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)

# ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 2
**по дисциплине**
**«Тестирование и верификация программного обеспечения»**

Выполнил:
студент группы ИКБО-36-22 Шумахер М.Е.

Проверил:
Доцент Чернов Е.А.

Москва 2024

**1 Документация по программе**

1. Функция reverse_string принимает строку и возвращает ее символы в обратном порядке.

– Параметры: одна строка.

– Возвращаемое значение: строка с символами в обратном порядке.

– Как это работает: с помощью срезов строка разворачивается и возвращается.

2. Функция to_uppercase преобразует все символы строки в верхний регистр.

– Параметры: одна строка.

– Возвращаемое значение: строка, где все символы в верхнем регистре.

– Как это работает: используется метод upper(), который преобразует каждый символ строки в заглавную версию.

3. Функция is_substring проверяет, является ли одна строка подстрокой другой.

– Параметры: две строки, основная строка и подстрока.

– Возвращаемое значение: True, если подстрока встречается в основной строке, иначе False.

– Как это работает: используется оператор in для проверки наличия подстроки в строке.

4. Функция concatenate_strings объединяет две строки и возвращает их как одну строку.

– Параметры: две строки.

– Возвращаемое значение: новая строка, являющаяся результатом конкатенации двух входных строк.

– Как это работает: используется оператор сложения для соединения двух строк.

5. Функция count_vowels подсчитывает количество гласных в строке.

– Параметры: одна строка.

– Возвращаемое значение: целое число, которое указывает количество гласных.

– Как это работает: для каждой буквы в строке проверяется, является ли она гласной, и если да, она добавляется в счетчик.

```python
def reverse_string(s):
    return s[::1]

def to_uppercase(s):
    return s.upper()

def is_substring(s, sub):
    return sub in s

def concatenate_strings(s1, s2):
    return s1 + s2

def count_vowels(s):
    vowels = "aeiouAEIOU"
    return sum(1 for char in s if char in vowels)
```

Рисунок 1 – Код программы

## 2 Ошибки, найденные у коллеги

```python
import pytest
from main import *

# Тест для функции plusOne
@pytest.mark.parametrize("digits, expected", [
    ([1, 2, 3], [1, 2, 4]),  # Простое добавление единицы
    ([4, 3, 2, 1], [4, 3, 2, 2]),  # Добавление к числу без переноса
    ([9, 9, 9], [1, 0, 0, 0]),  # Добавление к числу с полным переносом
    ([0], [1]),  # Одноразрядное число
    ([9], [1, 0]),  # Число с единичным переносом
    ([8, 9, 9], [9, 0, 0]),  # Число с частичным переносом
])
def test_plusOne(digits, expected):
    assert list(plusOne(digits)) == expected

# Тест для функции climbStairs
@pytest.mark.parametrize("n, expected", [
    (1, 1),
    (2, 2),
    (3, 3),
    (4, 5),
    (5, 8),
    (6, 13)
])
def test_climbStairs(n, expected):
    assert climbStairs(n) == expected

# Тест для функции mySqrt
@pytest.mark.parametrize("x, expected", [
    (0, 0),
    (1, 1),
    (4, 2),
    (8, 2),
    (16, 4),
    (25, 5),
    (27, 5)
])
def test_mySqrt(x, expected):
    assert mySqrt(x) == expected

# Тест для функции findKthNumber
@pytest.mark.parametrize("n, k, expected", [
    (13, 2, 10),
```

Рисунок 2 – Часть кода с тестами

```
-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
=========================== short test summary info ============================
FAILED test.py::test_climbStairs[1-1] - assert 2 == 1
FAILED test.py::test_climbStairs[2-2] - assert 1 == 2
FAILED test.py::test_climbStairs[3-3] - assert 2 == 3
FAILED test.py::test_climbStairs[4-5] - assert 3 == 5
FAILED test.py::test_climbStairs[5-8] - assert 5 == 8
FAILED test.py::test_climbStairs[6-13] - assert 8 == 13
========== 6 failed, 23 passed, 89 warnings in 0.20s ==========
```

Рисунок 3 – Тесты не пройдены

**Краткое описание ошибки**: Неверное вычисление количества способов подъёма по лестнице.

**Статус ошибки**: открыта («Open»).

**Категория ошибки**: серьезная («Major»).

**Тестовый случай**: Тестирование функции climbStairs.

**Описание ошибки**:

1. Загрузить программу с функцией climbStairs.

2. Вызвать функцию с параметром n = 1.

3. Ожидаемый результат: 1.

4. Полученный результат: 2.

Другие тесты:

n=2, ожидаемый результат: 2, фактический результат: 1.

n=3, ожидаемый результат: 3, фактический результат: 2.

n=4, ожидаемый результат: 5, фактический результат: 3.

n=5, ожидаемый результат: 8, фактический результат: 5.

n=6, ожидаемый результат: 13, фактический результат: 8.

Ожидаемый результат: правильное количество способов для каждого значения n.

Исправим ошибку в программе и протестируем еще раз.

```python
# 2
def climbStairs(n):
    layers = list([1, 2])
    for i in range(2, n):
        layers.append(layers[-1] + layers[-2])
    return layers[n-2] # Тут ошибка, поменять на 1
```

Рисунок 4 – Исправление ошибки

```
============================= test session starts =============================
platform win32 -- Python 3.12.6, pytest-7.1.3, pluggy-1.5.0
rootdir: C:\Users\user\Downloads
plugins: django-4.5.2, pythonpath-0.7.3, timeout-2.3.1
collected 29 items

test.py .............................                                    [100%]

============================== 29 passed in 0.06s ==============================
```

Рисунок 5 – Успешное прохождение тестов

**Вывод**

В ходе работы был создан простой модуль программы и написано техническое задание на программный продукт, а также подготовлена необходимая документация. Программный продукт второй члена команды был протестирован юнит-тестами. В результате тестирования были выявлены ошибки, допущенные при разработке, которые были описаны с указанием фактических результатов и описанием ошибок, затем все ошибки были исправлены. По итогам анализа сделан вывод о необходимости совершенствования процесса тестирования и более строгого соблюдения требований технического задания.