UTN – FACULTAD REGIONAL BUENOS AIRES
Sintaxis y semántica de los lenguajes
CURSO K2054

TRABAJO PRÁCTICO NRO. 1

GRUPO

Manrique, Karen Ailen 16.3237.1 kmanrique.utn@gmail.com

FECHA DE PRESENTACIÓN: 28/6/2018 CALIFICACIÓN: _____

FIRMA PROFESOR _____

TRABAJO PRACTICO NRO. 1:

"Analizador lexico"

- Descripcion del programa:

El analizador lexico, recibe por parametro dos rutas, una que permite la lectura del archivo, y otra (la cual es opcional) que permite la escritura de la tabla con los tokens reconocidos en un archivo .txt. Si no se ingresa la segunda ruta, entonces la tabla se mostrara por pantalla.

El programa lee letra por letra, y a partir de la letra leida da lugar al almacenamiento de la palabra en un vector para luego ser escaneado por el automata correspondiente. Se presentan cinco automatas que corresponden a:

1. Comentarios
2. Cadenas de caracteres
3. Identificadores
4. Digitos
5. Operadores

A la hora de almacenamiento se analiza las siguientes situaciones para su almacenamiento:

1. Si el carácter leido es una '/' y el siguiente carácter al mismo, es otra '/', entonces se trata de un comentario, por ende, se guarda carácter por carácter hasta reconocer un salto de linea y se envia al automata de los comentarios. Tambien, los comentarios pueden comentar con '/*', por ende, se sigue el mismo procedimiento.
2. Si se trata de un operador aritmetico, a diferencia del caso anterior, se procede al almacenamiento del mismo.
3. Si el carácter leido es una letra o guion bajo, entonces se estaria hablando de identificadores. Se va leyendo carácter por carácter y almacenando el mismo en un vector hasta que se reconozca un salto de linea, un punto y coma, un igual o un parentesis (aquellos signos de puntuacion, son los que

definen si se trata de identificadores de subprogramas, palabras reservadas o identificadores de variables.

4. Si el carácter leido se trata de un digito, se almacena el mismo. Si el siguiente se trata de otro digito, entonces se almacenara el mismo, y asi sucesivamente hasta leer un carácter distinto a un digito.
5. Si se tratase de comillas, entonces, como sucede en el caso de los comentarios con una unica diferencia, se lee carácter por carácter hasta reconocer las comillas de cierre.
6. A la hora de reconocer un signo de puntuacion o el carácter '=' (asignacion), no se procede al almacenamiento del mismo.

Al mismo tiempo en el que se almacenan los vectores, se van clasificando para que luego, a la hora de analizarlos a traves de nuestro automata (aquellos que deban ser reconocidos por uno), dependiendo de las situaciones dadas.

Una vez que se reconoce como palabra que pertenece al lenguaje, entonces, se pasa a mostrar por pantalla el token que le corresponde.

Cuando se reconocen identificadores, a diferencia del resto, se verifica si es una palabra reservada, o si el siguiente carácter es un '(', un '=' o un ';'. Si se tratra de un parentesis, entonces se hablaria de un identificador de un subprograma, si se tratase de un punto y coma o un igual, se trata de un identificador.

Una vez que se finaliza la lectura del archivo, se da por finalizado el analisis lexico.

- Hipotesis

1. El archivo de escritura, debe ser un archivo de texto, es decir, no puede ser un archivo cuyo formato sea, por ejemplo, un archivo de Excel (.xls).
2. A la hora de reconocer comentarios los cuales comienzan con '/*', el mismo, no almacenra aquellos que tengan saltos de linea, es decir, si se tratase de un bloque de codigo, no se reconocera como comentario.

- Lexamas propuestos

Palabra reconocida	Lexema
printf","return","if","for","while","do","int","char","double","switch"	Palabra reservada
[0-9]	Digito
“[a-Z]”	Cadena de caracteres
‘;’ , ':', '.', '{', '}', '(', ')', '[', ']', '^', ’ ’	Signos de puntuación
'*' , '+' , '/' ,'-' , '<' , '>' , '!' , '|', '&'	Operadores
Numero (si a la palabra le sigue un ‘=’ o un ‘;’)	Identificadores
Mayor (si a la palabra le sigue un ‘(‘)	Identificadores de subprogramas
=	Asignación

- Casos de prueba

✓ Primer caso

- Entrada

```
void mayor (int nroUno, int nroDos){
  if (nroUno > nroUno)
    printf("%d es mayor a %d \n", nroUno, nroDos);
  else
    printf("%d es mayor a %d \n", nroDos, nroUno);

  return;
}
```

