МИНИСТЕРСТВО ОБРАЗОВАНИЯ КИРОВСКОЙ ОБЛАСТИ

Кировское областное государственное профессиональное

образовательное бюджетное учреждение

«Вятско-Полянский механический техникум»

ОТЧЕТ О ВЫПОЛНЕНИИ

ЛАБОРАТОРНОЙ РАБОТЫ № 10 «Разработка тестовых сценариев программного средства, заполнение шаблона тестирования»

по дисциплине «Разработка ПО на базе платформы C#»

Выполнил студент

группы 2ИСП

Бурыкин Д.А.

Проверил преподаватель

Галимова

г. Вятские Поляны

2023 г.

					ПМ.02.УП.02.02.002.09.02.07.00.ОТ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Бурыкин			Учебная практика 02.02	Лит.	Лист	Листов
Провер.		Галимова						
						ВПМТ 2ИС		
Н. Контр.								
Утверд.								

Лабораторная работа №10
«Разработка тестовых сценариев программного средства, заполнение шаблона тестирования»

Отчёт

Вариант студента – 2.

Ссылка на репозиторий - https://github.com/MaddieChaotica/DCMDPRACTICE

Заполнение шаблона тестирования для проекта из практической работы №2.

Общая информация проекта (Таблица 1)

Название проекта	FormulaeCalculi
Номер версии	1.0.0
Имя тестера	DCMD
Даты тестирования	12.09.2023

Таблица 1 – Общая информация о проекте

Информация о конкретном тест кейсе (Таблица 2)

Наименование	**Описание**
Наименование проекта	FormulaeCalculi
Номер версии	1.0.0
Имя тестера	Имя тестера, который выполнял эти тесты
Даты тестирования	12.09.2023
Test Case #	TC_Funct_01
Приоритет тестирования	Highest priority
Название тестирования/Имя	Functionality testing
Резюме испытания	Using the application, we need to reach the correct answer of the given formula without any major errors.
Шаги тестирования	1. Turn the program on 2. Press the button 3. See if the answer is correct
Данные тестирования	Used OS – Windows 10 Application was used inside of Visual Studio interface
Ожидаемый результат	The test's result should be a correct answer to a formula with given values of x, y, and z

Фактический результат	The test has given a pop-up box with the correct values (within the margin of errors and deviation)
Постусловия	Close the program
Статус	Passed
Комментарии	The program works as intended, although the GUI looks a bit ugly and program works only with existing values (as intended by practice work)

Таблица 2 – Тест Кейс №1

Тесты пройдены успешно.

Контрольные вопросы:

1 Что такое unit-тест?

Модульное тестирование, иногда блочное тестирование или юнит-тестирование — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки.

2 Какие существуют типы тестирования?

- Ручные
- Автоматизированные

3 Что такое TDD?

TDD или же Test Driven Development – это вид процесса программной разработки, который основывается на том, что программное обеспечение должно быть полностью сделано в юнит-тестах до того как быть завершенным.

4 В чем заключается разница между валидацией и верификацией?

Валидация подтверждает, что «вы создали правильный продукт», а верификация подтверждает, что «вы создали продукт таким, каким и намеревались его сделать».

5 Как создать unit-тест в C#?

- Добавить «Тестовый Класс» в проект
- Добавить ссылку в «Тестовый Класс» на тот класс, который должен быть проверен юнит-тестом
- Написать тест

6 Как заставить себя писать unit-тесты?

Как заставить себя писать unit-тесты? Данный вопрос не является корректным из-за его субъективной натуры. Каждый человек имеет свои причины или способы чтобы «заставить» себя что-то делать, но есть ряд причин, которые *возможно* помогут вдохновиться писать unit-тесты:

- unit-тесты облегчают нахождение ошибок в больших проектах
- Держат программу более чистой и разделенной на модули
- Облегчают внедрение новых девелоперов
- Облегчают беспокойство, что какая-то часть программы перестанет работать, а деву придаться тратить своё драгоценное время для нахождения маленького бага

7 Перечислите и опишите лучшие практики использования unit-тестов?

- AAA (Arrange, Act, Assert) Паттерн

Если посмотреть на юнит-тест, то для большинства можно четко выделить 3 части кода: Arrange (настройка) - в этом блоке кода мы настраиваем тестовое окружение тестируемого юнита; Act - выполнение или вызов тестируемого сценария; Assert - проверка, что тестируемый вызов ведет себя определенным образом.

Этот паттерн улучшает структуру кода и его читабельность, однако начинать писать тест нужно всегда с элемента Act.

- Data Driven approach

Прежде чем продолжить рассмотрение структуры теста, хотелось бы рассказать немного о подходе, который зовется Driven Approach.

Суть его в том, что код, который вы пишите должен иметь причину своего существования. Важно, что бы причина была существующей, а не предполагаемой, и эта причина должна иметь в конечном итоге связь с бизнесом.

