TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство образования и науки Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Институт кибернетики
Направление подготовки (специальность) Информационные системы и технологии
Кафедра вычислительной техники

Отчёт по курсовому проекту

Часть 1. Модульное тестирование проектов в MS Visual Studio 2013

по дисциплине «Управление качеством промышленного программного обеспечения»

Выполнили студенты гр. 8ИМ4А _____ Щукова К.Б.
(Подпись) Паршина Д.С.

____ _____ 2015 г.

Отчёт приняла:

Доцент каф. ВТ _____ Сергеева Е.Е.
(Подпись)

____ _____ 2015 г.

Томск 2015 г.

ВВЕДЕНИЕ

Целью данной работы является проведение модульного тестирования проектов, реализованных на языке программирования C# в среде MS Visual Studio, а также документирование полученных результатов после проведения тестирования.

Модульное тестирование основано на тестировании отдельных компонент программы, то есть классов и методов. Существующая методология, называемая Test Driven Development, – это разработка программных продуктов, используя модульные тесты. Она предполагает на начальном этапе создание наборов тестов для будущей функциональности, оценку всех вариантов выполнения и только потом реализация рабочего кода тестов.

Модульное тестирование будет проводиться в среде Visual Studio 2013 Ultimate, которая включает в себя встроенную систему тестирования Unit Testing Framework.

Для проведения модульного тестирования взято несколько видов проектов, реализованных на языке программирования C#:

1. DDL-библиотека классов «Логика».
2. Приложение типа Windows Form «Калькулятор».
3. Проект базы данных SQL интернет-магазина продаж.

1. ОПИСАНИЕ РЕЗУЛЬТАТОВ МОДУЛЬНОГО ТЕСТИРОВАНИЯ

1.1. Тестирование проекта базы данных SQL

Для проведения модульного тестирования за основу взят проект базы данных интернет-магазина продаж. Проект создан в Visual Studio 2013 на языке программирования C#.

Данный проект основан на скрипте SQL, реализующий создание схемы базы данных.

Содержимое скрипта описано ниже:

```
CREATE SCHEMA [Продажи]
    AUTHORIZATION [dbo];
CREATE TABLE [Продажи].[Клиент] (
    [IDКлиента]   INT           IDENTITY (1, 1) NOT NULL,
    [ИмяКлиента] NVARCHAR (40) NOT NULL,
    [ГодовоеКоличествоЗаказов]    INT           NOT NULL,
    [ГодовоеКоличествоПродаж]     INT           NOT NULL
);
CREATE TABLE [Продажи].[Заказы] (
    [IDКлиента] INT      NOT NULL,
    [IDЗаказа]    INT      IDENTITY (1, 1) NOT NULL,
    [ДатаЗаказа]  DATETIME NOT NULL,
    [ДатаСозданияЗаказа] DATETIME NULL,
    [СостояниеЗаказа]     CHAR (1) NOT NULL,
    [КоличествоЗаказов]     INT      NOT NULL
);
ALTER TABLE [Продажи].[Клиент]
    ADD CONSTRAINT [Продажи_Клиент_ГодовоеКолвоЗаказов] DEFAULT 0 FOR
[ГодовоеКоличествоЗаказов]; ALTER TABLE [Продажи].[Клиент]
    ADD CONSTRAINT [Продажи_Клиент_ГодовоеКолвоПродаж] DEFAULT 0 FOR
[ГодовоеКоличествоПродаж]; ALTER TABLE [Продажи].[Заказы]
    ADD CONSTRAINT [Продажи_Заказы_ДатаЗаказа] DEFAULT GetDate() FOR [ДатаЗаказа];
ALTER TABLE [Продажи].[Заказы]
    ADD CONSTRAINT [Продажи_Заказы_СостояниеЗаказа] DEFAULT 'O' FOR [СостояниеЗаказа];
ALTER TABLE [Продажи].[Клиент]
    ADD CONSTRAINT [PK_Продажи_IDКлиента] PRIMARY KEY CLUSTERED ([IDКлиента] ASC) WITH
(ALLOW_PAGE_LOCKS = ON, ALLOW_ROW_LOCKS = ON, PAD_INDEX = OFF, IGNORE_DUP_KEY = OFF,
STATISTICS_NORECOMPUTE = OFF); ALTER TABLE [Продажи].[Заказы]
    ADD CONSTRAINT [PK_Заказы_IDЗаказа] PRIMARY KEY CLUSTERED ([IDЗаказа] ASC) WITH
(ALLOW_PAGE_LOCKS = ON, ALLOW_ROW_LOCKS = ON, PAD_INDEX = OFF, IGNORE_DUP_KEY = OFF,
STATISTICS_NORECOMPUTE = OFF); ALTER TABLE [Продажи].[Заказы]
```

```
    ADD CONSTRAINT [FK_Заказы_Клиент_IDКлиента] FOREIGN KEY ([IDКлиента]) REFERENCES
[Продажи].[Клиент] ([IDКлиента]) ON DELETE NO ACTION ON UPDATE NO ACTION; ALTER TABLE
[Продажи].[Заказы]
    ADD CONSTRAINT [FK_Заказы_ДатаСозданияЗаказа] CHECK ((ДатаСозданияЗаказа >=
ДатаЗаказа) AND (ДатаСозданияЗаказа < '01/01/2030')); ALTER TABLE [Продажи].[Заказы]
    ADD CONSTRAINT [FK_Заказы_ДатаЗаказа] CHECK ((ДатаЗаказа > '01/01/2014') and
(ДатаЗаказа < '01/01/2030'));
CREATE PROCEDURE [Продажи].[ОтменаЗаказа]
@IDЗаказа INT
AS
BEGIN
DECLARE @Дельта INT, @IDКлиента INT
BEGIN TRANSACTION
    SELECT @Дельта = [КоличествоЗаказов], @IDКлиента = [IDКлиента]
     FROM [Продажи].[Заказы] WHERE [IDЗаказа] = @IDЗаказа;
UPDATE [Продажи].[Заказы]
   SET [СостояниеЗаказа] = 'X'
WHERE [IDЗаказа] = @IDЗаказа;
UPDATE [Продажи].[Клиент]
   SET
   ГодовоеКоличествоЗаказов = ГодовоеКоличествоЗаказов - @Дельта
    WHERE [IDКлиента] = @IDКлиента
COMMIT TRANSACTION
END
CREATE PROCEDURE [Продажи].[СозданиеЗаказа]
@IDЗаказа INT, @ДатаСозданияЗаказа DATETIME
AS
BEGIN
DECLARE @Дельта INT, @IDКлиента INT
BEGIN TRANSACTION
    SELECT @Дельта = [КоличествоЗаказов], @IDКлиента = [IDКлиента]
     FROM [Продажи].[Заказы] WHERE [IDЗаказа] = @IDЗаказа;
UPDATE [Продажи].[Заказы]
   SET [СостояниеЗаказа] = 'F',
       [ДатаСозданияЗаказа] = @ДатаСозданияЗаказа
WHERE [IDЗаказа] = @IDЗаказа;
UPDATE [Продажи].[Клиент]
   SET
   ГодовоеКоличествоПродаж = ГодовоеКоличествоПродаж - @Дельта
    WHERE [IDКлиента] = @IDКлиента
COMMIT TRANSACTION
END
CREATE PROCEDURE [Продажи].[СоздатьНовогоКлиента]
@ИмяКлиента NVARCHAR (40)
AS
BEGIN
INSERT INTO [Продажи].[Клиент] (ИмяКлиента) VALUES (@ИмяКлиента);
SELECT SCOPE_IDENTITY()
END
CREATE PROCEDURE [Продажи].[РазместитьНовыйЗаказ]
@IDКлиента INT, @КоличествоЗаказов INT, @ДатаЗаказа DATETIME, @СостояниеЗаказа CHAR
(1)='O'
AS
```

```
BEGIN
DECLARE @Количество INT
BEGIN TRANSACTION
INSERT INTO [Продажи].[Заказы] (IDКлиента, ДатаЗаказа, ДатаСозданияЗаказа,
СостояниеЗаказа, КоличествоЗаказов)
     VALUES (@IDКлиента, @ДатаЗаказа, NULL, @СостояниеЗаказа, @КоличествоЗаказов)
SELECT @Количество = SCOPE_IDENTITY();
UPDATE [Продажи].[Клиент]
   SET
   ГодовоеКоличествоЗаказов = ГодовоеКоличествоЗаказов + @КоличествоЗаказов
    WHERE [IDКлиента] = @IDКлиента
COMMIT TRANSACTION
RETURN @Количество
END
CREATE PROCEDURE [Продажи].[ПоказатьДеталиЗаказов]
@IDКлиента INT=0
AS
BEGIN
SELECT [C].[ИмяКлиента], CONVERT(date, [O].[ДатаЗаказа]), CONVERT(date, [O].
[ДатаСозданияЗаказа]), [O].[СостояниеЗаказа], [O].[КоличествоЗаказов]
  FROM [Продажи].[Клиент] AS C
  INNER JOIN [Продажи].[Заказы] AS O
     ON [O].[IDКлиента] = [C].[IDКлиента]
  WHERE [C].[IDКлиента] = @IDКлиента
END
CREATE TABLE [Продажи].[Пользователи](
  [id] [int] IDENTITY(1,1) NOT NULL,
  [логин] [nchar](100) COLLATE CI_AS NOT NULL,
  [хэш] [nchar](50) COLLATE CI_AS NOT NULL,
  [уровень] [int] NULL,
CONSTRAINT [PK_Users] PRIMARY KEY CLUSTERED
([id] ASC)
WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON))
CREATE UNIQUE NONCLUSTERED INDEX [IX_UserName] ON [Продажи].[Пользователи]
([логин] ASC)
WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON)
INSERT [Продажи].[Пользователи] ([логин], [хэш], [уровень]) VALUES (N'Kristina',
N'827ccb0eea8a706c4c34a16891f84e7b', 1)
INSERT [Продажи].[Пользователи] ([логин], [хэш], [уровень]) VALUES (N'Tester',
N'3095C3E4F1465133E5E6BE134EB2EBE2', 1)
INSERT [Продажи].[Пользователи] ([логин], [хэш], [уровень]) VALUES (N'Admin',
N'E3AFED0047B08059D0FADA10F400C1E5', 2)
INSERT [Продажи].[Пользователи] ([логин], [хэш], [уровень]) VALUES (N'User1',
N'827CCB0EEA8A706C4C34A16891F84E7B', 1)
GO
CREATE PROCEDURE [Продажи].[ПолучитьIdПользователя]
  @login nchar(100),
  @hash nchar(50)
AS
BEGIN
```

```
SELECT ID FROM [Продажи].Пользователи WHERE логин = @login AND хэш = @hash
END
GO
```

В результате выполнения скрипта создаются таблицы: **Заказы, Клиент** и **Пользователи**. Также в скрипте создаются хранимые процедуры, которые необходимо протестировать:

1. ПоказатьДеталиЗаказа.
2. РазместитьНовыйЗаказ.
3. СозданиеЗаказа.
4. СоздатьНовогоКлиента.

