5. ANÁLISIS LÉXICO

5.1. ANALIZADORES LÉXICOS

La principal función del analizador léxico consiste en leer los caracteres que componen el programa fuente, agruparlos en secuencias, llamadas lexemas, y producir como salida un componente léxico para cada lexema. La secuencia de componentes léxicos se envía al analizador sintáctico.

Como el analizador léxico es la fase del compilador que lee el programa fuente también realiza otras funciones secundarias. Una de estas funciones es eliminar del programa fuente comentarios y espacios en blanco (espacios, tabulaciones y nuevas líneas). Otra de las funciones es relacionar los mensajes de error del compilador con el programa fuente. Por ejemplo, el analizador léxico puede:

- contar el número de caracteres de nueva línea leídos, para asociar un número de línea con cada mensaje de error.
- hacer una copia del programa fuente incluyendo los mensajes de error.

El analizador léxico se puede dividir en dos fases (ver Figura 5.1):

1. **Escáner** para eliminar los comentarios y compactar los espacios en blanco consecutivos en uno solo.
2. **Analizador léxico** para realizar el análisis léxico propiamente tal.

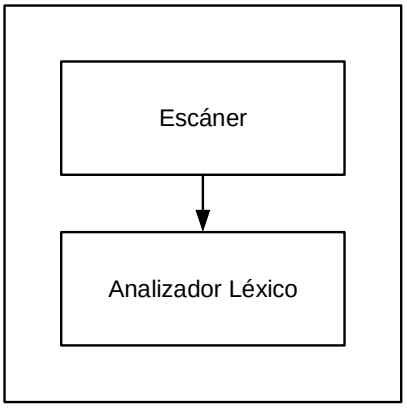


Figura 5.1. Estructura del analizador léxico.

Componentes léxicos

Un componente léxico (*token*) es un par que consta de un nombre y un valor de atributo opcional que apunta a una entrada en la tabla de símbolos para este componente léxico.

"Los componentes léxicos se tratan como símbolos terminales de la gramática del lenguaje fuente" (Aho, 1990, p. 88).

En la mayoría de los lenguajes de programación, se consideran componentes léxicos: (ver Tabla 5.1)

- Palabras claves.[8]
- Operadores.
- Identificador.
- Constantes:
  - Número.
  - Literal.
- Signos de puntuación.

Tabla 5.1. Componentes léxicos.

Componente léxico	Ejemplo de lexema
=	=
identificador	pi
número	3.14159
literal	"¡Hola, mundo!"

Cuando el analizador léxico encuentra un lexema que corresponde a un identificador, debe introducir ese lexema en la tabla de símbolos.

Errores léxicos

Posibles acciones de recuperación de errores son (Aho, 1990, p. 90):

- Borrar un carácter que sobra.
- Insertar un carácter que falta.
- Reemplazar un carácter incorrecto por otro correcto.
- Intercambiar dos caracteres adyacentes.

[8] Una palabra clave es una palabra reservada si no se puede usar también como un identificador.

Implementación

Dentro de los métodos de implementación de un analizador léxico se tienen:

1. Escribir el analizador léxico en un lenguaje de programación.
2. Utilizar un generador de analizadores léxicos.

5.2. GENERADORES DE ANALIZADORES LÉXICOS

Estas herramientas generan automáticamente analizadores léxicos, a partir de una descripción de los componentes léxicos del lenguaje fuente, utilizando expresiones regulares.

5.2.1. LEX

La figura 5.2 indica cómo generar un analizador léxico, utilizando LEX. Primero, se especifica el analizador léxico en el programa en LEX **lex.l**. Después, el compilador de LEX traduce **lex.l** al programa en C **lex.yy.c**. Por último, el compilador de C traduce **lex.yy.c** al programa objeto **a.out**, que es el analizador léxico que transforma un programa fuente en una secuencia de componentes léxicos.

El archivo **lex.yy.c** consta de una tabla de transiciones construida a partir de las expresiones regulares de **lex.l**.

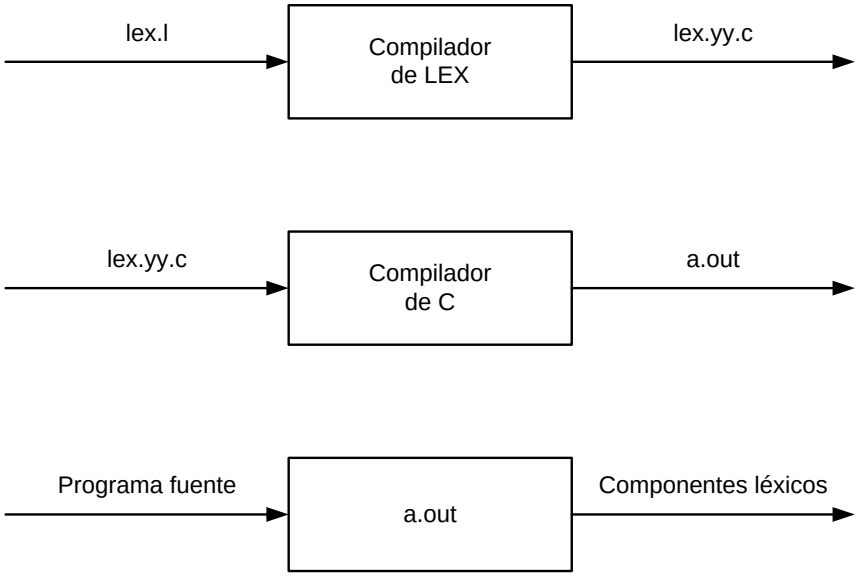


Figura 5.2. Analizador léxico con LEX.

