**Лабораторная работа №1**

**Введение в автономное тестирование**

**Цель работы:**

Приобретение практических навыков использования тестового каркаса NUnit для создания автономных тестов.

**Задание на лабораторную работу:**

Создать и настроить в Visual Studio решение, состоящие из двух проектов: проект с модульными тестами, и проект с тестируемыми классами.

Изучить тестовый каркас NUnit, его атрибуты методы, и применить для создания: простых тестов проверки возвращаемых значений, параметризованных тестов, тестов для проверки исключений, тестов для проверки состояний.

Реализовать в созданных проектах эти виды тестов и тестируемый код для них.

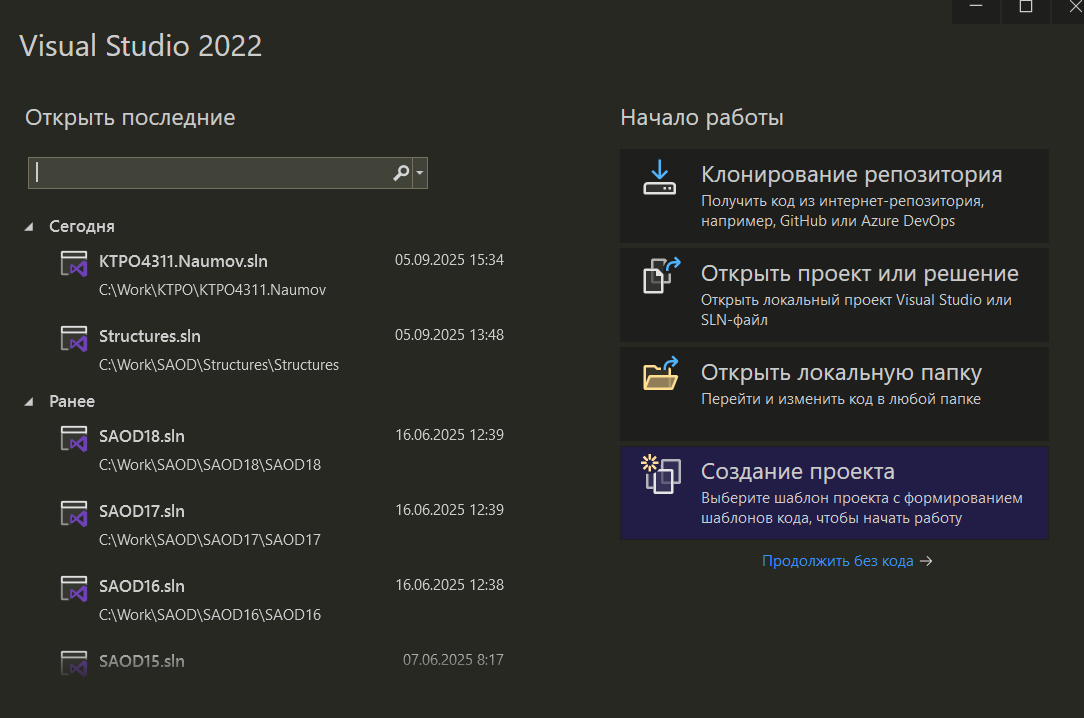
Изучить инструменты Visual Studio для запуска тестов и анализа покрытия кода тестами.

В процессе выполнения работы уделить внимание соблюдению соглашения именования тестовых классов и тестовым методов, также организации структуры папок файлов исходного кода.

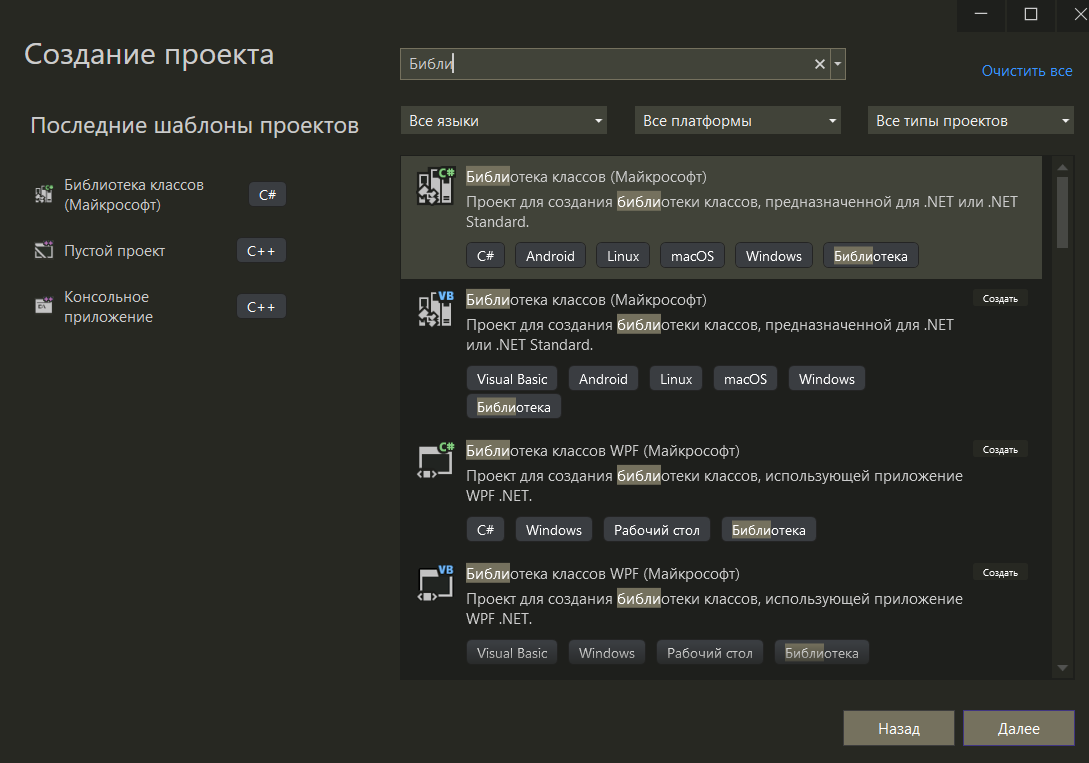
**Порядок выполнения работы**

**Создание проекта для автономных (модульных тестов)**

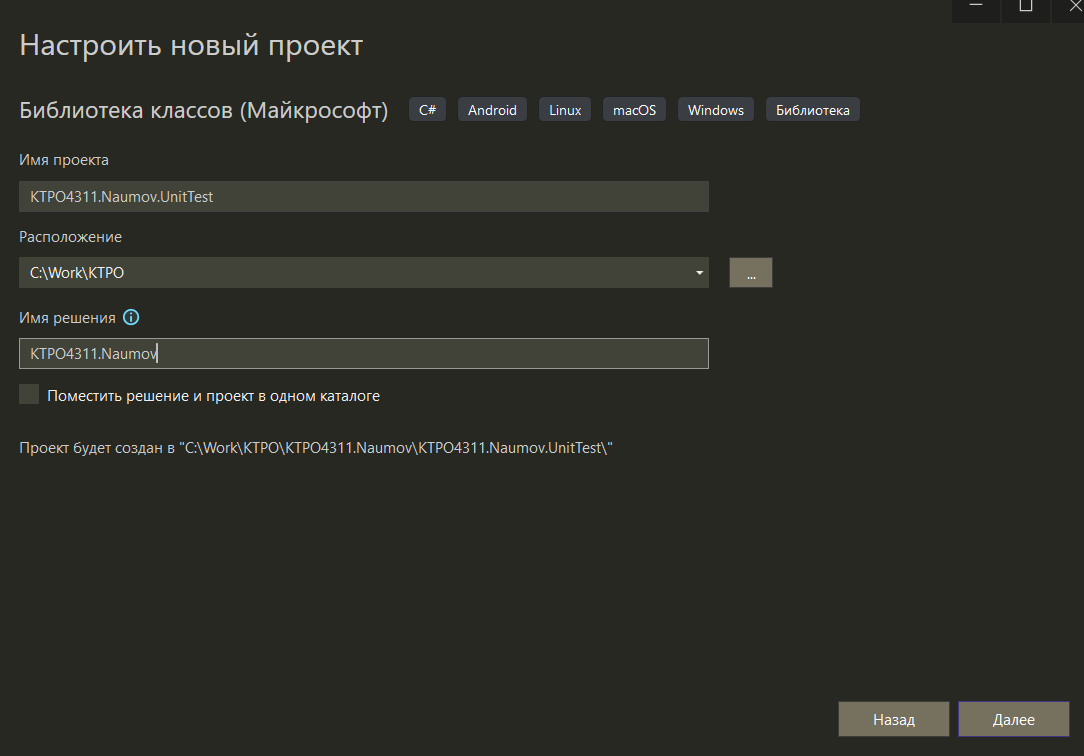
1. Открываем Visual Studio
2. Создаем новый проект



