[[1]](#footnote-1)

Analizador Lexicográfico

(Diciembre 2016)

Caiza Llumitaxi Jonathan Paul, Guzmán Alarcón Darwin Santiago.

**Escuela Politécnica Nacional del Ecuador**

***Resumen***—En esta investigación se manifiesta la importancia que tienen los analizadores lexicográficos en la creación de compiladores a partir del análisis investigativo realizado, para una determinada gramática de lenguaje de programación. Esta primera parte del proyecto se enfoca a la creación del analizador lexicográfico implementado en lenguaje C, utilizando la librería Lex para crear el analizador lexicográfico

***Índice de Términos—*** Analizador Lexicográfico, Compilador, Lenguaje de programación, gramática.

# **Introducción**

La presente investigación nos da a conocer la herramienta flex como un generador de escáneres, para el reconocimiento de patrones en un texto, cuya descripción se encuentra en forma de parejas de expresiones regulares y códigos C, denominadas reglas. **[1]**Al compilar el archivo “\*.l” con la herramienta flex nos genera un archivo “lex.yy.c”, el cual será compilado con el “gcc” para el ejecutable del compilador y poder hacer el ingreso de datos para analizar.

Dentro de la estructura para la generación de un analizador lexicográfico tenemos: la parte de declaraciones, reglas de traducción y las funciones auxiliares, las cuales representa el cuerpo donde definiremos todos los elementos de nuestra gramática, los toques, las expresiones regulares, etc.

Los errores generados al momento de ingresar un comando erróneo en analizador deben ser detectados y expresados con la información de la línea de comando del error con el tipo de fallo producido. Una forma para detectar un error es con el token punto ‘.’ , nos indicara el ingreso de comandos diferentes al definidos en nuestra gramática.

# **OBJETIVOS**

***Objetivo general***

* Conocer las definiciones, funcionamiento, estructuración de un analizador lexicográfico.

***Objetivo especifico***

* Crear un analizador lexicográfico que determine los tokens y detecte errores lexicales.

# **MARCO TEORICO [1] [2] [3]**

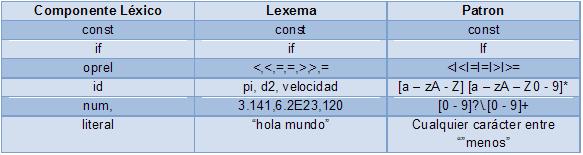
***Tokens. -*** Representan una unidad de información como son palabras reservadas, secuencia de caracteres, etc.

***Lexema. -*** Representan cadenas de caracteres que se pueden tratar juntas como una unidad lexical.

***Patrón. -*** Describe la forma y el conjunto de lexemas que pueden ajustarse a un determinado componente léxico.

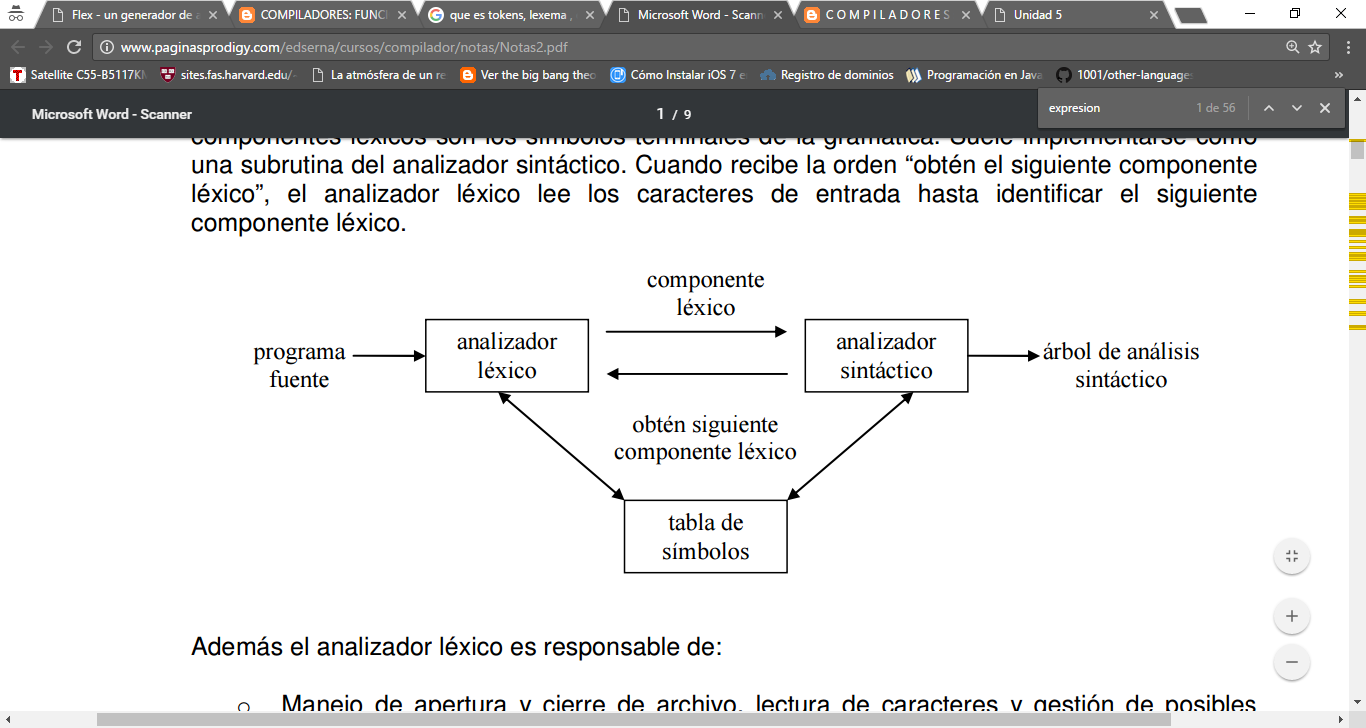
***Expresiones Regulares. –*** Representan patrones que sirven para encontrar una combinación de caracteres determinado dentro de un texto

