Analisis Leksikal / Scanning

- Scanning atau analisis leksikal merupakan tahap dari kompilator yang bertugas membaca sumber program dari suatu file berdasarkan karakter dan membaginya kedalam token / logika unit

Tugas analisa leksikal antara lain :

a. Melakukan pembacaan kode sumber dengan merunut karakter demi karakter.

b. Mengenali besaran leksik (identifier, keywords, dan konstanta).

c. Mentransformasi menjadi sebuah token dan menentukan jenis tokennya.

d. Mengirimkan token.

e. Membuang atau mengabaikan white-space dan komentar dalam program.

f. Menangani kesalahan.

g. Menangani tabel symbol.

Besaran Leksikal terbagi menjadi :

- Identifier dapat berupa keyword atau nama kunci, seperti IF..ELSE, BEGIN..END (pada Pascal), INTEGER (pascal), INT, FLOAT (Bhs C)
- Konstanta : Besaran yang berupa bilangan bulat (integer), bilangan pecahan (float/Real), boolean (true/false), karakter, string dan sebagainya
- Operator, Merupakan operator yang digunakan pada program. Operator arithmatika (+ - * /), operator logika (< = >)
- Delimiter; Berguna sebagai pemisah/pembatas, seperti kurung-buka, kurung - tutup, titik, koma, titik-dua, titik-koma, white-space White Space: pemisah yang diabaikan oleh program, seperti enter, spasi, ganti baris, akhir file

Lexical Analysis - Contoh

Contoh 1:

ada urutan karakter yang disebut dengan statement

fahrenheit := 32 + celcius * 1.8,

Maka akan diterjemahkan kedalam token-token seperti dibawah ini

identifier	fahrenheit
operator	:=
integer	32
operator penjumlahan	+
Identifier	celcius
operator perkalian	*
real / float	1 . 8

Lexical Analysis - Contoh 2

- Setiap bentuk dari token di representasi sebagai angka dalam bentuk internal, dan angkanya adalah unik
- ***Misalnya*** nilai 1 untuk variabel, 2 untuk konstanta, 3 untuk label dan 4 untuk operator, dst
- Contoh instruksi :
  - ***Kondisi : IF A > B THEN C = D***;
- Maka scanner akan mentransformasikan kedalam token-token, sbb:

Lexical Analysis - Contoh 2

Kondisi	3
:	26
IF	20
A	f
>	f5
B	f
THEN	2f
C	f
D	f
;	27

Token-token ini sebagai inputan untuk *syntax Analyser* , token-token ini bisa berbentuk pasangan item. Dimana Item pertama menunjukkan alamat atau lokasi dari token pada tabel simbol. Item kedua adalah representasi internal dari token. Semua token direpresentasikan dengan informasi yang panjangnya tetap (konstan), suatu alamat (address atau pointer) dan sebuah integer (bilangan bulat)

Lexical Analysis – Contoh 3

IF A=10 then

Begin

Writeln ‘nilai A=10’;

Else

Writeln ‘nilai bukan 10’;

Maka besaran leksikal adalah :

Identifier	Konstanta	Operator	Delimite
IF, Then, Begin, Writeln, Else, A	10, ‘ nilai A =10’, ‘nilai A bukan 10’	=	Spasi, ; , enter

EKSPRESI REGULER (ER)

Sebuah bahasa dikatakan regular jika terdapat finite state otomata yang dapat menerimanya.

Bahasa – bahasa yang diterima oleh suatu finite state otomata bisa dinyatakan secara sederhana dengan ekspresi regular *(regular expression)* yang disingkat ER.

Ekspresi Reguler Digunakan untuk merepresentasikan “pattern/ bentuk string untuk setiap karakter dan juga menjelaskan suati set string.

