



**Tecnológico
de Monterrey**

Propuesta de Compiladores

Patito++



La Propuesta propuesta
29/03/2020

Oscar Lerma A01380817

Cesar Buenfil A01207499

Objetivo del Lenguaje

Cumplir con las expectativas y funciones descritas en el proyecto en parejas: [Lenguaje Patito++](#)

Requerimientos de Lenguaje

I) Elementos Básicos (tokens, palabras reservadas)

Palabras reservadas

PROGRAMA → 'Programa'	VAR → 'var'
PRINCIPAL → 'principal'	INT → 'int'
FLOAT → 'float'	CHAR → 'char'
FUNCION → 'funcion'	Sí → 'si'
ENTONCES → 'entonces'	LEE → 'lee'
SINO → 'sino'	REGRESA → 'regresa'
MIENTRAS → 'mientras'	ESCRIBE → 'escribe'
HAZ → 'haz'	DESDE → 'desde'
HASTA → 'hasta'	HACER → 'hacer'
STRING → 'string'	NULL → 'null'
VOID → 'void'	

Tokens

COMMENT → '%%.*\n'	CTE_I → '(\+ -)?[0-9]+'
L_PAREN → '\('	CTE_F → '(\+ -)?[0-9]+(\.[0-9]+)?f'
R_PAREN → '\)'	CTE_STRING → '".*"'
L_BRACKET → '{'	CTE_CH → '\[A-Za-z]\''
R_BRACKET → '}'	DOT → '\.'
AND → '&&'	PLUS → '\+'
OR → ' '	MINUS → '\-'
TRANS_ARR → '↓'	INV_ARR → '↑'
COMPARE → '=='	DOTS → '...'

DIV \rightarrow '/'

MULT \rightarrow '*

MORE \rightarrow '>'

LESS \rightarrow '<'

NOT \rightarrow '!'

MOREEQUAL \rightarrow '>='

DOTCOMA \rightarrow ','

LSTAPLE \rightarrow '['

ID \rightarrow '[A-Za-z]([A-Za-z][0-9_]*)'

EQUAL \rightarrow '='

DIFFERENT \rightarrow '!='

LESSEQUAL \rightarrow '<='

MOD \rightarrow '%'

DET_ARR \rightarrow '\$'

COMA \rightarrow ','

RSTAPLE \rightarrow ']'

II) Descripción de funciones

- escribe(**cte_string** , **EXPRESION**, **id**, ...) : Retorna a pantalla (standard output) el parámetro, (en caso de expresión, todo valor de retorno). En caso de múltiples parámetros, se concatenan de manera inmediata en pantalla (standard output)
- lee(**id**, ...) : La función espera input del usuario (standard input) y intenta mapear el mismo en las variable especificada como parámetro. Si existen múltiples parámetros, la función espera inputs del usuario para cada parámetro separados por '\n' (línea nueva).

III) Tipos de Datos

- Int
- Float
- Char
- String
- Arreglos
- Matrices

IV) Sistema operativo y lenguaje.

- Linux (basado en debian)
- Python 3, analizador léxico y sintáctico PLY (3.1.1)

V) Gramáticas libres de contexto.

PROGRAM \rightarrow programa id ; VARS FUNCTIONS MAIN
MAIN \rightarrow principal () VARS BLOQUE
VARS \rightarrow var VAR_AUX ϵ

VAR_AUX → TIPO IDS VAR_AUX ϵ
TIPO → int float char string
IDS → id ARRDIM ; id ARRDIM , IDS
ARRDIM → [EXPRESION] [EXPRESION] [EXPRESION] [EXPRESION , EXPRESION] ϵ
FUNCTIONS → FUNCTION FUNCTIONS ϵ
FUNCTION → funcion TIPO id (PARAM) VARS BLOQUE funcion void id (PARAM) VARS BLOQUE
PARAM → TIPO id PARENTESIS PARAM_AUX
PARAM_AUX → , PARAM ϵ
PARENTESIS → [] [] [] ϵ
BLOQUE → { ESTATUTOS }
ESTATUTOS → ESTATUTO ESTATUTOS ϵ
ESTATUTO → ASIGNACION ; FUN ; COND WRITE ; READ ; RETURN ;
ASIGNACION → id ARRDIM = EXPRESION id ARRDIM = CTE_ARR
EXPRESION → SUBEXP && SUBEXP SUBEXP SUBEXP SUBEXP
SUBEXP → EXP EXP COMPARACION EXP
COMPARACION → > < == != >= <=
EXP → TERMINO TERMINO + EXP TERMINO - EXP
TERMINO → FACTOR FACTOR * TERMINO FACTOR / TERMINO FACTOR % TERMINO
FACTOR → (EXPRESION) + CTE - CTE !CTE CTE ARROP CTE
CTE → cte_i cte_f ct_ch cte_string FUN id ARRDIM
ARROP → \$ i ?
FUN → id (FUN_AUX)
FUN_AUX → EXPRESION , FUN_AUX EXPRESION EMPTY
COND → IF FOR WHILE

