Práctica 2:

Especificación e Implementación del Analizador de Léxico

Alejandro Cárdenas Barranco

Jesús García León

Manuel Moya Martín-Castaño

Juan Manuel Rodríguez Gómez

Procesadores de Lenguajes (Grupo A2, Equipo 7)

Curso 2023 – 2024

UNIVERSIDAD DE GRANADA

Lenguaje asignado: BBAAD

Nuestro lenguaje asignado tiene las siguientes características:

- Sintaxis inspirada en el lenguaje de programación **C**.
- Palabras reservadas en **inglés**.
- Estructura de datos considerada como tipo elemental: **Listas**.
- Subprogramas: **Funciones**.
- Estructura de control adicional: **do-until**.

Descripción formal de la sintaxis del lenguaje usando BNF

```
# Descripción de elementos básicos
<letra> ::= a | b | c |...| z | A | B | C | ... | Z
<digito> ::= 0 | 1 | 2 |...| 9
<identificador> ::= <letra> <letra_o_digito>
<letra_o_digito> ::= <letra> <letra_o_digito>
                   | <digito> <letra_o_digito>
                   |
<caracter_ascii> ::= *Todos los carácteres de la tabla ASCII excepto " *
<cadena_ascii> ::= <caracter_ascii> <cadena_ascii>
<cadena> ::= " <cadena_ascii> "

# Tipos de variables
<tipo_variable_elemental> ::= int
                            | float
                            | bool
                            | char
<tipo_variable_lista> ::= list_of <tipo_variable_elemental>
<tipo_variable> ::= <tipo_variable_elemental>
                  | <tipo_variable_lista>

# Programa principal
<programa> ::= main <bloque>

# Bloque
<bloque> ::= {
                <declaracion_variables_locales>
                <declaracion_subprogramas>
                <sentencias>
             }

# Variables locales
<declaracion_variables_locales> ::= local {
                                      <variables_locales>
                                      }
                                  |
<variables_locales> ::= <variables_locales> <cuerpo_variables_locales>
                      | <cuerpo_variables_locales>
<cuerpo_variables_locales> ::= <tipo_variable> <lista_variables> ;
<lista_variables> ::= <identificador> , <lista_variables>
```

```
                    | <identificador>

# Subprograma (funciones)
<declaracion_subprogramas> ::= <declaracion_subprog> <declaracion_subprogramas>
                             |
<declaracion_subprog> ::= <cabecera_subprog> <bloque>
<cabecera_subprog> ::= <tipo_variable> <identificador> ( <parametros> )
<parametros> ::= <parametro> , <parametros>
               | <parametro>
               |
<parametro> ::= <tipo_variable> <identificador>

# Sentencias
<sentencias> ::= <sentencias> <sentencia>
               | <sentencia>
               |
<sentencia> ::= <bloque>
              | <sentencia_asignacion> ;
              | <sentencia_if>
              | <sentencia_while>
              | <sentencia_entrada> ;
              | <sentencia_salida> ;
              | <sentencia_return> ;
              | <sentencia_lista>
              | <sentencia_do_until> ;

# Asignación de valores
<sentencia_asignacion> ::= <identificador> = <expresion>

# Condicional
<sentencia_if> ::= if( <expresion> ) <sentencia> <sentencia_else>
<sentencia_else> ::= else <sentencia>
                   |

# Bucle while
<sentencia_while> ::= while ( <expresion> ) <sentencia>

# Entrada
<sentencia_entrada> ::= cin <lista_variables>

# Salida
<sentencia_salida> ::= cout <lista_expresiones_cadenas>
<lista_expresiones_cadenas> ::= <lista_expresiones_cadenas> , <expresion_cadena>
                              | <expresion_cadena>
<expresion_cadena> ::= <expresion>
                     | <cadena>

# Return
<sentencia_return> ::= return <expresion>

# Listas
<sentencia_lista> ::= <expresion> <shift>
                    | $ <expresion>
<shift> ::= >>
          | <<
```

```
# Do-until
<sentencia_do_until> ::= do <sentencia> until ( <expresion> )

# Expresion
<expresion> ::= ( <expresion> )
              | <operador_unario> <expresion>
              | <expresion> <operador_binario> <expresion>
              | <expresion> ++ <expresion> @ <expresion>
              | <identificador>
              | <constante>
              | <llamada_funcion>
<lista_expresiones> ::= <expresion>
                      | <expresion>, <lista_expresiones>
<llamada_funcion> ::= <identificador> ( <lista_expresiones> )
                    | <identificador> ( )

# Operador Unario
<operador_unario> ::= !
                    | ?
                    | #
                    | +
                    | -

# Operador Binario
<operador_binario> ::= +
                     | -
                     | /
                     | *
                     | @
                     | --
                     | !=
                     | ==
                     | <=
                     | >
                     | <
                     | <=
                     | **
                     | %

# Constantes
<constante> ::= <entero>
              | <caracter>
              | <boolean>
              | <lista>
              | <real>
<entero> ::= <digito> <entero>
           | <digito>
<caracter> ::= ' <caracter_ascii> '
<boolean> ::= true
            | false
<lista> ::= [ <lista_expresiones> ]
          | [ ]
<real> ::= <entero> . <entero>
```

Tabla de Tokens

Nombre	Expresión regular	Código del token	Atributos
ID	"[a-z|A-Z][a-z|A-Z|0-9]*"	**257**	
ASIGN	“=”	**258**	
IF	“if”	**259**	
ELSE	“else”	**260**	
LPAR	“(”	**261**	
RPAR	“)”	**262**	
OPEBIN	“*” | “/” | “%” | “**” |”==” | “!=” | “&&” | “||” | “<” | “>” | "<=" | ">=" | “--” |	**263**	0: * 1: / 2: % 3: ** 4: == 5: != 6: && 7: || 8: < 9: > 10: <= 11: >= 12: --
INIBLOQUE	“{“	**264**	
FINBLOQUE	“}”	**265**	
LOCAL	“local”	**266**	
TIPO	“int” | “float” | “char” | “bool” | "list_of"" "+"int" | "list_of"" "+"float" | "list_of"" "+"char" | "list_of"" "+"bool"	**267**	0:int 1:float 2:char 3:bool 4:list_of int 5:list_of float 6:list_of char 7:list_of bool
PYC	“;”	**268**	
CIN	“cin”	**269**	
COUT	“cout”	**270**	
CADENA	“[^"]*”	**271**	
RETURN	“return”	**272**	
OPUNARIO	“!” | “#” | “?”	**273**	0: ! 1: # 2: ?
AT	“@”	**274**	
MASMAS	“++”	**275**	
BINYUN	“+” | “-”	**276**	0: + 1: -
CONSTANTS	"([0-9]+)" | "(([0-9]+)(.[0-9]+)?)" | " '[^']' " | ("true" | "false")	**277**	0:int 1:float 2:char 3:bool
COMA	“,”	**278**	

Nombre	Expresión regular	Código del token	Atributos
MAIN	“main”	**279**	
DO	“do”	**280**	
UNTIL	“until”	**281**	
WHILE	“while”	**282**	
LCOR	“[“	**283**	
RCOR	“]”	**284**	
DOLLAR	“$”	**285**	
SHIFTL	“<<”	**286**	
SHIFTR	“>>”	**287**	