Notasi Ekspresi Regular

- * yaitu karakter asterisk, berarti bisa tidak muncul, bisa juga muncul berhingga kali (0-n).
- $^{+}$ (posisi *superscript*) berarti minimal muncul satu kali (1-n).
- + atau u berarti union.
- .(titik) yang berarti konkatenasi. Biasanya dihilangkan (ab sama artinya dengan a.b)

Contoh Ekspresi Regular (ER)

Ekspresi Regular	Deskripsi
ab*cc	b bisa tidak muncul atau muncul sejumlah berhingga kali. Cth : acc, abcc, abbcc, abbbcc.
$a^{+}d$	a muncul minimal sekali. Cth : ad, aad, aaad.
a* ∪ b*	a bisa tidak muncul/muncul sejumlah kali atau b bisa tidak muncul/muncul sejumlah kali. Cth : a, b, aa, bb, aaa, bbb.
(a ∪ b) / (a + b)	a muncul 1 kali atau b muncul 1 kali. Cth : a,b.

Ekspresi Regular	Deskripsi
(a ∪ b)* / (a + b)*	semua a dan b. Cth : a, b, ab, ba, abb, aab, bba, aaaa, bbbb.
aa	a muncul sepasang sebanyak 1 kali
(aa)*	a muncul sepasang sebanyak 0-n kali (genap)
ab* + a	string yang berawalan dengan a, dan selanjutnya boleh diikuti dengan b

Ekspresi regular untuk programming

1. Pada program pascal, angka / digit terdiri dari decimal Contoh : 4578, 0933
   Dapat dibuat ER :

   Nama ER : nat Anggota ER : 0..9

   Sehingga ER : nat = [0..9]

2. **Identifier** terdiri dari semua karakter dari a..z,A..Z dan 0..9

   Dengan kemungkinan kemunculan abc, aabcad, a01, Z09 dan lain-lain.

   Nama ER : Letter anggota :{a..z,A..Z}

   Nama ER : Digit anggota : {0..9}

   Maka : Letter = [a..z,A..Z] Digit = [0..9]

   Sehingga : **Identifier = Letter (Letter|digit)***

Finite State Automata (FSA)

Digunakan untuk menjelaskan proses terbentuknya ‘pattern’/ bentuk dari inputan string dan juga dapat digunakan untuk konstruksi scanner.

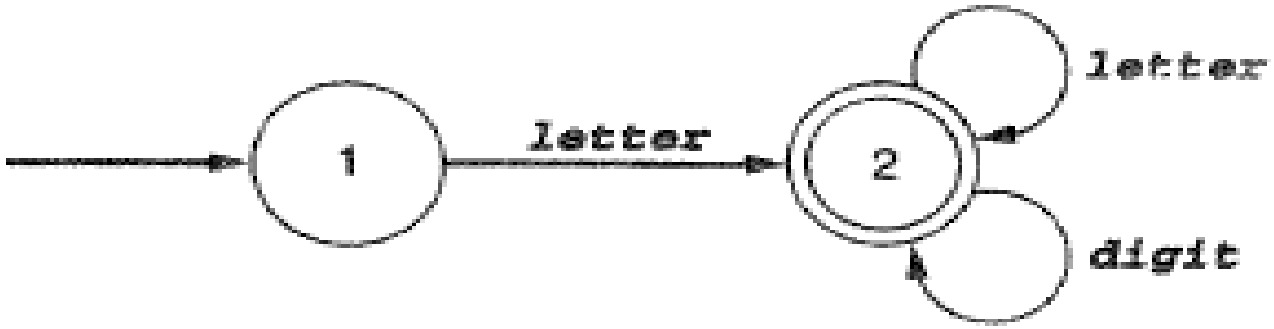
- model matematika yang dapat menerima input dan mengeluarkan output
- Analisis leksikal lebih mudah diimplementasikan pada FSA.
- Memiliki state yang berhingga banyaknya dan dapat berpindah dari satu state ke state lainnya berdasar input dan fungsi transisi
- Tidak memiliki tempat penyimpanan/memory, hanya bisa mengingat state terkini
- Mekanisme kerja dapat diaplikasikan pada : elevator, text editor, analisa leksikal, pencek parity.

Contoh FSA untuk Identifier

1.FSA untuk identifier : terdiri dari huruf (kecil maupun kapital), bisa diikuti huruf atau digit, tanpa batas panjang dapat dinyatakan : (huruf)(huruf+digit)*

Huruf berupa A..Z atau a..z

Digit berupa 0..9



FSA untuk **floating point number.**

- Kemungkinan string yang akan muncul : + 0.123E+47

  -147. 0E+31

  + 411. 8E-2