Подробное описание перечисленных выше хранимых процедур приведено в таблице 1.1.1.

Таблица 1.1.1 – Описания хранимых процедур для тестирования

Хранимая процедура проекта	**Назначение**
1. СозданиеНовогоКлиента	Добавление новой записи в таблицу Клиент, в которой ГодовоеКоличествоПродаж и ГодовоеКоличествоЗаказов равно нулевым значениям.
2. РазместитьНовыйЗаказ	Добавление записи в таблицу Заказы для конкретного Клиента и обновление значения ГодовоеКоличествоЗаказов на соответствующее значение из таблицы Клиент.
3. СозданиеЗаказа	Обновление записи в таблице Заказы посредством изменения состояния заказа «O» на «F» и увеличение значения ГодовоеКоличествоПродаж для соответствующей записи в таблице Клиент.
4. ПоказатьДеталиЗаказа	Процедура объединяет таблицу Заказы с Клиентом и отображает записи для конкретного клиента.
5. ОтменаЗаказа	Процедура обновляет запись в таблице Заказы и Клиент посредством изменения состояния заказа «O» на «X» (Отменен) и уменьшает ГодовоеКоличествоЗаказов на соответствующую запись в таблице Клиент.

Данный скрипт использован для импорта схемы БД в проекте. Модульное тестирование выполнено для вышеуказанных хранимых процедур.

В MS Visual Studio 2013 выбрана команда Create Unit Tests. Результаты тестирования сведены в таблицу 1.1.2. Для проведения модульного тестирования созданы 4 тестовых случая.

Таблица 1.1.2 – Описание тестовых случаев для схемы БД интернет-магазина

Предусловие	**Тестовый сценарий (шаги)**	**Ожидаемый результат**	**Фактический результат**	**Послесловие**
Добавление	1. Выполнить запрос типа	Проверить, что	Тест пройден.	Удаление клиента

нового клиента с именем Client1.	SELECT для получения имени созданного клиента. 2. Выполнить хранимую процедуру «СоздатьНовогоКлиента». 3. Выполнить запрос типа SELECT для просмотра всех записей в таблице «Клиент».	таблица Клиент содержит одну запись после запуска хранимой процедуры.		с именем Client1.
Добавление нового клиента с именем Client1 и удаление старых размещенных заказов, связанных с этим клиентом.	1. Выполнить запрос типа SELECT для получения имени созданного клиента. 2. Выполнить хранимую процедуру «РазместитьНовыйЗаказ». 3. Выполнить запрос типа SELECT, возвращающий годовую сумму заказов, сделанных клиентом с именем Client1.	Проверить, что годовая сумма заказов у клиента Client1 составляет 100 руб., а годовая сумма продаж = 0 руб.	Тест пройден.	Удаление клиента с именем Client1 и всех его размещенных заказов.
Добавление нового клиента с именем Client1, удаление всех старых заказов данного клиента и размещение для него заказа на сумму 100 руб.	1. Выполнить запрос типа SELECT для получения имени созданного клиента. 2. Выполнить запрос типа SELECT для подсчета максимального количества заказов, сделанных клиентом Client1. 3. Выполнить хранимую процедуру «СозданиеЗаказа». 4. Выполнить запрос типа SELECT, возвращающий годовую сумму продаж интернет-магазина, выполненных для клиента Client1.	Проверить, что общая сумма годовых заказов и годовых продаж = 100 руб.	Не пройден. Ошибка условия скалярного выражения: значения не совпадают фактическое -100, ожидаемое 100.	Удаление клиента с именем Client1 и всех его размещенных заказов.
Добавление клиента с именем Client2 и размещения для него заказов на сумму 100, 50 и 5 руб.	1. Выполнить запрос типа SELECT для получения имени созданного клиента. 2. Выполнить хранимую процедуру «ПоказатьДеталиЗаказа». 3. Выполнить запрос типа SELECT для просмотра характеристик размещенного заказа для клиента Client2.	Проверить, что ХП возвращает правильные номера столбцов, и что результирующие данные имеют ожидаемую контрольную сумму.	Тест пройден.	Удаление клиента с именем Client2 и всех его размещенных заказов.
Обновление хранимой процедуру для намеренного внесения в нее неправильных условий.	1. Выполнить запрос типа UPDATE для хранимой процедуры «ОтменаЗаказа» с целью внесения условий невозможности отмены выполненного заказа	Проверить, что тестовый случай должен завершится ошибкой в случае попытки отмены заказа, который уже был создан и	Тест пройден.	Обновление хранимой процедуры для возврата к начальным условиям.

	клиентом. 2. Выполнить запрос типа SELECT для получения имени созданного клиента. 3. Выполнить запрос типа SELECT для подсчета максимального количества заказов, сделанных клиентом Client1. 4. Выполнить хранимую процедуру «ОтменаЗаказа». 5. Выполнить запрос типа SELECT для просмотра неотмененного заказа со статусом «Выполнен» для клиента Client1.	имеет статус «Выполнен (О)». Намеренное внесение условий, которые приведут к ошибке при выполнение тестового случая. Тест должен выполниться с ошибкой «Невозможно отменить выполненный заказ».		

На рисунке 1.1.1 представлены выбранные хранимые процедуры для тестирования.

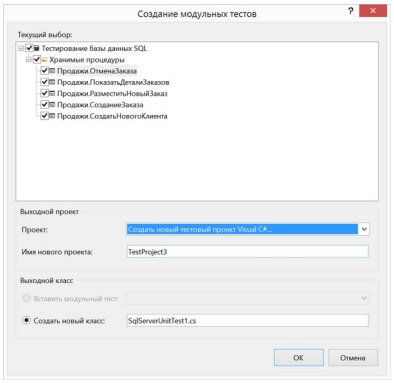

Рисунок 1.1.1 – Выбранные хранимые процедуры для тестирования

Далее настроена конфигурация тестов SQL Server (рис.1.1.2).

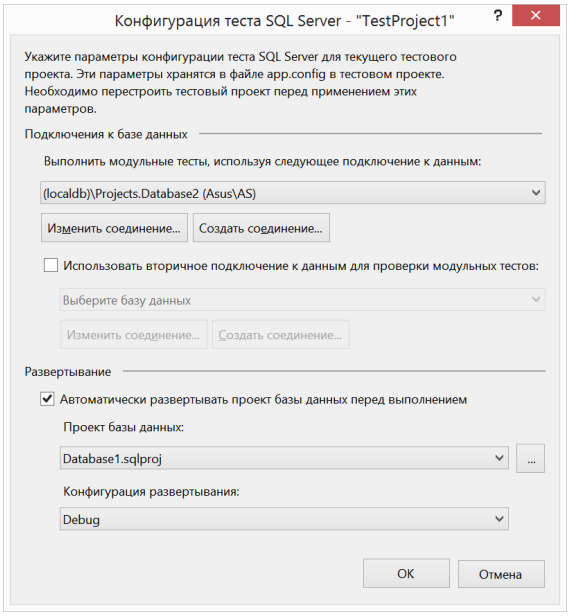

Рисунок 1.1.2 – Настройка конфигурации тестов SQL Server

В таблице 1.1.3 приведено описание ошибки, найденной при проверке выполнения хранимый процедуры «Создание заказа».

Таблица 1.1.3 – Описание ошибки для ХП «Создание заказа»

Короткое описание	Ошибка условия скалярного выражения
Проект	Тестирование схемы БД интернет-магазина
Номер версии	1.0
Серьезность	Незначительная
Приоритет	Высокий
Автор	-
Назначен на	Щукова Кристина
Окружение	
Конфигурация (ОС, браузер и т.д.)	ОС Windows 8 x64
Шаги воспроизведения	1. Добавление нового клиента с именем Client 1. 2. Размещения для него заказов на сумму 100. 3. Выполнение хранимой процедуры «Создание заказа».
Фактический результат	-100
Ожидаемый результат	100
Дополнения	

Фрагмент ошибки	Фрагмент ошибки при выполнении операции UPDATE в ХП «Создание заказа» (строка 17):
	`UPDATE [Продажи].[Клиент]`
	`SET`
	`ГодовоеКоличествоПродаж = ГодовоеКоличествоПродаж - @Delta`
	`WHERE [IDКлиента] = @CustomerID`
	`COMMIT TRANSACTION`

Предлагаемое решение устранения ошибки	Необходимо изменить знак перед переменной @Delta на «+»:
	`UPDATE [Продажи].[Клиент]`
	`SET`
	`ГодовоеКоличествоПродаж = ГодовоеКоличествоПродаж + @Delta`
	`WHERE [IDКлиента] = @CustomerID`
	`COMMIT TRANSACTION`

Ниже приведены скрипты предусловия и скрипты тестовых случаев.

