Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Manual Técnico Proyecto “2”

Santiago Gilberto Antonio

Rivadeneira Ruano

201313722

Objetivos

Generales:

- El estudiante aplique los conceptos sobre la fase de análisis léxico de un compilador en una solución de software.

Específicos:

- El estudiante refuerce los conocimientos del método del árbol, expresiones regulares y autómatas finitos deterministas.
- Identificar y programar el proceso de reconocimiento de lexemas mediante el uso de autómatas finitos deterministas.

Introducción:

Este manual técnico se crea con la única finalización de poder ayudar a describir cómo fue que se realizó la aplicación, mencionando el tipo de equipo utilizado para realizarla hasta mostrar todo el análisis léxico y sintáctico utilizado en la elaboración de dicha práctica.

Requisitos del Sistema:

Esta aplicación puede utilizarse con los siguientes requerimientos:

1. Sistema operativo: Windows 10
2. Espacio En Disco Disponible: 1Gb
3. Memoria Ram: 8Gb.
4. Navegador Chrome, Microsoft Edge
5. Procesador Core i5 de 8va generación

Análisis Léxico

Expresiones regulares utilizadas:

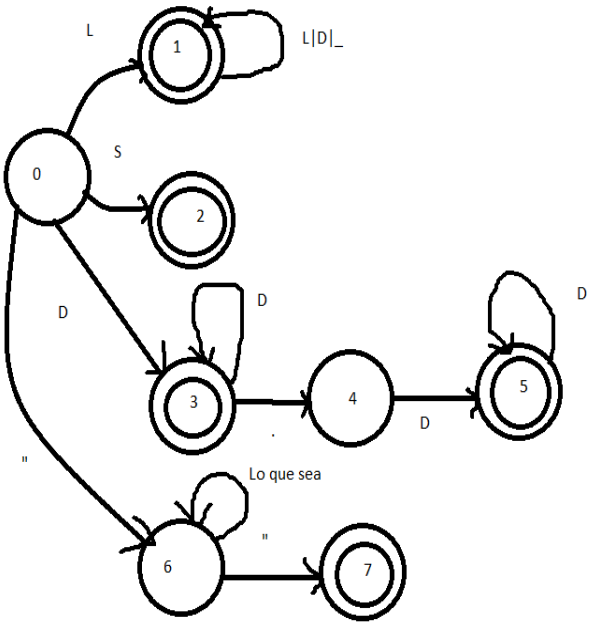
Er1 -> L(L|D|_)*
Er2 -> ;
Er3-> =
Er4-> ,
Er5-> +
Er6-> -
Er7-> *
Er8-> /
Er9-> && (And)
Er10-> || (or)
Er11-> ! (not)
Er12-> >(mayor)
Er13-> < (menor)
Er14-> >= (mayor o igual)
Er15-> <= (menor o igual)
Er16-> == (igual)
Er17-> != (distinto)
Er18-> (
Er19->)
Er20-> {
Er21-> }
Er22-> .
Er23-> "
Er24-> :
Er25-> D+(.D+)?

Palabras reservadas utilizadas

int
double
char
bool
string
void
main
if
switch
while

for
do
Console
Write
return
break
continue

Este es el autómata generado para poder realizar el análisis léxico de el archivo de entrada.



