

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
1  const main = require("./main");
2  const getDataFromFile = require("./read");
3  const sendDataToServer = require("./send");
4
5  jest.mock("./read");
6  jest.mock("./send");
7
8  describe("Сценарии файла main", () => {
9      afterEach(() => {
10         jest.resetAllMocks();
11     });
12
13     test(`Если один лабиринт, и у данного лабиринта путей соприкасающихся с полом и потолком больше,
14         чем всех остальных, то он считается правильным и функция возвращает true`, () => {
15         // Arrange
16         const labyrinth = [
17             [1, 1, 0, 1, 1, 0, 0, 1],
18             [1, 1, 0, 1, 1, 0, 0, 1],
19             [1, 0, 0, 0, 1, 0, 0, 1],
20             [1, 0, 0, 0, 1, 0, 0, 1],
21             [1, 0, 0, 0, 1, 0, 0, 1],
22             [1, 0, 0, 0, 1, 0, 0, 1],
23         ];
24
25         getDataFromFile.mockReturnValue([labyrinth]);
26
27         // Act
28         const result = main();
29
30         // Assert
31         expect(result).toBe(true);
32     });
33

```

Рисунок 1 - Пример теста для функции main

```

tasks > labyrinths > JS queue.test.js > describe("Сценарии файла queue") callback
1  const Queue = require("./queue");
2
3  describe("Сценарии файла queue", () => {
4      test(`Если создали очередь и добавляли в неё элементы, а затем поисключали все элементы из нее,
5          то возвращаются все элементы в порядке их добавления`, () => {
6          // Arrange
7          const queue = new Queue();
8
9          for (let i = 0; i < 5; i++) {
10             queue.enqueue(i);
11         }
12
13         const result = [];
14
15         // Act
16         for (let i = 0; i < 5; i++) {
17             result[i] = queue.dequeue();
18         }
19
20         // Assert
21         expect(result).toEqual([0, 1, 2, 3, 4]);
22     });
23
24     test(`Если создать пустую очередь и вызвать метод dequeue, то вернуться должен undefined`, () => {
25         // Arrange
26         const queue = new Queue();
27
28         // Act
29         const result = queue.dequeue();
30
31         // Assert
32         expect(result).toBeUndefined();
33     });
34 });
35

```

Рисунок 2 - Пример тестов для функции Queue

```

tasks > labyrinths > JS read.test.js > describe("Сценарии файла read") callback
1  const getDataFromFile = require("../read");
2  const fs = require("fs");
3
4  jest.mock("fs");
5
6  describe("Сценарии файла read", () => {
7    afterEach(() => {
8      jest.resetAllMocks();
9    });
10
11    test(`Если прочитали данные с разделителями между строк, то происходит парсинг массивов в массив`, () => {
12      // Arrange
13      fs.readFileSync.mockReturnValue("[[0, 1], [1, 0]]\n[[0, 1, 1]]\n[[1]]");
14      const expectedResult = [
15        [
16          [0, 1],
17          [1, 0]
18        ],
19        [
20          [0, 1, 1]
21        ],
22        [
23          [1]
24        ]
25      ];
26
27      // Act
28      const result = getDataFromFile();
29
30      // Assert
31      expect(result).toEqual(expectedResult);
32    });
33
34    test(`Если прочитали пустой массив, то функция возвращает пустой массив с пустым массивом`, () => {
35      // Arrange
36      fs.readFileSync.mockReturnValue("");
37      const expectedResult = [[]];
38
39      // Act
40      const result = getDataFromFile();
41
42      // Assert
43      expect(result).toEqual(expectedResult);
44    });
45

```

Рисунок 3 - пример тестов для функции getDataFromFile

