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

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

// Компиляция для Windows:
// javac -encoding utf8 IdentVsNumber.java

public class IdentVsNumber
{
    public static void main(String args[])
    {
        test_match("Альфа123");
        test_match("42");
    }

    public static void test_match(String text)
    {
        // Регулярные выражения
        String ident = "\\p{L}[\\p{L}0-9]*";
        String number = "[0-9]+";
        String pattern = "(?<ident>^"+ident+")|(?<number>^"+number+")";
        // Компиляция регулярного выражения
        Pattern p = Pattern.compile(pattern);
        // Сопоставление текста с регулярным выражением
        Matcher m = p.matcher(text);
        if (m.find()) {
            if (m.group("ident") != null) {
                System.out.println("Идентификатор " + m.group("ident"));
            } else {
                System.out.println("Число " + m.group("number"));
            }
        } else {
            System.out.println("Ошибка");
        }
    }
}
```

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 5, 34 и 36). 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.

Лексический анализатор должен иметь программный интерфейс для взаимодействия с парсером. Рекомендуется реализовывать его как итератор с методом `nextToken()` для императивных языков или функцию, возвращающую список лексем, для функциональных языков.

В регулярных выражениях рекомендуется использовать классы символов Unicode для обозначения букв, чисел и других подобных множеств. Многие движки регулярных выражений для задания классов используют синтаксис `\p{класс}`. Вместо нумерованных групп рекомендуется использовать именованные `(?<имя>regexp)`, при использовании нумерованных групп ненумеруемые обозначаются как `(?:regexp)`.

Варианты языков для лексического анализа приведены в таблицах 1, 2, 3, 4 и 5.

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 латинских букв, знаков подчёркивания и цифр, начинающиеся с буквы или подчёркивания.
7	Идентификаторы: последовательности латинских букв и десятичных цифр, оканчивающиеся на цифру. Числовые литералы: непустые последовательности десятичных цифр, органиченные знаками «`<`» и «`>`». Операции: «`<=`», «`=`», «`==`».
8	Комментарии: целиком строка текста, начинающаяся с «`*`». Идентификаторы: либо последовательности латинских букв нечётной длины, либо последовательности символов «`*`». Ключевые слова: «`with`», «`end`», «`**`».

Таблица 2: Краткое описание лексики вариантов языков (продолжение)

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», «**».
11	Комментарии: начинаются с «(*» или «{», заканчиваются на «*)» или «}» и могут пересекать границы строк текста. Целочисленные литералы: последовательности десятичных цифр. Дробные литералы: строки вида «digits/digits», где «digits» — последовательность десятичных цифр.
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 и десятичных цифр, начинающиеся и заканчивающиеся на 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» (здесь буквы «x» обозначают шестнадцатеричные цифры). Идентификаторы: последовательности буквенно-цифровых символов Unicode длиной от 2 до 10 символов, начинающиеся и заканчивающиеся буквой. Ключевые слова: «z», «for», «forward».
15	Идентификаторы: последовательности латинских букв и 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.

Таблица 3: Краткое описание лексики вариантов языков (продолжение)

17	Идентификаторы: последовательности буквенных символов Unicode и цифр, начинающиеся с буквы.Числовые литералы: десятичные литералы представляют собой последовательности десятичных цифр, шестнадцатиричные — начинаются на десятичную цифру, содержат шестнадцатеричные цифры (в любом регистре) 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; буквальные 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 «`NIHIL`», не чувствительны к регистру.
20	Идентификаторы: последовательности буквенных символов Unicode и цифр, начинающиеся с буквы, не чувствительны к регистру. Целочисленные константы: десятичные — последовательности десятичных цифр, шестнадцатеричные — последовательности шестнадцатиричных цифр, начинающиеся на «`&H`». Ключевые слова — «`PRINT`», «`GOTO`», «`GOSUB`» без учёта регистра.
21	Идентификаторы переменных: последовательности буквенных символов Unicode и цифр, начинающиеся на знаки «`$`», «`@`», «`%`». Имена функций: последовательности буквенных символов Unicode и цифр, начинающиеся на букву. Ключевые слова «`sub`», «`if`», «`unless`».
22	Регулярные выражения: ограничены знаками «`/`», не могут пересекать границы текста, содержат escape-последовательности «`\n`», «`\/`», «`\\`». Строковые литералы: ограничены тремя кавычками («`"""`»), могут занимать несколько строчек текста, не могут содержать внутри более двух кавычек подряд.
