Пара №9

*Теоретическая часть*

1.

1) Синтаксическая ошибка - ошибка, возникающая из-за нарушения правил синтаксиса языка программирования. Например, в Python: print("Hello, World!" — отсутствует закрывающая скобка.

2) Логическая ошибка - ошибка, при которой программа работает без синтаксических ошибок, но выдает неверный результат из-за логической ошибки. Например, если нужно сложить два числа, но вместо этого они перемножаются: a + b заменено на a \* b.

3) Ошибка при условии - ошибка, возникающая из-за неверно заданного условия, что приводит к неправильному выполнению программы. Например, условие if (x = 5) вместо if (x == 5)

4) Ошибка обработки граничных случаев - ошибка, возникающая, когда программа не учитывает крайние или необычные входные данные. Например, деление на ноль: result = 10 / 0.

5) Отклонение от алгоритма - ошибка, при которой код выполняет действия, не соответствующие заданному алгоритму. Например, если алгоритм требует сортировки массива по возрастанию, а код сортирует его по убыванию.

2. Визуальный анализ кода - этот метод включает в себя тщательное чтение и проверку кода на наличие синтаксических ошибок, опечаток и других очевидных проблем.

Сравнение с описанием алгоритма - при использовании этого метода код сравнивается с заданным алгоритмом или спецификацией. Это помогает выявить отклонения от алгоритма, когда код делает не то, что предполагалось.

Ручное тестирование на конкретных примерах - этот метод предполагает выполнение кода на различных наборах данных, включая граничные случаи, для проверки его корректности.

*Практическая часть*

1. def is\_perfect\_square(n):

if n < 0: return False

elif n in (0, 1):

return True

else: left, right = 2, n // 2

while left <= right:

mid = (left + right) // 2

square = mid \* mid

if square == n:

return True

elif square < n:

left = mid + 1

else: right = mid - 1

return False

Примеры использования:

Print (is\_perfect\_square(16)) # True

Print (is\_perfect\_square(14)) # False

1. 1 и 6 строка

В 1 строке ошибка граничного условия - if n <= 0: не учитывает случай, когда n равно 0.

В 6 строке переменная high инициализирована значением n. Для n > 1 диапазон бинарного поиска должен быть от 2 до n // 2, так как корень из n не может быть больше n/2 при n > 1.

2. В этом коде нет мест, которые выглядят подозрительно, но на самом деле являются корректными. Все строки либо соответствуют спецификации, либо содержат ошибки или отклонения.

3.

|  |  |  |  |
| --- | --- | --- | --- |
| Тестовый случай | Ожидаемый  результат | Фактический  результат | Совпадения |
| n = -1 | Ложь | Ложь | Да |
| n = 0 | Истинный | Ложь | Нет |
| n = 1 | Истинный | Истинный | Да |
| n = 2 | Ложь | Ложь | Да |
| n = 3 | Ложь | Ложь | Да |
| n = 4 | Истинный | Истинный | Да |
| n = 5 | Ложь | Ложь | Да |
| n = 7 | Ложь | Ложь | Да |
| n = 9 | Истинный | Истинный | Да |
| n = 15 | Ложь | Ложь | Да |
| n = 16 | Истинный | Истинный | Да |
| n = 17 | Ложь | Ложь | Да |
| n = 1000000 | Истинный | Истинный | Да |
| n = 999999 | Ложь | Ложь | Да |
| n = 1000001 | Ложь | Ложь | Да |

4. Задание №4 в подкрепленном скриншоте

5. 1. Наиболее серьезной ошибкой была неправильная обработка случая, когда n = 0. Это ошибка граничного условия, так как функция должна была возвращать True для n = 0, но возвращала False.

2. Наиболее частым типом ошибки были связанные с граничными условиями. Это включает в себя как неправильную обработку случая n = 0, так и неоптимальную инициализацию переменной high в бинарном поиске.

3. Самым эффективным тестовый случай оказался тест с n = 0. Этот тест явно выявил несоответствие между ожидаемым и фактическим результатом.