ДОДАТОК А

Слайди презентації

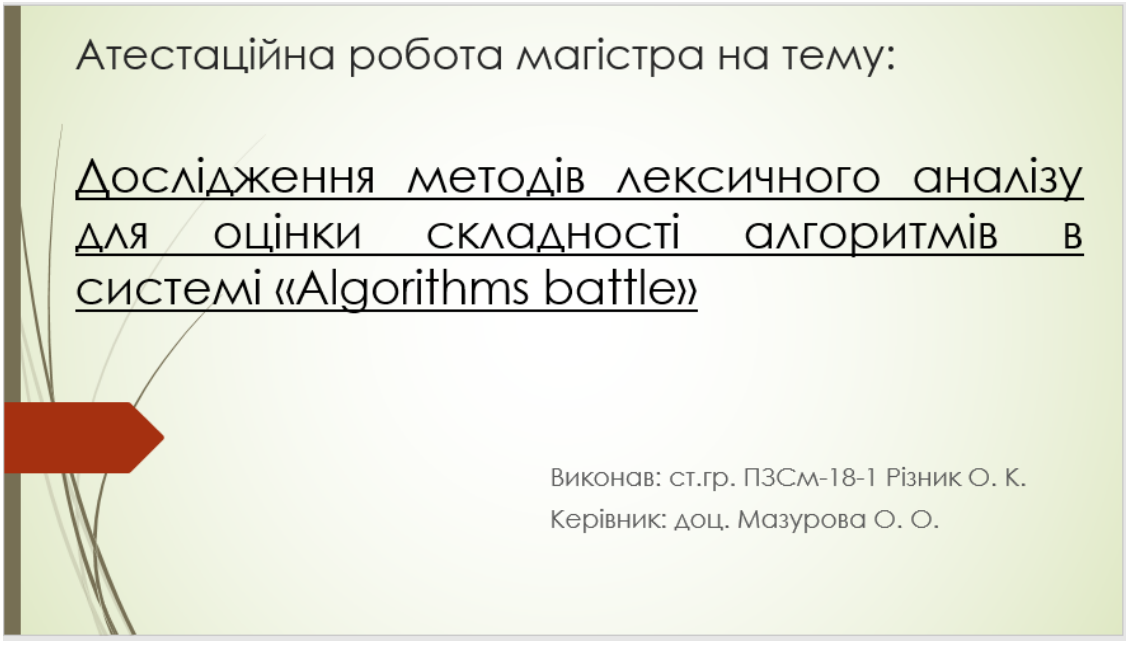

Рисунок А.1 – Титульний слайд

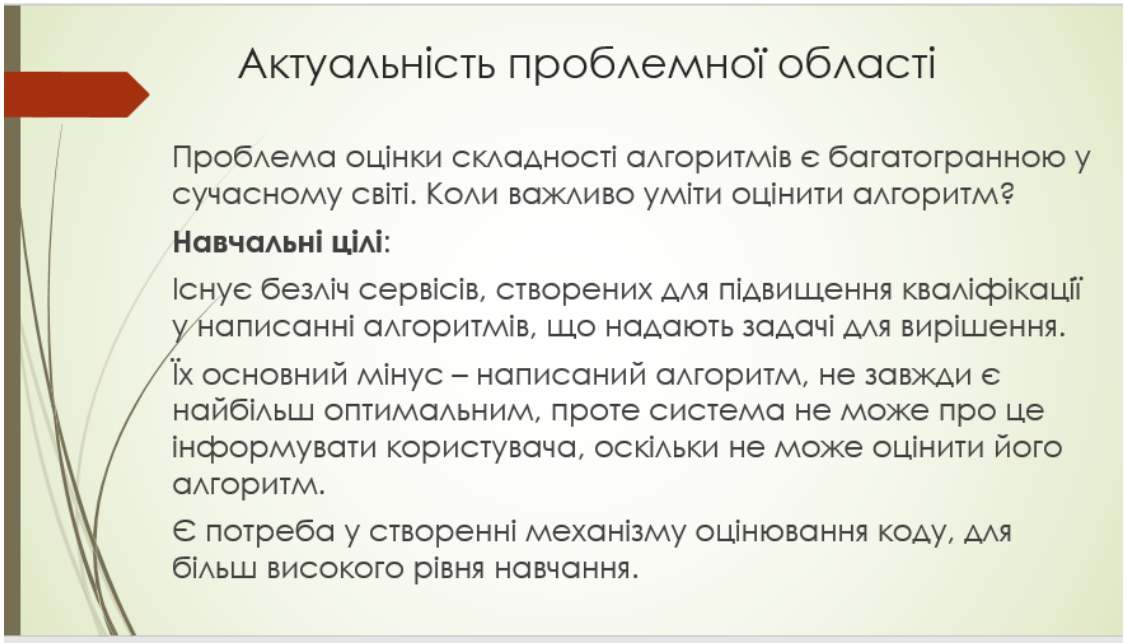

Рисунок А.2 – Слайд «Актуальність у навчальних цілях»

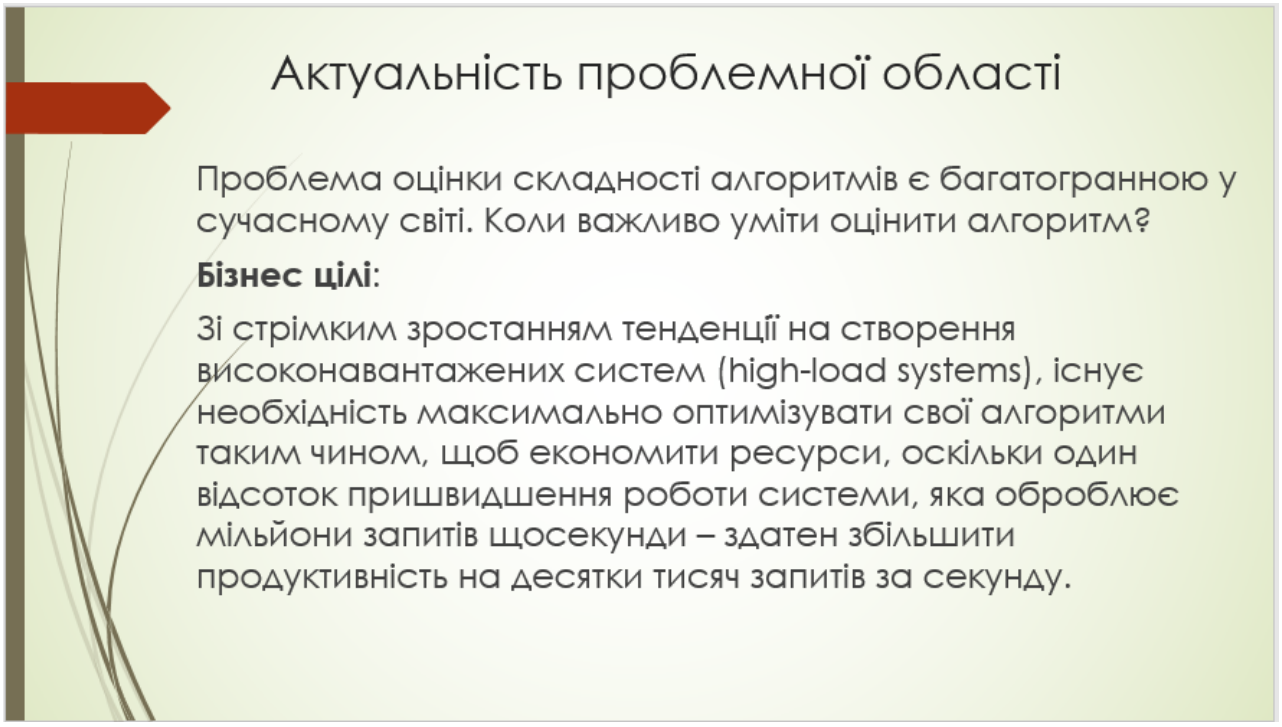

Рисунок А.3 – Слайд «Актуальність у бізнес цілях»