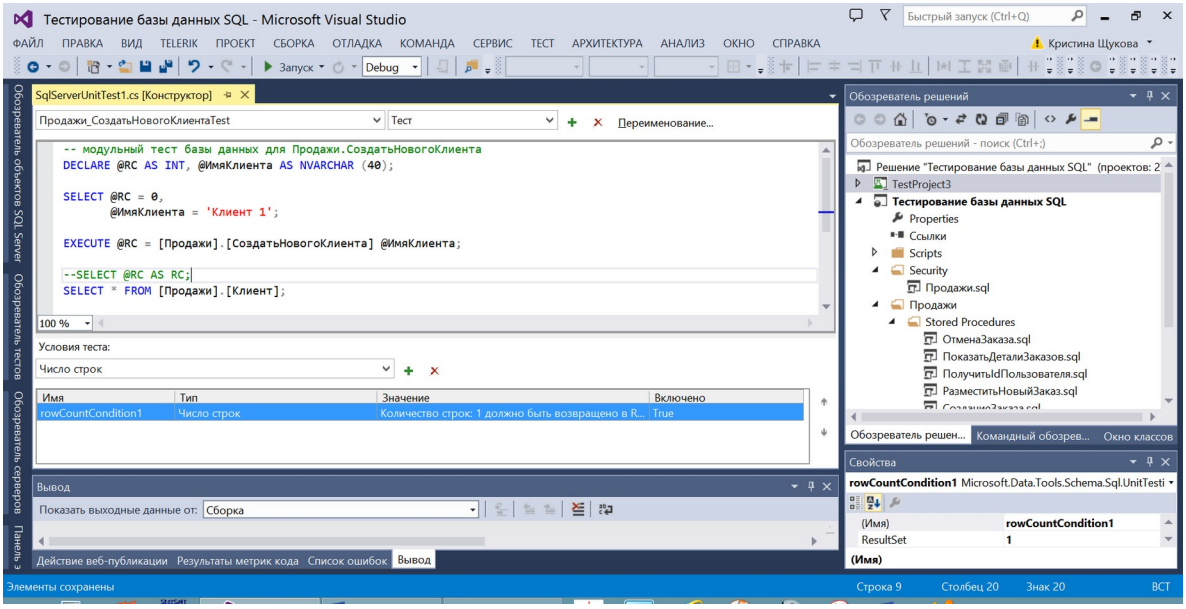

Рисунок 1.1.3 – Тестовый случай для хранимой процедуры «СоздатьНовогоКлиента»

В данном случае в качестве условия теста задано «число строк».

На рисунке 1.1.4 приведено предусловия для проверки выполнения ХП «Разместить новый заказ».

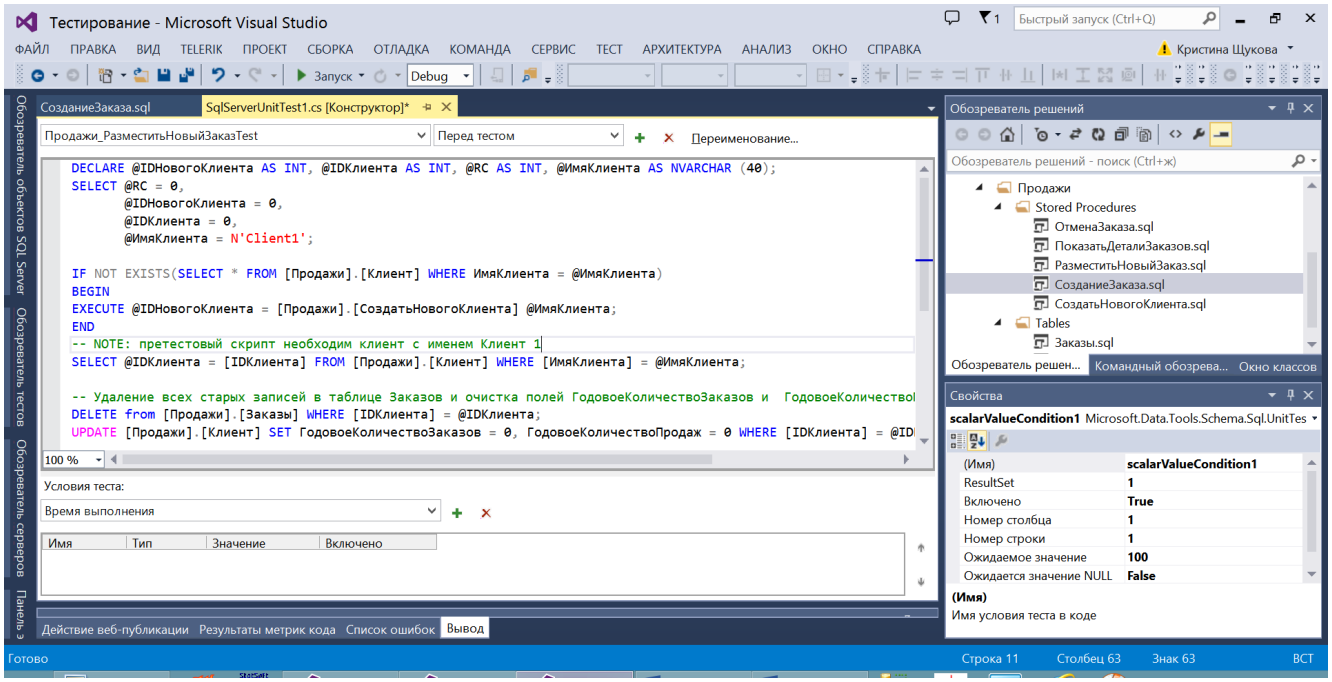

Рисунок 1.1.4 – Предусловие для хранимой процедуры «Разместить новый заказ»

На рисунке 1.1.5 приведен тестовый случай для проверки выполнения ХП «Разместить новый заказ».

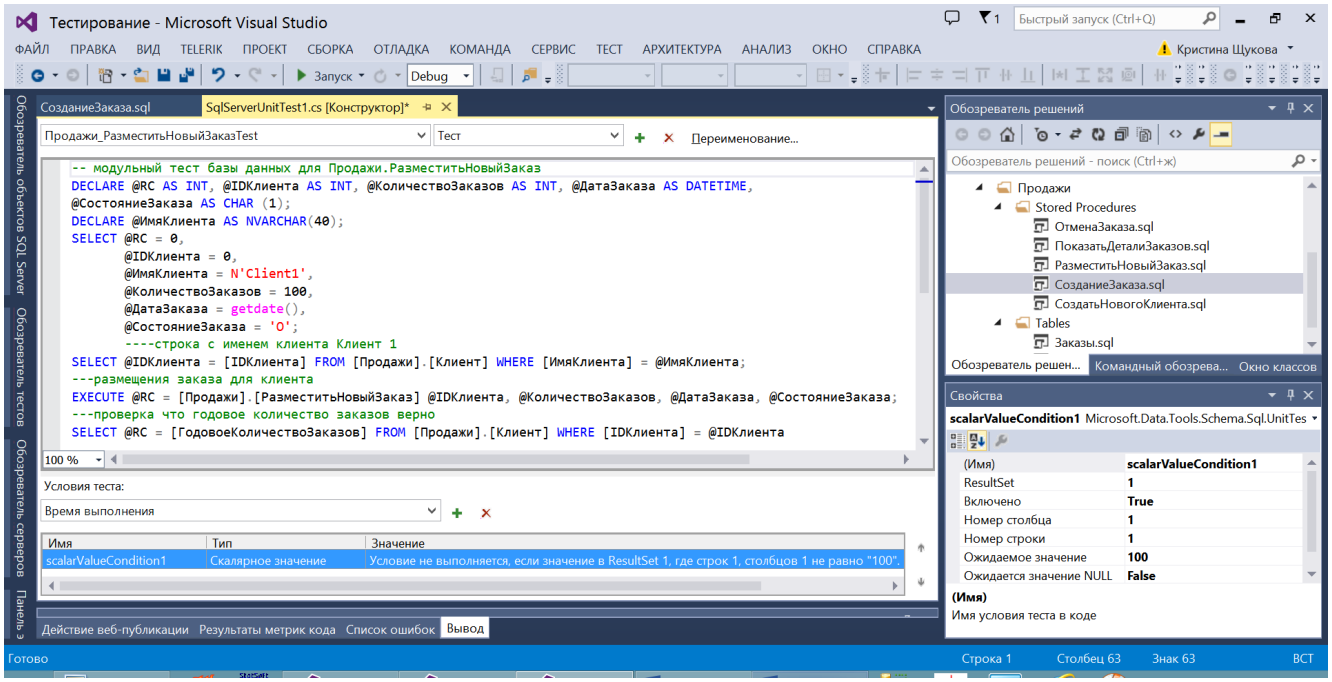

Рисунок 1.1.5 – Тестовый случай для хранимой процедуры «Разместить новый заказ»

На рисунке 1.1.6 приведено предусловие для проверки выполнения ХП «Создание заказа».

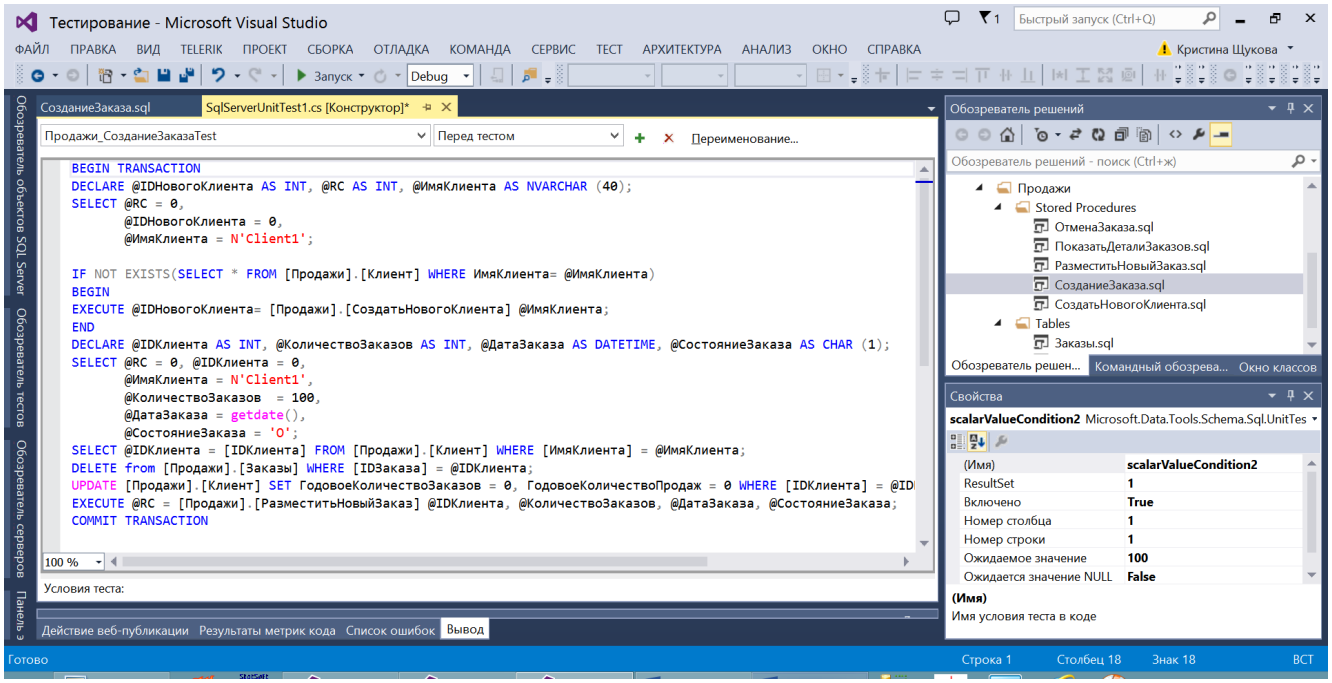

Рисунок 1.1.6 – Предусловие для хранимой процедуры «Создание заказа»

На рисунке 1.1.7 приведен тестовый случай для проверки выполнения ХП «Создание заказа».

