Лекция №4 Основы лексического анализа.

Основная задача лексического анализа - разбить входной текст состоит из последовательности одиночных символов, последовательность слов, или лексем, то есть выделить эти слова из непрерывной последовательности символов. все символы входной последовательности с этой точки зрения делятся на символы, принадлежат либо лексемам, и символы, разделяющие лексемы (разделители). В некоторых случаях между лексемами может и не быть разделителей. С другой стороны, в некоторых языках лексемы могут содержать незначительные символы (Например, символ пробела в Фортране). В Си разделительную значение символив-разделителей может блокироваться ("\" в конце строки внутри "...").

Обычно все лексемы делятся на классы. Примерами таких классов является числа (цели, восьмеричные, шестнадцатеричные, действительны и т.д.), идентификаторы, строки. Отдельно выделяются ключевые слова и символы пунктуации (иногда их называют символы-ограничители). Как правило, ключевые слова - это некоторое конечное подмножество идентификаторов. В некоторых языках (например, ПЛ / 1) смысл лексемы может зависеть от ее контекста и невозможно провести лексический анализ в отрыве от синтаксического.

С точки зрения дальнейших фаз анализа лексический анализатор выдает информацию двух сортов: для синтаксического анализатора, работающего вслед по лексическим, существенная информация о последовательности классов лексем, ограничителей и ключевых слов, а для контекстного анализа, работающего вслед за синтаксическим, важная информация о конкретных значениях отдельных лексем (идентификаторов, чисел и т.д.).

Таким образом, общая схема работы лексического анализатора такова. Сначала выделяется отдельная лексема (возможно, используя символы-разделители). Ключевые слова распознаются или явным выделением непосредственно в тексте, или сначала выделяется идентификатор, а затем делается проверка на принадлежность его множества ключевых слов.

Если выделенная лексема является ограничителем, то он (точнее, определенный его признак) выдается как результат лексического анализа. Если выделенная лексема является ключевым словом, то выдается признак соответствующего ключевого слова. если выделенная лексема является идентификатором - выдается признак идентификатора, а сам идентификатор хранится отдельно. Наконец, если выделенная лексема принадлежит какому-либо из других классов лексем (например, лексема представляет собой число, строка и т.д.), то выдается признак соответствующего класса, а значение лексемы хранится отдельно.

Лексический анализатор может быть как самостоятельной фазой трансляции, так и подпрограммой, работающий по принципу «дай лексему». В первом случае выходом анализатора является файл лексем, во втором (рис. 3.1, б) лексема выдается при каждом обращении к анализатора (при этом, как правило, признак класса лексемы возвращается как результат функции "лексический анализатор », а значение лексемы передается через глобальную переменную). С точки зрения обработки значений лексем, анализатор может либо просто выдавать значение каждой лексемы, и в этом случае построение таблиц объектов (Идентификаторов, строк, чисел и т.д.) переносится на более поздние фазы, или он может самостоятельно строить таблицы объектов. В этом случае в качестве значение лексемы выдается указатель на вход в соответствующую таблицу.

Работа лексического анализатора задается некоторым конечным автоматом. Однако, непосредственное описание конечного автомата неудобно с практической точки зрения. Поэтому для задача лексического анализатора, как правило, используется или регулярное выражение, или праволинейная грамматика. все три формализма (конечных автоматов, регулярных выражений и праволинейних грамматик) имеют одинаковую выразительную мощность. В частности, по регулярным выражением или праволинейной грамматике можно сконструировать конечный автомат, распознающий тот же язык.

Регулярные множества и выражения

Введем понятие регулярного множества, играет важную роль в теории формальных языков.

Регулярное множество в алфавите T определяется рекурсивно следующим образом:

∅ (Пустое множество) - регулярное множество в алфавите T;

{E} - регулярное множество в алфавите T (e - пустая строка)

{A} - регулярное множество в алфавите T для каждого a T; ∈

если P и Q - регулярные множества в алфавите T, то регулярными есть и множества

PQ (объединения),

PQ (конкатенация, то есть множество { pq | p P, q Q}); ∈

P * (итерация: P * = $\bigcup_{n=0} P^n$)

ничто другое не является регулярным множеством в алфавите T.

