**ТЕХНОЛОГІЇ РОЗРОБКИ ТА СУПРОВОДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

УДК 004.413(045)

**Костів М.М.**
**Національний авіаційний університет**

# ЛЕКСИЧНИЙ АНАЛІЗАТОР ДЛЯ РОЗРОБКИ СТИЛЮ ЕФЕКТИВНОГО ПРОГРАМУВАННЯ

*У статті розглянута задача створення лексичного аналізатору для розробки стилю ефективного програмування, наведені результати експериментів з вибору задач, які найчастіше виникають у веб-програмуванні, виконаний аналіз проектів з відкритим кодом, реалізована архітектура засобу і діаграма класів. На основі таблиць та гістограм, які отримані за допомогою «Лексичного аналізатору» створена таблиця з лексемами, які можуть бути використані в конструкціях для реалізації вирішення задач і підтверджувати їх популярність на основі значної частоти появи у коді. Після виконання лексичного аналізу і обрання найпопулярніших задач виникає можливість аналізу варіантів їх вирішення, які представлені у вигляді конструкцій і порівняння їх ефективності для створення стилю. Для дослідження коду мови програмування PHP з метою отримання на виході послідовності символів лексем використаний «Zend Engine PHP-Lexer». Інструмент створений без роботи з мовною специфікацією на лексичному рівні за допомогою використання лексичних функцій, які вбудовані у «Zend Engine PHP-Lexer».*

*В статье рассмотрена задача создания лексического анализатора для разработки стиля эффективного программирования, представлены результаты экспериментов по выбору задач, которые наиболее часто возникают в веб-программировании, выполнен анализ проектов с открытым кодом, реализована архитектура инструмента и диаграмма классов. На основе таблиц и гистограмм, полученных с помощью «Лексического анализатора» создана таблица с лексемами, которые могут быть использованы в конструкциях для реализации решения задач и подтверждать их популярность на основе значительной частоты появления в коде. После выполнения лексического анализа и выбора самых популярных задач появляется возможность анализа вариантов их решения, которые представлены в виде конструкций и сравнения их эффективности для создания стиля. Для исследования кода языка программирования PHP с целью получения на выходе последовательности символов лексем использован «Zend Engine PHP-Lexer». Инструмент создан без работы с языковой спецификацией на лексическом уровне посредством использования лексических функций, которые встроены в «Zend Engine PHP-Lexer».*

*In the article the task of creation of lexical analyzer for development of style of effective programming is considered, the results of experiments for choosing tasks, which appear in the web-programming was represented, the analyses of the open-source projects was made, the architecture of the tool and class diagram was created. The table with lexemes' was created on the base of the tables and histograms obtained by the "Lexical analyzer", that can be used in constructions for the implementation of the tasks and confirm their popularity through large frequency of occurrence in the code. After performing of the lexical analysis and selection of the most popular tasks, it is possible to analyze solutions of these tasks, which can be represented as constructions and compare their performance to create a style. For research of the code on the PHP language in order to obtain the output sequence of symbols lexemes' «Zend Engine PHP-Lexer» was used. The tool was created without working with the language specification at the lexical level by means of using lexical functions that are included into «Zend Engine PHP-Lexer».*

***Ключові слова:*** *лексичний аналіз, лексема, стиль ефективного програмування, конструкція мови програмування.*

**Постановка проблеми**

Використання стилю ефективного програмування допомагає збільшити продуктивність роботи програмного забезпечення на існуючих ресурсах без доповнення або заміни апаратної частини [1].

Продуктивність програмного забезпечення може бути збільшена шляхом зменшення часу виконання коду або пам'яті, яка необхідна програмі. Стиль ефективного програмування розроблений з метою зменшення часу виконання програми.

З підвищенням ефективності програмного забезпечення можливо знизити мінімальні вимоги до апаратного забезпечення та втрати, тобто отримати зелену програму [2].

Для створення стилю програмування необхідно виконати наступні загальні кроки наукового методу [3]: побудова гіпотез, проведення експерименту, розробка стилю.

Гіпотези необхідно формувати на основі задач, які виникають під час написання коду програмного забезпечення і можуть бути вирішені за допомогою конструкцій [4], які містять в собі одну або більше лексем.

Відповідно, потрібно виділити тільки ті задачі, які найчастіше виникають при розробці програмного забезпечення шляхом аналізу лексем. Кожна задача може мати декілька алгоритмів вирішення і може бути реалізована з використанням різних конструкцій.

Після виконання лексичного аналізу і обрання найпопулярніших задач виникає можливість аналізу варіантів їх вирішення, які представлені у вигляді конструкцій і порівняння їх ефективності для створення стилю.

**Розв'язання проблеми**

Для дослідження коду, який ґрунтується на аналізі вхідної послідовності символів (код на мові програмування PHP) з метою отримання на виході послідовності символів лексем необхідно використати «Zend Engine PHP-Lexer» [5].

Лексичні функції, які вбудовані у мову програмування PHP дають змогу отримати доступ до «Zend Engine PHP-Lexer» і можливість створити інструмент для аналізу коду без необхідності роботи з мовною специфікацією на лексичному рівні.

Для виконання лексичного аналізу за допомогою розробленого інструменту «Лексичний аналізатор» необхідно надати архів проекту з кодом на мові програмування PHP для формування результатів у вигляді таблиці або гістограми лексем, які найчастіше з'являються у вхідному коді ( Рис. 1).

Після аналізу результатів стає можливим виділити з усіх лексем мови програмування найпопулярніші відносно появи у коді програмного забезпечення.

Відповідно, після визначення лексем, які найчастіше використовують при створенні програм стає можливим здійснити відбір найпопулярніших конструкцій і задач, оскільки конструкції містять в собі лексеми, а задачі можуть бути реалізовані за допомогою конструкцій мови програмування.

Для завантаження архіву проекту для аналізу в інструменті «Лексичний аналізатор» створений інтерфейс, який зображено на рис. 2.

Інструмент рекурсивно аналізує всі файли з кодом в архіві і надає можливість формувати результати залежно від частоти появи лексем відносно всього проекту.

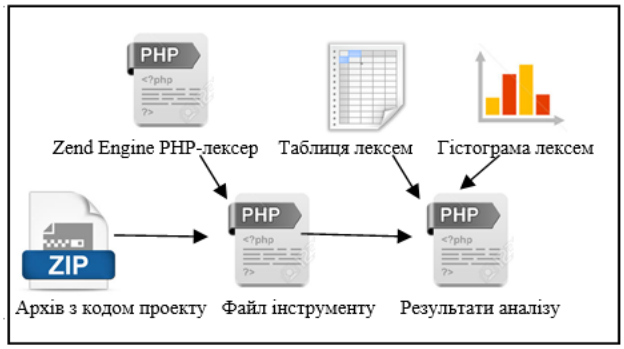


Рис. 1 Архітектура засобу «Лексичний аналізатор»

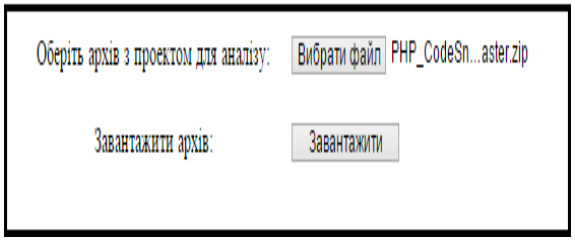


Рис. 2 Інтерфейс, що використовується для завантаження архіву проекту

Діаграма класів інструменту зображена на рис. 3

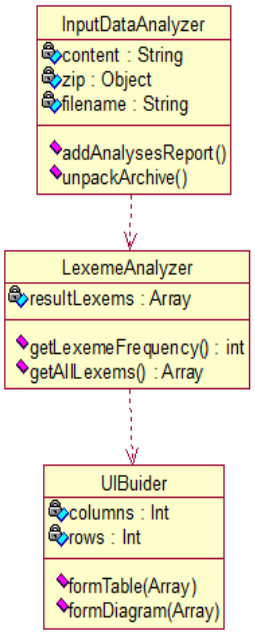


Рис. 3 Діаграма класів лексичного аналізатору

Результати аналізу проекту з відкритим кодом «PhpMyAdmin» [6] представлені у таблиці (Табл.1) і у вигляді гістограми (Рис. 4).

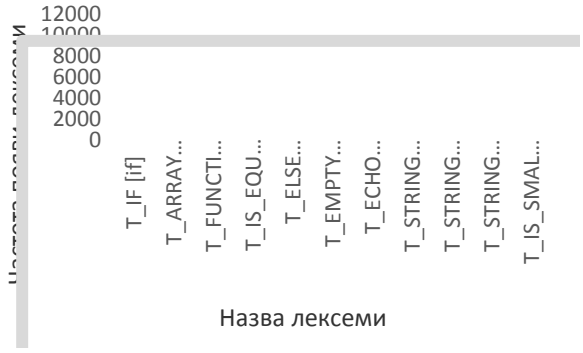


Рис. 4 Гістограма аналізу лексем проекту «PhpMyAdmin»

*Таблиця 1*
*Результат аналізу лексем проекту «PhpMyAdmin»*

| Назва лексеми | Частота появи у коді |
|---|---|
| T_IF [if] | 10147 |
| T_CONCAN_EQUAL [.=] | 8675 |
| T_ARRAY [array] | 4736 |
| T_RETURN [return] | 3874 |
| T_FUNCTION [function] | 3540 |
| T_ISSET [isset] | 3397 |
| T_IS_EQUAL [==] | 3069 |
| T_STRING [false] | 2379 |
| T_ELSE [else] | 2606 |
| T_STRING [true] | 1907 |
| T_EMPTY [empty] | 1467 |
| T_BREAK [break] | 1460 |
| T_ECHO [echo] | 1315 |
| T_FOREACH [foreach] | 1146 |
| T_STRING [null] | 1053 |
| T_IS_IDENTICAL [===] | 649 |
| T_STRING [count] | 489 |
| T_FOR [for] | 437 |
| T_STRING [strlen] | 227 |
| T_STRING [addHTML] | 217 |

Лексеми «T_IF[if]», «T_STRING [true]», T_IS_EQUAL [==] зустрічаються у коді проекту «PhpMyAdmin», відповідно, 10147, 1907, 3069 разів і необхідні для вирішення задачі «Порівняння зі значенням TRUE». На основі аналізу лексем можливо сформувати конструкції для вирішення задачі.

*Конструкція № 1:*

```
$var = true;
if (true == $var){
print 1;
}
```

Конструкція № 2:

```
$var = true;
if ($var) {
  print 1;
}
```

Кожна з конструкцій може мати різну ефективність з точки зору швидкості їх виконання.

*Таблиця 2*
*Результат аналізу лексем проекту «PHP_CodeSniffer»*

| Назва лексеми | Частота появи у коді |
|---|---|
| T_IF [if] | 3693 |
| T_IS_IDENTICAL[===] | 3060 |
| T_STRING [true] | 1883 |
| T_ARRAY [array] | 1202 |
| T_STRING [false] | 1152 |
| T_IS_NOT_IDENTICAL[!==] | 991 |
| T_RETURN [return] | 947 |
| T_ELSE [else] | 800 |
| T_ECHO [echo] | 709 |
| T_FUNCTION [function] | 697 |
| T_STRING [null] | 566 |
| T_ISSET [isset] | 527 |
| T_BOOLEAN_AND [&&] | 476 |
| T_BREAK [break] | 303 |
| T_FOREACH [foreach] | 250 |
| T_FOR [for] | 238 |
| T_STRING [strlen] | 215 |
| T_CONCAT_EQUAL [.=] | 185 |
| T_STRING [strtolower] | 127 |
| T_STRING [PHP_CodeSniffer] | 111 |
| T_STRING [count] | 104 |

Лексеми «T_IF[if]», «T_STRING [strlen]», зустрічаються у коді проекту «PhpMyAdmin», відповідно, 10147 і 227 разів і необхідні для вирішення задачі «Перевірка довжини рядка».

Результати аналізу проекту з відкритим кодом «PHP_CodeSniffer» [7] представлені у таблиці (Табл. 2) і у вигляді гістограми (Рис. 5).

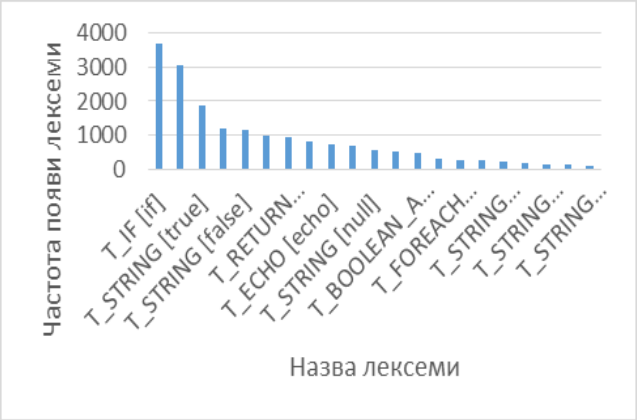


Рис. 5 Гістограма аналізу лексем проекту «Php_CodeSniffer»

Отже, на основі таблиць та гістограм, які отримано за допомогою «Лексичного аналізатору» після аналізу багатьох проектів можливо створити таблицю з лексемами, які можуть бути використані в конструкціях для реалізації вирішення задач і підтверджувати їх популярність на основі значної частоти появи у коді (Табл. 3).