IF \rightarrow si (EXPRESION) entonces BLOQUE SI_AUX si (EXPRESION) entonces COND
IF_AUX \rightarrow sino BLOQUE ϵ
WHILE \rightarrow mientras (EXPRESION) WHILE_AUX BLOQUE mientras (EXPRESION) WHILE_AUX COND
WHILE \rightarrow haz empty
FOR \rightarrow desde ASIGNACION hasta EXPRESION hacer BLOQUE desde ASIGNACION hasta EXPRESION hacer COND
WRITE \rightarrow escribe (WRITE_AUX)
WRITE_AUX \rightarrow EXPRESION WRITE_AUXSUB
WRITE_AUXSUB \rightarrow , WRITE_AUX ϵ
READ \rightarrow lee (READ_AUX)
READ_AUX \rightarrow id ARRDIM READ_AUXSUB
READ_AUXSUB \rightarrow , READ_AUX ϵ
RETURN \rightarrow regresa (EXPRESION) regresa (NULL)
CTE_ARR \rightarrow { CTE_ARR_AUX } { CTE_ARR_AUX2 }
CTE_ARR_AUX \rightarrow CTE CTE_ARR_AUXSUB
CTE_ARR_AUXSUB \rightarrow , CTE_ARR_AUX ϵ
CTE_ARR_AUX2 \rightarrow { CTE_ARR_AUX } CTE_ARR_AUX2SUB
CTE_ARR_AUX2SUB \rightarrow , CTE_ARR_AUX2 ϵ

VI) Bitácora

a) Errores

04/09/2020

Al principio tuvimos muchos errores de recursividad y que algunos símbolos no se podían llegar a ellos.

```

WARNING: Symbol 'READ_AUX' is unreachable
WARNING: Symbol 'READ_AUXSUB' is unreachable
WARNING: Symbol 'CTE_ARR' is unreachable
WARNING: Symbol 'CTE_ARR_AUX' is unreachable
WARNING: Symbol 'CTE_ARR_AUX2' is unreachable
WARNING: Symbol 'CTE_ARR_AUXSUB' is unreachable
WARNING: Symbol 'CTE_ARR_AUX2SUB' is unreachable
ERROR: Infinite recursion detected for symbol 'PROGRAM'
ERROR: Infinite recursion detected for symbol 'IF'
ERROR: Infinite recursion detected for symbol 'WHILE'
Traceback (most recent call last):
  File "PatitoPlusPlus.py", line 402, in <module>
    parser = yacc.yacc()
  File "/home/buenfo/Documents/Patitoplusplus/yacc.py", line 3432, in yacc
    raise YaccError('Unable to build parser')
yacc.YaccError: Unable to build parser
buenfo@buenfo:~/Documents/Patitoplusplus$

```

Durante el desarrollo del léxico, se resaltó la incapacidad de tener una operación anidada, es decir, no podían existir expresiones como:

A + B + C

Debido a una falla desde nuestro diagrama de sintaxis.

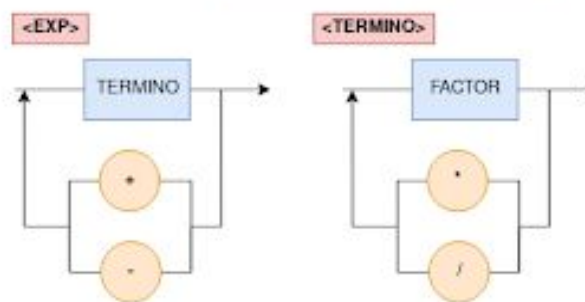


Foto 1. Antes del cambio a EXP y TERMINO (Visión Inicial)

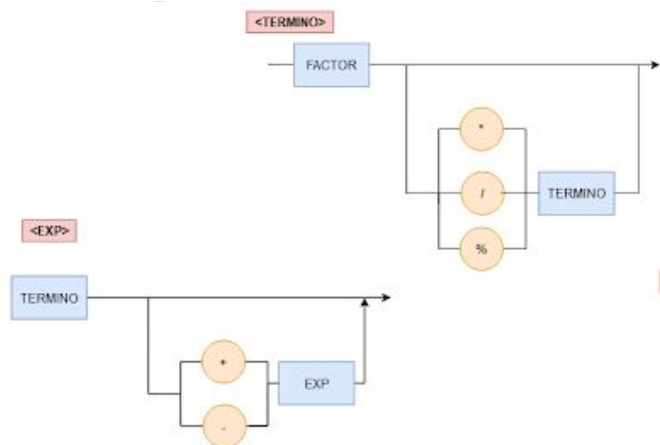
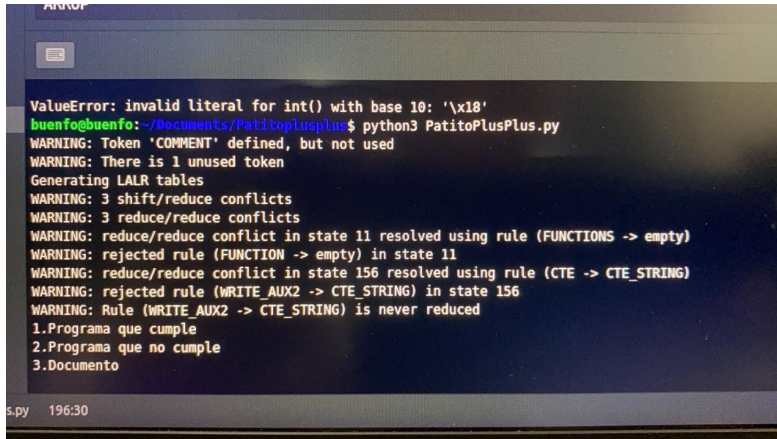


Foto 2. EXP y TERMINO después de correctivo.

