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