**1. Основные типы ошибок в программировании**

**Синтаксические ошибки**

**Суть:** Нарушение правил написания кода, приводящее к невозможности его выполнения.  
**Пример:** Пропущенное двоеточие после условия:

python

Copy

Download

if x == 5 *# Ошибка: отсутствует двоеточие*

print("x равно 5")

**Логические ошибки**

**Суть:** Код выполняется, но дает неверные результаты из-за ошибочной логики.  
**Пример:** Неправильная проверка четности:

python

Copy

Download

def is\_even(n):

return n % 2 != 0 *# Ошибка: должно быть n % 2 == 0*

**Ошибки в условиях**

**Суть:** Некорректные логические выражения в управляющих конструкциях.  
**Пример:** Противоречивое условие:

python

Copy

Download

if x > 10 and x < 5: *# Невозможное условие*

print("x в диапазоне")

**Ошибки обработки граничных значений**

**Суть:** Неучет особых случаев при работе с данными.  
**Пример:** Отсутствие проверки деления на ноль:

python

Copy

Download

def divide(a, b):

return a / b *# Ошибка: нет обработки случая b = 0*

**Отклонения от алгоритма**

**Суть:** Реализация не соответствует техническому заданию.  
**Пример:** Использование линейного поиска вместо бинарного:

python

Copy

Download

def find\_square(n):

for i in range(n): *# Должен быть бинарный поиск*

if i \* i == n:

return True

return False

**2. Основные методы выявления ошибок**

1. **Визуальная проверка** - анализ кода на соответствие требованиям
2. **Сопоставление с алгоритмом** - пошаговая проверка логики
3. **Экспериментальное тестирование** - проверка работы на различных данных

**Практический разбор функции is\_perfect\_square(n)**

**Анализ исходного кода**

python

Copy

Download

def is\_perfect\_square(n):

if n < 0:

return False

if n == 0 or n == 1:

return True

left, right = 2, n // 2 *# Проблемная строка*

while left <= right:

mid = (left + right) // 2

if mid \* mid == n:

return True

elif mid \* mid < n:

left = mid + 1

else:

right = mid - 1

return False

**Выявленные проблемы**

1. **Ошибка граничного случая** (строка 5):
   * Для n=2 или n=3 диапазон становится невалидным [2,1]
   * Решение: изменить правую границу на right = n
2. **Кажущаяся ошибка** (строка 6):
   * Условие left <= right корректно, так как включает проверку равенства

**Результаты тестирования**

| **Ввод (n)** | **Ожидаемый** | **Фактический** | **Соответствие** |
| --- | --- | --- | --- |
| -1 | False | False | ✓ |
| 0 | True | True | ✓ |
| 1 | True | True | ✓ |
| 2 | False | True | ✗ (Ошибка) |
| 4 | True | True | ✓ |
| 15 | False | False | ✓ |
| 16 | True | True | ✓ |

**Исправленная версия**

python

Copy

Download

def is\_perfect\_square(n):

if n < 0:

return False

if n == 0 or n == 1:

return True

left, right = 2, n *# Исправление границы*

while left <= right:

mid = (left + right) // 2

if mid \* mid == n:

return True

elif mid \* mid < n:

left = mid + 1

else:

right = mid - 1

return False

**Выводы и рекомендации**

1. **Критическая ошибка**: Неправильно заданный диапазон поиска приводил к ложным положительным результатам
2. **Типичная проблема**: Чаще всего встречались ошибки обработки граничных значений
3. **Эффективный тест**: Проверка на числе 2 сразу выявила основную проблему

**Рекомендации**:

* Всегда тестируйте граничные случаи
* Внимательно проверяйте начальные условия алгоритмов
* Используйте минимальные тестовые случаи для быстрой проверки