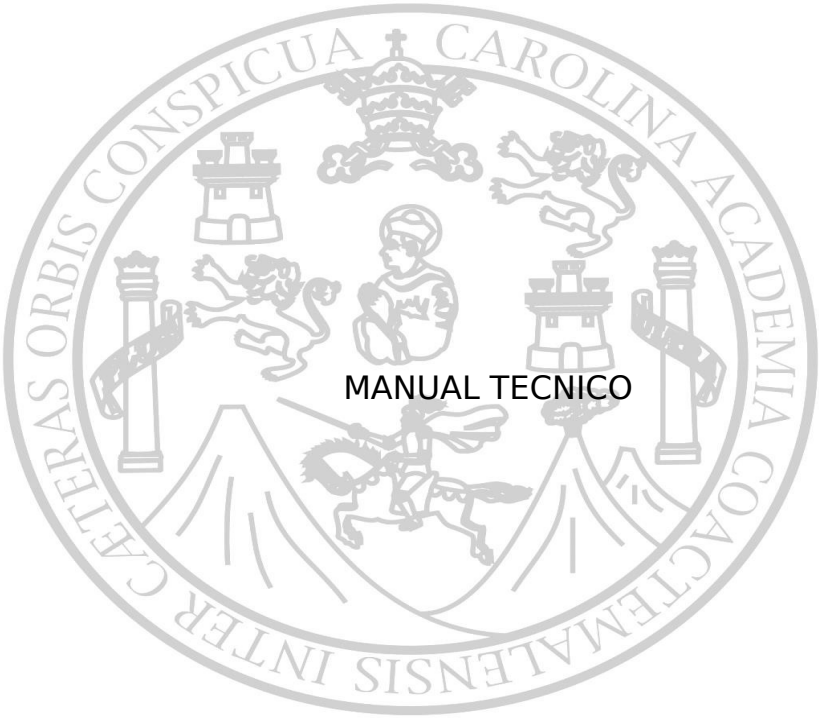
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA CIENCIAS Y SISTEMAS
ORGANIZACIÓN LENGUAJES COMPILADORES 1
SECCIÓN: A
ING BAUTISTA