- AAS (Act, Assert, Setup) Паттерн

AAS - тот же AAA паттерн, но с измененным порядком частей, отсортированных с учетом Driven approach и переименованной Arrange частью в Setup, чтобы отличать их по названию.

8 Какие атрибуты используются при написании unit-тестов?

При написании unit-тестов существуют множество различных атрибутов (Рисунок 1-2)

Attribute	Usage
Apartment Attribute	Indicates that the test should run in a particular apartment.
Author Attribute	Provides the name of the test author.
Category Attribute	Specifies one or more categories for the test.
Combinatorial Attribute	Generates test cases for all possible combinations of the values provided.
Culture Attribute	Specifies cultures for which a test or fixture should be run.
Datapoint Attribute	Provides data for Theories.
DatapointSource Attribute	Provides data for Theories.
DefaultFloatingPointTolerance Attribute	Indicates that the test should use the specified tolerance as default for float and double comparisons.
Description Attribute	Applies descriptive text to a Test, TestFixture or Assembly.
Explicit Attribute	Indicates that a test should be skipped unless explicitly run.
FixtureLifeCycle Attribute	Specifies the lifecycle of a fixture allowing a new instance of a test fixture to be constructed for each test case. Useful in situations where test case parallelism is important.
Ignore Attribute	Indicates that a test shouldn't be run for some reason.
LevelOfParallelism Attribute	Specifies the level of parallelism at assembly level.
MaxTime Attribute	Specifies the maximum time in milliseconds for a test case to succeed.
NonParallelizable Attribute	Specifies that the test and its descendants may not be run in parallel.
NonTestAssembly Attribute	Specifies that the assembly references the NUnit framework, but that it does not contain tests.
OneTimeSetUp Attribute	Identifies methods to be called once prior to any child tests.
OneTimeTearDown Attribute	Identifies methods to be called once after all child tests.
Order Attribute	Specifies the order in which decorated test should be run within the containing fixture or suite.
Pairwise Attribute	Generate test cases for all possible pairs of the values provided.
Parallelizable Attribute	Indicates whether test and/or its descendants can be run in parallel.
Platform Attribute	Specifies platforms for which a test or fixture should be run.
Property Attribute	Allows setting named properties on any test case or fixture.
Random Attribute	Specifies generation of random values as arguments to a parameterized test.
Range Attribute	Specifies a range of values as arguments to a parameterized test.
Repeat Attribute	Specifies that the decorated method should be executed multiple

Рисунок 1 – Список атрибутов 1

RequiresThread Attribute	Indicates that a test method, class or assembly should be run on a separate thread.
Retry Attribute	Causes a test to be rerun if it fails, up to a maximum number of times.
Sequential Attribute	Generates test cases using values in the order provided, without additional combinations.
SetCulture Attribute	Sets the current Culture for the duration of a test.
SetUICulture Attribute	Sets the current UI Culture for the duration of a test.
SetUp Attribute	Indicates a method of a TestFixture called just before each test method.
SetUpFixture Attribute	Marks a class with one-time setup or teardown methods for all the test fixtures in a namespace.
SingleThreaded Attribute	Marks a fixture that requires all its tests to run on the same thread.
TearDown Attribute	Indicates a method of a TestFixture called just after each test method.
Test Attribute	Marks a method of a TestFixture that represents a test.
TestCase Attribute	Marks a method with parameters as a test and provides inline arguments.
TestCaseSource Attribute	Marks a method with parameters as a test and provides a source of arguments.
TestFixture Attribute	Marks a class as a test fixture and may provide inline constructor arguments.
TestFixtureSetup Attribute	Deprecated synonym for OneTimeSetUp Attribute.
TestFixtureSource Attribute	Marks a class as a test fixture and provides a source for constructor arguments.
TestFixtureTeardown Attribute	Deprecated synonym for OneTimeTearDown Attribute.
TestOf Attribute	Indicates the name or Type of the class being tested.
Theory Attribute	Marks a test method as a Theory, a special kind of test in NUnit.
Timeout Attribute	Provides a timeout value in milliseconds for test cases.
Values Attribute	Provides a set of inline values for a parameter of a test method.
ValueSource Attribute	Provides a source of values for a parameter of a test method.

Рисунок 2 – Список Атрибутов 2

9 Как используется TestContext в unit-тестах?

TestContext используется для хранения информации для unit-тестов

10 Как используется класс Assert при работе с unit-тестами?

Класс Assert используется в NUnit и содержит методы часто используемые для unit-тестов для их завершения

11 Как используется атрибут DeploymentItem в MSTest?

Класс атрибутов DeploymentItem используется для указания пути к файлам для конкретного unit-тестах

12 Как используется OrderTest и GenericTest в MSTest?

Атрибут OrderTest протезирует тесты с этим атрибутом, в итоге они выполняться первыми GenericTest является атрибутом по умолчанию если другие атрибуты не указаны

13 Как создать Data Driven тест в MSTest?

Достаточно сделать цикл над основным тестом, и организовать сравнение выходных данных и эталонных, а также реализовать репортинг — путём логгирования, или другим способом.