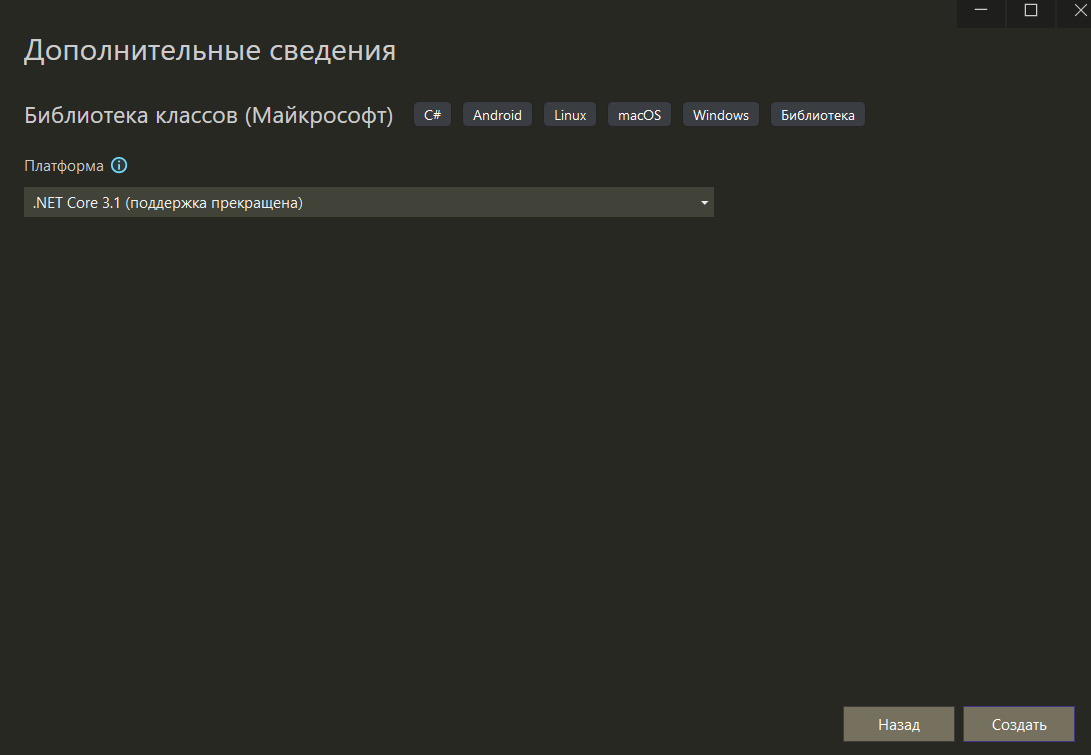
1. Выбираем шаблон



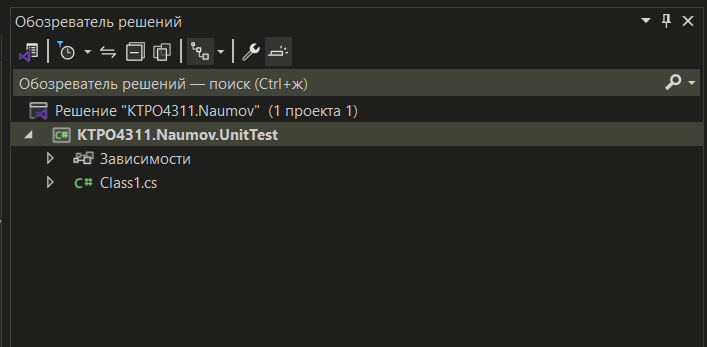
1. Настраиваем новый проект



1. Выбираем платформу и нажимаем “Создать”