MANUAL TECNICO

Nombre: Edi Yovani Tomás Reynoso
Carne: 2015-03783

Primer Analizador Lexico

Declaraciones

```
package FaseCompilador;
import java_cup.runtime.Symbol;
import java.util.ArrayList;

%%

//--------------------------------Codigo Java-----------------------------------------------------

%{
    // codigo java
    String texto = "";
    public PintarPalabras pintar = new PintarPalabras();
    public static ArrayList<ErrorLexico> listaError = new ArrayList();

%}


//------>   Directivas
%cupsym sym
%class Lexico
%state    COMENTARIO   COMEMTARIOMULTILINEAL STRING
%cup
%public
%line
%column
%char
%full
%8bit
```

```
%unicode
%ignorecase



//--------------------Expresiones regulares Numeros-------------------------------------
      letra = [a-zA-ZñÑ]+
      entero = [0-9]+
      decimal =[0-9]+ "." [0-9]*
      ID = {letra}({letra} |"_"| {entero})*
      space = [\t|\f|" "|\r|\n]
      enter = [\ \n]

%%

<YYINITIAL> {


//-------------------------------------------------------------------------------------------------
    "Archivo"                 {
                                 pintar.pintarAzul(yychar,yylength());
                                 return new Symbol(sym.archivo ,yyline, yycolumn,yytext());
                              }
//-------------------------------------------------------------------------------------------------

    "leerArchivo"             {
                                 pintar.pintarAzul(yychar,yylength());


                                 return new Symbol(sym.leerarchivo ,yyline, yycolumn,yytext());
                              }
//-------------------------------------------------------------------------------------------------


    "Numerico"                {
                                 pintar.pintarAzul(yychar,yylength());
                                 return new Symbol(sym.numero ,yyline, yycolumn,yytext());
                              }
//-------------------------------------------------------------------------------------------------
    "Sumar"                   {
                                 pintar.pintarAzul(yychar,yylength());
                                 return new Symbol(sym.suma ,yyline, yycolumn,yytext());
                              }
//-------------------------------------------------------------------------------------------------
    "Contar"                  {
                                 pintar.pintarAzul(yychar,yylength());
                                 return new Symbol(sym.contar ,yyline, yycolumn,yytext());
                              }

//-------------------------------------------------------------------------------------------------
    "Promedio"                {
                                 pintar.pintarAzul(yychar,yylength());
                                 return new Symbol(sym.promedio ,yyline, yycolumn,yytext());
                              }

//-------------------------------------------------------------------------------------------------
```

```
//-------------------------------------INICIALIZAN LOS ESTADOS-----------------------------------------------

 "//"  {yybegin(COMENTARIO);}
\"  {  yybegin(STRING);
     pintar.pintarVerde(yychar,yylength());
    }
"/*" {yybegin(COMEMTARIOMULTILINEAL);}


//-------------------------------------SE IGNORAN LOS ESPACIOS-----------------------------------------------

 {space} {/*Se ignora*/}
 {enter} {/*Se ignora*/}

 //-----------------------------------CUALQUIER OTRO ERROR-------------------------------------------------------
/* Cualquier otro error siempre se separa el punto con la llave para que no de error de compilacion. */

.

{
   ErrorLexico e = new ErrorLexico("Error lexico",yytext(),yyline, yycolumn,"El caracter no pertenece al lenguaje");
    listaError.add(e);
    System.err.println("Error lexicografico: "+yytext()+ " Linea:"+(yyline+1)+" Columna:"+(yycolumn+1));
}


}// fin del YYINITIAL
```

```
//--------------------------------ESTADOS-----------------------------------------------
<COMENTARIO>{
[\n]  {  yybegin(YYINITIAL);}
[^\n] {  /*Se ignora */}
}

<COMEMTARIOMULTILINEAL>
{
    "*"+"/"      {yybegin(YYINITIAL);}
    [^*\n]*      { /*Se ignora*/ }
    "*"+[^*/\n]* {/*Se ignora*/ }
    [\n]         {/*Se ignora*/ }
}


<STRING>{
 [\"]  {
            pintar.pintarVerde(yychar,yylength());
            String temporal=texto;
            texto ="";
            yybegin(YYINITIAL);
              return new Symbol(sym.texto,yyline, yycolumn,temporal);          }
 [^\"]  {
            texto+=yytext();
            pintar.pintarVerde(yychar,yylength());
```

Analisis Sintactico Gramtica

INICIO ::= ARCHIVOREPORTES ;

ARCHIVOREPORTES ::= ARCHIVOREPORTES DECLARACIONVARIABLES

| DECLARACIONVARIABLES:a
| ARCHIVOREPORTES IMPRIMIR
| IMPRIMIR
| ARCHIVOREPORTES GRAFICAR
| GRAFICAR

;

DECLARACIONVARIABLES ::= TIPO id eqa FUNCION puntoycoma
;

TIPO ::= archivo
| numero
| cadena

;

FUNCION ::= leerarchivo pa_open texto:val pa_close
| suma pa_open id:id coma texto:texto pa_close
| contar pa_open id:id pa_close
| promedio pa_open id coma texto pa_close
| contarsi pa_open id coma texto coma OPERADORRACIONAL coma VALOR pa_close

| obtenersi pa_open id coma texto coma OPERADORRACIONAL coma VALOR pa_close

;

OPERADORRACIONAL ::= menor
|mayor
|equals
|diferente
|menorigual
|mayorigual
;

GRAFICAR ::= graficar pa_open texto:a coma texto:b coma id:id coma texto:c coma texto:d pa_close puntoycoma

;

IMPRIMIR ::= imprimir pa_open LISTAEXPRESIONES pa_close puntoycoma

;

LISTAEXPRESIONES ::= LISTAEXPRESIONES coma VALOR
| VALOR

;
VALOR ::= texto:val
|SIGNO:a entero:val
|SIGNO:a decimal:val
|entero
|decimal
| id

;
SIGNO ::= menos

| mas

;