23	Идентификаторы: последовательности латинских букв и цифр, начинающиеся с буквы. Имена переменных: начинаются с префикса «`s`», «`t`» или «`e`», после которого может располагаться одна буква, одна цифра или точка («`.`»), за которой следует непустая последовательность латинских букв и цифр. Примеры имён переменных: «`s1`», «`tx`», «`e.FileName`», «`t.666`». (Домен имён переменных имеет приоритет.)
24	Идентификаторы: последовательности буквенных символов Unicode и цифр, которые начинаются с буквы и могут заканчиваться на один из знаков «`%`», «`$`», «`#`», «`&`» или «`!`». Ключевые слова `FOR`, `NEXT`. Комментарии — любой текст, следующий за ключевым словом «`REM`» и продолжающийся до конца строки (то есть после буквы «`M`» в «`REM`» не может следовать буква или цифра). Операции: «`+`», «`-`», «`/`», «`\`». Идентификаторы и ключевые не чувствительны к регистру.

Таблица 4: Краткое описание лексики вариантов языков (продолжение)

25	Идентификаторы: последовательности латинских букв и цифр, начинающиеся с буквы. Строковые константы — последовательности строковых секций, записанных слитно. Строковые секции: 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). Пример строковой константы: «'hello'#10#13'world'» (эта строковая константа состоит из 4 строковых секций, однако является единым токеном).
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 (прочерк не влияет на значение числа).
27	Идентификаторы: последовательности буквенных символов Unicode, цифр и дефисов, начинающиеся с заглавной буквы. Директивы: любой знак валюты, после которого следует непустая последовательность заглавных букв.
28	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».
29	Химические вещества: последовательности латинских букв и цифр, начинающиеся с заглавной буквы, при этом после цифры не может следовать строчная буква. Примеры: «CuSO4», «CH3CH2OH», «Fe2O3». Коэффициенты: последовательности десятичных цифр. Между коэффициентом и веществом пробел может отсутствовать. Операторы: «+», «->».
30	Числа фибоначчиевой системы счисления: последовательности знаков «0» и «1», причём две единицы не могут соседствовать друг с другом. Идентификаторы: последовательности латинских букв, в которых гласные и согласные чередуются.
31	Слова искусственного языка Токипона конструируются из слогов. Если слог находится в начале слова, то он может не содержать первую согласную, в остальных случаях слог — это согласная + гласная + опциональная n. n не ставится, если за ней идут n или m. Из всех вариантов слогов в токипоне запрещены слоги: ji, ti, wo, wu (как труднопроизносимые). Гласные: a, e, i, o, u. Согласные: j, k, l, m, n, p, s, t, w. Необходимо написать лексический анализатор, определяющий слова языка Токипона. Знаки препинания: «.», «,», «?», «!».

Таблица 5: Краткое описание лексики вариантов языков (продолжение)

32	Географические координаты: начинаются с одного из знаков «S», «E», «N», «W», после которых располагается целое десятичное число, за которым может следовать либо точка и последовательность десятичных цифр, либо знак «D», за которым следует необязательная запись угловых минут (число от 0 до 59, за которым пишется апостроф) и угловых секунд (число от 0 до 59, за которым следует двойная кавычка). Атрибут лексемы (для лабораторных работ № 4 и № 6): вещественное число, соответствующее широте или долготе. Широта («S», «N») не может превышать 90, долгота («E», «W») — 180.
33	Числа: последовательности десятичных цифр. Знаки операций: «+», «-», «*», «/», «(», «)». Комментарии начинаются на «(*», заканчиваются на «*)», не могут быть вложенными.
34	Целочисленные переменные: начинаются с буквы «I», «J», «K», «L», «M» или «N», за которой может следовать не более пяти латинских букв и цифр. Вещественные переменные начинаются с любой другой буквы, за которой может следовать не более пяти латинских букв и цифр. Знаки: «+», «,». Целые числа — последовательности десятичных цифр. Пробелы и табуляции игнорируются (т.е. могут встречаться внутри переменных и чисел, не меняя их смысл).
35	Целые числа одинарной длины: последовательности десятичных цифр. Целые числа двойной длины: последовательности десятичных цифр с точкой на конце. Слова: любые последовательности непробельных символов, не являющиеся числами.
36	Открывающий тег: последовательность букв и цифр, окружённая «<» и «>». Закрывающий тег: последовательность букв и цифр, окружённая «</» и «>». Пробел (значимый токен): любая последовательность пробельных символов, Ключевое слово: «<», «>», «&», символ: любой печатный символ, кроме «<», «>» и «&».
37	Целые числа: последовательности цифр определенной системы счисления, предваренные соответствующим индикатором, определяющим систему счисления (для десятичных чисел — пустой индикатор, для двоичных чисел — «0b», для восьмеричных чисел — «0t», для шестнадцатеричных чисел — «0x»). Ключевые слова: «and», «or». Знаки операций: «(», «)».