Тема: Исследование программного кода на предмет ошибок и отклонения от алгоритма

Формат: Индивидуальный проект

Цель задания:

1. Научиться внимательно анализировать код, сверяя его с описанием алгоритма или требованиями.

2. Выявлять различные типы ошибок: синтаксические, логические, ошибки в условиях, граничные случаи.

3. Научиться проверять корректность кода с помощью тестовых случаев.

4. Развивать критическое мышление при оценке чужого (и своего) кода.

Теоретическая часть

1) Типы ошибок в коде:

Синтаксическая ошибка — нарушение правил языка программирования

Пример — пропущенная скобка

Логическая ошибка — код работает, но выдает неверный результат из-за ошибки в логике

Пример — Использование high = n в бинарном поиске

Ошибка в условии — неправильное условие, приводящее к неверному выполнению кода

Пример — проверка if n<=0, а не if n<0

Ошибка обработки граничных случаев — неучтенные крайние значения

Пример — нет проверки случая n==0, который должен возвращать True

Отклонение от алгоритма (код делает не то, что описано) — код не соответствует описанию

Пример — Отсутствие бинарного поиска для n>1

2) Методы поиска ошибок:

Визуальный анализ кода — построчное сравнение с алгоритмом.

Ручное тестирование — проверка на конкретных примерах.

Сравнение с описанием алгоритма — выявление отклонений.

Контрольные вопросы — Граничные случаи: "Почему n=0 возвращает False?" = Ошибка в условии. Алгоритм — "Почему high=n, если корень не превышает n//2?" = Логическая ошибка.

Практическая часть (Анализ кода)

Задача:

Написать функцию is\_perfect\_square(n), которая возвращает True, если целое число n является точным квадратом (т.е. n = k \* k для некоторого целого k), и False в противном случае.

Алгоритм (Спецификация):

1. Если n < 0, вернуть False (отрицательные числа не могут быть квадратами).

Если n == 0 или n == 1, вернуть True (0\*0=0, 1\*1=1).

2. Для n > 1:

▪ Использовать бинарный поиск в диапазоне от 2 до n // 2 (т.к. корень из n не может быть больше n/2 при n>1).

▪ Найти целое mid такое, что mid \* mid == n. Если найдено, вернуть True.

▪ Если бинарный поиск завершился неудачей, вернуть False

1) Визуальный анализ и сравнение с алгоритмом:

Строка 1-2 — Ошибка в условии. Условие n <= 0 неверно, так как 0 — точный квадрат (0\*0=0).

Устранение — if n < 0.

Строка 6 — Логическая ошибка. high = n вместо high = n // 2. Для n > 2 корень не может быть больше n//2.

Устранение — high = n // 2.

Строка 3-4 — Не учтен случай n == 0.

Устранение — Добавить проверку if n == 0: return True

2) Поиск мнимых "ошибок":

Нет, здесь мнимых ошибок нет

3) Ручное тестирование:

n

Ожидаемый результат

Фактический результат

|  |  |  |  |
| --- | --- | --- | --- |
| n | Ожидаемый результат | Фактический результат | Совпадение |
| -1 | False | False | да |
| 0 | True | False | Нет |
| 1 | True | True | Да |
| 4 | True | True | Да |
| 5 | False | False | Да |
| 15 | False | False | Да |
| 16 | True | True | Да |
| 1000000 | True | True | Да |

4) Исправление кода:

def is\_perfect\_square(n):  
 if n < 0: # Line 1  
 return False # Line 2  
 if n == 1 or n == 0: # Line 3  
 return True # Line 4  
  
 low = 2 # Line 5  
 high = n // 2 # Line 6 # Should be n // 2 ??  
  
 while low <= high: # Line 7  
 mid = (low + high) // 2 # Line 8  
 mid\_sq = mid \* mid # Line 9  
  
 if mid\_sq == n: # Line 10  
 return True # Line 11  
 elif mid\_sq < n: # Line 12  
 low = mid + 1 # Line 13  
 else: # Line 14  
 high = mid - 1 # Line 15  
  
 return False # Line 16

Ключевые изменения:

Исправлено условие для n <= 0, так как оно некорректно обрабатывает 0

Оптимизирован диапазон бинарного поиска high = n//2

5) Анализ и выводы:

* Неверное условие для n <= 0, так как оно некорректно обрабатывало 0.
* Логические ошибки.
* n = 0, так как сразу выявил ошибку в условии.