Отчет

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

Теоретическая часть: Типы ошибок и методы поиска

1. Типы ошибок в коде

Синтаксическая ошибка

Определение: Ошибка, возникающая из-за нарушения правил языка программирования (например, пропущенный двоеточник или скобка).

Пример:

```python

if x == 5 # Пропущено двоеточие

print("Hello")

```

Логическая ошибка

Определение: Код работает без сбоев, но дает неправильный результат из-за неверной логики.

Пример:

```python

def average(a, b):

return a + b / 2 # Неверный порядок действий

```

Ошибка в условии

Определение: Условие написано некорректно (например, использован оператор `>` вместо `>=`).

Пример:

```python

if age > 17: # Должно быть >= для включения 18 лет

print("Доступ разрешен")

```

Ошибка обработки граничных случаев

Определение: Код работает правильно в большинстве случаев, но падает или даёт неверный результат на крайних значениях.

Пример:

```python

def find\_max(arr):

m = arr[0] # Ошибка при пустом списке arr

...

```

Отклонение от алгоритма

Определение: Реализация не соответствует заданному алгоритму — код делает не то, что должен.

Пример:

Функция должна находить минимальное значение, а находит максимальное.

2. Методы поиска ошибок

1. Визуальный анализ кода — внимательное чтение кода с целью выявления явных ошибок (например, опечаток, неверного порядка действий).

2. Сравнение с описанием алгоритма — проверка того, как реализация соответствует описанной логике.

3. Ручное тестирование на конкретных примерах — запуск функции с заранее известными входными данными и сравнение результата с ожидаемым.

Вот \*\*реализация функции `is\_perfect\_square(n)`\*\* с учётом заданного алгоритма, а также \*\*анализ кода и примеры тестирования\*\*. Текст оформлен так, чтобы его можно было легко скопировать в \*\*MS Word\*\*.

Практическая часть: Анализ функции `is\_perfect\_square`

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.

Реализация

```python

def is\_perfect\_square(n):

if n < 0:

return False # Отрицательные числа не могут быть квадратами

left = 0

right = n // 2 + 1 # Добавляем 1, чтобы захватить верхнюю границу

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

```

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Вход (`n`) | Ожидаемый результат | Фактический результат | Обоснование |
| 1 | 0 | `True` | `True` | 0 — точный квадрат |
| 2 | 1 | `True` | `True` | 1 — точный квадрат |
| 3 | 4 | `True` | `True` | 2\*2 = 4 |
| 4 | 16 | `True` | `True` | 4\*4 = 16 |
| 5 | 2 | `False` | `False` | Не квадрат |
| 6 | 15 | `False` | `False` | Не квадрат |
| 7 | -4 | `False` | `False` | Отрицательное число |
| 8 | 25 | `True` | `True` | 5\*5 = 25 |

Анализ корректности

Покрытие граничных случаев:

- \*\*n = 0\*\* → возвращается `True`

- \*\*n = 1\*\* → возвращается `True`

- \*\*n < 0\*\* → возвращается `False`

- \*\*n > 1\*\*, но не квадрат (например, 2, 3, 5, 7) → возвращается `False`

Корректность алгоритма:

- Использован \*\*бинарный поиск\*\*, что обеспечивает логарифмическую сложность `O(log n)`

- Условия выхода и обновления границ корректны

- Избегается переполнение типа (в Python это не проблема, но в других языках важно)

Возможные улучшения

1. Проверка типа входного значения:

Сейчас функция может получить строку или float, что вызовет ошибку. Можно добавить:

```python

if not isinstance(n, int):

return False

```

2. Учет больших чисел:

Для очень больших чисел стоит использовать более безопасное вычисление среднего:

```python

mid = left + (right - left) // 2 # Чтобы избежать переполнения

```

Вывод

Функция `is\_perfect\_square(n)` реализована корректно согласно заданным требованиям:

- Учитывает отрицательные числа.

- Верно обрабатывает ноль и единицу.

- Использует эффективный метод бинарного поиска.

- Покрывает все граничные случаи.