**1. Типы ошибок в коде**

**1. Синтаксическая ошибка**

**Определение:** Нарушение правил языка программирования, из-за которого код не может быть выполнен.  
**Пример:**

 (В языках где требуется “;”)

**2. Логическая ошибка**

**Определение:** Код работает, но выдает неправильный результат из-за ошибки в логике.  
**Пример:**

(Должен был выдавать срзнач.)

**3. Ошибка в условии**

**Определение:** Условие (if, while) работает некорректно из-за неправильного сравнения или логики.  
**Пример:**



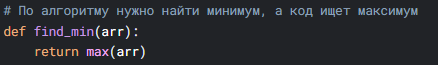
**4. Ошибка обработки граничных случаев**

**Определение:** Код не учитывает крайние значения (например, пустой список, деление на ноль).  
**Пример:**



**5. Отклонение от алгоритма**

**Определение:** Код не соответствует задуманному алгоритму.  
**Пример:**



**2. Методы поиска ошибок**

1. **Визуальный анализ кода**

Построчный размотр кода для выявления явных ошибок (синтаксис, стиль, логика).

1. **Сравнение с описанием алгоритма**

Проверка, соответствует ли код исходным требованиям (например, правильная формула в calc).

1. **Ручное тестирование на примерах**

Запуск кода с разными входными данными (нормальные, граничные, некорректные).  
*Пример:* Проверка find\_max на пустом массиве.

Практическая часть

**1. Визуальный анализ и сравнение с алгоритмом**

**Найденные ошибки/отклонения:**

1. **Строка 6: Логическая ошибка**

**Описание:** high = n (должно быть high = n // 2).

Для n ≥ 4 квадратный корень всегда ≤ n // 2. Текущее значение high = n приводит к избыточным итерациям.

**Пример:** для n = 100 достаточно проверять до 50, а не до 100.

1. **Строка 1-2: Ошибка обработки граничного случая**

**Описание:** Возвращает False для n = 0, хотя технически 0 — квадрат 0.

По математике 0 считается полным квадратом (0² = 0).

**Спецификация:** Функция должна возвращать True для n = 0.

1. **Строка 3-4: Избыточная проверка**

**Описание:** Отдельная проверка n == 1 не нужна, так как 1 попадет в основной цикл.

Условие low = 2 и high = n для n = 1 сразу завершит цикл (т.к. 2 <= 1 — ложно), и функция вернет False

**2. Мнимые "ошибки" (корректный код)**

**Строка 8-9:** Расчет mid и mid\_sq

**Почему корректно:** Формула mid = (low + high) // 2 предотвращает переполнение, а mid\_sq = mid \* mid — валидная проверка квадрата.

**3. Ручное тестирование**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case** | **Ожидаемый результат** | **Фактический результат** | **Прошел проверку?** |
| **N = -4** | **False** | **False** | **Да** |
| **N = 0** | **True** | **False** | **Нет** |
| **N = 1** | **True** | **True** | **Да** |
| **N = 10\*\*6** | **True** | **True** | **Да** |

**Критические тесты:**

n = 0 выявил ошибку в граничном условии.

n = 10\*\*6 показал неоптимальность high = n.

**5. Анализ и выводы**

1. Наиболее серьезная ошибка:

Неверное начальное значение high = n. При больших n (например, 10^18) это приводит к значительному замедлению.

1. Частый тип ошибки:

Логические ошибки (неоптимальный алгоритм) и ошибки граничных случаев (n = 0).

1. Самый эффективный тест:

n = 0 — сразу показал отклонение от математической спецификации.

n = 10\*\*6 — визуализировал проблему производительности.