PRÁCTICA OBLIGATORIA: ANALIZADOR LÉXICO Y SINTÁCTICO.

Grupo 10

Sergio Santamaría Carrasco G. Ing Informática y G. Matemáticas

José Pertierra das Neves G. Ing Informática y G. Matemáticas

Guillermo Nieto Barba G. Ing Informática y G. Matemáticas

Contenido

1.- Introducción

La práctica desarrollada consta de un analizador léxico y uno sintáctico, ambos integrados en un único ejecutable, junto con algunos ejemplos de prueba (4 erróneos y 4 correctos) de código escrito en Pascal.

2.- Desarrollo

Hemos realizado el analizador sintáctico gracias a la herramienta JFlex estudiada en clase, al principio solo reconocía los lexemas y los mostraba por pantalla con sus tokens y posición, después comenzamos a añadir detección y localización de errores, como comentarios mal cerrados o lexemas no reconocidos.

Los lexemas que reconoce y de los cuales crea tokens son: las palabras reservadas, operadores aritméticos, comparadores, constantes reales, constantes de enteros, constantes en hexadecimal, strings, identificadores y símbolos, como el punto y coma, paréntesis, corchetes…

Además reconoce lexemas pero de los que no crea tokens como son los comentarios y espacios, en estos casos, el analizador avisa de que los ha reconocido.

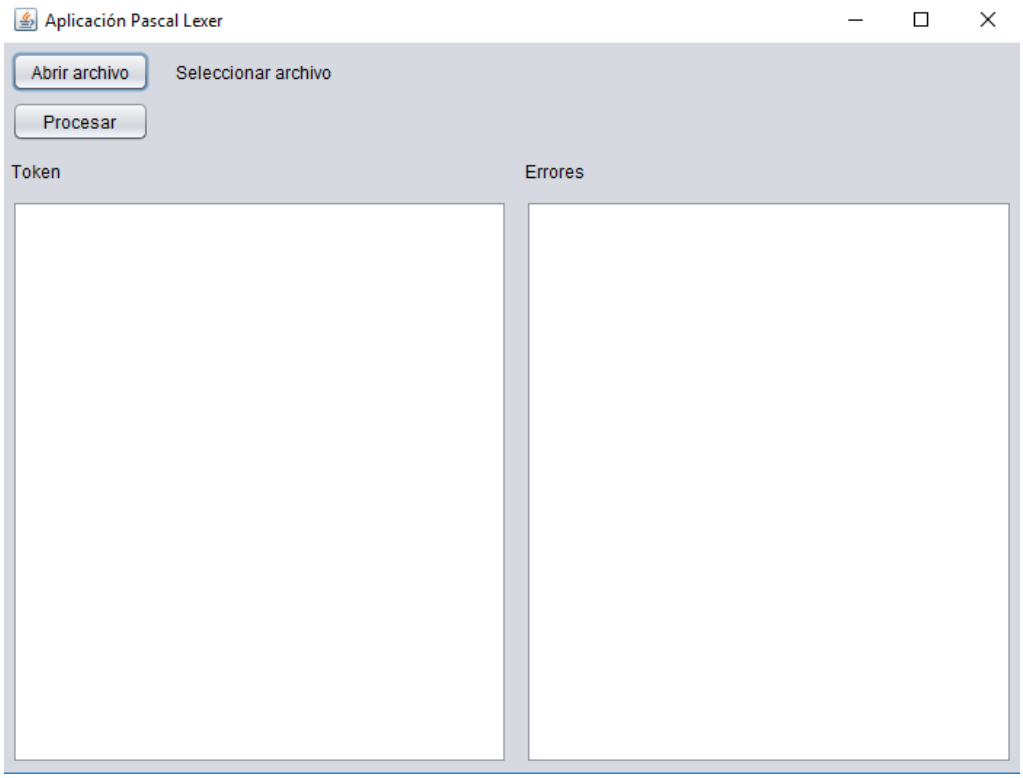
Finalmente en el caso de encontrar un error léxico lo localiza, comunica y continua con en análisis. En el caso de comentarios mal cerrados, el propio analizador los reconoce y avisa, en el caso de encontrar un lexema sin reconocer el analizador avisa del error.

