TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica

Escuela de Computación
IC-5701 Compiladores e Intérpretes
Prof. Ing. Erika Marín Schumann

Primer Proyecto del Compilador *PCL*

Integrantes:
Daniel Alberto Troyo Garro 2014073396
Bernal González Magaña 2013003475
Eros Hernández Romero 2014081083

Scanner de tokens del lenguaje *PCL*

Lunes 3 de Octubre
II Semestre
2016

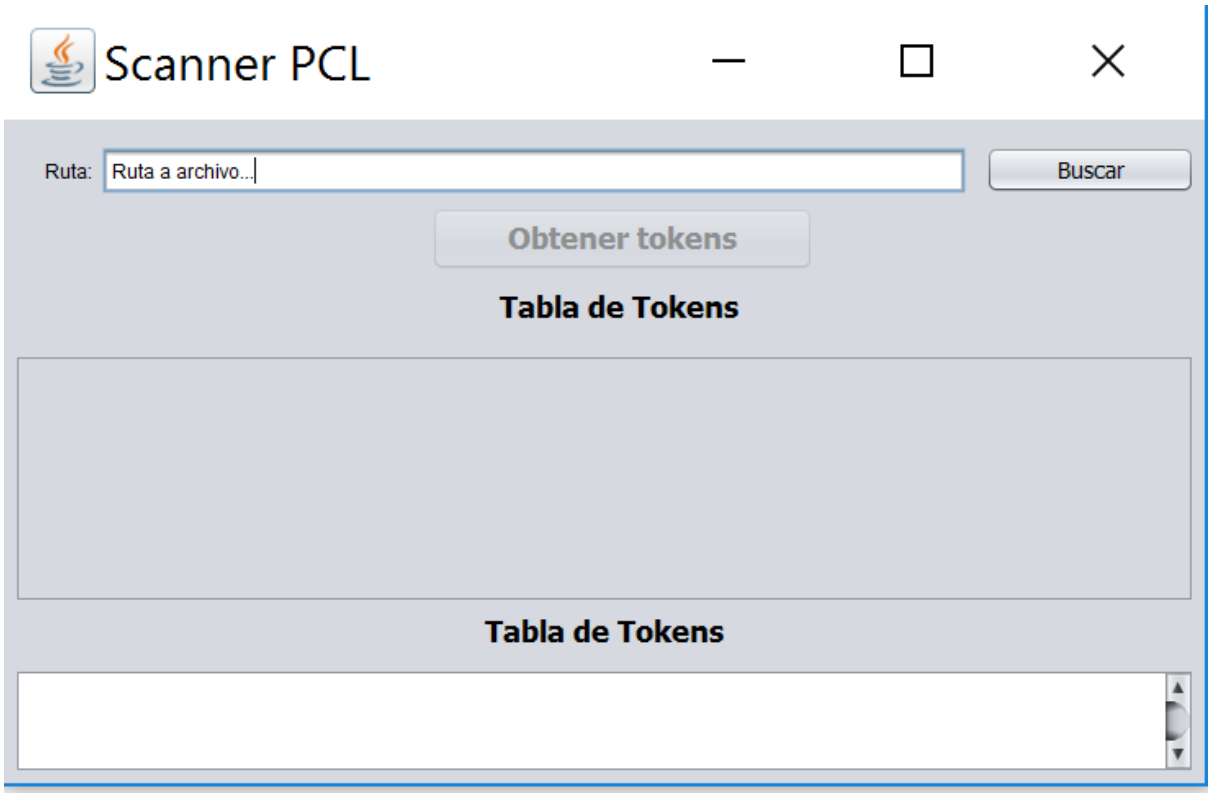
Índice

Scanner PCL

Manual de Uso:

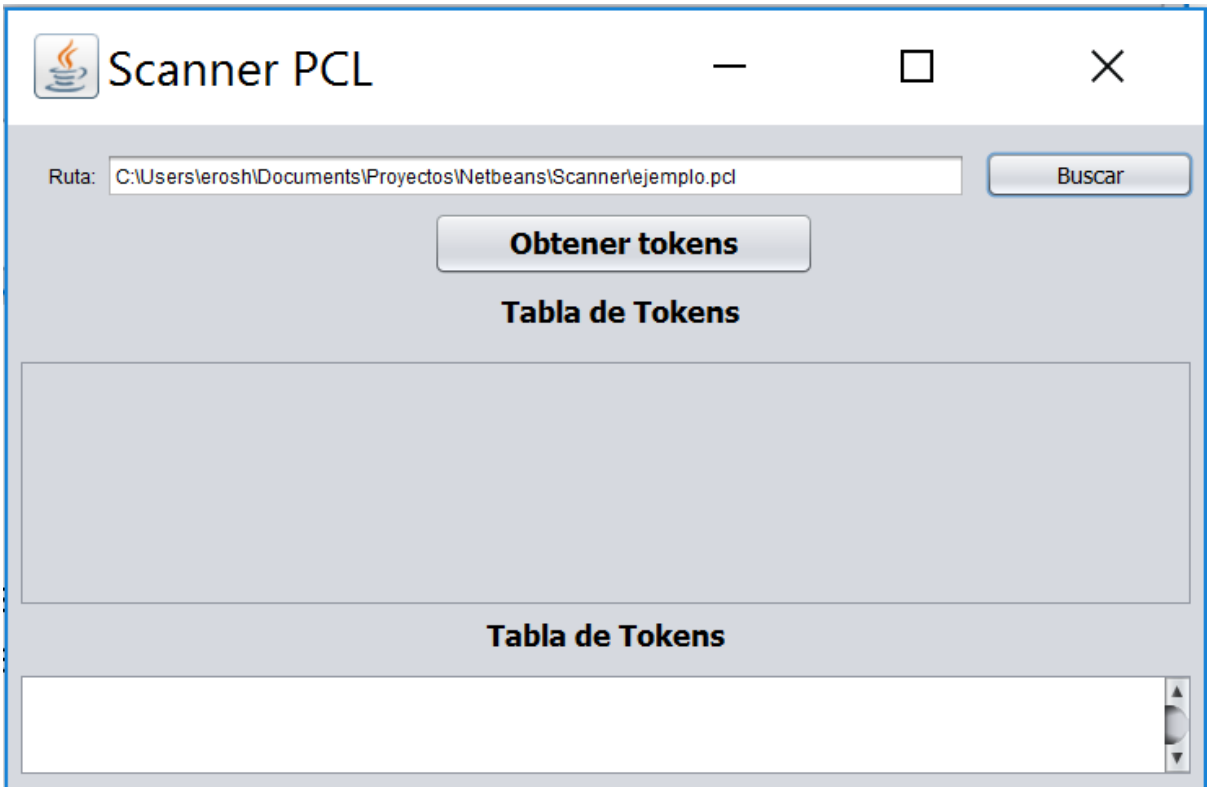
1. **Ejecutable**

Dentro de la carpeta del proyecto, se realiza ‘doble click’ sobre el archivo ‘Scanner.jar’. Cabe mencionar que se debe tener previamente instalado una versión actual del JDK. Al realizar esta acción, se abrirá la siguiente ventana:



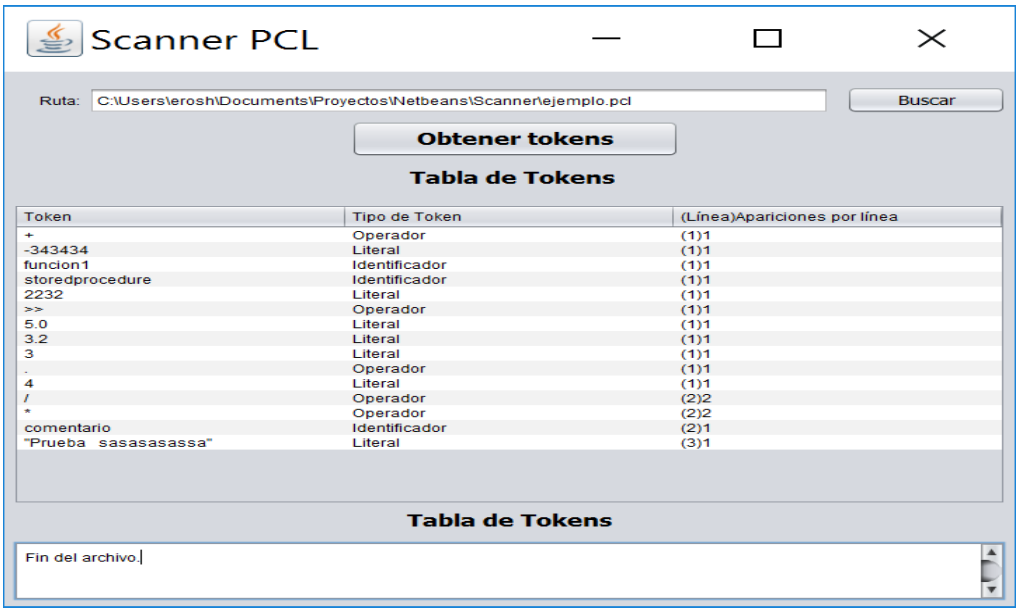
2. **Selección de archivo .pcl**

Al contar con un archivo ‘.pcl’ que contenga una estructura de código que se desea analizar, se procede a dar ‘click’ sobre el botón de ‘Buscar’ y se abrirá la siguiente ventana, en donde se busca y selecciona el archivo ‘.pcl’ a analizar.



