Compiladores

Gramática en notación BNF empleada para la construcción del Parser.

Trabajo a Cargo de : Nery A. Riquelme Granada y René Azcurra Irigoitia.

<fuente> -> <programa> | €
<programa> -> <instruccion> | <instruccion> | <comentario>
<comentario> -> // <comentario-textual>
<comentario-textual> -> PRINTABLECHAR | PRINTABLECHAR <comentario-textual> | €
<bloque-declaracion> -> VAR <lista-declaracion>
<lista-declaracion> -> <expresion-declaracion> | <expresion-declaracion> , <lista-declaracion>
<expresion-declaracion> -> <variable-declaracion> | <variable-inicializacion>
<variable-declaracion> -> <ident> | <ident> [<numero>]
<variable-inicializacion> -> <asignacion>
<asignacion> -> <variable-declaracion> = <expresion>
<instruccion> -> <bloque-declaracion> ;
| <bloque-sentencia> ;
| <bloque-declaracion> ; <instruccion>
| <bloque-sentencia> ; <instruccion>
<bloque-sentencia> -> <asignacion>|<sentencia>
<sentencia> -> <sentencia-if> | <sentencia-for> | <llamada-funcion>
<sentencia-if> -> IF <expresion> THEN <instruccion> END IF
|IF <expresion> THEN <instruccion> ELSE <instruccion> END IF
<sentencia-for> -> FOR <ctrl-init> TO <expresion> STEP <expresion> DO <instruccion> END FOR
<llamada-funcion> -> <ident> (<lista-argumentos>)
<lista-argumentos> -> <expresion> | <expresion> , <lista-argumentos> | €
<ctrl-init> -> <ident> = <expresion>
<ident> -> <inicio-palabra> | <inicio-palabra> <palabra>
<inicio-palabra> -> <alpha> | _
<palabra> -> <alnum> | _ | <alnum> <palabra> | _ <palabra>
<alnum> -> <alpha> | <digit>
<alpha> -> A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
<digit> -> 0|1|2|3|4|5|6|7|8|9
<expresion> -> <arith-expr> | <bool-expr> | <relat-expr>
<arith-expr> -> <term> | <term> + <arith-expr> | <term> - <arith-expr>
<term> -> <factor> | <factor> * <term> | <factor> / <term>
<factor> -> <numero> | <ident> | <llamada-funcion> | (<expresion>)
<relat-expr> -> <expresion> < <expresion> | <expresion> > <expresion> | <expresion> <= <expresion> | <expresion> >=
<expresion> | <expresion> == <expresion> | <expresion> <> <expresion>
<bool-expr> -> <bool-member> | <bool-member> OR <bool-expr> | <bool-member> AND <bool-expr>
<bool-member> -> <bool-term> | NOT <bool-term> | NOT <bool-member>
<bool-term> -> <bool> | <expresion>
<bool> -> TRUE | FALSE