Лекция 4.

Лексический анализ

Конечные автоматы

Лектор

Елена Болдырева eaboldyreva@itmo.ru

РАЗДЕЛ 1. ЛЕКСИЧЕСКИЙ АНАЛИЗ

Лексический анализ

Основная задача лексического анализа

разбить входной текст, состоящий из последовательности отдельных **литер** (символов), на последовательность **лексем** (слов)

Любой символ входной последовательности может

– принадлежать к какой-либо лексеме
– принадлежать к разделителю (разделять лексемы)

Лексический анализ

В зависимости от ситуации одна и та же последовательность символов может быть и частью лексемы, и частью разделителя.

```
if (a>b) /* if */
```

лексема разделитель

В некоторых случаях разделители между лексемами могут отсутствовать.

```
if ( a > b ) a -= b ; else b -= a ;
if(a>b)a-=b;else b-=a;
```

Лексический анализ

Кроме определения границ лексем, при работе лексического анализатора требуется определить их тип:

- идентификаторы, в т.ч.
  - ключевые слова;
  - пользовательские идентификаторы;
- пунктуаторы;
- числа;
- строки;

и т.д.

Лексический анализ

для синтаксического анализа

– ключевые слова – значение лексемы;
– пользовательские идентификаторы – тип лексемы;
– пунктуаторы – значение лексемы;
– числа – тип лексемы;
– строки – тип лексемы;

и т.д.

для семантического анализа

– пользовательские идентификаторы – значение лексемы;
– числа – значение лексемы;
– строки – значение лексемы;

и т.д.

для всех последующих фаз

– расположение лексемы в исходном тексте программы (имя файла, номер строки, позиция в строке)

(для локализации синтаксических ошибок и ошибок времени выполнения)

Регулярные множества

На этапе лексического анализа удобно считать, что лексемы каждого типа являются элементами отдельных *языков*, называемым ***регулярными множествами***.

Формальное определение регулярного множества

Регулярное множество в алфавите V определяется *рекурсивно* следующим образом:

(1) $\varnothing$ (пустое множество) – регулярное множество в алфавите V;

(2) $\{\varepsilon\}$ – регулярное множество в алфавите V (ε – пустая цепочка);

(3) $\{a\}$ – регулярное множество в алфавите V для каждого $a \in V$;

(4) если P и Q – регулярные множества в алфавите V, то регулярными являются и множества

- (а) $P \cup Q$ (объединение),
- (б) PQ (конкатенация, т.е. множество $\{pq \mid p \in P, q \in Q\}$),
- (в) P^* (итерация: $P^* = $);

(5) ничто другое не является регулярным множеством в алфавите V.

https://habr.com/ru/post/457312/

Неформальное определение регулярного множества

Множество в алфавите V регулярно тогда и только тогда, когда оно

– $\varnothing$,
– $\{\varepsilon\}$,
– $\{a\}$, где $a \in V$,
– его можно получить из этих множеств применением конечного числа операций объединения, конкатенации и итерации.

Средством записи регулярных множеств являются ***регулярные выражения.***

Формальное определение регулярного выражения

Регулярное выражение в алфавите V определяется *рекурсивно* следующим образом:

(1) $\varnothing$ – регулярное выражение, обозначающее множество $\varnothing$;

(2) ε – регулярное выражение, обозначающее множество $\{\varepsilon\}$;

(3) a – регулярное выражение, обозначающее множество $\{a\}$;

(4) если p и q – регулярные выражения, обозначающие регулярные множества P и Q соответственно, то

(а) $(p|q)$ – регулярное выражение, обозначающее регулярное множество $P \cup Q$,

(б) (pq) – регулярное выражение, обозначающее регулярное множество PQ,

(в) (p^*) – регулярное выражение, обозначающее регулярное множество P^*;

(5) ничто другое не является регулярным выражением в алфавите V.

Регулярные выражения

Для краткости записи регулярных выражений используются следующие соглашения:

– лишние скобки в регулярных выражениях опускаются с учетом приоритета операций:

1. операция итерации (наивысший приоритет)
2. операция конкатенации
3. операция объединения (наименьший приоритет).

– запись $p+$ обозначает выражение pp^*

Например: $(a \mid ((ba)(a^*))) \rightarrow a \mid ba+$

Регулярные выражения

Примеры регулярных выражений:

***a* (*a* | *b*) ***

обозначает множество всевозможных цепочек, состоящих из *a* и *b*, начинающихся с *a*;

(*a* | *b*)* (*a* | *b*) (*a* | *b*)*

обозначает множество всех непустых цепочек, состоящих из *a* и *b*, т.е. множество {*a, b*}+;

((0 / 1) (0 / 1) (0 / 1))*

обозначает множество всех цепочек, состоящих из нулей и единиц, длины которых делятся на 3.