*Таблиця 3*
*Задачі і лексеми для їх вирішення*

| Задача | Назви лексем |
|---|---|
| Порівняння зі значенням true | T_STRING [true], T_IS_EQUAL [==], T_IF [if]). |
| Виведення рядка | T_ECHO [echo]. |
| Перевірка довжини рядка | T_STRING [strlen]), T_IF [if]. |
| Перевірка на рівність значення змінної NULL | T_STRING[NULL], T_IF [if]. |
| Прохід по асоціативному масиву і виведення кожного значення | T_FOREACH[foreach], T_FOR [for], |
| Перевірка масиву на пустоту | T_EMPTY [empty], T_STRING [count]), T_IF [if]. |
| Перевірка існування елементу асоціативного масиву за індексом | T_ISSET[isset], T_IF [if]), T_ARRAY [array]). |

Задачі з таблиці 3 можливо використовувати у більших масштабних задачах для створення важливих частин веб-застосувань, наприклад, форми реєстрації або авторизації (Рис. 6).

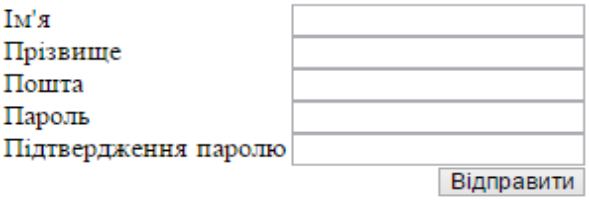


Рис. 6 Форма реєстрації або авторизації

Користувач заповнює форму у браузері і відправляє дані на сервер, де вони будуть оброблені за допомогою мови програмування PHP і занесені до асоціативних масивів $_POST або $_GET. Обробка містить в собі вирішення задачі перевірки введених даних на їх заповнення і коректність. Користувач міг забути заповнити деякі поля або зробити помилку. Так поле «Пошта» має обов'язково містити рядок довжиною не менше 8 символів, а користувач вводить лише 2 символи.

Отже, задача «Реєстрація та авторизація користувача у системі» містить в собі простіші задачі: «Перевірка існування елементу асоціативного масиву за індексом», «Перевірка довжини рядка», «Перевірка масиву на пустоту», «Виведення тексту».

Приклад коду форми з використанням вирішення простіших задач з табл.3:

```
// Перевірка існування елементу асоціативного масиву
if (isset($_POST['name'])) {
$name = $_POST['name'];
}
if ($name == '') {
unset($name);
}
// Перевірка існування елементу
if(isset($_POST['email'])){
$email=$_POST['email'];
}
if ($email =='') {
unset($email);
}

//Перевірка на пустоту
if (empty($_POST[‘name’) or $_POST[‘email’))
//Виведення тексту
echo “Ім’я або пошту не занесно до форми”
}
//Перевірка рядка на довжину
if(strlen($email)<8){
echo “Довжина поля «Пошта» не може бути < 8 символів”
}
```

**Висновки**

Для створення стилю ефективного програмування необхідно виконати відбір задач за критерієм частоти їх появи у розробці веб-засосувань.

Аналіз лексем у різних проектах з відкритим кодом надає можливість відслідковувати частоту появи задач і обирати найпопулярніші задачі при створенні стилю ефективного програмування.

Лексичний аналізатор можливо створити для будь-якої мови програмування і дослідження коду, що ґрунтується на аналізі вхідної послідовності символів з метою отримання на виході послідовності символів лексем і їх аналізу.

Існують такі інструменти лексичного аналізу, як «Lex» чи «Flex», «Yacc» чи «Bison», що можуть бути використані при створенні власних аналізаторів в інших мовах програмування.

**Список використаних джерел**

1. Магда Ю.С. Ассемблер. Разработка и оптимизация Windows-приложений / Ю.С. Магда. – С-Пб.: БХВ-Петербург, 2003. – 544 с.

2. Сидоров Н.А. Экология программного обеспечения // Інженерія програмного забезпечення. – 2010. – №1. – С.53 – 61.

3. Сидоров М.О., Костів М.М.. Метод створення ефективного стилю програмування // Інженерія програмного забезпечення – 2013. – № 3–4 (15–16) – С. 17–24.

4. Костів М.М., Крамар Ю.М., Інструмент для створення стилю ефективного програмування // Інженерія програмного забезпечення. – 2014. – № 1 (17) – С. 28 – 31.

5. PHP: Лексер (Tokenizer) - Manual [Електронний ресурс] – Режим доступа: http://php.net/manual/ru/book.tokenizer.php

6. PhpMyAdmin [Електронний ресурс] – Режим доступа: http://www.phpmyadmin.net/home_page/index.php

7. Package Information: PHP_CodeSniffer [Електронний ресурс] – Режим доступа: http://pear.php.net/package/PHP_CodeSniffer/redirected

.

**Відомості про автора:**

**Костів Мілана Миколаївна –** аспірантка кафедри інженерії програмного забезпечення Інституту комп’ютерних інформаційних технологій Національного авіаційного університету. Наукові інтереси: інженерія програмного забезпечення.

**E-mail:** milana.kostiv@livenau.net