

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
федерального государственного бюджетного образовательного учреждения  
высшего образования  
**«РОССИЙСКИЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ  
Г.В.ПЛЕХАНОВА»**

Техникум Пермского института (филиала)

Практическая работа  
по дисциплине «Поддержка и тестирование программных модулей»

по теме: Модульное тестирование в Visual studio (MSTest)

Руководитель  
\_\_\_\_\_ /Д.Б. Берестов/  
«\_\_\_\_» \_\_\_\_\_ 2026 г.

Исполнитель  
\_\_\_\_\_ /А.Д. Тамбасова/  
«\_\_\_\_» \_\_\_\_\_ 2026 г.

Пермь, 2026 г.

## **Содержание**

1 Создание проекта для тестирования .....	3
2 Создание проекта модульного теста .....	7
3 Создание тестового класса .....	11
4 Создание метода теста .....	12
5 Сборка и запуск теста .....	13
6 Исправление кода и повторный запуск тестов.....	16
7 Использование модульных тестов для улучшения кода .....	17
7.1 Создание и запуск новых методов теста.....	17
7.2 Рефакторинг тестируемого кода.....	20
7.3 Рефакторинг тестовых методов .....	21
7.4 Повторное тестирование, переписывание и анализ .....	22

# 1 Создание проекта для тестирования

1. Для проекта выбираю шаблон Консольное приложение (Майкрософт).

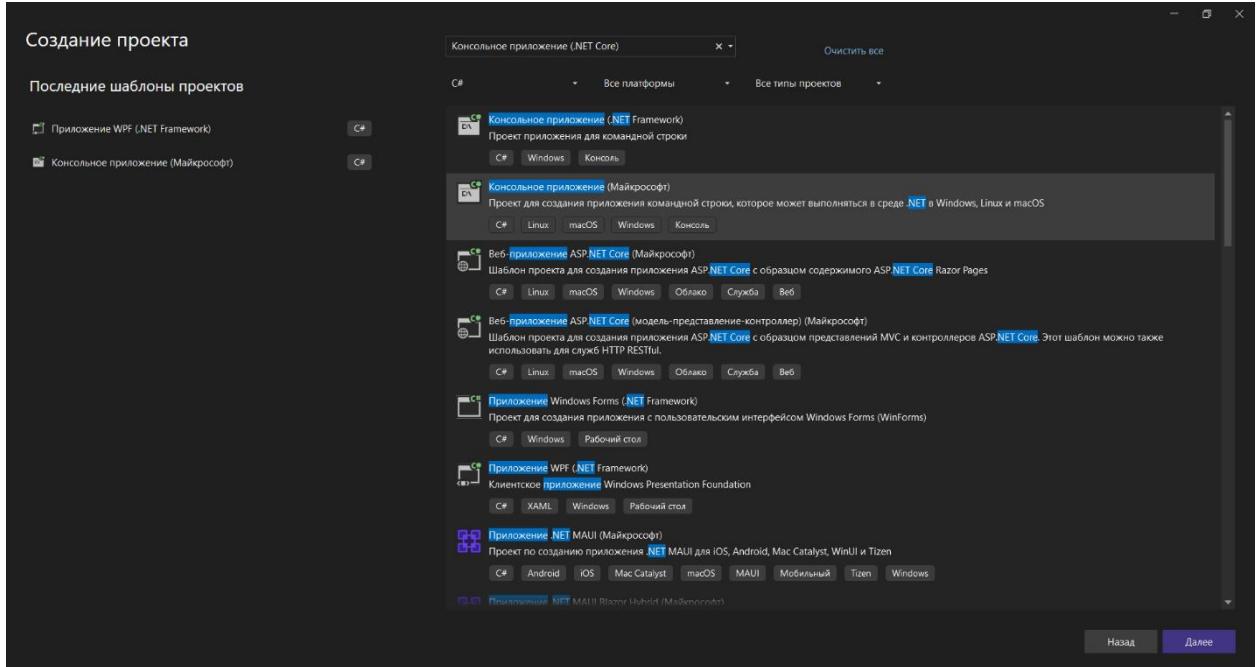


Рисунок 1 – Создание проекта Консольные приложения (Майкрософт)

2. Называю проект «tambasova\_pr1».

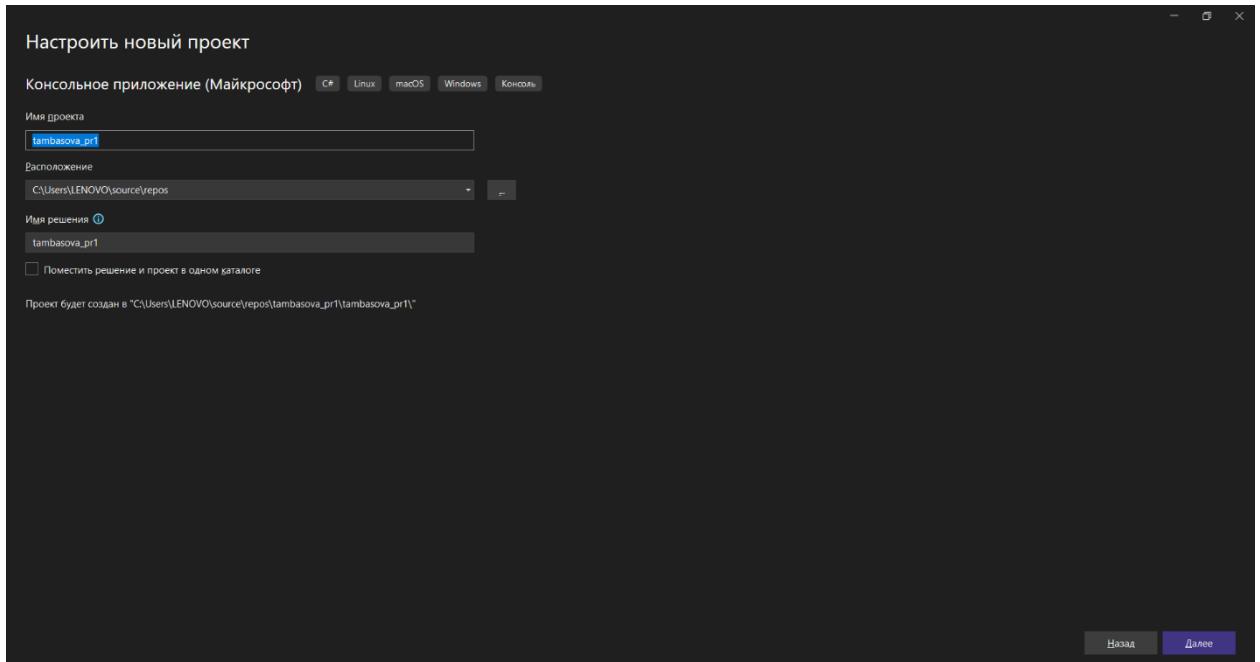


Рисунок 2 – Наименование проекта

### 3. В качестве платформы использую .NET 8.0 (долгосрочная поддержка).

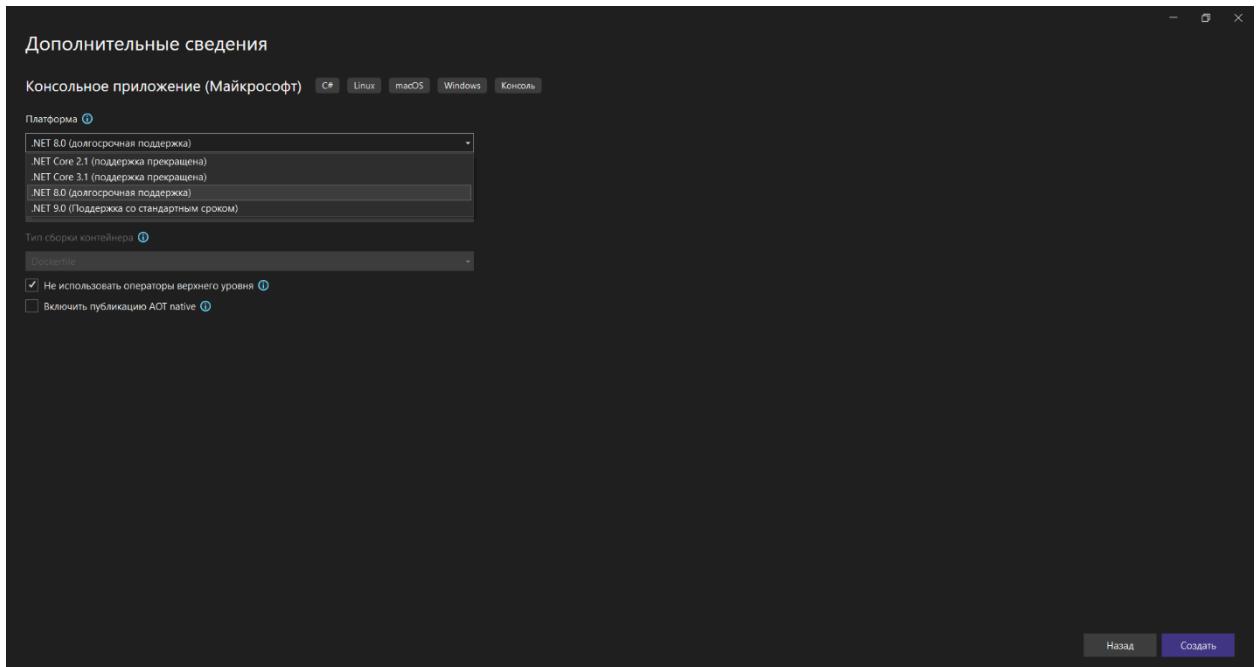


Рисунок 3 – Выбор платформы

4. Заменяю содержимое файла Program.cs следующими кодом на C#, который определяет класс BankAccount.

```
1  using System;
2  namespace BankAccountNS
3  {
4      /// <summary>
5      /// Bank account demo class.
6      /// </summary>
7      public class BankAccount
8      {
9          private readonly string m_customerName;
10         private double m_balance;
11         public BankAccount() { }
12         public BankAccount(string customerName, double balance)
13         {
14             m_customerName = customerName;
15             m_balance = balance;
16         }
17         public string CustomerName
18         {
19             get { return m_customerName; }
20         }
21         public double Balance
22         {
23             get { return m_balance; }
24         }
25         public void Debit(double amount)
26         {
27             if (amount > m_balance)
28             {
29                 throw new ArgumentOutOfRangeException("amount");
30             }
31             if (amount < 0)
32             {
33             }
34         }
35     }
36 }
```

Рисунок 4 – Код для Program.cs (1)

The screenshot shows the Microsoft Visual Studio interface with the 'Program.cs' file open in the editor. The code defines a 'BankAccount' class with methods for depositing and withdrawing money, and a static 'Main' method that creates an account, deposits \$5.77, withdraws \$11.22, and prints the current balance. The code includes several intentional errors, such as comparing amounts to -1.

```
29     if (amount < 0)
30     {
31         throw new ArgumentOutOfRangeException("amount");
32     }
33     if (amount < 0)
34     {
35         throw new ArgumentOutOfRangeException("amount");
36     }
37     m_balance += amount; // intentionally incorrect code
38
39     public void Credit(double amount)
40     {
41         if (amount < 0)
42         {
43             throw new ArgumentOutOfRangeException("amount");
44         }
45         m_balance += amount;
46
47     public static void Main()
48     {
49         BankAccount ba = new BankAccount("Mr. Bryan Walton",
50                                         11.99);
51         ba.Credit(5.77);
52         ba.Debit(11.22);
53         Console.WriteLine("Current balance is ${0}", ba.Balance);
54     }
55 }
```

Рисунок 5 – Код для Program.cs (2)

5. Переименовываю файл в BankAccount.cs.

