Практическая работа №1

Тестирование программных продуктов

Цели: ознакомление с методами тестирования программного продукта

Теоретические сведения:

Тестирование является завершающим и важным этапом разработки ПО, превосходящим простую отладку (обнаружение и исправление ошибок).

Даже полностью отлаженная программа может оказаться непригодной к использованию, если не соответствует потребительским требованиям. Проверка программы только на контрольных примерах не является полноценным тестированием, поскольку не может охватить все возможные условия эксплуатации.

Согласно ГОСТ 19.004-80, испытание - это установление соответствия программы заданным в техническом задании требованиям, гарантирующим ее пригодность к использованию.

Существуют разные методы тестирования, определяющие порядок сборки и проверки программы: восходящее (снизу-вверх), нисходящее (сверху-вниз, с изолированным тестированием головного модуля), «большого скачка» (автономное тестирование модулей с последующей интеграцией) и «сэндвича» (комбинированный подход, объединяющий восходящее и нисходящее тестирование). Выбор метода зависит от структуры программы.

Практическая часть:

План тестирования программы (нисходящее тестирование)

Цель: Проверить корректность работы программы, начиная с главной функции и постепенно переходя к более низким уровням системы.

Объекты тестирования:

* main(): Основная функция для взаимодействия с пользователем и выполнения операций.
* calculate(operation, num1, num2): Функция для определения арифметической операции над двумя числами.
* add(x, y): Функция, выполняющая сложение двух чисел.
* subtract(x, y): Функция, выполняющая вычитание одного числа из другого.
* multiply(x, y): Функция, выполняющая умножение двух чисел.
* divide(x, y): Функция, выполняющая деление одного числа на

Методология тестирования:

* Нисходящее тестирование: Начинаем с тестирования самой главной функции программы. В случае успешного выполнения программы тестируем более низкоуровневые функции, которые реализуют части логики программы. Основное внимание уделяется взаимодействию между компонентами.
* Черный ящик: Тестирование проводится на основе спецификаций функций, без знания внутреннего кода.
* Анализ граничных значений: Проверка работы функций на граничных значениях входных данных (например, пустой список, минимальные и максимальные значения).

Критерии приемки:

* Главная функция main() должна корректно координировать работу других функций и обеспечивать корректный вывод.
* Функции calculate, add, subtract, multiply, divide и generate должны корректно обрабатывать свои входные данные и возвращать ожидаемые результаты.

Тестовое окружение:

* Python 3.x
* Любая IDE или текстовый редактор для запуска и отладки кода.

Компоненты разработанной программы для ознакомления с методами тестирования представлены на рисунках:

* 1: Главная функция main для запуска всей программы;
* 2: Функция calculate для определения арифметической операции;
* 3: Функция add для сложения чисел;
* 4: Функция subtract для вычитания чисел;
* 5: Функция multiply для умножения чисел;
* 6: Функция divide для деления чисел.

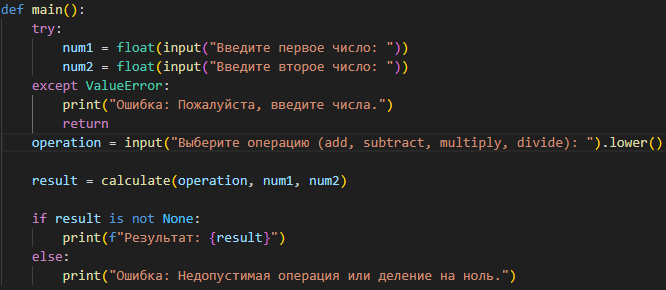


Рисунок 1 – Главная часть программы

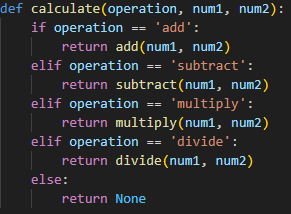


Рисунок 2 – Функция для определения арифметической операции

C:\Users\Student\Pictures\add.PNG

Рисунок 3 – Функция для сложения чисел

C:\Users\Student\Pictures\add.PNG

Рисунок 4 – Функция для вычитания чисел

C:\Users\Student\Pictures\add.PNG

Рисунок 5 – Функция для умножения чисел

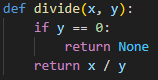


Рисунок 6 – Функция для деления чисел

Ожидаемые результаты тестирования функции main представлены в таблице 1.

Таблица 1 – Тестирование метода main (Рисунок 1)

| Тест (значения для входных данных) | Ожидаемый результат (значения для выходных данных) | Фактический результат (полученные значения выходных данных) | Результат тестирования (успешно/неуспешно) |
| --- | --- | --- | --- |
| Запуск программы (нормальные входные данные) | Вывод на экран числа, полученного после проведения арифметической операции. | 11.0 | Успешно |
| Запуск программы (некорректный ввод чисел) | Ошибка: Пожалуйста, введите числа. | Ошибка: Пожалуйста, введите числа. | Успешно |
| Запуск программы (некорректный ввод операции) | Ошибка: Недопустимая операция или деление на ноль. | Ошибка: Недопустимая операция или деление на ноль. | Успешно |

Ожидаемые результаты тестирования функции calculate представлены в таблице 2.

