# COMPILADOR PAR LENGUAJE LOOP

## a) Las especificaciones léxicas del lenguaje

```
/* Espacios en blanco */              {espacio} {/*Ignore*/}

/* Comentarios */                     ( "//"(.)* | "/*"(.)* ) {/*Ignore*/}

/* Salto de linea */                  ( "\n" ) {return Linea;}

/* Comillas */                        ( "\"" ) {lexeme=yytext(); return Comillas;}

/* Tipos de datos */                  ( "real"  ) {lexeme=yytext(); return T_dato;}

/* Tipo de dato Cadena */             ( "cadena" ) {lexeme=yytext(); return Cadena;}

/*Tipo dato Nulo*/                    ( "nulo" ) {lexeme=yytext(); return Nulo;}

/*Entero Main*/                       (entero) {lexeme=yytext(); return Entero;}

/* Palabra reservada Si */            (si) {lexeme=yytext(); return Si;}

/* Palabra reservada Entonces */(entonces) {lexeme=yytext(); return Entonces;}

/* Palabra reservada Sino */          ( "sino" ) {lexeme=yytext(); return Sino;}

/* Palabra reservada Hacer */         ( "hacer" ) {lexeme=yytext(); return Hacer;}

/* Palabra reservada Mientras */      ( "mientras" ) {lexeme=yytext(); return Mientras;}

/* Palabra reservada Para */          ( "para" ) {lexeme=yytext(); return Para;}

/* Palabra reservada Devolver */      ( "devolver" ) {lexeme=yytext(); return Devolver;}

/* Palabra reservada Instanciar */    ( "instanciar" ) {lexeme=yytext(); return
                                      Instanciar;}

/* Palabra reservada Leer */          ( "leer" ) {lexeme=yytext(); return Leer;}

/* Palabra reservada Escribir */      ( "escribir" ) {lexeme=yytext(); return Escribir;}

/* Punto */                           ( "." ) {lexeme=yytext(); return Punto;}

/* Operador Igual */                  ( "=" ) {lexeme=yytext(); return Igual;}

/* Operador Suma */                   ( "+" ) {lexeme=yytext(); return Suma;}
```

| | |
|---|---|
| /* Operador Resta */ | ( "-" ) {lexeme=yytext(); return Resta;} |
| /* Operador Multiplicacion */ | ( "*" ) {lexeme=yytext(); return Multiplicacion;} |
| /* Operador Division */ | ( "/" ) {lexeme=yytext(); return Division;} |
| /* Operador Mod */ | ( "%" ) {lexeme=yytext(); return Mod;} |
| /* Operadores logicos */ | ( "&&" \| "\|\|" \| "!" \| "&" \| "\|" ) {lexeme=yytext(); return Op_logico;} |
| /*Operadores Relacionales */ | ( ">" \| "<" \| "==" \| "!=" \| ">=" \| "<=" \| "<<" \| ">>" ) {lexeme = yytext(); return Op_relacional;} |
| /* Operadores Atribucion */ | ( "+=" \| "-=" \| "*=" \| "/=" \| "%=" \| ":") {lexeme = yytext(); return Op_atribucion;} |
| /* Operadores Incr. y dec. */ | ( "++" \| "--" ) {lexeme = yytext(); return Op_incremento;} |
| /*Operadores Booleanos*/ | (verdadero \| falso) {lexeme = yytext(); return Op_booleano;} |
| /* Parentesis de apertura */ | ( "(" ) {lexeme=yytext(); return Parentesis_a;} |
| /* Parentesis de cierre */ | ( ")" ) {lexeme=yytext(); return Parentesis_c;} |
| /* Llave de apertura */ | ( "{" ) {lexeme=yytext(); return Llave_a;} |
| /* Llave de cierre */ | ( "}" ) {lexeme=yytext(); return Llave_c;} |
| /* Corchete de apertura */ | ( "[" ) {lexeme = yytext(); return Corchete_a;} |
| /* Corchete de cierre */ | ( "]" ) {lexeme = yytext(); return Corchete_c;} |
| /* Marcador de inicio */ | ( principal \| Principal ) {lexeme=yytext(); return Principal;} |
| /* Marcador de constructor */ | ( constructor ) {lexeme=yytext(); return Constructor;} |
| /* Marcador inicio clase */ | (clase) {lexeme=yytext(); return Clase;} |
| /* Marcador Metodo */ | (Metodos \| metodos \| metodo \| Metodo ) {lexeme=yytext(); return Metodo;} |

