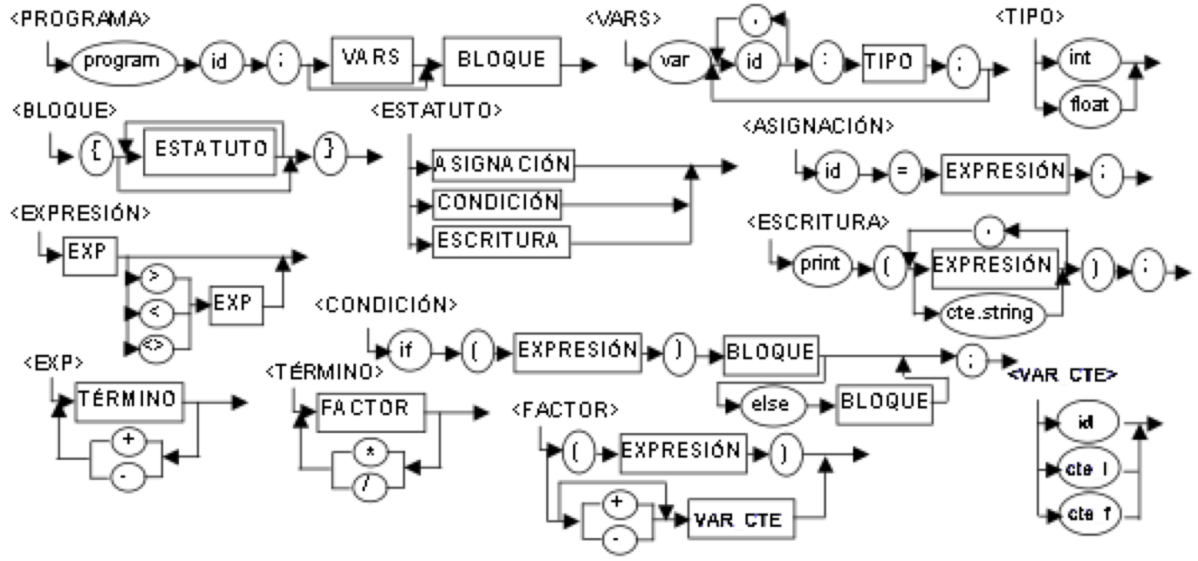
Diego García Rodriguez del Campo
A01700203
18/02/2022
Diseño de Compiladores

Expresiones Regulares y Gramática

“Little Duck 2020”



Gramática

PROGRAMA = program ID ; PROGRAMA2

PROGRAMA2 = BLOQUE | VARS BLOQUE

VARS = var VARS2

VARS2 = ID , VARS2 | ID : TIPO ; | ID : TIPO ; VARS2

TIPO = int | float

BLOQUE = { ESTATUTOS } | {}

ESTATUTOS = ESTATUTO ESTATUTOS | ESTATUTO

ESTATUTO = ASIGNACIÓN | CONDICIÓN | ESCRITURA

ASIGNACIÓN = ID = EXPRESIÓN ;

ESCRITURA = print (ESCRITURA2) ;

ESCRITURA2 = EXPRESION | EXPRESION , ESCRITURA2 | CTESTR | CTESTR , ESCRITURA2

EXPRESIÓN = EXP | EXP CMP EXP

CMP = > | < | <>
CONDICIÓN = if (EXPRESIÓN) BLOQUE ; | if (EXPRESIÓN) BLOQUE else BLOQUE ;
EXP = TÉRMINO | TÉRMINO + EXP | TÉRMINO - EXP
TÉRMINO = FACTOR | FACTOR * TÉRMINO | FACTOR / TÉRMINO
FACTOR = (EXP) | FACTOR2
FACTOR2 = VARCTE | + VARCTE | - VARCTE
VARCTE = ID | CTEI | CTEF

Notas

Las reglas siguen el lenguaje descrito en la foto mostrada al inicio del documento. Algunas fueron expandidas para ser más fáciles de comprender y poder aceptar los diferentes caminos disponibles para cada término
En FACTOR, se cambió el término entre paréntesis por EXP en vez de EXPRESIÓN, pues así ya tiene un sentido, porque si no era posible hacer cosas como (id > 3) > (id < 3), que, a mi parecer no tiene sentido y era un error, y que lo realmente esperado era poder tener EXP anhiladas

Expresiones Regulares

ID = [A-Za-z]\w*
CTEI = [+-]?(\d+)
CTEF = [+-]?(\d+)(\.\d+)?
CTESTR = \"[^\"]*\"

Notas

Para las expresiones regulares, los ids se permiten de todo, con que comience con una letra. Puede tener guion bajo, números y letras después
Para los enteros, cualquier número con un signo opcional antes
Para los flotantes, cualquier número, con punto opcional y dígitos después. Si hay punto, debe de haber al menos un dígito después.
Para un string, se admite cualquier carácter menos comillas, encerrado entre comillas (“”)