Para crear el analizador sintáctico usamos la herramienta CUP, el analizador léxico envía los tokens al analizador léxico. Tuvimos que modificar el léxico para que reconociese el archivo de entrada con un objeto de la clase parser, además al reconocer un lexema se muestra por pantalla el token y el lexema y además crear objetos de la clase sym, clase creada por la herramienta CUP, los cuales equivalen a los tokens y es con lo que trabaja el analizador sintáctico.

Recibidos los tokens en el parser que se encargará, mediante las producciones de la gramática especificadas en el archivo .cup, reconocer las combinaciones de éstos que podremos aceptar. Para la recuperación de errores, usamos en las distintas producciones la combinación “error token_de_recuperación”, este último será el que permita al analizador continuar y recuperarse, siempre que consiga reconocer un número determinado, por defecto 3. Para la notificación de errores, sobreescribimos las siguientes funciones por defecto que genera CUP: report_error, syntax_error, report_fatal_error. Además hacemos

uso de una función de propia implementación, lectura_error, que almacena, en una pila, mensajes de error que más tarde usará la función report_error.

Para finalizar creamos una aplicación java que usa todas las clases creadas para analizar un fichero introducido como parámetro. En esta aplicación también se ha implementado una interfaz gráfica.



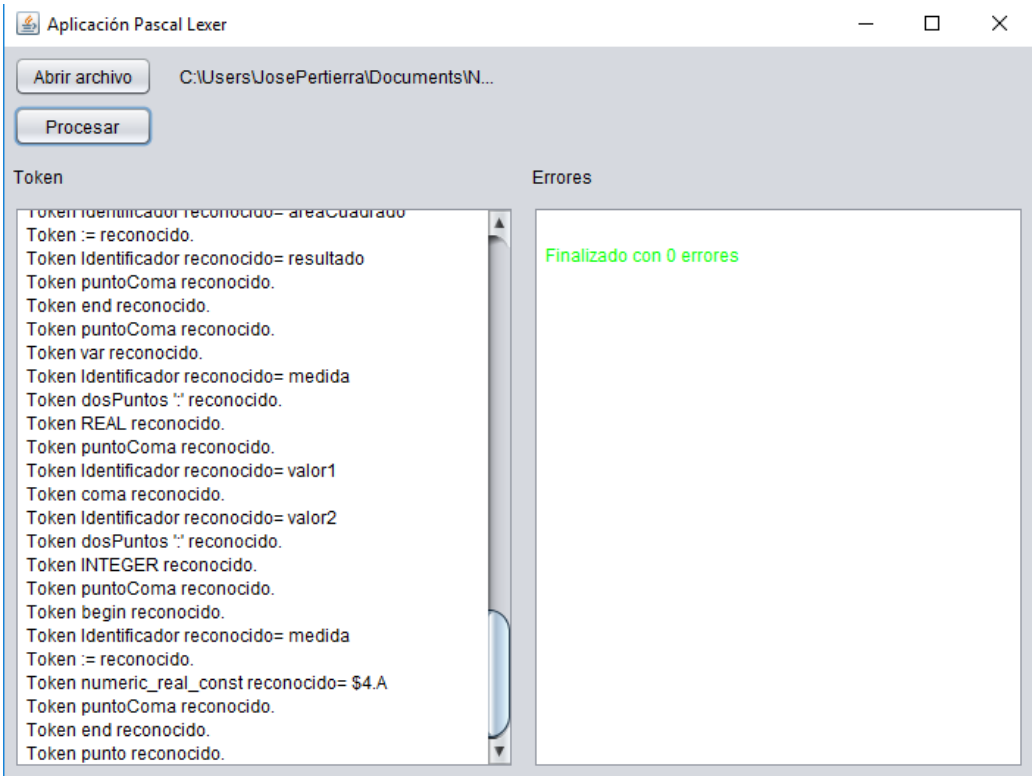
3.- Casos de funcionamiento correctos

3.1.- Ejemplo 1

- Veamos el correcto funcionamiento de declaraciones y uso de array y registros así como otras operaciones o sentencias más generales.

```
program EjemploAprobado;
function areaCuadrado ( variable : REAL ) : REAL ;
```

```
var
    resultado: REAL;
type
    vector1 = array [1..4] of INTEGER;
    datos = record
       campo1:INTEGER;
       campo2:REAL
    end;
begin
    vecto1[1] := 0;
    datos.campo1 := 0;
    resultado := +0.0;
    resultado := lado * lado;
    areaCuadrado := resultado;
end;
var
    medida :REAL;
    valor1, valor2: INTEGER;
begin
    medida := $4.A;
end.
```

