**Финансово-экономический колледж**

**РГЭУ (РИНХ)**



**Практика**

Студент: ИС-202

**Орлов Илья Вадимович**

Преподаватель:

**Журавлёв Д.Г.**

**Ростов-на-Дону**

**Теоретическая часть**

1. Метрики ПО: Назначение и Применение

Определение метрики программного обеспечения

Метрика ПО — это количественная мера, используемая для оценки характеристик программного кода, процесса разработки или качества продукта.

**Ключевые направления применения метрик:**

1.Оценка сложности – помогает выявить трудные для понимания и поддержки участки кода (например, цикломатическая сложность).

2.Прогнозирование трудозатрат – метрики (например, количество строк кода — SLOC) позволяют оценить время и ресурсы на разработку.

3.Контроль качества – выявление "кода с запахом" (code smells) и потенциальных уязвимостей.

4.Улучшение стиля кода – метрики стилистики (например, длина идентификаторов) помогают повысить читаемость.

**2.Метрики Сложности**

**Зачем измерять сложность кода?**

Высокосложный код:

1.Трудно понимать и модифицировать.

2.Повышает риск ошибок.

3.Усложняет тестирование и отладку.

**Примеры метрик сложности:**

1.Глубина вложенности – количество уровней вложенных if/for/while. Оптимально — не больше 3-4.

2.Количество параметров функции – слишком много параметров (более 3-4) усложняют её использование.

3.Цикломатическая сложность (McCabe) – измеряет количество линейно независимых путей в коде.

**3.Метрики Стилистики (Качества Кода)**

**Зачем нужны метрики стилистики?**

Они напрямую влияют на:

1.Читаемость – понятный код быстрее анализируется.

2.Сопровождаемость – код с единым стилем проще изменять.

**Примеры метрик стилистики:**

1.Длина идентификаторов – слишком короткие имена (x, tmp) снижают понятность. Оптимально — 8-20 символов.

2.Количество комментариев – соотношение комментариев/код (например, 20-30%). Слишком много — может означать сложный код.

3.Длина строки кода – рекомендуется не больше 80-120 символов для удобства чтения.

4.Глубина вложенности – вложенные if/for глубже 3-4 уровней затрудняют восприятие.

5.Кодстайл – соответствие стандартам именования (camelCase, snake\_case), отступов (пробелы vs табы).

**3. Практическая часть**

|  |  |
| --- | --- |
| Метрика | Результат |
| Средняя длина идентификаторов (основной код) | ~9.5 символов |
| Средняя длина идентификаторов (calc()) | ~1.3 символа |
| Средняя длина идентификаторов (find\_max()) | ~1.5 символа |
| Процент комментариев | 5.8% |
| Средняя длина строки | ~45 символов |

**2. Анализ и Рекомендации**

***Сильные стороны:***

**1.Хорошая структура основного кода:**

Четкое разделение на логические блоки (загрузка данных, обработка, визуализация).

**2.Использование контекстных менеджеров (with pd.ExcelWriter):**

Корректное управление ресурсами.

***Слабые стороны:***

**1.Плохие имена переменных в функциях (calc, find\_max):**

a, b, r — непонятно без контекста.

**2.Недостаточно комментариев:**

Нет пояснений к сложной логике (например, нормализация курса).

**Рекомендации:**

1.Улучшить именование:

В calc():

a → first\_operand, b → second\_operand, op → operator, r → result.

В find\_max():

arr → numbers, m → max\_value, x → current\_number.

**2.Добавить комментарии:**

Пояснить логику нормализации курса.

Описать настройки графиков (plt.figure(), plt.plot()).