Análisis Sintáctico

Se utilizo la gramática siguiente:

```
/*************** GRAMATICA ***************/


%%
INICIO
    : CONTENIDO EOF
;
CONTENIDO
    : CONTENIDO IMPORT
    | IMPORT
    | CONTENIDO CLASES
    | CLASES
;

IMPORT
    : R_Import LISTADO_IMPORT S_PuntoComa
;

LISTADO_IMPORT
    : LISTADO_IMPORT S_Punto DEFINE_IMPORT
    | DEFINE_IMPORT
;
```

```
DEFINE_IMPORT
   : Identificador
;
CLASES
   : R_Class Identificador S_LlaveAbre  LISTA_CLASES S_LlaveCierra
;
LISTA_CLASES
   : LISTA_CLASES CONTENIDO_CLASE
   | CONTENIDO_CLASE
;
CONTENIDO_CLASE
   : TIPO_DATO Identificador S_ParentesisAbre PARAMETROS S_ParentesisCierra
S_LlaveAbre INSTRUCCIONES S_LlaveCierra
   | VARIABLE
   | R_Void METODO_VOID
   | LLAMAR_METODOF_CLASE


;
METODO_VOID
   : R_Main S_ParentesisAbre S_ParentesisCierra S_LlaveAbre INSTRUCCIONES
S_LlaveCierra
   | Identificador S_ParentesisAbre PARAMETROS S_ParentesisCierra S_LlaveAbre
INSTRUCCIONES S_LlaveCierra
;
VARIABLE
   : TIPO_DATO LISTADO_ID_VARIABLE S_PuntoComa
;
```

```
LISTADO_ID_VARIABLE
   : LISTADO_ID_VARIABLE S_Coma CONTENIDO_VARIABLE
   | CONTENIDO_VARIABLE
;
CONTENIDO_VARIABLE
   //aqui tengo que agregar la asignacion de variables
   :Identificador S_Igual EXPRESION_G
   |Identificador
;
TIPO_DATO
   : T_Int
   | T_String
   | T_Boolean
   | T_Char
   | T_Double
;
MODIFICADORES_ACCESO
   : R_Protected
   | R_Public
   | R_Private
   |
;
EXPRESION_G
   : EXPRESION_G LOG_Concatenar EXPRESION_G
   | EXPRESION_G LOG_OR EXPRESION_G
   | EXPRESION_G REL_IgualIgual EXPRESION_G
   | EXPRESION_G REL_MayorIgualQue EXPRESION_G
```

```
    | EXPRESION_G REL_MayorQue EXPRESION_G
    | EXPRESION_G REL_MenorIgualQue EXPRESION_G
    | EXPRESION_G REL_MenorQue EXPRESION_G
    | EXPRESION_G REL_Distinto EXPRESION_G
    | EXPRESION_G OP_Mas EXPRESION_G
    | EXPRESION_G OP_Menos EXPRESION_G
    | EXPRESION_G OP_Multiplicacion EXPRESION_G
    | EXPRESION_G OP_Division EXPRESION_G
    | EXPRESION_G OP_Potencia EXPRESION_G
    | EXPRESION_G OP_Modulo EXPRESION_G
    | CONTENIDO_EXPRESION OP_Decremento %prec PRUEBA
    | CONTENIDO_EXPRESION OP_Incremento %prec PRUEBA
    | OP_Menos  CONTENIDO_EXPRESION     %prec UMINUS
    | LOG_Not   CONTENIDO_EXPRESION     %prec UMINUS
    | CONTENIDO_EXPRESION
;
 CONTENIDO_EXPRESION
   : Entero
    | Decimal
    | Identificador S_ParentesisAbre S_ParentesisCierra
    | Identificador S_ParentesisAbre OPCIONAL S_ParentesisCierra
    | R_True
    | R_False
    | S_ParentesisAbre EXPRESION_G S_ParentesisCierra
    | Identificador
    | Cadena
    | Char
```

```
    | CHAR_Especial
;
OPCIONAL
    : EXPRESION_G
    | OPCIONAL S_Coma EXPRESION_G
;
FUNC
    :EXPRESION_G
    |
;
PARAMETROS
    : DEFINIR_PARAMETRO LISTA_PARAMETROS
    | DEFINIR_PARAMETRO
    |
;
LISTA_PARAMETROS
    : LISTA_PARAMETROS S_Coma DEFINIR_PARAMETRO
    | S_Coma DEFINIR_PARAMETRO
;
DEFINIR_PARAMETRO
    : TIPO_DATO Identificador
;
METODOS_LL
    : Identificador S_Igual EXPRESION_G S_PuntoComa
    | Identificador S_ParentesisAbre PARAMETROS_FUNC S_ParentesisCierra S_PuntoComa
;
```

```
PARAMETROS_FUNC
    : PARAMETROS_FUNC S_Coma EXPRESION_G
    | EXPRESION_G
    |
;
LLAMAR_METODOF_CLASE
    : Identificador S_ParentesisAbre PARAMETROS_FUNC S_ParentesisCierra S_PuntoComa
;
INSTRUCCIONES
    : LISTA_INS
    |
;
LISTA_INS
    : LISTA_INS LISTA_INSTRUCCIONES
    | LISTA_INSTRUCCIONES
;
LISTA_INSTRUCCIONES
    : METODOS_LL
    | VARIABLE
    | IMPRIMIR
    | SENT_IF
    | LOOP_WHILE
    | LOOP_DO_WHILE
    | LOOP_FOR
    | SENT_SWITCH
    | S_TRANSFERENCIA
    | error S_PuntoComa
```

;

IMPRIMIR

: R_System S_Punto R_Out S_Punto TIPO_IMPRESION S_ParentesisAbre EXPRESION_G S_ParentesisCierra S_PuntoComa

;

TIPO_IMPRESION

: R_Print

| R_Println

;