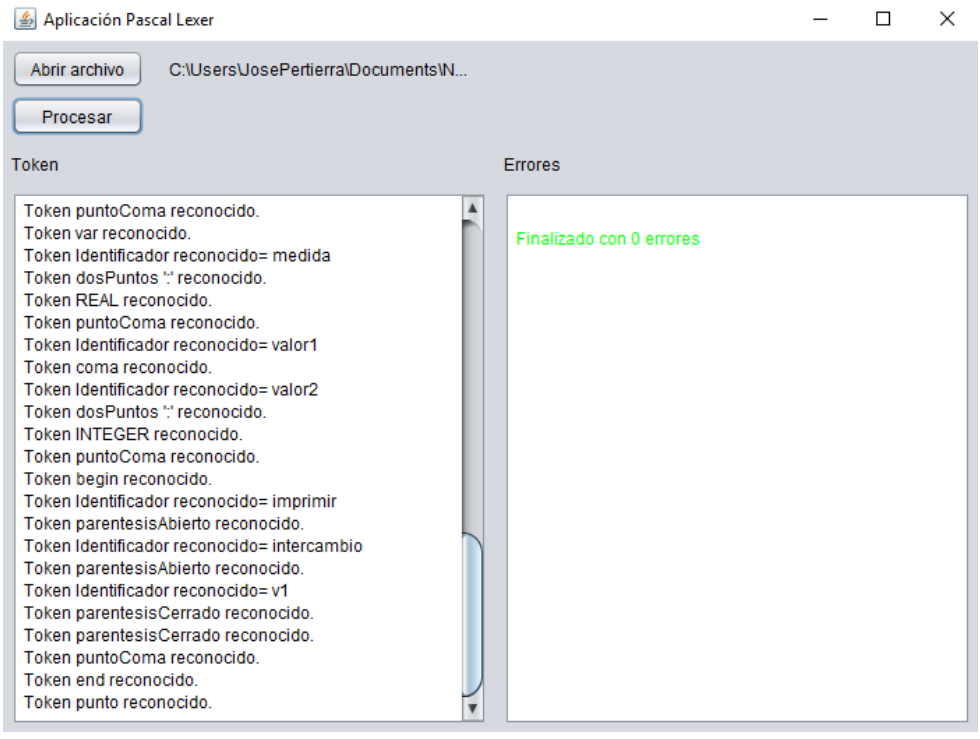


3.2 Ejemplo 2

- En este ejemplo concreto veremos el correcto funcio

namiento sobre un código que incluye un control de flujo If y uno Case, así como otras sentencias más generales.

```
program EjemploAprobado;
procedure intercambio ( v1, v2: INTEGER );
var
    aux: INTEGER;
begin
    if ( v1 > v2 ) then
       v1 := 2;
    else
    case v1 of
       4: resultado :=2 ;
    end;
end;
var
    medida :REAL;
    valor1, valor2: INTEGER;
begin
    imprimir(intercambio(v1));
end.
```