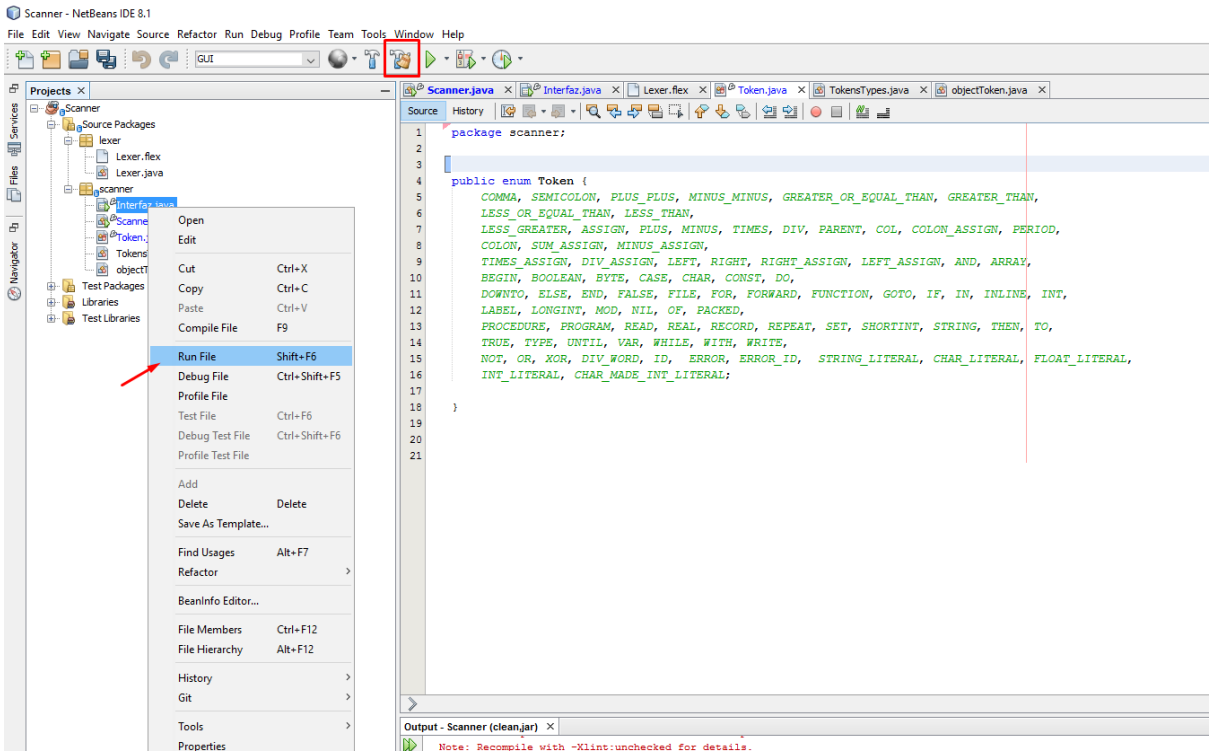
3. Obtención de Tokens

Para generar la Tabla de Tokens que contendrá el listado de los tokens encontrados, bastará con presionar el botón 'Obtener Tokens' para generarla. Se obtendrá un resultado como el de la siguiente ventana:



4. Compilación del programa

Para la compilación del programa desde el código fuente se requiere abrir el proyecto de *NetBeans* contenido dentro de la carpeta del código fuente y, una vez abierto, se puede realizar la compilación de dos maneras dependiendo de si se quiera obtener un nuevo archivo *.jar* o no. Si no se quiere obtener un archivo *.jar* y simplemente se quiere ver el código compilándose y ejecutándose desde el código fuente, basta con hacer *click derecho* sobre el archivo *Interfaz.java* en la barra lateral de *Proyectos* de *NetBeans* y presionar *Run File*.



Si se quiere generar un nuevo archivo *.jar* del proyecto, ya sea por alguna modificación al código o por algún otro motivo, se presiona el botón *Build and Clean* marcado en rojo en la imagen. Esto generará un nuevo archivo *jar* en la carpeta `dist` del proyecto. Este archivo será llamado por defecto *Scanner.jar.*

Análisis de resultados

Se ha desarrollado un programa escrito en Java, correspondiente a un Scanner que recibe un código fuente escrito en lenguaje ficticio PCL utilizando la librería JFlex. El programa realiza un análisis léxico del código fuente ingresado. Se cumple en su totalidad con las 2 funcionalidades requeridas:

- **Tabla de Tokens encontrados:** Al analizar el código, el programa lleva un control de todos los Tokens o palabras aceptadas. Al finalizar los presenta en un tabla, cumpliendo el requisito de mostrar cada uno de estos tokens, el tipo de token que son, la línea de código en la que se encuentran y la cantidad de ocurrencias por línea de dicho token.

- **Listado de errores léxicos encontrados:** Se despliega una lista que contiene todos los errores léxicos que fueron encontrados en el código fuente. El programa se recupera de los errores, es decir que no despliega errores en cascada y continua con el análisis a pesar de encontrar errores.

Casos de Prueba:

Nombre de caso de prueba:

Prueba de `CommentTest.pcl`

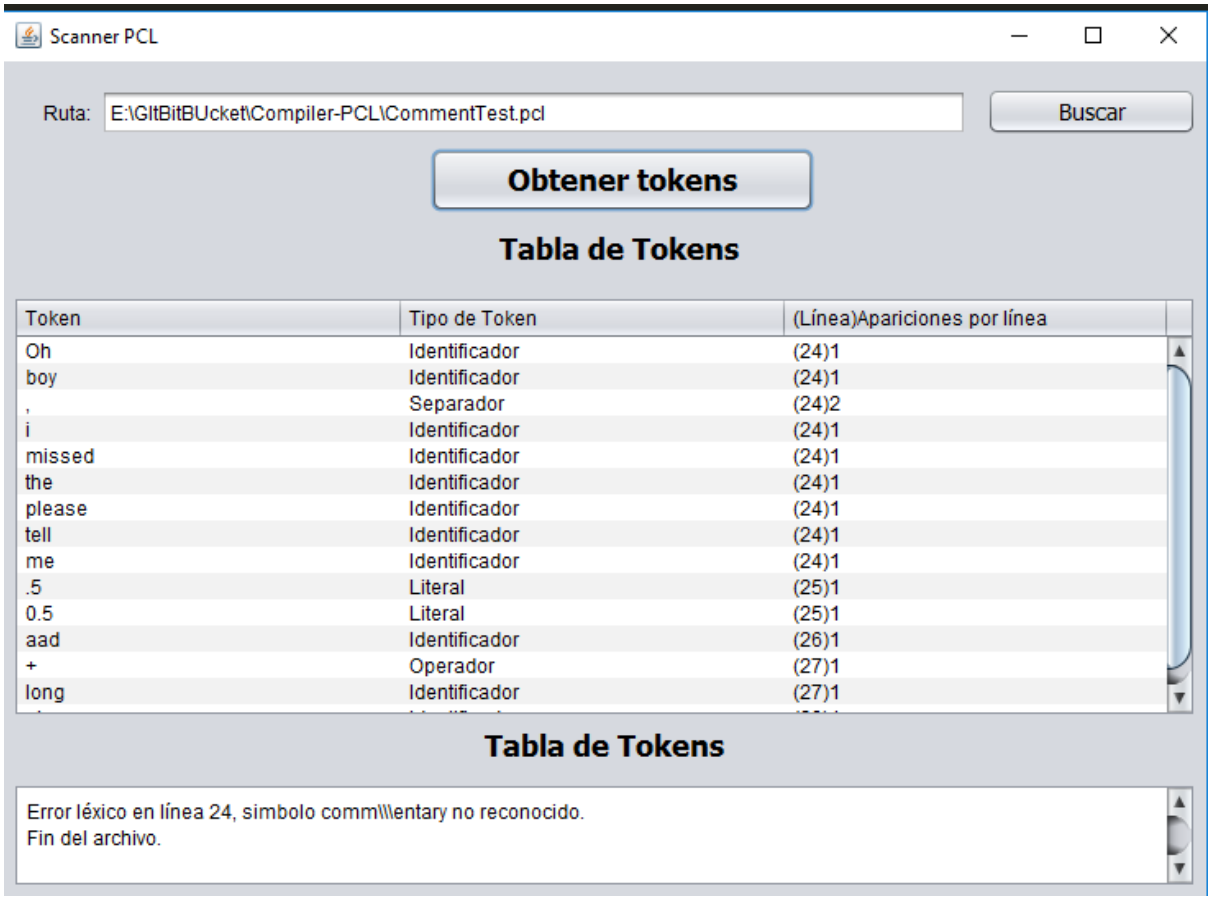


Tabla de Tokens:

Token	Tipo de Token	(Línea)Apariciones por línea
Oh	Identificador	(24)1
boy	Identificador	(24)1
,	Separador	(24)2
i	Identificador	(24)1
missed	Identificador	(24)1
the	Identificador	(24)1
please	Identificador	(24)1
tell	Identificador	(24)1

me	Identificador	(24)1
.5	Literal	(25)1
0.5	Literal	(25)1
aad	Identificador	(26)1
+	Operador	(27)1
long	Identificador	(27)1
oh	Identificador	(28)1

Error:

```
Error léxico en línea 24, símbolo comm\\\entary no reconocido.
```

Aspectos Relevantes:

1. Comentarios de línea
2. Comentarios de bloque en ambos formatos (**) {}
3. Palabras reservadas en comentarios
4. Identificadores, operadores literales en comentario
5. Identificadores con símbolos entre ellos
6. Identificadores no válidos

Nombre de caso de prueba:

Prueba de `GENERAL_TEST.pcl`

Tabla de Tokens:

Token	Tipo de Token	(Línea)Apariciones por línea
VAR1	Identificador	(4)1
=	Operador	(4)1 (5)1 (6)1 (7)1 (8)1 (9)1 (10)1 (16)1
5	Literal	(4)1
var2	Identificador	(5)1 (12)1 (17)1
6	Literal	(5)1
var3	Identificador	(6)1 (16)1 (17)1
7	Literal	(6)1
var	Identificador	(7)1 (16)1 (17)1
test4	Identificador	(7)1 (16)1 (17)1
0	Literal	(7)1

science1	Identificador	(8)1
3.0E4	Literal	(8)1
science2	Identificador	(9)2
(	Separador	(9)1 (13)1
2.7E-4	Literal	(9)1
)	Separador	(9)1 (13)1
-	Operador	(9)1 (16)1
whatif	Identificador	(10)1
"That kills the Scanner"	Literal	(10)1
IF	Palabra reservada	(12)1 (17)1
var1	Identificador	(12)1 (16)1 (17)1
<=	Operador	(12)1
THEN	Palabra reservada	(12)1 (17)1
WRITE	Palabra reservada	(13)1
"This is going great"	Literal	(13)1
ELSE	Palabra reservada	(15)1
<	Operador	(17)1
AND	Operador	(17)1
>	Operador	(17)1

Errores:

```
Error léxico en línea 7, símbolo _ no reconocido.
Error léxico en línea 16, símbolo _ no reconocido.
Error léxico en línea 17, símbolo _ no reconocido.
```

Aspectos Relevantes:

1. Asignación de literales a identificadores
2. Literales de distintos tipos: notación científica, integer, flotantes.
3. Identificadores caseless
4. Combinación entre identificadores y palabras reservadas
5. Asignación de strings
6. Identificadores comenzados en número

7. Identificadores de más de 127 caracteres
8. Operadores entre identificadores
9. Combinación de palabras reservadas

Nombre de caso de prueba:

Prueba de `ReservedTokens.pcl`

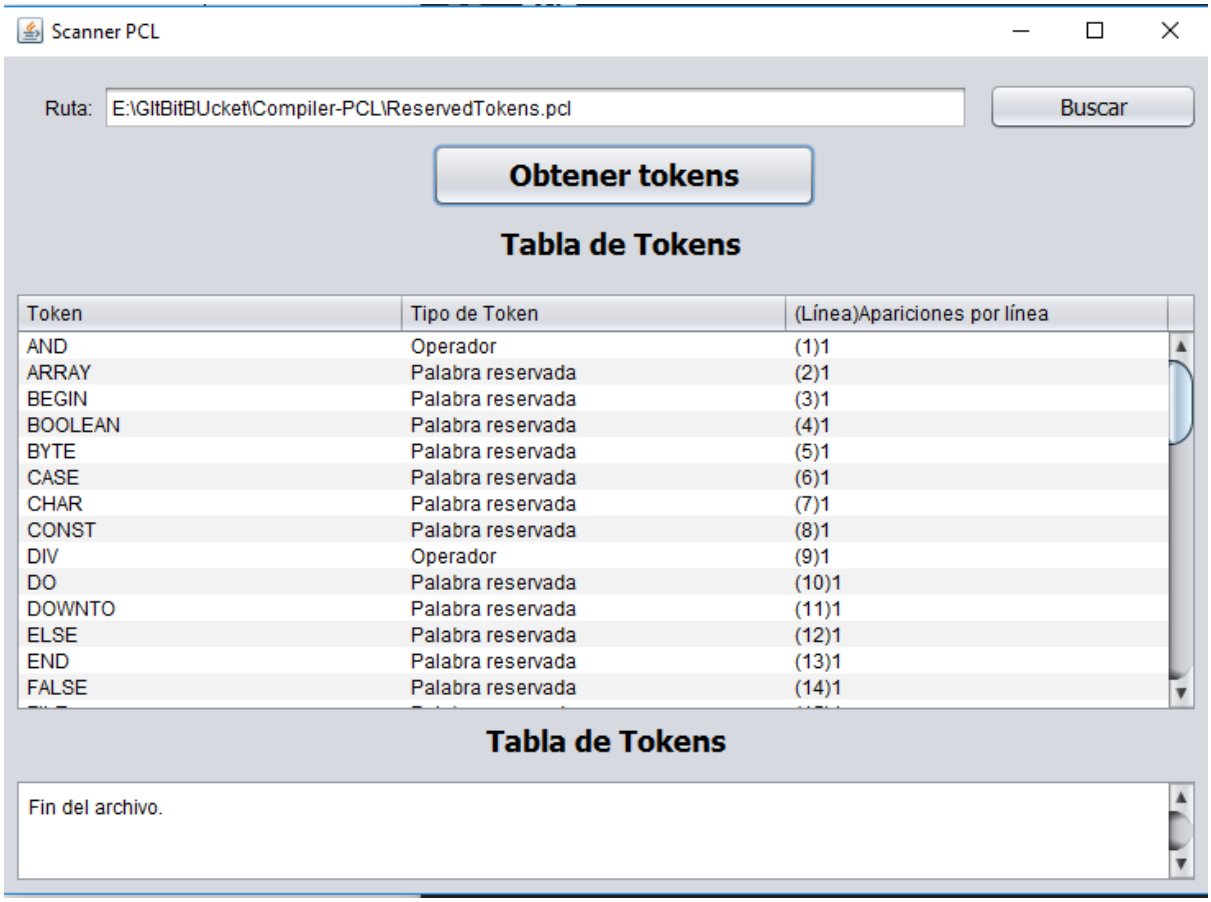


Tabla de Tokens:

Token	Tipo de Token	(Línea)Apariciones por línea
AND	Operador	(1)1
ARRAY	Palabra reservada	(2)1
BEGIN	Palabra reservada	(3)1
BOOLEAN	Palabra reservada	(4)1
BYTE	Palabra reservada	(5)1
CASE	Palabra reservada	(6)1
CHAR	Palabra reservada	(7)1
CONST	Palabra reservada	(8)1
DIV	Operador	(9)1

DO	Palabra reservada	(10)1
DOWNTO	Palabra reservada	(11)1
ELSE	Palabra reservada	(12)1
END	Palabra reservada	(13)1
FALSE	Palabra reservada	(14)1
FILE	Palabra reservada	(15)1
FOR	Palabra reservada	(16)1
FORWARD	Palabra reservada	(17)1
FUNCTION	Palabra reservada	(18)1
GOTO	Palabra reservada	(19)1
IF	Palabra reservada	(20)1
IN	Palabra reservada	(21)1
INLINE	Palabra reservada	(22)1
INT	Palabra reservada	(23)1
LABEL	Palabra reservada	(24)1
LONGINT	Palabra reservada	(25)1
MOD	Operador	(26)1
NIL	Palabra reservada	(27)1
NOT	Operador	(28)1
OF	Palabra reservada	(29)1
OR	Operador	(30)1
PACKED	Palabra reservada	(31)1
p	Identificador	(32)1
.5	Literal	(32)1
#23	Literal	(32)1
rocedure	Identificador	(33)1
PROGRAM	Palabra reservada	(34)1
READ	Palabra reservada	(35)1
REAL	Palabra reservada	(36)1

RECORD	Palabra reservada	(37)1
REPEAT	Palabra reservada	(38)1
SET	Palabra reservada	(39)1
SHORTINT	Palabra reservada	(40)1
STRING	Palabra reservada	(41)1
THEN	Palabra reservada	(42)1
TO	Palabra reservada	(43)1
TRUE	Palabra reservada	(44)1
TYPE	Palabra reservada	(45)1
UNTIL	Palabra reservada	(46)1
VAR	Palabra reservada	(47)1
WHILE	Palabra reservada	(48)1
WITH	Palabra reservada	(49)1
WRITE	Palabra reservada	(50)1
XOR	Operador	(51)1

Errores:

```
Error léxico en línea 52, Identificador:
WRITEXORRECORDREPEATSETSHORTINTSTRINGTHENTOTRUETYPEUNTILWRITEXORRECORDREPE
ATSETSHORTINTSTRINGTHENTOTRUETYPEUNTILWRITEXORRECORDREPEATSETSHORTINTSTRIN
GTHENTOTRUETYPEUNTIL contiene más de 127 caractéres.
```

Aspectos Relevantes:

1. Identificador de más de 127 caracteres.
2. Combinación de palabras reservadas y saltos de líneas.
3. Presencia de literales de flotantes.