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 №7**

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

Выполнили

студенты группы КТбо3-4 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. О. Локота

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. С. Селезнева

Принял

ассистент кафедры МОП ЭВМ А. А. Жиглатый

**СОДЕРЖАНИЕ**

[**ЦЕЛЬ РАБОТЫ 3**](#_Toc137065146)

[**1 Выбор задачи 5**](#_Toc137065147)

[**2 Постановка задачи 5**](#_Toc137065148)

[**2.1 Формулировка условия 5**](#_Toc137065149)

[**2.2 Входные данные 5**](#_Toc137065150)

[**2.3 Выходные данные 5**](#_Toc137065151)

[**2.4 Обработка исключительных ситуаций 5**](#_Toc137065152)

[**2.5 Примеры входных данных 6**](#_Toc137065153)

[**3 Алгоритм решения поставленной задачи 6**](#_Toc137065154)

[**4 Тестовый набор 7**](#_Toc137065155)

[**5 Юнит-тестирование 11**](#_Toc137065156)

[**6 Баг-репорты и исправление дефектов 12**](#_Toc137065157)

[**7 Итоговая функция и заключительный тест 14**](#_Toc137065158)

[**ЗАКЛЮЧЕНИЕ 16**](#_Toc137065159)

[**ЛИСТИНГ ПРОГРАММЫ 17**](#_Toc137065160)

## ЦЕЛЬ РАБОТЫ

**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.

1. Выбор задачи

В качестве прототипа задачи для проведения unit-тестирования было выбрано одно из заданий архива сайта acmp.ru под названием «Сбор земляники» (№ 755).

1. Постановка задачи
   1. Формулировка условия

Маша и Миша собирали землянику. Маше удалось сорвать X ягод, а Мише – Y ягод. Поскольку ягода была очень вкусной, то ребята могли какую-то часть ягод съесть. По нашим подсчетам вместе они съели Z ягод.

Числа X, Y и Z содержатся в строке А, в виде натуральных чисел, записанных в десятичном представлении без ведущих нулей, разделенных пробелами.

Необходимо сформировать строку B, содержащую количество собранных ягод в виде натурального числа, записанного в десятичном представлении без ведущих нулей.

* 1. Входные данные

Допустимыми входными данными является строка A, состоящая из трех натуральных десятичных цифр без ведущих нулей X, Y, Z разделенных пробелом при условиях:

Формат текстового представления: в единственной строке строка А.

* 1. Выходные данные

Строка В, с помощью которой выводится натуральное число K.

Формат текстового представления: в единственной строке строка B.

* 1. Обработка исключительных ситуаций

Если количество элементов во входной строке не соответствует условию, то функция должна сгенерировать и вывести стандартное исключение типа Error со строковым сообщением «Количество аргументов не соответствует условию».

Если хотя бы один элемент входной строки не является числом, то функция должна сгенерировать и вывести стандартное исключение типа Error со строковым сообщением «Аргумент не является числом».

Если хотя бы один элемент из входной строки не является целочисленным, то функция должна сгенерировать и вывести стандартное исключение типа Error со строковым сообщением «Аргумент не является целочисленным».

Если хотя бы один элемент входной строки, являющийся целочисленным, не входит в заданный диапазон, то функция должна сгенерировать и вывести стандартное исключение типа Error со строковым сообщением «Аргумент выходит за границы допустимых значений».

Если любой из элементов строки содержит ведущие нули, то функция должна сгенерировать и вывести стандартное исключение типа Error с сообщением «Элемент(-ы) содержат ведущие нули».

Если в строке передается значение Z и неравенство неверно, то функция должна сгенерировать и вывести стандартное исключение типа Error со строковым сообщением «Третий аргумент больше, чем сумма двух предыдущих».

