Міністерство освіти і науки України

Національний технічний університет «ХПІ»

Навчально-науковий інститут комп’ютерних наук та інформаційних технологій

Кафедра комп’ютерної інженерії та програмування

ЗВІТ

з лабораторної роботи № 11

з дисципліни «Сучасні технології безпечного програмування»

«**ОЦІНКА ЯКОСТІ КОДУ**»

Виконав:

студент гр. КН-Н922б

Кулик Д.І.

Перевірив:

Бульба С. С.

Харків – 2022

Мета роботи: Дослідити алгоритми визначення якості коду.

Індивідуальне завдання

Використовуючи код будь-якого додатку (неважливо, чи він створений вами, або взятий з github/gitlab):

- рівень якості програмування
- складність розуміння програми
- трудомісткість кодування програми
- цикломатичне число Мак-Кейба
- метрика Чепіна

Метрики потрібно виконувати для:

- source code
- декомпільованого коду

В звіті:

- Зробити висновки впливу компіляції та обфускації на якість коду
- Описати алгоритми / формули для обчислення реалізованих характеристик якості коду

При виконанні роботи рекомендовано використовувати:

- Roslyn (тільки для .net)
- або Antrl
- Будь-які інші бібліотеки для аналізу синтаксичного дерева потрібно обговорювати з викладачем

Хід роботи

Сьогодні існує дуже багато алгоритмів для обчислення реалізованих характеристик якості коду. В рамках даної лабораторної роботи розглянемо такі утиліти, як flake8, mccabe та radon.

Flake8 — це бібліотека Python, яка містить PyFlakes, pycodestyle і сценарій McCabe Неда Батчелдера. Це чудовий інструментарій для перевірки вашої кодової бази на стиль кодування (PEP8), програмні помилки (наприклад, «library imported but unused» та «Undefined name») та перевірка цикломатичної складності.

Для утиліти mccabe цикломатична складність приблизно еквівалентна одиниці плюс кількість «циклів і операторів if». Проста інтерпретація полягає в тому, що він показує верхню межу для кількості тестових випадків, необхідних для отримання покриття гілок коду, отже, це приблизно вказує зусилля, необхідні для написання тестів.

Код із високою цикломатичною складністю (зазвичай вважається 10+), імовірно, буде важко зрозуміти, і тому він має більшу ймовірність містити дефекти.

У Python є своя особиста утиліта для оцінки якості коду. Вона називається radon. Вона написана на цьому ж самому Python і працює виключно з файлами. Має в собі багато метрик для оцінок текстів програм, що будуть розглянуті далі.

Важливі фрагменти програми

Використаємо статичний аналізатор flake8 та утиліти для мови Python та виконаємо аналіз для коду лабораторної роботи №04.

```
(stbp_venv) C:\tmp\stbp>python -m flake8 kulyk04
kulyk04\main.py:66:80: E501 line too long (93 > 79 characters)
kulyk04\main.py:75:80: E501 line too long (88 > 79 characters)
kulyk04\main.py:82:80: E501 line too long (101 > 79 characters)
kulyk04\main.py:96:80: E501 line too long (111 > 79 characters)
kulyk04\main.py:99:5: F841 local variable 'checksum' is assigned to but never used
kulyk04\main.py:103:5: F841 local variable 'checksum_for_verification' is assigned to but never used
kulyk04\main.py:110:80: E501 line too long (83 > 79 characters)
kulyk04\main.py:147:80: E501 line too long (100 > 79 characters)
kulyk04\main.py:148:80: E501 line too long (116 > 79 characters)
```

Рисунок 1 – Результат виконання утиліти flake8

Використаємо утиліти mccabe та підрахунок метрики Мак Кейба для коду лабораторної роботи №04.

```
(stbp_venv) C:\tmp\stbp>python -m mccabe kulyk04/main.py
10:0: 'get_2048_words' 3
31:4: 'PRNG.__init__' 1
36:4: 'PRNG.__call__' 2
45:0: 'encrypt' 1
55:0: 'decrypt' 1
69:0: 'get_entropy_bits' 3
79:0: 'decode_phrase' 5
116:0: 'normalize_string' 1
122:0: 'get_seed' 1
If 144 4
```

Рисунок 2 – Результат виконання утиліти maccabe

Використаємо утиліту radon для обчислення різних метрик для коду лабораторної роботи №04

Чим більше у коді переходів (if-else), циклів, генераторів, обробників винятків та логічних операторів — тим більше у програми варіантів виконання і тим складніше пам'ятати різні стани системи. Метрика, яка вимірює складність коду, спираючись на кількість цих операцій, називається цикломатичною складністю програми.

CC score	Rank	Risk
1 - 5	A	low - simple block
6 - 10	B	low - well structured and stable block
11 - 20	C	moderate - slightly complex block
21 - 30	D	more than moderate - more complex block
31 - 40	E	high - complex block, alarming
41+	F	very high - error-prone, unstable block

```
(stbp_venv) C:\tmp\stbp>python -m radon cc kulyk04/
kulyk04\main.py
    F 79:0 decode_phrase - A
    F 10:0 get_2048_words - A
    C 29:0 PRNG - A
    F 69:0 get_entropy_bits - A
    F 116:0 normalize_string - A
    M 36:4 PRNG.__call__ - A
    F 45:0 encrypt - A
    F 55:0 decrypt - A
    F 122:0 get_seed - A
    M 31:4 PRNG.__init__ - A
    C 65:0 DecodingError - A
```

Рисунок 3 – Результат виконання утиліти radon cc (Cyclomatic Complexity)

Банальний підрахунок числа рядків у вихідниках. А ще числа рядків, які безпосередньо містять код і числа рядків-коментів.

- **LOC**: the total number of lines of code
- **LLOC**: the number of logical lines of code
- **SLOC**: the number of source lines of code - not necessarily corresponding to the **LLOC** [Wikipedia]
- **comments**: the number of Python comment lines (i.e. only single-line comments #)
- **multi**: the number of lines representing multi-line strings
- **blank**: the number of blank lines (or whitespace-only ones)

```
(stbp_venv) C:\tmp\stbp>python -m radon raw kulyk04/
kulyk04\main.py
    LOC: 167
    LLOC: 104
    SLOC: 120
    Comments: 7
    Single comments: 9
    Multi: 0
    Blank: 38
    - Comment Stats
        (C % L): 4%
        (C % S): 6%
        (C + M % L): 4%
```

Рисунок 4 – Результат виконання утиліти radon raw

Цей індекс говорить нам про те, наскільки складно буде підтримувати чи редагувати шматок програми. Цей параметр розраховується з урахуванням чисел, отриманих з метрик, порахованих вище.
У відповідь ми отримаємо список файлів у проекті та їхній індекс підтримуваності, від легкого до дуже важкого.

MI score	Rank	Maintainability
100 - 20	A	Very high
19 - 10	B	Medium
9 - 0	C	Extremely low

```
(stbp_venv) C:\tmp\stbp>python -m radon mi kulyk04/
kulyk04\main.py - A
```

Рисунок 5 – Результат виконання утиліти radon mi (Maintainability Index)

Висновки: в результаті виконання лабораторної роботи було досліджено алгоритми визначення якості коду.