Gracias a que no se tenía recursividad en el diagrama de sintaxis en operaciones binarias, era imposible tener más de dos elementos consecutivos en suma, resta, multiplicación o división.

Además se tuvo varios warnings sobre: shift/reduce y reduce/reduce. Corría bien el programa con los warnings pero queríamos tenerlo todo bien para la siguiente entrega y los resolvimos.



```
ValueError: invalid literal for int() with base 10: '\x18'
buenfo@buenfo:~/Documents/PatitoPlusPlus$ python3 PatitoPlusPlus.py
WARNING: Token 'COMMENT' defined, but not used
WARNING: There is 1 unused token
Generating LALR tables
WARNING: 3 shift/reduce conflicts
WARNING: 3 reduce/reduce conflicts
WARNING: reduce/reduce conflict in state 11 resolved using rule (FUNCTIONS -> empty)
WARNING: rejected rule (FUNCTION -> empty) in state 11
WARNING: reduce/reduce conflict in state 156 resolved using rule (CTE -> CTE_STRING)
WARNING: rejected rule (WRITE_AUX2 -> CTE_STRING) in state 156
WARNING: Rule (WRITE_AUX2 -> CTE_STRING) is never reduced
1.Programa que cumple
2.Programa que no cumple
3.Documento
```

b) Bitácora de Eventos

04/03/2020

Se inició con los elementos básicos, descripción de funciones, tipos de datos, sistema operativo y lenguaje y los diagramas de sintaxis.

04/06/2020

Se inició con la traducción del diagrama de sintaxis hacia las gramáticas libre de contexto (véase inciso IV), donde se implementó por primera vez auxiliares para elementos recursivos.

04/08/2020

Se realizó la traducción de hacia el lexer de python (lex.py), donde se pasó las expresiones regulares y palabras reservadas (véase inciso II).

04/09/2020

Se realizó la traducción de los diagramas de sintaxis hacia el lector de sintaxis de python (yacc.py), escribiendo las producciones encontradas en el mismo.