| | |
|---|---|
| /* Marcador de Propiedades */ | ( propiedad \| propiedades ) {lexeme=yytext(); return Propiedad;} |
| /* Dato por disponibilidad*/ | ( publicas \| privadas \| publicos \| privados) {lexeme=yytext(); return T_Disponible;} |
| /* Punto y coma */ | ( ";" ) {lexeme=yytext(); return P_coma;} |
| /* Identificador */ | {L}({L}\|{D})* {lexeme=yytext(); return Identificador;} |
| /* Numero */ | ("(-"{D}+")")\|{D}+ {lexeme=yytext(); return Numero;} |
| /* Error de analisis */ | {return ERROR;} |

## b) Las expresiones regulares y las reglas sintácticas

```
terminal Linea, Comillas, T_dato, Entero, Cadena, Si, Sino, Hacer, Mientras, Para,
    Igual, Suma, Resta, Multiplicacion, Division, Op_logico, Op_relacional,
    Op_atribucion, Op_incremento, Op_booleano, Parentesis_a, Parentesis_c,
    Llave_a, Llave_c, Corchete_a, Corchete_c, Principal, P_coma,
Identificador,Clase
    Numero, Entonces,  ERROR, Leer, Escribir, Coma, Boleano, Propiedad, Metodo
    Dos_puntos;
non terminal INICIO, SENTENCIA, DECLARACION, DECLARACION_FOR, IF,
IF_ELSE, ES,
    WHILE, DO_WHILE,SENTENCIA_DOWHILE, FOR, SENTENCIA_BOOLEANA,
SENTENCIA_FOR,
    DECAFUN, CODFUN, DECACLASE, CODCLASE;

start with INICIO;

INICIO ::=
    Entero Principal Parentesis_a Parentesis_c SENTENCIA
;
SENTENCIA ::=
    DECLARACION |
    SENTENCIA_SI|
    SI |
    SENTENCIA IF_ELSE |
    IF_ELSE |
    SENTENCIA_WHILE |
    WHILE |
    SENTENCIA DO_WHILE
    DO_WHILE |
```

SENTENCIA_FOR |
FOR |
DO |
ES P_coma|
ES |
Devolver Numero |
Devolver Numero P_coma |
Devolver Op_booleano P_coma |
Devolver Op_booleano
;

ES ::=
Leer Identificador P_coma|
Leer Identificador |
Escribir Identificador coma |
Escribir Comillas Identificador Comillas