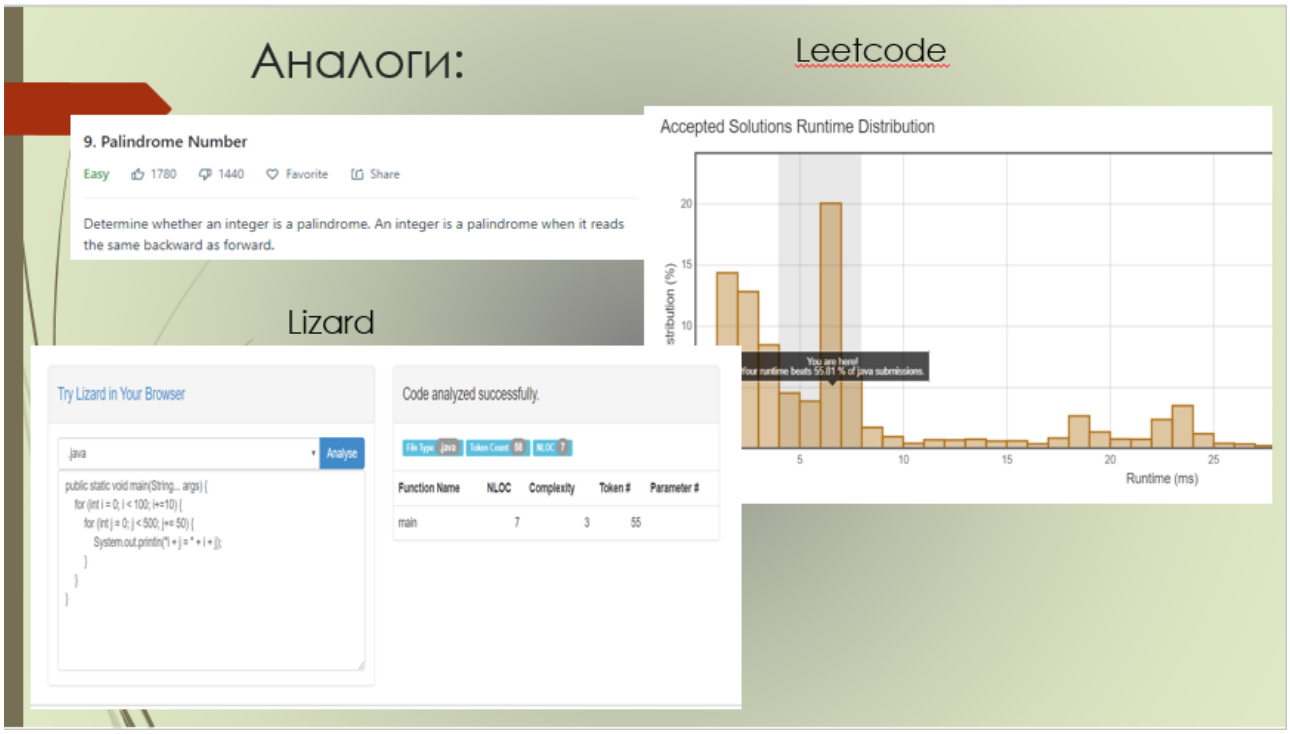

Рисунок А.4 – Слайд «Аналоги»

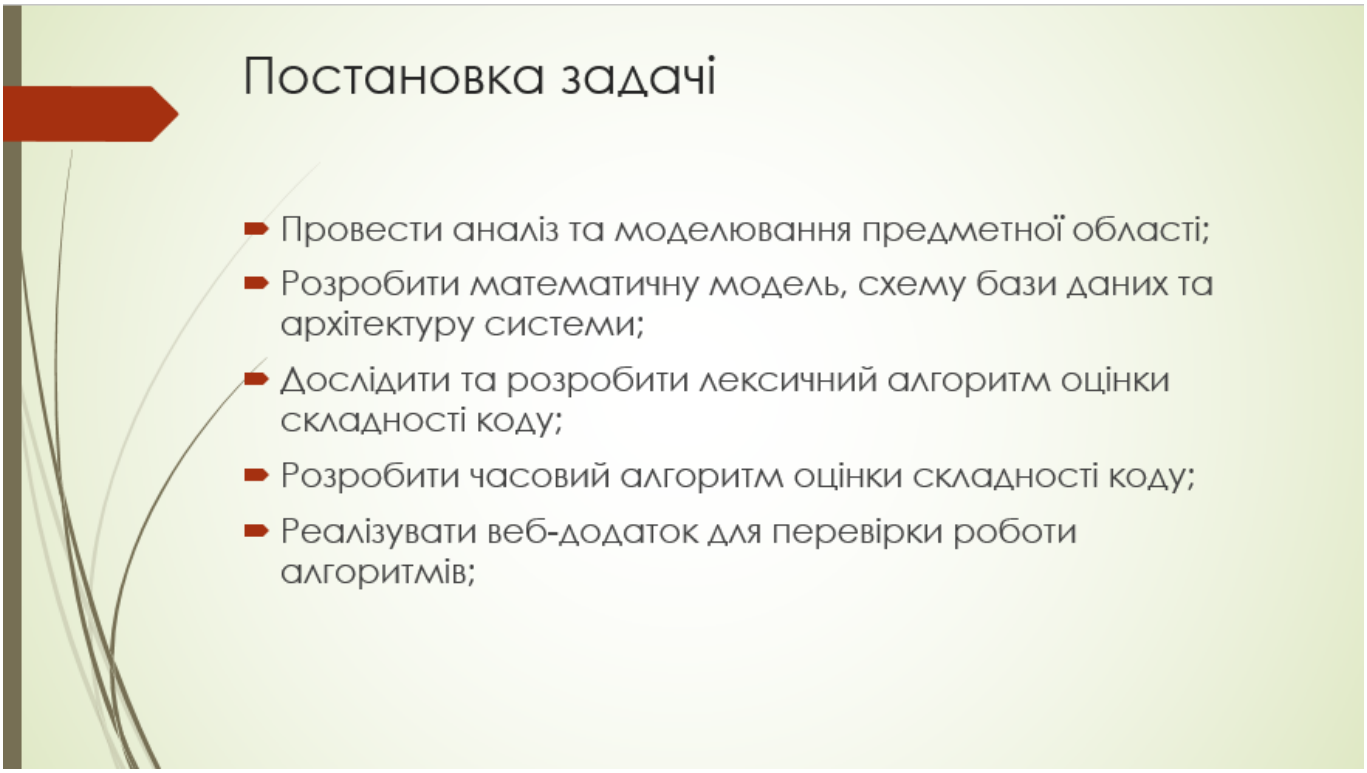

Рисунок А.5 – Слайд «Постановка задачі»

