Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное  
учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» (ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

**Практическая работа №5**

по курсу «Технологии разработки серверной части веб-приложений» модуль «Unit-тестирование»

Выполнил

студент группы КТбо3-4 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Попов Л. А.

Принял

ассистент кафедры МОП ЭВМ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жиглатый А. А.

# Введение

## Цель работы

Научиться применять стратегии при проектировании тестовых наборов для проведения модульного тестирования. Освоить существующие средства автоматизации проведения юнит-тестирования.

## Задание

1. Выполнить подбор задачи, решение которой может быть оформлено в виде одной отдельной основной функции, не включающей в себя чтение входных данных и вывод результата. Входные данные не должны исчерпываться одним входным параметром. Выбранную задачу обсудить с преподавателем.

2. Выполнить корректную постановку выбранной задачи, которая должна включать: формулировку условия; описание входных данных с указанием их состава, формата представления и области допустимых значений для каждого элемента входных данных; описание выходных данных с указанием их состава и формата представления; примеры входных и выходных данных; возможные способы обработки входных данных из области недопустимых значений; возможные указания и дополнительные требования к реализации.

3. Выполнить оформление решения поставленной задачи в виде одной отдельной основной функции. Допускается наличие вспомогательных 2 функций, реализующих решение подзадач, возникающих в ходе решения поставленной задачи. Также допускается вызов вспомогательных функций из основной. Однако, ввод входных данных и вывод выходных данных должен быть исключен из основной функции, в том числе и посредством вспомогательных функций. Источником входных данных должен являться файл. Вывод результата также необходимо осуществлять в файл.

4. Разработать тестовый набор для проверки корректности решения поставленной задачи (основной функции), используя стратегию, позволяющую минимизировать количество тестовых случаев при сохранении максимально возможного тестового покрытия. По возможности оценить величину тестового покрытия для разработанного тестового набора.

5. Используя встроенные в среду разработки средства юнит-тестирования и разработанный тестовый набор, выполнить юнит-тестирование основной функции, реализующей решение поставленной задачи. При этом необходимо использовать файлы в качестве источника входных и эталонных выходных данных. Не допускается их явное задание в коде проекта юнит-тестирования.

6. При обнаружении дефектов в ходе юнит-тестирования выполнить отладку основной функции и повторное её тестирование.

# Ход работы

## 1. Выбор задачи

В качестве задачи для проведения unit-тестирования было выбрано одно из заданий архива сайта acmp.ru под названием «Три толстяка» (№754).

## 2. Постановка задачи

### 2.1. Формулировка условия

Три толстяка решили поспорить: кто из них самый тяжелый. После взвешивания оказалось, что их масса равняется соответственно M1, M2 и M3 килограмм с точностью до одного килограмма. Считается, что масса толстяка должна быть не менее 94 и не более 727 килограммов.

Помогите определить массу m самого тяжелого из них.

### 2.2. Входные данные

Натуральные числа *M1, M2, M3*. Диапазон значений: **94 ≤ M1, M2, M3 ≤ 727.**

Формат текстового представления: в единственной строке через пробел три числа *M1, M2, M3*.

### 2.3. Выходные данные

Натуральное число m.

Формат текстового представления: в единственной строке одно число m.

### 2.4. Обработка исключительных ситуаций

В случае, если функция вызвана с меньшим количеством аргументов, чем 3, функция должна сгенерировать и вывести стандартное исключение Error со строковым сообщением «Недостаточное количество входных данных».

В случае, если функция вызвана с большим количеством аргументов, чем 3, функция должна сгенерировать и вывести стандартное исключение Error со строковым сообщением «Избыточное количество входных данных».

В случае, если значение Mi не является числом, функция должна сгенерировать и вывести стандартное исключение Error со строковым сообщением «Значения не являются числовыми».

В случае, если значение Mi не является целым числом, функция должна сгенерировать и вывести стандартное исключение Error со строковым сообщением «Значения не являются целочисленными».

