Министерство образования и науки
федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

Отчет по дисциплине: **«Тестирование программного обеспечения»**

Лабораторная работа 1

Выполнила:

Королева А.С.

Группа:

К3322

Проверил:

Кочубеев Н.С.

Санкт-Петербург,

2024

Цель: научиться писать unit тесты с использованием AAA и FIRST.

Задачи:

- Выбрать репозиторий для тестирования с GitHub;
- Проанализировать функциональность приложения и определить, какие модули или компоненты требуют тестирования;
- Написать тесты с использованием AAA и FIRST;
- Подготовить отчет о проделанной работе.

Ход работы

Ссылка на репозиторий: https://github.com/nastyakrlv/Testing_PO

Выбор репозитория для тестирования с GitHub

Проект в рамках лабораторной работы использует алгоритмы для обработки лабиринтов. Это отличный пример для тестирования, так как он включает различные сценарии и условия.

Краткое описание проекта:

Реализована задача по анализу 2D массивов, представляющих собой лабиринты, где 1 обозначает проход, а 0 — стену. Программа считывает данные из файла tasks/labyrinths/text.txt, анализирует лабиринты и определяет их корректность по заданным критериям.

Проект написан на JavaScript, тестирование произведено на Jest.

Анализ тестируемых функциональностей

Функциональные элементы:

- getDataFromFile: считывает лабиринты из файла;
- scan: анализирует лабиринт и возвращает статистику по путям (количество путей от пола, потолка, изолированных путей и путей, проходящих в обе стороны);
- main: определение корректности лабиринта на основе полученных данных;

- queue: создание очереди для поиска в ширину;
- sendDataToServer: отправка результатов на сервер, если количество корректных лабиринтов больше некорректных.

Критические части системы, которые должны быть протестированы:

- Обработка файловых данных: как программа реагирует на отсутствие файла или некорректные данные;
- Правильность подсчета путей и определение корректности лабиринтов;
- Условия отправки данных на сервер.

Важные случаи использования:

- Пустой файл или отсутствие данных;
- Лабиринты с разными конфигурациями;
- Проверка корректных и некорректных лабиринтов.

Написание тестов

Были написаны модульные тесты.

tasks > labyrinths > JS main.test.js > describe("Сценарии файла main") callback

```
const main = require("./main");
const getDataFromFile = require("./read");
const sendDataToServer = require("./send");

jest.mock("./read");
jest.mock("./send");

describe("Сценарии файла main", () => {
  afterEach(() => {
    jest.resetAllMocks();
  });

  test(`Если один лабиринт, и у данного лабиринта путей соприкасающихся с полом и потолком больше,
    чем всех остальных, то он считается правильным и функция возвращает true`, () => {
    // Arrange
    const labyrinth = [
      [1, 1, 0, 1, 1, 0, 0, 1],
      [1, 1, 0, 1, 1, 0, 0, 1],
      [1, 0, 0, 0, 1, 0, 0, 1],
      [1, 0, 0, 0, 1, 0, 0, 1],
      [1, 0, 0, 0, 1, 0, 0, 1],
      [1, 0, 0, 0, 1, 0, 0, 1],
    ];

    getDataFromFile.mockReturnValue([labyrinth]);

    // Act
    const result = main();

    // Assert
    expect(result).toBe(true);
  });