Segundo Analizador

Analizador Lexico

Declaraciones

```
package FaseCompilador2;
import FaseCompilador.ErrorLexico;
import static FaseCompilador.Lexico.listaError;
import java_cup.runtime.Symbol;
import java.util.ArrayList;

%%
//--------------------------------Codigo Java--------------------
%{
    // codigo java
    String txt = "";
%}
//------>   Directivas
%cupsym  sym
%class scanner
%state    COMENTARIO   COMEMTARIOMULTILINEAL STRING
%cup
%public
%line
%column
%char
%full
%8bit
%unicode
%ignorecase


//--------------------Expresiones regulares Numeros-------------
      // letra = [a-zA-ZñÑ]+
       entero = [0-9]+
       decimal =[0-9]+ "." [0-9]*
      // ID = {letra}({letra} |"_"| {entero})*
       space = [\t|\f|" "|\r|\n]
       enter = [\ \n]

%%
```

```
<YYINITIAL> {
//-----------------------------PALABRAS RESERVADAS-----------------------------------------------------


    "Claves"                {
                              return new Symbol(sym.clave ,yyline, yycolumn,yytext());
                            }
//-------------------------------------------------------------------------------------------------
    "Registros"             {
                              return new Symbol(sym.registro ,yyline, yycolumn,yytext());
                            }

//-------------------------------------------------------------------------------------------------
    "{"                     {
                              return new Symbol(sym.llaveabierta ,yyline, yycolumn,yytext());
                            }
//-------------------------------------------------------------------------------------------------
    "}"                     {
                              return new Symbol(sym.llavecerrada ,yyline, yycolumn,yytext());
                            }
//-------------------------------------------------------------------------------------------------
    "["                     {
                              return new Symbol(sym.CorcheteAbierto ,yyline, yycolumn,yytext());
                            }
//-------------------------------------------------------------------------------------------------
    "]"                     {
                              return new Symbol(sym.CorcheteCerrado ,yyline, yycolumn,yytext());
                            }
```

```
                                }
// ------------ RETORNAR EXPRESIONES REGULARES-----------------------------------


//-------------------------------------------------------------------------------------------------
{entero}   {
                return new Symbol(sym.entero ,yyline, yycolumn,yytext());
           }

//-------------------------------------------------------------------------------------------------
 {decimal} {
                   return new Symbol(sym.decimal ,yyline, yycolumn,yytext());
           }
//---------------------------------------INICIALIZAN LOS ESTADOS--------------------


 "//"  {yybegin(COMENTARIO);}
\"  {yybegin(STRING);}
"/*" {yybegin(COMEMTARIOMULTILINEAL);}


//-------------------------------------SE IGNORAN LOS ESPACIOS--------------------

 {space} {/*Se ignora*/}
 {enter} {/*Se ignora*/}
```

```
//---------------------------------CUALQUIER OTRO ERROR-------------------------------------------------------
/* Cualquier otro error siempre se separa el punto con la llave para que no de error de compilacion. */

.

{
    ErrorLexico e = new ErrorLexico("Error lexico",yytext(),yyline, yycolumn,"El caracter no pertenece al lenguaje");
     listaError.add(e);
     System.err.println("Error lexico: "+yytext()+ " Linea:"+(yyline+1)+" Columna:"+(yycolumn+1));
}


} // fin del YYINITIAL
```

```
//---------------------------------ESTADOS-------------------------------------
<COMENTARIO>{
[\n]   {  yybegin(YYINITIAL);}
[^\n] {  /*Se ignora */}
}

<COMEMTARIOMULTILINEAL>
{
    "*"+"/"      {yybegin(YYINITIAL);}
    [^*\n]*      { /*Se ignora*/ }
    "*"+[^*/\n]* {/*Se ignora*/ }
    [\n]         {/*Se ignora*/ }
}


<STRING>{
 [\"]  {
            String temporal=txt;
            txt ="";
            yybegin(YYINITIAL);
              return new Symbol(sym.txt,yyline, yycolumn,temporal);
 [^\"]  {
            txt+=yytext();

        }
}
```

Analizador Sintactico

package FaseCompilador2;
import FaseCompilador.ErrorParser;
import FaseCompilador.Sintactico;
import FaseCompilador.ErrorSemantico;

