ANALIZADOR LÉXICO TINY(1)

Óscar Morujo Fernández

Sofía Capmany Fernández

G16

CLASES LÉXICAS

Clases léxicas univaluadas:

&&	Separa la sección de declaraciones de la de instrucciones
;	Semicolon
+	Operador más
-	Operador menos
*****	Operador por
/	Operador entre
=	Operador asignación
<	Operador comparación menor que
>	Operador comparación mayor que
<=	Operador comparación menor o igual que
>=	Operador comparación mayor o igual que
==	Operador comparación igual que
!=	Operador comparación distinto que
)	Apertura de paréntesis
(	Cierre de paréntesis
%	Operador modulo entero división
{	Curly Brackets open
}	Curly Brackets close
[	Square Brackets open
]	Square Brackets close
.	Operador acceso a campo de variable (tipo record)
->	Operador acceso a campo de variable (tipo pointer)

,	Separador parámetros función
&	Dirección de la variable (paso por variable)

<u>También tenemos las siguientes palabras reservadas:</u>

int	Tipo básico números enteros
real	Tipo básico números reales
bool	Tipo básico booleanos
true	Expresión básica booleana
false	Expresión básica booleana
and	Operador lógico conjunción
or	Operador lógico disyunción
not	Operador lógico negación
string	Tipo básico literales cadena
null	Expresión básica de tipo null
proc	
if	
then	
else	
endif	
while	
do	
endwhile	
call	Instrucción invocación a procedimiento
record	Tipo registro

array	Tipo array
of	Instrucción declaración tipo
pointer	Tipo puntero
new	Instrucción de reserva de memoria
delete	Instrucción de liberación de memoria
read	Instrucción de lectura
write	Instrucción de escritura
nl	Instrucción de nueva línea
var	Declaración de variables
type	Palabra reservada para declaraciónes de tipo

Clases léxicas multivaluadas:

identificadores	Comienzan por una letra, seguida de una secuencia de cero o más letras, dígitos o "_".
Literales enteros	Un literal entero, comienza por '+' o '–' opcionalmente (signo del número) y va seguido del número entero (secuencia de uno o más dígitos sin 0 no significativos a la izquierda).
Literales reales	Un literal real comienza con una parte entera, seguida de una parte decimal o una parte exponencial, o bien una parte decimal seguida de una parte exponencial.
Literales cadena	Comienzan con comilla doble ("), seguida de una secuencia de 0 o más caracteres distintos de comilla doble ("), retroceso (\b), retorno de carro (\r), y salto de línea (\n), seguida de comilla doble(").

ESPECIFICACIÓN DEL LÉXICO DEL LENGUAJE MEDIANTE E.R

Definiciones auxiliares:

- Letra = **[a-z, A-Z]**

- DigitoPositivo = **[1-9]**

- Digito = **DigitoPositivo | 0**
  Dígito = [1-9] | 0
- ParteEnt = **0 | DigPositivo Dígito***
  ParteEnt = 0 | [1-9]([1-9] | 0)*
- Pdec = **. Dígito* DigPositivo**
  Pdec = .([1-9] | 0)*[1-9])
- Pexp = **(E | e) (\- | \+)? ParteEnt**
  Pexp = (E | e)(\+ | \-)? 0 | [1-9]([1-9] | 0)*

- AllowedCharacters **= [^"\r\b\n]**

Definiciones de cadenas ignorables:

- IgnoredStrings = **[\t\r\b\n]**
- Comentario = **# [^\n]***

Definiciones léxicas :

#Multivaluadas

- Identificadores = **Letra (Letra | dígito | _)***
  Identificadores = [a-z, A-Z] ([a-z, A-Z] | (= [1-9] | 0)| _)
- Entero = **(\+ | \-)? ParteEnt**
  Entero = (\+ | \-)? 0 | [1-9]([1-9] | 0)*
- Real = **Entero (Pdec | Pexp | Pdec Pexp)**
  Real = (\+ | \-)? 0 | [1-9]([1-9] | 0)*
  ((.([1-9] | 0)*[1-9])) |
  (= (E | e)(\+ | \-)? 0 | [1-9]([1-9] | 0)*) |
  (.([1-9] | 0)*[1-9])(E | e)(\+ | \-)? 0 | [1-9]([1-9] | 0)*))
- Str = **"allowedCharacters* "**
  Str = [^"\r\b\n]*

#Univaluadas

- SEP_DI= **&&**
- MAS**= +**
- MENOS= **-**
- POR= *****
- DIV= **/**
- BLT = **<**
- BGT = **>**
- BLE = **<=**
- BGE = **>=**
- BEQ = **==**
- BNE = **!=**
- IGUAL = **=**
- PAP = **(**
- PCIERRE = **)**
- SEMICOLON = **;**
- MOD= **%**
- CAP= **[**
- CCIERRE= **]**
- LLAP= **{**
- LLCIERRE= **}**
- PUNTO= **.**
- FLECHA= **->**
- COMA = **,**
- AMP= **&**

#Palabras reservadas

- R_INT= **int**
- R_REAL= **real**
- R_BOOL= **bool**
- R_STRING= **string**
- R_TRUE= **true**
- R_FALSE= **false**
- R_AND= **and**
- R_OR= **or**
- R_NOT= **not**
- R_NULL= **null**
- R_PROC**= proc**
- R_IF= **if**
- R_THEN= **then**
- R_ELSE= **else**

- R_ENDIF= **endif**
- R_WHILE= **while**
- R_DO**= do**
- R_ENDWHILE**= endwhile**
- R_CALL**= call**
- R_RECORD= **record**
- R_ARRAY= **array**
- R_OF= **of**
- R_POINTER= **pointer**
- R_NEW**= new**
- R_DELETE**= delete**
- R_READ= **read**
- R_WRITE**= write**
- R_NL= **nl**
- R_VAR= **var**
- R_TYPE= **type**