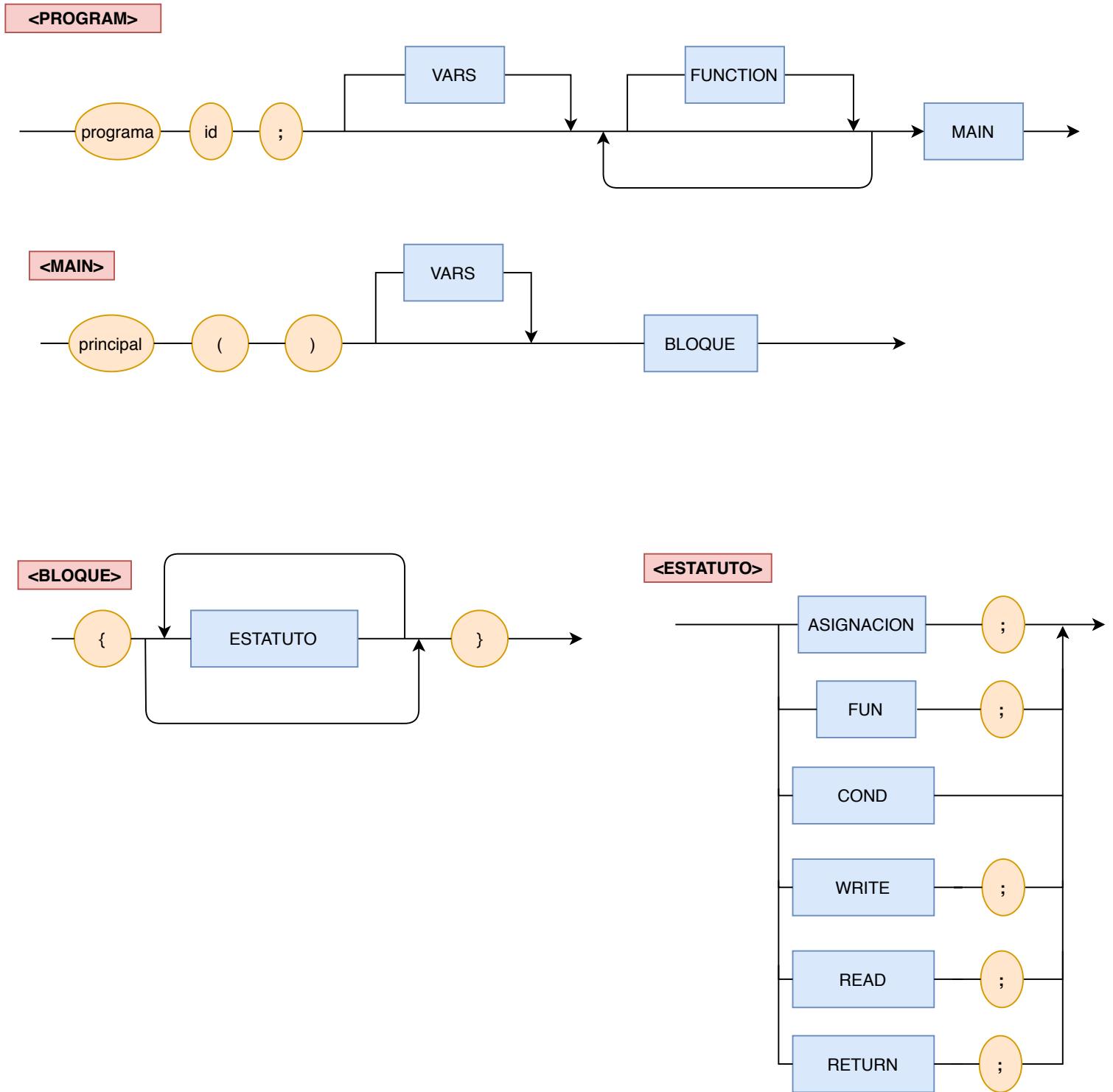
04/10/2020

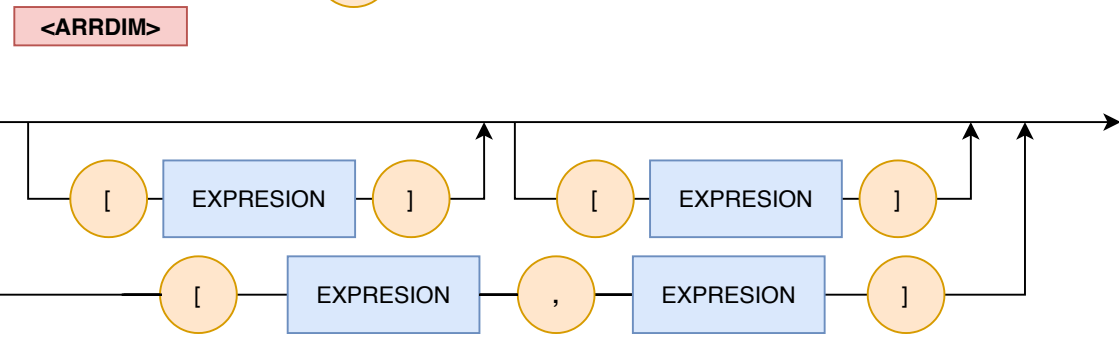
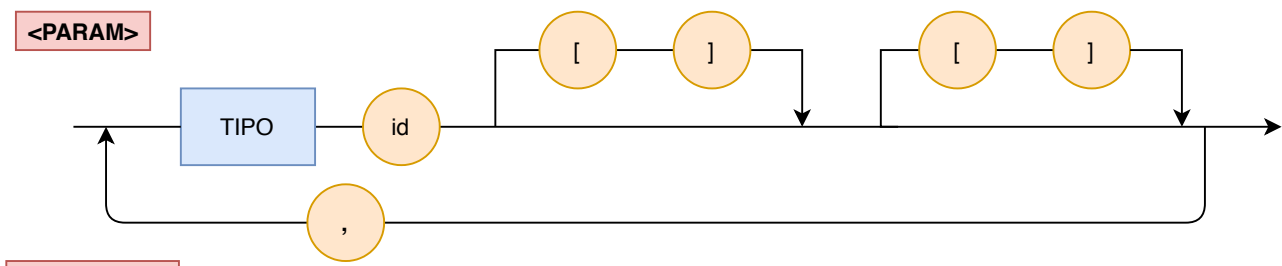
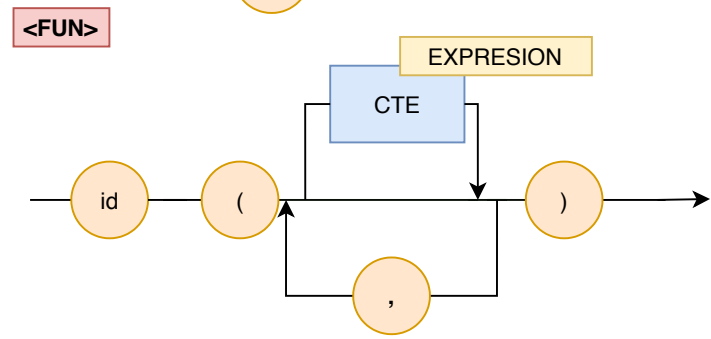