Таблица 2 – Тестирование метода calculate (Рисунок 2)

| Тест (значения для входных  данных) | Ожидаемый результат (значения для выходных данных) | Фактический результат (полученные значения выходных данных) | Результат тестирования (успешно/неуспешно) |
| --- | --- | --- | --- |
| (6.5, -5, 'add') | 1.5 | 1.5 | Успешно |
| (6.5, -5, 'substract') | 11.5 | 11.5 | Успешно |
| (6.5, -5, 'multiply') | -32.5 | -32.5 | Успешно |
| (6.5, -5, 'divide') | -1.3 | -1.3 | Успешно |
| (6.5, 0, 'divide') | Ошибка: Недопустимая операция или деление на ноль. | Ошибка: Недопустимая операция или деление на ноль. | Успешно |

Ожидаемые результаты тестирования функции add представлены в таблице 3.

Таблица 3 – Тестирование метода add (Рисунок 3)

| Тест (значения для входных  данных) | Ожидаемый результат (значения для выходных данных) | Фактический результат (полученные значения выходных данных) | Результат тестирования (успешно/неуспешно) |
| --- | --- | --- | --- |
| (6, 5) | 11.0 | 11.0 | Успешно |
| (6.5, 5) | 11.5 | 11.5 | Успешно |
| (6, 0) | 6.0 | 6.0 | Успешно |
| (6, -5) | 1.0 | 1.0 | Успешно |

Ожидаемые результаты тестирования функции subtract представлены в таблице 4.

Таблица 4 – Тестирование метода subtract (Рисунок 4)

| Тест (значения для входных  данных) | Ожидаемый результат (значения для выходных данных) | Фактический результат (полученные значения выходных данных) | Результат тестирования (успешно/неуспешно) |
| --- | --- | --- | --- |
| (6, 5) | 1.0 | 1.0 | Успешно |
| (6.5, 5) | 1.5 | 1.5 | Успешно |
| (6, 0) | 6.0 | 6.0 | Успешно |
| (6, -5) | 11.0 | 11.0 | Успешно |

Ожидаемые результаты тестирования функции multiply представлены в таблице 5.

Таблица 5 – Тестирование метода multiply (Рисунок 5)

| Тест (значения для входных  данных) | Ожидаемый результат (значения для выходных данных) | Фактический результат (полученные значения выходных данных) | Результат тестирования (успешно/неуспешно) |
| --- | --- | --- | --- |
| (6, 5) | 30.0 | 30.0 | Успешно |
| (6.5, 5) | 32.5 | 32.5 | Успешно |
| (6, 0) | 0.0 | 0.0 | Успешно |
| (6, -5) | -30.0 | -30.0 | Успешно |

Ожидаемые результаты тестирования функции divide представлены в таблице 6.

Таблица 6 – Тестирование метода divide (Рисунок 6)

| Тест (значения для входных  данных) | Ожидаемый результат (значения для выходных данных) | Фактический результат (полученные значения выходных данных) | Результат тестирования (успешно/неуспешно) |
| --- | --- | --- | --- |
| (6, 5) | 1.2 | 1.2 | Успешно |
| (6.5, 5) | 1.3 | 1.3 | Успешно |
| (6, 0) | Ошибка: Недопустимая операция или деление на ноль. | Ошибка: Недопустимая операция или деление на ноль. | Успешно |
| (6, -5) | -1.2 | -1.2 | Успешно |

Рекомендации для корректировки программы:

* Добавить в функции divide более конкретные исключения. При делении на ноль лучше выбрасывать исключение ZeroDivisionError вместо None. Это позволит вызывающей функции более гибко обрабатывать эту ситуацию (Рисунок 6).

Ответы на контрольные вопросы:

1. Тестирование ПО: Это процесс оценки качества и проверки соответствия программного продукта требованиям. Целью является обнаружение ошибок, дефектов и обеспечение надежности программы.

2. Тестирование vs Отладка: Тестирование - это процесс выявления дефектов и несоответствий. Отладка - это процесс поиска, локализации и исправления обнаруженных дефектов. Тестирование показывает наличие проблемы, отладка её решает.

3. Функциональное тестирование: Это вид тестирования, проверяющий, выполняет ли программное обеспечение заявленные функции правильно. Проверяется ввод, обработка, вывод данных и взаимодействие с другими системами, согласно спецификации.

4. Комплексное тестирование: Это тестирование взаимодействия между отдельными модулями или компонентами системы после модульного тестирования. Выявляет ошибки интеграции.

5. «Черный ящик»: Тестировщик не знает внутреннюю структуру кода. Он разрабатывает тесты, основываясь только на функциональных требованиях, входных и ожидаемых выходных данных. Важно покрытие всех требований и анализ граничных значений.

6. «Белый ящик»: Тестировщик знает внутреннюю структуру кода и разрабатывает тесты для проверки логики, путей выполнения и условий программы. Используются методы покрытия кода (операторов, решений, условий).

7. Модульное тестирование: Это тестирование отдельных, наименьших тестируемых частей программы (модулей), таких как функции или классы, для проверки их корректной работы.

8. Сборка при модульном тестировании: Каждый модуль тестируется изолированно. Для имитации взаимодействия с другими модулями используются «заглушки» (имитируют вызываемые модули) и «драйверы» (вызывают тестируемый модуль). После этого проводится интеграция модулей.

Вывод: в ходе работы проверил корректность работы программы, начиная с главной функции и постепенно переходя к более низким уровням системы с помощью.