The screenshot shows the Microsoft Visual Studio interface with the 'BankAccount.cs' file open in the editor. The code defines a 'BankAccount' class with properties for customer name and balance, and methods for depositing and withdrawing money. The code is identical to the one in Program.cs but has been moved to a separate file.

```
1  using System;
2  namespace BankAccountNS
3  {
4      /// <summary>
5      /// Bank account demo class.
6      /// </summary>
7      public class BankAccount
8      {
9          private readonly string m_customerName;
10         private double m_balance;
11         Свойство 0
12         private BankAccount() { }
13         Свойство 1
14         public BankAccount(string customerName, double balance)
15         {
16             m_customerName = customerName;
17             m_balance = balance;
18         }
19         Свойство 0
20         public string CustomerName
21         {
22             get { return m_customerName; }
23         }
24         Свойство
25         public double Balance
26         {
27             get { return m_balance; }
28         }
29         Свойство 1
30         public void Debit(double amount)
31         {
32             if (amount > m_balance)
33             {
34                 throw new ArgumentOutOfRangeException("amount");
35             }
36             if (amount < 0)
37             {
38             }
39         }
40     }
41 }
```

Рисунок 6 – Переименовывание файла в BankAccount.cs

6. В меню Сборка нажимаю Собрать решение.

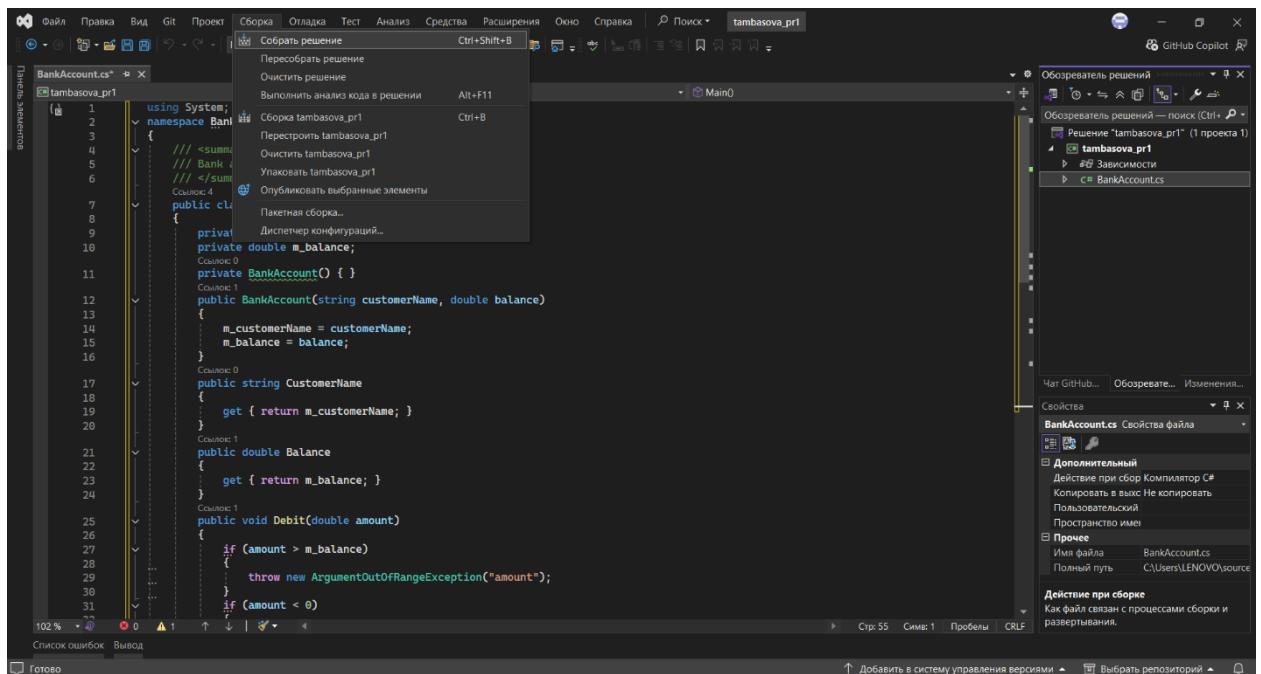


Рисунок 7 – Сборка – Собрать решение (1)

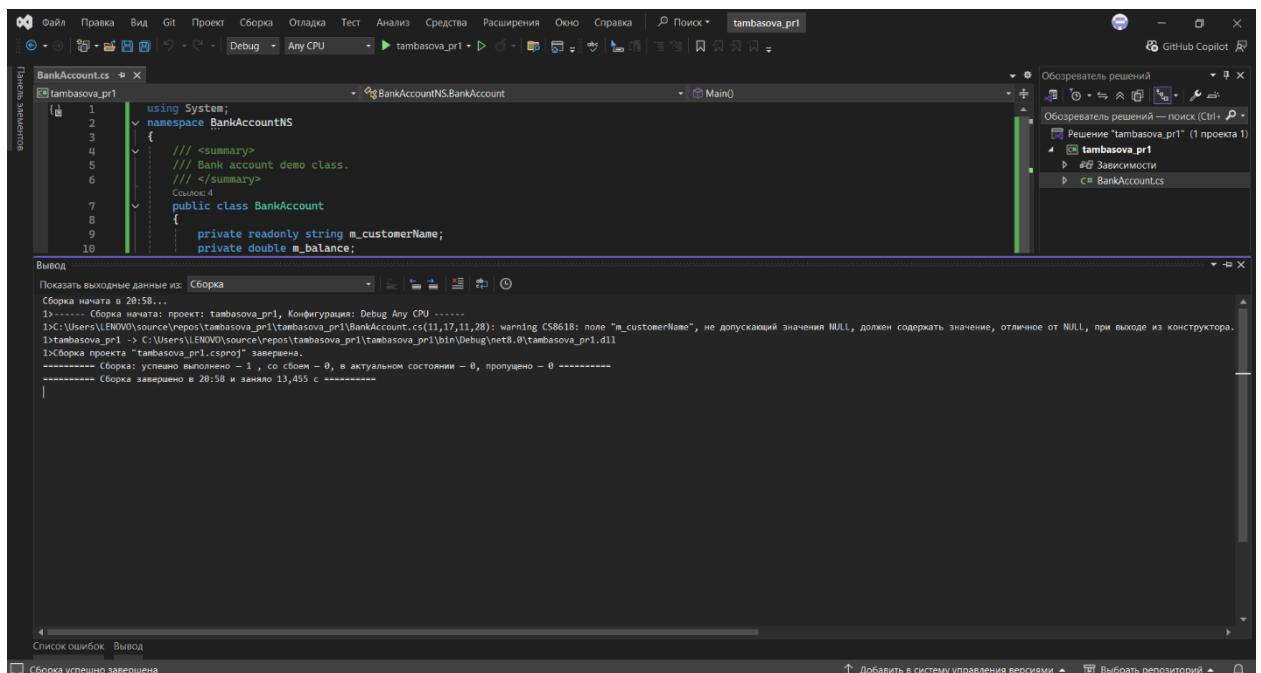


Рисунок 8 – Сборка – Собрать решение (2)

## 2 Создание проекта модульного теста

1. В меню Файл выбираю Добавить>Создать проект.

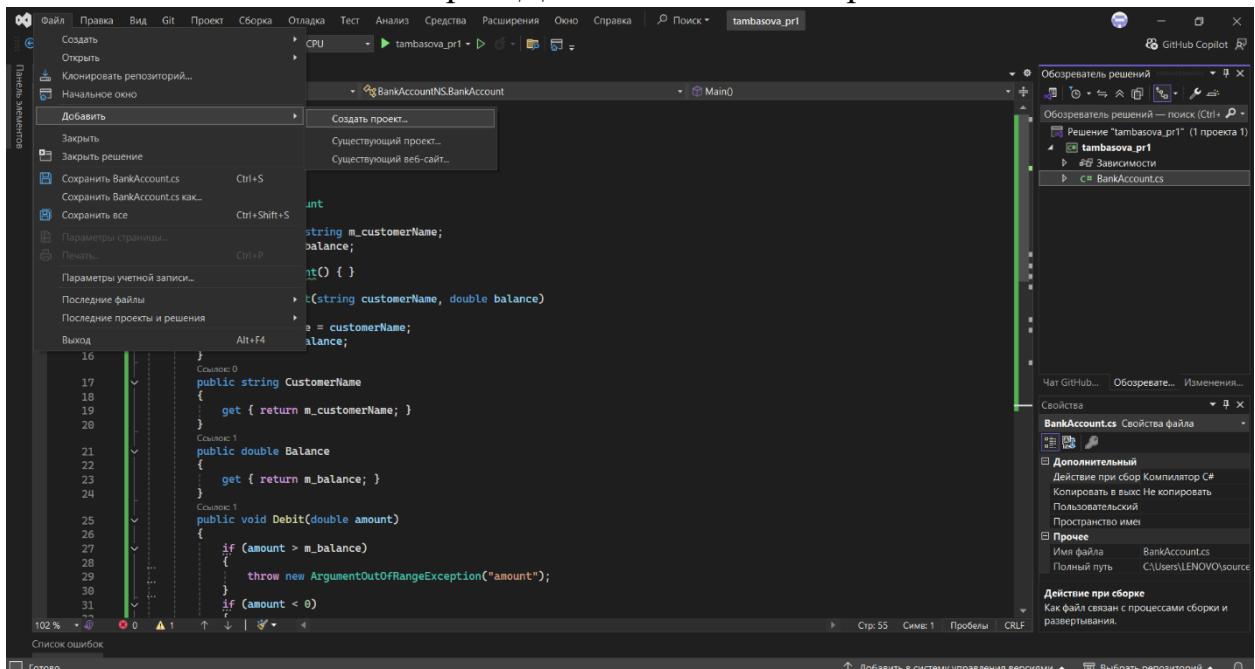


Рисунок 9 – Добавить>Создать проект

2. Выбираю шаблон Тестовый проект MSTest (Майкрософт).

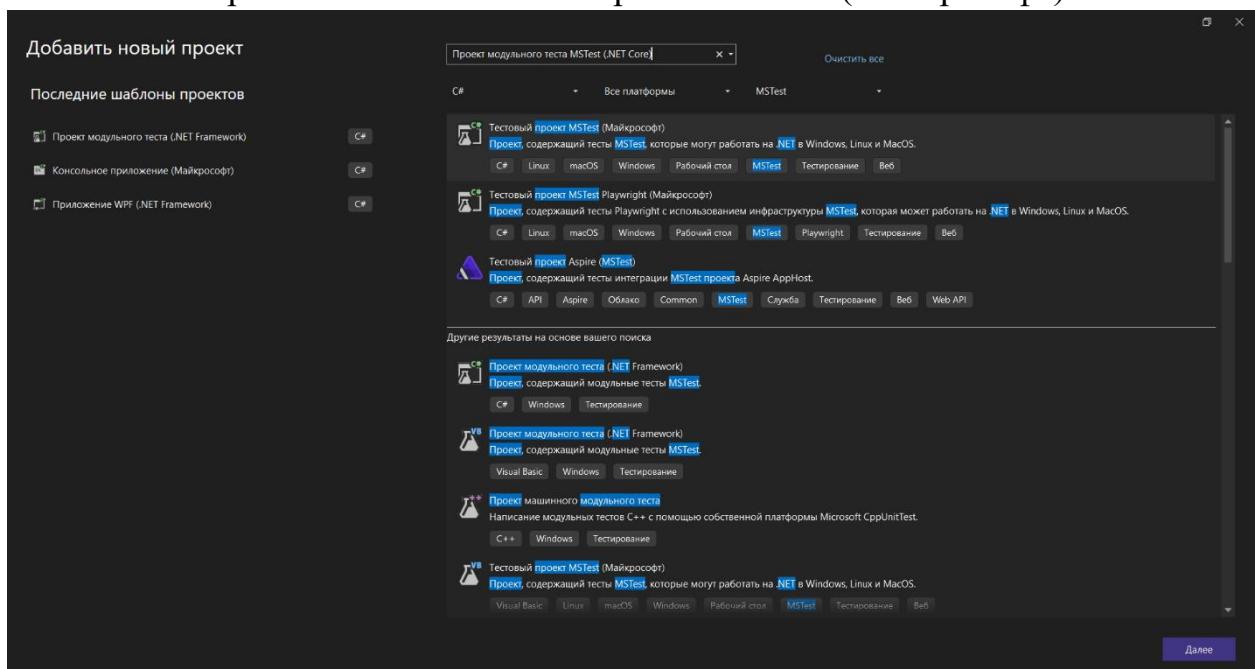


Рисунок 10 – Создание проекта Тестовый проект MSTest (Майкрософт)

3. Называю проект tambasova\_pr1\_tests.