```

Рисунок 1 - Пример теста для функции main

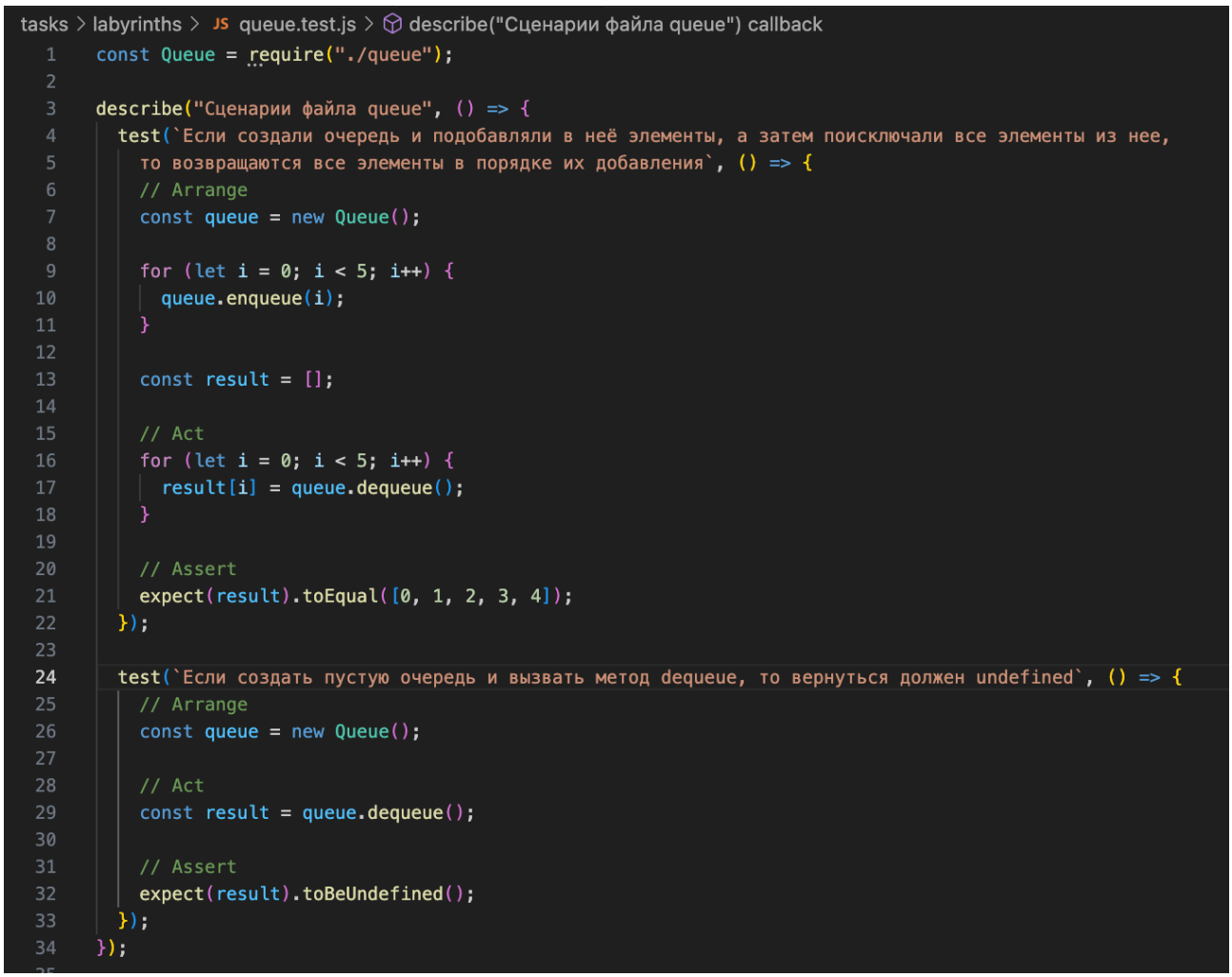

Рисунок 2 - Пример тестов для функции Queue

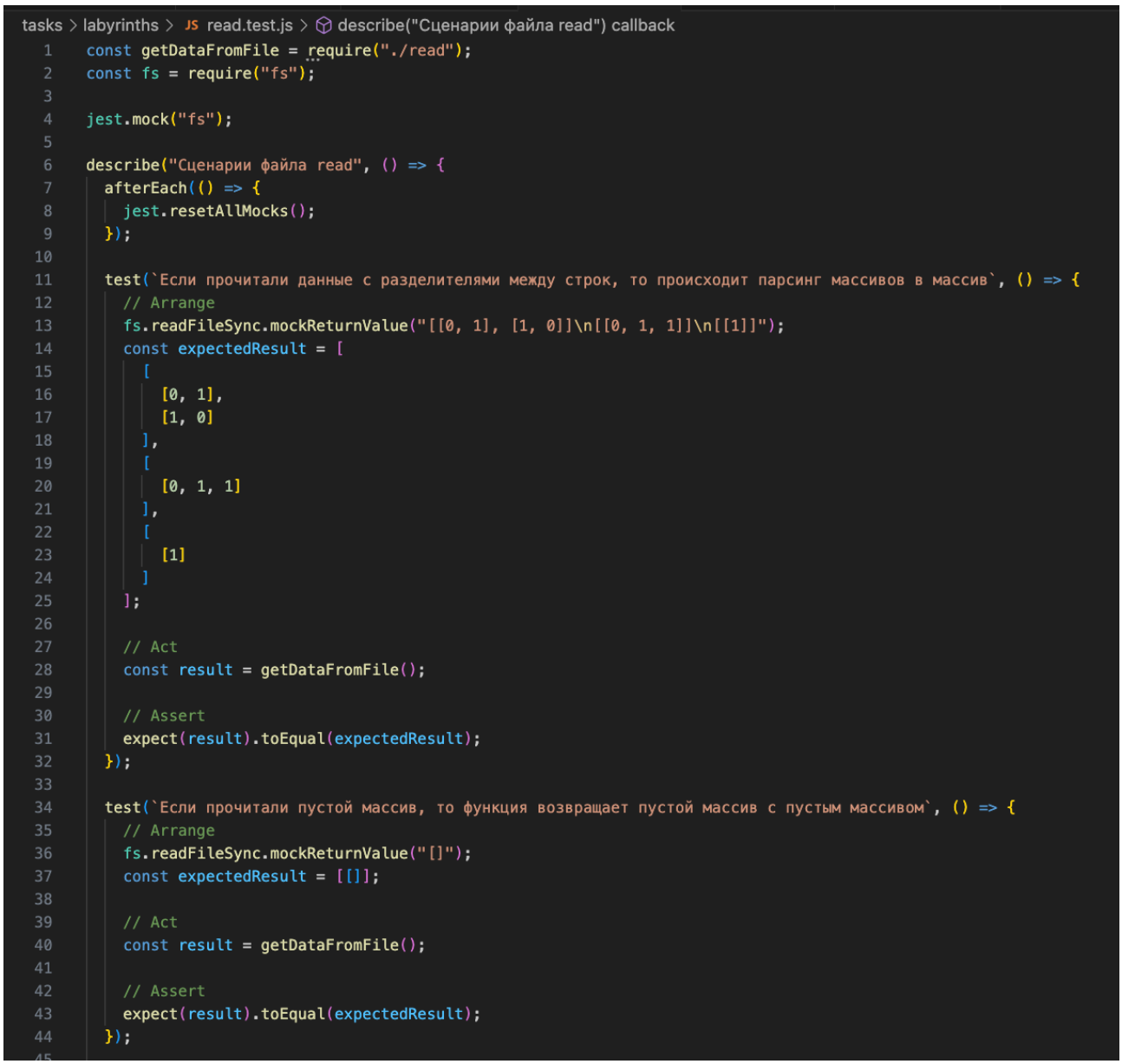

Рисунок 3 - пример тестов для функции getDataFromFile

```
const scan = require("./scan");

describe("Сценарии файла scan", () => {
  test(`
    Если вызываем метод сканирования лабиринта и передаем двумерный массив, где должен быть только один изолированный путь,
    то вернется объект, где только поле isolated будет равно 1, а остальные 0
  `, () => {
    // Arrange
    const labyrinth = [
      [0, 0, 0, 0, 0, 0, 0, 0],
      [0, 0, 0, 0, 0, 0, 0, 1],
      [0, 0, 0, 0, 0, 0, 0, 1],
      [0, 0, 0, 0, 0, 0, 0, 1],
      [0, 0, 0, 0, 0, 0, 0, 1],
      [0, 0, 0, 0, 0, 0, 0, 0]
    ];
    const expectedResult = { ceil: 0, floor: 0, both: 0, isolated: 1 };

    // Act
    const result = scan(labyrinth);

    // Assert
    expect(result).toStrictEqual(expectedResult);
  });

  test(`
    Если вызываем метод сканирования лабиринта и передаем двумерный массив, где должен быть только один путь,
    начинающийся с потолка, то вернется объект, где только поле ceil будет равно 1, а остальные 0
  `, () => {
    // Arrange
    const labyrinth = [
      [0, 0, 0, 0, 0, 0, 0, 1],
      [0, 0, 0, 0, 0, 0, 0, 1],
      [0, 0, 0, 0, 0, 0, 0, 1],
      [0, 0, 0, 0, 0, 0, 0, 1],
      [0, 0, 0, 0, 0, 0, 0, 1],
      [0, 0, 0, 0, 0, 0, 0, 0]
    ];
    const expectedResult = { ceil: 1, floor: 0, both: 0, isolated: 0 };

    // Act
    const result = scan(labyrinth);

    // Assert
    expect(result).toStrictEqual(expectedResult);
  });
```

Рисунок 4 - Пример тестов для функции scan

Рисунок 5 - Пример тестов для функции sendDataToServer

Результаты покрытия кода и выводы о качестве тестирования

После написания тестов и их запуска, было получено следующее покрытие кода:

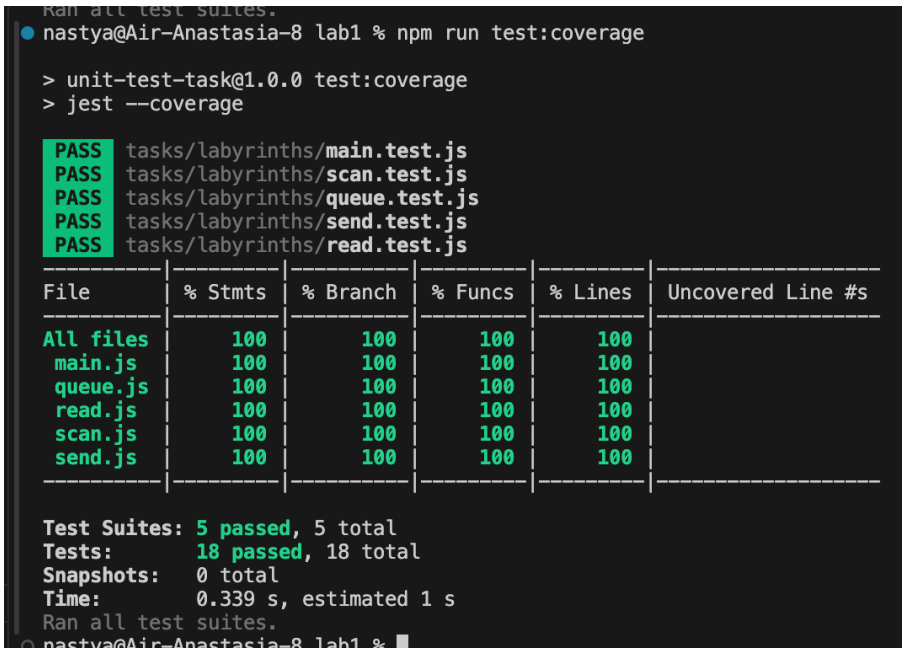

Рисунок 6 - результаты тестирования

Вывод: был проведен анализ приложения, написаны unit тесты, подготовлен отчет о проведенном тестировании. Тестирование охватывает основную часть проекта, включая чтение данных, анализ лабиринтов и проверку их корректности.