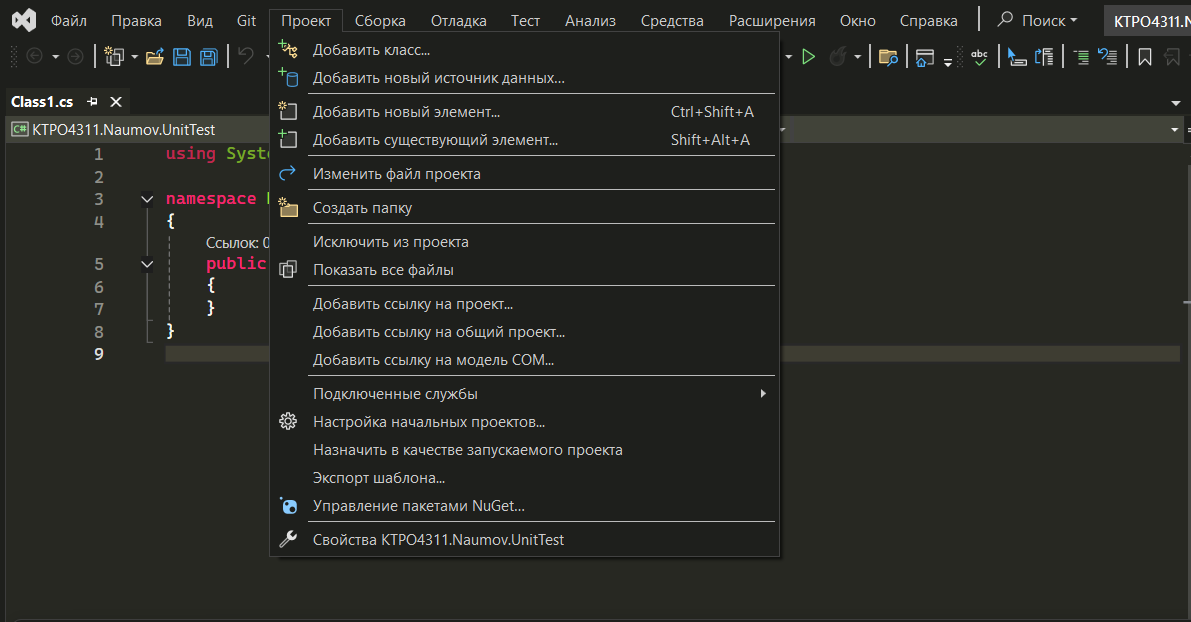


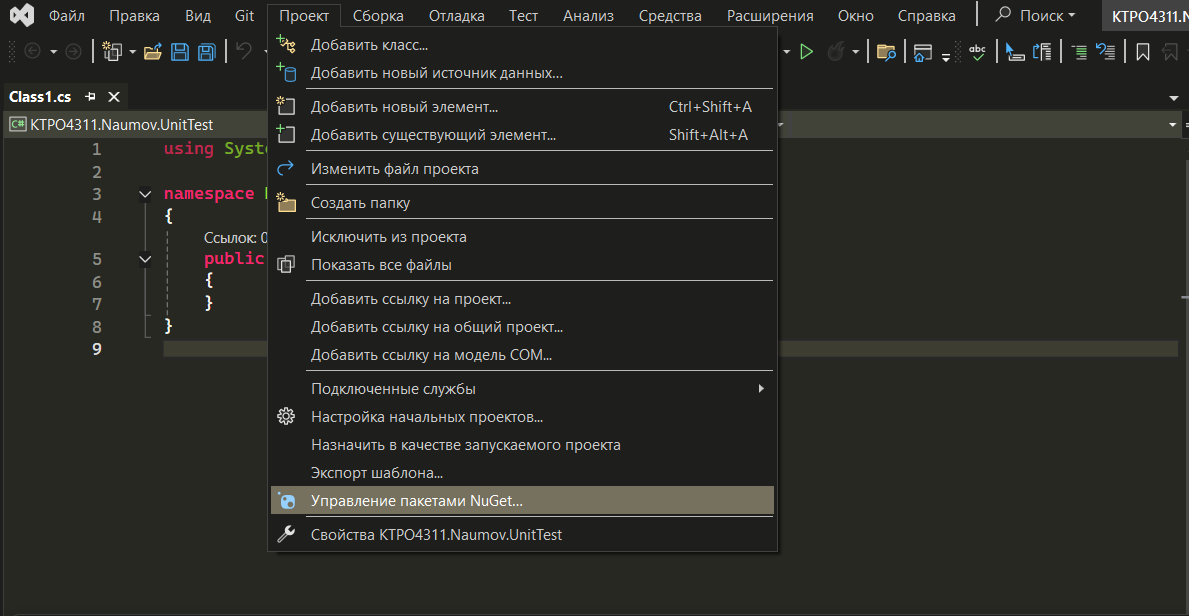
1. В обозревателе решений получаем следующее



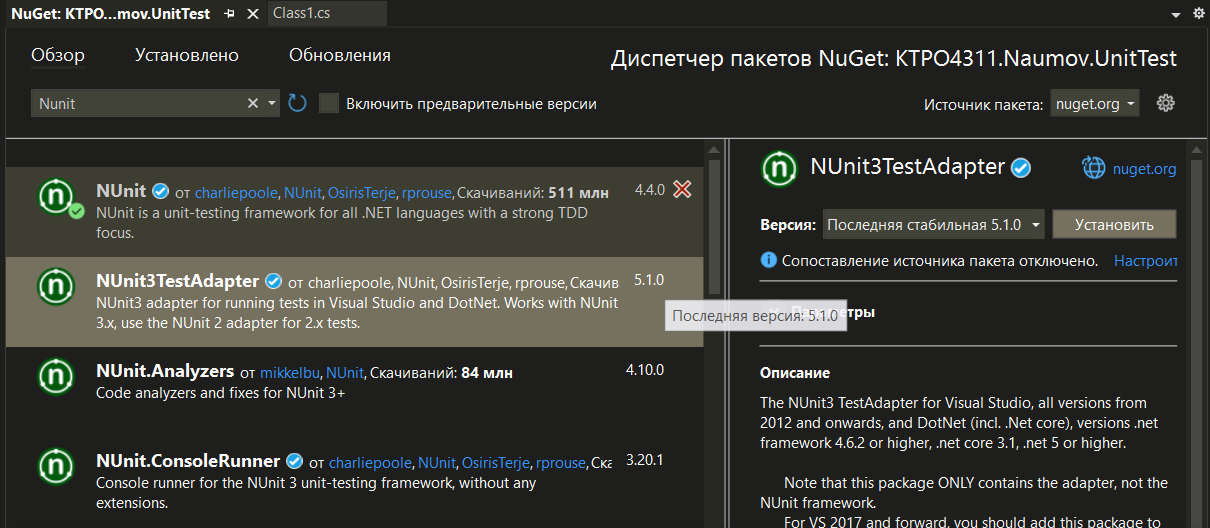
**Подключение в проект “.UnitTest” тестовый каркас NUnit**