В случае выхода Mi за граничные значения функция должна сгенерировать и вывести стандартное исключение Error со строковым сообщением «Значения за пределами допустимого диапазона».

### 2.5. Примеры входных и выходных данных

Таблица – Входные и выходные данные

|  |  |  |
| --- | --- | --- |
| ***№*** | ***Входные данные*** | ***Выходные данные*** |
| 1 | 98 106 214 | 214 |
| 2 | 100 100 100 | 100 |
| 3 | 93 500 1000 | Исключение «Значения за пределами допустимого диапазона» |

## 3. Алгоритм решения поставленной задачи

Решение задачи было реализовано с помощью языка Javascript. Код программы приведен ниже:

function sum(a, b, c, ...rest) {

    if (typeof a === 'undefined' || typeof b === 'undefined')

        throw new Error('Недостаточное количество входных данных');

    if (rest.length != 0)

        throw new Error('Избыточное количество входных данных');

    if (typeof a != 'number' || typeof c != 'number')

        throw new Error('Значения не являются числовыми');

    if (a - Math.trunc(a) != 0 || b - Math.trunc(b) != 0 || c - Math.trunc(c) != 0)

        throw new Error('Значения не являются целочисленными');

    if (a < 94 || b < 94 || c < 94 || a > 727 || b > 727 || c > 727)

        throw new Error('Значения за пределами допустимого диапазона');

    if (a >= b && a>= c) { return a; }

    else if (b >= a) { return b;}

    else {return c;}

  }

  module.exports = sum;

Листинг 1 – Исходный алгоритм решения

## 4. Составление тестов

Для составления минимального количества тестов, имеющих максимальное тестовое покрытие выделим следующие тесты:

1. Позитивные тесты: 7 тестов. 3 теста, в каждом из которых только одно Mi является наибольшим, 3 теста, в каждом из которых два Mi являются наибольшими, и тест, в котором все Mi равны.

2. Тесты граничных значений: 6 тестов на каждую границу интервала каждого Mi.

3. Негативные тесты: 16 тестов. 3 теста на вызов функции с двумя, одним и нулём переменных соответственно. 1 тест на вызов функции с избыточным количеством переменных. 3 теста на вызов функции с не числовыми переменными. 3 теста на вызов функции с не целочисленными переменами. 6 тестов на проверку каждого Mi на выход за правую и левую границу допустимого интервала.

