целью выявления ошибок, а также для проверки работы программы и соответствие требованиям. В результате проведенного тестирования были обнаружены ошибки: Метод GetKopecks не проверяет диапазон копеек (0–99), позволяя вводить > 99 или < 0, что приводит к некорректным суммам. Также Метод GetRubles принимает буквы (например, "abc"), преобразуя их в 0, вместо запроса чисел. (см. Рисунок 9).

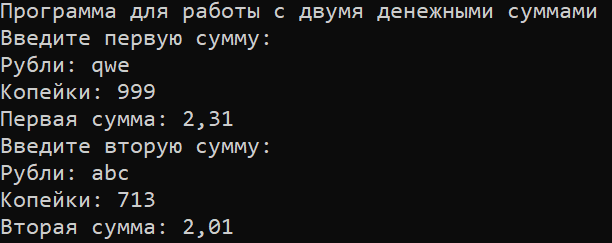


Рисунок 9 – Тестирование программы

Обнаруженные ошибки были занесены в таблице «Баг-репорт» (см. Таблицу 2), который также был направлен автору проекта для устранения заечаний. В результате была успешно завершена значимая часть практики — участие в ревьюировании и тестировании программного обеспечения, что способствовало развитию навыков анализа кода других разработчиков и правильного составления технической документации.

Таблица 2 – Баг-репорт

ТАБЛИЦА

2.4 Нотация языка

Процесс нотации языка представляет собой структурированное описание языковых элементов с использованием строгой системы правил, часто опирающейся на формальные метаязыки, такие как BNF (Backus-Naur Form) или EBNF. В контексте программирования нотация позволяет раскрыть организацию компонентов языка — от лексических символов до управляющих конструкций и типов данных. Код (Рисунок 10) демонстрирует типичные элементы C#: подключение пространств имён через директивы using, определение класса Money, конструкторы, методы, свойства, переопределение метода ToString(), а также реализацию перегрузки операторов для арифметических операций и сравнений. Кроме того, включены вспомогательные функции для обработки ввода и вывода, а также обработка исключений.

В методе Main создаются экземпляры класса Money, осуществляется ввод данных (рубли и копейки), выполняются операции сложения, вычитания, умножения, деления и сравнения, а результаты отображаются пользователю. Эта структура отражает чёткую и логичную систему нотации C#, где каждая конструкция имеет определённое назначение и следуют строгим правилам синтаксиса.

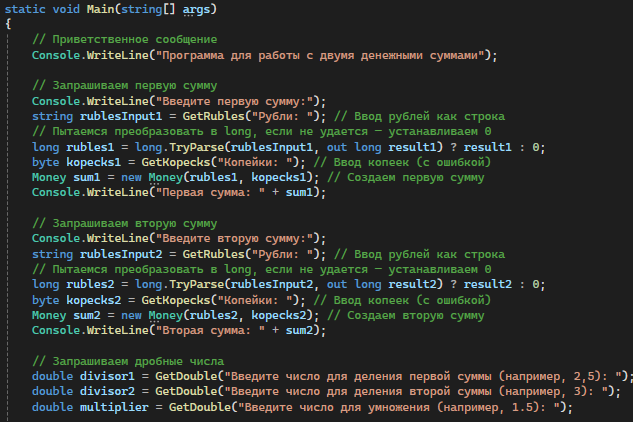


Рисунок 10 – Нотация языка

2.5 Ревьюирование алгоритма на сложность

Цель ревьюирования — проанализировать вычислительную сложность алгоритмов, применённых в приложении для работы с денежными суммами, и определить потенциальные пути оптимизации. Разработанное консольное приложение на языке C# позволяет пользователю управлять денежными суммами, представленными в виде рублей и копеек. Программа реализует такие функции, как создание денежных сумм, выполнение арифметических операций (сложение, вычитание, умножение, деление), сравнение сумм, а также ввод и вывод данных в удобном формате. (Рисунок 11)

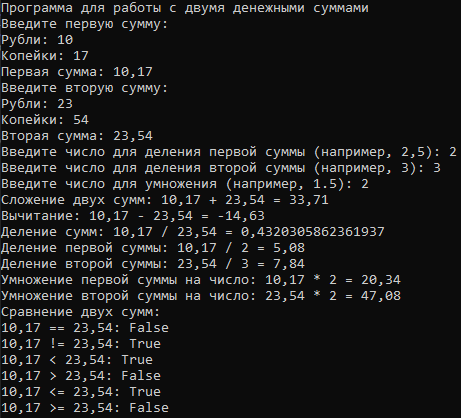


Рисунок 11 - Ревьюирование алгоритма на сложность

2.6 Написание коммитов

Оформление комментариев способствует фиксации выявленных проблем, предложений по доработке и других наблюдений, сделанных при анализе кода. Комментарии могут включать описание найденных ошибок, рекомендации по оптимизации структуры кода и другие полезные замечания. В ходе ревьюирования кода был добавлен комментарий, указывающий на некорректную обработку ввода данных при создании денежной суммы с некорректными значениями копеек (например, превышающих 99). (Рисунок 12)