DECLARACION ::=

Entero Identificador Op_atribucion Numero Coma identificador Op_atribucion |
Entero Identificador Op_atribucion Numero Coma identificador Op_atribucion Numero |
Entero Identificador Op_atribucion Numero P_coma |
Entero Identificador Op_atribucion Numero|
Entero Identificador Igual Numero P_coma |
Entero Identificador Op_incremento P_coma |
Entero Identificador Op_incremento |
Entero Identificador Coma Identificador |
Entero Identificador P_coma|
Entero Identificador Igual Numero|
Entero Identificador |
Boleano Identificador Igual Numero |
Boleano Identificador P_coma |
Boleano Identificador |
T_dato Identificador P_coma |
T_dato Identificador |
T_dato Identificador Op_atribucion Numero P_coma |
T_dato Identificador Op_atribucion Numero |
T_dato Identificador Igual Numero P_coma |
T_dato Identificador Igual Numero|
T_dato Identificador Op_incremento P_coma |
T_dato Identificador Op_incremento |
T_dato Op_incremento Identificador P_coma |
T_dato Op_incremento Identificador |
Cadena Identificador Op_atribucion Comillas Comillas P_coma |
Cadena Identificador Op_atribucion Comillas Comillas |
Cadena Identificador Igual Comillas Comillas P_coma |

```
   Cadena Identificador Igual Comillas Comillas |
   Cadena Identificador Igual Comillas Identificador Comillas P_coma |
   Cadena Identificador Igual Comillas Identificador Comillas
   Cadena Identificador |
;

SI ::= Si Parentesis_a SENTENCIA_BOOLEANA Parentesis_c Entonces
SENTENCIA P_coma |
       Si Parentesis_a SENTENCIA_BOOLEANA Parentesis_c Entonces
SENTENCIA|
       Si SENTENCIA_BOOLEANA Entonces SENTENCIA P_coma |
       Si SENTENCIA_BOOLEANA Entonces SENTENCIA
;
SENTENCIA_BOOLEANA ::=
   Op_booleano |
   Identificador Op_relacional Op_booleano |
   Identificador Op_relacional Numero |
   Identificador Op_relacional Identificador |
   Identificador Op_relacional Comillas Comillas |
   Identificador Op_relacional Comillas Identificador Comillas |

;
IF_ELSE ::=
   Si Parentecis_a SENTENCIA_BOOLEANA Parentesis_c Entonces SENTENCIA
|
   Si Parentesis_a SENTENCIA_BOOLEANA Parentesis_c Entonces SENTENCIA
Sino  SENTENCIA P_coma
;
WHILE ::= Hacer Parentesis_a SENTENCIA Parentesis_c Mientras
SENTENCIA_BOOLEANA |
        Hacer SENTENCIA Mientras SENTENCIA_BOOLEANA
;
DO_WHILE ::= Desde SENTENCIA_BOOLEANA Mientras identificador
Op_relacional Numero Op_incremento numero Hacer SENTENCIA_DOWHILE|
          Desde identificador Mientras identificador Op_relacional Numero
Op_incremento numero Hacer SENTENCIA_DOWHILE
;
SENTENCIA_DOWHILE ::=
   DECLARACION
   Identificador Op_atribucion Numero |
   Identificador Op_incremento |
   Op_incremento Identificador
   ES P_coma|
   ES

FOR ::= Para Parentesis_a SENTENCIA_FOR Parentesis_c Llave_a SENTENCIA
Llave_c
```

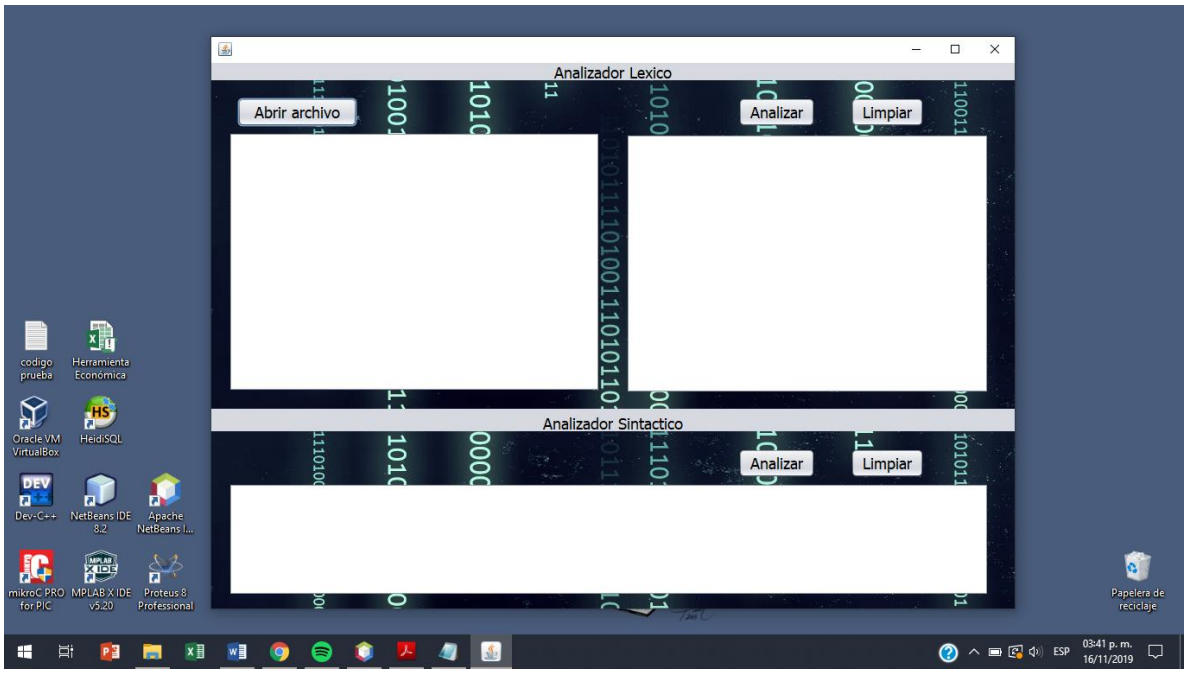
```
;
SENTENCIA_FOR ::=
   T_dato Identificador Igual Numero P_coma SENTENCIA_BOOLEANA P_coma
DECLARACION_FOR |
   Identificador Igual Numero P_coma SENTENCIA_BOOLEANA P_coma
DECLARACION_FOR
;
DECLARACION_FOR ::=
   Identificador Op_atribucion Numero |
   Identificador Op_incremento |
   Op_incremento Identificador
   ES P_coma |
   ES
;
DECAFUN::=
   Tipo_dato Identificador Parentecias_a (DECLARACION) Parentecis_C
SENTENCIA |
   Tipo_dato Identificador Parentecis_a Parentecis_c SENTENCIA |
   Tipo_dato Identificador Parentecis_a Identificador Parentecis_c SENTENCIA|
   Entero Identificador Parentecias_a (DECLARACION) Parentecis_C
SENTENCIA |
   Entero Identificador Parentecis_a Identificador Parentecis_c SENTENCIA |
   Entero Identificador Parentecis_a Parentecias_a SENTENCIA
;

CODCLASE ::=
  Propiedad Disponible Dos_puntos SENTENCIA|
  Metodo Disponible Dos_puntos SENTENCIA
;

DECACLASE ::=
   Clase Identificador Parentecis_a DECLARACION Parentecis_c CODCLASE |
   Clase Identificador Parentecis_a Identificador Parentecis_c CODCLASE |
   Clase Identificador Parentecis_a Parentecis_c CODCLASE|
   Clase Identificador Parentecis_a Parentecis_c |
   Clase Identificador CODCLASE |
   Clase Identificador|
;
```