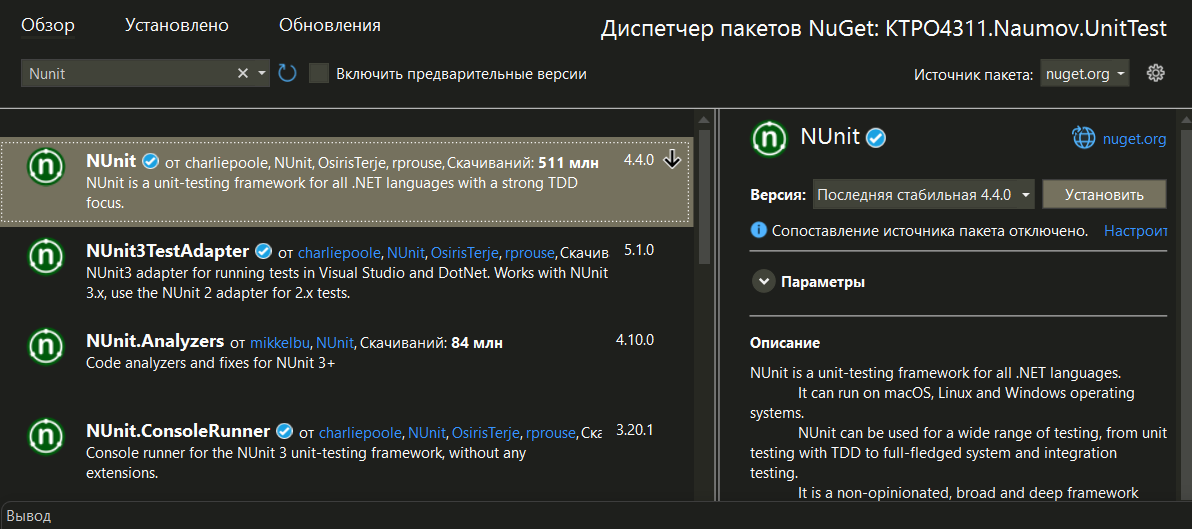
1. В панели сверху

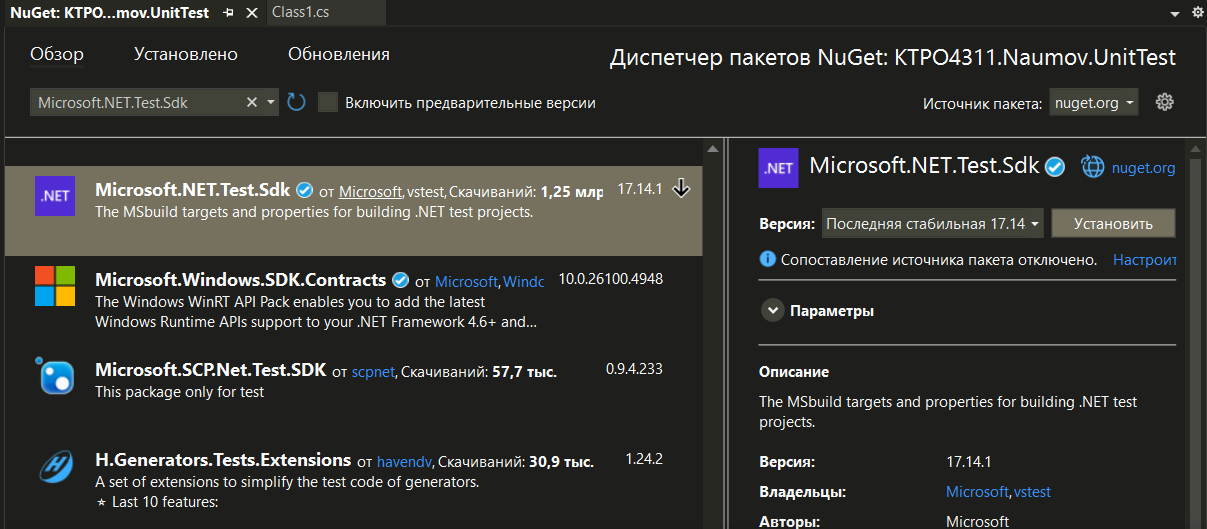
****

****

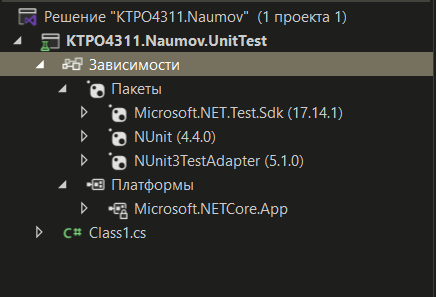
1. Устанавливаем необходимые пакеты

****

****

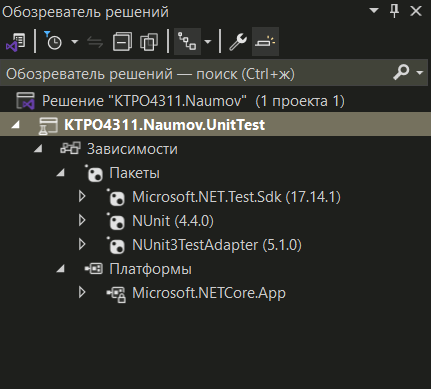
****

1. Получаем следующее

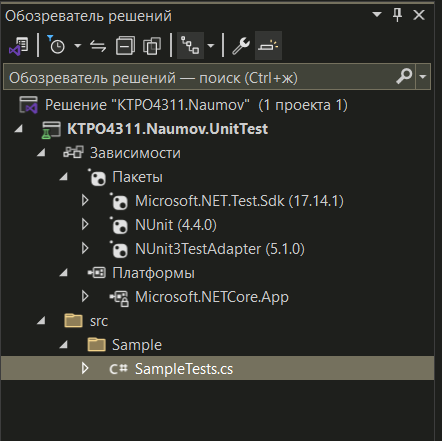


**Добавление первого модульного теста**

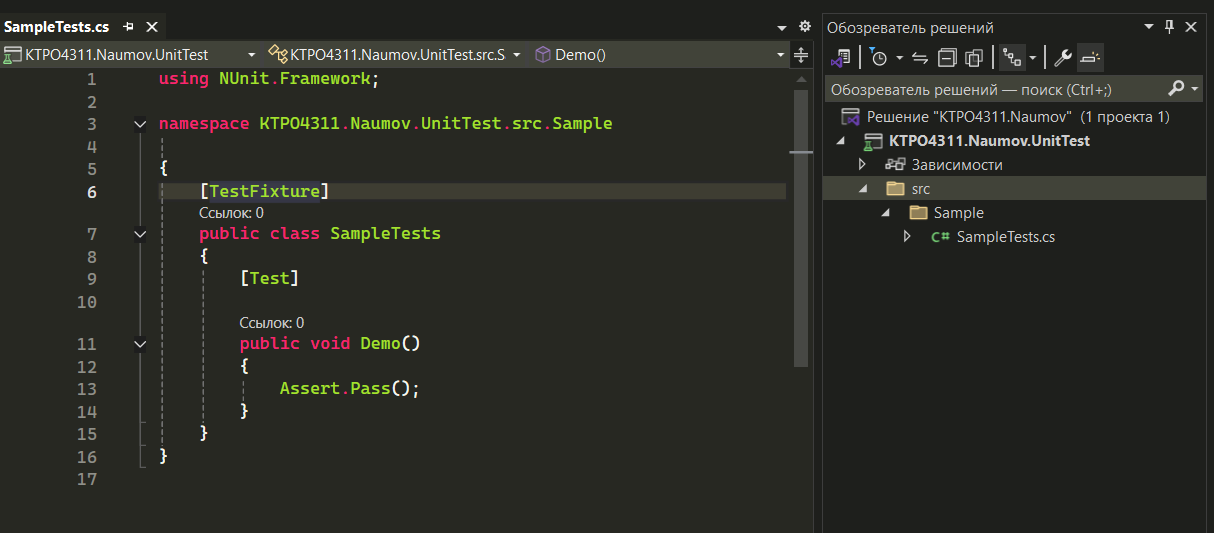
1. Удаляем файл Class.1cs



