Primer programa, Analizador Léxico

Este programa fue realiza de manera individual por:

José Donaldo Fernández Montenegro

Fecha de elaboración: martes 7 de marzo de 2017

Descripción del problema:

Se requiere Elaborar un analizador léxico en *flex* o ANSI C que reconozca los componentes léxicos pertenecientes a las clases abajo descritas.

Clase	Descripción	Expresión Regular
0	Operadores de Asignación	**({oparit})"="|({opespe})"="|"="**
1	Operadores de Relación	**">"|"<"|">="|"<="|"=="|"!="**
2	Identificadores	**{let}+({dig}|{let})***
3	Palabras reservadas	**DOBLE|ENTERO|HAZ|MIENTRAS|PARA|REAL|SI|SINO**
4	Símbolos especiales ; , ( ) [ ]	**";"|","|"("|")"|"["|"]"**
5	Cadenas entre comillas	**\"(\\.|[^\\"])*\"**
6	Operadores aritméticos + - * / %	**"+"|"-"|"*"|"/"|"%"{1,1}**

7	Constantes enteras (Octal, decimal, hexadecimal)	Decimal: **[1-9]{dig}+** Octal**: 0[1-7]+** Hexa: **0(x|X)({dig}|[a-f]|[A-F])+**
8	Constantes reales (Con punto decimal)	**0|{dig}+\.{dig}***

Para la solución del problema propuesto, se utilizará la mejor herramienta para su resolución que es el lenguaje de programación flex/lex.

Fases del desarrollo del sistema:

Instalando ambiente de desarrollo: Antes de comenzar con cualquier fase de diseño lo primero que se realizo fue encontrar un ambiente óptimo de trabajo tomando en consideración que se había tomado previamente la utilización del lenguaje de programación Flex/Lex.

Como se está Trabajando en un sistema Unix con S.O Mac OSX, el compilador de Flex ya viene precargado sin embargo al momento de ejecutar necesita un par de comandos para su correcta compilación.

Como editor de Texto se utilizó el editor TextMate debido a que reconoce la gramática del lenguaje de programación.

Planificación:

Durante la planificación del proyecto se plantea un avance diario durante 7 días, buscando que el avance del compilador se vaya generando al realizar el reconocimiento de una clase por día.

Diseño:

Para el diseño se toma en cuenta el hecho de que se necesita realizar la creación

de tablas, para la tabla de Símbolos se almacenarán 3 campos posición, nombres del identificador y tipo, dicha tabla se crea al momento de ejecutar el código C.

La tabla de símbolos almacena identificadores que va a encontrado, cada identificador que encuentra lo compara con todos los anteriores, si existe un anterior regresa el valor del primer identificador que apareció con ese nombre.

Para la tabla de cadenas, va identificado cada una de las cadenas que aparecen en el texto ya las guarda tal cual aparecen sin comparar con las cadenas previas.

Para todos los demás elementos se crean token que guardan los valores de clave valor, guardado en un archivo tras la ejecución del programa.

Las Técnicas de búsqueda para este algoritmo son lineales lo que significa que nos da una notación de big O, del siguiente modo: O(n), no sé usa como tal una estructura de datos sin embargo se utiliza un arreglo de estructuras que tiene un funcionamiento similar, donde el primer dato en agregarse al arreglo es el primero que sale (FIFO).

Implementación

Durante la implementación se llevó a cabo de manera práctica la fase de planeación, dentro de la implementación surgieron algunos problemas con el identificador de cadenas sobre todo por el formato de comillas que ponen ciertos sistemas operativos, otra complicación fue el requerimiento del S.O Mac de linkear algunas bibliotecas externas para que pueda funcionar el código de Flex.

Indicaciones de cómo correr el programa

Dentro del sistema operativo Mac OS la instrucciones son las siguientes:

- Acceder al directorio donde está guardado el archivo desde la terminal y ejecutar los siguiente comandos:

1. lex AnalizadorLexico.l
2. gcc lex.yy.c **-ll**
3. ./a.out <archivo con texto>

Después de hacer esto se crearan 4 archivos cada uno con un nombre específico de cuál es la función y cuáles son los datos que tiene almacenado ese archivo.

Para los sistemas operativos Linux seguir los siguientes pasos:

- Acceder al directorio donde está guardado el archivo desde la terminal y ejecutar los siguiente comandos:

1. lex AnalizadorLexico.l
2. gcc lex.yy.c
3. ./a.out <archivo con texto>

El resultado tras la ejecución será el mismo.

Conclusiones

El desarrollo del analizador léxico fue muy enriquecedor para la clase, nos deja ver el funcionamiento y como es el proceso interno de una computadora para poder desarrollas un compilador.