Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

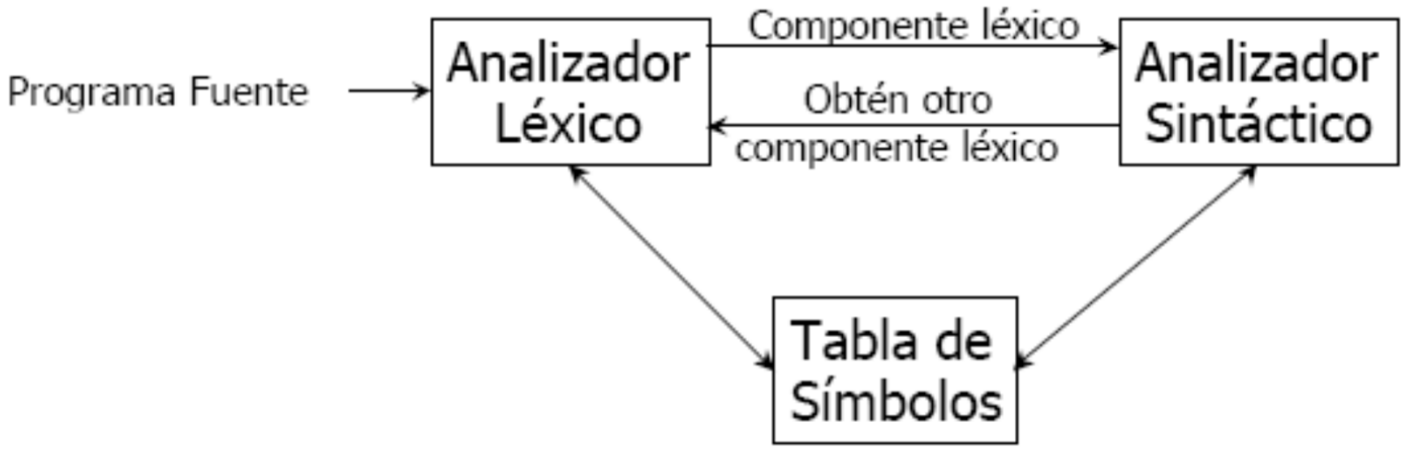
UNIDAD III:
ANALISIS LÉXICO

ING. SANDRA RODRIGUEZ AVILA

CONTENIDO

- ANALISIS LÉXICO (AL) o SCANNER
- FUNCIONES DEL ANALIZADOR LÉXICO O AL
- ASPECTOS DEL ANÁLISIS LÉXICO O AL
- TOKENS ó COMPONENTES LÉXICOS, PATRONES Y LEXEMAS
- REPRESENTACIÓN DEL ANALISIS LÉXICO
- IMPLEMENTACIÓN DE UN ANALIZADOR LÉXICO
- ACCIONES SEMÁNTICAS
- TRATAMIENTO DE ERRORES
- AL y LENGUAJES DE PROGRAMACIÓN

ANALISIS LÉXICO (AL) o SCANNER



FUNCIONES DEL ANALIZADOR LÉXICO O AL

- Leer los caracteres de la entrada.
- Generar una secuencia de componentes léxicos (TOKENS) que serán entregados al Analizador Sintáctico.
- Eliminar comentarios, delimitadores (espacios, símbolos de puntación, fin de línea).
- Relacionar los mensajes de error con las líneas del programa fuente.
- Introducir los identificadores en la tabla de símbolos.

ASPECTOS DEL ANÁLISIS LÉXICO O AL

- Diseño más sencillo
  Gramática más simple que la del parser: gramática regular
- Mejora la eficiencia
  Gran parte del tiempo de compilación se consume en la lectura y exploración de caracteres
- Mejora la transportabilidad.
  Se pueden tener varias versiones del scanner una para distintos códigos (EBCDID, ASCII, ...), con el mismo parser.
- Descarga el análisis sintáctico

TOKENS ó COMPONENTES LÉXICOS, PATRONES Y LEXEMAS

- Componente léxico o token: símbolo terminal de la gramática del analizador sintáctico
- Varias cadenas diferentes en la entrada pueden dar el mismo token a la salida
- Token: se describe mediante un patrón
- Lexema: conjunto de caracteres que concuerdan con el patrón de un token
- En la mayoría de lenguajes son tokens: Palabras clave, operadores, identificadores, constantes, cadenas literales y signos de puntación.
- En la tabla de símbolos se almacenan principalmente los identificadores y sus atributos

TOKENS ó COMPONENTES LÉXICOS, PATRONES Y LEXEMAS

COMPONENTE LEXICO O TOKEN	PATRON (regla)	LEXEMA
<if>	if→if	if
<then>	then→then	then
<id>	id→letra(letra|digito)* id→(a|…|z|A|…|Z)(a|…|z|A|…|Z|0|…|9)*	x1, a, b
<rel>	rel→<|>|<=|>=|<>|=	>
<asig>	asig→:=	:=
<op>	op→+|-|*|/|%	+
<num>	num→digito^{+} num→(0|…|9) (0|…|9)*	3

LINEA DE PROGRAMA FUENTE: if x1>3 then a:=b+x1

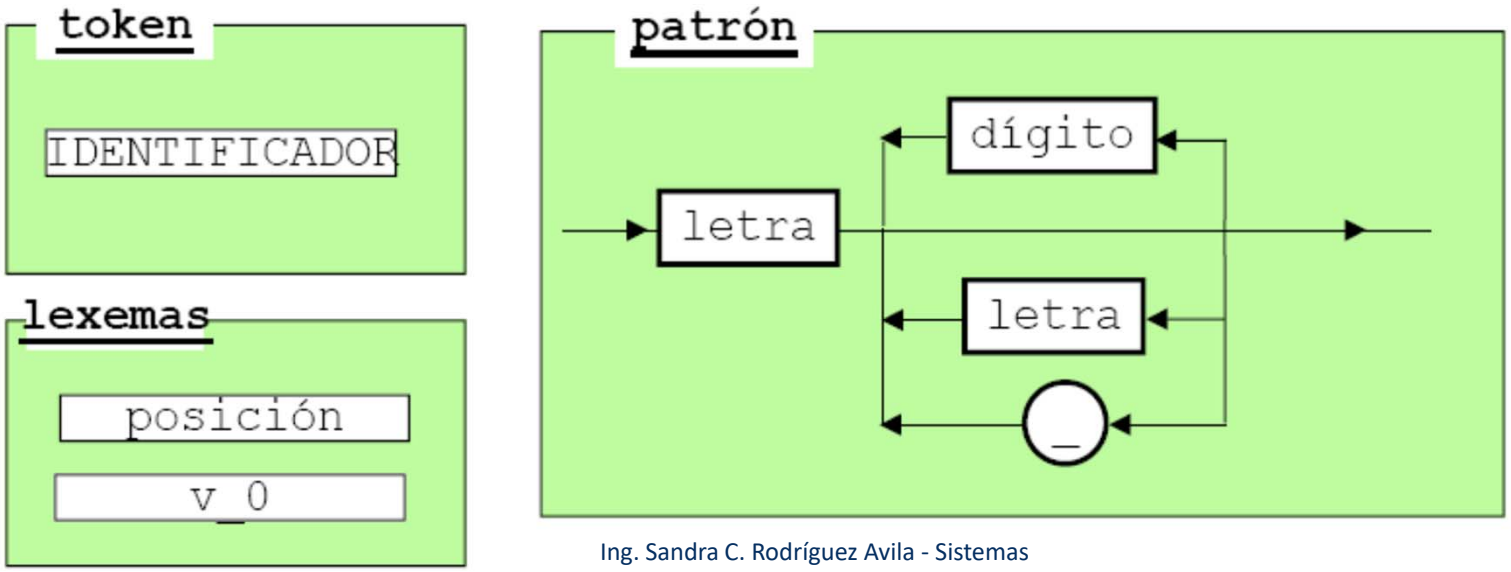
COMPONENTES LEXICOS: <if> <id><rel,>><then><num><id> <asig><id><op,+><id>

TOKENS ó COMPONENTES LÉXICOS, PATRONES Y LEXEMAS

- Ejemplo: Identificadores

– un token se corresponde con un patrón

– un token se puede corresponder con muchos lexemas



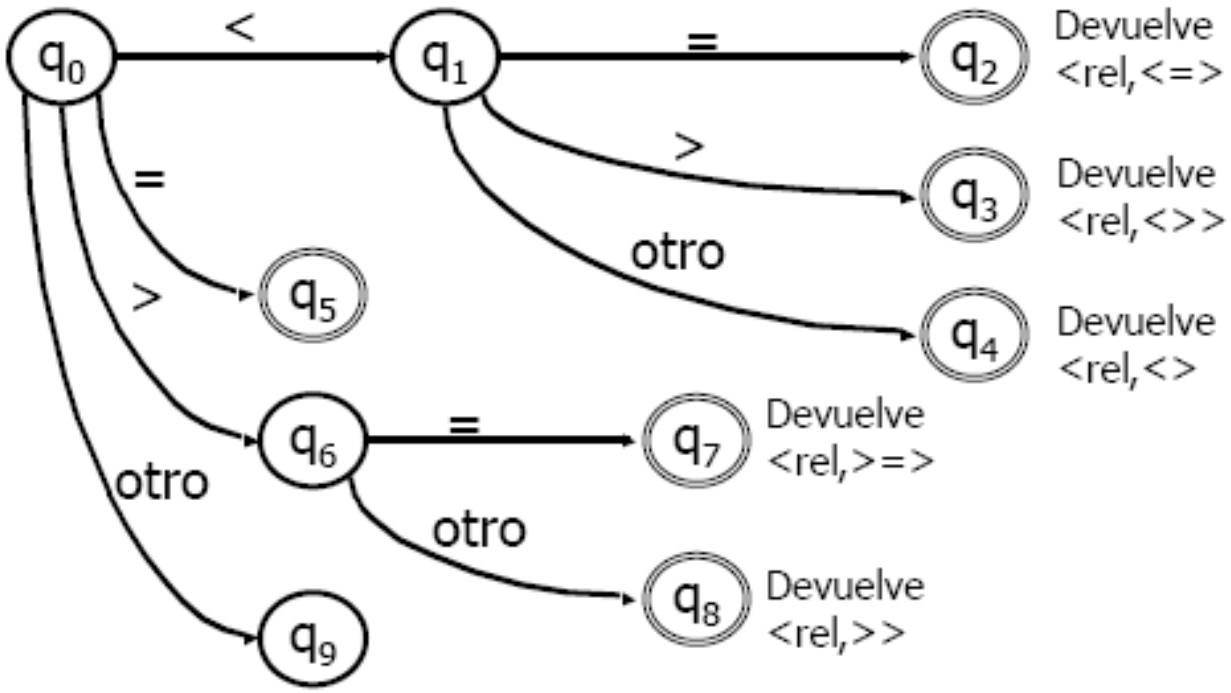
REPRESENTACIÓN DEL ANALISIS LÉXICO

- Expresión Regular
  (a|..|z|A|..|Z) (a|..|z|A|..|Z)*
- Autómata finito (diagrama o tabla de transición)

	a	...	z	A	...	Z	0	...	9
$\rightarrow q_0$	q_1	...	q_1	q_1	...	q_1	ϕ	...	ϕ
$*q_1$	q_1	...	q_1	q_1	...	q_1	q_1	...	q_1

- Gramática Lineal (regular)
  - S::= aR | ... | zR | AR | ... | ZR
  - R::= aR | ... | zR | AR | ... | ZR | 0R | ... | 9R | λ

EJEMPLO: ANALIZADOR LÉXICO DE RELACIONES MATEMATICAS



IMPLEMENTACIÓN DE UN ANALIZADOR LÉXICO

- Utilizando un lenguaje de alto nivel
  - Programación
  - Tabla compacta
  - Hashing
  - Autómata programado
- Utilizando ensamblador
  - Más eficiente
  - Más difícil
- Utilizando un generador de Analizadores Léxicos (LEX, JLEX, JFLEX)
  - Más cómodo
- Consejo: Ordenar las reglas/transiciones de acuerdo a la frecuencia de utilización

IMPLEMENTACIÓN DE UN AL: PROGRAMACIÓN

- Dos punteros de lectura:
  - Puntero actual (PA, “current pointer”): El último carácter aceptado
  - Puntero de búsqueda (PB, “lookahead pointer”): El último carácter leído
- Funciones de lectura:
  - GetChar: mueve el PB hacia delante y devuelve el siguiente carácter
  - Fail: mueve el PB a donde está el PA
  - Retract: mueve el PB un carácter hacia atrás
  - Accept: mueve el PA a donde está el PB
- Predicados:
  - IsLetter(x):= xÎ [A..Za..z]
  - IsDigit(x):= xÎ [0..9]
  - IsDelimiter(x):= xÎ [.,;<newline><eof>]
- Acciones:
  - InstallName(id): introduce un nombre en la tabla de símbolos

IMPLEMENTACIÓN DE UN AL: PROGRAMACIÓN

Identificador::= letra·(letra+dígito)*

- Pseudocódigo

```
c:= GetChar
If IsLetter(c) Then
   identificador:=""
   Repeat
        identificador:=identificador+c
        c:=GetChar
   Until not(IsLetter(c) OR IsDigit(c))
   Retract
   token:=(Id, Install(identificador))
   Accept
   Return (token)
Else Fail
```

Funciones de lectura:

- GetChar: mueve el PB hacia delante y devuelve el siguiente carácter
- Fail: mueve el PB a donde está el PA
- Retract: mueve el PB un carácter hacia atrás
- Accept: mueve el PA a donde está el PB

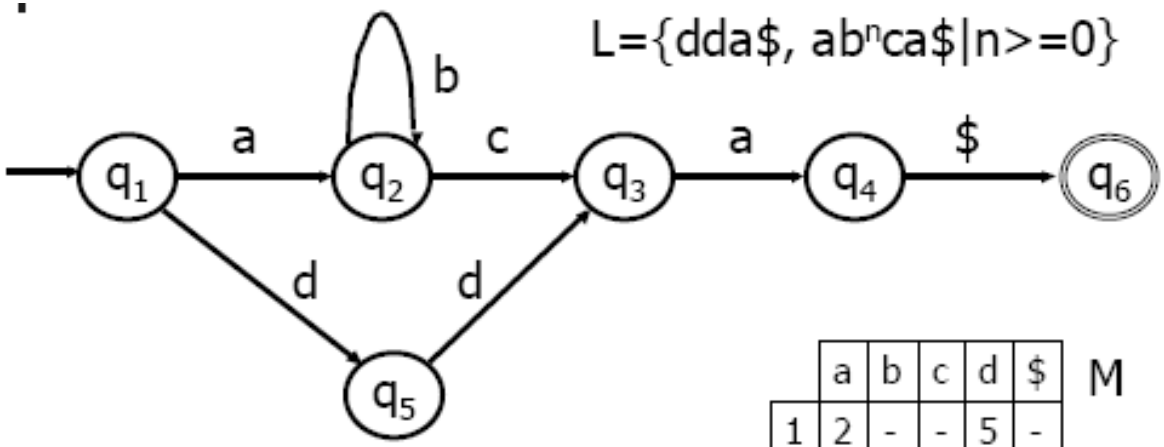
IMPLEMENTACIÓN DE UN AL: PROGRAMACIÓN

Entero::= dígito^{+}

- Pseudocódigo

```
c:= GetChar
If IsDigit(c) Then
  Valor:=Convertir(c)
  c:=GetChar
  While IsDigit(c) do
       Valor:=10 * Valor + Convertir(c)
       c:=GetChar
  EndWhile
  Retract
  token:=(Entero, Valor)
  Accept
  Return (token)
Else Fail
```

IMPLEMENTACIÓN DE UN AL: TABLA COMPACTA



	a	b	c	d	$
1	2	-	-	5	-
2	-	2	3	-	-
3	4	-	-	-	-
4	-	-	-	-	6
5	-	-	-	3	-
6	-	-	-	-	-

M

IMPLEMENTACIÓN DE UN AL: TABLA COMPACTA

- Para ahorrar memoria se guardan los elementos no nulos en una matriz VALOR de dos valores y con tantos elementos como posiciones no nulas de M
- Se crea otra tabla, PRIFIL, con el primer VALOR de cada línea
- Número de elementos (no nulos) de VALOR para esa línea

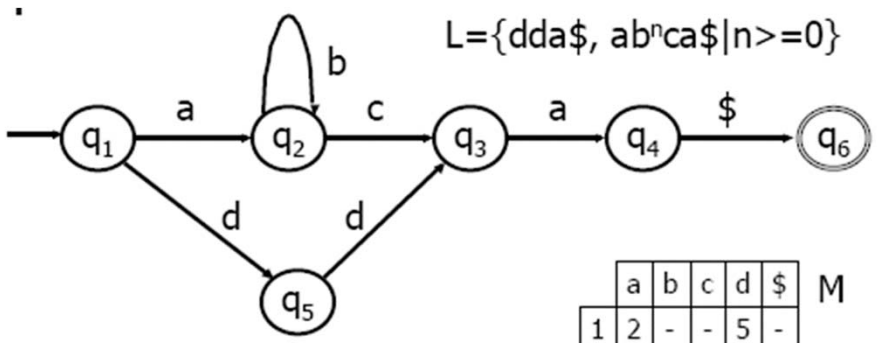
	VALOR	COL
1	2	1
2	5	4
3	2	2
4	3	3
5	4	1
6	6	5
7	3	4