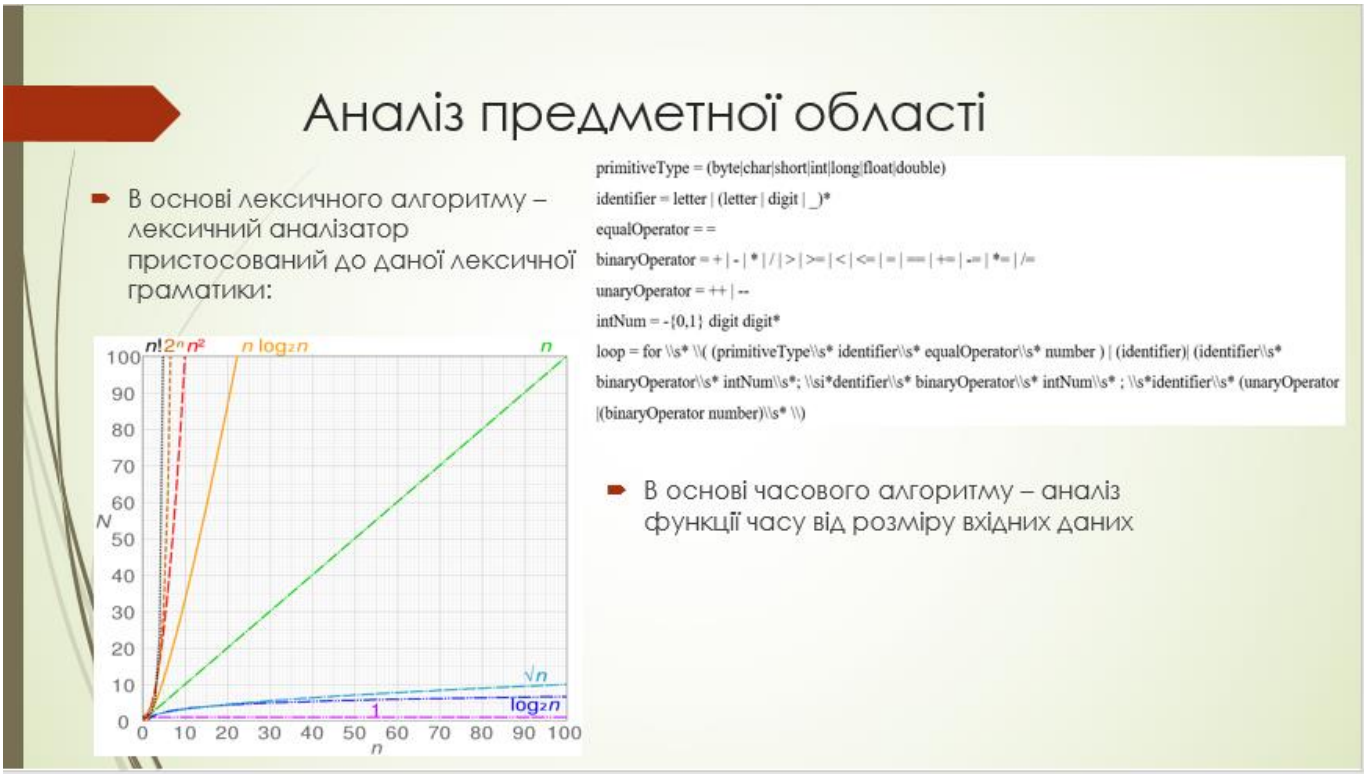

Рисунок А.6 – Слайд «Аналіз предметної області»

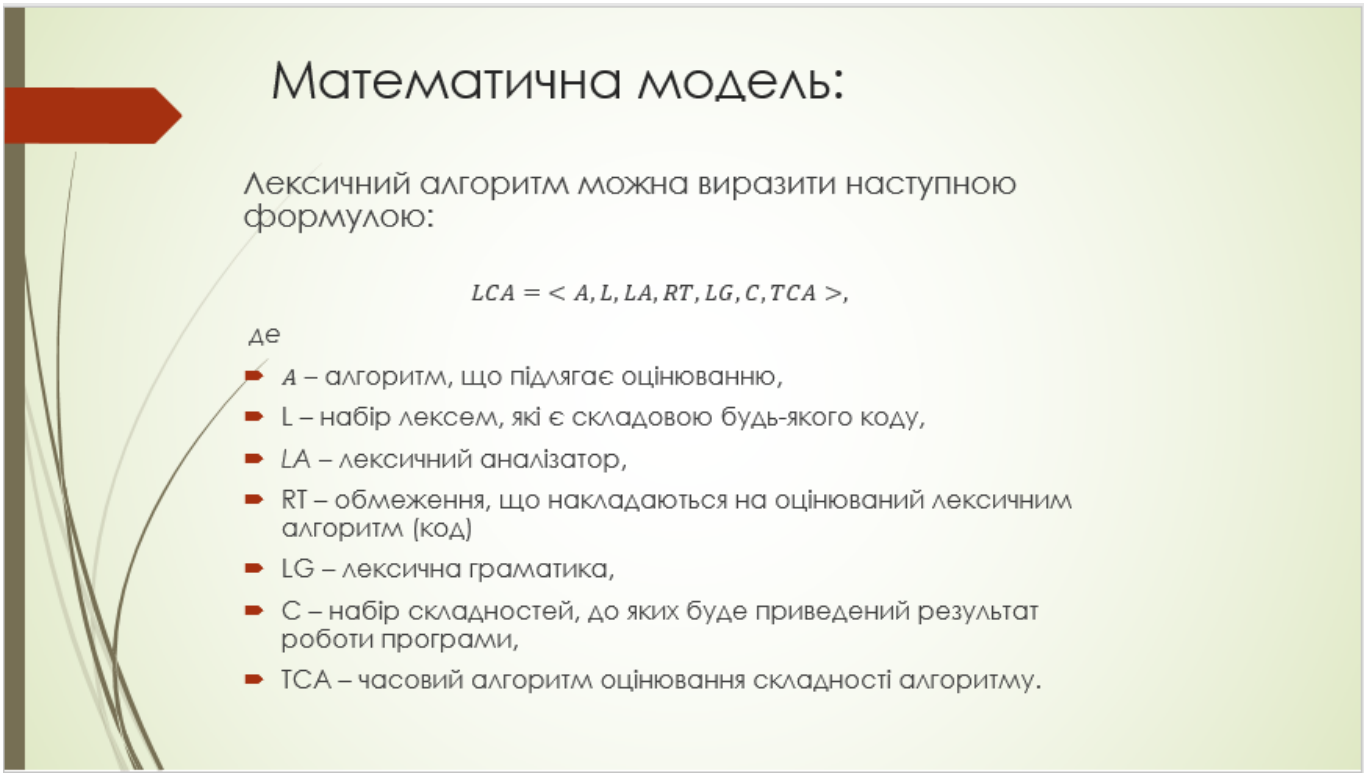

Рисунок А.7 – Слайд «Математична модель»

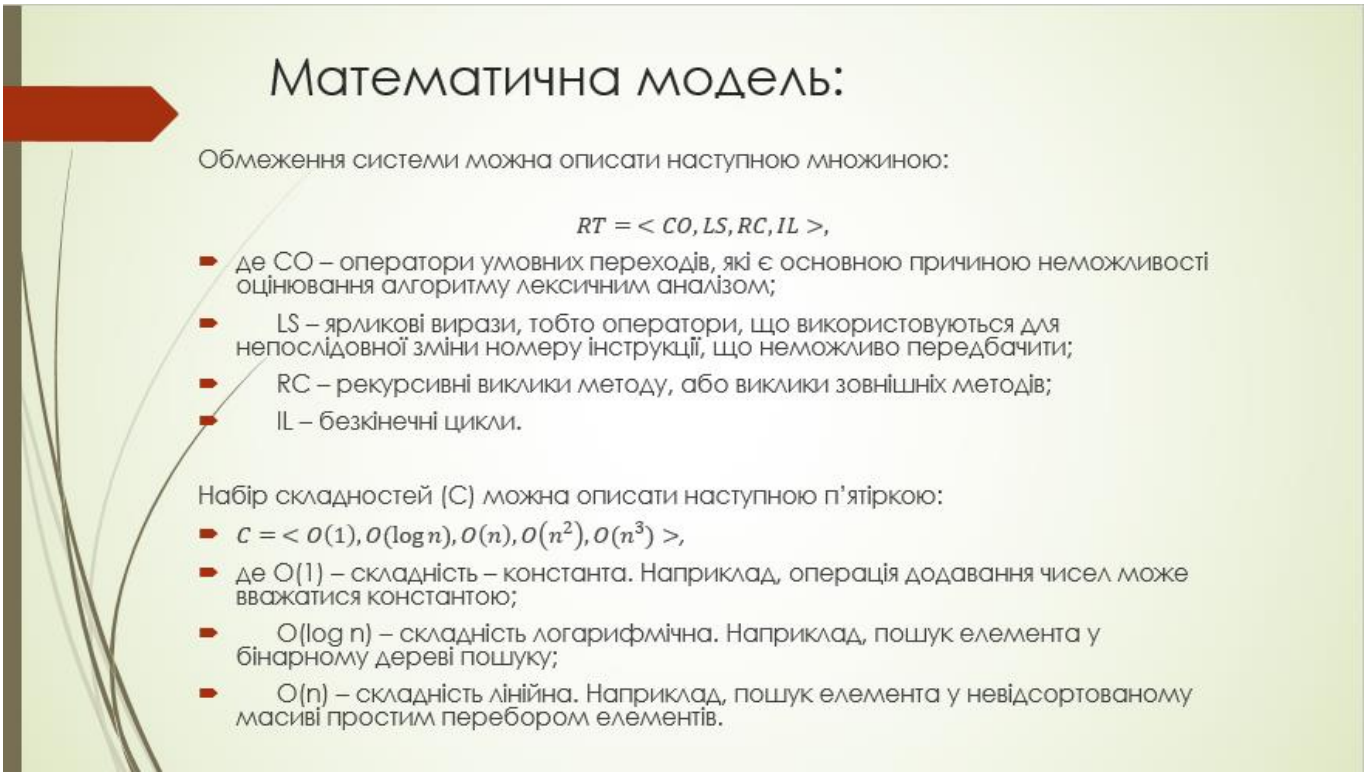

Рисунок А.8 – Слайд «Математична модель»

Рисунок А.9 – Слайд «Загальна схема лексичного алгоритму»

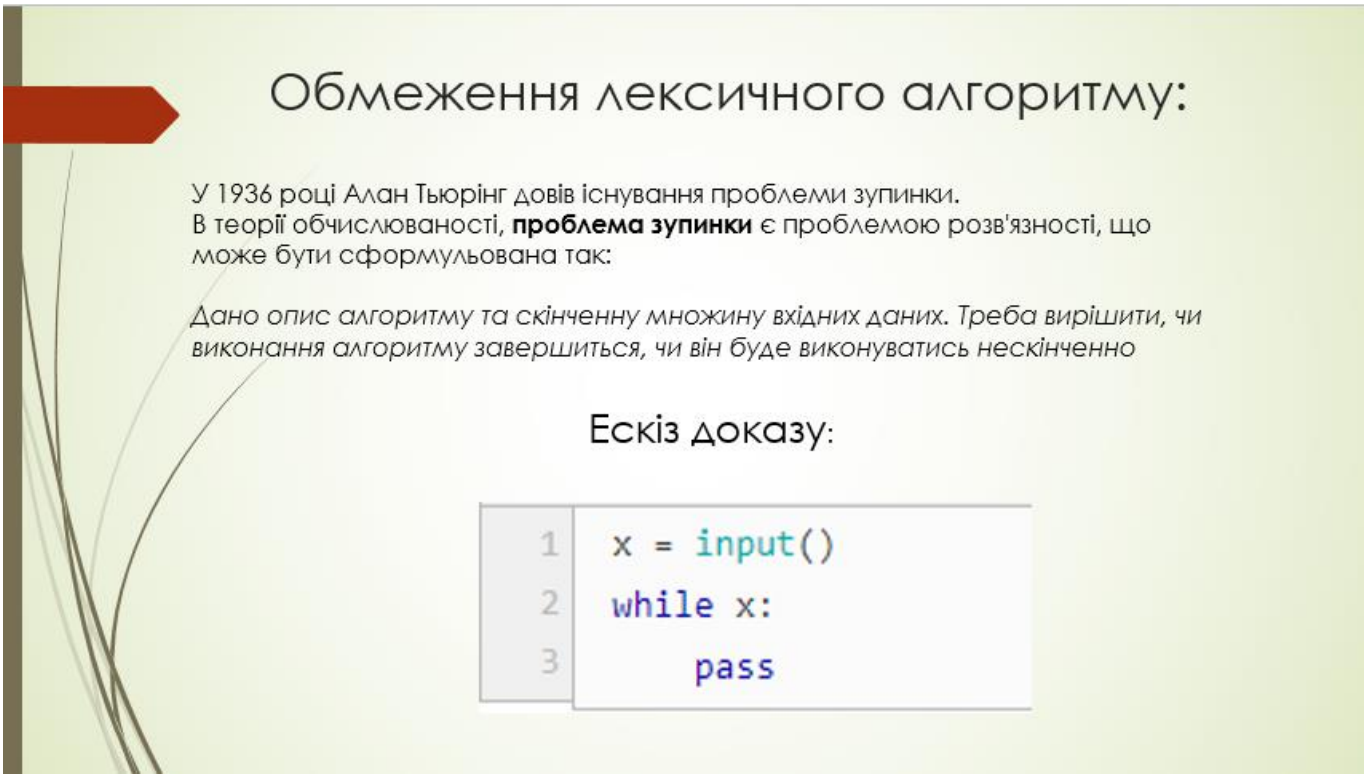

Рисунок А.10 – Слайд «Обмеження лексичного алгоритму»

Рисунок А.11 – Слайд «Дослідження часового алгоритму»