* 1. Примеры входных данных

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 2 1 | 4 |
| 12 13 5 | 20 |

1. Алгоритм решения поставленной задачи

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

function berries(str){

      const param = String(str).split(' ');

      if (param.length != 3){

          throw new Error('Количество аргументов не соответствует условию');

      }

  for (let param of params) {

    if (param.indexOf('0') == 0 && param.length > 1) {

    throw new Error('Элемент(-ы) содержат ведущие нули');}

  }

      const [x, y, z] = [Number(param[0]), Number(param[1]),Number(param[2])];

      if (isNaN(x) || isNaN(y) || isNaN(z)){

          throw new Error('Аргумент не является числом');

        }

      if (!Number.isInteger(x) || !Number.isInteger(y) || !Number.isInteger(z)){

          throw new Error('Аргумент не является целочисленным');

        }

      if((x<0 || x>1000) || (z<0 || z>2000)){

          throw new Error('Аргумент выходит за границы допустимых значений');

        }

    const k = (x + y) - z;

    return k.toString();

}

  module.exports = berries

Листинг 1 – Алгоритм неполного решения

1. Тестовый набор

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

1. Позитивные тесты: 1 тест, когда входные данные корректны и можно посчитать результат K.

2. Тесты граничных значений: 6 тестов на границу интервала X, Y, Z, причем при правой границе третьего аргумента Z, аргументы X и Y также принимают значение правой границы интервала.

3. Негативные тесты: 18 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 переменными. 6 тестов на проверку X, Y, Z на выход за правую и левую границу допустимого интервала. 1 тест на Z большее X+Y, 1 тест на проверку ведущих нулей.

|  |  |  |  |
| --- | --- | --- | --- |
| № теста | Входные данные | Ожидаемый вывод | Комментарий |
| 1 | ‘12 13 5’ | ‘20’ | Позитивный |
| 2 | ‘0 4 1’ | ‘3’ | Позитивный, левая граница первого аргумента |
| 3 | ‘1000 5 100’ | ‘905’ | Позитивный, правая граница первого аргумента |
| 4 | ‘40 0 35’ | ‘5’ | Позитивный, левая граница второго аргумента |
| 5 | ‘70 1000 350’ | ‘720’ | Позитивный, правая граница второго аргумента |
| 6 | ‘34 67 0’ | ‘101’ | Позитивный, левая граница третьего аргумента |
| 7 | ‘1000 1000 2000’ | ‘0’ | Позитивный, правая граница третьего аргумента |
| 8 | ‘12 45 ‘ | Исключение «Количество аргументов не соответствует условию» | Негативный, недостаточно аргументов |
| 9 | ‘678’ | Исключение «Количество аргументов не соответствует условию» | Негативный, недостаточно аргументов |
| 10 | ‘’ | Исключение «Количество аргументов не соответствует условию» | Негативный, недостаточно аргументов |
| 11 | ‘234 456 34 456’ | Исключение «Количество аргументов не соответствует условию» | Негативный, избыточное количество переменных |
| 12 | ‘c 456 45’ | Исключение «Аргумент не является числом» | Негативный, первый аргумент не является числом |
| 13 | ‘125 true 45’ | Исключение «Аргумент не является числом» | Негативный, второй аргумент не является числом |
| 14 | ‘123 765 34m5’ | Исключение «Аргумент не является числом» | Негативный, третий аргумент не является числом |
| 15 | ‘12.4 45 50’ | Исключение «Аргумент не является целочисленным» | Негативный, первый аргумент не является целым числом |
| 16 | ‘564 345.23 400’ | Исключение «Аргумент не является целочисленным» | Негативный, второй аргумент не является целым числом |
| 17 | ‘73 409 76.9’ | Исключение «Аргумент не является целочисленным» | Негативный, третий аргумент не является целым числом |
| 18 | ‘-1 234 45’ | Исключение «Аргумент выходит за границы допустимых значений» | Негативный, выход за левую границу первого аргумента |
| 19 | ‘1001 344 500’ | Исключение «Аргумент выходит за границы допустимых значений» | Негативный, выход за правую границу первого аргумента |
| 20 | ‘234 -1 126’ | Исключение «Аргумент выходит за границы допустимых значений» | Негативный, выход за левую границу второго аргумента |
| 21 | ‘563 1001 1980’ | Исключение «Аргумент выходит за границы допустимых значений» | Негативный, выход за правую границу второго аргумента |
| 22 | ‘456 565 -1’ | Исключение «Аргумент выходит за границы допустимых значений» | Негативный, выход за левую границу третьего аргумента |
| 23 | ‘654 65 2001’ | Исключение «Аргумент выходит за границы допустимых значений» | Негативный, выход за правую границу третьего аргумента |
| 24 | ‘673 892 1999’ | Исключение «Третий аргумент больше, чем сумма двух предыдущих» | Негативный, третий аргумент превышает сумму двух других |
| 25 | "010 10 10" | Исключение «Элемент(-ы) содержат ведущие нули» | Негативный, неверный формат числа |

