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 №8**

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

Выполнили

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. С. Селезнёва

Принял

ассистент кафедры МОП ЭВМ А. А. Жиглатый

**СОДЕРЖАНИЕ**

[**ЦЕЛЬ РАБОТЫ 3**](#_Toc137001318)

[**1 Выбор задачи 5**](#_Toc137001319)

[**2 Постановка задачи 5**](#_Toc137001320)

[**2.1 Формулировка условия 5**](#_Toc137001321)

[**2.2 Входные данные 5**](#_Toc137001322)

[**2.3 Выходные данные 5**](#_Toc137001323)

[**2.4 Обработка исключительных ситуаций 5**](#_Toc137001324)

[**2.5 Примеры входных данных 6**](#_Toc137001325)

[**3 Алгоритм решения поставленной задачи 6**](#_Toc137001326)

[**4 Тестовый набор 6**](#_Toc137001327)

[**5 Юнит-тестирование 8**](#_Toc137001328)

[**6 Баг-репорты и исправление дефектов 10**](#_Toc137001329)

[**7 Итоговая функция и заключительный тест 12**](#_Toc137001330)

[**ЗАКЛЮЧЕНИЕ 14**](#_Toc137001331)

[**ЛИСТИНГ ПРОГРАММЫ 15**](#_Toc137001332)

## 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 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.

Необходимо определить количество K собранных ягод.

* 1. Входные данные

Допустимыми входными данными является три натуральных числа X, Y, Z, при условиях:

Формат текстового представления: в единственной строке через пробел три числа X, Y, Z.

* 1. Выходные данные

Натуральное число K.

Формат текстового представления: в единственной строке одно число K.

* 1. Обработка исключительных ситуаций

Если входных параметров меньше трех, вывести -1.

Если входных параметров больше трех, вывести -2.

Если хотя бы одно входное значение не является целочисленным или числом, вывести -3.

Если хотя бы одно входное значение, являющееся целочисленным, не входит в заданный диапазон, вывести -4.

Если неравенство неверно, вывести -5.

* 1. Примеры входных данных

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 2 1 | 4 |
| 12 13 5 | 20 |
| 2 5 9 | -5 |

1. Алгоритм решения поставленной задачи

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

function berries(x, y, z, ...rest){

    if (x===undefined || y===undefined || z==undefined){

        return -1;

    }

    else if(rest.length != 0){

        return -2;

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

        return -3;

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

        return -4;

      } else { return x+y-z }

}

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: 17 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 X, Y, Z на выход за правую и левую границу допустимого интервала. 1 тест на Z большее X+Y.

|  |  |  |  |
| --- | --- | --- | --- |
| № 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 | -1 | Негативный, недостаточно аргументов |
| 9 | 678 | -1 | Негативный, недостаточно аргументов |
| 10 | Нет данных | -1 | Негативный, недостаточно аргументов |
| 11 | 234 456 34 456 | -2 | Негативный, избыточное количество переменных |
| 12 | ‘c’ 456 45 | -3 | Негативный, первый аргумент не является числом |
| 13 | 125 true 45 | -3 | Негативный, второй аргумент не является числом |
| 14 | 123 765 ‘345’ | -3 | Негативный, третий аргумент не является числом |
| 15 | 12.4 45 50 | -3 | Негативный, первый аргумент не является целым числом |
| 16 | 564 345.23 400 | -3 | Негативный, второй аргумент не является целым числом |
| 17 | 73 409 76.9 | -3 | Негативный, третий аргумент не является целым числом |
| 18 | -23 234 45 | -4 | Негативный, выход за левую границу первого аргумента |
| 19 | 1456 344 500 | -4 | Негативный, выход за правую границу первого аргумента |
| 20 | 234 -56 126 | -4 | Негативный, выход за левую границу второго аргумента |
| 21 | 563 4506 1980 | -4 | Негативный, выход за правую границу второго аргумента |
| 22 | 456 565 -4 | -4 | Негативный, выход за левую границу третьего аргумента |
| 23 | 654 65 3059 | -4 | Негативный, выход за правую границу третьего аргумента |
| 24 | 673 892 1999 | -5 | Негативный, третий аргумент превышает сумму двух других |