Для каждого регулярного множества можно найти регулярное выражение, обозначающее это множество, и наоборот.

Регулярные выражения

При записи регулярных выражений оказывается удобно дать им индивидуальное обозначение.

Пример 1. Регулярное выражение для множества идентификаторов.

Letter = *a* | *b* | *c* | ... | *x* | *y* | *z*
Digit = 0 | 1 | ... | 9
Identifier = *Letter* (*Letter* | *Digit*)*

Пример 2. Регулярное выражение для множества вещественный чисел с плавающей запятой.

Digit = 0 | 1 | ... | 9
Integer = *Digit* +
Fraction = .*Integer* | ε
Exponent = (*E*(+ | – | ε)*Integer*) | ε
Number = *Integer Fraction Exponent*

Конечный автомат

Для определения принадлежности последовательности символов регулярному множеству (т.е. для реализации процедуры *распознавания языка*) чаще всего используются ***конечные автоматы***.

Формальное определение конечного автомата

Конечным автоматом (КА) называют пятерку
(**Q**, **V**, f, q_0, **F**), где

Q – конечное множество состояний автомата;

V – конечное множество допустимых входных символов (алфавит автомата);

f – функция переходов, отображающая декартово произведение множеств $\mathbf{V}\times\mathbf{Q}$ во множество подмножеств **Q**:

$$f(a, q)=\mathbf{R},\ a\in\mathbf{V},\ q\in\mathbf{Q},\ \mathbf{R}\subseteq\mathbf{Q};$$

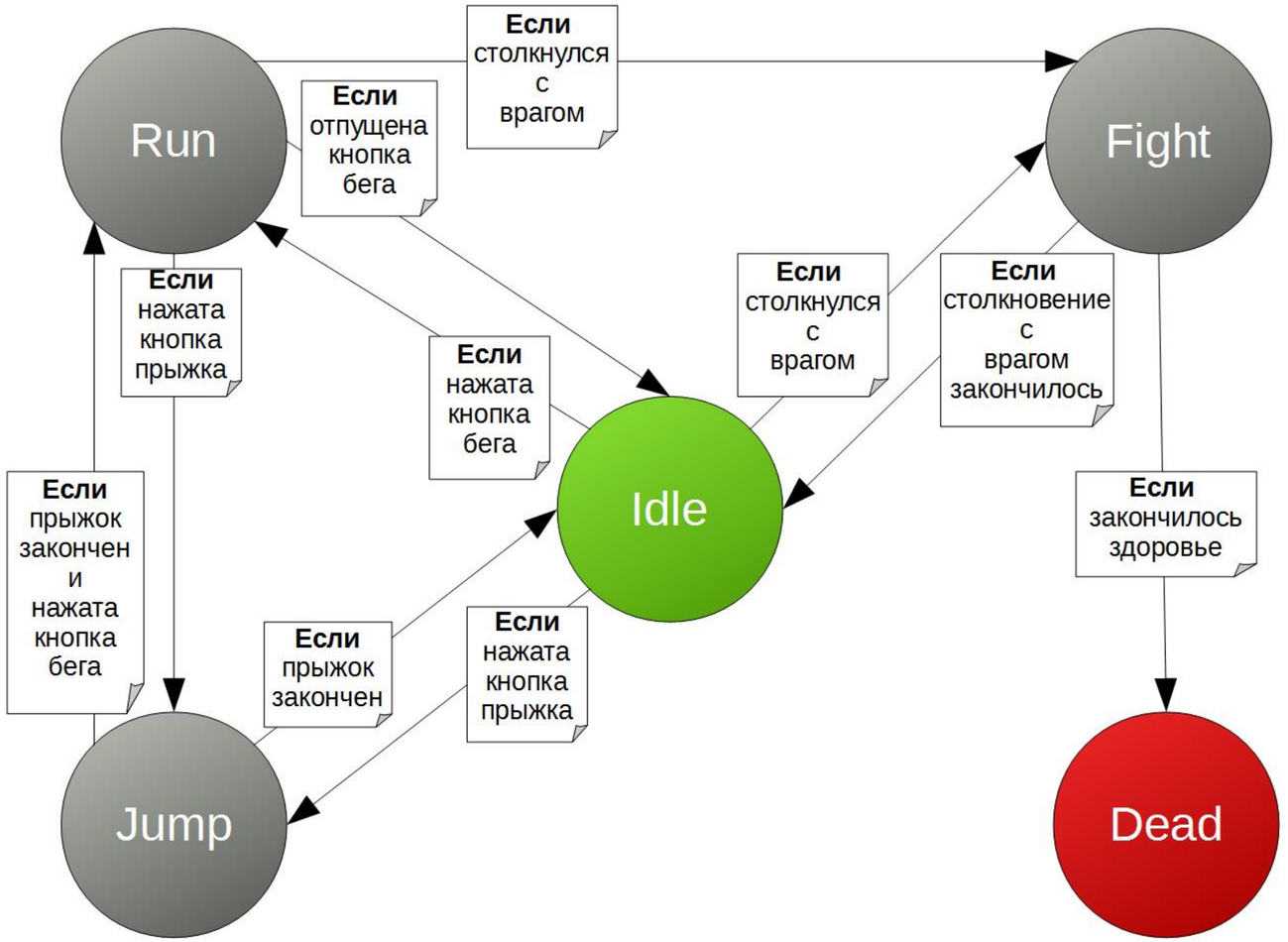
q_0 – начальное состояние автомата, $q_0\in\mathbf{Q}$;

