TALLER 3: LENGUAJE BÁSICO 1

INTEGRANTES

VICTOR MANUEL MARÍN DUQUE
2015556071

DIEGO DANILO DELGADO
201556272

ANDRÉS CAMILO PULGARÍN
201556017

DOCENTE

CARLOS ANDRÉS DELGADO

UNIVERSIDAD DEL VALLE SEDE TULUÁ
FACULTAD DE INGENIERÍA Y COMPUTACIÓN
SIMULACIÓN COMPUTACIONAL
TULUÁ - VALLE
FEBRERO 2020

ESPECIFICACIÓN GRAMATICAL DEL LENGUAJE 1 EN FORMA DE BACKUS-NAUR (BNF)

La siguiente sintaxis se basa en el lenguaje de programación JavaScript; sin embargo, algunas características se modifican debido a los problemas por la izquierda (*LL*) de la librería *SLLGEN*.

<program> ::= <expresion>
un-programa (expresion)

<expresion> ::= <numero>
numero-exp (numero)

::= <identificador>
identificador-exp (identificador)

::= <flotante>
flotante-exp (flotante)

::= <”0x” 0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F
{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F}*>
hexadecimal-exp (hexadecimal)

::= <”0o” 0|1|2|3|4|5|6|7 {0|1|2|3|4|5|6|7}*>
octal-exp (octal)

::= var ({identificador = <expresion>}*(,))
definicion-exp (identificadores valores)

::= if (<expresion>) { {<expresion>}*(;) } else { {<expresion>}*(;) }
condicional-exp (condicion sentencia-verdad sentencia-falsa)

::= length (<expresion>)
longitud-exp (cadena)

::= concat (<expresion> <expresion>)
concatenacion-exp (cadena1 cadena2)

::= function <identificador> ({<identificador>}*(,)) { {<expresion>}*(;) }
procedimiento-exp (nombre-funcion parametros cuerpo)

::= call <identificador> ({<expresion>}*(,))
invocacion-proc-exp (nombre-funcion argumentos)

::= function-rec <identificador> ({<identificador>}*(,)) { {<expresion>}*(;) }
procedimiento-rec-exp (nombre-funcion parametros cuerpo)

::= call-rec <identificador> ({<expresion>}*(,))
invocacion-proc-rec-exp (nombre-funcion argumentos)

::= for (<expresion>; <expresion>; <expresion>) { {<expresion>}*(";") }
iteracion-exp (inicial-exp, condicion-for, incrementador, cuerpo)

::= <expresion> <primitiva-aritmetica> <expresion>
primitiva-aritmetica-exp (componente1 operando componente2)

::= <expresion> <primitiva-booleana> <expresion>
primitiva-booleana-exp (componente1 operando componente2)

<primitiva> ::= + | - | * | % | / | ++ | --

<primitiva-booleana> ::= < | > | <= | >= | == | != | && | || | ! | true | false