- Salida

```
LINEA NRO.: 1.     LEXEMA:  void .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 1.     LEXEMA:  mayor .   TOKEN: IDENTIFICADOR DE SUBPROGRAMA
LINEA NRO.: 1.     LEXEMA:  ( .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 1.     LEXEMA:  int .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 1.     LEXEMA:  nroUno .    TOKEN: IDENTIFICADOR
LINEA NRO.: 1.     LEXEMA:  , .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 1.     LEXEMA:  int .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 1.     LEXEMA:  nroDos .    TOKEN: IDENTIFICADOR
LINEA NRO.: 1.     LEXEMA:  ) .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 1.     LEXEMA:  { .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 2.     LEXEMA:  if .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 2.     LEXEMA:  ( .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 2.     LEXEMA:  nroUno .    TOKEN: IDENTIFICADOR
LINEA NRO.: 2.     LEXEMA:  > .    TOKEN: OPERADOR
LINEA NRO.: 2.     LEXEMA:  nroUno .    TOKEN: IDENTIFICADOR
LINEA NRO.: 2.     LEXEMA:  ) .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 3.     LEXEMA:  printf .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 3.     LEXEMA:  ( .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 3.     LEXEMA:  "%d es mayor a %d \n" .      TOKEN: CADENA DE CARACTERES
LINEA NRO.: 3.     LEXEMA:  , .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 3.     LEXEMA:  nroUno .    TOKEN: IDENTIFICADOR
LINEA NRO.: 3.     LEXEMA:  , .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 3.     LEXEMA:  nroUno .    TOKEN: IDENTIFICADOR
LINEA NRO.: 3.     LEXEMA:  , .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 3.     LEXEMA:  nroDos .    TOKEN: IDENTIFICADOR
LINEA NRO.: 3.     LEXEMA:  ) .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 3.     LEXEMA:  ; .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 4.     LEXEMA:  else .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 5.     LEXEMA:  printf .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 5.     LEXEMA:  ( .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 5.     LEXEMA:  "%d es mayor a %d \n" .      TOKEN: CADENA DE CARACTERES
LINEA NRO.: 5.     LEXEMA:  , .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 5.     LEXEMA:  nroDos .    TOKEN: IDENTIFICADOR
LINEA NRO.: 5.     LEXEMA:  , .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 5.     LEXEMA:  nroUno .    TOKEN: IDENTIFICADOR
LINEA NRO.: 5.     LEXEMA:  ) .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 5.     LEXEMA:  ; .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 7.     LEXEMA:  return .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 7.     LEXEMA:  ; .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 8.     LEXEMA:  } .    TOKEN:    CARACTER DE PUNTUACION
```

- ✓ Segundo caso
  - Entrada

```c
int mi-nombre(){
  int 9x = 0;
  printf("%d \n", 9x);

  return 0;
}
```

  - Salida

```
LINEA NRO.: 1.     LEXEMA:  int .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 1.     LEXEMA:  mi-nombre .      ERROR LEXICO
LINEA NRO.: 1.     LEXEMA:  ( .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 1.     LEXEMA:  ) .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 1.     LEXEMA:  { .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 2.     LEXEMA:  int .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 2.     LEXEMA:  9x .      ERROR LEXICO
LINEA NRO.: 2.     LEXEMA:  = .    TOKEN: ASIGNACION
LINEA NRO.: 2.     LEXEMA:  0 .    TOKEN: DIGITO
LINEA NRO.: 2.     LEXEMA:  ; .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 3.     LEXEMA:  printf .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 3.     LEXEMA:  ( .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 3.     LEXEMA:  "%d \n" .      TOKEN: CADENA DE CARACTERES
LINEA NRO.: 3.     LEXEMA:  , .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 3.     LEXEMA:  9x .      ERROR LEXICO
LINEA NRO.: 3.     LEXEMA:  ) .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 3.     LEXEMA:  ; .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 5.     LEXEMA:  return .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 5.     LEXEMA:  0 .    TOKEN: DIGITO
LINEA NRO.: 5.     LEXEMA:  ; .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 6.     LEXEMA:  } .    TOKEN:    CARACTER DE PUNTUACION
```

- ✓ Tercer caso
  - Entrada

```
int main (){

  if (funcionaPrograma(programa))
    printf("Hola\n");

    // Funciona

  return 0;

}
```

  - Salida

```
LINEA NRO.: 1.     LEXEMA:  int .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 1.     LEXEMA:  main .   TOKEN: IDENTIFICADOR DE SUBPROGRAMA
LINEA NRO.: 1.     LEXEMA:  ( .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 1.     LEXEMA:  ) .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 1.     LEXEMA:  { .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 3.     LEXEMA:  if .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 3.     LEXEMA:  ( .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 3.     LEXEMA:  funcionaPrograma .   TOKEN: IDENTIFICADOR DE SUBPROGRAMA
LINEA NRO.: 3.     LEXEMA:  ( .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 3.     LEXEMA:  programa .    TOKEN: IDENTIFICADOR
LINEA NRO.: 3.     LEXEMA:  ) .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 3.     LEXEMA:  ) .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 4.     LEXEMA:  printf .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 4.     LEXEMA:  ( .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 4.     LEXEMA:  "Hola\n" .      TOKEN: CADENA DE CARACTERES
LINEA NRO.: 4.     LEXEMA:  ) .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 4.     LEXEMA:  ; .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 6.     LEXEMA:  //Funciona .    TOKEN: COMENTARIO
LINEA NRO.: 8.     LEXEMA:  return .    TOKEN: PALABRA RESERVADA
LINEA NRO.: 8.     LEXEMA:  0 .    TOKEN: DIGITO
LINEA NRO.: 8.     LEXEMA:  ; .    TOKEN:    CARACTER DE PUNTUACION
LINEA NRO.: 10.     LEXEMA:  } .    TOKEN:    CARACTER DE PUNTUACION
```