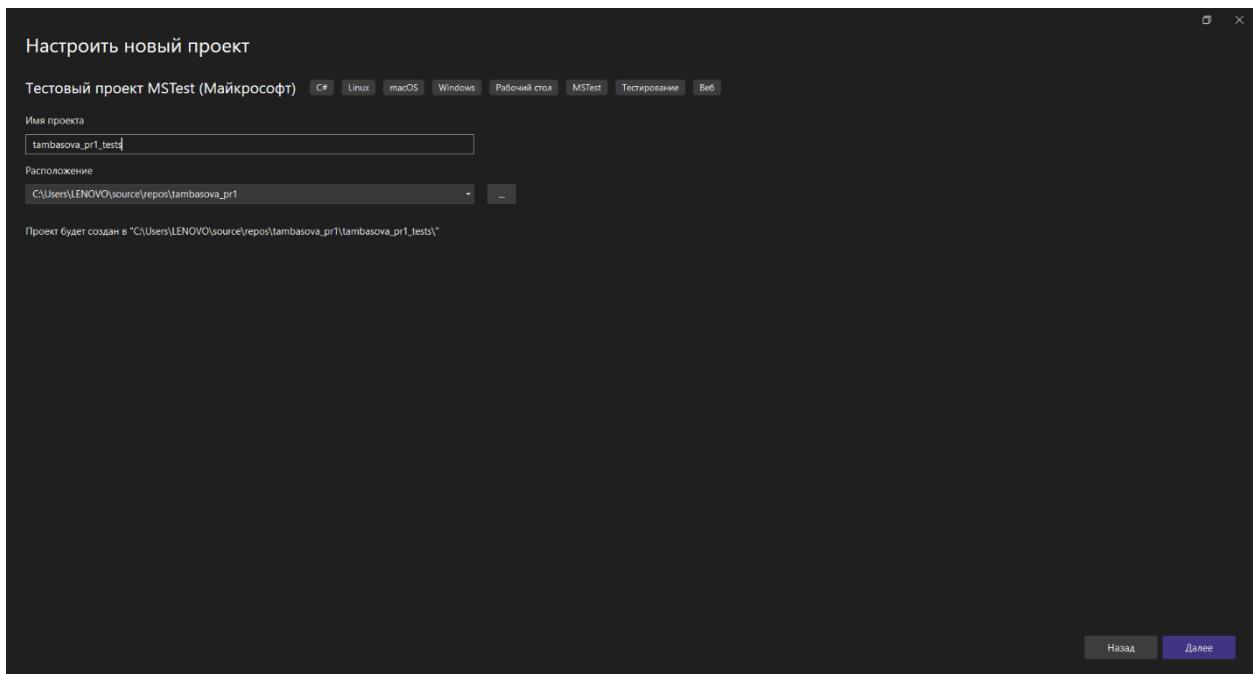


Рисунок 11 – Наименование проекта

4. Выбираю платформу .NET 8.0 (долгосрочная поддержка) и Средство выполнения тестов MSTest.

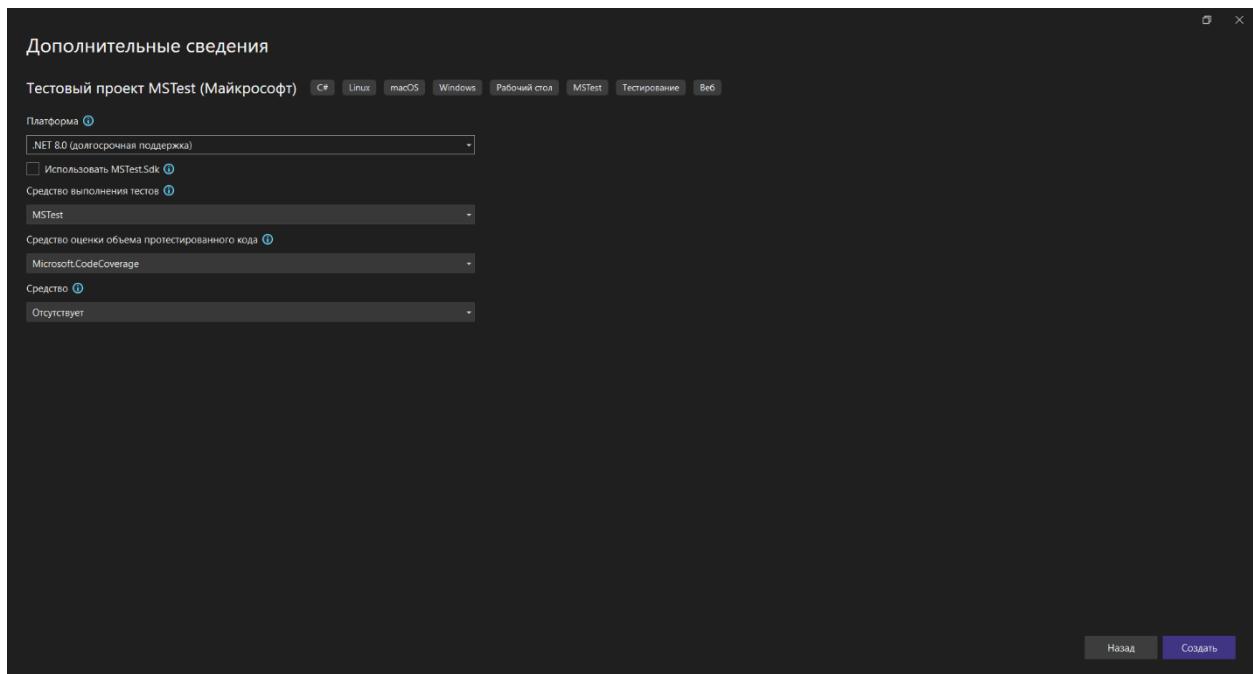


Рисунок 12 – Выбор платформы и средства выполнения тестов

## 5. Добавляю ссылку на проект tambasova\_pr1.

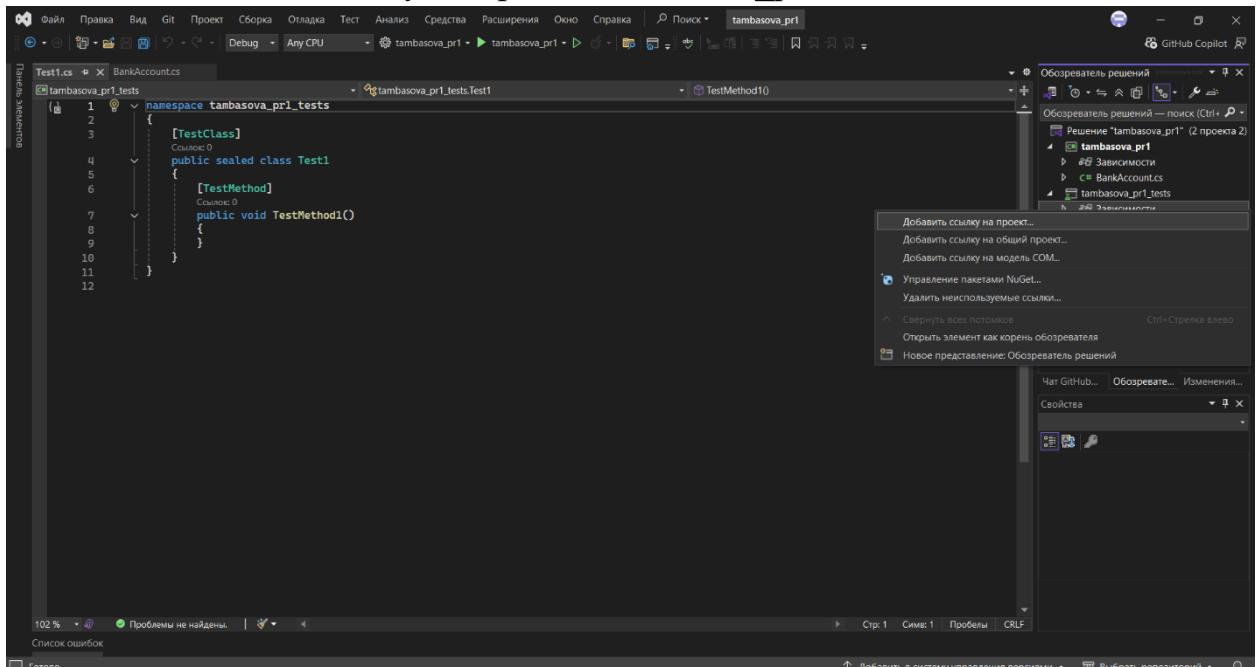


Рисунок 13 – Добавление ссылки на проект (1)

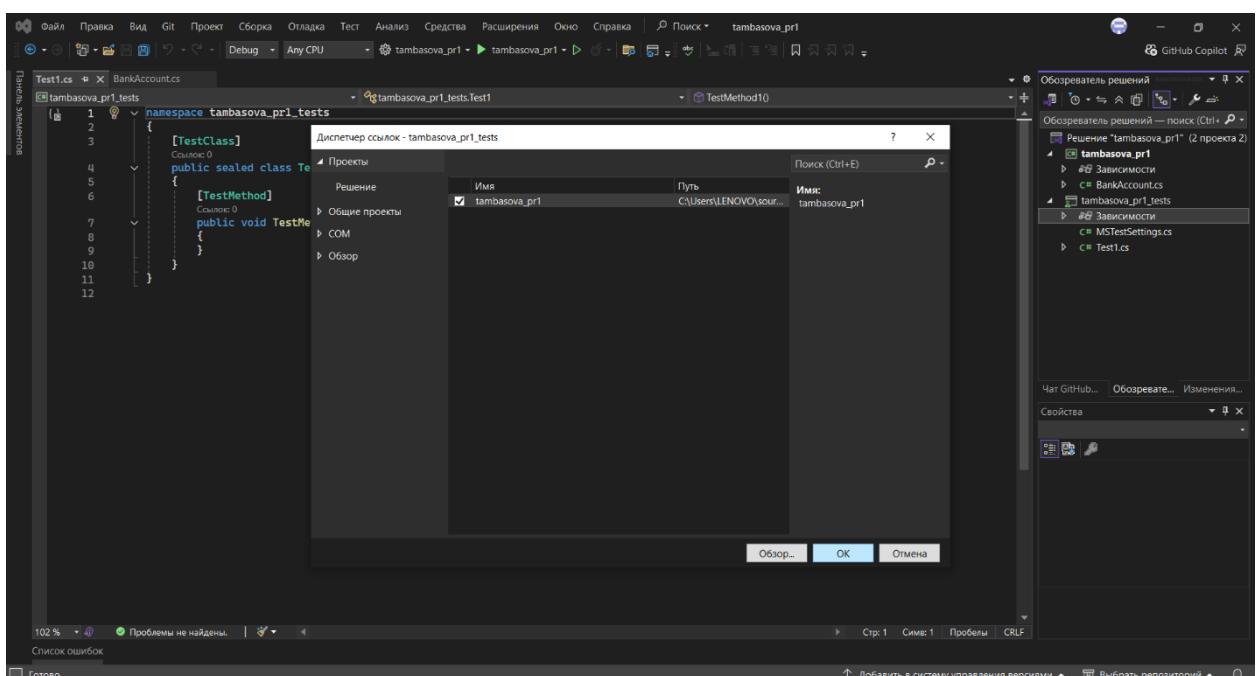


Рисунок 14 – Добавление ссылки на проект (2)