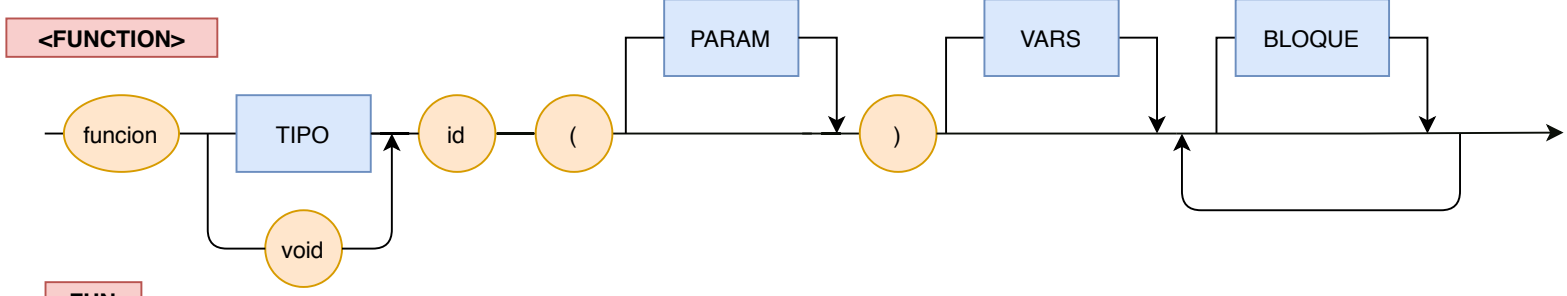
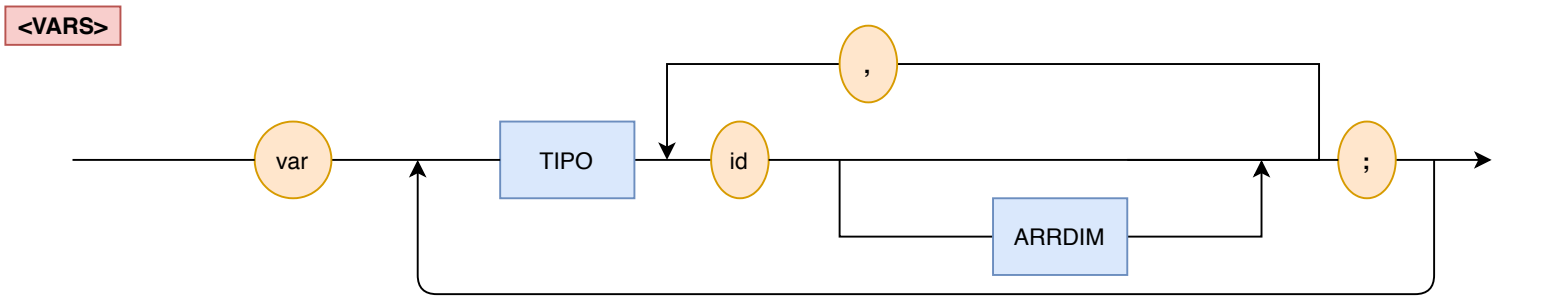
Se realizaron pruebas del código con documentos con sintaxis básico y el documento otorgado en la descripción del proyecto.

