

# Лабораторная работа №3

## «Лексический анализатор на основе регулярных выражений»

Скоробогатов С.Ю., Коновалов А.В.

16 марта 2016

### 1 Цель работы

Целью данной работы является приобретение навыка разработки простейших лексических анализаторов, работающих на основе поиска в тексте по образцу, заданному регулярным выражением.

### 2 Исходные данные

Стандартная библиотека любого современного языка программирования содержит средства для поиска в тексте образцов, заданных регулярными выражениями. При этом используется расширенный синтаксис записи регулярных выражений, позволяющий по сути выйти за рамки регулярных языков. Механизм поиска по таким регулярным выражениям годится для написания простейших лексических анализаторов. Однако, для этого механизма характерна нелинейная зависимость времени работы от длины распознаваемой лексемы, поэтому в промышленных компиляторах он не используется.

В качестве языка реализации в данной лабораторной работе выберем язык Java, стандартная библиотека которого содержит пакет `java.util.regex`, в котором располагаются классы `Pattern` и `Matcher`, предназначенные для поиска по регулярным выражениям. Документация по этому пакету находится по адресу:

<http://docs.oracle.com/javase/7/docs/api/java/util/regex/package-summary.html>.

Вводную статью по синтаксису регулярных выражений можно прочитать здесь:

<http://www.quizful.net/post/Java-RegExp>.

Идея лексического анализа на основе поиска по регулярным выражениям состоит в использовании групп, представляющих собой фрагменты регулярных выражений, заключённые в круглые скобки, значения которых запоминаются при сопоставлении текста с образцом. Например, на листинге 1 показано, как с использованием групп отличить идентификаторы от числовых литералов.

### 3 Задание

В лабораторной работе необходимо реализовать на языке Java две первые фазы стадии анализа: чтение входного потока и лексический анализ. Чтение входного потока должно осу-

---

**Алгоритм 1** Распознавание идентификаторов и числовых литералов

---

```
1  import java.util.regex.Matcher;
2  import java.util.regex.Pattern;
3
4  // Компиляция для Windows:
5  // javac -encoding utf8 IdentVsNumber.java
6
7  public class IdentVsNumber
8  {
9      public static void main(String args[])
10     {
11         test_match("Альфа123");
12         test_match("42");
13     }
14
15     public static void test_match(String text)
16     {
17         // Регулярные выражения
18         String ident = "\\p{L}[\\p{L}0-9]*";
19         String number = "[0-9]+";
20         String pattern = "(^"+ident+")|(^"+number+)";
21         // Компиляция регулярного выражения
22         Pattern p = Pattern.compile(pattern);
23         // Сопоставление текста с регулярным выражением
24         Matcher m = p.matcher(text);
25         if (m.find()) {
26             if (m.group(1) != null) {
27                 System.out.println("Идентификатор_" + m.group(1));
28             } else {
29                 System.out.println("Число_" + m.group(2));
30             }
31         } else {
32             System.out.println("Ошибка");
33         }
34     }
35 }
```

---

ществляться из файла (в UTF-8), при этом лексический анализатор должен вычислять текущие координаты в обрабатываемом тексте. В результате работы программы в стандартный поток вывода должны выдаваться описания распознанных лексем в формате

Тег (координаты): значение

Например,

```
IDENT (1, 2): count  
ASSIGN (1, 8): :=  
NUMBER (1, 11): 100
```

Лексемы во входном файле могут разделяться пробельными символами (пробел, горизонтальная табуляция, маркеры окончания строки), а могут быть записаны слитно (если это не приводит к противоречиям).

Идентификаторы и числовые литералы не могут содержать внутри себя пробельных символов, если в задании явно не указано иного (варианты 4 и 14). Комментарии, строковые и символьные литералы могут содержать внутри себя пробельные символы.

Входной файл может содержать ошибки, при обнаружении которых лексический анализатор должен выдавать сообщение с указанием координаты:

```
syntax error (10,2)
```

После обнаружения ошибки лексический анализатор должен восстанавливаться по следующей схеме: из входного потока пропускаются все подряд идущие символы до нахождения следующей лексемы.

Варианты языков для лексического анализа приведены в таблицах 1, 2 и 3.