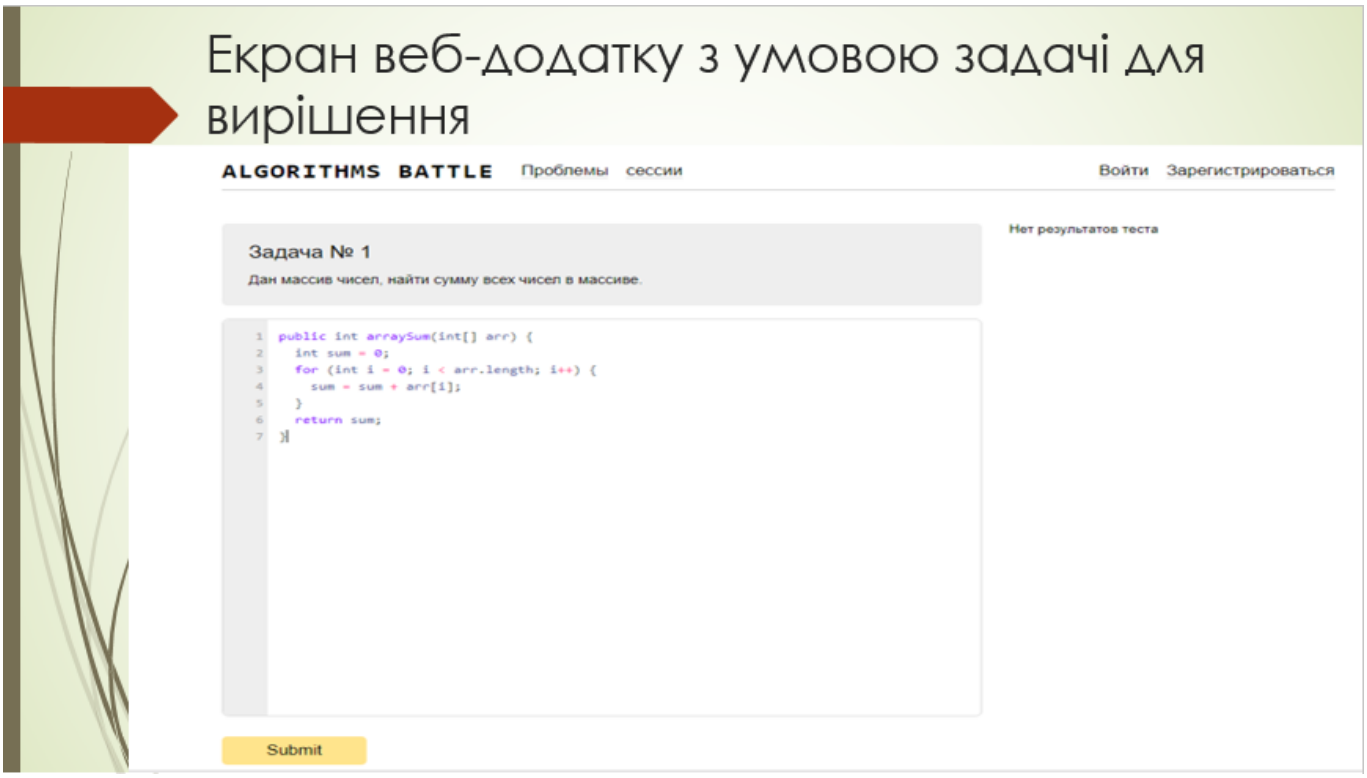

Рисунок А.12 – Слайд «Сторінка з умовою задачі для вирішення»

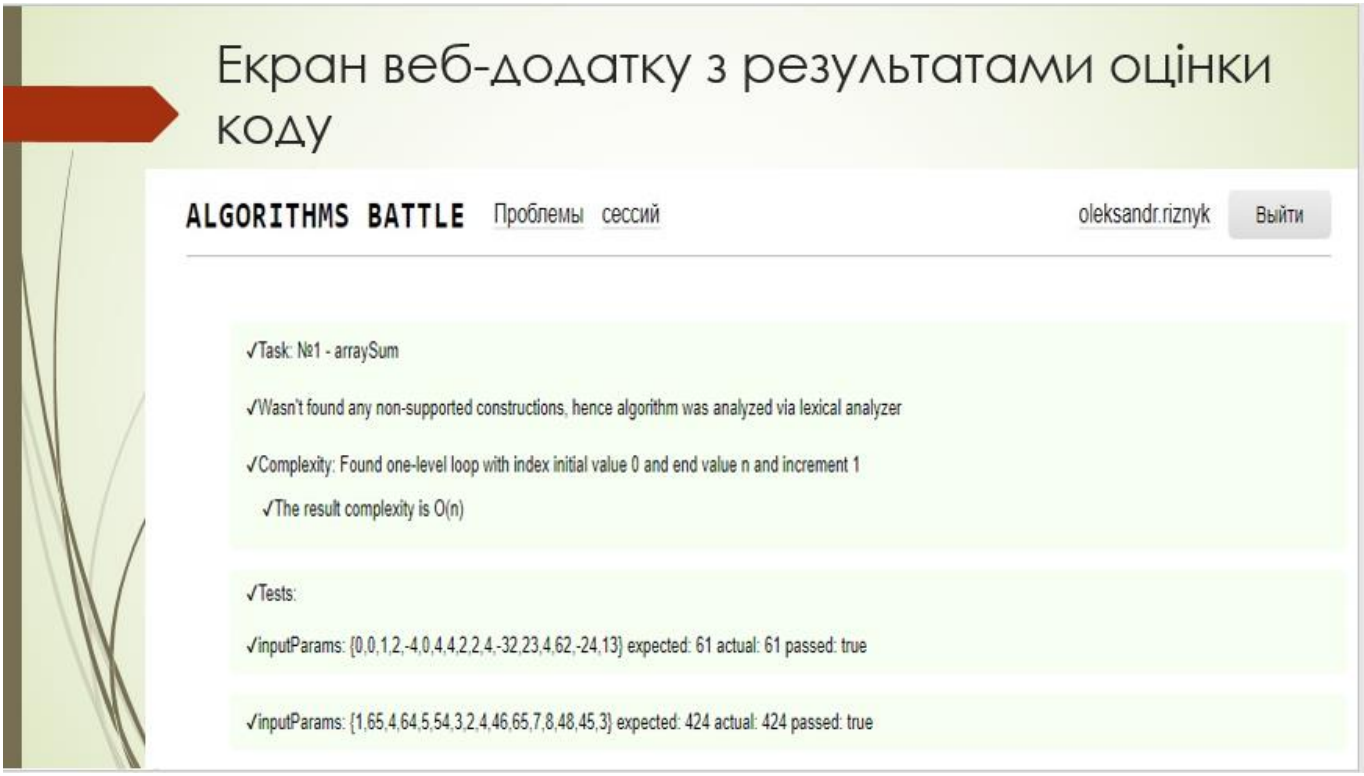

Рисунок А.13 – Слайд «Сторінка з результатами оцінки коду»

