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 № 3
по дисциплине
«Тестирование и верификация программного обеспечения»

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

Проверил:
Доцент Чернов Е.А.

Москва 2024

1 Реализация методологии TDD

Вариант 14: Шифратор и дешифратор файлов в заданной папке (шифруем все файлы и расшифровываем несколькими методами).

Создадим тесты, описывающие поведение функции.

D: > code > 5 > Тестирование > 3 > test.py > ...

```python
import unittest
import os
from main import *

class TestFileEncryption(unittest.TestCase):
    test_folder = 'D:\code\\5\Тестирование\\3\\test_folder'

    def test_encrypt_decrypt_file_caesar(self):
        encrypt_file(os.path.join(self.test_folder, "test.txt"), 3)
        with open(os.path.join(self.test_folder, "test.txt"), 'r') as f:
            content = f.read()
        self.assertEqual(content, "Khoor Zruog!")
        decrypt_file(os.path.join(self.test_folder, "test.txt"), 3)
        with open(os.path.join(self.test_folder, "test.txt"), 'r') as f:
            content = f.read()
        self.assertEqual(content, "Hello World!")

    def test_encrypt_decrypt_file_xor(self):
        encrypt_file_xor(os.path.join(self.test_folder, "test.txt"), 123)
        with open(os.path.join(self.test_folder, "test.txt"), 'r') as f:
            content = f.read()
        self.assertNotEqual(content, "3\x1e\x17\x17\x14[,\x14\t\x17\x1fC")
        decrypt_file_xor(os.path.join(self.test_folder, "test.txt"), 123)
        with open(os.path.join(self.test_folder, "test.txt"), 'r') as f:
            content = f.read()
        self.assertEqual(content, "Hello World!")

if __name__ == '__main__':
    unittest.main()
```

Рисунок 1 – Код тестов

Здесь мы используем 2 метода шифрования и прогоняем в тестах для проверки правильности шифрования и дешифрования.

Теперь реализуем функции шифрования и дешифрования.

D: > code > 5 > Тестирование > 3 > main.py > ...

```python
def caesar_cipher(text, shift):
    encrypted = ""
    for char in text:
        if char.isalpha():
            shift_base = 65 if char.isupper() else 97
            encrypted += chr((ord(char) + shift - shift_base) % 26 + shift_base)
        else:
            encrypted += char
    return encrypted

def encrypt_file(file_path, shift):
    with open(file_path, 'r') as file:
        content = file.read()
    encrypted_content = caesar_cipher(content, shift)
    with open(file_path, 'w') as file:
        file.write(encrypted_content)

def decrypt_file(file_path, shift):
    with open(file_path, 'r') as file:
        content = file.read()
    decrypted_content = caesar_cipher(content, -shift)
    with open(file_path, 'w') as file:
        file.write(decrypted_content)

def xor_cipher(text, key):
    encrypted = "".join(chr(ord(char) ^ key) for char in text)
    return encrypted

def encrypt_file_xor(file_path, key):
    with open(file_path, 'r') as file:
        content = file.read()
    encrypted_content = xor_cipher(content, key)
    with open(file_path, 'w') as file:
        file.write(encrypted_content)

def decrypt_file_xor(file_path, key):
    with open(file_path, 'r') as file:
        content = file.read()
    decrypted_content = xor_cipher(content, key)
    with open(file_path, 'w') as file:
        file.write(decrypted_content)
```

Рисунок 2 – Код реализации функций

Запускаем тесты и проверяем успех их прохождения.

```
PS C:\Users\shumila> python -u "d:\code\5\Тестирование\3\test.py"
..
----------------------------------------------------------------------
Ran 2 tests in 0.003s

OK
```

Рисунок 3 – Запуск тестов

Как видно из рисунка, тесты были пройдены успешно.

2 Реализация методологии BDD

Описание сценариев BDD для функциональности шифратора и дешифратора:

```
Feature: File encryption and decryption
  Scenario: Encrypt and decrypt file using Caesar cipher
    Given a folder with a file containing "Hello World!"
    When I encrypt the file with Caesar cipher and shift 3
    Then the file content should be "Khoor Zruog!"
    When I decrypt the file with Caesar cipher and shift 3
    Then the file content must be "Hello World!"
  Scenario: Encrypt and decrypt file using XOR cipher
    Given a folder with a file containing "Hello World!"
    When I encrypt the file with XOR cipher and key 123
    Then the file content should not be "Hello World!"
    When I decrypt the file with XOR cipher and key 123
    Then the file content should be "Hello World!"
```

Автоматизация сценариев с использованием behave python:

D: > code > 5 > Тестирование > 3 > steps > steps.py > ...

```python
# Задаем путь к тестовой папке
TEST_FOLDER = r'D:\code\5\Тестирование\3\test_folder'

@given('a folder with a file containing "Hello World!"')
def create_test_file(context):
    context.test_file = os.path.join(TEST_FOLDER, "test.txt")
    with open(context.test_file, 'w') as f:
        f.write("Hello World!")

@when('I encrypt the file with Caesar cipher and shift 3')
def encrypt_caesar_file(context):
    encrypt_file(context.test_file, 3)

@then('the file content should be "Khoor Zruog!"')
def check_caesar_encryption(context):
    with open(context.test_file, 'r') as f:
        content = f.read()
    assert content == "Khoor Zruog!"

@when('I decrypt the file with Caesar cipher and shift 3')
def decrypt_caesar_file(context):
    decrypt_file(context.test_file, 3)

@then('the file content must be "Hello World!"')
def check_caesar_decryption(context):
    with open(context.test_file, 'r') as f:
        content = f.read()
    assert content == "Hello World!"

@when('I encrypt the file with XOR cipher and key 123')
def encrypt_xor_file(context):
    encrypt_file_xor(context.test_file, 123)

@then('the file content should not be "Hello World!"')
def check_xor_encryption(context):
    with open(context.test_file, 'r') as f:
        content = f.read()
    assert content != "Hello World!"

@when('I decrypt the file with XOR cipher and key 123')
def decrypt_xor_file(context):
    decrypt_file_xor(context.test_file, 123)

@then('the file content should be "Hello World!"')
def check_xor_decryption(context):
    with open(context.test_file, 'r') as f:
        content = f.read()
    assert content == "Hello World!"
```

Рисунок 4 – Тесты на behave

Пишем реализацию шифратора дешифратора.