VII) Diagramas sintácticos

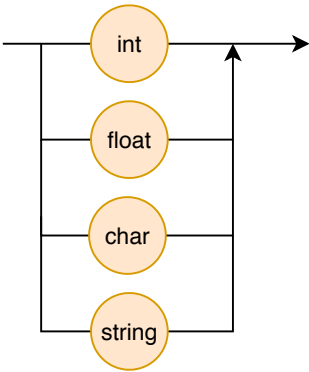
Diagramas Sintácticos

Patito++

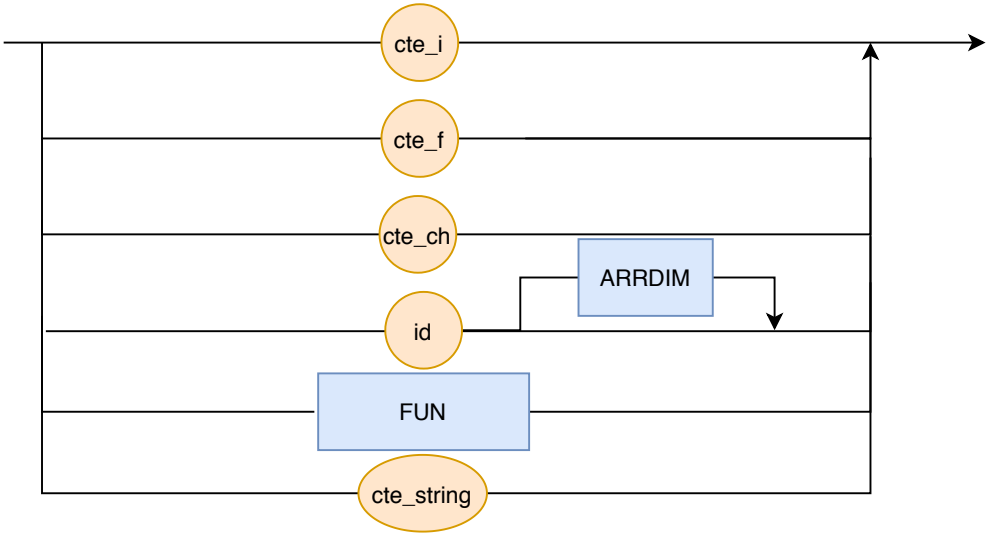




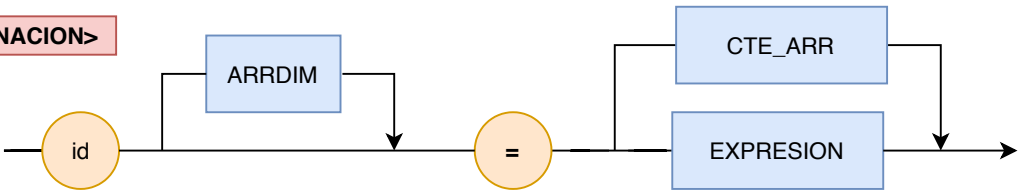
<TIPO>



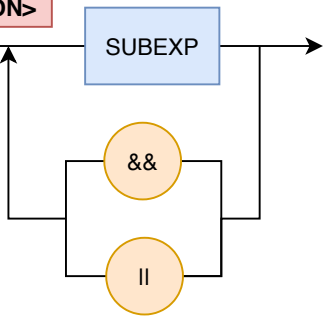
<CTE>



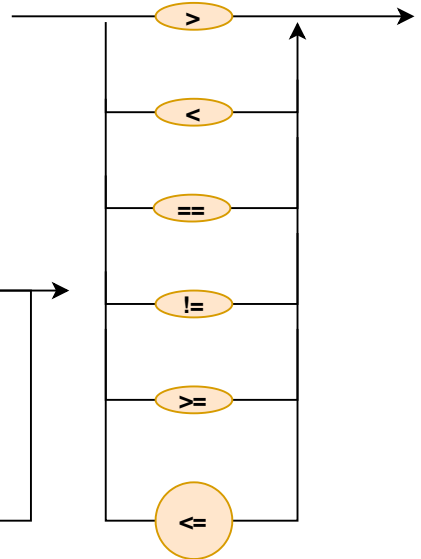
<ASIGNACION>



