Лабораторная работа №3

Исследование способов модульного тестирования программного обеспечения

1. Цель работы

Исследовать основные подходы к модульному тестированию программного обеспечения. Приобрести практические навыки составления модульных тестов для объектно-ориентированных программ.

2. Постановка задачи

Вариант — 22

Выбрать в качестве тестируемого один из классов, спроектированных в лабораторной работе №1. Составить спецификацию тестового случая для одного из методов выбранного класса. Реализовать тестируемый класс и необходимое тестовое окружение. Выполнить тестирование с выводом результатов на экран и сохранением в log-файл. Проанализировать результаты тестирования, сделать выводы.

3. Ход работы

3.1 В связи с тем, что в предыдущих лабораторных работах была написана программа без использования ООП, было принято решение написать программу с использованием ООП выполняющая одну из требуемых функций, а именно «Преобразование строки: если нет символа #, то оставить её без изменения, иначе заменить каждый символ, встречающийся после первого вхождения # на символ @». Текст этой программы представлен в листинге 1.

Листинг 1 – Программа преобразования строки с использованием ООП

```
#include <iostream>
#include <string>

class StringTransformer {
public:
    static std::string transform(const std::string& str) {
        std::string transformed_str = str;
        bool found_first_hash = false;
```

```
        for (char& c : transformed_str) {
            if (c == '#') {
                found_first_hash = true;
            }
            else if (found_first_hash) {
                c = '@';
            }
        }

        return transformed_str;
    }
};

int main() {

    std::string str;
    std::cout << "Input string:" << std::endl;
    std::cin >> str;

    std::string transformed_str = StringTransformer::transform(str);

    std::cout << "Original string: " << str << std::endl;
    std::cout << "Transformed string: " << transformed_str << std::endl;

    return 0;
}
```

3.2 Для тестируемого класса был определён тестовый случай: проверяется правильность работы класса StringTransformer – который преобразует строку по следующему правилу: “если нет символа #, то оставить её без изменения, иначе заменить каждый символ, встречающийся после первого вхождения # на символ @”. В тесте подаются значения, определённые в лабораторной работе №1.

На основе спецификации был создан тестовый драйвер, реализованный в виде функции и тестирования в функции main, вызывающего public и protected методы тестируемого класса. Текст драйвера представлен в листинге 2.

Листинг 2 – Тестирование класса StringTransformer

```
// Функция для записи результатов тестирования в log-файл
void writeLog(const std::string& message, const std::string& fileName) {
    std::ofstream file(fileName, std::ios_base::app);
    if (!file.is_open()) {
        std::cerr << "Ошибка при открытии файла " << fileName << std::endl;
        return;
    }
    file << message << std::endl;
    file.close();
}
```

```
int main() {
    // Тестовый сценарий 1: проверка преобразования строки "Hello, #world!" в "Hello, @@@@@@"
    std::string str1 = "Hello, #world!";
    std::string expected1 = "Hello, #@@@@@@";
    std::string result1 = StringTransformer::transform(str1);
    assert(result1 == expected1);
    std::string message1 = "Тест 1 пройден успешно";
    std::cout << message1 << std::endl;
    writeLog(message1, "test.log");

    // Тестовый сценарий 2: проверка преобразования строки "#dvjnsialw" в "#@@@@@@@@@"
    std::string str2 = "#dvjnsialw";
    std::string expected2 = "#@@@@@@@@@";
    std::string result2 = StringTransformer::transform(str2);
    assert(result2 == expected2);
    std::string message2 = "Тест 2 пройден успешно";
    std::cout << message2 << std::endl;
    writeLog(message2, "test.log");

    // Тестовый сценарий 3: проверка преобразования строки "no hashes" в "no hashes"
    std::string str3 = "no hashes";
    std::string expected3 = "no hashes";
    std::string result3 = StringTransformer::transform(str3);
    assert(result3 == expected3);
    std::string message3 = "Тест 3 пройден успешно";
    std::cout << message3 << std::endl;
    writeLog(message3, "test.log");

    return 0;
}
```

Результаты тестирования можно видеть в log файле или на экране консоли (рис. 1).

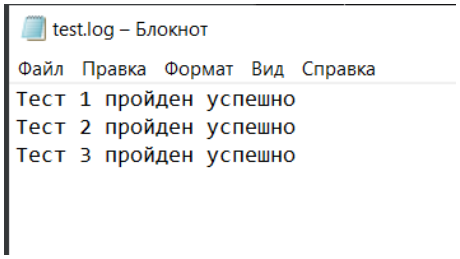

Рисунок 1 – Результат тестирования

Полученные данных были проанализированы и сравнены со спецификацией. Данные полностью соответствуют спецификации.

Выводы

При выполнении данный лабораторной работы были получены навыки составления модульных тестов. Было выделено, что модульные тесты необходимо применять при тестировании модуля программы, или, в случае этой работы, при тестировании класса.