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, Мультимедийное 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 -56 126 |
| Фактический результат | 52 |
| Ожидаемый результат | -4 |
| Вид, тип, область тестирования | Функциональное, модульное тестирование |
| Вложения | Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей  Автоматически созданное описание |

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: 1456 344 500 |
| Фактический результат | 3089 |
| Ожидаемый результат | -4 |
| Вид, тип, область тестирования | Функциональное, модульное тестирование |
| Вложения | Изображение выглядит как текст, Шрифт, снимок экрана  Автоматически созданное описание |

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

Изображение выглядит как текст, программное обеспечение, снимок экрана, дисплей

Автоматически созданное описание

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 | -434 |
| Ожидаемый результат | -5 |
| Вид, тип, область тестирования | Функциональное, модульное тестирование |
| Вложения | Изображение выглядит как текст, снимок экрана, Шрифт  Автоматически созданное описание |

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

Изображение выглядит как Шрифт, текст, рукописный текст, снимок экрана

Автоматически созданное описание

Рисунок 3 – Исправление дефектов №2 и №3

1. Итоговая функция и заключительный тест

Функция со всеми корректировками:

function berries(x, y, z, ...rest){

    if (x===undefined || y===undefined || z==undefined){

        return -1;

    }

    else if(rest.length != 0){

        return -2;

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

        return -3;

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

        return -4;

      } else if ((x+y-z)<0){

        return -5;

    } else { return x+y-z }

}

module.exports = berries

Листинг 2 – Исправленный код

Изображение выглядит как текст, снимок экрана, Шрифт

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(x, y, z, ...rest){

  if (x===undefined || y===undefined || z==undefined){

      return -1;

  }

  else if(rest.length != 0){

      return -2;

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

      return -3;

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

      return -4;

    } else if ((x+y-z)<0){

      return -5;

  } else { return x+y-z }

}

module.exports = berries

Файл **index.test.js**

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

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

test('border value - first left', ()=> {expect(berries(0, 4, 1)).toBe(3);});

test('border values - first right', ()=> {expect(berries(1000, 5, 100)).toBe(905);});

test('border values - second left', ()=> {expect(berries(40, 0, 35)).toBe(5);});

test('border values - second right', ()=> {expect(berries(70, 1000, 350)).toBe(720);});

test('border values - third left', ()=> {expect(berries(34, 67, 0)).toBe(101);});

test('border values - third right', ()=> {expect(berries(1000, 1000, 2000)).toBe(0);});

test('negative - two arguments', ()=> {expect(berries(12, 44)).toBe(-1);});

test('negative - one arguments', ()=> {expect(berries(678)).toBe(-1);});

test('negative - zero arguments', ()=> {expect(berries()).toBe(-1);});

test('negative - so much arguments', ()=> {expect(berries(234, 456, 34, 456)).toBe(-2);});

test('negative - first not an integer', ()=> {expect(berries(14.4, 45, 50)).toBe(-3);});

test('negative - second not an integer', ()=> {expect(berries(564, 345.23, 400)).toBe(-3);});

test('negative - third not an integer', ()=> {expect(berries(73, 409, 76.9)).toBe(-3);});

test('negative - first not a number', ()=> {expect(berries('c', 456, 45)).toBe(-3);});

test('negative - second not a number', ()=> {expect(berries(125, true, 45)).toBe(-3);});

test('negative - third not a number', ()=> {expect(berries(123, 765, '34m5')).toBe(-3);});

test('negative - going beyond the left border of the first number', ()=> {expect(berries(-23, 234, 45)).toBe(-4);});

test('negative - going beyond the right border of the first number', ()=> {expect(berries(1456, 344, 500)).toBe(-4);});

test('negative - going beyond the left border of the second number', ()=> {expect(berries(234, -56, 126)).toBe(-4);});

test('negative - going beyond the right border of the second number', ()=> {expect(berries(563, 4506, 1980)).toBe(-4);});

test('negative - going beyond the left border of the third number', ()=> {expect(berries(456, 565, -4)).toBe(-4);});

test('negative - going beyond the right border of the third number', ()=> {expect(berries(654, 65, 3059)).toBe(-4);});

test('negative - the third argument bigger than X+Y', ()=> {expect(berries(673, 892, 1999)).toBe(-5);});