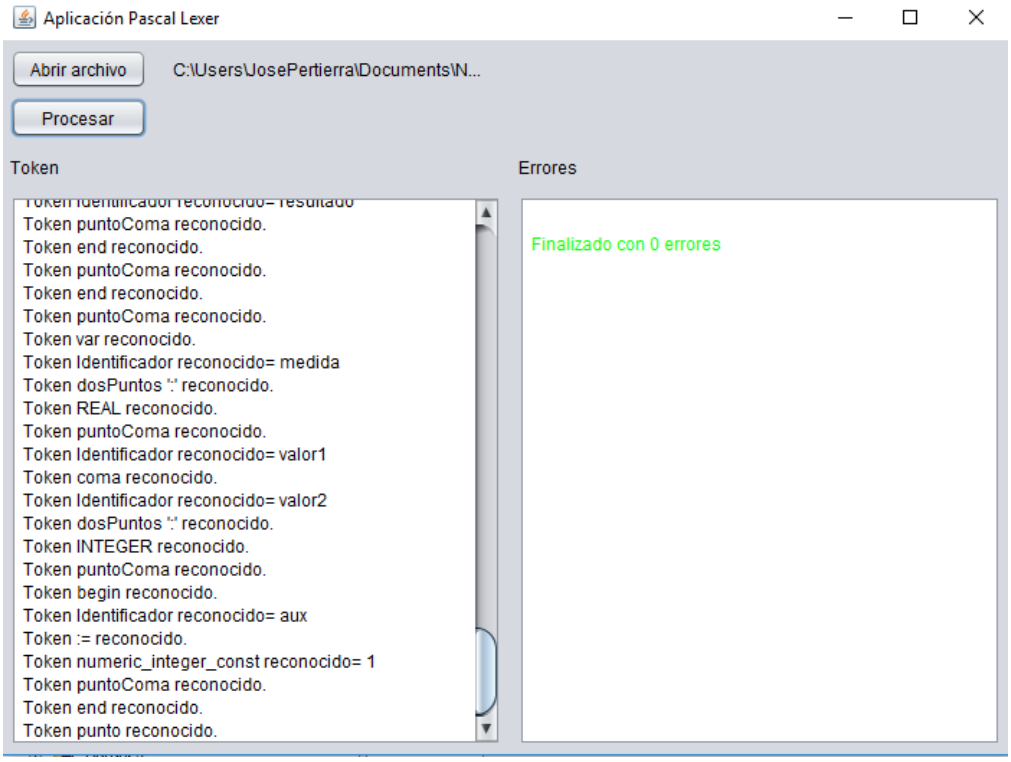


3.3.- Ejemplo 3

- En este ejemplo veremos el funcionamiento de arrays, registros y controles de flujo como If, bucle While y For, y Case, así como otras sentencias más generales.

```
program EjemploAprobado;
procedure intercambio ( v1, v2: INTEGER );
type
    vector1 = array [1..3] of INTEGER;
    datos = record
       campo1:INTEGER;
       campo2:REAL
    end;
var
    aux: INTEGER;
begin
    aux := 0;
    aux := v1;
    v1 := v2;
    v2 := aux;
    if ( v1>v2) then
       v1 := 2;
    else
       v1 := 3;
    while (2>3) do
       resultado := lado + lado;
    for aux := datos.campo1 to 5 do
       resultado := aux;
    case aux of
       3: resultado;
    end;
end;
var
    medida :REAL;
```

```
    valor1, valor2: INTEGER;
begin
    aux:=1;
end.
```