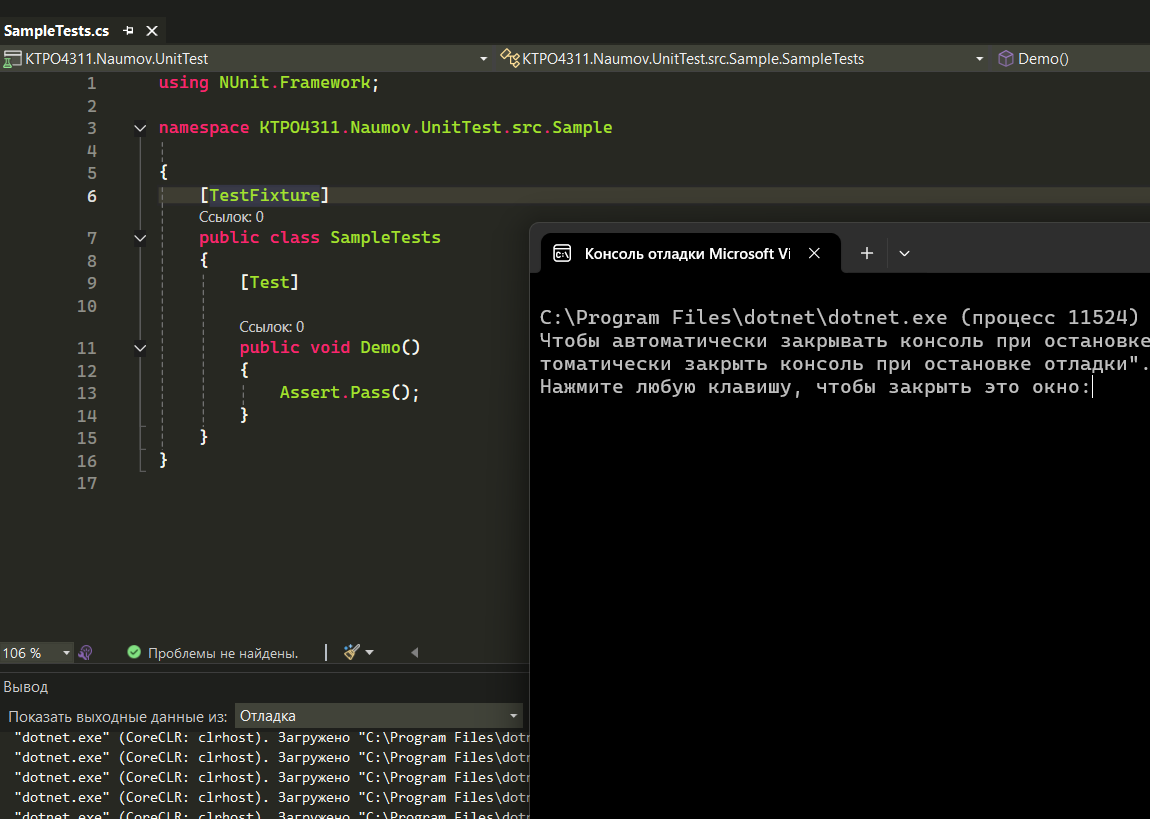
1. Создаем следующую структуру



1. Пишем следующий код

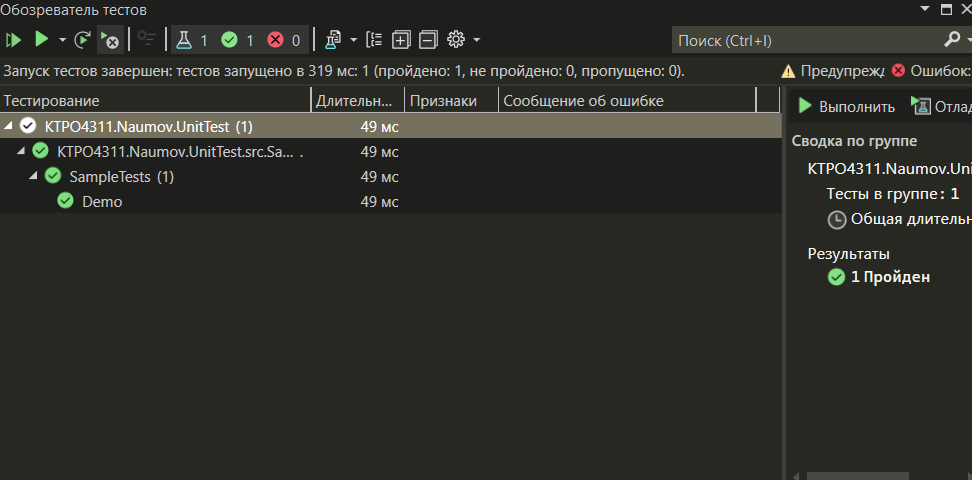


1. Убедимся, что проект компилируется без ошибок



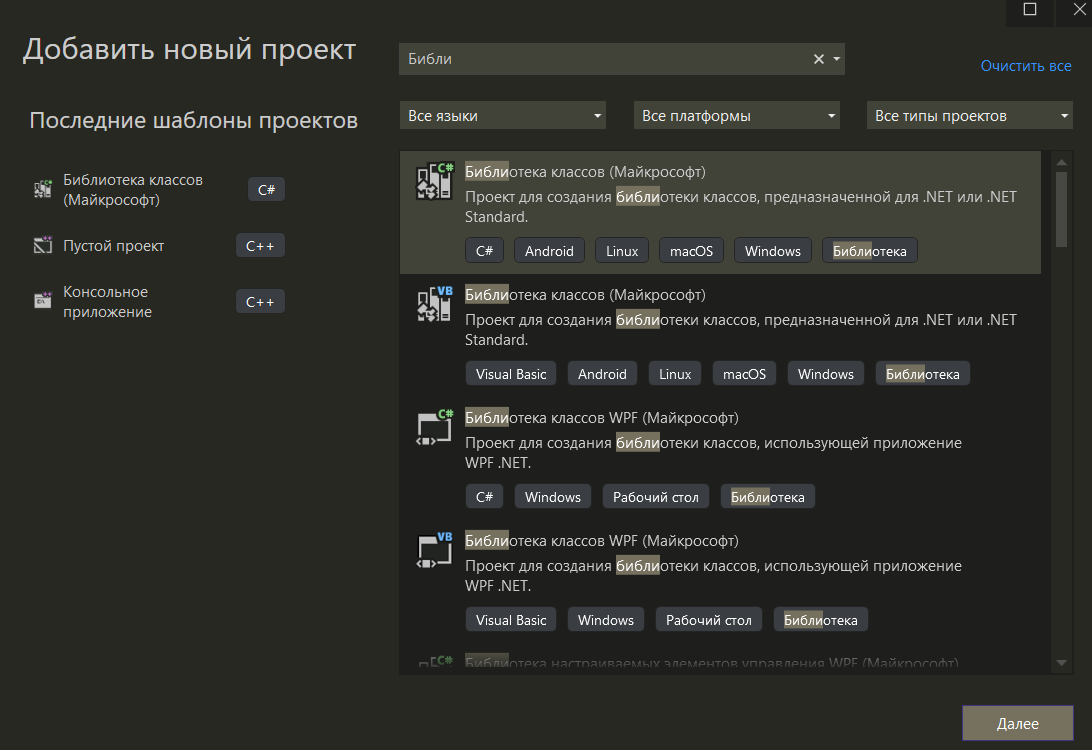
**Запуск модульного теста**

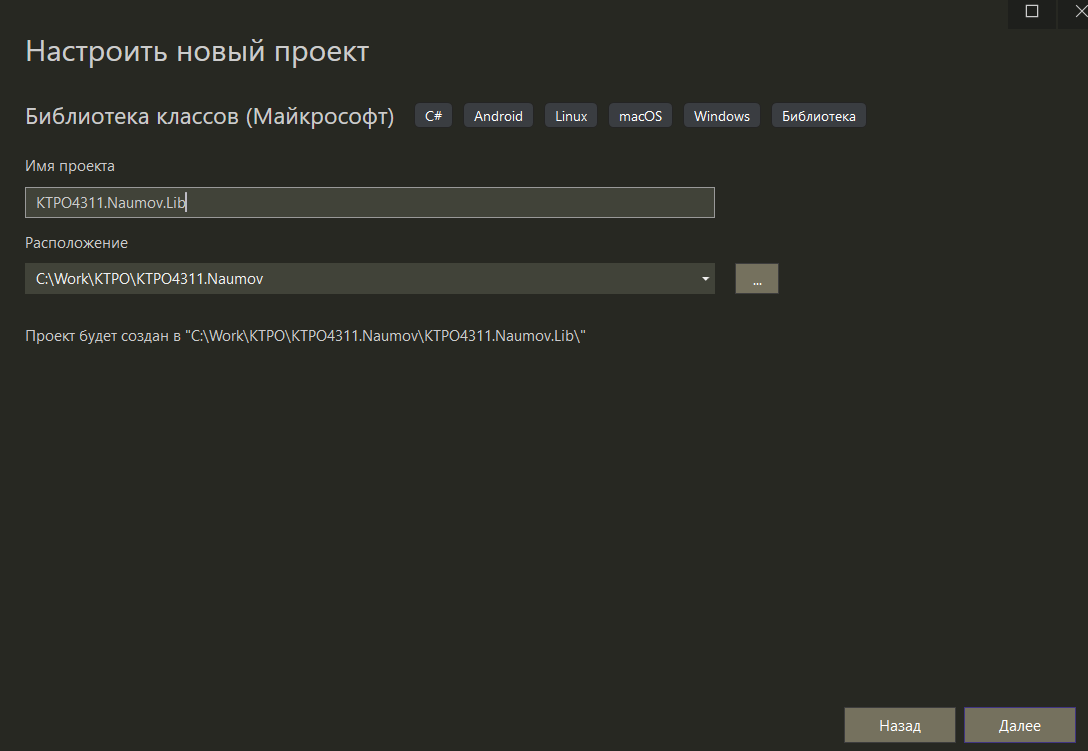
1. Переходим в обозреватель тестов и запускаем тесты



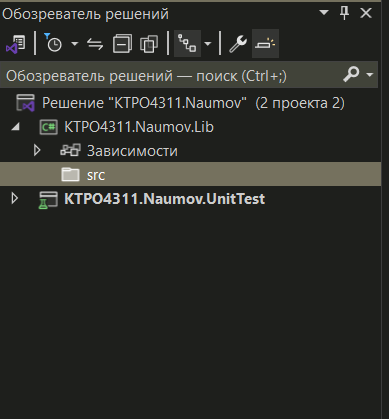
**Добавление в решение проекта для тестируемого кода**