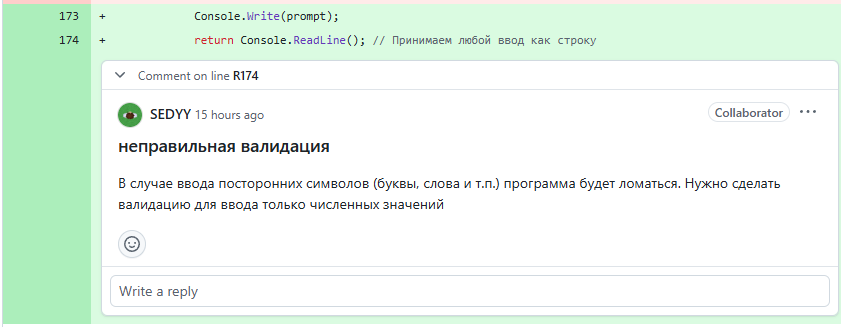


Рисунок 12 – Комментарий

2.7 Анализ решения через GitHub

После получения замечаний от другого студента (рисунок 13) и баг-репорта, были исправлены проблемы и замечания, а впоследствии написан баг-репорт на исправление ошибок (таблица 3).

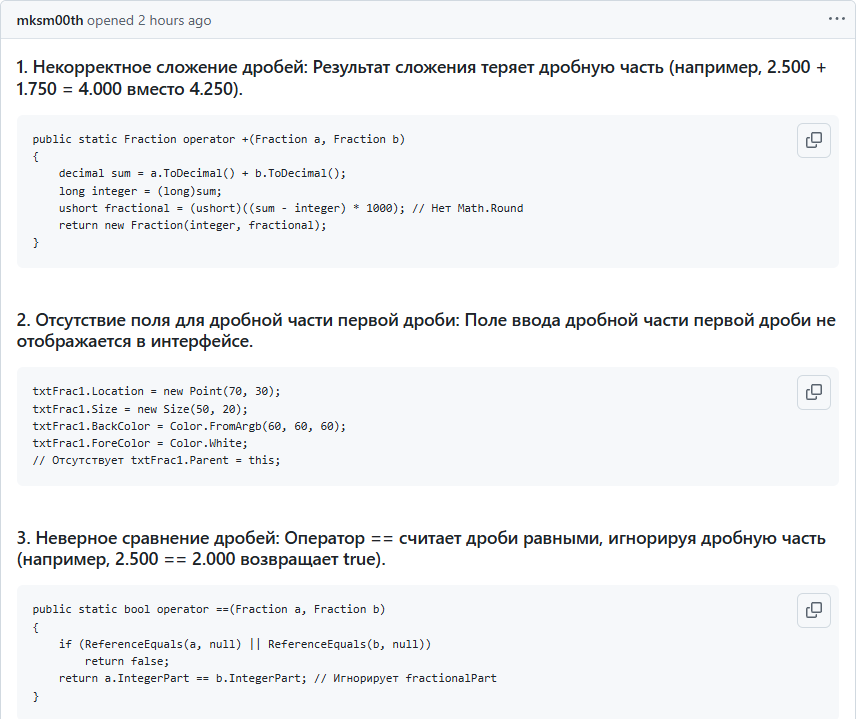
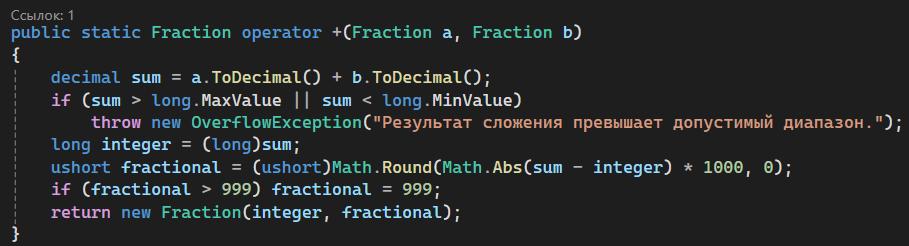
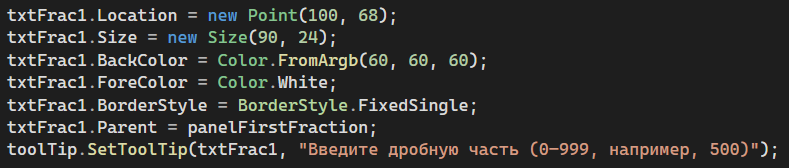
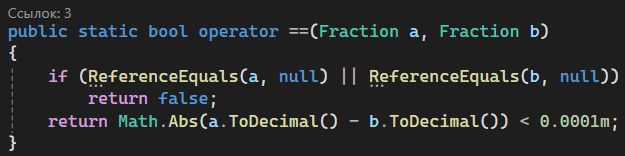


Рисунок 13 – Комментарий с ошибками

Рисунок 14 – Исправление ошибок

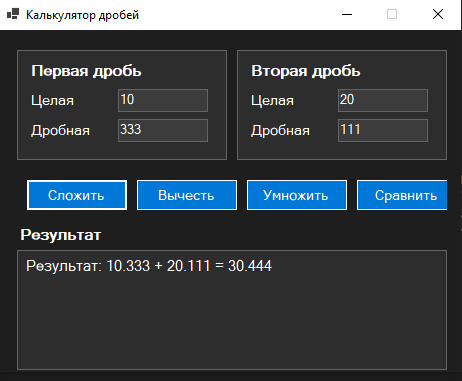


Рисунок 15 - Вывод

Таблица 3 – Баг-репорт на исправление ошибок