1. 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:

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 1 – Результаты тестирования

Во время выполнения тестов были найдены некоторые ошибки, которые были расписаны и исправлены в следующем разделе.

1. Баг-репорты и исправление дефектов

Дефект №1

|  |  |
| --- | --- |
| Название дефекта | Некорректная обработка случая, когда значение второго аргумента уходит за пределы левой границы |
| Приоритет дефекта | Высокий |
| Серьезность дефекта | Критический |
| Версия системы | ОС Windows 11, NodeJS v18.16.0 |
| Инструкция по воспроизведению | Входные данные: ‘234 -1 126’ |
| Фактический результат | Received function did not throw |
| Ожидаемый результат | Исключение «Аргумент выходит за границы допустимых значений» |
| Вид, тип, область тестирования | Функциональное, модульное тестирование |
| Вложения |  |

Дефект №2

|  |  |
| --- | --- |
| Название дефекта | Некорректная обработка случая, когда значение второго аргумента уходит за пределы правой границы |
| Приоритет дефекта | Высокий |
| Серьезность дефекта | Критический |
| Версия системы | ОС Windows 11, NodeJS v18.16.0 |
| Инструкция по воспроизведению | Входные данные: ‘563 1001 1980’ |
| Фактический результат | Received function did not throw |
| Ожидаемый результат | Исключение «Аргумент выходит за границы допустимых значений» |
| Вид, тип, область тестирования | Функциональное, модульное тестирование |
| Вложения |  |

Дефекты №1 и №2 были исправлены с помощью кода на рисунке 2.

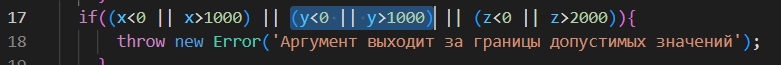


Рисунок 2 – Исправление дефектов №1 и №2

Дефект №3

|  |  |
| --- | --- |
| Название дефекта | Некорректная обработка случая, когда третий параметр больше суммы двух других |
| Приоритет дефекта | Высокий |
| Серьезность дефекта | Критический |
| Версия системы | ОС Windows 11, NodeJS v18.16.0 |
| Инструкция по воспроизведению | Входные данные: ‘673 892 1999’ |
| Фактический результат | Received function did not throw |
| Ожидаемый результат | Исключение «Третий аргумент больше, чем сумма двух предыдущих» |
| Вид, тип, область тестирования | Функциональное, модульное тестирование |
| Вложения |  |

Дефект №3 был исправлен с помощью кода на рисунке 3.

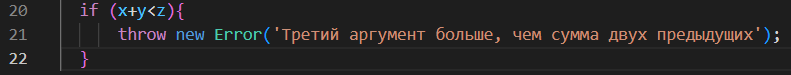


Рисунок 3 – Исправление дефекта №3

1. Итоговая функция и заключительный тест

Функция со всеми корректировками:

function berries(str){

  const param = String(str).split(' ');

  if (param.length != 3){

      throw new Error('Количество аргументов не соответствует условию');

  }

  for (let param of params) {

    if (param.indexOf('0') == 0 && param.length > 1) {

    throw new Error('Элемент(-ы) содержат ведущие нули');

    }

  }

  const [x, y, z] = [Number(param[0]), Number(param[1]),Number(param[2])];

  if (isNaN(x) || isNaN(y) || isNaN(z)){

      throw new Error('Аргумент не является числом');

    }

  if (!Number.isInteger(x) || !Number.isInteger(y) || !Number.isInteger(z)){

      throw new Error('Аргумент не является целочисленным');

