UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SEMESTRE 2013-2

COMPILADORES

GUZMÁN VILLANUEVA JULIO CÉSAR

PROGRAMA 1 : Analizador Léxico

FECHA DE ENTREGA: Martes 5 de marzo, 2013

Descripción del problema

Se desea hacer un analizador léxico de un preprocesador del lenguaje C para crear una representación intermedia entre el analizador léxico y el analizador sintáctico. El problema consiste en identificar los componentes que tienen un significado colectivo, es decir, a cada componente se le identificará con una clase. El análisis léxico partirá de un archivo fuente el cual será analizado léxicamente. La salida del programa serán los tokens (clase, valor), tablas y errores del programa fuente.

Para el análisis léxico se contemplan las siguientes clases:

Clase	Descripción
0	Identificadores
1	Constantes enteras decimales
2	Cadenas (delimitadas con @)
3	Palabras reservadas (iniciadas con #)
4	Simbolos especiales
5	Constantes enteras octales
7	Constantes enteras hexadecimales
8	Operadores relacionales

La expresión regular de los componentes léxicos es la siguiente:

Identificadores

Expresión regular de los identificadores en la notación de flex:

letra_minuscula	[a-z]
letra_mayuscula	[A-Z]
letra	{letra_minuscula}|{letra_mayuscula}
guion_bajo	[_]
digito	[0-9]
identificador	({letra}|{guion_bajo})({letra}|{digito}|{guion_bajo})*

Constantes enteras decimales

Expresión regular de las constantes enteras decimales en la notación de flex:

```
cero                        [0]
digito_uno_al_nueve         [1-9]
constante_entera_decimal    {cero}|({digito_uno_al_nueve}({digito_uno_al_nueve}|{cero})*)
```

Cadenas delimitadas con @

Expresión regular de las cadenas delimitadas con @ en la notación de flex:

```
arroba                          [@]
no_es_arroba_ni_salto_de_linea  [^@]|\\n
cadena                          {arroba}{no_es_arroba_ni_salto_de_linea}*{arroba}
```

Palabras reservadas (iniciadas con #)

Expresión regular de las palabras reservada en la notación de flex:

```
palabra_reservada   [#]("define"|"ifdef"|"if"|"elif"|"else"|"endif"|"undef"|"indef")
```

Simbolos especiales

Expresión regular de los simbolos especiales en la notación de flex:

```
simbolo_especial    [;,()!]
```

Constantes enteras octales

Expresión regular de las constantes enteras octales en la notación de flex:

```
cero                    [0]
digito_uno_al_siete     [1-7]
constante_entera_octal  {cero}({digito_uno_al_siete}|{cero})+
```

Constantes enteras hexadecimales

Expresión regular de las constantes enteras hexadecimales en la notación de flex:

```
cero                        [0]
digito_uno_al_nueve         [1-9]
constante_entera_hexadecimal    {cero}([x]|[X])({digito_uno_al_nueve}|[a-f]|[A-F]|{cero})+
```