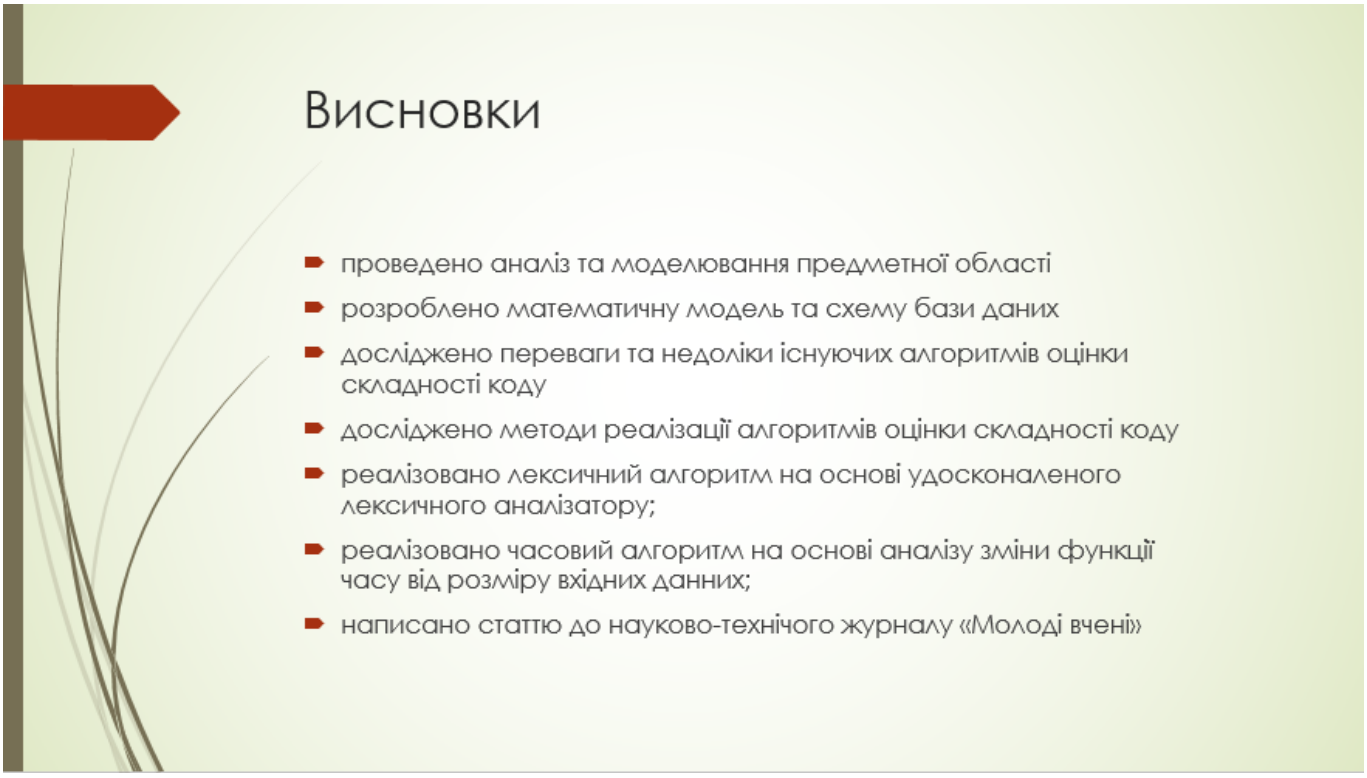

Рисунок А.14 – Слайд «Висновки»

ДОДАТОК Б

Стаття «Дослідження та розробка алгоритму оцінки складності програми на основі лексичного та часового аналізу коду»

О.К. Різник[1], О.О. Мазурова[1]

[1]*Харьківский національний університет радіоелектроніки*

Дослідження та розробка алгоритму оцінки складності програми на основі лексичного та часового аналізу коду

Робота присвячена дослідженню способів точної оцінки складності програмного коду на основі його лексичного аналізу та часових замірів. Для лексичного аналізу використовується доповнена та адаптована версія лексичного аналізатора. Задача створеного аналізатора – знайти всі інструкції програми, які мають вплив на складність програми, підрахувати кількості таких операцій та ітерацій програми. Оскільки, на лексичний аналіз складності накладається набір обмежень, додатково було вирішено доповнити його часовим алгоритмом.

Ключові слова: *алгоритм, складність алгоритму, лексичний аналіз, big-o notation, лексичний аналізатор, часовий алгоритм.*

Постановка проблеми

На сьогодні, основною мірою оптимальності алгоритму є часова складність алгоритму. Часова складність алгоритму в комп'ютерних науках є обчислювальною величиною, яка описує час потрібний для виконання алгоритму. Вона зазвичай визначається шляхом підрахунку кількості елементарних операцій, виконуваних алгоритмом, при цьому вважають, що кожна елементарна операція виконується за фіксовану кількість часу [1].

Гострою проблемою під час оцінки складності алгоритму є відсутність єдиного алгоритму, здатного визначити складність певної ділянки коду/методу/алгоритму. Переважна більшість систем використовують просте заміреня часу, затраченого на виконання задачі, що є неточної метрикою, оскільки одна й та ж програма на різних комп'ютерах може виконуватись різну кількість часу. У разі відповідної потреби аналіз проводиться мануально інженерами програмного забезпечення.

Таким чином, є потреба в автоматизації мануальних процесів та реалізація автоматизованого способу оцінки складності програмного коду.

Аналіз основних досліджень

Зазвичай складність алгоритму в комп'ютерних науках оцінюють на основі роботи звичайного лексичного аналізатора (ЛА) [2]. Основою даної роботи є реалізація самописного ЛА, задачею якого є підтримка лексем, що використовуються при підрахунку кількості операцій програми та можливість обробки їх, тобто визначення кількості операцій.

В залежності від розташування інструкції коду, зокрема в залежності від вкладеності оператору циклу, кількість операцій може рахуватися по різному. Виникає задача проаналізувати найбільш підходящі структури даних, що можна використати для зберігання вкладеностей лексем. Однією із таких структур можуть розглядатися регулярні вирази [3]. Для пошуку необхідних лексем необхідно створити їх регулярні визначення та реалізувати їх у вигляді регулярих виразів в системі. В ході дослідження необхідно проаналізувати обмеження, які накладаються на лексичний алгоритм підрахунку кількості операцій, а також проаналізувати спосіб підрахунку складності у разі знаходження у коді обмежуючих конструкцій, через які неможливо порахувати кількість операцій. Цю проблему можуть допомогти вирішити часові алгоритми [4]. Отже, треба дослідити спосіб точного визначення складності програмного коду з використанням часового алгоритму та врахувати такі складові, як: кількість замірів, розміри вхідних даних та їх розміщення, тощо.

Постановка задачі

Отже, на основі апарату регулярних виразів необхідно розробити математичну модель системи оцінки складності програмного коду. На її основі необхідно розробити алгоритм лексичного та часового аналізу, який повинен аналізувати написаний код, розбиваючи його на лексеми та підраховуючи кількість операцій програми. У випадку, якщо роботу програми неможливо оцінити типовим лексичним аналізом з причин наявності операторів умовного переходу, рекурсії, тощо, необхідно використати для оцінки розширення, яке дозволить виконати написану програму з різними наборами вхідних даних та проаналізувати графік росту функції часу від розміру вхідних даних, що забезпечить велику точність оцінки. Розроблені модель та алгоритм необхідно реалізувати у складі відповідного програмного додатку.

Математичне моделювання системи оцінювання складності коду

Основними об'єктами предметної області оцінювання складності алгоритму є оцінюваний алгоритм; набір лексем, які є складовими частинами алгоритму; лексичний аналізатор, на основі якого реалізовується лексичний алгоритм; лексичний алгоритм оцінювання складності коді; лексична граматика; обмеження на код для коректної роботи алгоритму; часовий алгоритм оцінювання складності коду та набір найпоширеніших складностей.

Отже, система оцінювання складності програмного коду може бути представлена, як:

$$LCA = < A, L, LA, RT, LG, C, TCA >,$$

- де A – алгоритм, що підлягає оцінюванню;
- L – набір лексем, які є складовою будь-якого коду;
- LA – лексичний аналізатор;
- RT – обмеження, що накладаються на оцінюваний лексичним алгоритм алгоритм (код) ;
- LG – лексична граматика;
- C – набір усіх можливих складностей коду [5];
- TCA – часовий алгоритм оцінювання складності алгоритму.

В свою чергу, алгоритм A ,даний для оцінювання, можна описати як:

$$A = \{L_i\}_{i=1}^{n},$$

де L_i – це лексема, які є основою та складовою частиною будь-якої програми і яка буде розпізнаватися лексичним аналізатором.

Лексичний аналізатор LA може бути описаний як:

$$LA = < T, LG, RD >,$$

де T – набір токенів, отриманих з розпізнаних лексем;

LG – лексична граматика, тобто набір правил формальної граматики, що визначають синтакс токенів;

RD – регулярні визначення, що визначають можливу структуру валідного токену. Регулярні визначення можна описати всіма можливими метасимволами та алфавітами.