F – непустое множество заключительных состояний автомата, $\mathbf{F}\subseteq\mathbf{Q}$, $\mathbf{F}\neq\varnothing$.

Конечный автомат

Работа конечного автомата продолжается до тех пор, пока на его вход поступают символы.

Если после окончания работы конечного автомата он находится в одном из *заключительных состояний*, то говорят, что конечный автомат *принял цепочку (допускает цепочку).*



Множество цепочек, принимаемых (допускаемых) конечным автоматом, называют ***языком, распознаваемым (допускаемым) конечным автоматом.***

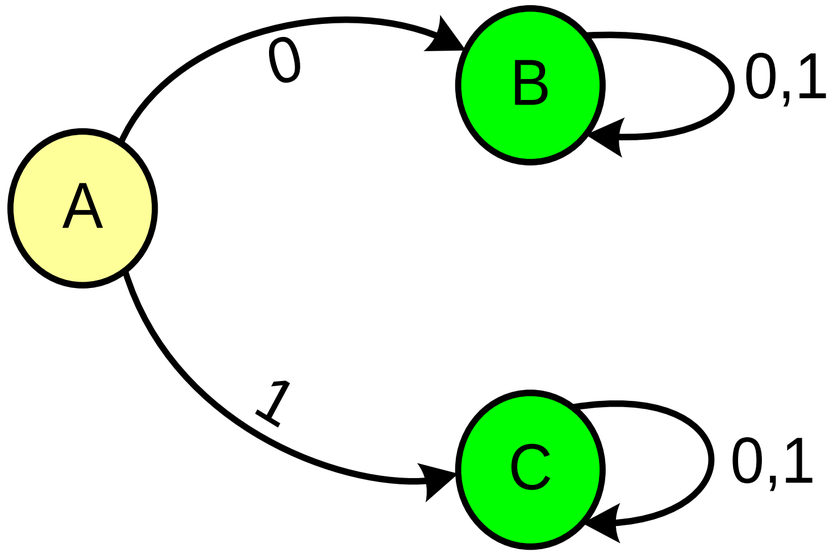
Пример 1. Диаграмма всюду определенного детерминированного конечного автомата

