

**Módulo 3: Compiladores: Patito- entrega #0**

**Desarrollo de aplicaciones avanzadas de ciencias computacionales (Gpo 502)**

**Estudiante**

Bo Hyeon Cha- A01023804

**15/10/2025**

**Profesores:**

Andres Saldaña Rodriguez

Elda Guadalupe Quiroga González

Mauricio González Soto

**Tabla de contenidos**

[**Diccionario de tokens y expresión regular 3**](#_9c6821kejulx)

[**Gramática libre de contexto CFG 5**](#_8lhud5c5n3fw)

[**Github link 9**](#_ybbuwo4qek9k)

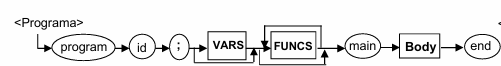
# 

# Diccionario de tokens y expresión regular

| Tokens | Expresión regular |
| --- | --- |
| Palabras clave | |
| PROGRAM | program |
| MAIN | main |
| END | end |
| VAR | var |
| INT | int |
| FLOAT | float |
| VOID | void |
| PRINT | print |
| IF | if |
| ELSE | else |
| WHILE | while |
| DO | do |
| Identificador | |
| ID | [A-Za-z][A-Za-z0-9\_]\* |
| Literales | |
| CTE\_INT | [0-9]+ |
| CTE\_FLOAT | [0-9]+\.[0-9]+ |
| CTE\_STRING | “[A-Za-z0-9\_]\*” |
| Operadores aritméticos | |
| PLUS | \+ |
| MINUS | - |
| MULT | \\* |
| DIV | / |
| Operaciones relaciones | |
| EQ | == |
| NEQ | != |
| LT | < |
| GT | > |
| LEQ | <= |
| GEQ | >= |
| Operaciones de asignación | |
| ASSIGN | = |
| Delimitadores y otros símbolos | |
| LPAREN | \( |
| RPAREN | \) |
| LBRACE | \{ |
| RBRACE | \} |
| COMMA | , |
| SEMICOLON | ; |
| COLON | : |

# 

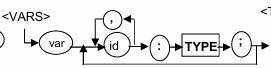
# Gramática libre de contexto CFG



<Programa> -> program ID ; VARS’ FUNCS’ main <Body> end

VARS’ -> <VARS> | ε

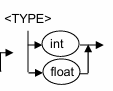
FUNCS’ -> <FUNCS> FUNCS’ | <FUNCS> |ε



<VARS> -> vars LISTAVAR

LISTAVAR -> ID’ : <TYPE> ; LISTAVAR | ID’ : <TYPE> ;

ID’ -> ID , ID’ | ID

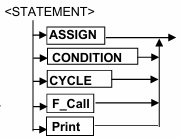


<TYPE> -> int | float

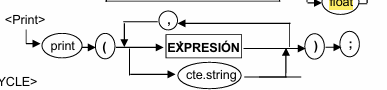


<Body> -> {STATEMENT’ }

STATEMENT’ -> <STATEMENT> STATEMENT | ε



<STATEMENT> -> <ASSIGN> | <CONDITION> | <CYCLE> | <F\_CALL> | <Print>



<Print> -> print ( EXPRESION’ ) ;

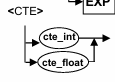
EXPRESION’ -> <EXPRESION> , EXPRESION’ | <EXPRESION> | CTE\_STRING



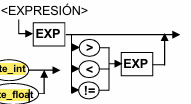
<ASSIGN> -> ID = <EXPRESION> ;



<CYCLE> -> while = <EXPRESION> ) do <BODY> ;



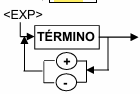
<CTE> -> CTE\_INT | CTE\_FLOAT



<EXPRESION> -> EXP’

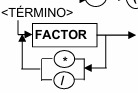
EXP’ -> GT <EXP> | LT <EXP> | GT <EXP> | NEQ <EXP> | <EXP>

*p.s. No se porque no vendria EQ, GEQ, LEQ*



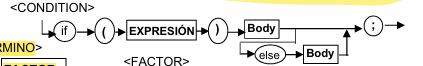
<EXP> -> TERMINO’

TERMINO’ -> <TERMINO> + TERMINO’ | <TERMINO> - TERMINO’ | <TERMINO>



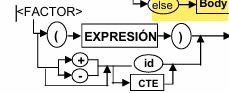
<TERMINO> -> FACTOR’

FACTOR’ -> <FACTOR> \* FACTOR’ | <FACTOR> / FACTOR’ | <FACTOR>



<CONDITION> -> if ( <EXPRESION> ) BODY’ ;

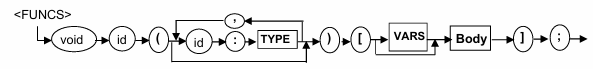
BODY’ -> <BODY> | <BODY> else BODY’



<FACTOR> -> EXPRESIONPARENTESIS | SIGNO FACTOR | ID | <CTE>

EXPRESIONPARENTESIS -> ( EXPRESION )

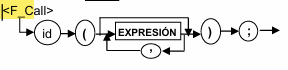
SIGNO -> + | -



<FUNCS> -> void ID ( IDYSUTIPO ) [ VARSONO ] <BODY> ] ;

IDYSUTIPO -> ID : <TYPE> , IDYSUTIPO | ID : <TYPE> | ε

VARSONO -> <VARS> | ε



<F\_Call> -> ID ( EXPCONOSINCOMA ) ;

EXPSCONOSINCOMA -> <EXPRESION> | <EXPRESION> , EXPSCONOSINCOMA | ε

# Github link

<https://github.com/Boyacto/golang_playground.git>