В свою чергу, лексичну граматику $LG = \{R_j\}_{j=1}^{m}$, можна описати як набір правил, де R_j – правило, яке описується регулярними виразами, що задаються для даної системи, а саме:

$$R_j = < primitiveType, identifier, equalOperator,$$

$$binaryOperator, unaryOperator, intNum, loop >,$$

де кожному з правил буде відповідати описаний регулярний вираз.

RD – регулярні визначення, що визначають можливу структуру валідного точену, можна описати всіма можливими метасимволами та алфавітами.

Обмеження системи можна описати як:

$$RT = < CO, LS, RC, IL >,$$

де CO – оператори умовних переходів, які є основною причиною неможливості оцінювання алгоритму лексичним аналізом;

LS – ярликові вирази, тобто оператори, що використовуються для непослідовної зміни номеру інструкції, що неможливо передбачити;

RC – рекурсивні виклики методу або виклики зовнішніх методів;

IL – безкінечні цикли.

Набір складностей C можна описати наступною наступною п'ятіркою:

$$C = < O(1), O(\log n), O(n), O(n^2), O(n^3) >,$$

де O(1) – складність – константа, наприклад, операція додавання чисел може вважатися константою;

O(log n) – складність логарифмічна, наприклад, пошук елемента у бінарному дереві пошуку;

O(n) – складність лінійна, наприклад, пошук елемента у невідсортованому масиві простим перебором елементів.

Опис алгоритму оцінювання складності коду

На основі розробленої моделі було розроблено алгоритм (схему алгоритму дивись на рис. 1).

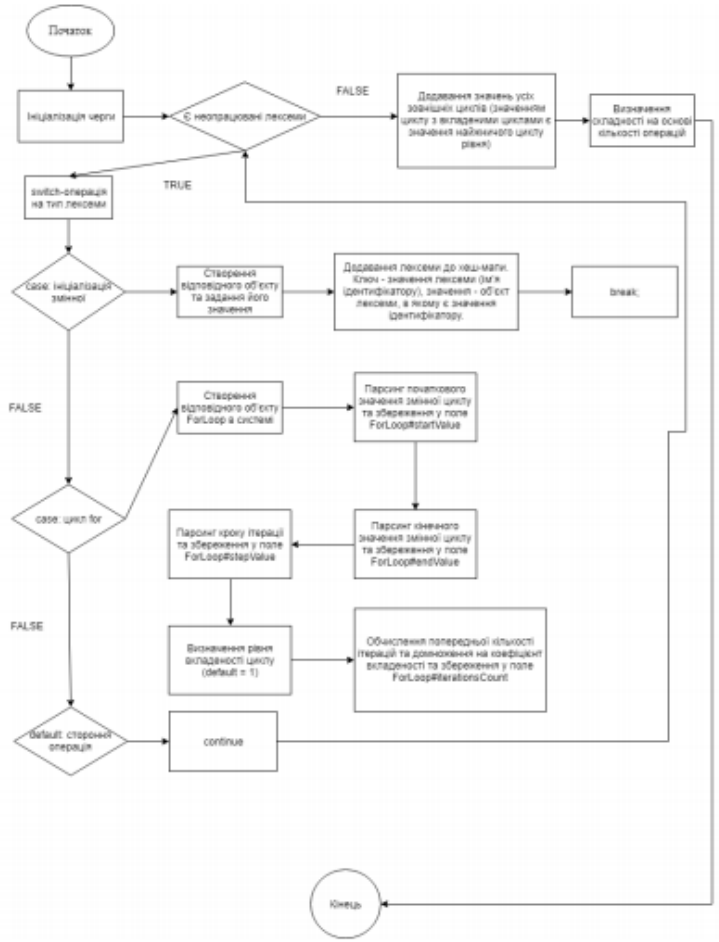

Рисунок 1 – Схема адаптованого алгоритму оцінювання складності коду

Основними етапами розробленого алгоритму виступають:

- ініціалізація необхідних структур даних для зберігання проміжних результатів роботи програми;
- циклічна обробка усіх лексем коду – тобто пошук змінних та циклів (основних лексем, що впливають складність програми) за допомогою використання регулярних виразів, що відповідають за перевірку синтаксису структур;
- калькуляція кількості ітерацій за допомогою пошуку рівня вкладеності циклів;
- в разі знаходження лексем, що описують інструкції з використанням умовних операторів – виконується часовий аналіз на базі кількісного заміряння часу виконання програми на різних розмірах даних;
- визначення складності коду на основі росту часу виконання програми в залежності від росту розмірів вхідних даних для часового алгоритму або на основі кількості операцій для лексичного алгоритму.

Розроблений алгоритм може використовуватися для аналізу складності довільної вхідної програми, проте при наявності операторів умовного переходу, вічних циклів або операторів безумовного переходу, тобто при наявності обмежень, що накладаються на аналіз лексичного алгоритму – аналіз програми передається часовому алгоритму, який працює досить точно, проте для забезпечення цієї точності калькуляції необхідно жертвувати часовими ресурсами серверу, оскільки потрібно велику кількість разів запускати алгоритм на вхідних даних різних розмірів, що збільшує точність оцінки, проте є досить часозатратною операцією.

Висновки та перспективи

Під час дослідження було проаналізовано основні етапи створення алгоритмів оцінки складності програмного коду; досліджено причини, через які неможливо створити універсальний лексичний алгоритм для довільної програми; проаналізовано набори задач, які задовільняють умовам оцінювання лексичним алгоритмом. На основі проведеного аналізу розроблено математичну модель системи оцінювання програмного коду та на її основі розроблено алгоритм оцінки складності коду, що покладено в основу вдосконаленого лексичного аналізатора. В запропонованому лексичному аналізаторі було враховано відповідні лексеми, способи їх обробки та підрахунку кількості операцій. Було проаналізовано типові складності алгоритмів для мапінгу їх на результуючі дані.

Для наборів задач, що містять в собі конструкції, які неможливо оцінити лексичним алгоритмом, було вирішено використовувати часовий алгоритм, який заміряє час виконання програми на вхідних даних різного розміру і отримує формулу складності. Було проведено аналіз вхідних, на яких треба тестувати складність часовим алгоритмом. В ході дослідження було прийнято рішення використовувати подібні дані, тобто або всі вхідні дані в рамках однієї оцінки повинні бути відсортовані, або усі довільні, що збільшить точність оцінювання. Функціонал лексичного алгоритму можна доповняти більш складними конструкціями, підтримкою різних операторів циклу та посилань на зовнішні методи.

Розроблена математична модель та алгоритм було покладено в основу програмного додатку «Algorithms battle», який використовується для вирішення алгоритмічних задач для турнірів та олімпіад різного рівня. Додаток було розроблено за домогою мови програмування Java, з використанням фреймворку Spring Boot.

Список літератури

1. HackerNoon // HackerNoon. – URL: https://hackernoon.com/lexical-analysis-861b8bfe4cb0 (дата звернення: 22.10.2019).
2. ScienceDirect // ScienceDirect. – URL: https://www.sciencedirect.com/topics/computer-science/lexical-analysis (дата звернення: 24.10.2019).
3. Downey, A. Think Complexity, 113 с. – 2012.
4. Wilf, H. Algorithms and complexity, 187 с. – 1986.
5. Leeuwen, J. Handbook of Theoretical Computer Science, Vol. A, 212 с. – 1990.
6. Schmuki, R., Wagner, F., Wagner, T. Modeling Software with Finite State Machines: A Practical Approach, 47 с. – 2006.