Итак, множество в алфавите T регулярно тогда и только тогда, когда оно или, ∅ или {e} или {a} для некоторого a T, или его можно получить из этих множеств применением конечного числа операций объединения, конкатенации и итерации.

Приведенное выше определение регулярного множества позволяет ввести следующую удобную форму его записи, называемую регулярным выражением.

Регулярное выражение в алфавите T и обозначается им регулярное множество в алфавите T определяются рекурсивно следующим образом:

∅ - Регулярное выражение, обозначающее множество;

e - регулярное выражение, обозначающее множество {e};

a - регулярное выражение, обозначающее множество {a};

если p и q - регулярные выражения, обозначающие регулярные множества P и Q соответственно, то

(P | q) - регулярное выражение, обозначающее регулярное множество PQ,

(Pq) - регулярное выражение, обозначающее регулярное множество PQ,

(P *) - регулярное выражение, обозначающее регулярное множество P *;

ничто другое не является регулярным выражением в алфавите T.

Мы будем опускать лишние скобки в регулярных выражениях, договорившись о том, что операция итерации имеет наивысший приоритет, затем идет операции конкатенации, наконец, операция объединения имеет наименьший приоритет.

Кроме того, мы будем пользоваться записью p + для обозначения pp *. Таким образом, запись (a | ((ba) (a *))) эквивалентна a | ba +.

Наконец, мы будем использовать запись L (r) для регулярного множества, обозначаемого регулярным выражением r.

Пример 3.1. Несколько примеров регулярных выражений и обозначаемые ими регулярных множеств:

a (e | a) | b - обозначает множество {a, b, aa};

a (a | b) * - обозначает множество всевозможных цепочек, состоящих из a и b, начинающиеся с a;

(A | b) * (a | b) (a | b) * - обозначает множество всех непустых цепочек, состоящие из a и b, то есть множество {a, b} +;

((0 | 1) (0 | 1) (0 | 1)) * - обозначает множество всех цепочек, состоящих из нулей и единиц, длины которых делятся на 3.

Ясно, что для каждого регулярного множества можно найти регулярный выражение, обозначающее это множество, и наоборот. Более того, для каждого регулярного множества существует бесконечно много обозначают регулярных выражений.

Будем говорить, что регулярные выражения равны или эквивалентные (=), Если они обозначают одно и то же регулярное множество.

Существует ряд алгебраических законов, позволяющих осуществлять эквивалентную преобразования регулярных выражений.

Лемма. Пусть p, q и r - регулярные выражения. Тогда справедливы следующие соотношение:

(1) p | q = q | p; (7) pe = ep = p;

(2) ∅* = e; (8) p∅ = ∅p = ∅;

(3) p | (q | r) = (p | q) | r; (9) p * = p | p *;

(4) p (qr) = (pq) r; (10) (p *) * = p *;

(5) p (q | r) = pq | pr; (11) p | p = p;

(6) (p | q) r = pr | qr; (12) p | ∅ = P.

Следствие. Для любого регулярного выражения существует эквивалентное регулярное выражение ∅, которое либо есть, либо не содержит в своей записи ∅.

В дальнейшем будем рассматривать только регулярные выражения, не содержащие в своей записи ∅.

При практическом описании лексических структур бывает полезно сопоставлять регулярными выражениями некоторые имена, и ссылаться на них по этим именам. для определения таких имен мы будем использовать запись вида

$d_1 = r_1$

$d_2 = r_2$

…

$d_n = r_n$

где d_i - разные имена, а каждое ri - регулярное выражение над символами T $\{d_1, d_2, ..., d_{i-1}\}$ то есть символами основного алфавита и ранее определенными символами (именами). Таким образом, для любого r_i можно построить

регулярное выражение над T, повторно заменяя имена регулярных выражений обозначаемые ими регулярные выражения.

Пример 3.2. Использование имен для регулярных выражений.

Регулярное выражение для множества идентификаторов.

Letter = a | b | c | ... | x | y | z

Digit = 0 | 1 | ... | 9

Identifier = Letter (Letter | Digit) *

Регулярное выражение для множества чисел в десятичной записи.

Digit = 0 | 1 | ... | 9

Integer = Digit +

Fraction =. Integer | e

Exponent = (E (+ | - | e) Integer) | e

Number = Integer Fraction Exponent