Таблица – Список тестов

|  |  |  |  |
| --- | --- | --- | --- |
| ***№*** | ***Входные данные*** | ***Выходные данные*** | ***Комментарий*** |
| 1 | 690 540 320 | 690 | Позитивный. Наибольшим является первый аргумент |
| 2 | 120 380 255 | 380 | Позитивный. Наибольшим является второй аргумент |
| 3 | 340 112 456 | 456 | Позитивный. Наибольшим является третий аргумент |
| 4 | 710 710 325 | 710 | Позитивный. Наибольшими являются первый и второй аргумент |
| 5 | 560 165 560 | 560 | Позитивный. Наибольшими являются первый и третий аргумент |
| 6 | 212 310 310 | 310 | Позитивный. Наибольшими являются второй и третий аргумент |
| 7 | 180 180 180 | 180 | Позитивный. Все три аргумента равны. |
| 8 | 94 667 256 | 667 | Позитивный. Первый аргумент равен правой границе допустимого интервала. |
| 9 | 727 345 512 | 727 | Позитивный. Первый аргумент равен левой границе допустимого интервала |
| 10 | 456 94 333 | 456 | Позитивный. Второй аргумент равен правой границе допустимого интервала. |
| 11 | 654 727 128 | 727 | Позитивный. Второй аргумент равен левой границе допустимого интервала |
| 12 | 341 567 94 | 567 | Позитивный. Третий аргумент равен правой границе допустимого интервала. |
| 13 | 344 344 727 | 727 | Позитивный. Третий аргумент равен левой границе допустимого интервала |
| 14 | 690 540 | Исключение «Недостаточное количество входных данных» | Негативный. Вызов функции с двумя аргументами |
| 15 | 690 | Исключение «Недостаточное количество входных данных» | Негативный. Вызов функции с одним аргументом |
| 16 | «Нет данных» | Исключение «Недостаточное количество входных данных» | Негативный. Вызов функции с нулём аргументов |
| 17 | 96 567 433 221 | Исключение «Избыточное количество входных данных» | Негативный. Вызов функции с четырьмя аргументами |
| 18 | 'six hundred and fifty' 540 320 | Исключение «Значения не являются числовыми» | Негативный. Первый аргумент не является числом |
| 19 | 120 true 255 | Исключение «Значения не являются числовыми» | Негативный. Второй аргумент не является числом |
| 20 | 340 112 '2n8' | Исключение «Значения не являются числовыми» | Негативный. Третий аргумент не является числом |
| 21 | 255.212334 540 320 | Исключение «Значения не являются целочисленными» | Негативный. Первый аргумент не является целым числом |
| 22 | 120 600.3 255 | Исключение «Значения не являются целочисленными» | Негативный. Второй аргумент не является целым числом |
|  | 340 112 243.12 | Исключение «Значения не являются целочисленными» | Негативный. Третий аргумент не является целым числом |
| 23 | 93 677 677 | Исключение «Значения за пределами допустимого диапазона» | Негативный. Первый аргумент выходит за правое граничное значение |
| 24 | 728 211 342 | Исключение «Значения за пределами допустимого диапазона» | Негативный. Первый аргумент выходит за левое граничное значение |
| 25 | 444 93 432 | Исключение «Значения за пределами допустимого диапазона» | Негативный. Второй аргумент выходит за правое граничное значение |
| 26 | 324 728 122 | Исключение «Значения за пределами допустимого диапазона» | Негативный. Второй аргумент выходит за левое граничное значение |
| 27 | 231 133 93 | Исключение «Значения за пределами допустимого диапазона» | Негативный. Третий аргумент выходит за правое граничное значение |
| 28 | 233 655 728 | Исключение «Значения за пределами допустимого диапазона» | Негативный. Третий аргумент выходит за левое граничное значение |

## 5. Юнит-тестирование

Для проведения автоматизированного юнит-тестирования была использована библиотека Jest. Тесты описаны в файле index.test.js в виде набора методов, которые являются тест-кейсами. Результаты выполнения тестов можно увидеть на скриншоте ниже:

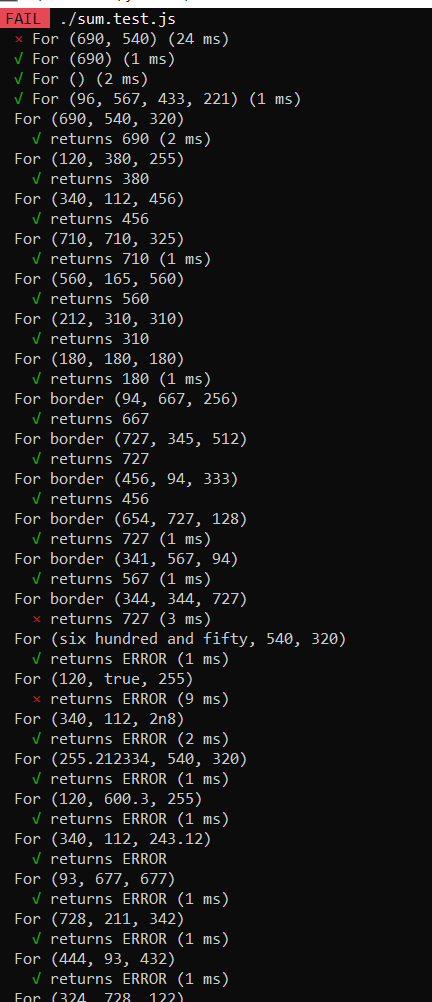


Рис. – результат юнит тестирования

6. Баг-репорты

Дефект №1

|  |  |
| --- | --- |
| Название дефекта | Провал теста, когда третий аргумент равен левому граничному значению. Полученное значение не соответствует ожидаемому |
| Приоритет дефекта | Высокий |
| Серьезность дефекта | Критический |
| Версия системы | ОС Windows 10, NodeJS v18.14.0 |
| Инструкция по воспроизведению дефекта | Входные данные:  344 344 727 |
| Фактический результат | 344 |
| Ожидаемый результат | 727 |
| Вид, тип, область тестирования | Функциональное, модульное тестирование |
| Вложения |  |