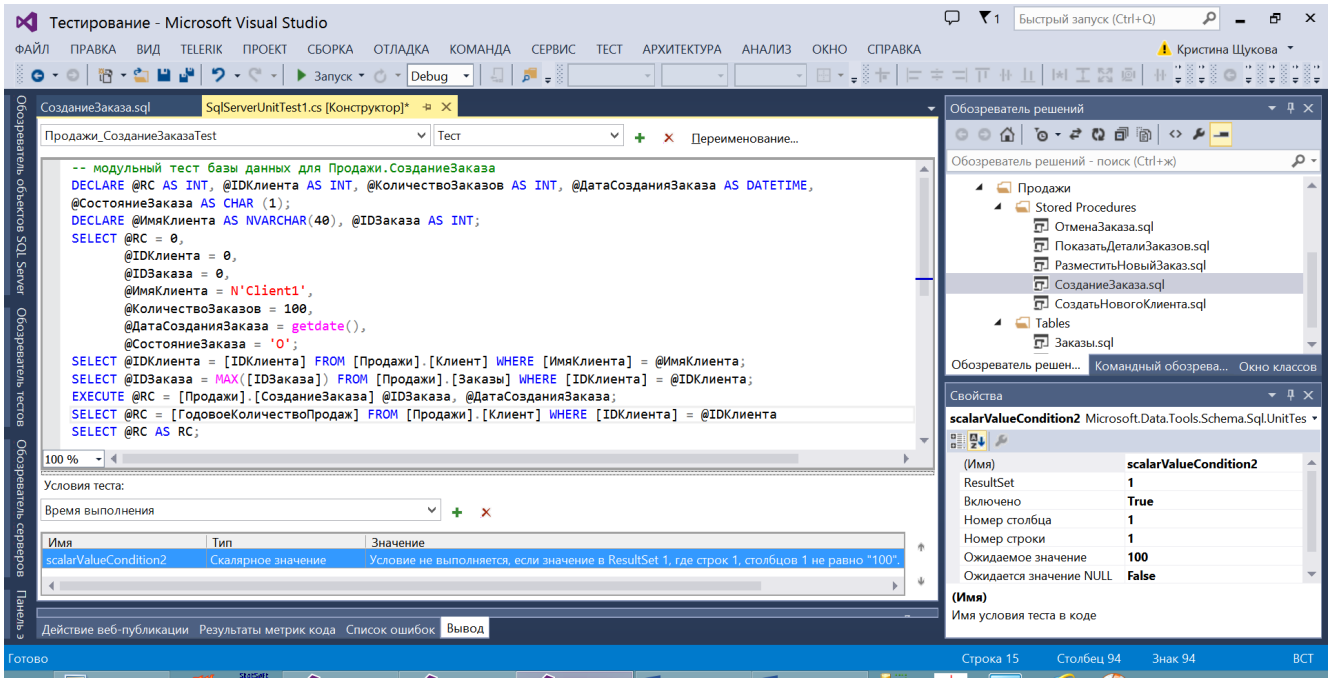

Рисунок 1.1.7 – Тестовый случай для хранимой процедуры «Создание заказа»

На рисунке 1.1.8 приведено предусловие для проверки выполнения ХП «Показать детали заказа».

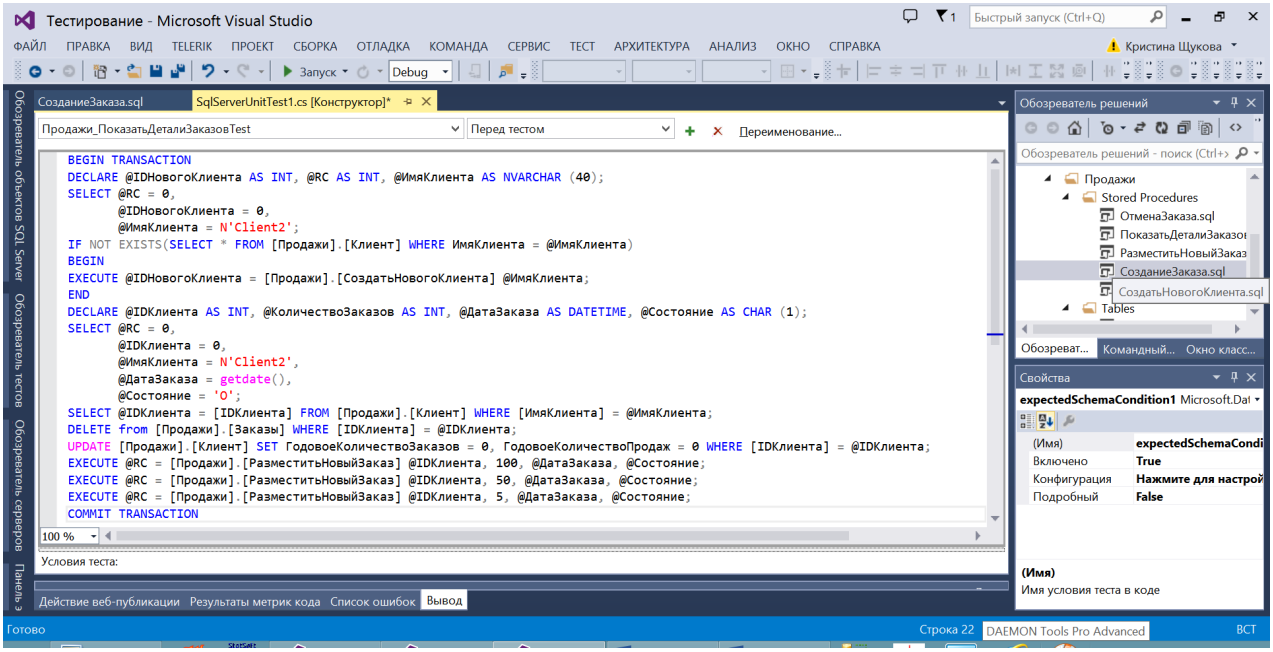

Рисунок 1.1.8 – Предусловие для хранимой процедуры «Показать детали заказа»

На рисунке 1.1.9 приведен тест для проверки выполнения ХП «Показать детали заказа». В качестве условия теста были использованы «Контрольная сумма» и «Ожидаемая сумма».

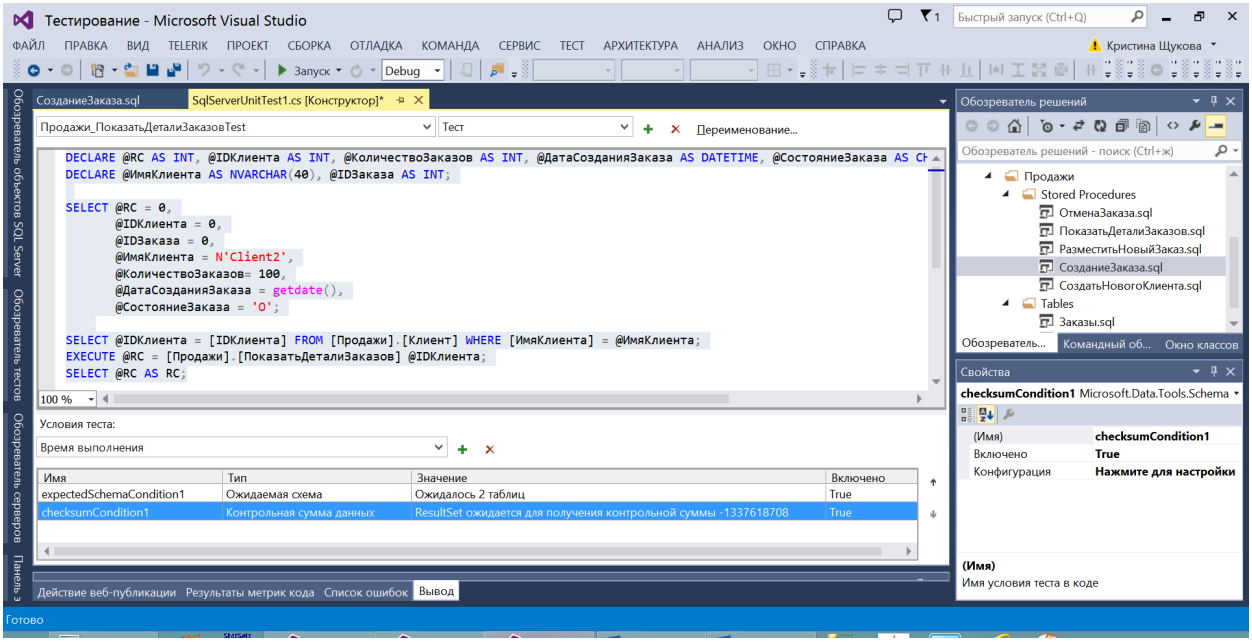

Рисунок 1.1.9 – Тестовый случай для хранимой процедуры «Показать детали заказа»

Для определения контрольной суммы настроена конфигурация контрольной суммы с помощью скрипта, представленного на рисунке 1.1.10.