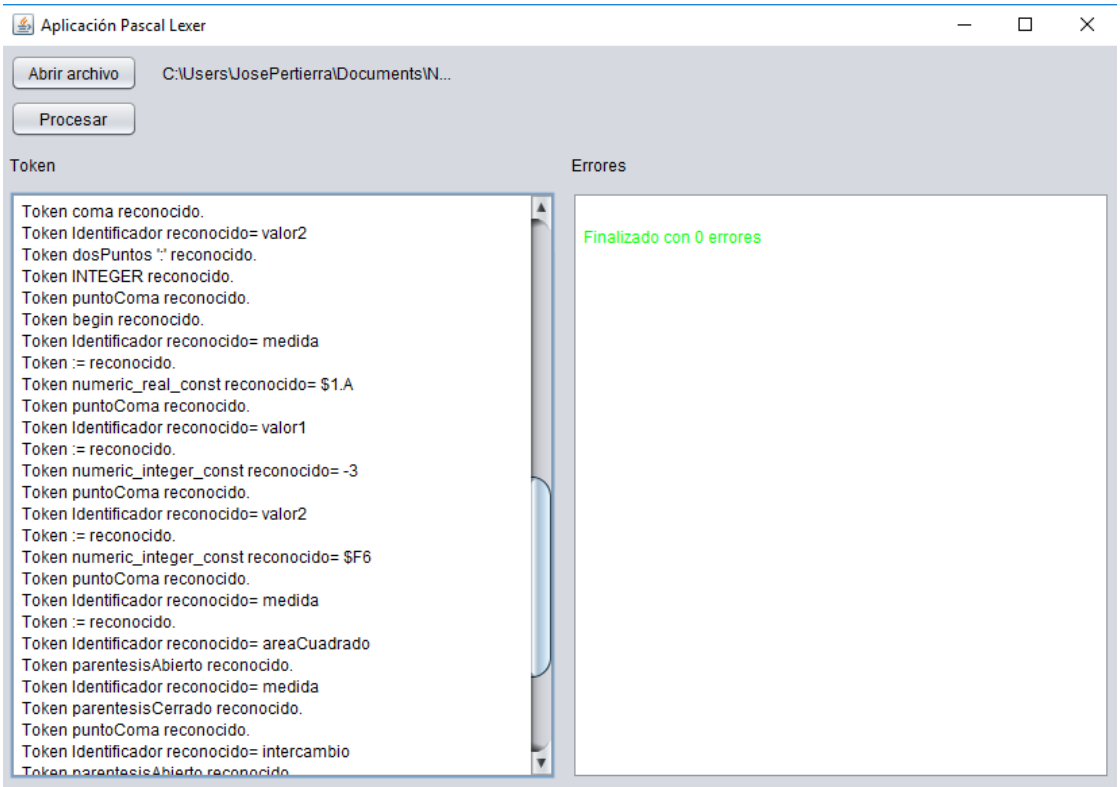


3.4.- Ejemplo 4

- Comprobaremos el correcto funcionamiento de los comentarios (de ambos tipos) así como otras sentencias generales.

```
program Prueba;
procedure intercambio ( v1, v2: INTEGER );
var
    aux: INTEGER;
begin
    aux := 0;
end;
(* Esto es un
 comentario *)
var
```

```
    medida :REAL;
    valor1, valor2: INTEGER;
begin
    medida := $1.A;
    valor1 := -3;
    valor2 := $F6;
    medida := areaCuadrado( medida );
    intercambio(valor1, valor2);
    {Esto es otro
      comentario}
end.
```