Дефект №2

|  |  |
| --- | --- |
| Название дефекта | Провал теста вызова функции с двумя аргументами. Возвращает несоответствующее ожидаемому исключение |
| Приоритет дефекта | Высокий |
| Серьезность дефекта | Критический |
| Версия системы | ОС Windows 10, NodeJS v18.14.0 |
| Инструкция по воспроизведению дефекта | Входные данные:  690 540 |
| Фактический результат | Исключение "Значения не являются числовыми" |
| Ожидаемый результат | Исключение "Недостаточное количество входных данных" |
| Вид, тип, область тестирования | Функциональное, модульное тестирование |
| Вложения |  |

Дефект №3

|  |  |
| --- | --- |
| Название дефекта | Провал теста, когда второй аргумент функции не является числовым. Возвращает несоответствующее ожидаемому исключение |
| Приоритет дефекта | Высокий |
| Серьезность дефекта | Критический |
| Версия системы | ОС Windows 10, NodeJS v18.14.0 |
| Инструкция по воспроизведению дефекта | Входные данные:  120, true, 255 |
| Фактический результат | Исключение "Значения за пределами допустимого диапазона" |
| Ожидаемый результат | Исключение "Значения не являются числовыми" |
| Вид, тип, область тестирования | Функциональное, модульное тестирование |
| Вложения |  |

7. Исправление дефектов

Для того, чтобы функция проходила все тесты, необходимо внести в неё следующие изменения:

function sum(a, b, c, ...rest) {

    if (typeof a === 'undefined' || typeof b === 'undefined' || typeof c === 'undefined')

        throw new Error('Недостаточное количество входных данных');

    if (rest.length != 0)

        throw new Error('Избыточное количество входных данных');

    if (typeof a != 'number' || typeof b != 'number' || typeof c != 'number')

        throw new Error('Значения не являются числовыми');

    if (a - Math.trunc(a) != 0 || b - Math.trunc(b) != 0 || c - Math.trunc(c) != 0)

        throw new Error('Значения не являются целочисленными');

    if (a < 94 || b < 94 || c < 94 || a > 727 || b > 727 || c > 727)

        throw new Error('Значения за пределами допустимого диапазона');

    if (a >= b && a>= c) { return a; }

    else if (b >= a && b >= c) { return b;}

    else {return c;}

  }

  module.exports = sum;

Листинг 2 — Исправленный алгоритм решения

После этого юнит-тестирование завершается без ошибок.

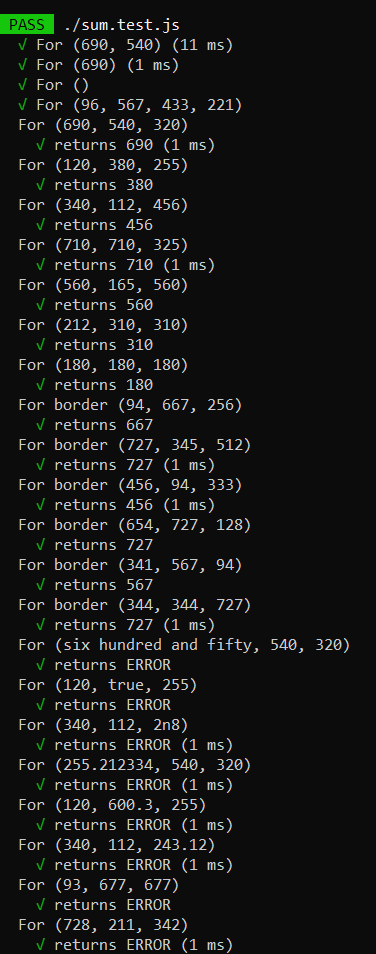


Рис. 2 — Пройденное тестирование

# Заключение

При выполнении лабораторной работы были получены навыки в базовом юнит-тестировании, подборе тест кейсов и анализе покрытия тестов. Была разработана корректно работающая программа, решающая поставленную задачу, а также были использованы средства автоматизации языка JavaScript (Jest).

