**PRÁCTICA 1: Diseño del lenguaje**

**Grupo Martes 4 - Lenguaje BAAAB**

Andrés Arco López, Alberto Estepa Fernández, Rafael Ruz Gómez, Carlos Sánchez Muñoz

1. **Sintaxis inspirada en un lenguaje de programación:** (B) Lenguaje C
2. **Palabras reservadas en:** (A) Castellano
3. **Estructura de datos considerada como tipo elemental:** (A) Listas
4. **Subprogramas:** (A) Funciones
5. **Estructuras de control adicional:** (B) for (usando para ello la sintaxis de Pascal)

**Introducción, descripción del lenguaje asignado**

Es un lenguaje basado en C en el que las palabras reservadas van a estar escritas en Castellano, la estructura de datos elemental será las listas, los subprogramas estarán constituidos básicamente por funciones y por último las estructuras de control adicional serán los “for“ en la que utilizaremos para ello la sintaxis de Pascal.

**Descripción formal de la sintaxis del lenguaje usando BNF**

<programa> ::= <cabecera\_programa> <bloque>

<bloque> ::= <inicio\_bloque>   
 <declar\_variables\_locales>   
 <declar\_subprogs>  
 <sentencias>   
 <fin\_bloque>

<declar\_subprogs> ::= <declar\_subprogs> <declar\_subprog>  
 |

<declar\_subprog> ::= <cabecera\_subprograma> <bloque>

<declar\_variables\_locales> ::= <marca\_ini\_declar\_variables>  
 <variables\_locales>   
 <marca\_fin\_declar\_variables>  
 |

<marca\_ini\_declar\_variables> ::= VAR

<marca\_fin\_declar\_variables> ::= FINVAR

<cabecera\_programa> ::= PRINCIPAL

<inicio\_bloque> ::= {

<fin\_bloque> ::= }

<variables\_locales> ::= <variables\_locales><cuerpo\_declar\_variables> | <cuerpo\_declar\_variables>

<cuerpo\_declar\_variables> ::= <tipo><lista\_identificadores> “;”

<lista\_identificadores> ::= <lista\_identificadores> “,” <identificador>

| <identificador>

<cabecera\_subprog> ::= <tipo> <identificador>

“(“ <lista\_parametros> “)”

<sentencias> ::= <sentencias> <sentencia>  
 | <sentencia>

<sentencia> ::= <bloque>  
 | <sentencia\_asignacion>  
 | <sentencia\_if>  
 | <sentencia\_while>

| <sentencia\_entrada>

| <sentencia\_salida>

| <sentencia\_return>

| <sentencia\_for>

<sentencia\_asignacion> ::= <identificador> = <expresion> “;”

<sentencia\_if> ::= SI “(“ <expresion> “)” <sentencia>

[SINO <sentencia> ]

<sentencia\_while> ::= MIENTRAS “(“ <expresion> “)” <sentencia>

<sentencia\_entrada> ::= <nomb\_entrada> <lista\_variables> “;”

<sentencia\_salida> ::= <nomb\_salida> <lista\_exp\_cadena> “;”

<sentencia\_return> ::= DEVOLVER <expresion> “;”

<sentencia\_for> ::= PARA <identificador> “:=” <expresion> HASTA

<expresion> PASO <expresion> <sentencia>

<lista\_parametros> ::= <lista\_parametros>”,” <tipo> <identificador>

| <tipo> <identificador>

<lista\_exp\_cadena> ::= <lista\_exp\_cadena> “,” <exp\_cad>

| <exp\_cad>

<exp\_cad> ::= <expresion>

| <cadena>

<nomb\_entrada> ::= LEER

<lista\_variables> ::= <lista\_variables> “,” <identificador>

| <identificador>

<nomb\_salida> ::= IMPRIMIR

<lista\_expresiones> ::= <lista\_expresiones> “,” <expresion>

| <expresion>

<cadena> ::= ““”<lista\_ascii>“””

<lista\_ascii> ::= <lista\_ascii> <ASCII>

| <ASCII>

<expresion> ::= ¨(¨ <expresion> ¨)¨

| <op\_unario> <expresion>

| <expresion> <op\_binario> <expresion>

| <identificador>

| <constante>

| <funcion>

| <expresion> “++” <expresion> “@” <expresion>

<funcion> := <identificador> “(“<lista\_expresiones>”)”

<op\_unario> ::= ! | + | - | # | ?

<op\_binario> ::= ==

| !=

| <

| >

| <=

| >=

| &&

| ||

| ^

| +

| -

| \*

| /

| %

| \*\*

| @

| --

<constante> ::= <const\_entero>

| <const\_real>

| <const\_caracter>

| <const\_booleano>

| <const\_lista>

<const\_entero> ::= <const\_entero><digito>

| <digito>

<const\_real> ::= <const\_entero> “.” <const\_entero>

<const\_booleano> ::= VERDADERO

| FALSO

<const\_caracter> ::= “ ‘ ”<ASCII>“ ’ ”

<const\_lista> ::= <const\_lista\_entero>

| <const\_lista\_real>

| <const\_lista\_caracter>

| <const\_lista\_booleano>

<const\_lista\_entero> ::= “[“ <lista\_enteros> ”]”

<lista\_enteros> ::= <lista\_enteros> “,”<signo> <const\_entero>

| <signo> <const\_entero>

<const\_lista\_real> ::= “[“ <lista\_real> ”]”

<lista\_real> ::= <lista\_real> “,”<signo> <const\_real>

| <signo> <const\_real>

<const\_lista\_caracter> ::= “[“ <lista\_caracter> ”]”

<lista\_caracter> ::= <lista\_caracter> “,” <const\_caracter>

| <const\_caracter>

<const\_lista\_booleano> ::= “[“ <lista\_booleano> ”]”

<lista\_booleano> ::= <lista\_booleano> “,” <const\_booleano>

| <const\_booleano>

<tipo> ::= <tipo\_base>

| LISTA DE <tipo\_base>

<tipo\_base> ::= ENTERO

| REAL

| BOOLEANO

| CARACTER

<identificador> ::= <alfanumerico>

| <alfanumerico> <identificador>

<alfanumerico> ::= <letra>

| <digito>

<letra> ::= a | b | … | z | A | B | … | Z

<digito> ::= 0 | 1 | … | 9

<ASCII> ::= “cualquier carácter ASCII”

<signo> ::= + | - |

**Definición de la semántica en lenguaje natural**

Nuestro lenguaje funciona como una versión traducida de C al español. Explicamos

ahora las distintas sentencias que utilizamos, si bien no es necesario explicar su

funcionamiento:

* Variables : Se traducirá con la forma : int = entero, float = real, bool = booleano; char

= caracter.

* Sentencia if : Se traducirá con la forma SI “(” <expresión> “)
* Sentencia while: Se traducirá con la forma MIENTRAS “(” <expresión> “)”;
* Sentencia cout : Se traducirá con la forma IMPRIMIR >>
* Sentencia cin : Se traducirá con la forma LEER <<
* Sentencia FOR : Se traducirá de la forma PARA <identificador> “:=” <const\_entero> HASTA <const\_entero> PASO <const\_entero> <sentencias>
* Las funciones serán declaradas del mismo modo que en C, con una cabecera compuesta por el tipo de dato a devolver y un cuerpo entre llaves. El programa principal responde al nombre principal().

**Identificación de los tokens con máximo nivel de abstracción**

|  |  |  |  |
| --- | --- | --- | --- |
| **TOKENS** | **IDENTIFICADOR** | **ATRIBUTOS** | **PATRÓN** |
| CABECERA | 257 |  | PRINCIPAL |
| IDENTIFICADOR | 258 |  | [a-Z]+[a-zA-Z\_0-9]\* |
| OPBINARIO | 259 | **0**:== **1**:< **2**:> **3**:<= **4**:>= **5**:&& **6**:|| **7**:\* **8**:/ **9**:!=  **10:** \*\* **11:**^ **12**:% **13**:-- | ==, <, >, <=, >=, &&, ||, \*, /, !=,\*\*,^, %, -- |
| OPTERNARIO\_1 | 260 |  | ++ |
| OPTERNARIO\_2 | 261 |  | @ |
| SIGNO | 262 | **0**:+ **1**:- | +, - |
| ENTERO | 263 | [0-9]+ | [0-9]+ |
| REAL | 264 | [0-9]\* "." [0-9]+ | [0-9]\* "." [0-9]+ |
| TIPO | 265 | **0**:ENTERO **1**:REAL **2**:BOOLEANO **3**:CARACTER | ENTERO, REAL, BOOLEANO, CARACTER |
| BUCLE | 266 |  | PARA |
| DESDE | 267 |  | DESDE |
| HASTA | 268 |  | HASTA |
| PASO | 269 |  | PASO |
| CONDICION | 270 |  | SI |
| SUBCONDICION | 271 |  | SINO |
| CICLO | 272 |  | MIENTRAS |
| ASIGNACION | 273 |  | = |
| ENTRADA | 274 |  | LEER |
| SALIDA | 275 |  | IMPRIMIR |
| RETURN | 276 |  | DEVOLVER |
| INIBLOQUE | 277 |  | { |
| FINBLOQUE | 278 |  | } |
| INIVARIABLES | 279 |  | VAR |
| FINVARIABLES | 280 |  | FINVAR |
| CONSTANTE  \_BOOLEANA | 281 | **0**:VERDADERO **1**:FALSO | VERDADERO,  FALSO |
| CADENA | 282 |  | \"[^\"]\*\" |
| CONSTANTE \_CARACTER | 283 |  | \’ [^\’] \’ |
| PARIZQ | 284 |  | ( |
| PARDER | 285 |  | ) |
| COMA | 286 |  | , |
| FINLINEA | 287 |  | ; |
| DOSPUNTOSIGUAL | 288 |  | := |
| ABRIRCORCHETE | 289 |  | [ |
| CERRARCORCHETE | 290 |  | ] |
| OPUNARIO | 291 | **0**:!, **1**:#, **2**:? | !, #, ? |
| LISTA\_DE | 292 |  | LISTA DE |

**PRÁCTICA 2: Especificación e**

**Implementación del Analizador de Léxico**

**Especificación Lex**

Es