# Теория автоматов и формальных языков

## Лабораторная работа №5

# Создание лексического анализатора

**Лексический анализ** – процесс разбиения строки на отдельные токены по некоторым признакам и отнесение токенов к тем или иным типам.

**Лексер** – устройство, осуществляющее лексический анализ.

Лексический анализ начинается с выделения лексем.

**Лексема** – это последовательность символов входной строки, подходящая под паттерн **токена**.

**Токен** – это пара название-значение:

* в поле «Название» записывается тип токена,
* в поле «Значение» – найденная лексема.

Для каждой найденной лексемы лексер создает **токен**.

В таблице ниже приведены примеры токенов и их значений:

|  |  |
| --- | --- |
| Имя токена | Примеры значений токенов |
| identifier | x, color |
| keyword | **if**, **while**, **return** |
| separator | }, (, ; |
| operator | **+, <, =** |
| literal | true, 6.02e23, "music" |
| comment | */\* Текст комментария\*/*,  *// текст комментария* |

**Задание на лабораторную работу:**

1. Для вашего любимого языка программирования составить таблицу токенов аналогично таблице выше. Таблица может не быть исчерпывающей.
2. Для каждого типа токенов придумать и описать паттерн, которому соответствуют значения такого токена. Для этих целей можно использовать регулярные выражения или другой способ, кажущийся вам более подходящим.
3. Написать программу, которая будет получать на вход строку на вашем любимом языке программирования, а на выход выдавать последовательность токенов (с указанием как имени токена, так и его значения).

Важно! На этапе лексического анализа не проверяется корректность поданной на вход строки, только разбиение ее на токены.

**На защите лабораторной работы необходимо** показать таблицу токенов и работу программы.

**В отчете надо привести:**

* Таблицу токенов.
* Исходный код программы.
* Скриншоты работы программы.

**Пример**

Для для запросов SELECT в языке SQL можно предложить следующую структуру токенов (не исчерпывающий список):

|  |  |  |
| --- | --- | --- |
| Имя токена | Примеры значений токенов | Шаблон токена |
| keyword | Select, all, distinct, from, where, group by, order by, or, and, not, exists, like, asc, desc | \b(?:select|all|distinct|from|  where|group by|order by|  or|and|not|exists|like|asc|desc)\b |
| separator | , ( ) | , |
| operator | = + - \* / < > <= >= <> | ([<>]=|<>|[=+\-\*<>/()]) |
| identifier | table, name, employee, telephone, … | [A-Za-z][A-Za-z0-9]\* |
| int number | 1, 2, 3, 198 | [0-9]+ |
| real number | 0.0001, -198.67 | ([0-9]+\.[0-9]\*| [0-9]\*\.[0-9]+) |
| space |  | \s+ |
| string | ' …. ' | '[^']\*' |
| comment | /\* …\*/ -- … | --[^\n\r]\*|\/\\*.\*?\/\\* |

В программе по распознаванию лексем очень важна очередность проверки. Так, сначала важно выделить токены типа **comment**, чтобы их содержимое не распознавалось как другие виды токенов. Токены **keyword** следует выделять до токенов **identifier**, чтобы они определились именно как ключевые слова, а не как все остальное. Аналогично, токены **real number** стоит выделять до токенов **int number**. Также рекомендуется проводить проверки без чувствительности к регистру букв (например, применяя для регулярных выражений флаг i – case insensitive).

Если на вход программы подается строка «SELECT \* FROM students WHERE group LIKE 'ИКПИ' AND averagemark = 5.0 --поиск программистов-отличников», то на выходе должен быть массив вида

SELECT keyword

\* operator

FROM keyword

students identifier

WHERE keyword

group identifier

LIKE keyword

'ИКПИ' string

AND keyword

averagemark identifier

= operator

5.0 real number

(пробелы и комментарии возможно опустить как не несущие дальнейшей нагрузки).