    }

  if((x<0 || x>1000) || (y<0 || y>1000)||(z<0 || z>2000)){

      throw new Error('Аргумент выходит за границы допустимых значений');

    }

  if (x+y<z){

      throw new Error('Третий аргумент больше, чем сумма двух предыдущих');

  }

const k = (x + y) - z;

return k.toString();

}

module.exports = berries

Листинг 2 – Исправленный код

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 3 – Успешное прохождение всех тестов

## 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).

## ЛИСТИНГ ПРОГРАММЫ

Файл **index.js**

function berries(str){

  const param = String(str).split(' ');

  if (param.length != 3){

      throw new Error('Количество аргументов не соответствует условию');

  }

  for (let param of params) {

    if (param.indexOf('0') == 0 && param.length > 1) {

    throw new Error('Элемент(-ы) содержат ведущие нули');

    }

  }

  const [x, y, z] = [Number(param[0]), Number(param[1]),Number(param[2])];

  if (isNaN(x) || isNaN(y) || isNaN(z)){

      throw new Error('Аргумент не является числом');

    }

  if (!Number.isInteger(x) || !Number.isInteger(y) || !Number.isInteger(z)){

      throw new Error('Аргумент не является целочисленным');

    }

  if((x<0 || x>1000) || (y<0 || y>1000)||(z<0 || z>2000)){

      throw new Error('Аргумент выходит за границы допустимых значений');

    }

  if (x+y<z){

      throw new Error('Третий аргумент больше, чем сумма двух предыдущих');

  }

const k = (x + y) - z;

return k.toString();

}

module.exports = berries

Файл **index.test.js**

const berries = require('./index');

describe('calculateBerries', () => {

    test('positive', ()=> {expect(berries('12 13 5')).toBe('20');});

    test('border values', ()=> {

        expect(berries('0 4 1')).toBe('3');

        expect(berries('1000 5 100')).toBe('905');

        expect(berries('40 0 35')).toBe('5');

        expect(berries('70 1000 350')).toBe('720');

        expect(berries('34 67 0')).toBe('101');

        expect(berries('1000 1000 2000')).toBe('0');

    });

    test('number of arguments', ()=> {

        expect(() => berries('12 44')).toThrow('Количество аргументов не соответствует условию');

        expect(() => berries('678')).toThrow('Количество аргументов не соответствует условию');

        expect(() => berries('')).toThrow('Количество аргументов не соответствует условию');

        expect(() => berries('234 456 34 456')).toThrow('Количество аргументов не соответствует условию');

    });

    test('not an integer', ()=> {

        expect(() => berries('14.4 45 50')).toThrow('Аргумент не является целочисленным');

        expect(() => berries('564 345.23 400')).toThrow('Аргумент не является целочисленным');

        expect(() => berries('73 409 76.9')).toThrow('Аргумент не является целочисленным');

    });

    test('not a number', ()=> {

        expect(() => berries('c 456 45')).toThrow('Аргумент не является числом');

        expect(() => berries('125 true 45')).toThrow('Аргумент не является числом');

        expect(() => berries('123 765 34m5')).toThrow('Аргумент не является числом');

     });

    test('abroad', ()=> {

        expect(() => berries('-1 234 45')).toThrow('Аргумент выходит за границы допустимых значений');

        expect(() => berries('1001 344 500')).toThrow('Аргумент выходит за границы допустимых значений');

        expect(() => berries('234 -1 126')).toThrow('Аргумент выходит за границы допустимых значений');

        expect(() => berries('563 1001 1980')).toThrow('Аргумент выходит за границы допустимых значений');

        expect(() => berries('456 565 -1')).toThrow('Аргумент выходит за границы допустимых значений');

        expect(() => berries('654 65 2001')).toThrow('Аргумент выходит за границы допустимых значений');

     });

    test('the third argument bigger than X+Y', ()=> {

        expect(() => berries('673 892 1999')).toThrow('Третий аргумент больше, чем сумма двух предыдущих');

    });

  test('Ved Null', () => {

        expect(() => berries("010 10 10")).toThrow("Элемент(-ы) содержат ведущие нули");

    });

});