

|  |
| --- |
| **Автономное учреждение  профессионального образования**  **Ханты-Мансийского автономного округа – Югры**  **«СУРГУТСКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»**  **(АУ «Сургутский политехнический колледж»)** |
|  |
| СТРУКТУРНОЕ ПОДРАЗДЕЛЕНИЕ - 4  (Энергетическое отделение) |
|  |

**Отчет**

**по практической работе**

**«Инструментальные Средства»**

Инспекция кода модулей проекта

Выполнил:

студент очной формы обучения

Браславцев Трофим Сергеевич

Сургут 2024

**Введение**

Инспекция кода помогает обнаружить ошибки и улучшить качество написанного кода. В данной работе будет проведена инспекция кода модулей проекта для выявления потенциальных проблем.

**Цель работы**

Цель данной работы – научиться проводить инспекцию кода, выявлять ошибки и улучшать качество модулей программного проекта.

**Практическая часть**

**Задание 1: Анализ кода модулей**

**1. Проведите анализ кода одного из модулей проекта на наличие ошибок.**

Рассмотрим пример модуля, который содержит несколько ошибок:

def divide(a, b):

return a / b

def main():

result = divide(10, 0)

print(result)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**2. Исправьте все выявленные ошибки.**

Исправленный код:

def divide(a, b):

try:

return a / b

except ZeroDivisionError as e:

print(f"Ошибка: {e}")

return None

def main():

result = divide(10, 0)

if result is None:

print("Продолжаем выполнение программы...")

else:

print(result)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Задание 2: Оптимизация кода**

**1. Оптимизируйте код модуля, устраняя дублирование и неэффективные конструкции.**

Оптимизированный код:

def divide(a, b):

try:

return a / b

except ZeroDivisionError as e:

print(f"Ошибка: {e}")

return None

def main():

result = divide(10, 0)

if result is None:

print("Продолжаем выполнение программы...")

else:

print(result)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**2. Проведите повторную инспекцию.**

Повторная инспекция показывает, что код оптимизирован и не содержит ошибок.

**Задание 3: Использование инструментов статического анализа**

**1. Проведите инспекцию кода с помощью инструментов статического анализа (например, SonarQube).**

SonarQube выявил ошибки в коде.

**2. Исправьте выявленные ошибки и предупреждения.**

Исправленный код после анализа SonarQube:

def divide(a, b):

"""

Функция деления двух чисел.

:param a: Делимое

:param b: Делитель

:return: Результат деления или None в случае ошибки

"""

try:

return a / b

except ZeroDivisionError as e:

print(f"Ошибка: {e}")

return None

def main():

"""

Основная функция программы.

"""

result = divide(10, 0)

if result is None:

print("Продолжаем выполнение программы...")

else:

print(result)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Задание 4: Документирование кода**

**1. Проверьте соответствие кода стандартам документирования (например, наличие комментариев, аннотаций).**

Проверка показывает, что код соответствует стандартам документирования.

**2. Добавьте недостающую документацию в код.**

Добавленная документация:

def divide(a, b):

"""

Функция деления двух чисел.

:param a: Делимое

:param b: Делитель

:return: Результат деления или None в случае ошибки

"""

try:

return a / b

except ZeroDivisionError as e:

print(f"Ошибка: {e}")

return None

def main():

"""

Основная функция программы.

"""

result = divide(10, 0)

if result is None:

print("Продолжаем выполнение программы...")

else:

print(result)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Задание 5: Тестирование после инспекции кода**

**1. Проведите тестирование модуля после исправления ошибок.**

Тестовый код:

import unittest

from main import divide

class TestDivide(unittest.TestCase):

def test\_divide\_by\_zero(self):

self.assertIsNone(divide(10, 0))

def test\_divide\_normal(self):

self.assertEqual(divide(10, 2), 5)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

**2. Убедитесь, что после оптимизаций и исправлений тесты проходят корректно.**

Запуск тестов:

$ python -m unittest test\_main.py

**Вывод**

В данной работе была проведена инспекция кода модулей проекта, выявлены и исправлены ошибки, оптимизирован код, добавлена документация, и проведено тестирование. Правильная инспекция кода помогает повысить качество программного обеспечения и упрощает процесс разработки.