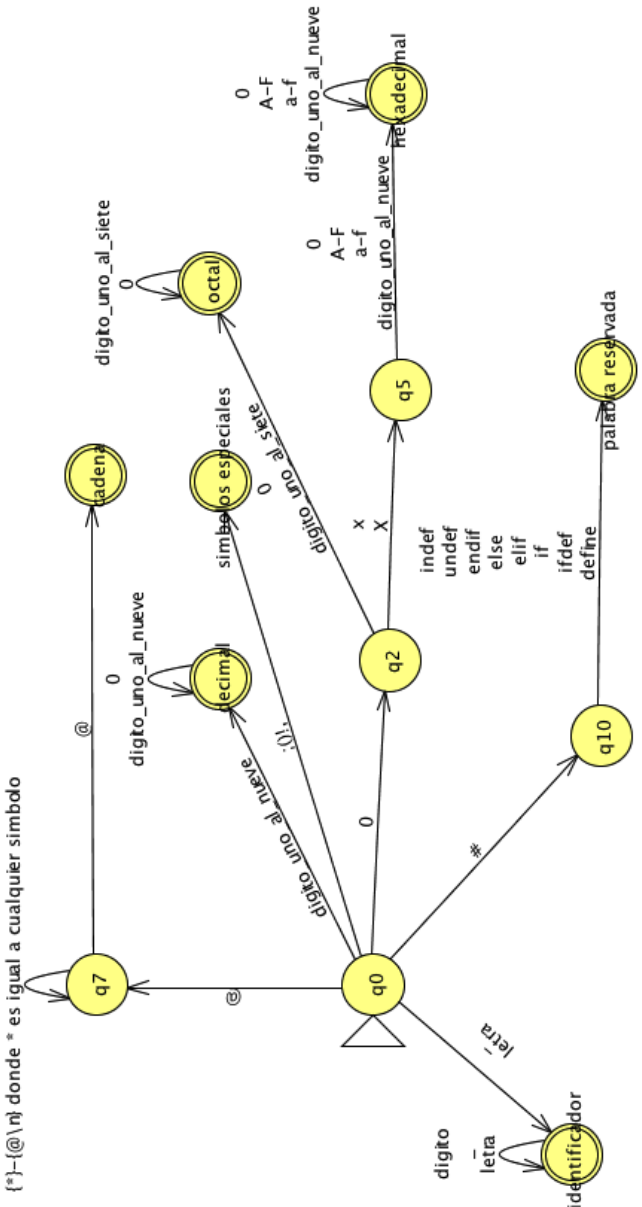
Operadores relacionales

Expresión regular de los operadores relacionales en la notación de flex:

```
operador_relacional     \<|"<="|\>|">="|"=="|"!="
```

Automata finito deterministico del analizador

Para reconocer a las cadenas que pertenecen al lenguaje en necesario hacer un automata finito deterministico que reconozca a las cadenas que pertenezcan al lenguaje. Este autómata deberá ser capaz de discernir entre clases para asignarle a la cadena analizada una clase.



Propuesta de solución

Análisis.

Ya que un analizador léxico comparte las tablas con el analizador sintáctico, el analizador semántico, el generador de código y el optimizador de código; decidí que la estructura de datos en la cual se almacenarán las tablas serán archivos.

Diseño e Implementación.

Usaré el analizador léxico flex para facilitar el desarrollo del programa debido a la facilidad con la que se pueden identificar a los componentes léxicos además de que ya cuento con las expresiones regulares en el formato de flex. Después crearé funciones para manejar los archivos de las tablas y crearé funciones para insertar y buscar dentro de las tablas.

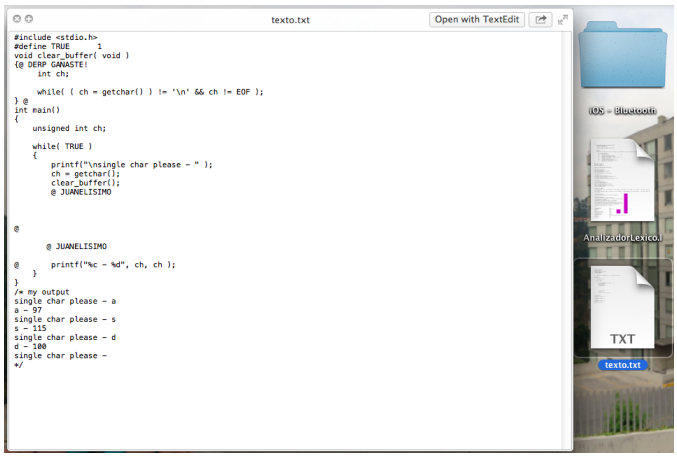
- La función que inserta no regresará nada.
- Mientras que la función que busca implementará una búsqueda secuencial y regresará el carácter de la posición en la que se encuentra el valor de la tabla.

Así mismo usaré variables globales de los índices para llevar un control durante el proceso de inserción.

- La tabla de símbolos como ya fue explicado anteriormente existirá en un archivo.
- Los tokens se irán agregando al programa conforme vaya leyendo nuevos componentes léxicos para que sólo lo haga en una pasada.

Indicaciones de como correr el programa.

- Crear archivo entrada:



- Abrir la terminal, ubicarse en la carpeta en la que se encuentra el analizador léxico y

escribir los siguientes comandos:

```
julios-macbook:~ Julio$ cd desktop
julios-macbook:desktop Julio$ lex AnalizadorLexico.l
julios-macbook:desktop Julio$ gcc lex.yy.c -ll
julios-macbook:desktop Julio$ ./a.out texto.txt
```

Nota: La bandera -ll funciona en MACOS. Para compilar el archivo en distribuciones de linux funciona la bandera -lfl

- La salida en la consola debe ser la siguiente aunada a los archivos generados.