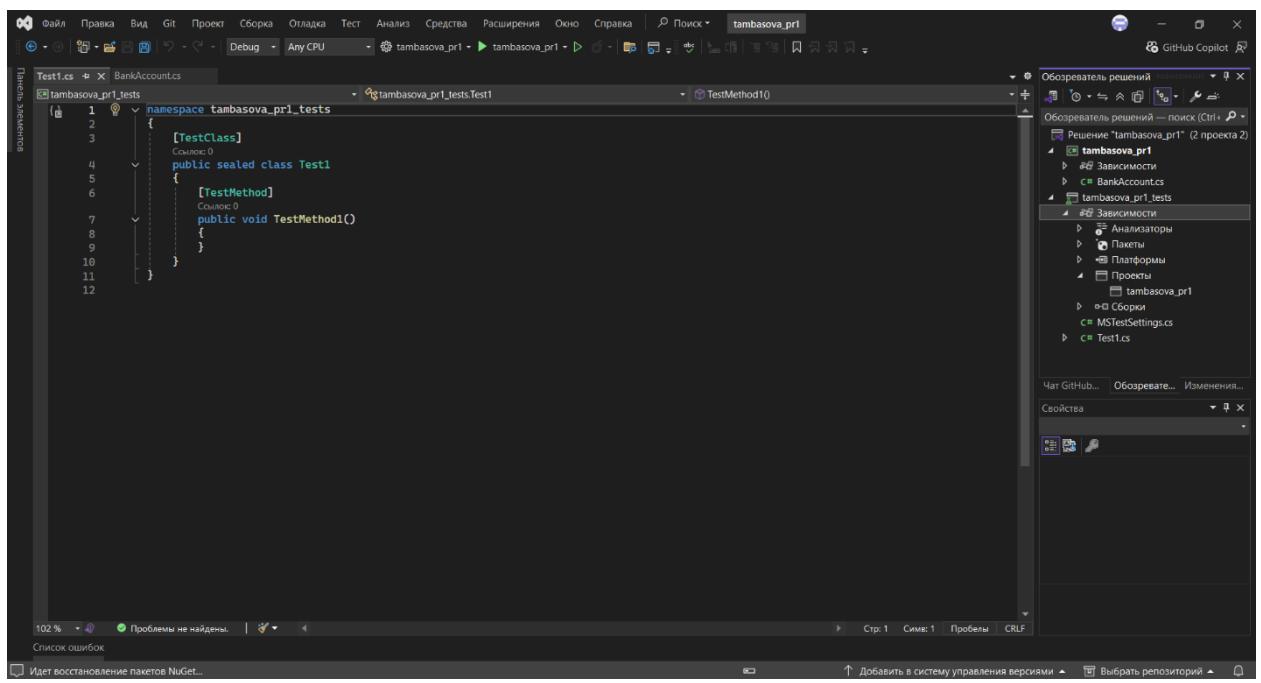


Рисунок 15 – Добавление ссылки на проект (3)

### 3 Создание тестового класса

1. Переименовываю файл Test1.cs в BankAccountTests.cs.

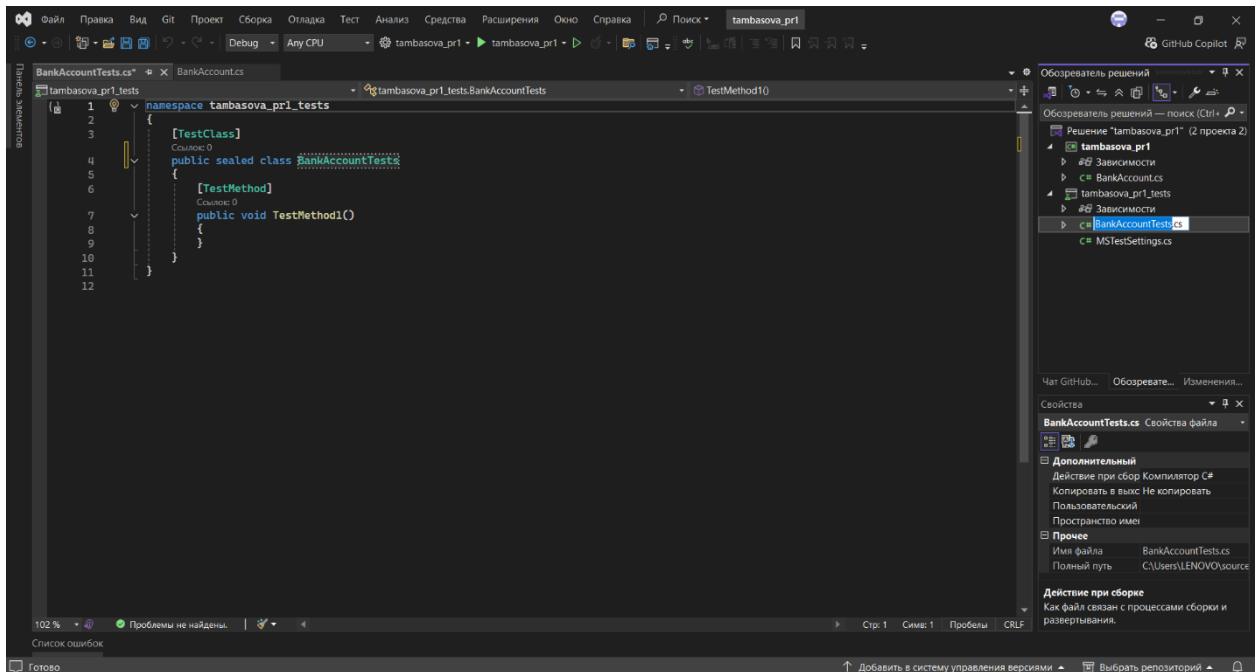


Рисунок 16 – Переименовывание файла в BankAccountTests.cs

2. Добавляю операторов using Microsoft.VisualStudio.TestTools.UnitTesting; и using BankAccountNS;;

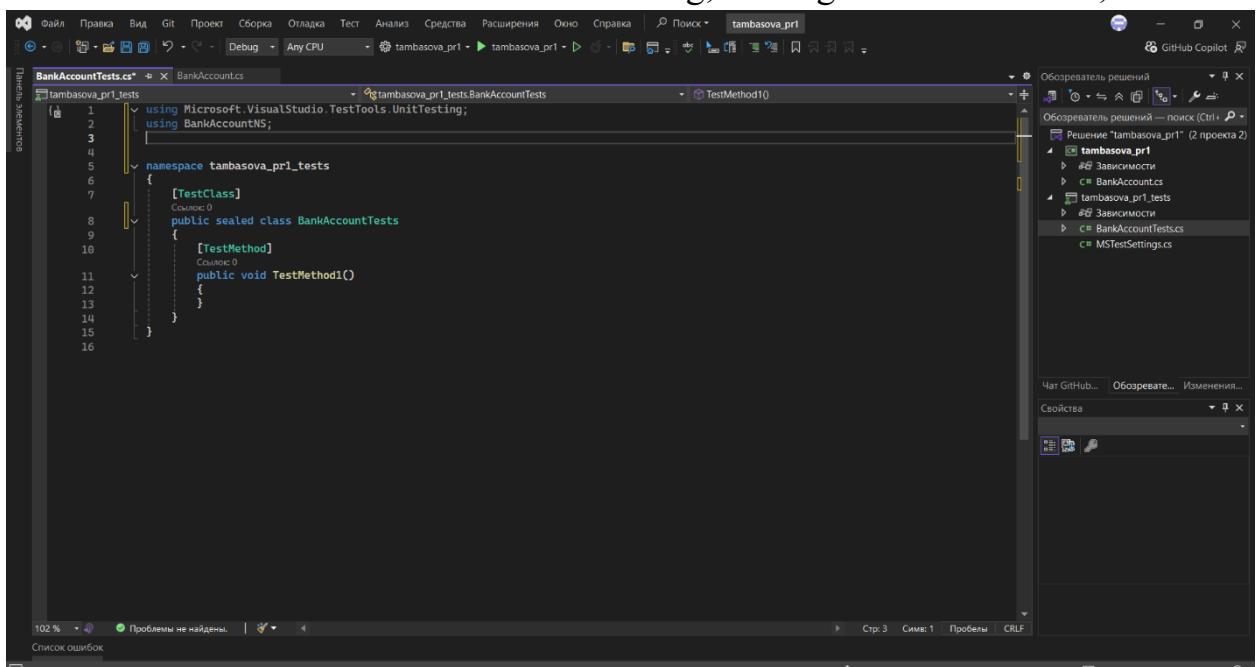
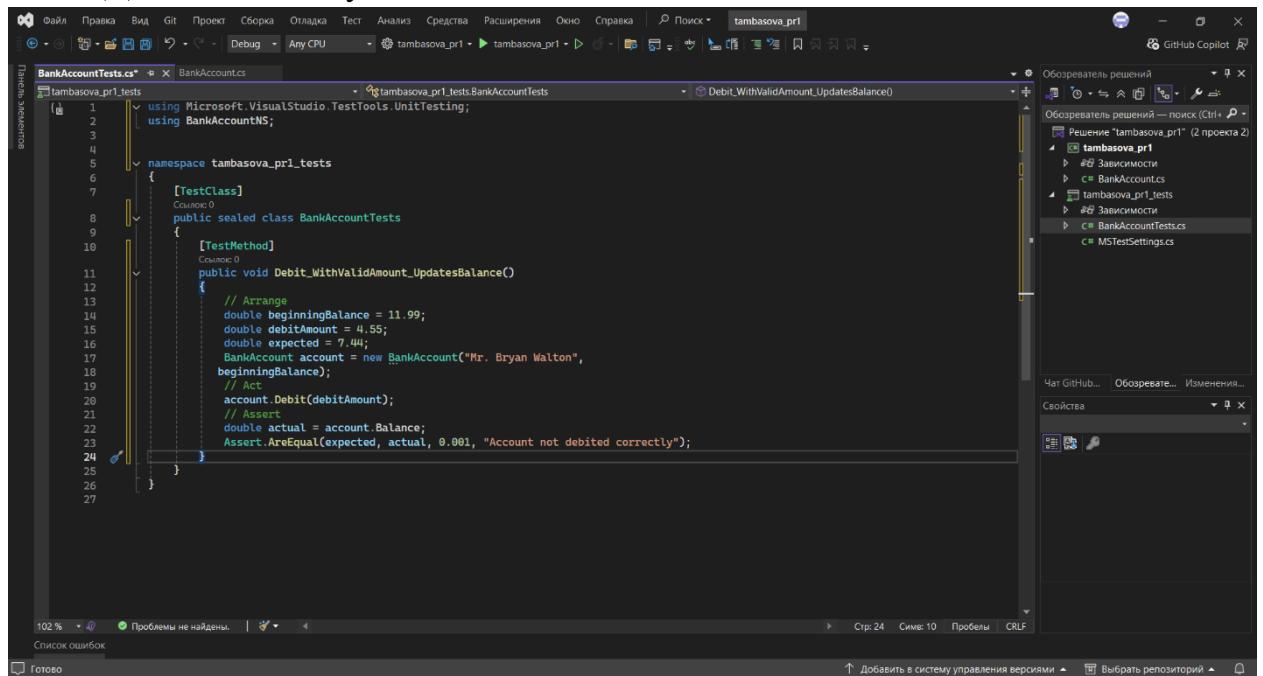


Рисунок 17 – Добавление операторов

## 4 Создание метода теста

Добавляю следующий метод в класс BankAccountTests.



The screenshot shows the Microsoft Visual Studio interface. The code editor window displays the file `BankAccountTests.cs` from the project `tambasova_pr1_tests`. The code defines a test class `BankAccountTests` with a single test method `Debit_WithValidAmount_UpdatesBalance`. The Solution Explorer on the right shows the solution structure with projects `tambasova_pr1` and `tambasova_pr1_tests`, and files like `BankAccount.cs` and `BankAccountTests.cs`.

```
1  using Microsoft.VisualStudio.TestTools.UnitTesting;
2  using BankAccountNS;
3
4  namespace tambasova_pr1_tests
5  {
6      [TestClass]
7      public sealed class BankAccountTests
8      {
9          [TestMethod]
10         public void Debit_WithValidAmount_UpdatesBalance()
11         {
12             // Arrange
13             double beginningBalance = 11.99;
14             double debitAmount = 4.55;
15             double expected = 7.44;
16             BankAccount account = new BankAccount("Mr. Bryan Walton",
17                 beginningBalance);
18             // Act
19             account.Debit(debitAmount);
20             // Assert
21             double actual = account.Balance;
22             Assert.AreEqual(expected, actual, 0.001, "Account not debited correctly");
23         }
24     }
25 }
```

Рисунок 18 – Добавление метода в BankAccountTests

## 5 Сборка и запуск теста

1. В меню Сборка нажимаю Собрать решение.