D: > code > 5 > Тестирование > 3 > main.py > ...

```python
def caesar_cipher(text, shift):
    encrypted = ""
    for char in text:
        if char.isalpha():
            shift_base = 65 if char.isupper() else 97
            encrypted += chr((ord(char) + shift - shift_base) % 26 + shift_base)
        else:
            encrypted += char
    return encrypted

def encrypt_file(file_path, shift):
    with open(file_path, 'r') as file:
        content = file.read()
    encrypted_content = caesar_cipher(content, shift)
    with open(file_path, 'w') as file:
        file.write(encrypted_content)

def decrypt_file(file_path, shift):
    with open(file_path, 'r') as file:
        content = file.read()
    decrypted_content = caesar_cipher(content, -shift)
    with open(file_path, 'w') as file:
        file.write(decrypted_content)

def xor_cipher(text, key):
    encrypted = "".join(chr(ord(char) ^ key) for char in text)
    return encrypted

def encrypt_file_xor(file_path, key):
    with open(file_path, 'r') as file:
        content = file.read()
    encrypted_content = xor_cipher(content, key)
    with open(file_path, 'w') as file:
        file.write(encrypted_content)

def decrypt_file_xor(file_path, key):
    with open(file_path, 'r') as file:
        content = file.read()
    decrypted_content = xor_cipher(content, key)
    with open(file_path, 'w') as file:
        file.write(decrypted_content)
```

Рисунок 5 – Код реализации функций

Запускаем тесты и смотрим на их результат.

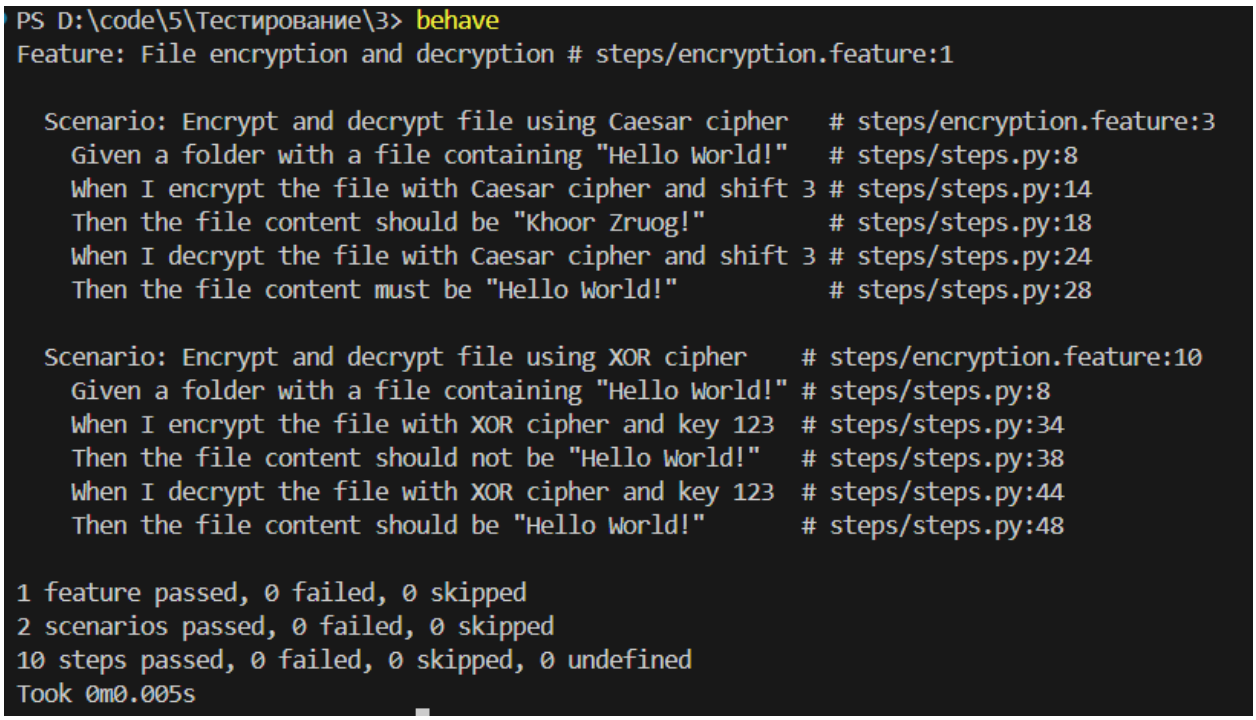

Рисунок 6 – Тестирование

Все тесты пройдены успешно.

Вывод

В процессе работы, применяя методологии TDD и BDD, был создан модуль программы, начальная реализация которого велась через написание тестов, что позволило эффективно выявить ошибки на ранних этапах разработки. Юнит-тесты в рамках TDD помогли контролировать соответствие программы ожидаемому поведению, а сценарии BDD обеспечили прозрачность требований.