Таблица 1: Краткое описание лексики вариантов языков

|    |   |
|----|---|
| 1  | Идентификаторы: последовательности латинских букв, начинающиеся с гласной буквы. Числовые литералы: последовательности десятичных цифр, перед которыми может стоять знак «минус». Операции: «--», «<», «<=».  |
| 2  | Комментарии: начинаются с «/*», заканчиваются на «*/» и могут пересекать границы строк текста. Идентификаторы: последовательности латинских букв и десятичных цифр, в которых буквы и цифры чередуются. Ключевые слова: «for», «if», «m1».  |
| 3  | Строковые литералы: ограничены апострофами, для включения апострофа в литерал он удваивается, не пересекают границы строк текста. Числовые литералы: последовательности десятичных цифр, которые могут включать точку и предваряться знаком «минус». Идентификаторы: последовательности буквенных символов Unicode, точек и цифр, начинающиеся с буквы.   |
| 4  | Идентификаторы: либо последовательности латинских букв, либо непустые последовательности десятичных цифр, ограниченные круглыми скобками. Числовые литералы: либо последовательности десятичных цифр, не начинающиеся с нуля, либо «0». Операции: «()», «:», «:=».  |
| 5  | Комментарии: начинаются с «//» и продолжаются до окончания строки текста. Идентификаторы: любой текст, не содержащий «/» и ограниченный символами «/». Ключевые слова: «/while/», «/do/», «/end/».  |
| 6  | Строковые литералы: ограничены двойными кавычками, могут содержать Escape-последовательности «\», «\n», «\t» и «\\», не пересекают границы строк текста. Числовые литералы: последовательности десятичных цифр, разбитые точками на группы по три цифры («100», «1.000», «1.000.000»). Идентификаторы: последовательности буквенных символов Unicode, знаков подчёркивания и цифр, начинающиеся с буквы или подчёркивания.                  |
| 7  | Идентификаторы: последовательности латинских букв и десятичных цифр, оканчивающиеся на цифру. Числовые литералы: последовательности десятичных цифр, органиченны знаками «<» и «>». Операции: «<=», «=», «==».  |
| 8  | Комментарии: целиком строка текста, начинающаяся с «*». Идентификаторы: либо последовательности латинских букв нечётной длины, либо последовательности символов «*». Ключевые слова: «with», «end», «**».   |
| 9  | Строковые литералы: органичены двойными кавычками, для включения двойной кавычки она удваивается, для продолжения литерала на следующей строчке текста в конце текущей строчки ставится знак «\». Числовые литералы: либо последовательности десятичных цифр, либо последовательности шестнадцатеричных цифр, начинающиеся с «\$». Идентификаторы: последовательности буквенных символов Unicode, цифр и знаков «\$», начинающиеся с буквы. |
| 10 | Идентификаторы: последовательности заглавных латинских букв, за которыми могут располагаться последовательности знаков «+», «-» и «*». Числовые литералы: знак «*» или последовательности, состоящие целиком либо из знаков «+», либо из знаков «-». Ключевые слова: «ON», «OFF», «**».   |

Таблица 2: Краткое описание лексики вариантов языков (продолжение)

|    |  |
|----|--|
| 11 | Комментарии: начинаются с «(*)» или «{», заканчиваются на «*)» или «}» и могут пересекать границы строк текста. Идентификаторы: последовательности латинских букв, представляющие собой конкатенации двух одинаковых слов («zz», «abab»). Ключевые слова: «if if», «do», «dodo».   |
| 12 | Строковые литералы: ограничены обратными кавычками, могут занимать несколько строчек текста, для включения обратной кавычки она удваивается. Числовые литералы: десятичные литералы представляют собой последовательности десятичных цифр, двоичные — последовательности нулей и единиц, оканчивающиеся буквой «b». Идентификаторы: последовательности десятичных цифр и знаков «?», «*» и « », не начинающиеся с цифры.   |
| 13 | Идентификаторы: последовательности буквенных символов Unicode десятичных цифр, начинающиеся и заканчивающиеся на одну и ту же букву. Числовые литералы: последовательности шестнадцатеричных цифр (чтобы литерал не был похож на идентификатор, его можно предварять нулём). Ключевые слова: «qeq», «xx», «xxx».   |
| 14 | Символьные литералы: ограничены апострофами, могут содержать Escape-последовательности «\», «\n», «\» и «\xxxx» (в последней Escape-последовательности буквы «x» обозначают шестнадцатеричные цифры). Идентификаторы: последовательности буквенных символов Unicode длиной от 2 до 10 символов, начинающиеся и заканчивающиеся буквой. Ключевые слова: «z», «for», «forward».  |
| 15 | Идентификаторы: последовательности буквенных символов Unicode цифр, начинающиеся с буквы. Знаки операций: либо последовательности, состоящие из знаков !, #, \$, %, &, *, +, ., /, <, =, >, ?, @, \, ^,  , - и ~, либо идентификаторы, записанные в обратных кавычках (например, «'plus'»). Ключевые слова «where», «->», «=>».  |
| 16 | Комментарии: начинаются с «--» и продолжаются до конца строки, либо ограничены «{-» и «-}», могут занимать несколько строк текста. Целочисленные литералы: последовательности десятичных цифр. Вещественные литералы: последовательности десятичных цифр, за которой следует либо дробная часть (десятичная точка и последовательность десятичных цифр, возможно, пустая), либо показатель степени (буква «e» или «E», за которой следует не менее одной десятичной цифры), либо дробная часть и показатель степени. |
| 17 | Идентификаторы: последовательности буквенных символов Unicode цифр, начинающиеся с буквы. Числовые литералы: десятичные литералы представляют собой последовательности десятичных цифр, шестнадцатеричные — начинаются на десятичную цифру и заканчиваются символом «h». Ключевые слова «mov», «eax».  |
| 18 | Числовые литералы: знак «0» либо последовательности знаков «1». Строковые литералы: регулярные строки — ограничены двойными кавычками, могут содержать escape-последовательности «\», «\t», «\n», не пересекают границы строк текста; буквальное строки — начинаются на «@», заканчиваются на двойную кавычку, пересекают границы строк текста, для включения двойной кавычки она удваивается.   |
| 19 | Идентификаторы: последовательности десятичных цифр. Числовые литералы: римские цифры или ключевое слово «NIL», не чувствительны к регистру.  |

Таблица 3: Краткое описание лексики вариантов языков (продолжение)

|    |   |
|----|---|
| 20 | Идентификаторы: последовательности буквенных символов Unicodeи цифр, начинающиеся с буквы. Целочисленные константы: десятичные — последовательности десятичных цифр, шестнадцатеричные — последовательности шестнадцатеричных цифр, начинающиеся на «&H». Ключевые слова — «PRINT», «GOTO», «GOSUB» без учёта регистра.   |
| 21 | Идентификаторы переменных: последовательности буквенных символов Unicodeи цифр, начинающиеся на знаки «\$», «@», «%». Имена функций: последовательности буквенных символов Unicodeи цифр, начинающиеся на букву. Ключевые слова «sub», «if», «unless».  |
| 22 | Регулярные выражения: ограничены знаками «/», не могут пересекать границы текста, содержат escape-последовательности «\n», «\», «\\». Строковые литералы: ограничены тремя кавычками («""»), могут занимать несколько строчек текста, не могут содержать внутри более трёх кавычек подряд.  |
| 23 | Идентификаторы: последовательности буквенных символов Unicodeи цифр, начинающиеся с буквы. Имена переменных: начинаются с префикса «s», «t» или «e», после которого может располагаться одна буква, одна цифра или точка («.»), за которой следует непустая последовательность буквенных символов Unicodeи цифр. Примеры имён переменных: «s1», «tx», «e.FileName», «t.666».  |
| 24 | Идентификаторы: последовательности буквенных символов Unicodeи цифр, которые начинаются с буквы и могут заканчиваться на один из знаков «%», «\$», «#», «&» или «&». Комментарии — любой текст, следующий за словом «REM» и продолжающийся до конца строки (то есть после буквы «M» в «REM» не может следовать буква или цифра). Операции: «+», «-», «/», «\».  |
| 25 | Идентификаторы: последовательности буквенных символов Unicodeи цифр, начинающиеся с буквы. Строковые константы: либо последовательность символов, ограниченных апострофами, апостроф внутри строки описывается как два апострофа подряд, не пересекают границы строк текста, либо знак «#», за которым следует десятичная константа (код символа). Несколько строковых констант могут быть записаны слитно. Пример: «'hello'#10#13'world'». |
| 26 | Строковые литералы: ограничены двойными кавычками, не могут пересекать границы текста, содержат escape-последовательности «\n», «\"», «\t» и «\\». Числовые литералы: последовательности десятичных знаков и знаков «_», начинающиеся с цифры.  |
| 27 | Слова: последовательности непробельных символов, которые не могут начинаться на «:» либо любой текст, окружённый кавычками (сами содержать кавычки не могут), не могут пересекать границы текста. Метки — последовательности из букв и цифр, начинающиеся со знака «:». Комментарии: начинаются с «::» и продолжаются до окончания строки текста.   |
| 28 | Идентификаторы: последовательности буквенных символов Unicode, цифр и дефисов, начинающиеся с заглавной буквы. Директивы: любой знак валюты, после которого следует непустая последовательность заглавных букв.   |
| 29 | Целочисленные константы: последовательности десятичных цифр, предваряемые символом #. Имена переменных: последовательности десятичных цифр, начинающиеся со знаков «.» или «:», имена массивов: последовательности десятичных цифр, начинающиеся с «,» или «;». Ключевые слова «PLEASE», «DO», «FORGET».  |