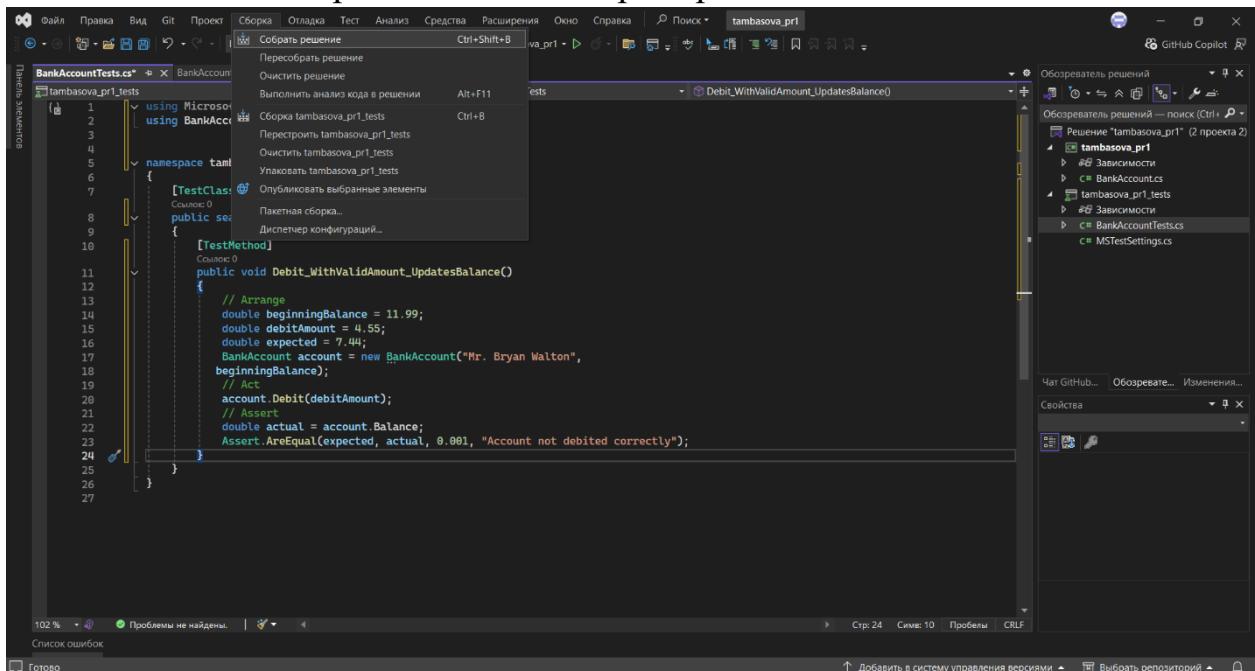


Рисунок 19 – Сборка – Собрать решение (1)

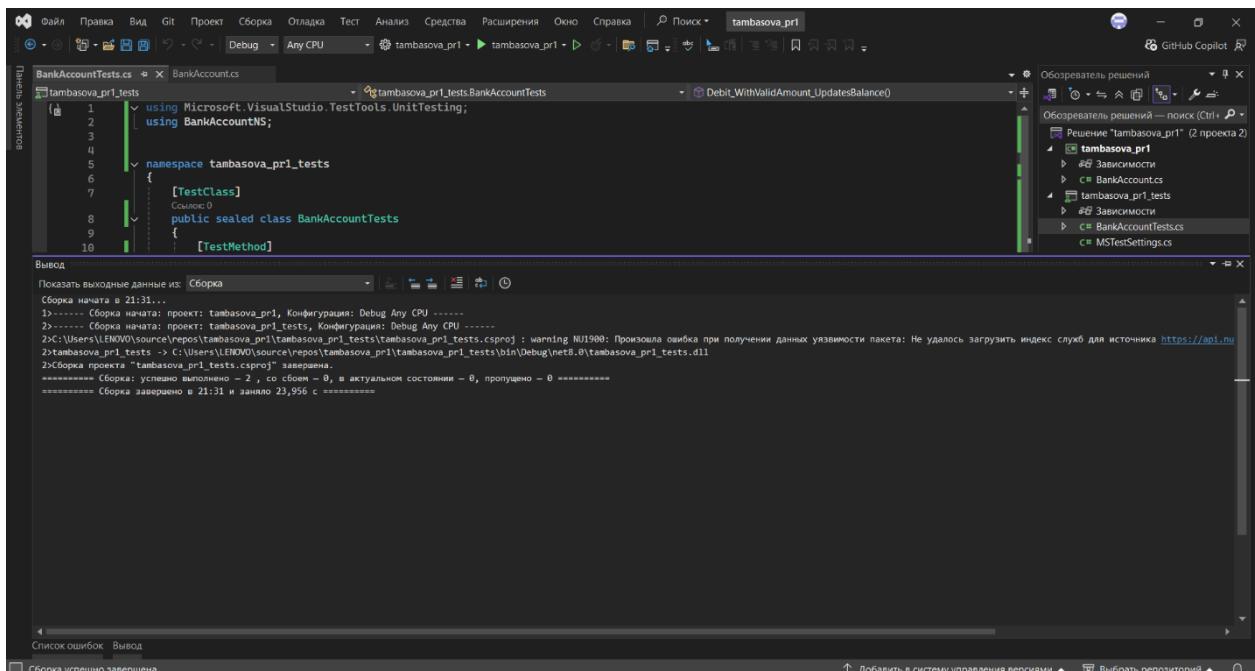


Рисунок 20 – Сборка – Собрать решение (2)

2. Открываю Обозреватель тестов выбрав вкладку Тест>Обозреватель тестов.

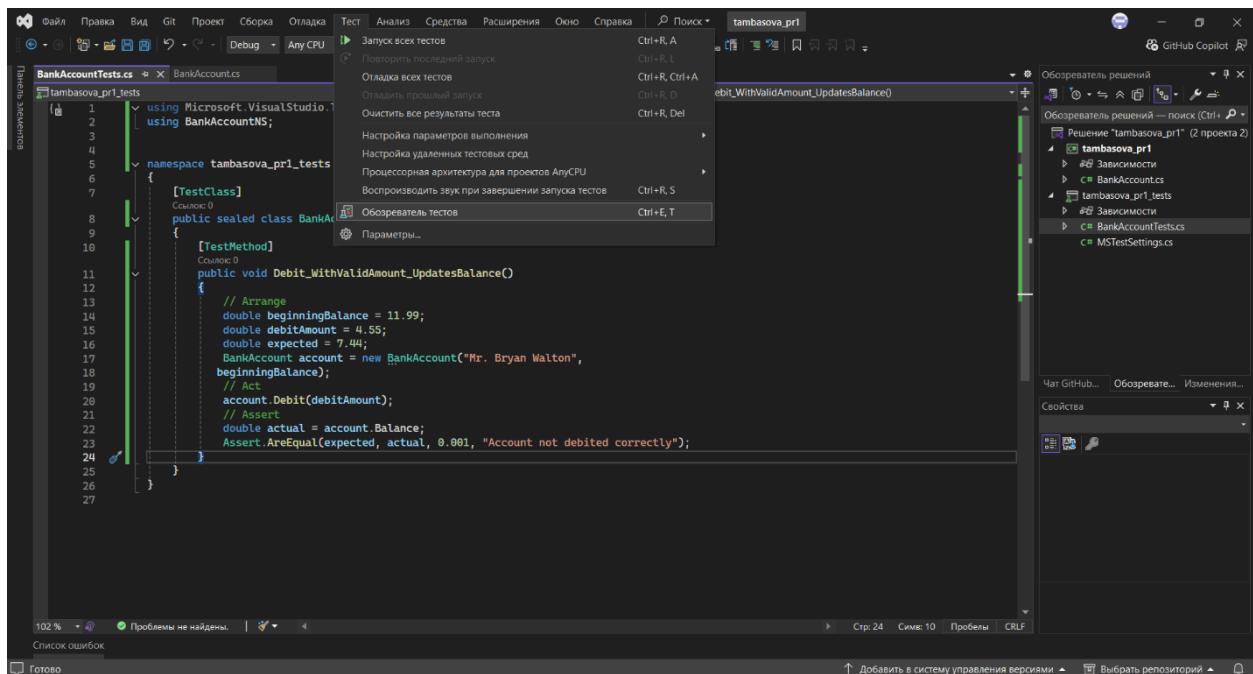


Рисунок 21 – Обозреватель тестов

3. Выбираю Выполнить все тесты в представлении, чтобы выполнить тест.

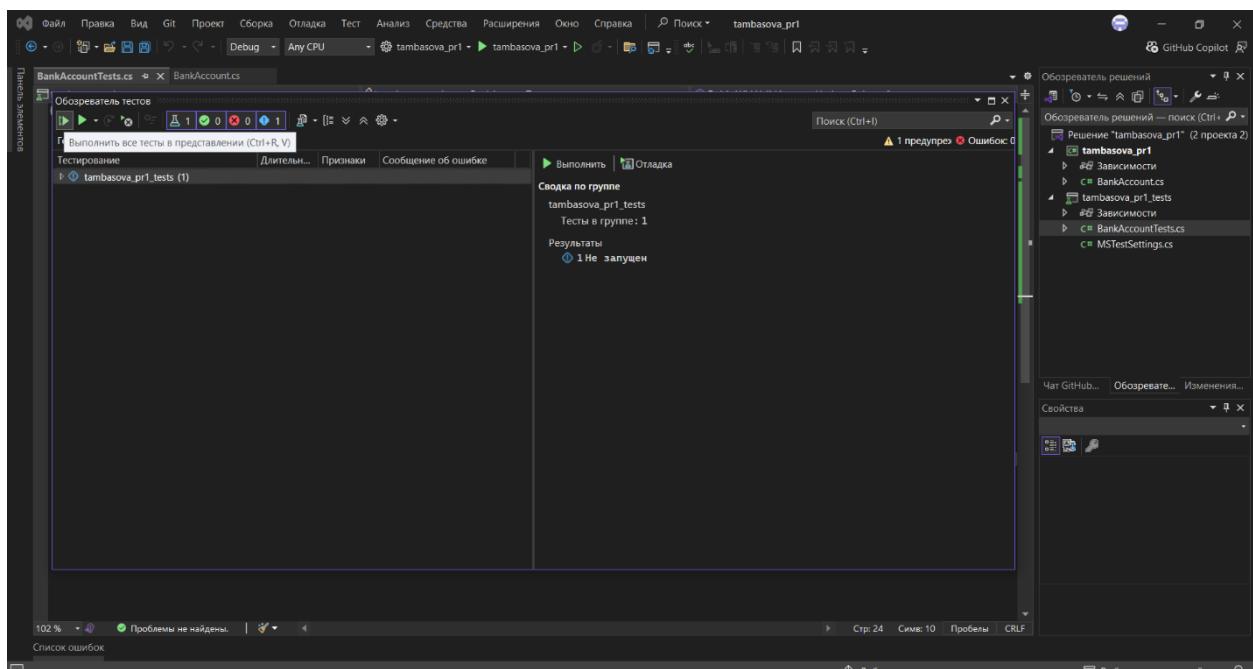


Рисунок 22 – Выполнить все тесты в представлении

#### 4. Тест не пройден.

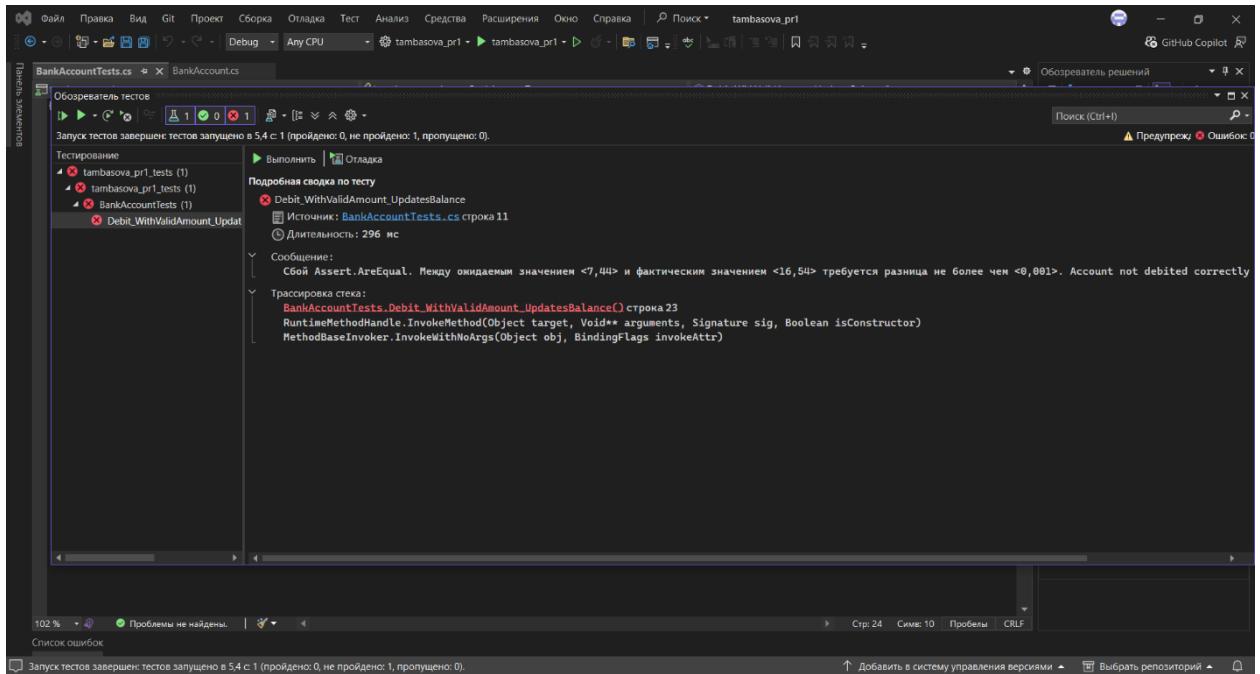
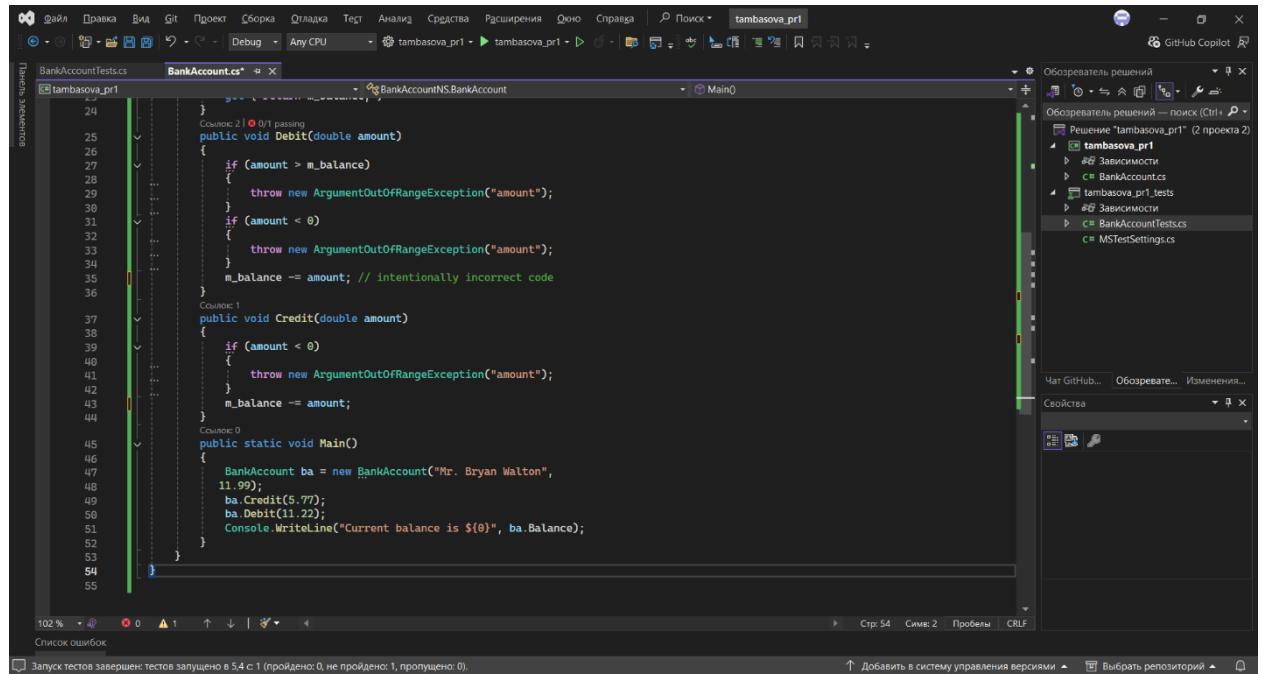


Рисунок 23 – Тест не пройден

## 6 Исправление кода и повторный запуск тестов

1. Исправляю ошибку того, что сумма списания добавляется на баланс счета, вместо того чтобы вычитаться. В файле BankAccount.cs заменяю строку `m_balance += amount;` на `m_balance -= amount;`.

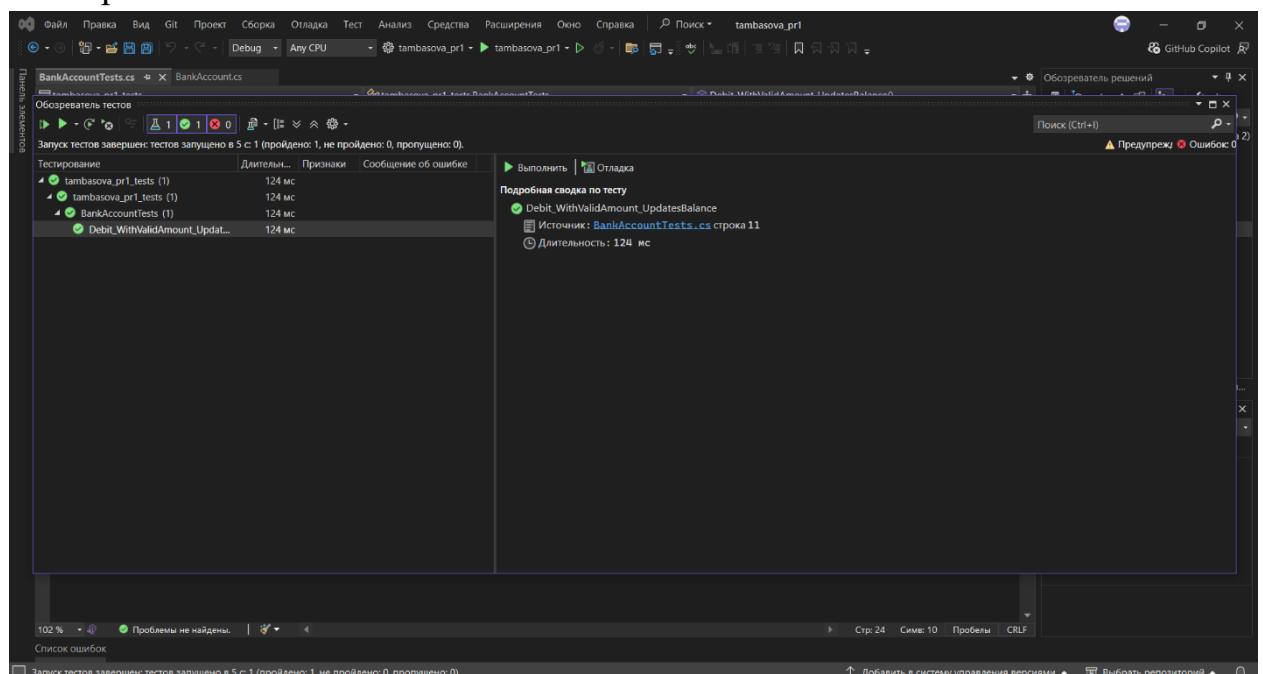


```
24     }
25     // Свойство 2 | 0/1 passing
26     public void Debit(double amount)
27     {
28         if (amount > m_balance)
29         {
30             throw new ArgumentOutOfRangeException("amount");
31         }
32         if (amount < 0)
33         {
34             throw new ArgumentOutOfRangeException("amount");
35         }
36         m_balance -= amount; // intentionally incorrect code
37     }
38     // Свойство 1
39     public void Credit(double amount)
40     {
41         if (amount < 0)
42         {
43             throw new ArgumentOutOfRangeException("amount");
44         }
45         m_balance += amount;
46     }
47     // Свойство 0
48     public static void Main()
49     {
50         BankAccount ba = new BankAccount("Mr. Bryan Walton",
51                                         11.99);
52         ba.Credit(5.77);
53         ba.Debit(11.22);
54         Console.WriteLine("Current balance is ${0}", ba.Balance);
55     }

```

Рисунок 24 – Исправление ошибки

2. В обозревателе тестов выбираю Запустить все, чтобы запустить тест повторно.



Запуск тестов завершен: тестов запущено в 5 с 1 (пройдено: 1, не пройдено: 0, пропущено: 0).

Тестирование	Длительн...	Признаки	Сообщение об ошибке
tambasova.pr1_tests (1)	124 мс		
tambasova.pr1_tests (1)	124 мс		
BankAccountTests (1)	124 мс		
Debit_WithValidAmount_UpdatesBalance	124 мс		

Выполнить | Отладка

Помощник по тестам

Debit\_WithValidAmount\_UpdatesBalance

Источник: BankAccountTests.cs строка 11

Длительность: 124 мс

Рисунок 25 – Повторный запуск

## 7 Использование модульных тестов для улучшения кода

### 7.1 Создание и запуск новых методов теста

1. Создаю метод теста для проверки правильного поведения в случае, когда сумма по дебету меньше нуля.

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for a unit test method:

```
25
26     [TestMethod]
27     [ExpectedException(typeof(ArgumentOutOfRangeException))]
28     public void Debit_WhenAmountIsLessThanZero_ShouldThrowArgumentOutOfRangeException()
29     {
30         // Arrange
31         double beginningBalance = 11.99;
32         double debitAmount = -100.00;
33         BankAccount account = new BankAccount("Mr. Bryan Walton", beginningBalance);
34
35         // Act and assert
36         Assert.ThrowsException<System.ArgumentOutOfRangeException>(() => account.Debit(debitAmount));
37     }
38 }
```

The code is part of a file named `BankAccountTests.cs` located in the `tambasova_pr1_tests` project. The `BankAccount` class is defined in a separate file `BankAccount.cs`. The `BankAccountTests.cs` file also contains other methods and classes. The `Debit_WhenAmountIsLessThanZero_ShouldThrowArgumentOutOfRangeException` method is annotated with `[ExpectedException]` to indicate that it expects an `ArgumentOutOfRangeException` to be thrown. The `Assert.ThrowsException` method is used to verify that the exception is indeed thrown when the debit amount is negative.

