Universidad de La Laguna

E. T. S. DE INGENIERÍA INFORMÁTICA
Departamento de E.I.O. y Computación
Compiladores

PRÁCTICA 2: Diseño e implementación de un analizador léxico para *Pascal-*

Factor de ponderación [0-10]: 8

2.3. Objetivo

La práctica consiste en escribir en Python un analizador léxico que reconozca los tokens del lenguaje [1] *Pascal-* (este lenguaje es un subconjunto de Pascal). En la práctica se pretende determinar los tokens del lenguaje y manipular tablas de símbolos.

2.4. Requisito de codificación

Cada fichero de código fuente de los que se utilicen en la práctica ha de llevar comentarios de cabecera. También es imprescindible que cada función lleve comentarios de cabecera con una breve indicación de la operación que implementa.

2.5. Descripción

El analizador léxico ha de reconocer las palabras reservadas que aparecen en el Cuadro 2.1.

También ha de reconocer los símbolos que aparecen en el Cuadro 2.2 Ha de reconocer números enteros e identificadores (en este aspecto, *Pascal-* no presenta diferencia alguna con respecto a Pascal estándar). Si se introduce un símbolo que no se corresponde con ninguno de los anteriores se ha de devolver el token TOKEN_ERROR. El Listado 2.1 contiene la lista de todos los componentes léxicos de *Pascal-*.

[1] Hansen, Brinch. “On Pascal Compilers”. Prentice/Hall. 1985

and	**array**	**begin**	**const**	**div**
do	**else**	**end**	**if**	**mod**
not	**of**	**or**	**procedure**	**program**
record	**then**	**type**	**var**	**while**

Cuadro 2.1: Palabras reservadas

+	-	*	<	=
>	≤	<>	≤	:=
(	)	[	]	,
.	:	;	..	

Cuadro 2.2: Signos especiales

```
AND, ARRAY, ASTERISK, BECOMES, BEGIN,
COLON, COMMA, CONST, DIV, DO, DOUBLEDOT,
ELSE, END, ENDTEXT, EQUAL, GREATER,
ID, IF, LEFTBRACKET, LEFTPARENTHESIS,
LESS, MINUS, MOD, NOT, NOTEQUAL,
NOTGREATER, NOTLESS, NUMERAL, OF, OR,
PERIOD, PLUS, PROCEDURE, PROGRAM,
RECORD, RIGHTBRACKET, RIGHTPARENTHESIS,
SEMICOLON, THEN, TYPE, TOKEN_ERROR, VAR,
WHILE
```

Listado 2.1: Tokens

El analizador léxico ignorará los comentarios.

También llevará la cuenta del número de línea y número de columna (posición dentro de la línea) del programa fuente que está siendo analizado. Esta información será útil a la hora de generar posibles mensajes de error.

Los tokens `ID` y `NUMERAL` tendrán un atributo asociado: su valor.

Se considerará un identificador toda aquella palabra que comience con una letra y vaya seguida de letras, dígitos o caracteres “underscore” (guión bajo). El lexema asociado cada identificador se almacenará en la tabla de símbolos. A cada identificador se le asociará como atributo un puntero que indica la posición correspondiente en la tabla de símbolos.

La tabla de símbolos se implementará como una tabla Hash.

Se distinguirá entre palabras reservadas e identificadores utilizando una resolución implícita.

Para un uso futuro en la implementación del compilador, a cada identificador se asociará un atributo (index) que será un número entero positivo diferente para cada uno de los identificadores que aparezcan en el código fuente. Los identificadores estándar y su número (index) asociado aparecen en el Cuadro 2.3.

integer	1
boolean	2
false	3
true	4
read	5
write	6

Cuadro 2.3: Identificadores estándar

El resto de identificadores que aparezcan en el programa fuente tendrán valores de `index` diferentes para cada identificador, comenzando en 7.

Utilizar programas *Pascal-* implementados a propósito para comprobar el analizador léxico diseñado. Aparte de comprobar el analizador en "condiciones normales", ponerlo también a prueba con programas erróneos, para comprobar un funcionamiento adecuado del analizador léxico.