	PRIFIL	FIL
1	1	2
2	3	2
3	5	1
4	6	1
5	7	1
6	0	0

- M(2,3)?
- PRIFIL(2)=3
- FIL=2, esto significa que los elementos 3 y 4 de VALOR contienen las transiciones del estado 2, de los dos el que tiene valor COL=3 tiene una transición a 3

IMPLEMENTACIÓN DE UN AL: AUTOMATA PROGRAMADO

```
estado:=1
while estado<>6 do
  Lee car {devuelve en car el siguiente carácter leído}
  case estado of
  1: if car="a" then estado:=2 else
          if car="d" then estado:=5 else error
  2: if car="c" then estado:=3 else
          if car="b" then estado:=2 else error
  3: if car="a" then estado:=4 else error
  4: if car="$" then estado:=6 else error
  5: if car="d" then estado:=3 else error
  6: error
  end case
end while
```



M

	a	b	c	d	$
1	2	-	-	5	-
2	-	2	3	-	-
3	4	-	-	-	-
4	-	-	-	-	6
5	-	-	-	3	-
6	-	-	-	-	-

IMPLEMENTACIÓN DE UN AL: GENERADOR DE AL

```
%{LE, LT, GE, GT, EQ, NE, IF, THEN, ELSE %}
Delimitador                 [\t\n]
blancos          {delimitador}+
letra                       [A-Za-z]
Digito                      [0-9]
id                                  {letra}({letra}|{digito})*
Numero                      {digito}+(|.{digito}+)?(E [+|-]?{digito}+)?
%                           %
{blancos} {}
if                          {return(IF)}
then                        {return(THEN)}
while                       {return(WHILE)}
... ...
{id}                        {yylval = InstallName(); return(ID)}
{numero}         {yylval = InstallName(); return(NUMERO)}
"<"                         {yylval = LT; return(RELACION)}
... ...
%%
```

ACCIONES SEMÁNTICAS

- El autómata no debe mostrar únicamente “si” o “no”
- Realmente realiza una “Traducción”
- El explorador realiza comprobaciones en la tabla de símbolos
- Las acciones semánticas más comunes son:
  - AÑADIR: concatena caracteres para construir símbolos
  - LEECAR: lee caracteres de la entrada
  - VER: Comprueba si un identificador está o no en la TdS
  - ADD: Añade un identificador a la TdS

TRATAMIENTO DE ERRORES

- Hay pocos detectables por el analizador léxico
  - then:=3+x1 (¿=if..then?. ¿=identificador?)
- Detectables
  - Número de caracteres de los identificadores
  - Caracteres ilegales
  - Otros (si el lenguaje no admite .5 en lugar de 0.5)...
- Acciones posibles
  - Borrar un carácter
  - Insertar un carácter
  - Reemplazar un carácter
  - Intercambiar dos caracteres
- Programa con K errores: hacen falta K cambios para poder ser correcto
- No se suelen utilizar las acciones de corrección de errores por ser muy costosas

AL y LENGUAJES DE PROGRAMACIÓN

- Delimitador: Carácter que delimita el token sin pertenecer a él
- Palabra reservada: se prohíbe el uso de determinadas palabras que tienen un significado específico y único en el lenguaje
- Se pueden clasificar los lenguajes de programación por el uso de los delimitadores y palabras reservadas:
  - Delimitadores blancos con palabras reservadas

    Caso más sencillo de lenguaje (PASCAL, COBOL)
  - Delimitadores blancos sin palabras reservadas

    PL/I
  - Blancos se ignoran sin palabras reservadas

    El tipo más difícil de lenguaje, aparecen ambigüedades (FORTRAN)
  - Blancos se ignoran con palabras reservadas

BIBLIOGRAFIA

- SANCHIS F. J., GALAN C. ***Compiladores. Teoría y Construcción.*** 1986. Madrid. Editorial Paraninfo.
- http://www.lcc.uma.es/~galvez/ftp/libros/Compiladores.pdf
- http://di002.edv.uniovi.es/~cueva/publicaciones/monografias/10_Conceptos_Basicos_Procesadores_Lenguaje.pdf

RECURSOS GRAFICOS

- Pixabay
- Pexels
- Icon-Icons