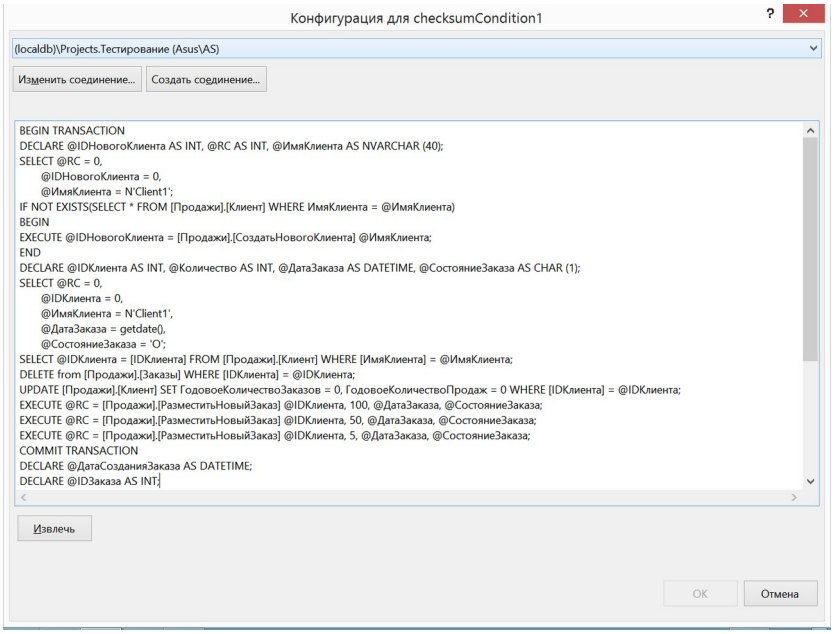

Рисунок 1.1.10 – Настройка конфигурации контрольной суммы

Для проверки ХП «Отмена заказа» необходимо обновить процедуру.

На рисунке 1.1.11. представлен скрипт первоначальной ХП «Отмена заказа».

```
ОтменаЗаказа.sql | СозданиеЗаказа.sql | SqlServerUnitTest1.cs [Конструктор]

CREATE PROCEDURE [Продажи].[ОтменаЗаказа]
@OrderID INT
AS
BEGIN
DECLARE @Delta INT, @CustomerID INT
BEGIN TRANSACTION
    SELECT @Delta = [КоличествоЗаказов], @CustomerID = [IDКлиента]
     FROM [Продажи].[Заказы] WHERE [IDЗаказа] = @OrderID;

UPDATE [Продажи].[Заказы]
   SET [СостояниеЗаказа] = 'X' --- заказ отменен
WHERE [IDЗаказа] = @OrderID;

UPDATE [Продажи].[Клиент]
   SET
   ГодовоеКоличествоЗаказов = ГодовоеКоличествоЗаказов - @Delta
    WHERE [IDКлиента] = @CustomerID
COMMIT TRANSACTION
END
```

Рисунок 1.1.11 – Первоначальный вид ХП «Отмена заказа»

После обновления ХП внесены изменения в рамках предусловия, которое представлено на рисунке 1.1.12.

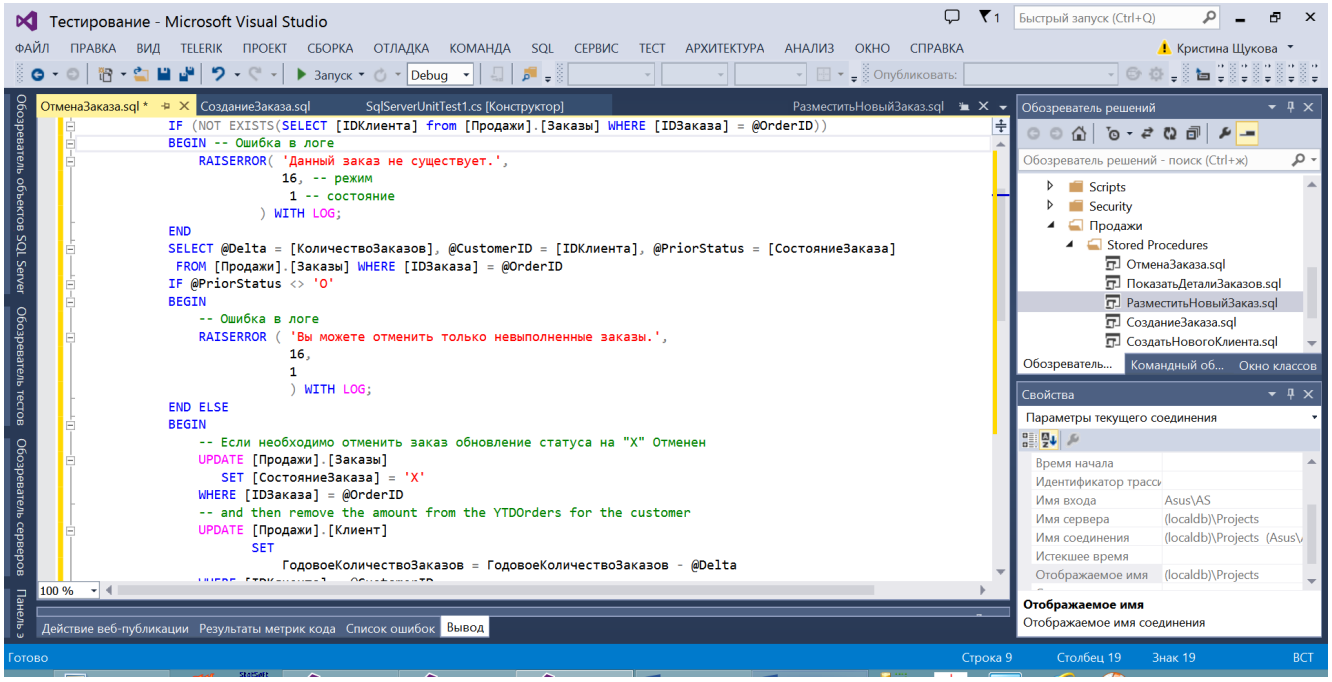

Рисунок 1.1.12 – Обновление ХП «Отмена заказа»

На рисунке 1.1.13 приведено предусловие для проверки выполнения ХП «Отмена заказа».

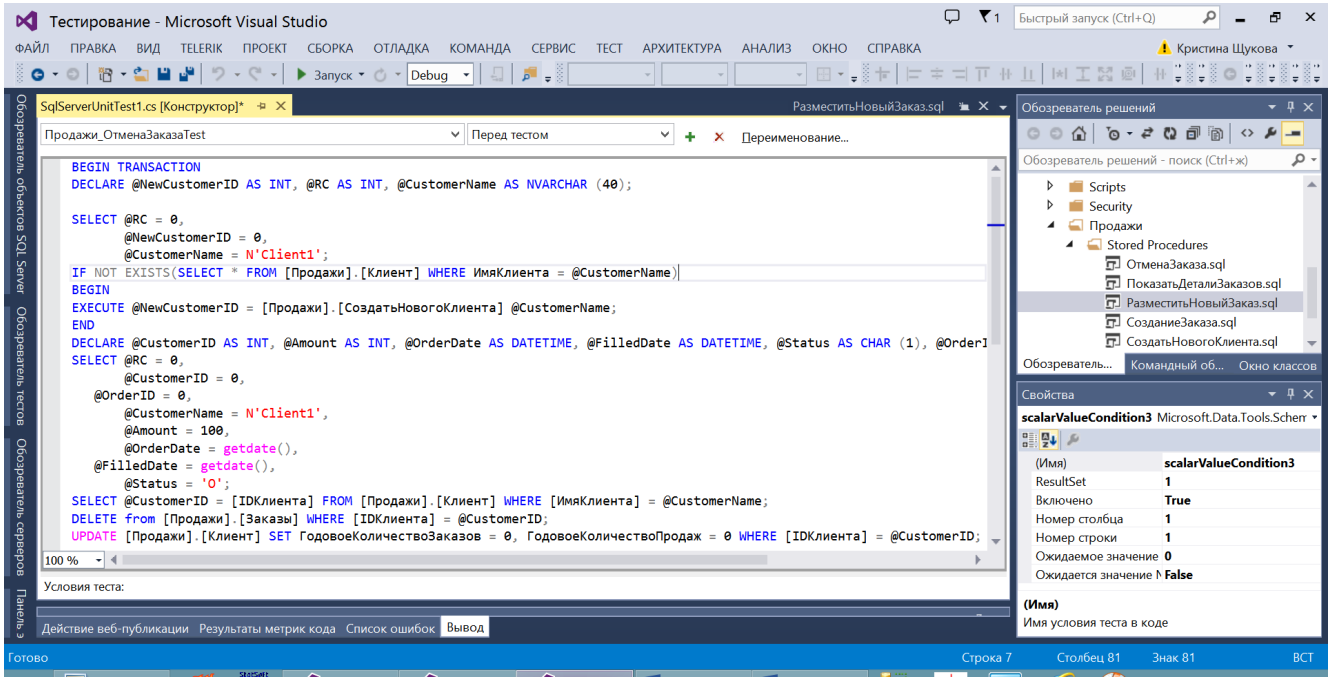

Рисунок 1.1.13 – Предусловие для проверки ХП «Отмена заказа»

На рисунке 1.1.14 приведен тестовый случай для проверки ХП «Отмена заказа».

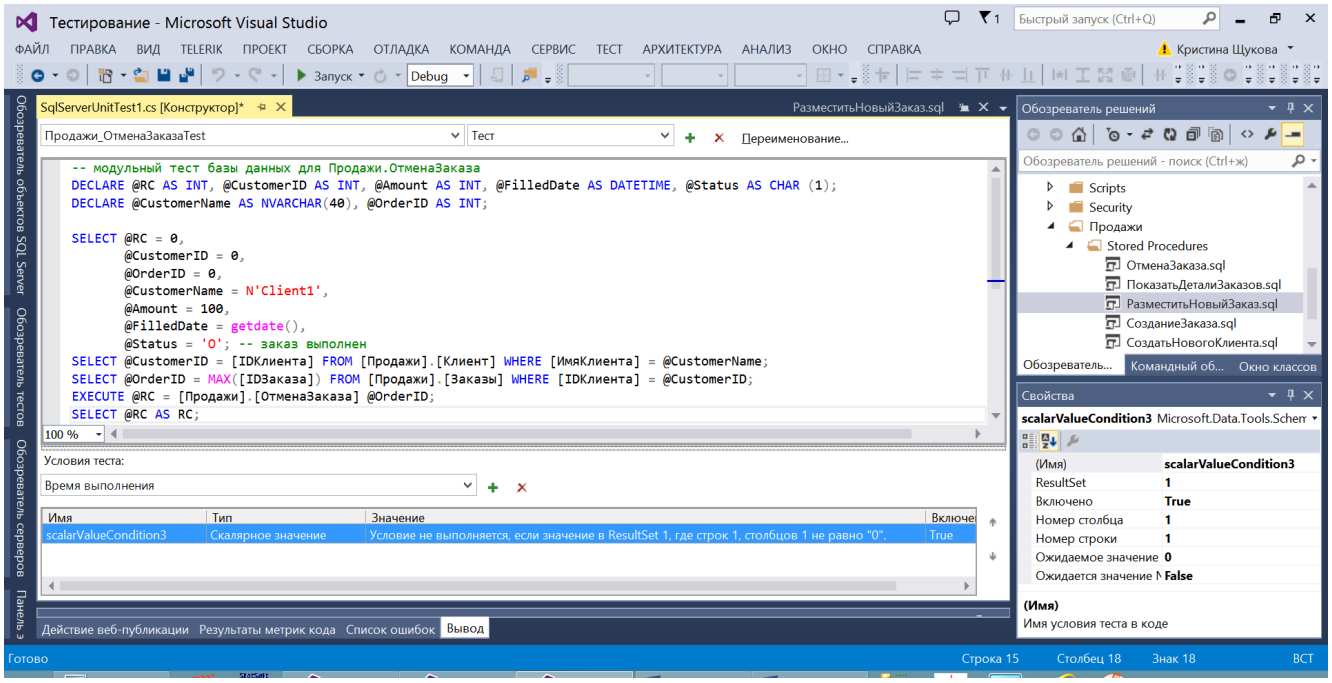

Рисунок 1.1.14 – Тестовый случай для проверки выполнения ХП «Отмена заказа»

На рисунке 1.1.15 представлены результаты выполнения всех тестовых случаев, которые подробно описаны в таблице 1.1.2.

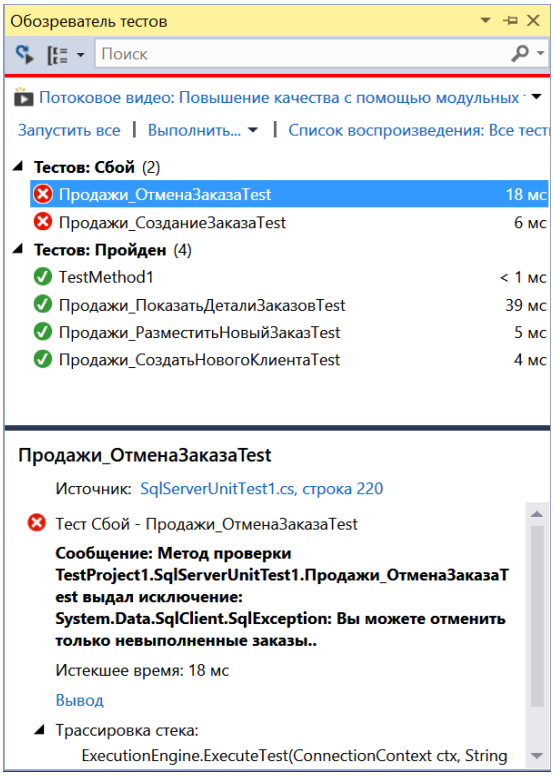

Рисунок 1.1.15 – Результаты выполнения тестовых скриптов SQL

1.2. Тестирование приложения типа Windows Form «Калькулятор»

Для проведения модульного тестирования взят проект «Калькулятор», созданный в Visual Studio 2013 на языке программирования C#. Проект разбит на две части Интерфейс и Логику. Логика приложения – это классы и функции, реализованные в приложении, которые хранятся в одноименной DLL-библиотеке.

Содержимое DLL-библиотеки «Логика»:

```
public class Var
{
    public static double result = 0;
    public static char op = Convert.ToChar(0);
}

public class Constants
{
    public static string Pi = Convert.ToString(Math.PI);
}

public class Operations
{
    public static string neg(string answ)
    {
```

```
            return Convert.ToString(-Convert.ToDouble(answ));
        }
        public static string sin(string answ)
        {
            return Convert.ToString(Math.Round(Math.Sin(Convert.ToDouble(answ) *
Math.PI / 180), 10));
        }
        public static string cos(string answ)
        {
            return Convert.ToString(Math.Round(Math.Cos(Convert.ToDouble(answ) *
Math.PI / 180), 10));
        }
        public static string tg(string answ)
        {
            if (!double.IsInfinity(Math.Round(Math.Sin(Convert.ToDouble(answ) * Math.PI
/ 180), 10) / Math.Round(Math.Cos(Convert.ToDouble(answ) * Math.PI / 180), 10)))
            {
                return Convert.ToString(Math.Round(Math.Sin(Convert.ToDouble(answ) *
Math.PI / 180), 10) / Math.Round(Math.Cos(Convert.ToDouble(answ) * Math.PI / 180),
10));
            }
            else
            {
                return null;
            }
        }
        public static string ctg (string answ)
        {
            if (!double.IsInfinity(1 / (Math.Round(Math.Sin(Convert.ToDouble(answ) *
Math.PI / 180), 10) / Math.Round(Math.Cos(Convert.ToDouble(answ) * Math.PI / 180),
10))))
            {
                return Convert.ToString(1 / (Math.Round(Math.Sin(Convert.ToDouble(answ)
* Math.PI / 180), 10) / Math.Round(Math.Cos(Convert.ToDouble(answ) * Math.PI / 180),
10)));
            }
            else
            {
                return null;
            }

        }
        public static string sqr(string answ)
        {
            return Convert.ToString(Math.Pow((Convert.ToDouble(answ)),2));
        }
        public static string sqrt(string answ)
        {
            return Convert.ToString(Math.Sqrt(Convert.ToDouble(answ)));
        }
        public static string ret(string answ)
        {
            if ((Convert.ToDouble(answ)!=0))
```

```
            {
                return Convert.ToString(1/(Convert.ToDouble(answ)));
            }
            else
            {
                return null;
            }
        }

    }
```

Таблица 1.2.1 – Описания классов и функций

Название класса/функции	**Назначение**
1. Var	Класс необходимый для вывода и подсчета результатов выполняемых вычислений в калькуляторе.
2. Constants	Класс, в котором хранятся используемые в приложении константы.
3. Operations	Класс, содержащий функции, реализованные в приложении.
4. neg	Функция изменяющая знак на противоположный.
5. sin	Функция возвращающая синус введенного угла (в градусах).
6. cos	Функция возвращающая косинус введенного угла (в градусах).
7. tg	Функция возвращающая тангенс введенного угла (в градусах).
8. ctg	Функция возвращающая котангенс введенного угла (в градусах).
9. sqr	Функция возвращающая квадрат введенного числа.
10. sqrt	Функция возвращающая корень введенного неотрицательного числа.
11. ret	Функция возводящая введенное число в степень -1.

Для проведения модульного тестирования создано 15 тестов, проверяющих правильную работу функций:

```
        [TestMethod]
        public void Sum_2Plus5_7Returned()
        {
            Var.op='+';
            Var.result = Convert.ToDouble(2);
            Var.result += Convert.ToDouble(5);
            Assert.AreEqual(7, Var.result);
        }

        [TestMethod]
        public void Difference_5Minus3_2Returned()
        {
            Var.op = '-';
            Var.result = Convert.ToDouble(5);
            Var.result -= Convert.ToDouble(3);
            Assert.AreEqual(2, Var.result);
        }

        [TestMethod]
        public void Multiplication_3Multiply6_18Returned()
        {
```

```
    Var.op = '*';
    Var.result = Convert.ToDouble(3);
    Var.result *= Convert.ToDouble(6);
    Assert.AreEqual(18, Var.result);
}

[TestMethod]
public void Division_9Split3_3Returned()
{
    Var.op = '/';
    Var.result = Convert.ToDouble(9);
    Var.result /= Convert.ToDouble(3);
    Assert.AreEqual(3, Var.result);
}

[TestMethod]
public void Negative_5Neg_n5Returned()
{
    string str = Operations.neg("5");
    Assert.AreEqual("-5", str);
}

[TestMethod]
public void Sin_90Sin_1Returned()
{
    string str = Operations.sin("90");
    Assert.AreEqual("1", str);
}

[TestMethod]
public void Cos_0Cos_1Returned()
{
    string str = Operations.cos("0");
    Assert.AreEqual("1", str);
}

[TestMethod]
public void Tg_0Tg_0Returned()
{
    string str = Operations.tg("0");
    Assert.AreEqual("0", str);
}

[TestMethod]
public void Tg_90Tg_NullReturned()
{
    string str = Operations.tg("90");
    Assert.AreEqual(null, str);
}

[TestMethod]
public void Ctg_45Ctg_1Returned()
{
```

```
        string str = Operations.ctg("45");
        Assert.AreEqual("1", str);
    }

    [TestMethod]
    public void Ctg_0Ctg_NullReturned()
    {
        string str = Operations.ctg("0");
        Assert.AreEqual(null, str);
    }

    [TestMethod]
    public void Square_4Sqr_16Returned()
    {
        string str = Operations.sqr("4");
        Assert.AreEqual("16", str);
    }

    [TestMethod]
    public void SquareRoot_4Sqrt_2Returned()
    {
        string str = Operations.sqrt("4");
        Assert.AreEqual("2", str);
    }

    [TestMethod]
    public void SquareRoot_n4Sqrt_NanReturned()
    {
        string str = Operations.sqrt("-4");
        Assert.AreEqual("NaN", str);
    }

    [TestMethod]
    public void Reverse_4Ret_025Returned()
    {
        string str = Operations.ret("4");
        Assert.AreEqual("0,25", str);
    }
```

Результаты тестирования сведены в таблицу 1.2.2.

Таблица 1.2.2 – Описание модульных тестов

Предусловие	**Тестовый сценарий (шаги)**	**Ожидаемый результат**	**Фактический результат**	**Послесловие**
Инициализация переменных значениями 2 и 5.	Выполнить метод сложения двух чисел.	Результат равен 7.	Тест пройден.	
Инициализация переменных значениями 5 и 3.	Выполнить метод нахождения разности двух чисел.	Результат равен 2.	Тест пройден.	
Инициализация переменных значениями 3 и 6.	Выполнить метод умножения двух чисел.	Результат равен 18.	Тест пройден.	
Инициализация	Выполнить метод деления	Результат равен 3.	Тест пройден.	

переменных значениями 9 на 3.	чисел.			
Инициализация переменной значением 5.	Изменить знак числа на отрицательный.	Результат равен -5.	Тест пройден.	
Инициализация переменной значением 90 градусов.	Выполнить метод расчёта синуса угла.	Результат равен 1.	Тест пройден.	
Инициализация переменной значением 0 градусов.	Выполнить метод расчёта косинуса угла.	Результат равен 0.	Тест пройден.	
Инициализация переменной значением 0 градусов.	Выполнить метод расчёта тангенса 0 градусов.	Результат равен 0.	Тест пройден.	
Инициализация переменной значением 90 градусов.	Выполнить метод расчёта тангенса 90 градусов. Проверить, что он не существует.	Результат равен null.	Тест не пройден.	
Инициализация переменной значением 45 градусов.	Выполнить метод расчёта котангенса 45 градусов.	Результат равен 1.	Тест пройден.	
Инициализация переменной значением 0 градусов.	Выполнить метод расчёта котангенса 0 градусов. Проверить, что он не существует.	Результат равен null.	Тест пройден.	
Инициализация переменной значением 4.	Выполнить метода извлечения квадрата числа.	Результат равен 16.	Тест пройден.	
Инициализация переменной значением 4.	Выполнить метод нахождения квадратного корня из числа.	Результат равен 2.	Тест пройден.	
Инициализация переменной значением -4.	Выполнить метод извлечения квадратного корня из числа -4. Проверить, что он не существует.	Результат равен NaN.	Тест пройден.	
Инициализация переменной значением 4 и -1.	Выполнить метод возведения числа в степень.	Результат равен 0,25.	Тест пройден.	

Найдена ошибка при тестировании метода расчета тангенса от 90 градусов. Описание ошибки приведено в таблице 1.2.3.

Таблица 1.2.3 – Описание ошибки при тестировании метода расчета тангенса угла

Короткое описание	Ошибка расчета метода тангенса угла
Проект	Калькулятор
Номер версии	1.0
Серьезность	Незначительная
Приоритет	Средний

Автор	-
Назначен на	Щукова Кристина
Окружение	
Конфигурация (ОС, браузер и т.д.)	ОС Windows 8 x64
Шаги воспроизведения	1. Инициализация переменной угла значением 90 градусов. 2. Выполнение метода расчета тангенса угла.
Фактический результат	1,63317787283838E+16
Ожидаемый результат	Null
Дополнения	

Фрагмент ошибки	Фрагмент ошибки при выполнении метода расчета тангенса угла:
	`public static string tg(string answ)`
	`{`
	`  if (!double.IsInfinity(Math.Tan(Convert.ToDouble(answ) * Math.PI / 180) ))`
	`  {`
	`  return Convert.ToString(Math.Tan(Convert.ToDouble(answ) * Math.PI / 180));`
	`  }`
	`  else`
	`  {`
	`  return null;`
	`  }`
	`}`

Предлагаемое решение устранения ошибки	Необходимо внести следующие изменения в метод:
	`public static string tg(string answ)`
	`{`
	`  if (!double.IsInfinity(Math.Round(Math.Sin(Convert.ToDouble(answ) * Math.PI / 180), 10) / Math.Round(Math.Cos(Convert.ToDouble(answ) * Math.PI / 180), 10)))`
	`  {`
	`  return Convert.ToString(Math.Round(Math.Sin(Convert.ToDouble(answ) * Math.PI / 180), 10) / Math.Round(Math.Cos(Convert.ToDouble(answ) * Math.PI / 180), 10));`
	`  }`
	`  else`
	`  {`
	`  return null;`
	`  }`
	`}`

1.3. Тестирование интерфейса приложения типа Windows Form «Калькулятор»

Выполнено тестирование интерфейса проекта «Калькулятор». Интерфейс представлен на рисунке 1.3.1.

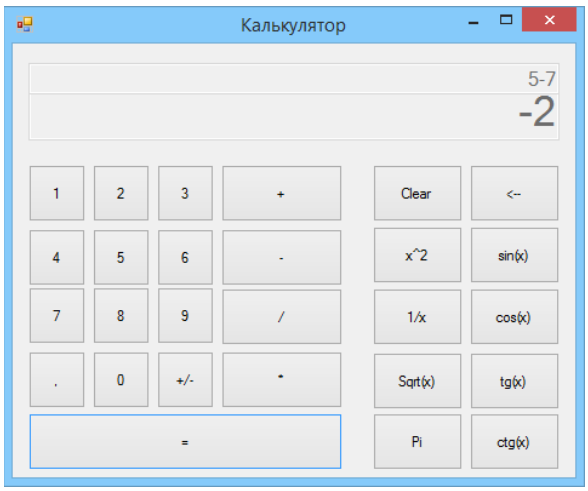

Рисунок 1.3.1 – Интерфейс приложения «Калькулятор»

Для тестирования были созданы 5 методов, содержащие записанные шаги, и 5 методов проверки выполнения методов.

```
public void Plus()
        {
            #region Variable Declarations
            WinButton uIItem1Button = 
this.UIКалькулятopWindow.UIItem1Window.UIItem1Button;
            WinButton uIItemButton = 
this.UIКалькулятopWindow.UIItemWindow.UIItemButton;
            WinButton uIItem3Button = 
this.UIКалькулятopWindow.UIItem3Window.UIItem3Button;
            WinButton uIItemButton1 = 
this.UIКалькулятopWindow.UIItemWindow1.UIItemButton;
            #endregion
            // Запуск "%USERPROFILE%\Desktop\Калькулятор.exe"
            ApplicationUnderTest uIКалькулятopWindow = 
ApplicationUnderTest.Launch(this.PlusParams.UIКалькулятopWindowExePath, 
this.PlusParams.UIКалькулятopWindowAlternateExePath);
            // Щелкните "1" кнопка
            Mouse.Click(uIItem1Button, new Point(25, 31));
            // Щелкните "+" кнопка
            Mouse.Click(uIItemButton, new Point(35, 28));
            // Щелкните "3" кнопка
            Mouse.Click(uIItem3Button, new Point(17, 23));
            // Щелкните "=" кнопка
            Mouse.Click(uIItemButton1, new Point(142, 17));
        }
public void AssertMethodPlus()
        {
            #region Variable Declarations
            WinEdit uIInputBoxEdit = 
this.UIКалькулятopWindow.UIInputBoxWindow.UIInputBoxEdit;
            #endregion
            // Убедитесь, что свойство Text "inputBox" надпись равняется "4"
```

```
            Assert.AreEqual(this.AssertMethodPlusExpectedValues.UIInputBoxEditText,
uIInputBoxEdit.Text, "Not equal!");
        }
public void SqrtNaN()
        {
            #region Variable Declarations
            WinButton uIClearButton =
this.UIКалькулятopWindow.UIClearWindow.UIClearButton;
            WinButton uIItem4Button =
this.UIКалькулятopWindow.UIItem4Window.UIItem4Button;
            WinButton uIItemButton =
this.UIКалькулятopWindow.UIItemWindow2.UIItemButton;
            WinButton uISqrtxButton =
this.UIКалькулятopWindow.UISqrtxWindow.UISqrtxButton;
            #endregion
            // Запуск "%USERPROFILE%\Desktop\Калькулятор.exe"
            ApplicationUnderTest uIКалькулятopWindow =
ApplicationUnderTest.Launch(this.PlusParams.UIКалькулятopWindowExePath,
this.PlusParams.UIКалькулятopWindowAlternateExePath);
            // Щелкните "4" кнопка
            Mouse.Click(uIItem4Button, new Point(18, 25));
            // Щелкните "+/-" кнопка
            Mouse.Click(uIItemButton, new Point(25, 18));
            // Щелкните "Sqrt(x)" кнопка
            Mouse.Click(uISqrtxButton, new Point(42, 25));
        }
public void AssertMethodSqrtNaN()
        {
            #region Variable Declarations
            WinEdit uIInputBoxEdit =
this.UIКалькулятopWindow.UIInputBoxWindow.UIInputBoxEdit;
            #endregion
            // Убедитесь, что свойство Text "inputBox" надпись равняется "NaN"
            Assert.AreEqual(this.AssertMethodSqrtNaNExpectedValues.UIInputBoxEditText,
uIInputBoxEdit.Text, "Error!");
        }
public void Tg90()
        {
            #region Variable Declarations
            WinButton uIClearButton =
this.UIКалькулятopWindow.UIClearWindow.UIClearButton;
            WinButton uIItem9Button =
this.UIКалькулятopWindow.UIItem9Window.UIItem9Button;
            WinButton uIItem0Button =
this.UIКалькулятopWindow.UIItem0Window.UIItem0Button;
            WinButton uITgxButton = this.UIКалькулятopWindow.UITgxWindow.UITgxButton;
            #endregion
            // Запуск "%USERPROFILE%\Desktop\Калькулятор.exe"
            ApplicationUnderTest uIКалькулятopWindow =
ApplicationUnderTest.Launch(this.PlusParams.UIКалькулятopWindowExePath,
this.PlusParams.UIКалькулятopWindowAlternateExePath);
            // Щелкните "9" кнопка
            Mouse.Click(uIItem9Button, new Point(21, 9));
```

```
            // Щелкните "0" кнопка
            Mouse.Click(uIItem0Button, new Point(29, 18));
            // Щелкните "tg(x)" кнопка
            Mouse.Click(uITgxButton, new Point(24, 26));
        }
public void AssertMethodTg90()
        {
            #region Variable Declarations
            WinText uIНедопустимоезначениеText =
this.UIНедопустимоезначениеWindow.UIНедопустимоезначениеText;
            #endregion
            // Убедитесь, что свойство DisplayText "Недопустимое значение аргумента!"
надпись равняется "Недопустимое значение аргумента!"
Assert.AreEqual(this.AssertMethodTg90ExpectedValues.UIНедопустимоезначениеTextDisplayTe
xt, uIНедопустимоезначениеText.DisplayText, "Error!");
        }
public void Ctg0()
        {
            #region Variable Declarations
            WinButton uIOKButton = this.UIOKWindow.UIOKButton;
            WinButton uIClearButton =
this.UIКалькулятopWindow.UIClearWindow.UIClearButton;
            WinButton uIItem0Button =
this.UIКалькулятopWindow.UIItem0Window.UIItem0Button;
            WinButton uICtgxButton =
this.UIКалькулятopWindow.UICtgxWindow.UICtgxButton;
            #endregion
            // Запуск "%USERPROFILE%\Desktop\Калькулятор.exe"
            ApplicationUnderTest uIКалькулятopWindow =
ApplicationUnderTest.Launch(this.PlusParams.UIКалькулятopWindowExePath,
this.PlusParams.UIКалькулятopWindowAlternateExePath);
            // Щелкните "0" кнопка
            Mouse.Click(uIItem0Button, new Point(18, 24));
            // Щелкните "ctg(x)" кнопка
            Mouse.Click(uICtgxButton, new Point(50, 23));
        }
public void AssertMethodCtg0()
        {
            #region Variable Declarations
            WinText uIНедопустимоезначениеText =
this.UIНедопустимоезначениеWindow.UIНедопустимоезначениеText;
            #endregion
            // Убедитесь, что свойство DisplayText "Недопустимое значение аргумента!"
надпись равняется "Недопустимое значение аргумента!"
Assert.AreEqual(this.AssertMethodCtg0ExpectedValues.UIНедопустимоезначениеTextDisplayTe
xt, uIНедопустимоезначениеText.DisplayText, "Error!");
        }
public void Div0()
        {
            #region Variable Declarations
            WinButton uIOKButton = this.UIOKWindow.UIOKButton;
            WinButton uIClearButton =
this.UIКалькулятopWindow.UIClearWindow.UIClearButton;
```

```
            WinButton uIItem1Button =
this.UIКалькулятopWindow.UIItem1Window.UIItem1Button;
            WinButton uIItemButton =
this.UIКалькулятopWindow.UIItemWindow3.UIItemButton;
            WinButton uIItem0Button =
this.UIКалькулятopWindow.UIItem0Window.UIItem0Button;
            WinButton uIItemButton1 =
this.UIКалькулятopWindow.UIItemWindow1.UIItemButton;
            #endregion
            // Запуск "%USERPROFILE%\Desktop\Калькулятор.exe"
            ApplicationUnderTest uIКалькулятopWindow =
ApplicationUnderTest.Launch(this.PlusParams.UIКалькулятopWindowExePath,
this.PlusParams.UIКалькулятopWindowAlternateExePath);
            // Щелкните "1" кнопка
            Mouse.Click(uIItem1Button, new Point(18, 29));
            // Щелкните "/" кнопка
            Mouse.Click(uIItemButton, new Point(81, 22));
            // Щелкните "0" кнопка
            Mouse.Click(uIItem0Button, new Point(18, 27));
            // Щелкните "=" кнопка
            Mouse.Click(uIItemButton1, new Point(114, 23));
        }
public void AssertMethodDiv0()
        {
            #region Variable Declarations
            WinText uIДелениенанольневозмоText =
this.UIДелениенанольневозмоWindow.UIДелениенанольневозмоText;
            #endregion
            // Убедитесь, что свойство DisplayText "Деление на ноль невозможно!"
надпись равняется "Деление на ноль невозможно!"
Assert.AreEqual(this.AssertMethodDiv0ExpectedValues.UIДелениенанольневозмоTextDisplayTe
xt, uIДелениенанольневозмоText.DisplayText, "Error!");
        }
```