***Analizador Léxico (Scaner). -*** Tiene la función de leer un texto y dividirlo en tokens para reconocer que representa dicho token dentro del compilador.



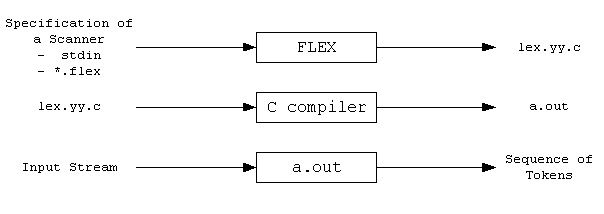
**Fig. 1** Tabla de ejemplos de tokens, lexemas y patornes

<http://10380054.galeon.com/images/12.JPG>



**Fig. 2** Esquema de la estructura del Analizador Lexicográfico **[3]**

***Herramienta Flex. –*** Esta herramienta es un generador de scaners que nos reconoce los patrones de un texto según los defínasenos previamente, esta herramienta esta descrita por expresiones regulares y códigos C, al compilar un archivo \*.l este nos genera un nuevo archivo “lex.yy.c”, que será el que compilaremos para generar el ejecutable y así poder usar el analizador lexicográfico.



**Fig. 3** Proceso de generación del analizador lexicográfico con la herramienta flex

<http://alumni.cs.ucr.edu/~lgao/teaching/Img/flex.jpg>

La descripción de la estructura de flex esta expresada de la siguiente forma:

%%

DECLARACIONES

%%

REGLAS DE TRADUCCIÓN

%%

FUNCIONES AUXILIARES

En la parte de las declaraciones se coloca lo que se va usar para la traducción.

%{

/\*DEFINICION DE CONSTANTES\*/

%}

delim [“ ”,\n,\t]

ws {delim}+

letra [a-zA-Z]

num [0-9]

id {letra}({letra}[num])\*

numero {num}+(\.{num}+)?(E[+-]?{num}+)?

.

.

.

En la parte de las reglas de traducción se coloca lo que significa cada token para el compilador.

%%

{ws}

if {return (IF);}

then {return (THEN);}

else {return (ELSE);}

{id} {yyval=(INT) installID (ID)

return (ID);}

{numero} {yyval=(INT) instalNUM(NUMERO)

return (ID);}

“=” {yyval=EQ instalNUM(RELOAP);}

“<” {yyval=LT instalNUM(RELOAP);}

“<=” {yyval=LE instalNUM(RELOAP);}

.

.

.

En la parte de las funciones auxiliares se coloca las funciones necesarias q se van a utilizar.