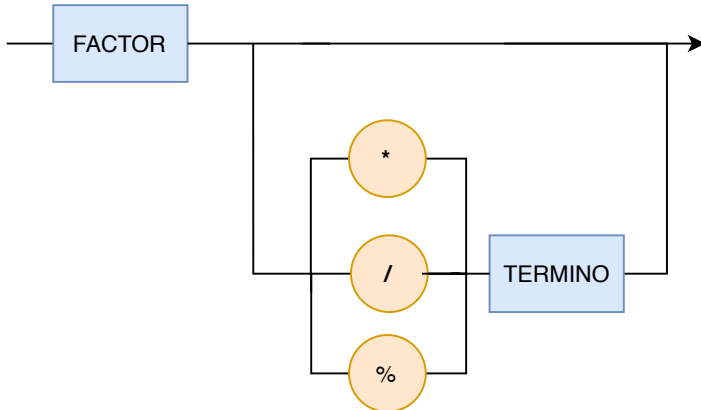
<EXPRESION>



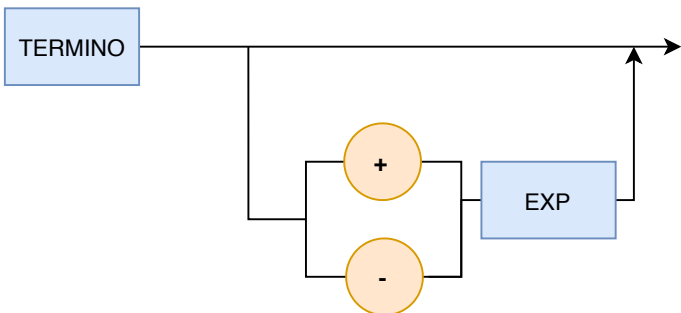
<COMPARACION>



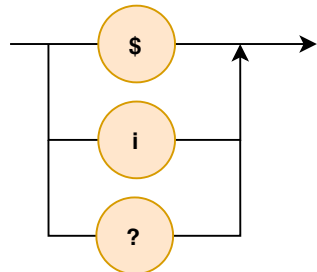
<TERMINO>



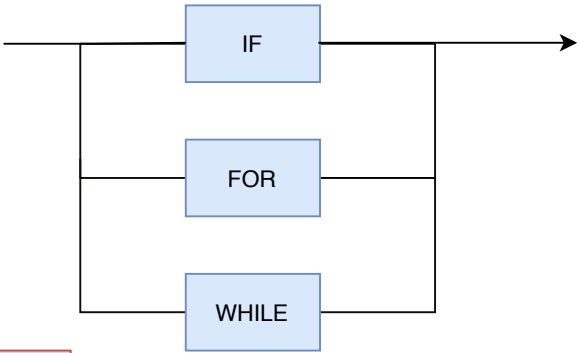
<EXP>



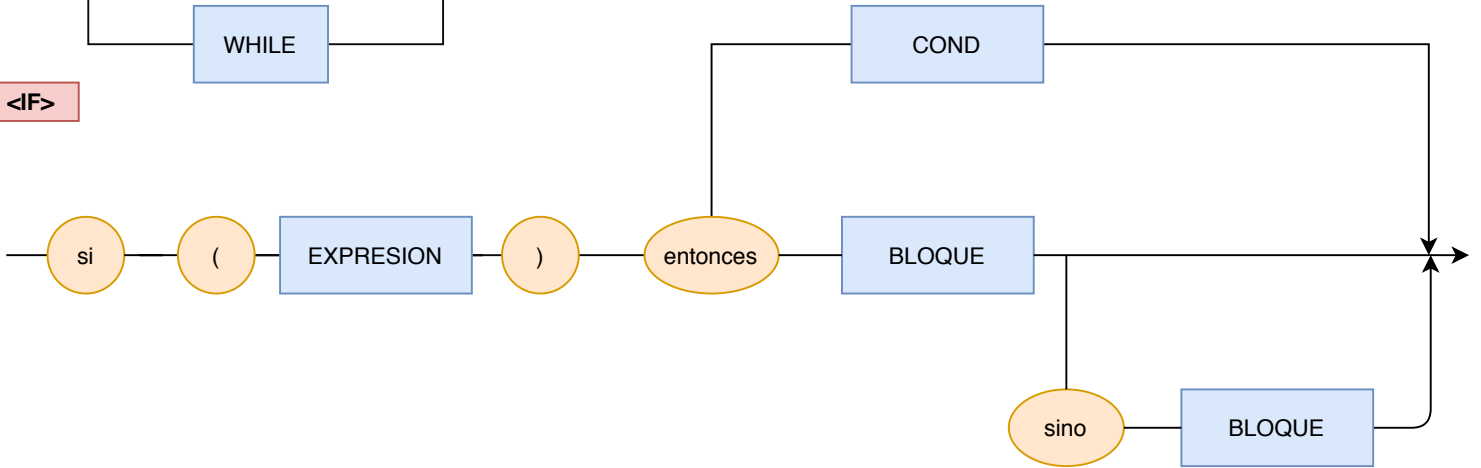
<ARROP>



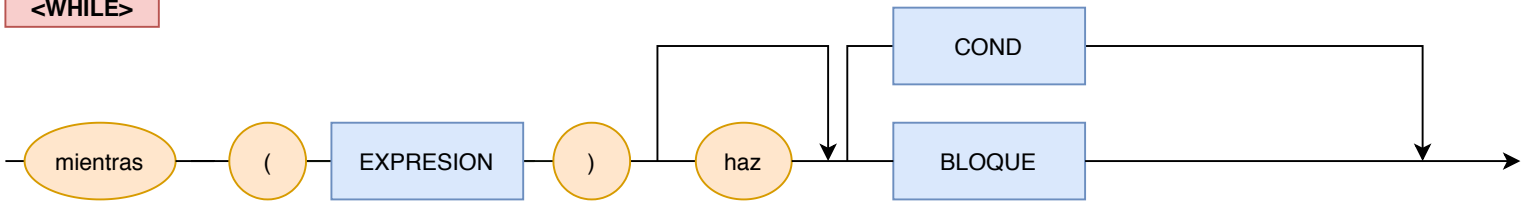
<COND>



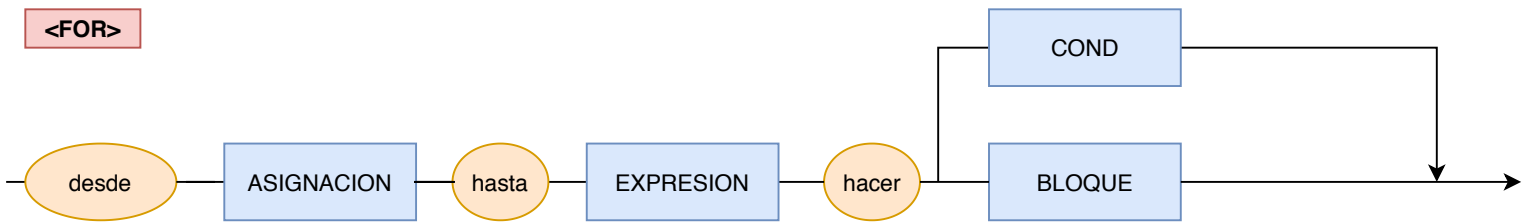
<IF>



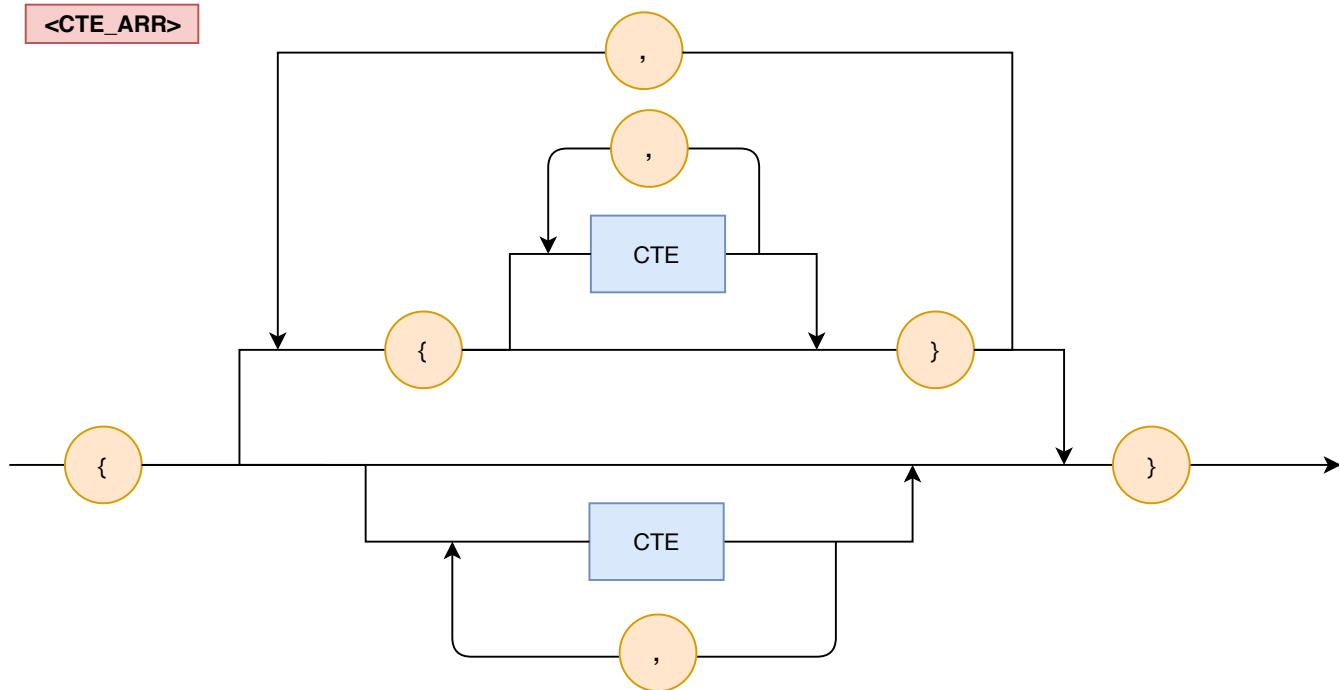
<WHILE>



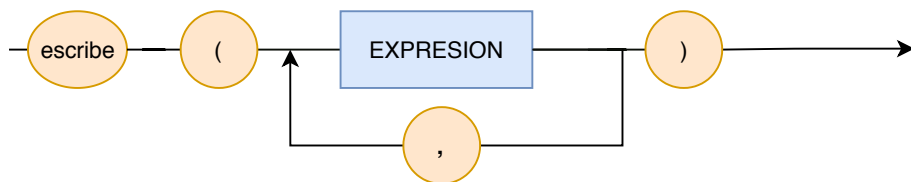
<FOR>



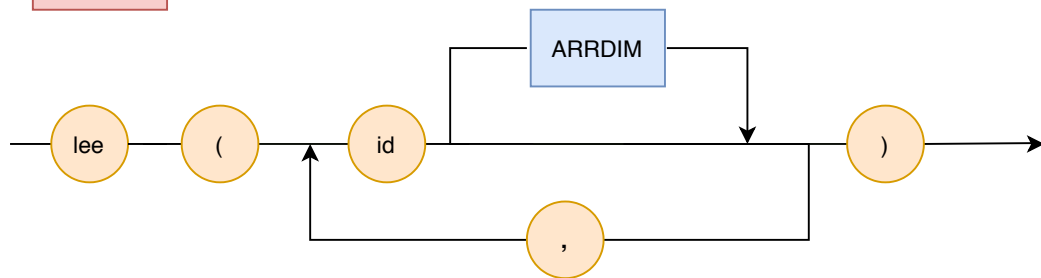
<CTE_ARR>



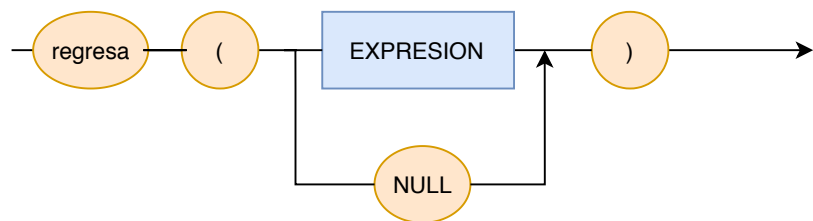
<WRITE>



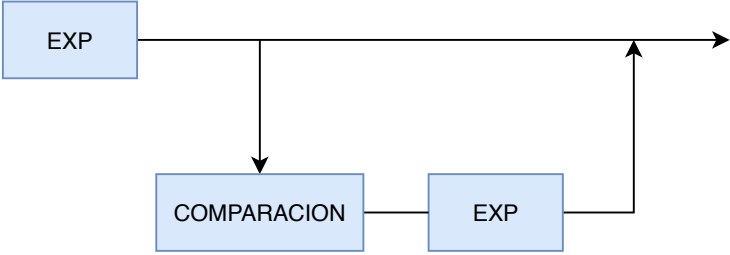
<READ>



<RETURN>



<SUBEXP>



<FACTOR>

