UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

INGENIERÍA EN COMPUTACIÓN

COMPILADORES

ANALIZADOR LÉXICO

GRUPO:02

FECHA DE ENTREGA LÍMITE: 19*/SEPTIEMBRE/2017*

EQUIPO DE TRABAJO:

- CHÁVEZ DELGADO JORGE LUIS
- SÁNCHEZ NERI DAVID YAXKIN

SEMESTRE 2018-1

INDICE

1.- DESCRIPCIÓN DEL PROBLEMA

Se elaborará un analizador léxico en flex y ANSI C que reconozca los componentes léxicos pertenecientes a las siguientes clases:

Clase	Descripción
0	Identificadores (sólo letras minúsculas)
1	Palabras reservadas
2	Operador de asignación =
3	Operadores relacionales
4	Operadores aritméticos
5	Símbolos especiales , ; [ ] ( )
6	Constantes numéricas enteras (base 10)
7	Constantes numéricas reales (siempre con .)
8	Constante cadena (entre comillas)

Los tokens serán representados en una lista ligada simple con dos campos:

- Campo1: Para la clase
- Campo2: Para el valor

2.-EXPRESIONES REGULARES POR CLASE

Se utilizará la notación de flex para definir las expresiones regulares.

- Clase 0, identificadores (sólo minúsculas):
  let [a-z]
  ident {let}({let}|{dig}|0){0,15}
- Clase 1, palabras reservadas:
  palRes ENT|REAL|SI|SINO|MIENTRAS|HASTA|ESCRIBE|LEE
- Clase 2, operador de asignación:
  opAsig =
- Clase 3, operadores relacionales:
  opRel (>|<)=?|(!|=)=
- Clase 4, operadores aritméticos:
  opArit MAS|MENOS|MULTIPLICA|DIVIDE
- Clase 5, símbolos especiales (), ; []
  simbEsp [,;\[\]\(\)]
- Clase 6, constantes numéricas en base 10:
  dig [1-9]
  *entBD {dig}({dig}|0)**
- Clase 7, constantes numéricas reales, siempre con . :
  dig [1-9]
  *entBD {dig}({dig}|0)**
  numReal {entBD}\.[0-9]+
- Clase 8, constante cadena, entre comillas:
  constCad \"[^\"]+\"

3.- PROPUESTA DE SOLUCIÓN /ANÁLISIS

Para la fase de análisis léxico, se hará uso de flex el cual nos permitirá manejar con mayor facilidad las expresiones regulares puesto que ya es un automáta finito deterministico.
Además de flex también haremos uso de lenguaje C para realizar el almacenamiento e impresión en consola. Utilizaremos una lista ligada simple para ello.

4.-DISEÑO E IMPLEMENTACIÓN

- Para las tablas estáticas(operadores aritméticos, operadores relacionales y palabras reservadas) se definirán arreglos.
- Para el manejo de los tokens, tabla de símbolos especiales y tabla de cadenas se realizarán funciones las cuales almacenarán sus correspondientes datos.
- El método de búsqueda en cuanto a cadenas e identificadores es elemento por elemento, es decir, si se encuentra alguna cadena o identificador ya existente se compara con el elemento leído en ese momento.

5.- ¿CÓMO CORRER EL PROGRAMA?

Para este proceso necesitamos una terminal linux, y tener instalado el compilador gcc y flex, en caso de no tenerlos instalados, en sistemas operativos basados en Debian se instalaría de la siguiente manera:

Para gcc:

$ sudo apt-get install gcc

Para flex:

$ sudo apt-get install flex

Teniendo ya instalado lo anterior procedemos a ejecutar nuestro código escrito en flex y lenguaje C de la siguiente manera:

$ flex analizadorL.l

La salida al ejecuar el comando anterior nos devolverá un archivo llamado ***lex.yy.c*** el cuál podrá ser compilado con ayuda de gcc de la siguiente forma:

$ gcc lex.yy.c -lfl

El resultado de la compilación nos generará un archivo ejecutable el cual podremos correr y visualizar el análisis léxico de nuestro futuro compilador, para esto se utilizó un archivo de prueba llamado “entrada” de la siguiente manera:

$ ./a.out entrada

6.-CONCLUSIONES

Chávez Delgado Jorge Luis:
Se puede concluir que el objetivo de este proyecto se alcanzó ya que logramos construir el analizador léxico que abarca los componentes léxico de cada clase, así como la interpretación de comentarios para ignorarlos. Fue de gran ayuda el realizarlo ya que se pudo comprender mejor este proceso y se reforzaron los conceptos vistos en la teoría.

Sánchez Neri David Yaxkin:
Se cumplió el objetivo de escribir el programa de manera satisfactoria, siendo lo más interesante pensar cómo guardar nuestras tablas, tanto estáticas como dinámicas. Además se pudo observar la capacidad que nos ofrece flex para desarrollar un analizador léxico de manera fácil. Al finalizar las pruebas se pudo comprender mejor el funcionamiento de un analizador léxico.