Q = {A,B,C}

V = {0,1}

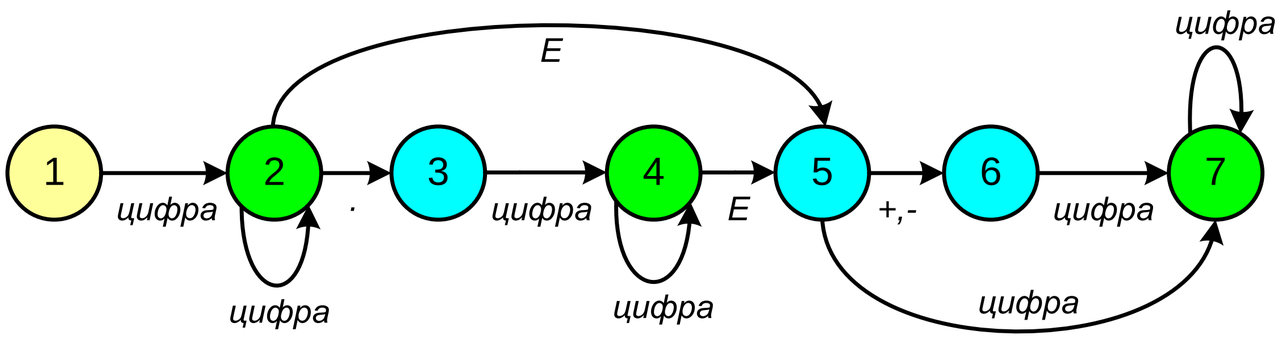
q_0 = A

f	0	1
A	B	C
B	B	B
C	C	C



Конечный автомат

Пример 2. Диаграмма конечного автомата, принимающего множество положительных действительных чисел



Различными цветами показаны различные состояния конечного автомата: начальное, промежуточные, заключительные

Состояниям этого конечного автомата соответствуют:

1 – Начало числа

2 – Целая часть

3 – Начало дробной части

4 – Дробная часть

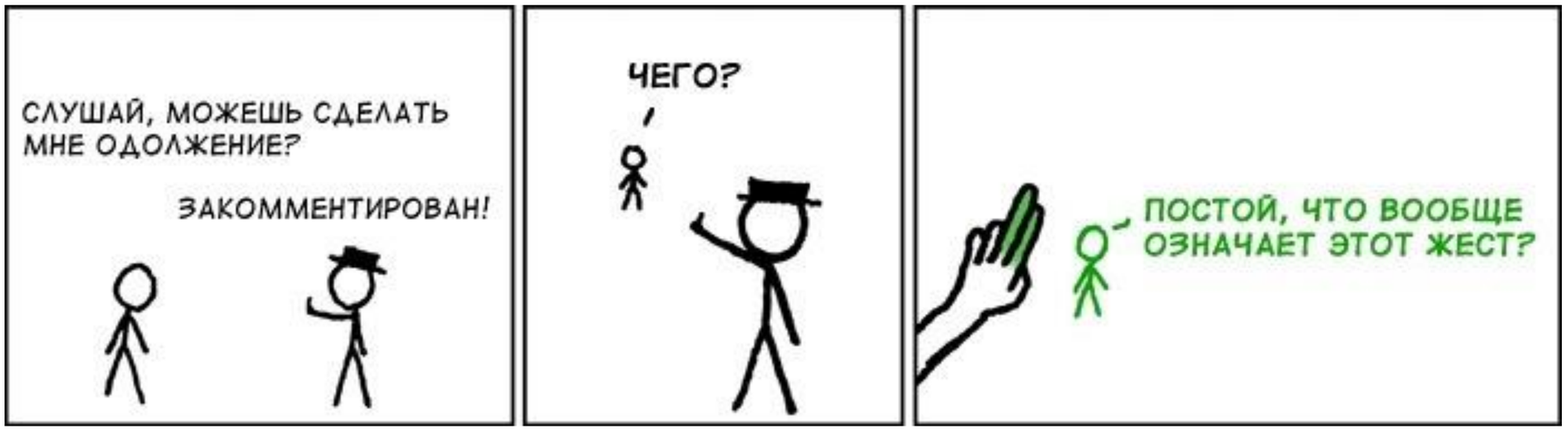
5 – Начало экспоненциальной части

6 – Начало модуля показателя

7 – Показатель

f	.	+, –	E	0..9
1				2
2	3		5	2
3				4
4			5	4
5		6		7
6				7
7				7

РАЗДЕЛ 2. ВВЕДЕНИЕ В ЯЗЫКИ РАЗМЕТКИ



Искусство парсинга или DOM своими руками
https://habr.com/ru/post/442964/, https://habr.com/ru/post/444876/

Пишем изящный парсер на Питоне: https://habr.com/ru/post/309242/

Подробно про веб парсинг в Python с примерами: https://pythonpip.ru/examples/parsing-python

Парсим на Python: Pyparsing для новичков: https://habr.com/ru/post/239081/

Пример пошагового исполнения в Jupiter
https://nbviewer.org/urls/gist.githubusercontent.com/tbicr/cd584138ce183839946f/raw/e0c335bd57103e200279302eff3c667d5dd470b1/Pyparsion.ipynb

Пишем парсер на Python за 30 минут: https://www.youtube.com/watch?v=pqsNYDuRg-o

Пишем простой парсер JSON на Python: https://proghub.ru/p/writing-a-simple-json-parser

Использование Python для парсинга файлов конфигурации
https://pythonist.ru/python-dlya-parsinga-fajlov-konfiguraczii/