**Финансово-экономический колледж**

**РГЭУ (РИНХ)**



**Практика**

Студент: ИС-202

**Орлов Илья Вадимович**

Преподаватель:

**Журавлёв Д.Г.**

**Ростов-на-Дону**

**Теоретическая часть**

**1. Корректность программы**

**Определение:**

Программа называется корректной, если она удовлетворяет своей спецификации, то есть: для всех допустимых входных данных (удовлетворяющих предусловиям) программа завершается и выдает результат, соответствующий постусловиям, если предусловия нарушены, поведение программы может быть любым (например, ошибка, неопределенный результат).

**2. Спецификации корректности**

Спецификация — формальное описание того, что должна делать программа (без указания как). Без нее нельзя определить корректность.

Виды спецификаций:Функциональные требования (например, "функция возвращает отсортированный массив"). Предусловия (условия, которые должны быть истинны перед вызовом: numbers != null). Постусловия (гарантии после выполнения: "результат — среднее положительных чисел в массиве").

**3. Методы проверки корректности**

Тестирование

Суть: Запуск программы на конкретных входных данных и проверка соответствия ожидаемому результату.

**Сильные стороны:**

Простота реализации для небольших случаев, позволяет находить ошибки в реальных сценариях.

**Слабые стороны:**

Проверяет только отдельные случаи, а не все возможные, не гарантирует отсутствие ошибок.

**Тест-кейс:** Набор входных данных, шагов выполнения и ожидаемого результата (например, numbers = [1, -2, 3] → ожидаемый результат 2).

Статический анализ

**Суть:** Анализ кода без его выполнения для выявления потенциальных ошибок.

**Что выявляет:**

Синтаксические ошибки, нарушения стиля кода, потенциальные утечки памяти (например, в C/C++).

**Инструменты:**

PyLint (Python), ESLint (JavaScript), SonarQube, пример ошибки: Обнаружение недостижимого кода или использования неинициализированной переменной.

**Формальные методы**

Идея: Математическое доказательство того, что программа удовлетворяет спецификации.

Инвариант цикла: Условие, которое:

Истинно перед началом цикла.

Сохраняется после каждой итерации.

Позволяет доказать корректность работы цикла.

**Практическая часть**

**1.Спецификация**

Предусловия:

*numbers* должен быть списком (или итерируемым объектом), элементы которого можно сравнивать с нулём (например, числа).

Элементы списка numbers должны поддерживать операции сложения и деления (т.е. быть числового типа).

**Постусловие:**

Функция возвращает среднее арифметическое всех положительных элементов списка numbers.

Если положительных элементов нет, функция возвращает 0.

**Инвариант цикла:**

На каждой итерации цикла:

*total* равно сумме всех положительных элементов, обработанных на данный момент.

*count* равно количеству всех положительных элементов, обработанных на данный момент.

**2.Ручное тестирование**

Тест 1: Обычный случай (есть положительные числа)

Входные данные: [1, -2, 3, -4, 5]

Ожидаемый результат: (1 + 3 + 5) / 3 = 3

Фактический результат: 3 (корректно).

Тест 2: Граничный случай (нет положительных чисел)

Входные данные: [-1, -2, -3]

Ожидаемый результат: 0

Фактический результат: 0 (корректно).

Тест 3: Нарушение предусловия (некорректные входные данные)

Входные данные: ["a", "b", "c"] (элементы не числа)

Ожидаемый результат: Ошибка (например, TypeError при сравнении с нулём или сложении).

Фактический результат: Ошибка TypeError (корректно, но не обработана в функции).

**3. Анализ корректности и ошибок**

Строка A (average = total / count):

Деление на ноль гарантированно не произойдёт, потому что перед этим выполняется проверка if count > 0. Если count == 0, выполнится ветка else, где average присваивается 0.

Обработка пустого списка:

Пустой список numbers = [] корректно обрабатывается: цикл не выполняется, count остаётся 0, функция возвращает 0. Это соответствует постусловию.

Потенциальные ошибки при нарушении предусловия:

Если numbers не является списком (например, строка или None), возникнет ошибка (например, TypeError при итерации).

Если элементы не числа, возникнет ошибка при сравнении или сложении (например, TypeError).

**4. Выводы**

**Корректность функции:**

Функция корректна относительно своей очевидной спецификации: она правильно вычисляет среднее положительных чисел и возвращает 0, если таких чисел нет. Тесты подтверждают это.

Метод проверки для обнаружения ошибки в строке A:

Наиболее эффективным методом был бы статический анализ (например, с помощью инструментов вроде mypy), который мог бы обнаружить потенциальное деление на ноль до выполнения кода. Тестирование также могло бы выявить эту ошибку, но только при наличии теста с count == 0. Формальная верификация была бы избыточна для такой простой функции.