## c) Un manual de usuario (no es un manual técnico)

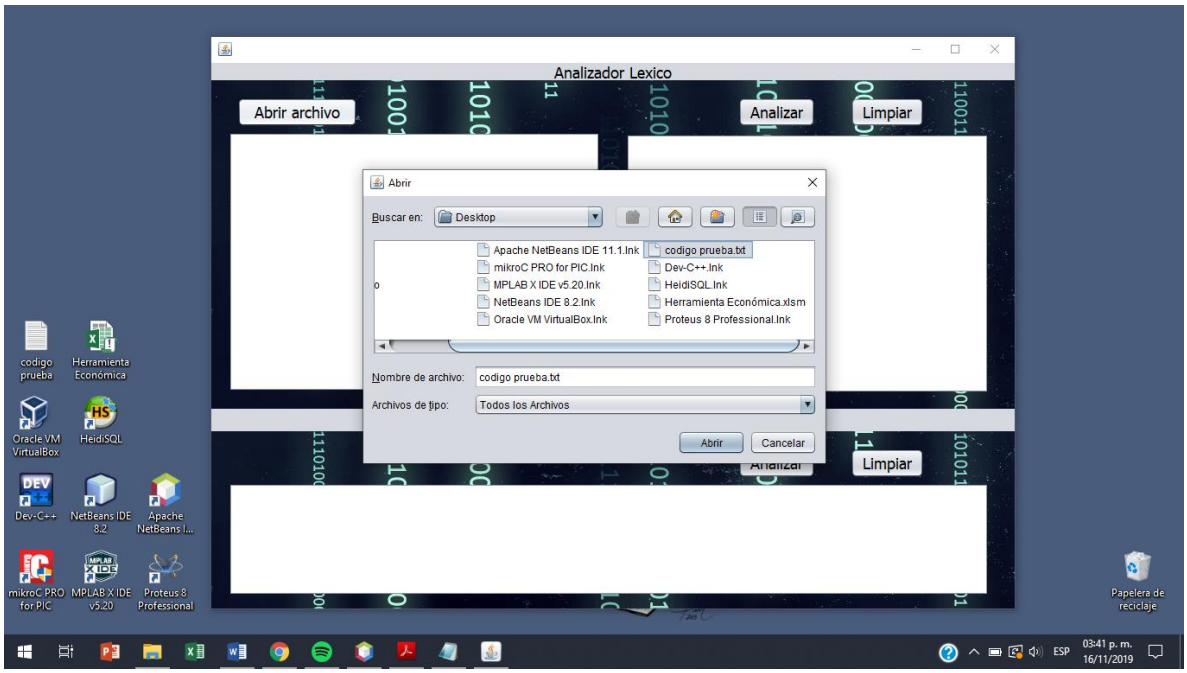
Al abrir el archivo, encontraremos una pantalla principal, esta, está dividida en dos secciones:

a. Analizador Léxico
b. Analizador Sintáctico

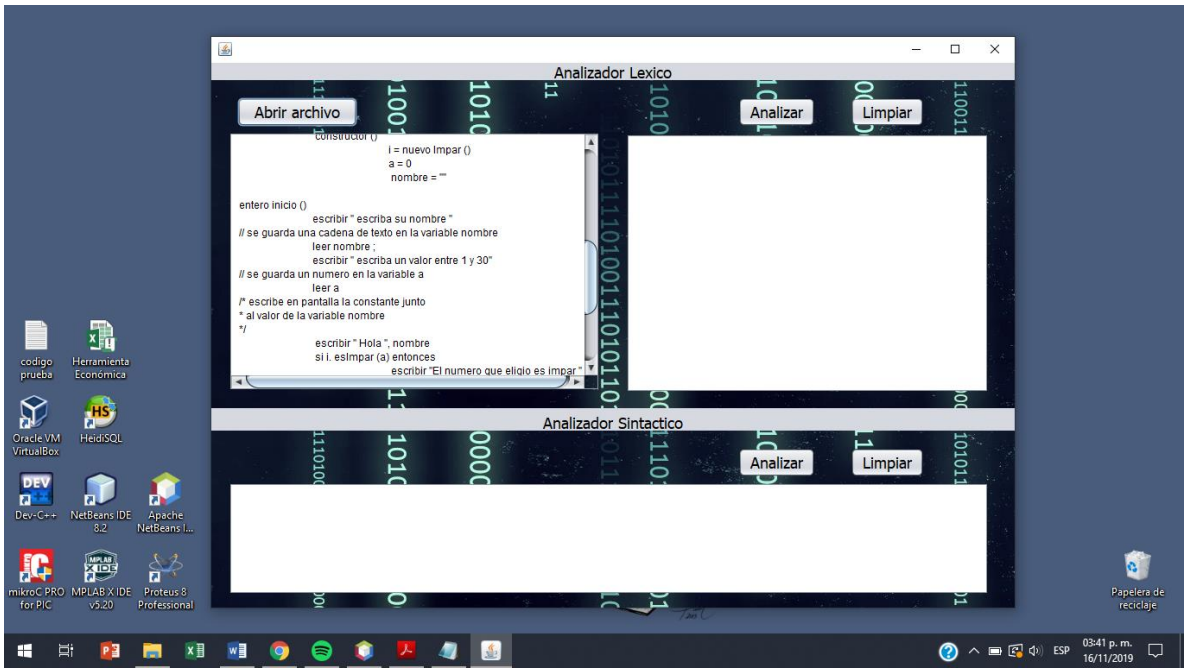
Para comenzar a analizar nuestro archivo original es necesario darle clic en el botón “Abrir Archivo”.




Al darle clic al botón, se abrirá otra ventana en la que podremos seleccionar el archivo que queremos analizar (archivo en lenguaje loop), luego darle clic al botón “Aceptar”