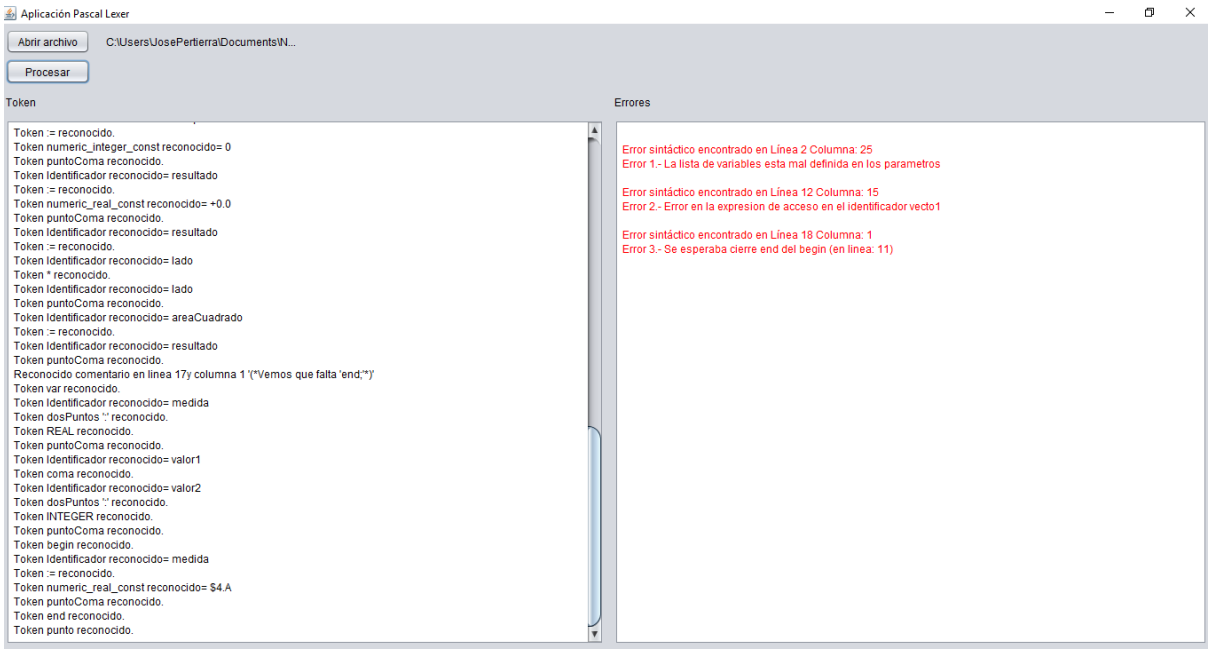
4.- Casos de error

4.1.- Ejemplo 1

-Veremos fallos en lista de parámetros, sentencia de asignación con un vector y cierre programa. El código viene dado por:

```
program EjemploAprobado;
function areaCuadrado ( : REAL ) : REAL ;
var
    resultado: REAL;
type
    vector1 = array [1..4] of INTEGER;
    datos = record
       campo1:INTEGER;
       campo2:REAL
    end;
begin
    vecto1 [  ] := 0;
    datos.campo1 := 0;
    resultado := +0.0;
    resultado := lado * lado;
    areaCuadrado := resultado;
(*Vemos que falta 'end;'*)
var
    medida :REAL;
    valor1, valor2: INTEGER;
begin
    medida := $4.A;
end.
```

Y su salida correspondiente es:



Error sintáctico encontrado en Línea 2 Columna: 25
Error 1.- La lista de variables esta mal definida en los parametros

Error sintáctico encontrado en Línea 12 Columna: 15
Error 2.- Error en la expresion de acceso en el identificador vecto1

Error sintáctico encontrado en Línea 18 Columna: 1
Error 3.- Se esperaba cierre end del begin (en linea: 11)

4.2.- Ejemplo 2

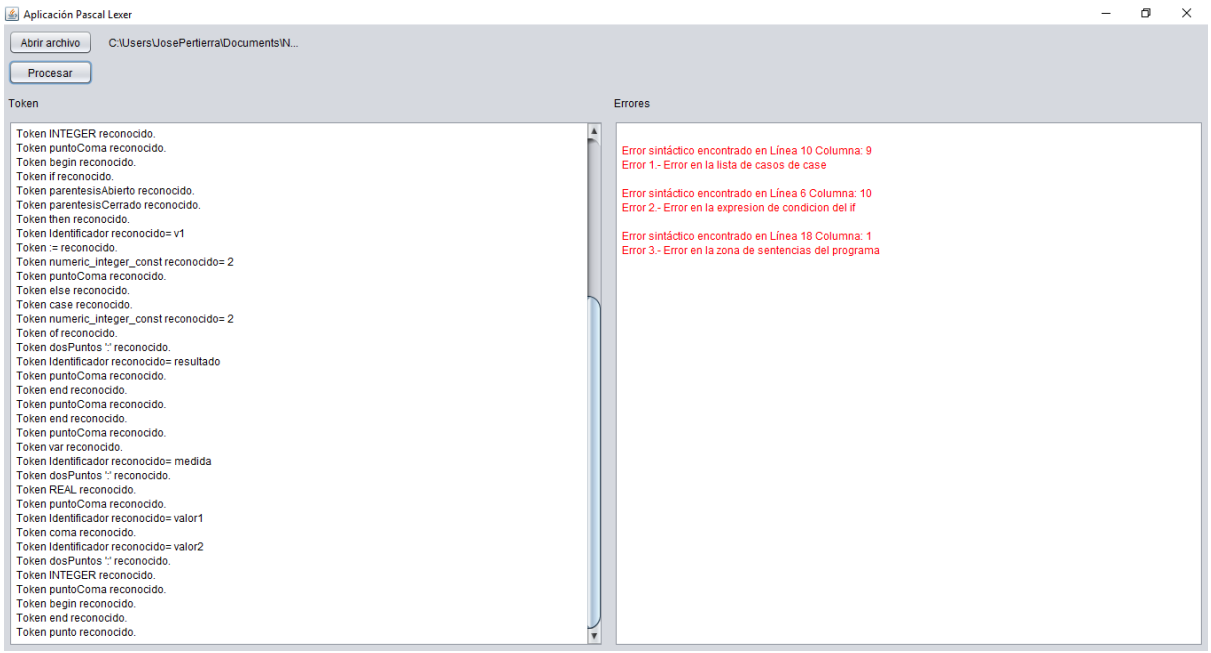
-Veremos fallos en la declaración de sentencias de un case, en la declaración de if y en un cuerpo de sentencias vacío. El código viene dado por:

```
program EjemploAprobado;
procedure intercambio ( v1, v2: INTEGER );
var
    aux: INTEGER;
begin
    if ( ) then
       v1 := 2;
    else
    case 2 of
       : resultado;
    end;
```

```
end;
var
    medida :REAL;
    valor1, valor2: INTEGER;
begin


end.
```

Y su salida correspondiente es:



Error sintáctico encontrado en Línea 10 Columna: 9
Error 1.- Error en la lista de casos de case

Error sintáctico encontrado en Línea 6 Columna: 10
Error 2.- Error en la expresion de condicion del if

Error sintáctico encontrado en Línea 18 Columna: 1
Error 3.- Error en la zona de sentencias del programa

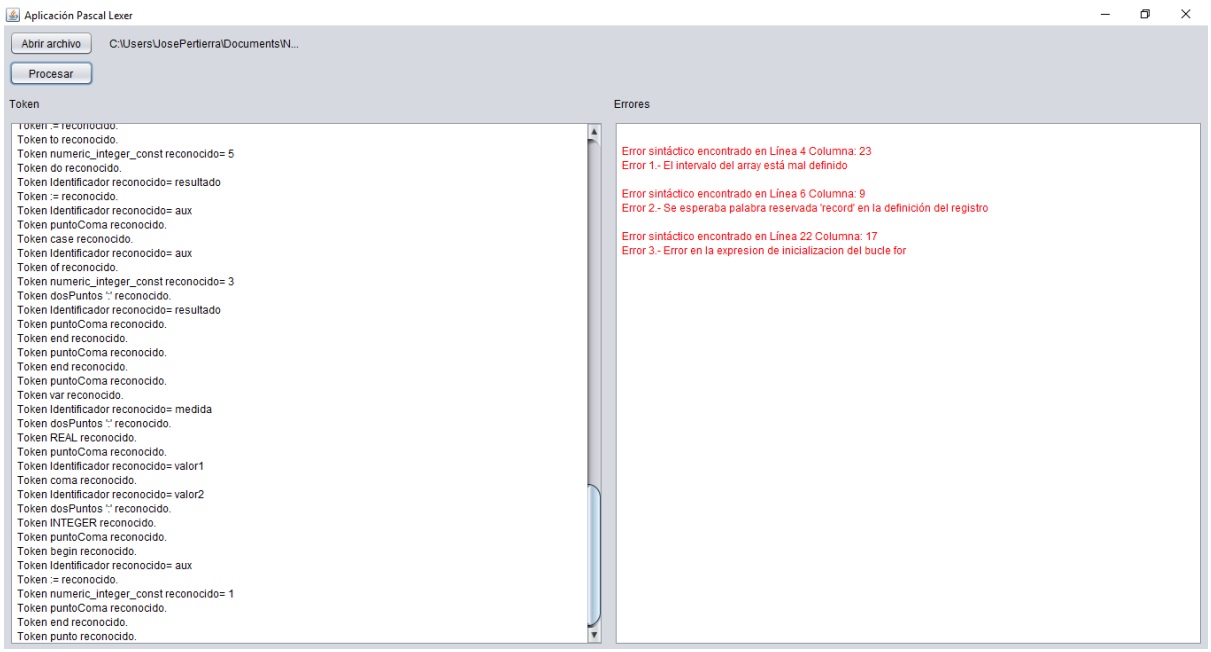
4.3.- Ejemplo 3

-Veremos fallos en la declaración un array, declaración de un tipo registro y en la declaración de un bucle for . El código viene dado por:

```
program EjemploAprobado;
```

```
procedure intercambio ( v1, v2: INTEGER );
type
    vector1 = array [1] of INTEGER;
    datos =
       campo1:INTEGER;
       campo2:REAL
    end;
var
    aux: INTEGER;
begin
    aux := 0;
    aux := v1;
    v1 := v2;
    v2 := aux;
    if ( v1>v2) then
       v1 := 2;
    else
       v1 := 3;
    while (2>3) do
       resultado := lado + lado;
    for aux :=  to 5 do
       resultado := aux;
    case aux of
       3: resultado;
    end;
end;
var
    medida :REAL;
    valor1, valor2: INTEGER;
begin
    aux:=1;
end.
```

Y su salida correspondiente es:



Error sintáctico encontrado en Línea 4 Columna: 23
Error 1.- El intervalo del array está mal definido

Error sintáctico encontrado en Línea 6 Columna: 9
Error 2.- Se esperaba palabra reservada 'record' en la definición del registro

Error sintáctico encontrado en Línea 22 Columna: 17
Error 3.- Error en la expresion de inicializacion del bucle for

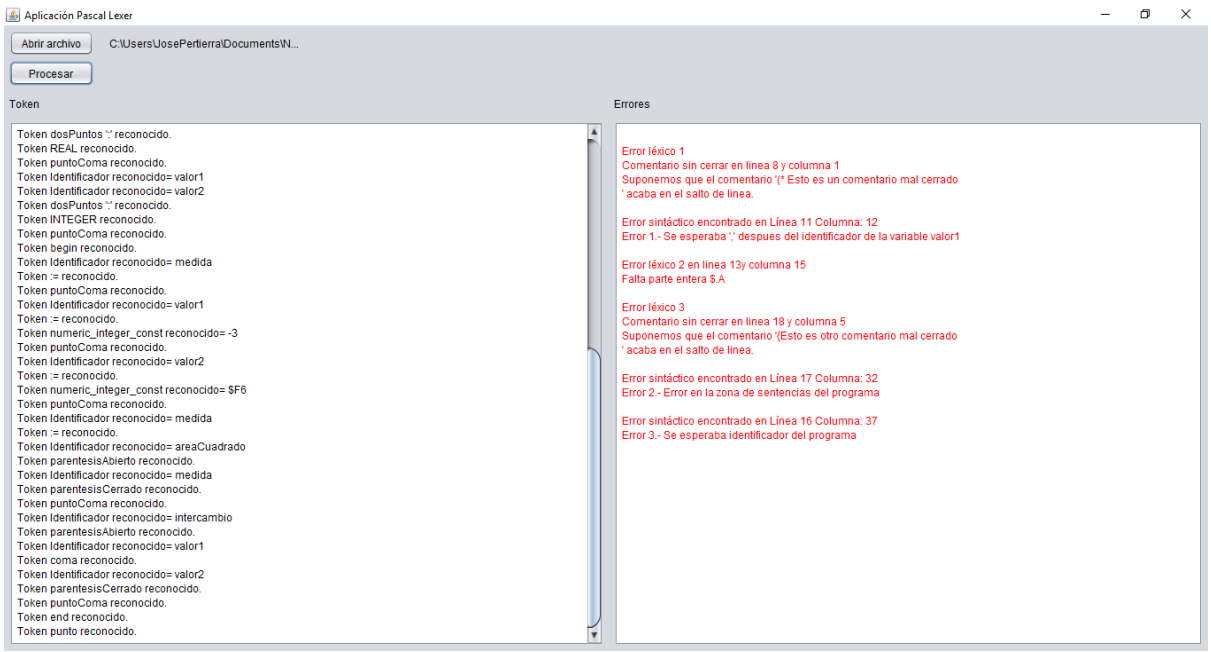
4.4.- Ejemplo 4

-Veremos fallos en la declaración de comentarios (*...*) y {...}, lista de declaración de variables, expresión de valor Hexadecimal que lleva a un error léxico y por tanto sintáctico en la asignación y en el cuerpo del programa, y en la falta de identificador de programa. El código viene dado por:

```
program ;
procedure intercambio ( v1, v2: INTEGER );
var
    aux: INTEGER;
begin
    aux := 0;
end;
(* Esto es un comentario mal cerrado
```

```
var
    medida :REAL;
    valor1 valor2: INTEGER;
begin
    medida := $.A;
    valor1 := -3;
    valor2 := $F6;
    medida := areaCuadrado( medida );
    intercambio(valor1, valor2);
    {Esto es otro comentario mal cerrado
end.
```

Y su salida correspondiente es:



Error léxico 1
Comentario sin cerrar en linea 8 y columna 1
Suponemos que el comentario '(* Esto es un comentario mal cerrado
' acaba en el salto de linea.

Error sintáctico encontrado en Línea 11 Columna: 12
Error 1.- Se esperaba ',' despues del identificador de la variable valor1

Error léxico 2 en linea 13y columna 15
Falta parte entera $.A

Error léxico 3
Comentario sin cerrar en linea 18 y columna 5
Suponemos que el comentario '{Esto es otro comentario mal cerrado
' acaba en el salto de linea.

Error sintáctico encontrado en Línea 17 Columna: 32
Error 2.- Error en la zona de sentencias del programa

Error sintáctico encontrado en Línea 16 Columna: 37
Error 3.- Se esperaba identificador del programa