Описание методов тестирования интерфейса приведено в таблице 1.3.1.

Таблица 1.3.1 – Описания методов

Название методов	**Назначение**
1. Plus	Метод содержащий шаги для выполнения сложения двух чисел 1 и 3.
2. AssertMethodPlus	Метод проверяющий равен ли результат сложения 4.
3. SqrtNaN	Метод содержащий шаги для взятия квадратного корня из отрицательного числа -4.
4. AssertMethodSqrtNaN	Метод проверяющий был ли результат равен NaN.
5. Tg90	Метод содержащий шаги для расчета тангенса 90 градусов (не существует).
6. AssertMethodTg90	Метод проверяющий была ли выдана ошибка о недопустимых аргументах.
7. Ctg0	Метод содержащий шаги для расчета котангенса 0 градусов (не существует).
8. AssertMethodCtg0	Метод проверяющий была ли выдана ошибка о недопустимых аргументах.

9. Div0	Метод содержащий шаги для выполнения деления числа 1 на 0 (не существует).
10. AssertMethodDiv0	Метод проверяющий была ли выдана ошибка о делении на 0.

Для тестирования были созданы 5 закодированных теста пользовательского интерфейса.

```
[TestMethod]
public void CodedUITestMethodPlus()
{
    this.UIMap.Plus();
    this.UIMap.AssertMethodPlus();
}
[TestMethod]
public void CodedUITestMethodSqrtNaN()
{
    this.UIMap.SqrtNaN();
    this.UIMap.AssertMethodSqrtNaN();
}
[TestMethod]
public void CodedUITestMethodTg90()
{
    this.UIMap.Tg90();
    this.UIMap.AssertMethodTg90();
}
[TestMethod]
public void CodedUITestMethodCtg0()
{
    this.UIMap.Ctg0();
    this.UIMap.AssertMethodCtg0();
}
[TestMethod]
public void CodedUITestMethodDiv0()
{
    this.UIMap.Div0();
    this.UIMap.AssertMethodDiv0();
}
```

Результаты тестирования сведены в таблицу 1.3.2.

Таблица 1.3.2 – Описание закодированных тестов

Предусловие	**Тестовый сценарий (шаги)**	**Ожидаемый результат**	**Фактический результат**	**Послесловие**
Инициализация переменных значениями 1 и 3.	Выполнить метод сложения двух чисел.	Результат равен 4.	Тест пройден.	
Инициализация переменной значением -4.	Выполнить метод извлечение квадратного корня из -4.	Результат равен NaN.	Тест пройден.	
Инициализация переменной значением 90	Выполнить метод расчёта тангенса угла.	Ошибка о недопустимых аргументах.	Тест пройден.	

градусов.				
Инициализация переменной значением 0 градусов.	Выполнить метод расчёта котангенса угла.	Ошибка о недопустимых аргументах.	Тест пройден.	
Инициализация переменных значениями 0 и 1.	Выполнить метод деления для проверки деление числа 1 на 0.	Ошибка о делении на 0.	Тест пройден.	

ВЫВОДЫ

Проведено модульное тестирование 2 проектов: «База данных интернет-магазина продаж» и «Калькулятор». Для проекта «Калькулятор» протестирован интерфейс пользователя посредством автоматического создания закодированных тестов пользовательского интерфейса и DLL-библиотека методов калькулятора. Для проектов «База данных интернет-магазина продаж» и «Калькулятор» созданы ручные модульные тесты и скрипты предусловия для проверки выполнения тестовых случаев.

В проекте «База данных интернет-магазина продаж» созданы четыре ручные тестовые случаи для проверки выполнения хранимых процедур: «Показать детали заказа», «Разместить новый заказ», «Создание заказа», «Создать нового клиента».

Также для хранимых процедур – «Показать детали заказа», «Разместить новый заказ», «Создание заказа» – созданы ручные скрипты предусловия, которые необходимы для задания начальных условий выполнения тестовых случаев.

В результате тестирования хранимой процедуры «Создать нового клиента» проверено, что выполняется создание нового клиента в таблицу «Клиент». Для

этого был использован оператор «Число строк», который возвращает одну запись, что указывает на то, что процедура успешно создает новых клиентов.

В результате тестирования хранимой процедуры «Разместить новый заказ» проверено, что выполняется размещение нового заказа в таблицу «Заказы» для добавленного клиента. Создан скрипт предусловия, который создает нового клиента. Также создан модульный тест, проверяющий, что выполняется размещение заказа на сумму 100 руб. для созданного клиента, при этом сумма продаж составляет 0 руб. (так как заказ не выполнен, а только размещен пользователем в интернет-магазине). Для этого использован оператор «Скалярное выражение», в свойствах которого ожидаемое значение составляет 100. В результате выполнения теста фактическое значение совпало с ожидаемым, что указывает на то, что заказ успешно размещен для указанного клиента.

В результате тестирования хранимой процедуры «Создание заказа» проверено, что создается заказ для добавленного клиента с общей годовой суммой заказов и продаж = 100 руб. (это означает, что заказ выполнен, и интернет-магазин осуществил продажу товара клиенту на сумму 100 руб.). Создан скрипт предусловия, добавляющий нового клиента и удаляющий все раннее связанные с ним заказы. Также создан модульный тест, проверяющий, что для добавленного клиента годовая сумма заказа и продаж составляет 100 руб. Для этого использован оператор «Скалярное выражение», в свойствах которого ожидаемое значение составляет 100. В результате выполнения теста фактическое значение не совпала с ожидаемым, так как допущена ошибка в формуле вычисления суммы годового количество продаж. Предложено решение для устранения данной ошибки.

В результате тестирования хранимой процедуры «Показать все детали заказа» проверено, что процедура правильно возвращает все данные заказов путем оценки правильности возвращаемых номеров столбцов атрибутов, и что результирующие данные имеют правильное значение контрольной суммы. Создан скрипт предусловия, добавляющий клиента и размещающий для него заказы на

сумму 100, 50 и 5 руб. Для проверки правильности возвращения ХП номеров столбцов и контрольной суммы использованы операторы «Контрольная сумма» и «Ожидаемая схема БД». В результате выполнения модульного теста ожидаемое число столбцов и значение контрольной суммы совпало с фактическими значениями.

В результате тестирования хранимой процедуры «Отмена заказа» проверено, что тестовый случай завершится с ошибкой в случае попытки отмены заказа со статусом «Выполнен», так как нельзя отменить заказ, который уже выполнен. Для этого создан скрипт предусловия, добавляющий нового клиента со статусом заказа «Выполнен». В результате выполнения модульного теста при попытке отменить выполненный заказ произошла ошибка, которая указывает, на то что хранимая процедура работает правильно.

В результате тестирования калькулятора проверена корректность выполнения следующих методов: сложение двух чисел, нахождение разности двух чисел, умножение чисел, метод деления чисел, изменения знака числа, деление на ноль, расчёт косинуса, синуса, тангенса и котангенса угла, извлечения числа из квадратного корня, возведения числа в квадрат. В модульных тестах созданы предусловия инициализации начальными значениями переменных для проверки правильности выполнения методов.

Для тестирования пользовательского интерфейса калькулятора созданы отдельные закодированные тесты. Среда Visual Studio позволяет автоматизировать этот процесс за счет автоматического генерирования последовательности выполняемых действий в виде закодированных тестов. Протестированы методы валидации данных для проверки запрета ввода недопустимых значений (деление на ноль, значение тангенса и котангенса – 0 и 90 градусов, попытка извлечения квадратного корня из отрицательного числа).