

Primer programa Analizador Léxico

Objetivo

Elaborar un analizador léxico en flex o C que reconozca los componentes léxicos pertenecientes a las clases abajo descritas y que son del lenguaje Pascal—acordadas en clase.

Descripción del problema

Se deberán diseñar las expresiones regulares correctas para que el analizador léxico puede definir de manera correcta lo que le pedimos, algunas tablas de clases ya están definidas por lo tanto son estáticas dado que se conoce su tamaño desde antes de ejecutar el programa, las tablas de identificadores y cadenas no están definidas dado que se crearán cuando se encuentren los componentes correspondientes por lo tanto éstas tablas deberán manejarse con memoria dinámica y cargarse en tiempo de ejecución.

Las expresiones regulares usadas para identificar los componentes de cada clase son las siguientes:

Expresiones regulares

/*Digitos 0 - 9*/

dig [0-9]

/*Letras a - z*/

char [a-zA-Z]

/*[0]Palabras reservadas*/

preservadas

AND|ARRAY|BEGIN|DIV|DO|DOWNTO|ELSE|END|FOR|IF|MOD|NOT|OR|PROGRAM|REPEAT|THEN

/*[1]Identificadores definidos por el usuario*/

ident {char}({char}){dig}*

/*[2]Tipos de datos estandar*/

tdatos boolean|char|int|real|text

/*[3]Funciones estandar*/

festandar abs|chr|cos|eof|eoln|exp|ln|sin|sqr|sqrt|trunc

/*[4]Procedimientos estandar*/

pestandar get|put|read|readln|write|writeln

/*[5]Operadores aritmeticos*/

oparit [+|-*]/

/*[6]Operadores relacionales*/

oprela <|<=|=|<>|>|>=

/*[7]Simbolos especiales*/

simesp [,;::()\\[]]

/*[8]Constantes enteras*/

cteint {dig}+

```

/*[9]Constantes reales*/
ctereal {cteint}+(\.)(\{dig\})+
/*[10]Cadenas*/
cadena ('').*(')
/*[11]Operador de asignacion*/
opasig :=
/*Comentarios*/
coment (\*).*(\*)

```

Desarrollo del sistema

Los principales pasos a seguir para el desarrollo del sistema fueron los siguientes:

1. Se leyeron los requisitos e hicieron pruebas en papel para comprender a que se refiere cada uno.
2. Se modeló la estructura general del programa con base en su entrada y salida.
3. El desarrollo se hizo con respecto a la estructura de los programas en flex, por lo cual lo primero que se desarrolló fueron las expresiones regulares, posteriormente las reglas y al ultimo el programa.
4. Para el programa se analizaron los requisitos específicos de cada objetivo para el diseño de funciones.
5. Se desarrollaron las funciones individualmente.
6. Una vez que funcionaron todas las funciones se integraron.
7. Ya que el programa estaba totalmente integrado se hicieron pruebas globales y depuración de código.

Se usaron listas ligadas para cargar las tablas dinámicas de identificadores y cadenas esto dado que las tablas contenían cadenas y la manera de cargarlas a una lista más viable que se contempló fue una lista ligada.

Las búsquedas se hicieron lineales se recorrieron todas las tablas comparando elemento con elemento hasta coincidir con la búsqueda, no fue la mejor manera de hacerlo pero para implementar los algoritmos para búsquedas de cadenas en c, se hubiera llevaría más tiempo con respecto a cómo se hizo.

Pruebas

Las pruebas del programa se hicieron con un programa de pascal descargado de internet, las tablas generadas se guardaron en un archivo llamado "salida.txt"

Los comando para correr el programa son

```

$ lex programa1.l
$gcc -ansi lex.yy.c -lfl
$./a.out <programa.txt>

```

*El nombre del archivo txt entre pico paréntesis puede tener otro nombre

Pruebas del programa

```
Terminal - jlma@jlma-pc:/home/jlma/Escritorio/Programa3
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda
[jlma@jlma-pc Programa3]$ lex programa1.l
[jlma@jlma-pc Programa3]$ gcc -ansi lex.yy.c -lfl
[jlma@jlma-pc Programa3]$ ./a.out programa.txt

*****Se descartaron comentarios*****
*****A continuacion se muestran los errores lexicos*****

$

Se genero el archivo con las tablas exitosamente
[jlma@jlma-pc Programa3]$
```

Archivo programa.txt con el que se hicieron algunas de las pruebas.

```

▼
programa.txt - Mousepad
Archivo  Editar  Búsqueda  Ver  Documento  Ayuda
A = (l*l) * 6
V = l3

PROGRAM EJER12;
  USES CRT;
  VAR lado:REAL;
  VAR area,volumen:REAL;
  $

BEGIN
  lado:=4;


  area:= (lado * lado) * 6;
  volumen:= sqr(lado) * lado;

  clrscr;

  write('AREA DEL HEXAEDRO');
  write(area);
  writeln(' m2');
  write('VOLUMEN DEL HEXAEDRO');
  write (volumen); write (' m3');

END.
```

Tablas que se generaron y guardaron en el archivo “salida.txt” (última tabla incompleta por la impresión de pantalla)

-----TABLA DE SIMBOLOS-----		
POSICION	NOMBRE DEL IDENTIFICADOR	TIPO
22	volumen	
21	area	
20	ClrScr	
19	lado	
18	lado	
17	volumen	
16	lado	
15	area	
14	lado	
13	REAL	
12	volumen	
11	area	
10	VAR	
9	REAL	
8	lado	
7	VAR	
6	CRT	
5	USES	
4	EJER12	
3	l3	
2	V	
1	l	
0	A	
-----TABLA DE CADENAS-----		
POSICION	CADENA	
3	' m3 '	
2	'VOLUMEN DEL HEXWM'	
1	' m2 '	
0	'AREA DEL HEXAED	
-----TOKENS-----		
CLASE	POSICION	
7	.	
0	7	
7	.	

Conclusiones

El programa para el manejo de expresiones regulares lex me pareció muy eficiente para aplicaciones de este tipo ya que si las expresiones se realizan correctamente se pueden realizar programa muy rápidos y eficientes al identificar componentes, el programa en general me pareció muy interesante, se me complicaron algunas cosas para el desarrollo en c principalmente en el uso de cadenas en listas pero se solucionaron después de investigar cómo se usaban algunos métodos aplicado a cadenas especialmente, al final el programa no fue lo mejor con respecto a buenas prácticas pero fue concluido así dado que cumple con las funciones principales y puede depurarse en un futuro.