%%

Int installID(){

}

Int installNUM(){

}

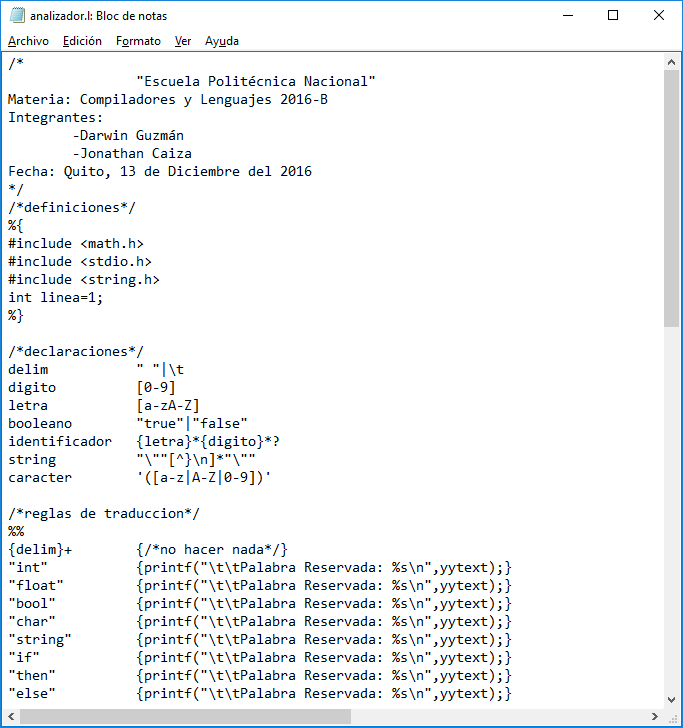
Int main(){

}

# **Ejecución del proyecto**

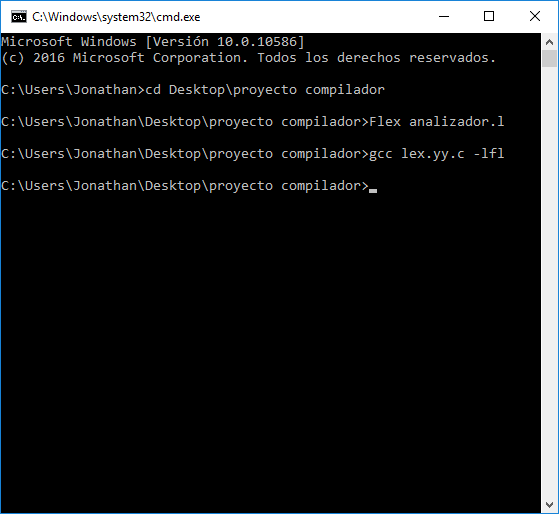
## **Código en Flex**

El respectivo código se encuentra en github, la estructura del analizador lexicográfico se presenta a continuación.



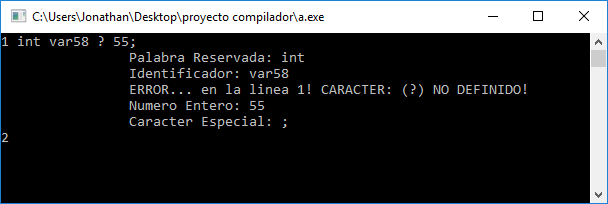
**Fig. 4** Código del Analizador Lexicográfico

## **Compilación y creación del ejecutable**



**Fig. 4** Compilación y creación del ejecutable del Analizador Lexicográfico

## **Prueba del Analizador Lexicográfico**



**Fig. 4** Prueba de ejecución del Analizador lexicográfico

# **Conclusión**

Concluimos que el analizador lexicográfico es un de las partes más fundamental en la creación del compilador ya que es el encargado del reconocimiento te identificadores, palabras claves números, etc., reconocimiento de patrones léxicos en un texto.

La herramienta flex es la parte fundamental para la creación del analizador lexicográfico y la creación del ejecutable, gracias a los archivos que este genera.

**References**

|  |  |
| --- | --- |
| [1] | V. Paxson, «Un generador de analizadores léxicos rápidos.,» Abril 1995. [En línea]. Available: http://es.tldp.org/Manuales-LuCAS/FLEX/flex-es-2.5.html#SEC2. |
| [2] | C. 7. 1, «UNIDAD 2: ANÁLISIS LÉXICO,» 20 Noviembre 2010. [En línea]. Available: http://bloggcompliladores7mo1.blogspot.com/2010/11/unidad-2\_20.html. |
| [3] | E. Serna-Pérez, «Análisis léxico (Scanner),» UAA – Sistemas Electrónicos , [En línea]. Available: http://www.paginasprodigy.com/edserna/cursos/compilador/notas/Notas2.pdf. |

1. [↑](#footnote-ref-1)