5.2.1.1. ESTRUCTURA DE UN PROGRAMA EN LEX

Un programa en LEX posee la siguiente estructura:

```
%{
declaraciones
%}
definiciones regulares
%%
reglas de traducción
%%
funciones auxiliares
```

La declaraciones incluyen declaraciones de variables y constantes manifiestas[9] usando instrucciones #**define** de C. Las declaraciones se copian literalmente al archivo **lex.yy.c**.

Las definiciones regulares constan de un nombre y una expresión regular representada por ese nombre. Las definiciones regulares que se utilizan en definiciones posteriores o como componentes de las expresiones regulares en las reglas de traducción se encierran entre **{** y **}**.

Las reglas de traducción tienen la siguiente forma:

r_1 { A_1 }
r_2 { A_2 }
r_3 { A_3 }
. . . . . .
r_n { A_n }

donde $\mathbf{r_i}$ es una expresión regular y cada $\mathbf{A_i}$ es la acción del analizador léxico cuando $\mathbf{r_i}$ concuerda con un lexema. Las acciones son fragmentos de programa en C y se copian literalmente al archivo **lex.yy.c**.

Las funciones auxiliares se utilizan en las acciones y también se copian literalmente al archivo **lex.yy.c**.

[9] Una constante manifiesta es un identificador que representa una constante.

El analizador léxico generado por LEX trabaja en conjunto con el analizador sintáctico del siguiente modo. Cuando el analizador sintáctico llama al analizador léxico, este comienza a leer el resto de su entrada, un carácter a la vez, hasta que encuentra el prefijo más largo de la entrada que concuerda con una de las expresiones regulares $\mathbf{r_i}$. Entonces, ejecuta la acción asociada $\mathbf{A_i}$. Generalmente, $\mathbf{A_i}$ devolverá el control al analizador sintáctico, pero si no lo hace (debido a que $\mathbf{r_i}$ describe espacios en blanco o comentarios), el analizador léxico procede a buscar más lexemas, hasta que una de las acciones correspondientes provoque que el control regrese al analizador sintáctico, mediante una instrucción **return** explícita. El analizador léxico devuelve un solo valor, el nombre del componente léxico, al analizador sintáctico. Para pasar un valor de atributo del lexema se utiliza la variable entera global **yyval**. Además, el analizador léxico generado por LEX establece de manera automática las siguientes variables:

- **yytext** que es un puntero al primer carácter del lexema.
- **yyleng** que es un entero que indica la longitud del lexema.

LEX utiliza las siguientes reglas para seleccionar el lexema cuando varios prefijos de la entrada coinciden con una o más expresiones regulares:

1. Preferir el prefijo más largo.
2. Si el prefijo más largo coincide con dos o más expresiones regulares entonces preferir la expresión regular que aparece primero en el programa en LEX.

5.2.1.2. EXPRESIONES REGULARES EN LEX

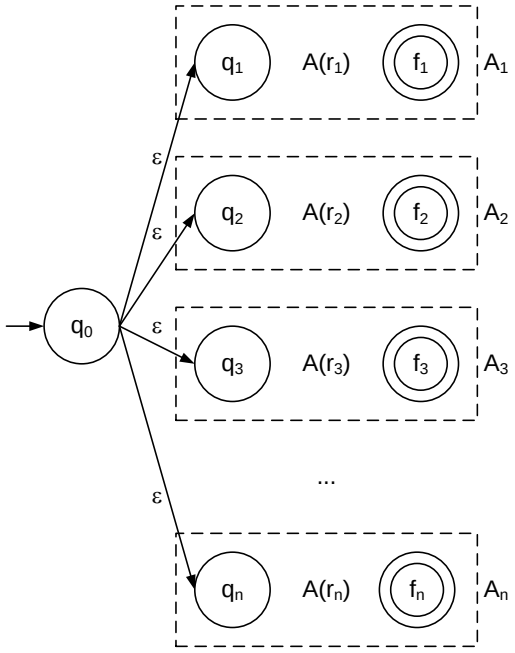
La tabla 5.2 describe la notación extendida de expresiones regulares en LEX.

Tabla 5.2. Expresiones regulares en LEX.

Expresión Regular	Descripción	Ejemplo
c	cualquier carácter c que no sea operador	a
\c	el carácter c literalmente	*
"s"	el string s literalmente	"**"
.	cualquier carácter excepto nueva línea	a.*u
^	el inicio de línea	^a
$	el fin de línea	a$
[s]	cualquier carácter que esté en el string s	[aeiou]
[^s]	cualquier carácter que no esté en el string s	[^aeiou]
r*	cero o más r	a*
r+	una o más r	a+
r?	cero o una r	a?
r{m,n}	entre m y n ocurrencias de r	a{2,4}
r{m,}	m o más ocurrencias de r	a{2,}
r{m}	m ocurrencias de r	a{4}
r_1 r_2	r_1 seguida de r_2	au
r_1 | r_2	r_1 o r_2	a | u
(r)	r	(a)
r_1 / r_2	r_1 cuando va seguida de r_2	a/u
<<EOF>>	fin de archivo	a<<EOF>>

5.2.1.3. LEX → AFN-ε

r_1 { A_1 }
r_2 { A_2 }
r_3 { A_3 }
…
r_n { A_n }



Ejemplo:
a { A_1 }
abb { A_2 }
a^*b^+ { A_3 }

ω_1 = aaba
ω_2 = aba
ω_3 = abba