1. Добавляем в решение новый проект



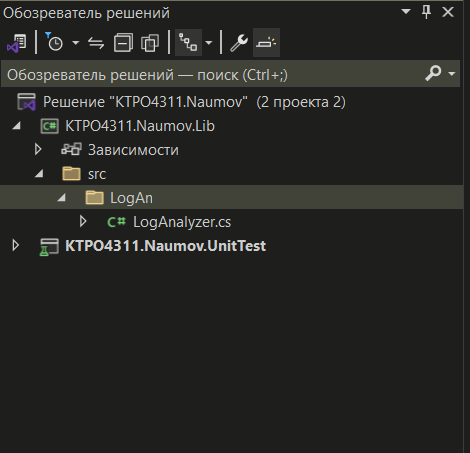


1. Удаляем из проекта дефолтный файл и создаём папку src

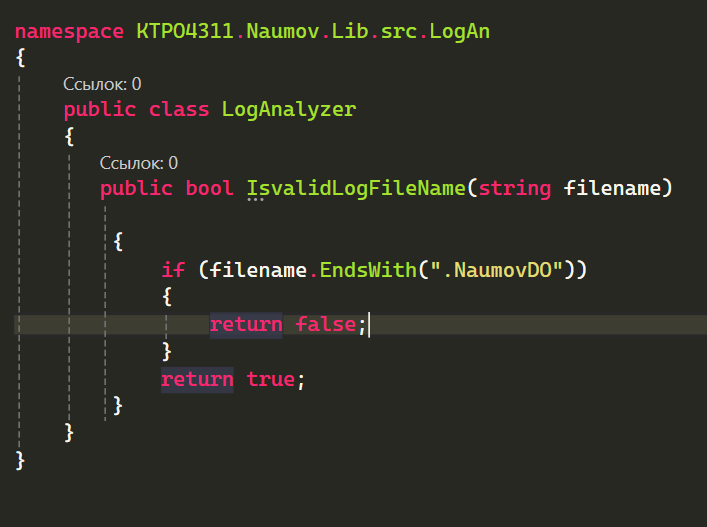


**Добавление тестируемого кода и тестов для него**

1. Создаём следующую структуру папок и класс

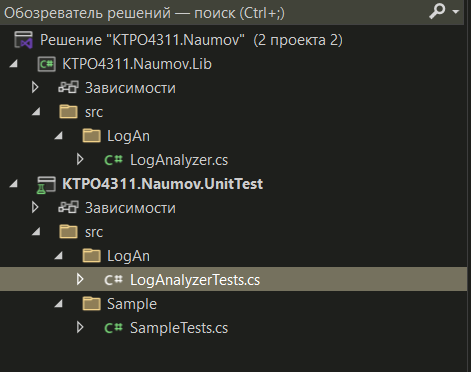


1. Добавляем в класс метод для проверки расширения файла и пересобираем решение

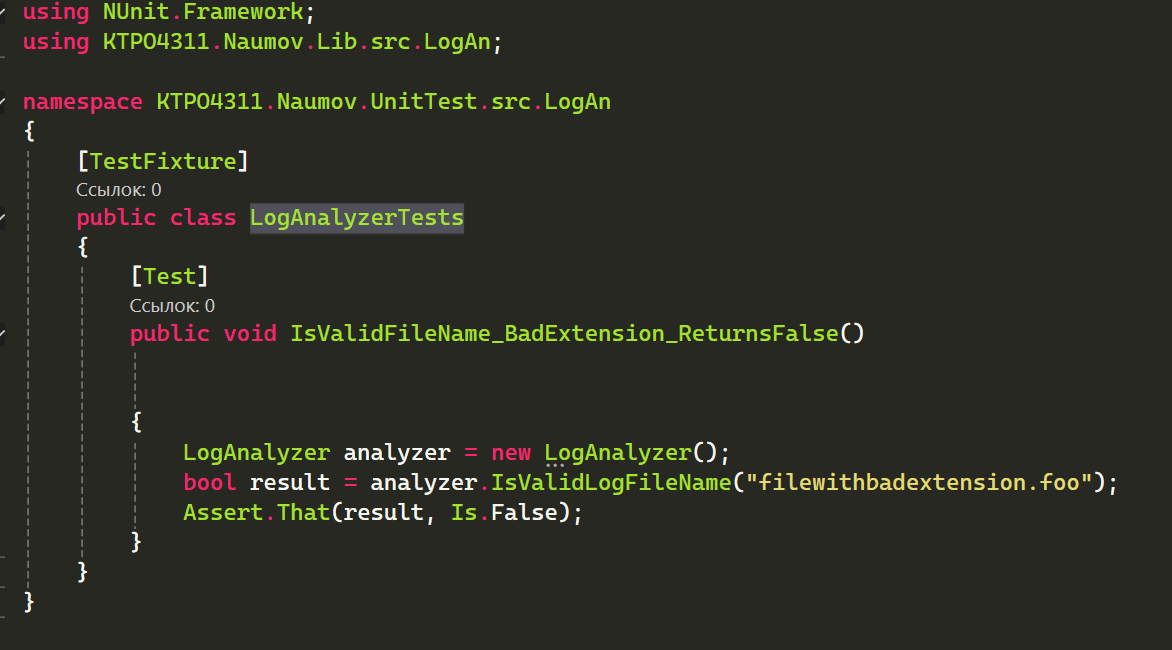


**Создание теста для метода LogAnalyzer.IsValidLogFileName**

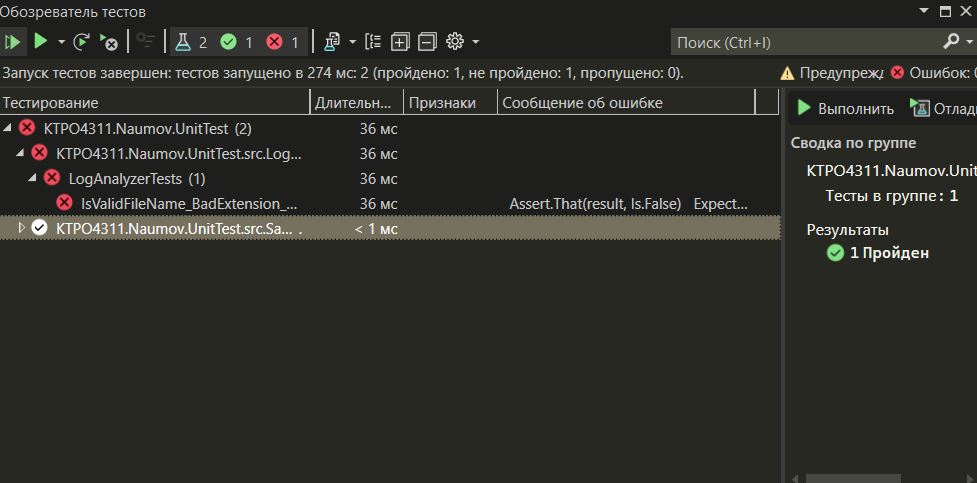
1. Создаем в проекте. UnitTest файл для теста