ПРИЛОЖЕНИЕ А

function sum(a, b, c, ...rest) {

    if (typeof a === 'undefined' || typeof b === 'undefined' || typeof c === 'undefined')

        throw new Error('Недостаточное количество входных данных');

    if (rest.length != 0)

        throw new Error('Избыточное количество входных данных');

    if (typeof a != 'number' || typeof b != 'number' || typeof c != 'number')

        throw new Error('Значения не являются числовыми');

    if (a - Math.trunc(a) != 0 || b - Math.trunc(b) != 0 || c - Math.trunc(c) != 0)

        throw new Error('Значения не являются целочисленными');

    if (a < 94 || b < 94 || c < 94 || a > 727 || b > 727 || c > 727)

        throw new Error('Значения за пределами допустимого диапазона');

    if (a >= b && a>= c) { return a; }

    else if (b >= a && b >= c) { return b;}

    else {return c;}

  }

  module.exports = sum;

const sum = require('./sum');

describe.each([

    [690, 540, 320, 690],

    [120, 380, 255, 380],

    [340, 112, 456, 456],

    [710, 710, 325, 710],

    [560, 165, 560, 560],

    [212, 310, 310, 310],

    [180, 180, 180, 180]

  ])('For (%i, %i, %i)', (a, b, c, expected) => {

    test(`returns ${expected}`, () => {

      expect(sum(a, b, c)).toBe(expected);

    });

  });

describe.each([

    [94, 667, 256,  667],

    [727, 345, 512, 727],

    [456, 94, 333,  456],

    [654, 727, 128, 727],

    [341, 567, 94,  567],

    [344, 344, 727, 727],

  ])('For border (%i, %i, %i)', (a, b, c, expected) => {

    test(`returns ${expected}`, () => {

      expect(sum(a, b, c)).toBe(expected);

    });

  });

test ('For (690, 540)', () => {

    expect(() => sum(690, 540)).toThrow('Недостаточное количество входных данных');

});

test ('For (690)', () => {

  expect(() => sum(690)).toThrow('Недостаточное количество входных данных');

});

test ('For ()', () => {

  expect(() => sum()).toThrow('Недостаточное количество входных данных');

});

test ('For (96, 567, 433, 221)', () => {

  expect(() => sum(96, 567, 433, 221)).toThrow('Избыточное количество входных данных');

});

describe.each([

    {a:'six hundred and fifty', b:540, c:320},

    {a:120, b:true, c:255},

    {a:340, b:112, c:'2n8'},

  ])('For ($a, $b, $c)', ({a,b,c}) => {

    test(`returns ERROR`, () => {

        expect(() => sum(a, b, c)).toThrow('Значения не являются числовыми');

    });

  });

  describe.each([

    {a:255.212334, b:540, c:320},

    {a:120, b:600.3, c:255},

    {a:340, b:112, c:243.12},

  ])('For ($a, $b, $c)', ({a,b,c}) => {

    test(`returns ERROR`, () => {

        expect(() => sum(a, b, c)).toThrow('Значения не являются целочисленными');

    });

  });

describe.each([

    [93, 677, 677],

    [728, 211, 342],

    [444, 93, 432],

    [324, 728, 122],

    [231, 133, 93],

    [233, 655, 728]

  ])('For (%i, %i, %i)', (a, b, c) => {

    test(`returns ERROR`, () => {

        expect(() => sum(a, b, c)).toThrow('Значения за пределами допустимого диапазона');

    });

  });