Рисунок 26 – Метод теста для проверки правильного поведения в случае, когда сумма по дебету меньше нуля

2. Создаю метод теста `Debit_WhenAmountIsMoreThanBalance_ShouldThrowArgumentOutOfRangeException` и копирую тело метода `Debit_WhenAmountIsLessThanZero_ShouldThrowArgumentOutOfRangeException`.

```

25
26     [TestMethod]
27     [DataSource(0)]
28     public void Debit_WhenAmountIsLessThanZero_ShouldThrowArgumentOutOfRangeException()
29     {
30         // Arrange
31         double beginningBalance = 11.99;
32         double debitAmount = -100.00;
33         BankAccount account = new BankAccount("Mr. Bryan Walton", beginningBalance);
34
35         // Act and assert
36         Assert.ThrowsException<System.ArgumentOutOfRangeException>(() => account.Debit(debitAmount));
37     }
38
39     [TestMethod]
40     [DataSource(0)]
41     public void Debit_WhenAmountIsMoreThanZero_ShouldThrowArgumentOutOfRangeException()
42     {
43         // Arrange
44         double beginningBalance = 11.99;
45         double debitAmount = 20.0;
46         BankAccount account = new BankAccount("Mr. Bryan Walton", beginningBalance);
47
48         // Act and assert
49         Assert.ThrowsException<System.ArgumentOutOfRangeException>(() => account.Debit(debitAmount));
50     }

```

**Рисунок 27 – Метод Debit\_WhenAmountIsMoreThanBalance\_ShouldThrowArgumentOutOfRangeException**

### 3. Сделала два теста. Оба прошли.

Тестирование	Длительность	Признаки	Сообщение об ошибке
tambasova.pr1.tests (3)	53 мс		
tambasova.pr1_tests (3)	53 мс		
BankAccountTests (3)	53 мс		
Debit_WithValidAmount_UpdatesBalance	22 мс		
Debit_WhenAmountsLessThanZero_ShouldThrowArgumentOutOfRangeException	2 мс		
Debit_WhenAmountsMoreThanBalance_ShouldThrowArgumentOutOfRangeException	29 мс		

**Рисунок 28 – Первый тест**

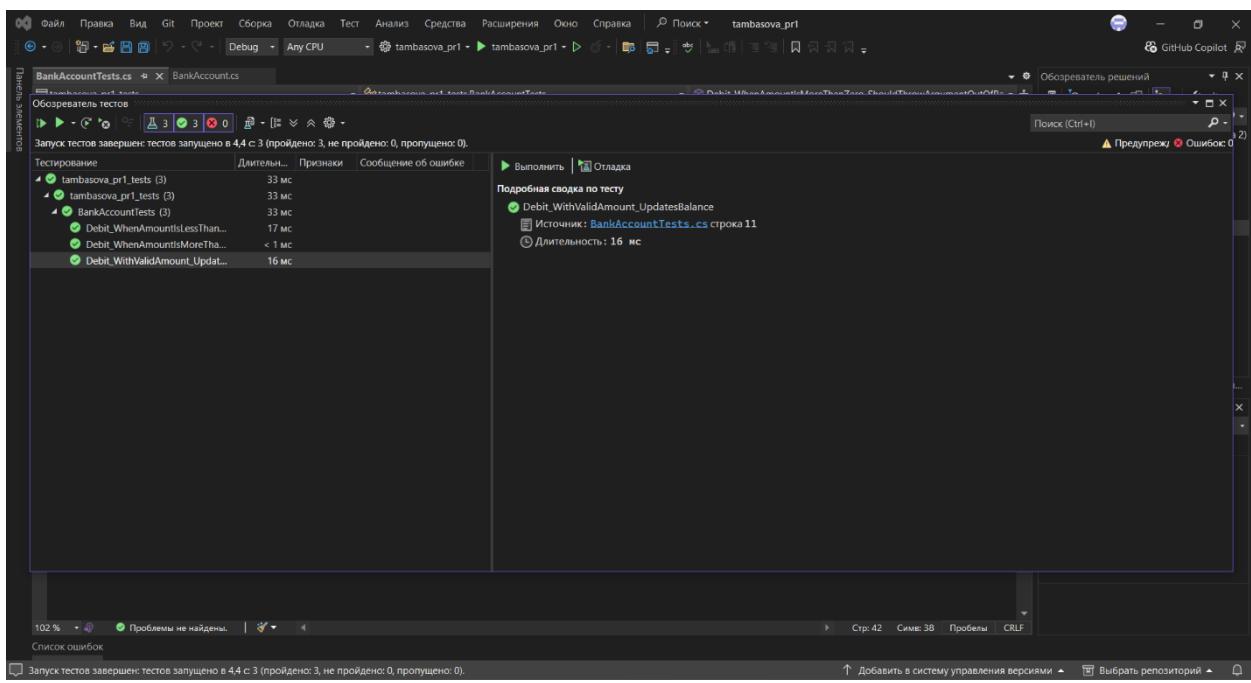


Рисунок 29 – Второй тест

## 7.2 Рефакторинг тестируемого кода

1. Добавляю в тестируемый класс (BankAccount) две константы для сообщений об ошибках в области видимости класса.

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code editor for `BankAccount.cs` within the `tambasova.pri` project. The code defines a `BankAccount` class with two new public constants: `DebitAmountExceedsBalanceMessage` and `DebitAmountLessThanZeroMessage`. The solution explorer on the right shows the project structure, including `BankAccountTests.cs` and `MTestSettings.cs`.

```
1 using System;
2 namespace BankAccountNS
3 {
4     /// <summary>
5     /// Bank account demo class.
6     /// </summary>
7     public class BankAccount
8     {
9         public const string DebitAmountExceedsBalanceMessage = "Debit amount exceeds balance";
10        public const string DebitAmountLessThanZeroMessage = "Debit amount is less than zero";
11        private readonly string _customerName;
12        private double _balance;
13        private BankAccount() { }
14        public BankAccount(string customerName, double balance)
15        {
16            _customerName = customerName;
17            _balance = balance;
18        }
19        public string CustomerName
20        {
21            get { return _customerName; }
22        }
23        public double Balance
24        {
25            get { return _balance; }
26        }
27        public void Debit(double amount)
28        {
29            if (amount > _balance)
30            {
31                throw new ArgumentOutOfRangeException("amount");
32            }
33            if (amount < 0)
34            {
35                throw new ArgumentOutOfRangeException("amount", amount, DebitAmountLessThanZeroMessage);
36            }
37            _balance -= amount; // intentionally incorrect code
38        }
39        public void Credit(double amount)
40        {
41            if (amount < 0)
42            {
43                throw new ArgumentOutOfRangeException("amount");
44            }
45            _balance += amount;
46        }
47        public static void Main()
48        {
49            BankAccount ba = new BankAccount("Mr. Bryan Walton",
50                11.99);
51            ba.Credit(5.77);
52            ba.Debit(11.22);
53            Console.WriteLine("Current balance is ${0}", ba.Balance);
54        }
55    }
56}
```

Рисунок 30 – Добавление двух констант для сообщений об ошибках в области видимости класса

2. Изменила два условных оператора в методе Debit.

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code editor for `BankAccount.cs` within the `tambasova.pri` project. The code defines a `BankAccount` class with the `Debit` method modified to use `if` statements instead of `if-else` blocks. The solution explorer on the right shows the project structure, including `BankAccountTests.cs` and `MTestSettings.cs`.

```
27 public void Debit(double amount)
28 {
29     if (amount > _balance)
30     {
31         throw new System.ArgumentOutOfRangeException("amount", amount, DebitAmountExceedsBalanceMessage);
32     }
33     if (amount < 0)
34     {
35         throw new System.ArgumentOutOfRangeException("amount", amount, DebitAmountLessThanZeroMessage);
36     }
37     _balance -= amount; // intentionally incorrect code
38 }
39 public void Credit(double amount)
40 {
41     if (amount < 0)
42     {
43         throw new ArgumentOutOfRangeException("amount");
44     }
45     _balance += amount;
46 }
47 public static void Main()
48 {
49     BankAccount ba = new BankAccount("Mr. Bryan Walton",
50         11.99);
51     ba.Credit(5.77);
52     ba.Debit(11.22);
53     Console.WriteLine("Current balance is ${0}", ba.Balance);
54 }
```

Рисунок 31 – Изменение двух условных операторов

## 7.3 Рефакторинг тестовых методов

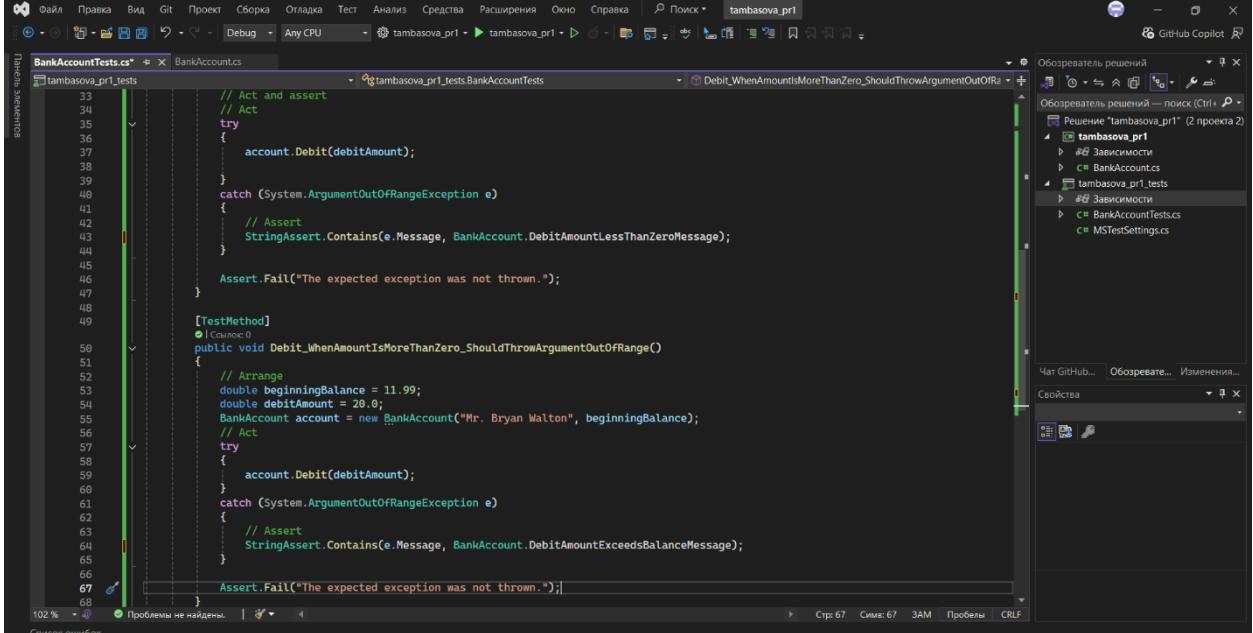
Заключаю вызов Debit() в блок try/catch.

```
28     // Arrange
29     double beginningBalance = 11.99;
30     double debitAmount = -100.00;
31     BankAccount account = new BankAccount("Mr. Bryan Walton", beginningBalance);
32     // Act and assert
33     try
34     {
35         account.Debit(debitAmount);
36     }
37     catch (System.ArgumentOutOfRangeException e)
38     {
39         // Assert
40         StringAssert.Contains(e.Message, BankAccount.DebitAmountExceedsBalanceMessage);
41     }
42 }
43
44
45
46 [TestMethod]
47 public void Debit_WhenAmountIsMoreThanZero_ShouldThrowArgumentOutOfRangeException()
48 {
49     // Arrange
50     double beginningBalance = 11.99;
51     double debitAmount = 20.0;
52     BankAccount account = new BankAccount("Mr. Bryan Walton", beginningBalance);
53     // Act
54     try
55     {
56         account.Debit(debitAmount);
57     }
58     catch (System.ArgumentOutOfRangeException e)
59     {
60         // Assert
61         StringAssert.Contains(e.Message, BankAccount.DebitAmountExceedsBalanceMessage);
62     }
63 }
```

Рисунок 32 – Блок try/catch

## 7.4 Повторное тестирование, переписывание и анализ

1. Добавляю утверждение `Assert.Fail` в конце тестового метода для обработки случая, когда исключение не создается.



```
    // Act and assert
    // ...
    try
    {
        account.Debit(debitAmount);
    }
    catch (System.ArgumentOutOfRangeException e)
    {
        // Assert
        StringAssert.Contains(e.Message, BankAccount.DebitAmountLessThanZeroMessage);
    }

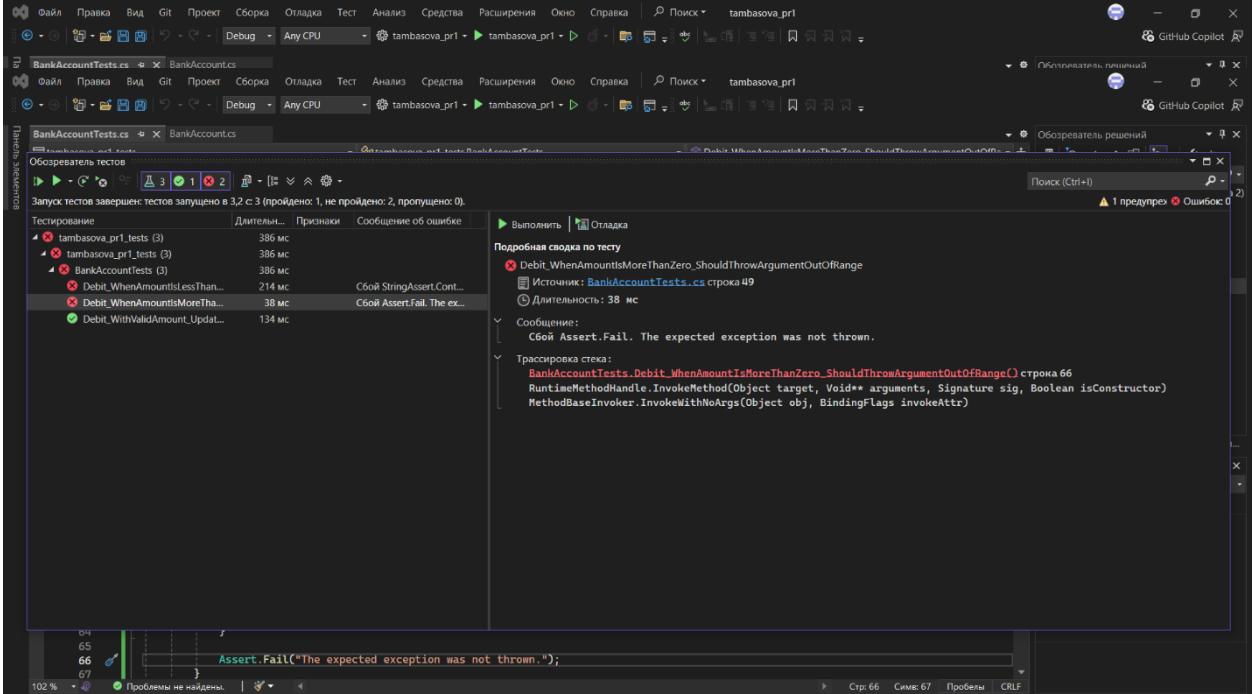
    Assert.Fail("The expected exception was not thrown.");
}

[TestMethod]
public void Debit_WhenAmountIsMoreThanZero_ShouldThrowArgumentOutOfRangeException()
{
    // Arrange
    double beginningBalance = 11.99;
    double debitAmount = 20.0;
    BankAccount account = new BankAccount("Mr. Bryan Walton", beginningBalance);
    // Act
    try
    {
        account.Debit(debitAmount);
    }
    catch (System.ArgumentOutOfRangeException e)
    {
        // Assert
        StringAssert.Contains(e.Message, BankAccount.DebitAmountExceedsBalanceMessage);
    }

    Assert.Fail("The expected exception was not thrown.");
}
```

Рисунок 33 – `Assert.Fail`

2. Тест не прошел.



Запуск тестов завершен: тестов запущено в 3.2 с 3 (пройдено: 1, не пройдено: 2, пропущено: 0).

Тестирование	Длительность	Признаки	Сообщения об ошибке
tambasova_pr1_tests (3)	386 мс		
tambasova_pr1_tests (3)	386 мс		
BankAccountTests (3)	386 мс		
Debit_WhenAmountIsLessThan... (1)	214 мс	Сбой StringAssert.Cont...	
Debit_WhenAmountIsMoreTha... (1)	38 мс	Сбой Assert.Fail. The ex...	
Debit_WithValidAmount_UpdatingBalance (1)	134 мс		

Рисунок 34 – Тест не прошел (1)

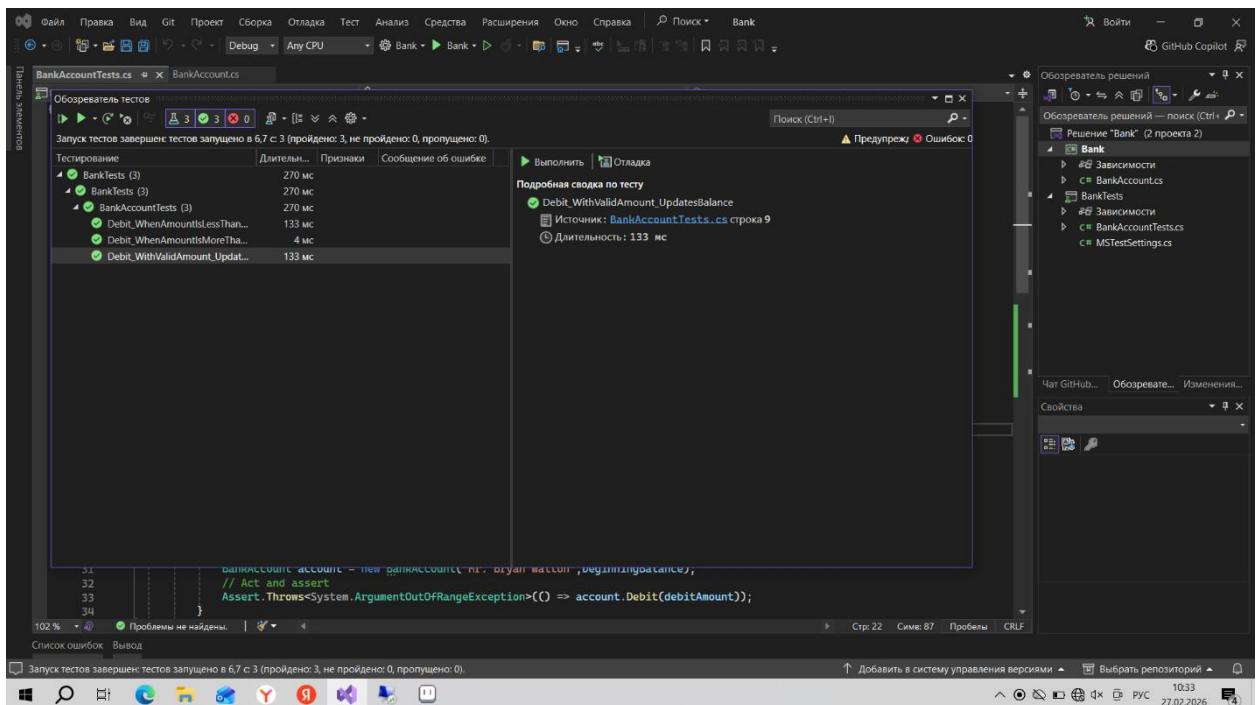


Рисунок 35 – Тест не прошел (2)

### 3. Добавляю оператор return;

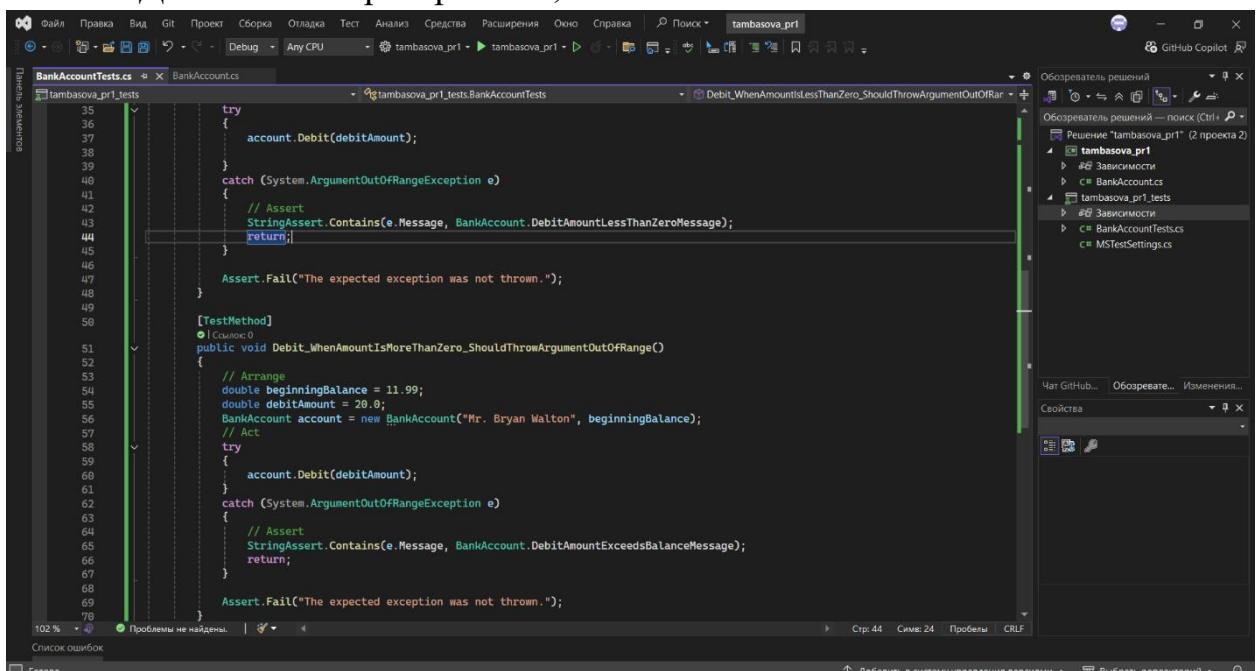


Рисунок 36 – Оператор return;

### 4. Тест прошел.

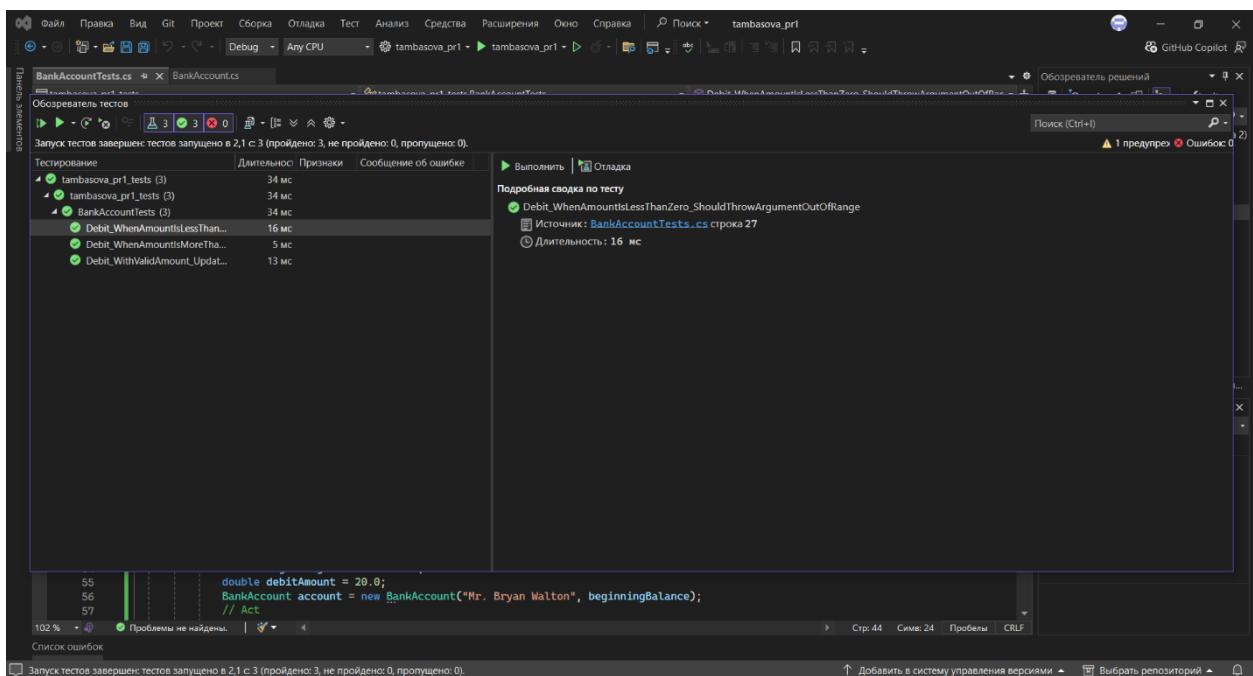


Рисунок 37 – Оператор return;