

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение*  *высшего образования*  ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных

технологий (МОСИТ)

ПРАКТИЧЕСКАЯ РАБОТА №2

по дисциплине

**«Тестирование и верификация программного обеспечения»**

Тема: **«Тестирование черного ящика»**

**Студент группы** ИКБО-20-21

Савельев С.А.

(подпись студента)

**Принял руководитель работы** Овчинникова М.А.

(подпись руководителя)

Практические работы выполнены « » 2023 г.

Зачтено « » 2023 г.

Содержание

[1. Цель работы 3](#_Toc146666897)

[1.1 Задание для выполнения 3](#_Toc146666898)

[1.2 Материальная часть 3](#_Toc146666899)

[1.2.1 Модульное тестирование 3](#_Toc146666900)

[1.2.3 Спецификация AAA 3](#_Toc146666901)

[2. Инструменты 3](#_Toc146666902)

[2.1. Используемое ПО 3](#_Toc146666903)

[3. Выполнение задания 4](#_Toc146666904)

[3.1. Составление документации к собственному программному продукту. 4](#_Toc146666905)

[3.2. Документация к программному продукту другого участника группы 4](#_Toc146666906)

[3.3 Тестирование программного продукта другого участника группы 5](#_Toc146666907)

[План тестирования. 5](#_Toc146666908)

[4. Оценка 8](#_Toc146666909)

[5. Выводы 9](#_Toc146666910)

[6. Список использованных источников 10](#_Toc146666911)

# Цель работы

## 1.1 Задание для выполнения

Составление документации к программному продукту, анализ документации другого участница группы, написание модульных тестов в соответствии с спецификацией AAA (Arrange, Act, Assert).

## 1.2 Материальная часть

## 1.2.1 Модульное тестирование

Модульное тестирование -метод тестирования, 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.

## 1.2.3 Спецификация AAA

Спецификация AAA (Arrange, Act, Assert) - (входные данные, действие, ожидаемый результат.

# Инструменты

## Используемое ПО

Для проведения тестирования программного продукта, написанного на языке Python, использовалась библиотека unittest.

# Выполнение задания

## Составление документации к собственному программному продукту.

Программный продукт состоит из двух методов.

**Метод 1 - factorial**.

Функционал: вычисление факториала.

Входные данные:

int n - число;

Ограничения на входные данные: n – целое число.

Выходные данные:

Число типа int;

Возможные варианты выходных данных:

“число” – полученный факториал;

**Метод 2 - fibonacci.**

Функционал: вычисление последовательности Фибоначчи.

Входные данные:

int n – количество членов последовательности;

Ограничения на входные данные: n – целое.

Выходные данные:

Список list;

Возможные варианты выходных данных

“[]” – пустая последовательность;

“[число, …]” - последовательность;

## Документация к программному продукту другого участника группы

Программный продукт состоит из двух методов.

**Метод 1 - chebyshev**.

Функционал: считает полином Чебышева.

Входные данные:

Int n - коэффициент степени полинома;

Int x – неизвестная переменная;

Ограничения на входные данные: отсутствуют

Выходные данные: целое число;

Возможные варианты выходных данных: значение многочлена;

**Метод 2 - markov.**

Функционал: вычисление значения цепи Маркова.

Входные данные:

Int n - коэффициент степени;

Int x - неизвестная переменная;

Ограничения на входные данные: отсутствуют.

Выходные данные: целое число;

Возможные варианты выходных данных: значение многочлена;

# Тестирование программного продукта другого участника группы

## План тестирования.

Выполнение практической работы подразумевает проведение следующих этапов.

* Проведение модульного тестирования
  + **Тестирование метода chebyshev**
    - Тестирование корректной работы при стандартных данных
    - Тестирование корректной работы при крайних случаях
  + **Тестирование метода markov**
    - Тестирование корректной работы при стандартных данных
    - Тестирование корректной работы при крайних случаях

**Создание модульных тестов**

* + - * **Тесты метода chebyshev**

Для тестирования метода chebyshev были проверены четыре различных тестовых случая.

Случай первый с стандартным набором данных при неграничных значениях продемонстрирован на рисунке 1.



Рисунок 1 - Тесты на стандартных значениях

Случай второй с граничными значениями продемонстрирован на рисунке 2.



Рисунок 2 - Тесты на граничных значениях

Случай третий с отрицательным значением степени продемонстрирован представлен на рисунке 3.



Рисунок 3 - Тесты на граничных значениях

Случай четвёртый, направленный на проверку возможного переполнения, представлен на рисунке 4.



Рисунок 4 - Тест на переполнение

* + - * **Тесты метода markov**

Для тестирования метода markov были проверены 8 тестовых случаев.

Первые два случая со стандартными входными данными в различном порядке представлены на рисунке 5.



Рисунок 5 - Тесты метода markov на стандартных значениях

Также тестирование было проведено на граничных и отрицательных значениях (рис. 6-7).



Рисунок 6 - Тесты метода markov на граничных значениях



Рисунок 7 - Тесты метода markov на отрицательных значениях

**Проведение тестирования**

* + - * **Тестирование метода chebyshev**

Таблица 1 - результаты проведения тестирования метода chebyshev

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Полученный результат** |
| 0, 1 | 1 | 1 |
| 1, 0.5 | 0.5 | 0.5 |
| 2, -4 | 31 | 31 |
| 8, 85 | 348691330125868801 | 348691330125868801 |

* + - * **Тестирование метода markov**

Таблица 2 - Результаты тестирования метода markov

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Полученный результат** |
| 0, 1 | 1 | 1 |
| 1, 0.5 | 0.5 | 0.5 |
| 2, 2.5 | 8.875 | 8.875 |
| 3, -5 | -305 | -305 |

Подтверждение описанных ранее результатов представлено на рисунках 8,9.

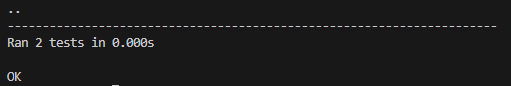


Рисунок 8 - результаты тестирования методов

# Оценка

В ходе анализа документации ошибок не было выявлено. В ходе проведённого тестирования не было выявлено ошибок, код отрабатывает во всех ожидаемых случаях.

# Выводы

В ходе данной практической работы были продемонстрированы возможности тестирования программного продукта с помощью модульного тестирования согласно патерну AAA.

# Список использованных источников

1. Статический анализ кода [Электронный ресурс] – <https://www.jetbrains.com/ru-ru/resharper/features/code_analysis.html>
2. Динамический анализ кода [Электронный ресурс] – <https://habr.com/ru/companies/pvs-studio/articles/580196/>
3. WebStrom – анализ кода [Электронный ресурс] – <https://www.jetbrains.com/ru-ru/resharper/features/code_analysis.html>
4. Использование статического и динамического анализа для повышения качества продукции и эффективности разработки [Электронный ресурс] – <https://www.swd.ru/print.php3?pid=828>