```

1  const scan = require("../scan");
2
3  describe("Сценарии файла scan", () => {
4    test(`
5      Если вызываем метод сканирования лабиринта и передаем двумерный массив, где должен быть только один изолированный путь,
6      то вернется объект, где только поле isolated будет равно 1, а остальные 0
7    `, () => {
8      // Arrange
9      const labyrinth = [
10        [0, 0, 0, 0, 0, 0, 0, 0],
11        [0, 0, 0, 0, 0, 0, 0, 1],
12        [0, 0, 0, 0, 0, 0, 0, 1],
13        [0, 0, 0, 0, 0, 0, 0, 1],
14        [0, 0, 0, 0, 0, 0, 0, 1],
15        [0, 0, 0, 0, 0, 0, 0, 0]
16      ];
17      const expectedResult = { ceil: 0, floor: 0, both: 0, isolated: 1 };
18
19      // Act
20      const result = scan(labyrinth);
21
22      // Assert
23      expect(result).toStrictEqual(expectedResult);
24    });
25
26    test(`
27      Если вызываем метод сканирования лабиринта и передаем двумерный массив, где должен быть только один путь,
28      начинающийся с потолка, то вернется объект, где только поле ceil будет равно 1, а остальные 0
29    `, () => {
30      // Arrange
31      const labyrinth = [
32        [0, 0, 0, 0, 0, 0, 0, 1],
33        [0, 0, 0, 0, 0, 0, 0, 1],
34        [0, 0, 0, 0, 0, 0, 0, 1],
35        [0, 0, 0, 0, 0, 0, 0, 1],
36        [0, 0, 0, 0, 0, 0, 0, 1],
37        [0, 0, 0, 0, 0, 0, 0, 0]
38      ];
39      const expectedResult = { ceil: 1, floor: 0, both: 0, isolated: 0 };
40
41      // Act
42      const result = scan(labyrinth);
43
44      // Assert
45      expect(result).toStrictEqual(expectedResult);
46    });
47

```

Рисунок 4 - Пример тестов для функции scan

```

tasks > labyrinths > JS send.test.js > describe("Сценарии файла send") callback
1  const http = require("http");
2  const sendDataToServer = require("../send");
3
4  jest.mock("http");
5
6  const mockWrite = jest.fn();
7  const mockEnd = jest.fn();
8  http.request.mockReturnValue({
9    write: mockWrite,
10   end: mockEnd,
11 });
12
13 describe("Сценарии файла send", () => {
14   test("Если вызываем метод отправки данных на сервер, то создается запрос с корректным хостом", () => {
15     // Arrange
16     const data = { passed: true };
17
18     // Act
19     sendDataToServer(data);
20
21     // Assert
22     expect(http.request).toHaveBeenCalledWith(expect.objectContaining({
23       hostname: "localhost"
24     }));
25   });
26
27   test("Если вызываем метод отправки данных на сервер, то создается запрос с корректным заголовком типа контента", () => {
28     // Arrange
29     const data = { passed: true };
30
31     // Act
32     sendDataToServer(data);
33
34     // Assert
35     expect(http.request).toHaveBeenCalledWith(expect.objectContaining({
36       headers: expect.objectContaining({
37         "Content-Type": "application/json"
38       })
39     }));
40   });
41
42   test("Если вызываем метод отправки данных на сервер, то происходит отправка запроса с данными", () => {
43     // Arrange
44     const data = { passed: true };
45     const expectedData = JSON.stringify(data);
46
47     // Act
48     sendDataToServer(data);
49
50     // Assert
51     expect(mockWrite).toHaveBeenCalledWith(expectedData);
52   });
53
54   // ...

```

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:


```
Ran all test suites.  
● nastya@Air-Anastasia-8 lab1 % npm run test:coverage  
  
> unit-test-task@1.0.0 test:coverage  
> jest --coverage  
  
PASS tasks/labyrinths/main.test.js  
PASS tasks/labyrinths/scan.test.js  
PASS tasks/labyrinths/queue.test.js  
PASS tasks/labyrinths/send.test.js  
PASS tasks/labyrinths/read.test.js  


| File      | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s |
|-----------|---------|----------|---------|---------|-------------------|
| All files | 100     | 100      | 100     | 100     |                   |
| main.js   | 100     | 100      | 100     | 100     |                   |
| queue.js  | 100     | 100      | 100     | 100     |                   |
| read.js   | 100     | 100      | 100     | 100     |                   |
| scan.js   | 100     | 100      | 100     | 100     |                   |
| send.js   | 100     | 100      | 100     | 100     |                   |

  
Test Suites: 5 passed, 5 total  
Tests: 18 passed, 18 total  
Snapshots: 0 total  
Time: 0.339 s, estimated 1 s  
Ran all test suites.  
● nastya@Air-Anastasia-8 lab1 %
```

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.