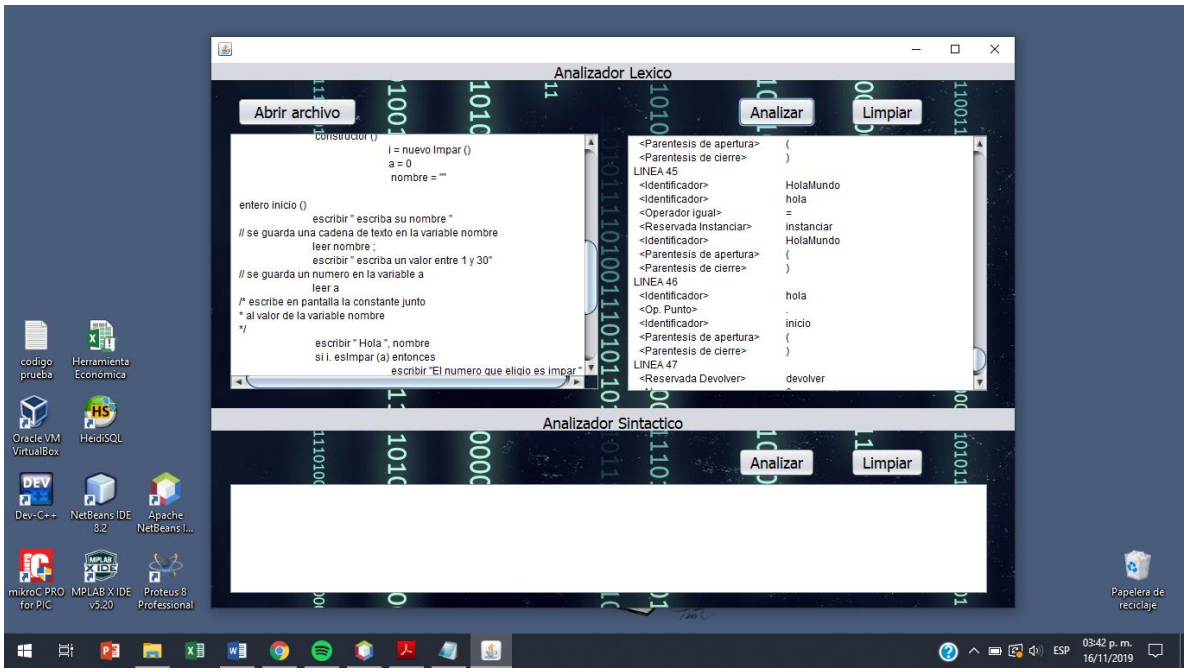

Para corroborar que abrimos el archivo, este, se abrirá en el cuadro superior izquierdo, en el que, veremos exactamente el código del archivo original.
Para someterlo al analizador léxico, dar clic en el botón “Analizar”