import java_cup.runtime.Symbol;
import java.util.ArrayList;
import Datos.Registro;
import Datos.ArchivoDatos;
parser code

```
{:
  public ArchivoDatos  archivo ;

/************----------------Metodo al que se llama automáticamente ante
algún error sintactico.--------------------*/

    public void syntax_error(Symbol s)
    {
    try{
                System.err.println("Error Sintactico En la linea" + (s.left+1)
+" Columna "+(s.right+1)+ ". Identificador "
                +  s.value.toString()   + " no reconocido1." );
                int columna = s.right;
                int fila = s.left;
                String lexema = s.value.toString();
                ErrorParser datos = new ErrorParser("Error
Sintactico",lexema,fila, columna,"Se esperaba caracter");
                Sintactico.ErrorSintactico.add(datos);
          }
                catch(Exception error){}

}

/**----------------Metodo al que se llama en el momento en que ya no es
posible una recuperación de errores.---------------------*/
public void unrecovered_syntax_error(Symbol s) throws
java.lang.Exception
   {
            try{

                System.err.println("Error Sintactico En la linea" + (s.left+1)
+" Columna "+(s.right+1)+ ". Identificador "
                +  s.value.toString()   + " no reconocido1." );
                 int columna = s.right;
                 int fila = s.left;
                 String lexema = s.value.toString();
                 ErrorParser datos = new ErrorParser("Error
Sintactico",lexema,fila, columna,"Se esperaba caracter");
                Sintactico.ErrorSintactico.add(datos);
                } catch(Exception er){}

   }


:}

//--------- Se declaran los todos terminales reservados de
jflex----------------------------------------------------
```

```
terminal String entero, decimal;
terminal String txt;
terminal String coma;
terminal String mas,menos;
terminal String eqa;
terminal String
llaveabierta,llavecerrada,CorcheteAbierto,CorcheteCerrado;

//----------- palabra reservadas- jflex----------

terminal String clave;
terminal String registro;

//********************************************************** Aquí están los no
terminales ************************************

non terminal INICIO;
non terminal ArchivoDatos ARCHIVODATOS;
non terminal CLAVE,REGISTRO;
non terminal ArrayList<String> LISTASTRING;
non terminal LISTATIPO;
non terminal ArrayList<Registro> LIST ;
non terminal ArrayList<ArrayList<Registro>>  LISTADO;
non terminal SIGNO;

//------------------------------Formato
BNF----------------------------------------------------------------
start with INICIO; // start with sirve para indicarle al parser con que
produccion empezar

INICIO ::=  ARCHIVODATOS:a
{:
    parser.archivo = a;
:}
;

ARCHIVODATOS ::= CLAVE:a  REGISTRO:b
             {:
                RESULT = new ArchivoDatos((ArrayList<String>)a,
(ArrayList<ArrayList<Registro>>)b);
             :}
             ;
CLAVE ::=  clave eqa CorcheteAbierto LISTASTRING:cabecera
CorcheteCerrado
              {:
                 RESULT = cabecera;
              :}
```

```
;
LISTASTRING ::= LISTASTRING:l coma txt:a
              {:
                 RESULT = l;
                 RESULT.add(a);
              :}
             | txt:a
              {:
                  RESULT = new ArrayList();
                  RESULT.add(a);
              :}
            ;
REGISTRO ::=   registro eqa CorcheteAbierto LISTADO:r CorcheteCerrado
             {:
                RESULT =r;
             :}
           ;
LISTADO ::=   LISTADO:a llaveabierta LIST:lista llavecerrada
           {:
                 RESULT = a;
                 RESULT.add(lista);
           :}
           | llaveabierta LIST:lista llavecerrada
            {:
                 RESULT = new ArrayList();
                RESULT.add(lista);
            :}
          ;

LIST  ::=  LIST:a coma LISTATIPO:b
           {:
             RESULT = a;
             RESULT.add(new Registro(b,bright+1,bleft+1));
           :}
         | LISTATIPO:b
           {:
              RESULT = new ArrayList();
              RESULT.add(new Registro(b,bright+1,bleft+1));
           :}
;


LISTATIPO ::=   txt:val
             {:
                  RESULT = val;
             :}
           |SIGNO:a entero:val
             {:
```

```
                RESULT = Double.parseDouble(a.toString()
+""+val.toString());
             :}
           |SIGNO:a decimal:val
             {:
                RESULT = Double.parseDouble(a.toString()
+""+val.toString());
             :}
           |entero:val
             {:
                RESULT = Double.parseDouble(val.toString());
             :}
           |decimal:val
            {:
                RESULT = Double.parseDouble(val.toString());
            :}
   ;
SIGNO ::=  menos:a
           {:
             RESULT = a;
           :}
       | mas:a
         {:
           RESULT = a;
         :}
;
```