1. Корректность программы

Определение корректности:

Программа называется **корректной**, если она удовлетворяет своей спецификации, то есть для всех допустимых входных данных она завершается и выдает результат, соответствующий ожидаемому поведению.

Отличие корректности от надежности:

**Корректность** — соответствие программы своей спецификации.

**Надежность** — способность программы работать без сбоев в различных условиях (включая некорректные входные данные, аппаратные сбои и т. д.).

Пример надежной, но некорректной программы:

Программа, которая всегда возвращает `0` при любом входе, является **надежной** (не падает), **но некорректной**, если по спецификации она должна вычислять что-то осмысленное.

2. Спецификации корректности

**Спецификация программы** — формальное описание того, что программа должна делать (ее требования и ожидаемое поведение). Без спецификации нельзя определить, корректна ли программа.

Виды спецификаций:

**Функциональные требования** — что программа должна делать (например, "функция должна возвращать среднее арифметическое положительных чисел").

**Предусловия** — условия, которые должны выполняться перед вызовом функции (например, "список чисел не пуст").

**Постусловия** — что функция гарантирует после выполнения (например, "возвращаемое значение ≥ 0").

**Инварианты** — условия, которые сохраняются на протяжении всей работы (например, "размер буфера никогда не превышает заданный лимит").

Пример для calculate\_positive\_average(numbers):

**Предусловие**:

Numbers не пуст (len(numbers) > 0).

В numbers есть хотя бы одно положительное число.

**Постусловие:**

Возвращает среднее арифметическое всех положительных чисел в numbers.

Результат ≥ 0.

3. Методы проверки корректности

Тестирование

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

**Сильные стороны:**

Простота реализации.

Позволяет находить ошибки в конкретных сценариях.

**Слабые стороны:**

Не гарантирует отсутствие ошибок на всех входах.

Требует написания множества тест-кейсов.

**Тест-кейс** — конкретный пример входных данных и ожидаемого результата (например, для calculate\_positive\_average([1, 2, 3]) ожидается 2.0).

Статический анализ

Суть: Анализ кода без его выполнения (проверка синтаксиса, типов, потенциальных ошибок).

**Примеры ошибок:**

Недостижимый код.

Использование неинициализированной переменной.

**Формальные методы (доказательство корректности)**

**Суть:** Математическое доказательство, что программа удовлетворяет спецификации.

**Инвариант цикла:** Условие, которое верно перед каждой итерацией цикла и после него.

**Пример:** В цикле суммирования sum = sum + a[i] инвариант — sum равно сумме всех пройденных элементов.

**Сравнение методов**

**Для небольшой функции**: лучше подходит **тестирование + статический анализ (**быстро и достаточно надежно).

**Для большой системы**: Комбинация **тестирования, статического анализа и формальных методов**.

**Практическая часть**

**(для calculate\_positive\_average)**

Спецификация:

**Предусловие:**

**-**Список должен иметь внутри себя данные

-Данные должны быть строго типа integer

-integer не может быть отрицательным

**Постусловие:**

**-В случае успешного выполнения возвращает значение average(total/count)**

**-В случае некорректного выполнения возвращает значение average(0)**

**Инвариант:**

**-Цикл не будет прерван пока не проитерировал весь список**

**Test case**

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Ожидаемый** | **Фактический** | **Прошел?** |
| **1,2,3** | **2.0** | **2.0** | **Да** |
| **0,0,3** | **3.0** | **3.0** | **Да** |
| **“a”, 2, 3** | **0** | **TypeError** | **Нет** |

**Вопросы анализа**

1. Строка average = total / count

Деление на ноль невозможно, потому что \*\*предусловие\*\* функции требует, чтобы:

numbers не был пустым (len(numbers) > 0),

в numbers было хотя бы одно положительное число.

Таким образом, count (количество положительных чисел) всегда будет ≥ 1, и деление на ноль не произойдет.

Что произойдет, если предусловие нарушено?

Если в списке нет положительных чисел (например, numbers = [-1, -2, -3), то count = 0, и произойдет деление на ноль → ошибка ZeroDivisionError.

2. Обработка пустого списка (numbers = [])

Нет, функция не обрабатывает этот случай, если не проверяет предусловие явно.

Соответствует ли поведение спецификации?

Постусловие предполагает, что функция возвращает среднее положительных чисел (т. е. результат ≥ 0).

3. Ошибки при нарушении предусловий

Примеры потенциальных сбоев:

|  |  |
| --- | --- |
| Numbers = [] | ZeroDivisionError |
| Numbers = 1 | TypeError(При итерации) |