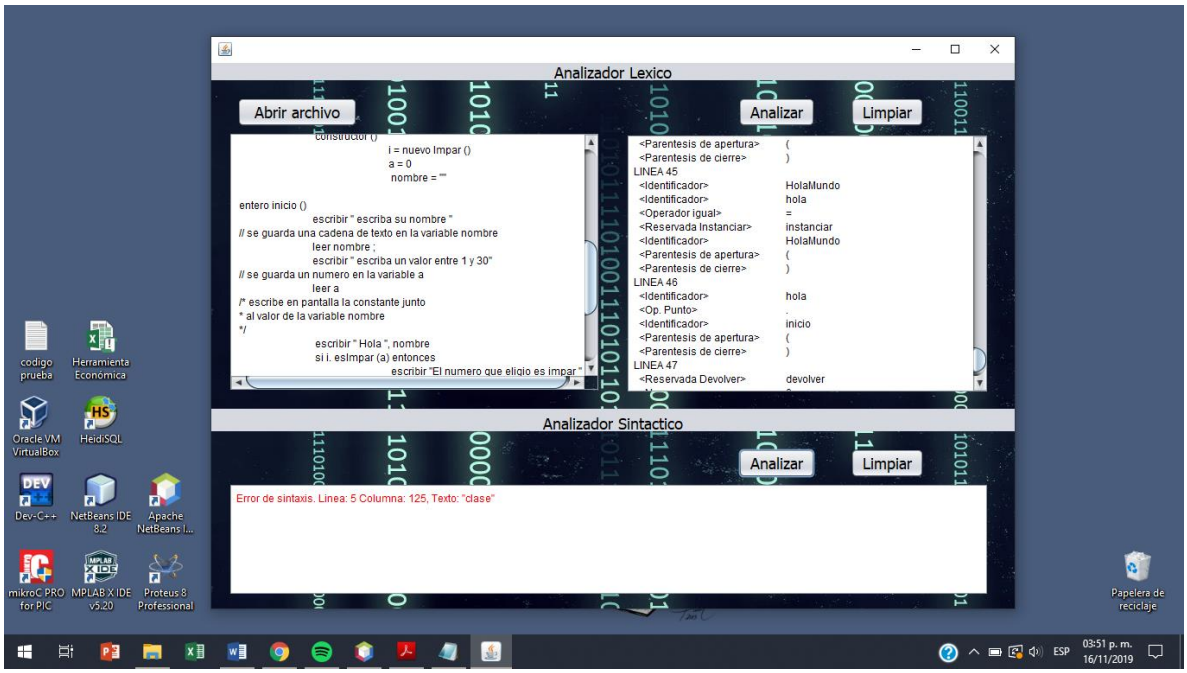

Al darle clic a dicho botón, en nuestro cuadro superior derecho veremos la lista de tokens que se generaron del archivo en lenguaje loop, dichos tokens se especifican en la primera parte de este documento.

Si se llegara a necesitar, dar clic en botón “Limpiar” para borrar de pantalla los datos generados.

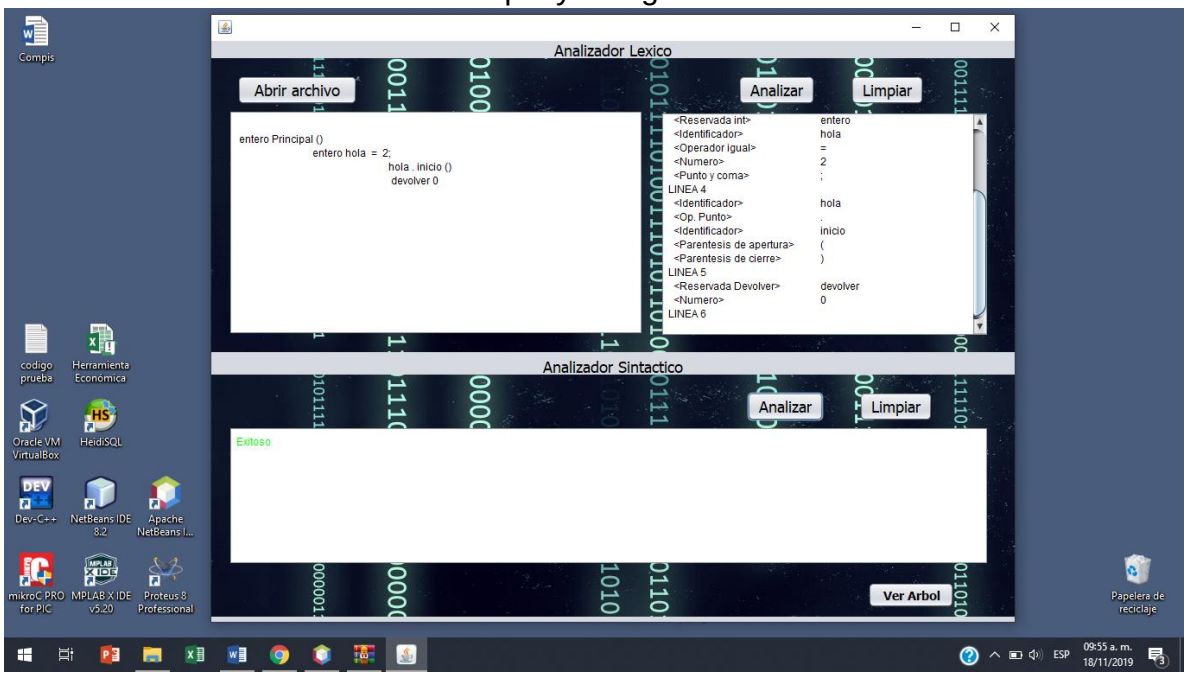
Para que nuestro archivo pase por el analizador sintáctico, dar clic en el botón “analizar” de abajo.




Al darle clic al botón “Analizar” en la parte de abajo, mostrará el resultado del análisis sintáctico, si existe algún error se mostrará de la siguiente manera, identificando el número de línea y columna en la que tenemos el error de sintaxis.




De lo contrario, activa la visibilidad del botón “Abrir Arbol” en la que visualizaremos el árbol semántico que ya se generó.



Dar click en ver árbol para visualizar lo que generó el árbol semántico, luce de este modo: