Diego Fernando Montaño Pérez A01282875

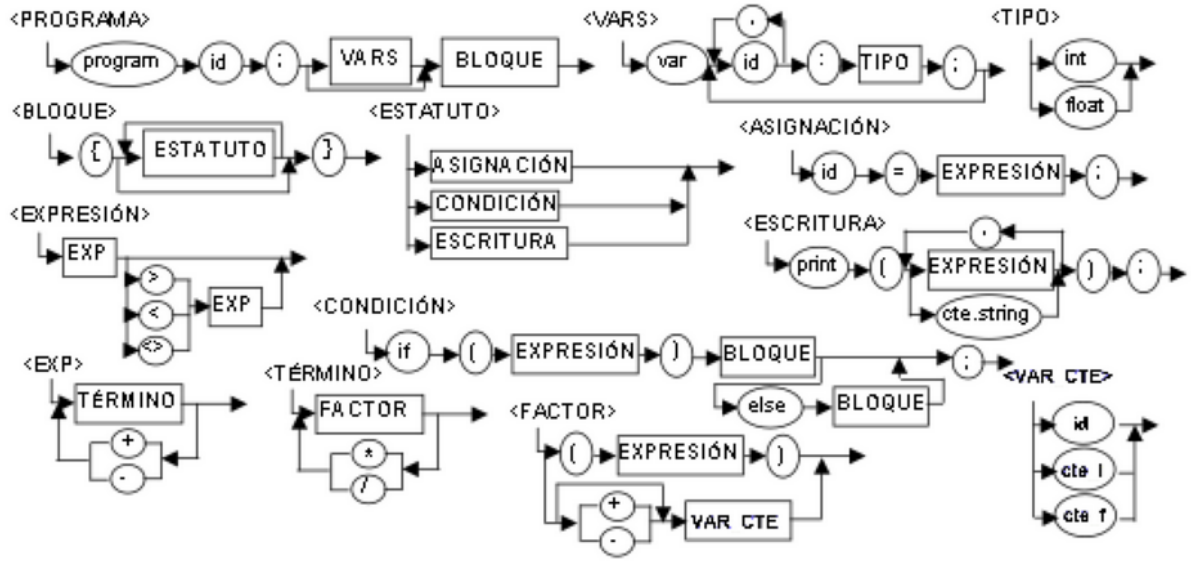
Tarea 3.1

A continuación se describe un pequeño lenguaje de programación. Se te pide que implementes 2 versiones de los analizadores de léxico y sintaxis (scanner y parser) necesarios para reconocerlo. Debes utilizar la herramienta que planeas usar para tu proyecto. El programa a ejecutar leerá los elementos de un archivo fuente (texto), revisará el léxico y la sintaxis y, en caso de existir error, desplegará en pantalla el mensaje “apropiado”.

LENGUAJE: LittleDuck 2020

A continuación se describen las reglas sintácticas del lenguaje LittleDuck 2020. Considerando los tokens que se usan dentro de estas reglas, diseña las reglas de los elementos de léxico (tokens) que sean necesarias (a tu criterio). Considera que esta tarea te dará la pauta para desarrollar las etapas de léxico y sintaxis de tu proyecto.

Entregar las expresiones regulares y la gramática en un documento.



Gramática

1. **PROGRAMA**
    a. program → id
    b. id → ;
    c. ; → VARS | BLOQUE
    d. BLOQUE → ε

2. **VARS**
    a. var → id
    b. id → . | :
    c. . → id
    d. : → TIPO
    e. TIPO → ;
    f. : → id | ε
3. **TIPO**
    a. int → ε
    b. float → ε
4. **BLOQUE**
    a. { → ESTATUTO | }
    b. ESTATUTO → } | {
    c. } → ε
5. **ESTATUTO**
    a. ASIGNACION → ε
    b. CONDICION → ε
    c. ESCRITURA → ε
6. **EXPRESIÓN**
    a. EXP → > | < | <> | ε
    b. > → EXP
    c. < → EXP
    d. <> → EXP
7. **ASIGNACION**
    a. id → =
    b. = → EXPRESION
    c. EXPRESION → ;
    d. ; → ε
8. **CONDICION**
    a. if → {
    b. { → EXPRESION
    c. EXPRESION → }
    d. } → BLOQUE
    e. BLOQUE → else | ;
    f. else → BLOQUE
    g. ; → ε
9. **ESCRITURA**

a. print → (
b. (→ EXPRESION | cte.string
c. EXPRESION → , |)
d. cte.string →) | ,
e. , → EXPRESION | cte.string
f.) → ;
g. ; → ε

10. **EXP**
    a. TERMINO → + | - | ε
    b. + → TERMINO
    c. - → TERMINO
11. **TERMINO**
    a. FACTOR → * | / | ε
    b. * → FACTOR
    c. / → FACTOR
12. **FACTOR**
    a. (→ EXPRESION
    b. EXPRESION →)
    c. + → VAR CTE
    d. - → VAR CTE
    e. VAR CTE → ε
    f.) → ε
13. **VAR CTE**
    a. id → ε
    b. cte I → ε
    c. cte f → ε

Expresiones Regulares

Start program	program
letter	[a-z] or [A-Z]
digit	[0-9]
sign	[+ | - | * | /]
id	letter(letter | digit)*

Diego Fernando Montaño Pérez A01282875

var	letter(letter)*
equal to	=
greater than	>
smaller than	<
different	<>
left parenthesis	(
right parenthesis	)
left curley brace	{
right curley brace	}
conditional if	if
conditional else	else
print result	print
cte id	“letter(letter | digit)*”
cte i	digit*
cte f	digit(digit)* . digit(digit)*