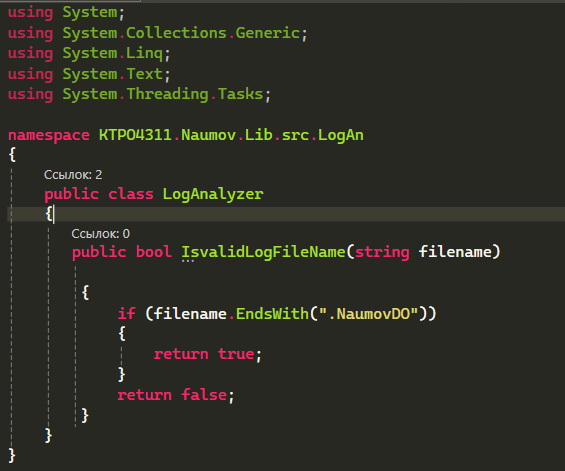
1. Пишем код

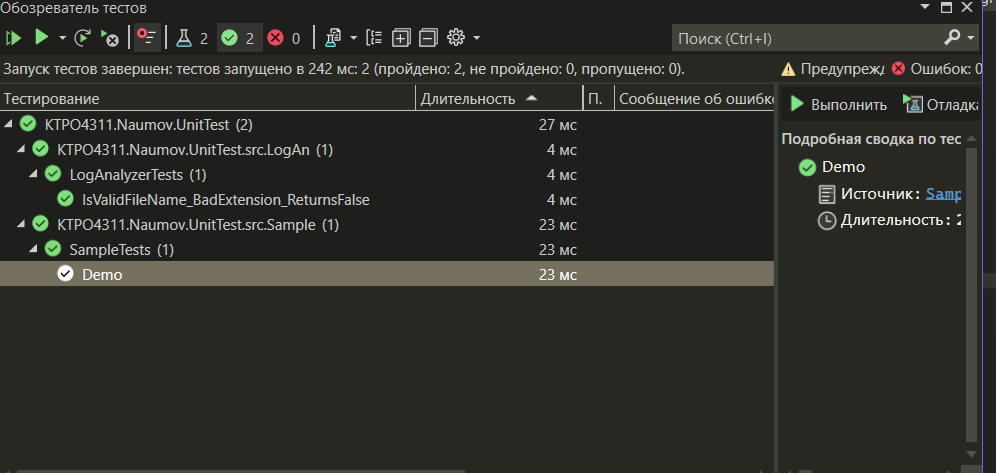


1. Запускаем тесты

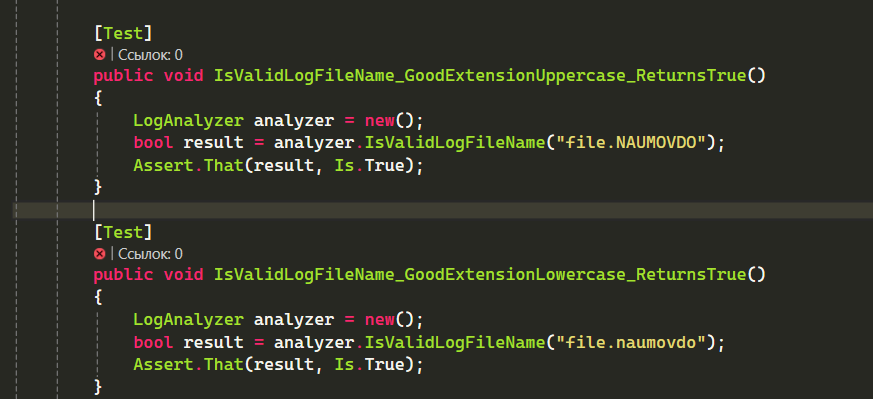


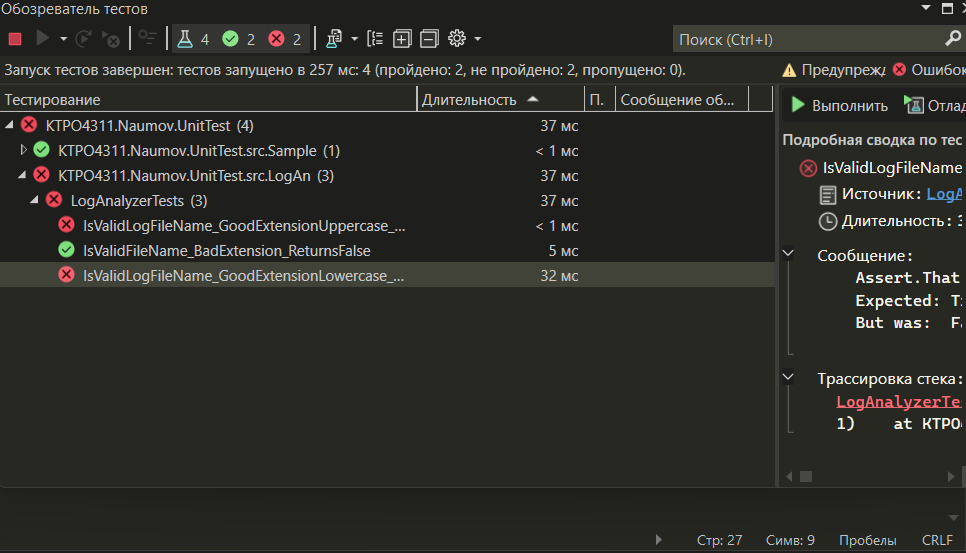
1. Исправляем метод и запускаем тесты



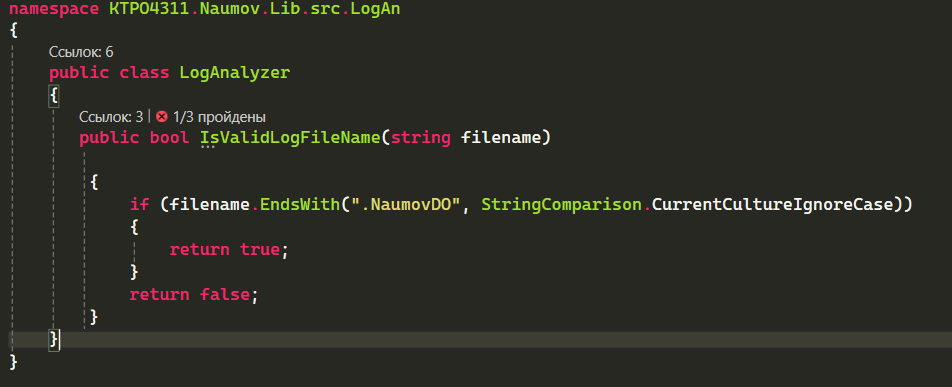


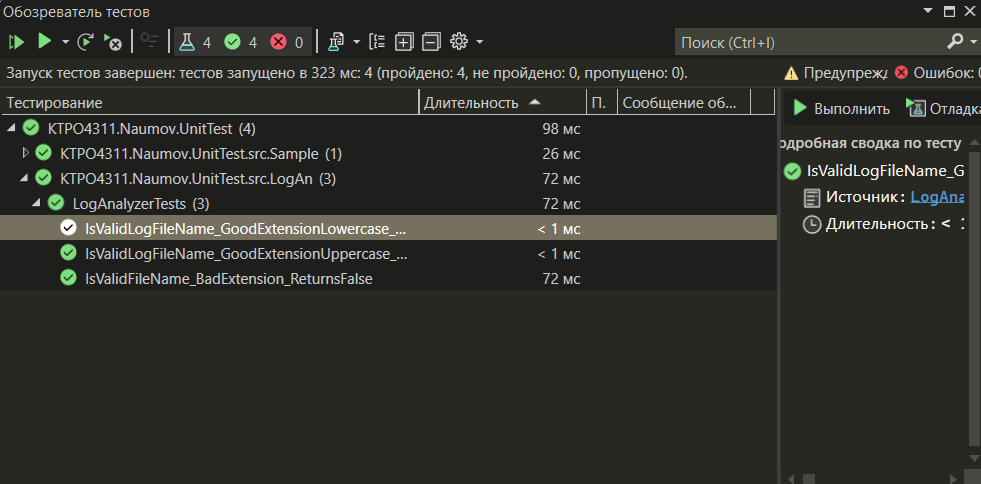
1. Добавляем тестирующие методы





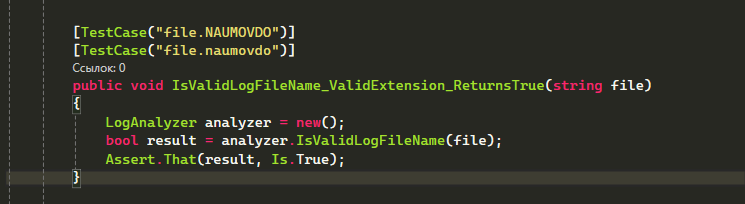
1. Исправляем тестируемый метод



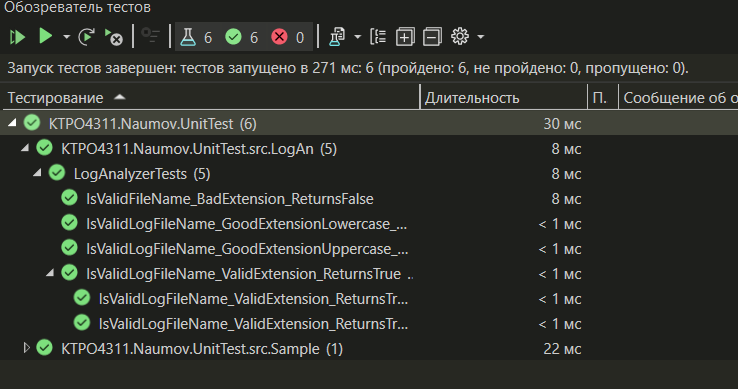


**Использование параметризованных тестов**

1. Создаём метод, добавления атрибут [TestCase]

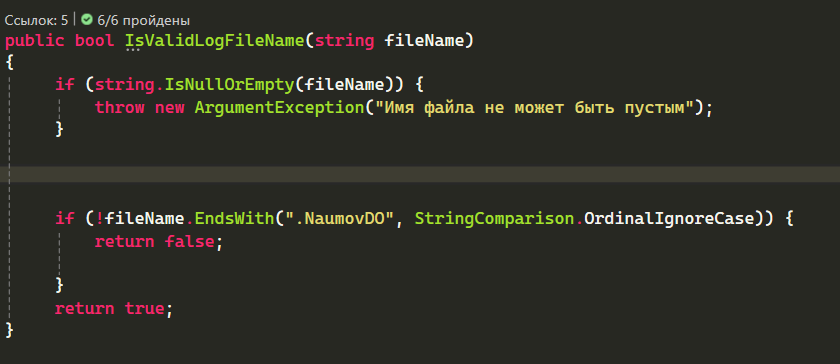


1. Выполним тесты

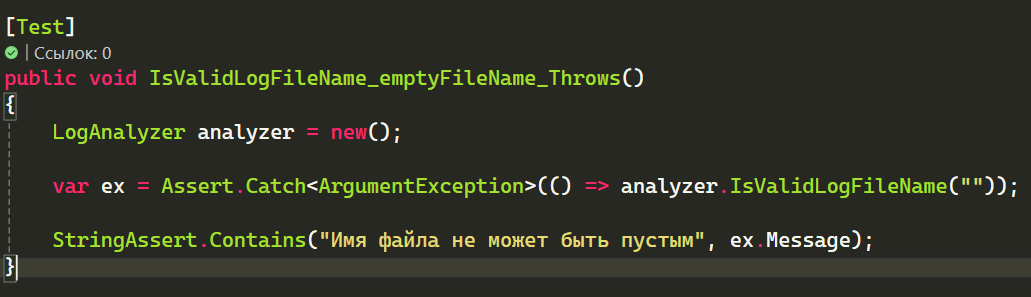


**Проверка ожидаемых исключений. Анализ покрытия кода тестами**

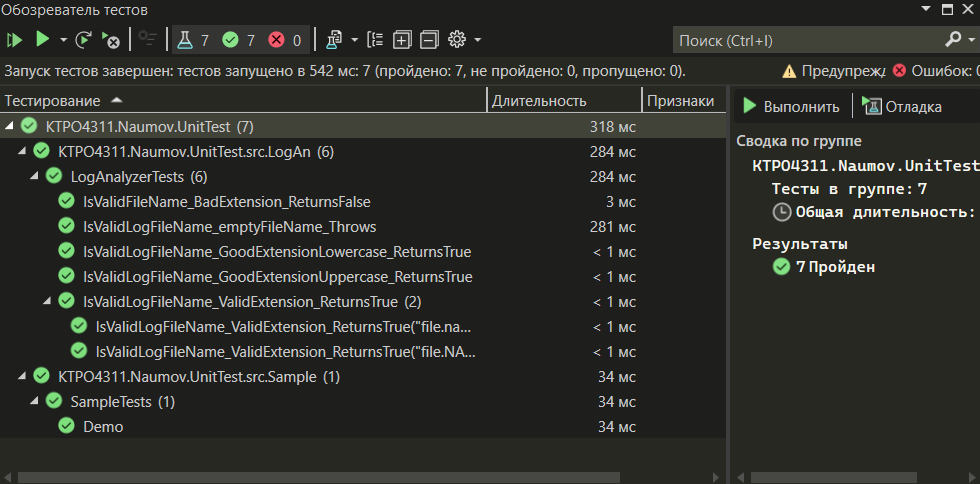
1. Обновляем тестируемый метод



1. Добавляем тест для сценария с пустым именем файла



3. Запускаем тесты

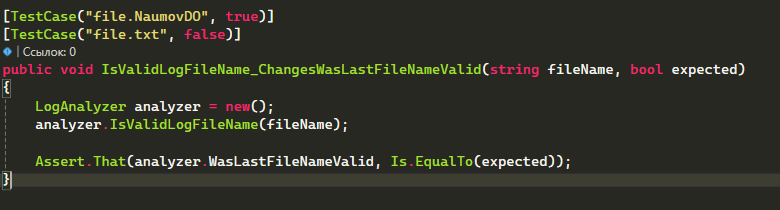


**Проверка изменения состояния**

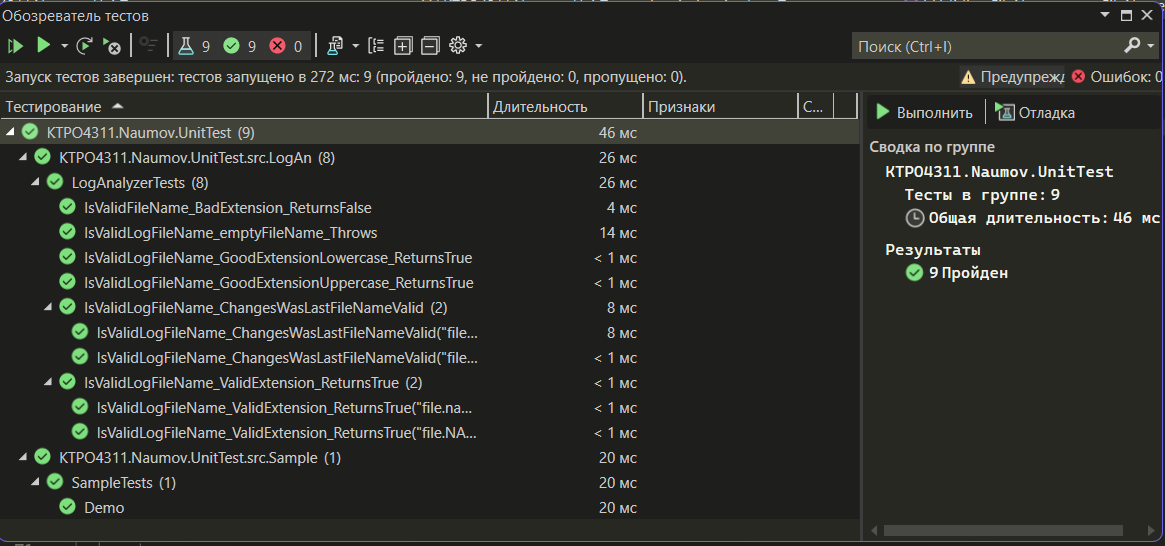
1. Обновляем LogAnalyzer и тестируемые метод



1. Создаём новый тестовый метод



1. Запускаем тесты



**Выводы**

В ходе лабораторной работы были приобретены практические навыки разработки автономных модульных тестов с использованием фреймворка NUnit в среде .NET.

Были реализованы и проверены различные виды тестов:

* простые тесты, возвращающие значения;
* параметризованные тесты для проверки нескольких наборов входных данных;
* тесты на генерацию исключений;
* тесты, контролирующие изменения состояния объекта.

Особое внимание уделялось:

* поддержанию правильной структуры проекта (src/Sample, src/LogAn);
* анализу покрытия кода, который помог выявить недостатки в тестировании и их устранить;
* исправлению ошибок в тестируемом коде (чувствительность к регистру, обработка значений null).