ДОДАТОК В

Відгук керівника роботи

ВІДГУК

на атестаційну роботу магістра

Різника Олександра Костянтиновича, гр. ПЗСм-18-1,

(спеціальність – **121- Інженерія програмного забезпечення,**

освітньо-професійна програма – **Програмне забезпечення систем**)

Тема атестаційної роботи «Дослідження методів лексичного аналізу для оцінки складності алгоритмів в системі «Algorithms battle»».

Структура атестаційної роботи: пояснювальна записка 91 сторінка; 5 розділів; 4 додатки; графічна частина 17 слайдів.

В магістерській атестаційній роботі розглядається досить актуальна на даний час проблема визначення складності програмних алгоритмів методом аналізу коду.

В першому розділі проведено аналіз проблемної області та розроблено постановку задачі. Дано обґрунтування актуальності досліджуваної теми. В другому розділі сформульовано вимоги до програмної системи. В третьому розділі розглянуто популярні підходи визначення ефективності програмного коду та проведено аналіз існуючих підходів. Описані сильні та слабкі сторони кожного з підходів. Розроблено лексичний алгоритм з використанням елементів часового алгоритму. В четвертому розділ наведено опис програмної реалізації системи аналізу коду. В п'ятому розділі наведено результати тестування системи.

Магістрант провів аналіз технічної та спеціальної літератури, використав отриману інформацію для обґрунтування прийнятих в роботі науково-дослідних рішень. Записка написана грамотно, вимоги стандартів дотримані. Результати роботи досить повно відображені в пояснювальній записці та на слайдах презентації. За результатами атестаційної роботи зроблено підготовлено наукову статтю.

Вважаю, що магістрант гр. ПЗСм-18-1 Різник О. К. готовий до самостійної інженерної діяльності. Атестаційну роботу можна подати до захисту в ЕК за спеціальністю 121- «Інженерія програмного забезпечення» (освітньо-професійна програма «Програмне забезпечення систем»).

Керівник атестаційної роботи магістра
к.т.н., доц. каф. ПІ Мазурова О.О.

«16» 12 2019 р.

ДОДАТОК Г

Зовнішня рецензія

Рецензія
на атестаційну роботу магістра 69
студента групи ПЗСм-18-1 Різника Олександра Костянтиновича
(спеціальність – 121- Інженерія програмного забезпечення,
освітньо-професійна програма – Програмне забезпечення систем)
Тема роботи: «Дослідження методів лексичного аналізу для оцінки складності алгоритмів в системі «Algorithms battle»»

Структура атестаційної роботи: пояснювальна записка 70 сторінок, графічна частина 15 слайдів.

У зв'язку з постійним збільшенням кількості програмних продуктів в повсякденному житті змінюються і вимоги до темпів розробки програмного забезпечення. Швидкі темпи розробки впливають на якість програмного продукту. Тому задача визначення складності програмних алгоритмів є досить актуальною. Вона щільно пов'язана з повсякденною роботою інженерів програмного забезпечення та має високий вплив на якість програмного продукту. В роботі розглядається така важлива задача як аналіз коду з використанням вдосконаленого лексичного аналізатору.

В першому розділі пояснювальної записки проведено аналіз предметної області та проаналізовано існуючі методи вирішення даної проблеми. В другому розділі наведено вимоги до програмної системи. В третьому розділі наведено математичну модель системи оцінки складності коду, розроблені лексичний та часовий алгоритми оцінки коду. В четвертому розділі наведено програмну реалізацію системи «Algorithms battle», де використовуються розроблені алгоритми. В п'ятому розділі проведено результати тестування програмної системи.

Магістрант опрацював науково-технічну та спеціальну літературу. Результати роботи наочно і досить повно відображені в пояснювальній записці та на слайдах презентації. Проект є актуальним і має практичну спрямованість. Пояснювальна записка написана грамотно, вимоги стандартів дотримані.

До зауважень можна віднести те, що в розробленій математичній моделі задано невелику множину лексем, що накладає певні обмеження на роботу лексичного аналізатора.

Вважаю, що магістрант гр. ПЗСм-18-1 Різник О. К. готовий до самостійної інженерної діяльності. Атестаційна робота відповідає вимогам до атестаційних робіт магістрів та заслуговує оцінку «добре (85)». Атестаційну роботу можна подати до захисту в ЕК за спеціальністю 121- «Інженерія програмного забезпечення» (освітньо-професійна програма «Програмне забезпечення систем»).

Рецензент к. т. н., доцент, проф. кафедри СТ
Н. І. Клименко 18.12.19

ДОДАТОК Ж

Внутрішня рецензія

Рецензія

на атестаційну роботу магістра

студента групи **ПЗСм-18-1 Різника Олександра Костянтиновича** 70

(спеціальність **– 121- Інженерія програмного забезпечення,**

освітньо-професійна програма – **Програмне забезпечення систем**)

Тема роботи: «Дослідження методів лексичного аналізу для оцінки складності алгоритмів в системі «Algorithms battle»»

Структура атестаційної роботи: пояснювальна записка 70 сторінок, графічна частина 13 слайдів, 5 розділів та 5 додатків..

На сьогодні, основною мірою оптимальності алгоритму є його часова складність. Гострою проблемою при цьому є відсутність єдиного алгоритму, здатного визначити складність певної ділянки коду алгоритму. Переважна більшість систем використовують просте заміреннія часу, затраченого на виконання задачі, що є неточної метрикою, оскільки одна й та ж програма на різних комп'ютерах може виконуватись різну кількість часу. У разі відповідної потреби аналіз проводиться мануально інженерами програмного забезпечення. Таким чином, є потреба в реалізації автоматизованого способу оцінки складності програмного коду.

Робота присвячена дослідженню способів точної оцінки складності програмного коду на основі його лексичного аналізу та часових замірів.

В першому розділі роботи проведено аналіз проблемної області та розроблено постановку задачі, дано обґрунтування актуальності. В другому розділі сформульовано вимоги до програмної системи. В третьому розділі проведено дослідження основних підходів до визначення ефективності програмного коду, проведено математичне моделювання та розроблено лексичний алгоритм з використанням часових замірів. В четвертому розділ наведено опис програмної реалізації системи аналізу коду. В п'ятому розділі наведено результати тестування системи.

Магістрант опрацював науково-технічну та спеціальну літературу. На основі проведених досліджень та експериментів описав сильні та слабкі сторони опрацьованих підходів. Представлена магістерська атестаційна робота за змістом відповідає темі. Пояснювальна записка магістра викладена у логічній послідовності, написана грамотно, якість оформлення – висока, вимоги стандартів дотримані. Результати роботи наочно і досить повно відображені на слайдах презентації та в демонстраційному ролику програмної системи.

До зауважень можна віднести те, що під час дослідження використовувався не досить великий набір шаблонів для визначення неправильних алгоритмів чи помилок в коді.

Вважаю, що магістрант гр. ПЗСм-18-1 Різник О. К. готовий до самостійної інженерної діяльності. Атестаційна робота відповідає вимогам до атестаційних робіт магістрів та заслуговує оцінки «добре (80)». Атестаційну роботу можна подати до захисту в ЕК за спеціальністю 121-Інженерія програмного забезпечення» (освітньо-професійна програма «Програмне забезпечення систем»).

Рецензент

доцент каф. ПЗ Голан В.В.