Axioma: “program”
Keywords: **else if int return void while**
Special symbols: **+ - * / < <= > >= == != = ; , ( ) [ ] { } /* */**
Comments: **/* … */**

Tokens:
ID = letter letter*
NUM = digit digit*
Letter = a | .. | z | A | .. | Z
Digit = 0 | … | 9

program → declaration-list
declaration-list → declaration-list declaration | declaration
declaration → var-declaration | fun-declaration
var-declaration → type-specifier **ID ;** | type-specifier **ID [NUM]** ;
type-specifier → **int** | **void**
fun-declaration → type-specifier **ID (** params **)** compound-stmt
params → param-list | **void**
param-list → param-list **,** param | param
param → type-specifier **ID** | type-specifier **ID [ ]**
compound-stmt → **{** local-declarations statement-list **}**
local-declarations → local-declarations var-declarations | empty
statement-list → statement-list statement | empty
statement → expression-stmt | compound-stmt | selection-stmt | iteration-stmt | return-stmt
expression-stmt → expression ; | ;
selection-stmt → **if (** expression **)** statement | **if (** expression **)** statement **else** statement
iteration-stmt → **while (** expression **)** statement
return-stmt → **return ;** | **return** expression **;**
expression → var **=** expression | simple-expression
var → **ID** | **ID [** expression **]**
simple-expression → additive-expression relop additive-expression | additive-expression
relop → **<=** | **<** | **>** | **>=** | **==** | **!=**
additive-expression → additive-expression addop term | term
addop → **+** | **-**
term → term mulop factor | factor
mulop → * | **/**
factor → **(** expression **)** | var | call | NUM
call → **ID (** args **)**
args → arg-list | empty
arg-list → arg-list , expression | expression

Left recursion

P → DL

—-----------------------

//EBNF
DL → D DL’
DL’ → D DL’ | eps

DL → D (D)* // { } términos repetitivos

—------------------------

D → VD | FD

—------------------------

VD → TS **ID** VD’
VD’ → **;** | **[NUM] ;**

VD → TS **ID** [**[NUM]**] **;** **//** [ ] elemento opcional

—------------------------

TS → **int** | **void** | **float**

—------------------------

FD → TS **ID (** PS **)** CST

—------------------------

PS → PL | **void**

—------------------------

PL → P PL’
PL’ → , P PL’ | eps

PL → P (**,** P)*

—------------------------

P → TS **ID** | TS **ID [ ]**

P → TS **ID []** | TS **ID**   **CHECK**

—------------------------

CST → **{** LD SL **}**

—------------------------

LD → VD LD’
LD’ → VD LD’ | eps

LD → (VD)*

—-----------------------

SL → S SL’
SL → S SL’ | eps

SL → (S)*

—-----------------------

S → EST | CST | SST | IST | RST

—-----------------------

EST → E **;** | **;**

—-----------------------

SST → **if (** E **)** S | **if (** E **)** S **else** S

SST → **if (** E **)** S [**else** S]

—-----------------------

IST → **while (** E **)** S

—-----------------------

RST → **return ;** | **return** E **;**

RST → **return** [E] **;**

—-------------------------

E → V **=** E | SE

—-------------------------

V → **ID** | **ID [** E **]**

V → **ID [** E **] | ID**

—-------------------------

SE → AE R AE | AE

—-------------------------

R → **<=** | **<** | **>** | **>=** | **==** | **!=**

—------------------------

AE → T AE’
AE’ → A T AE’ | eps

AE → T (A T)*

—------------------------

A → **+** | **-**

—------------------------

T → F T’
T’ → M F T’ | eps

T → F (M F)*

—------------------------

M → * | /

—------------------------

F → **(** E **)** | V | C | **NUM**

F → C | V | **(** E **)** | **NUM**

—------------------------

C → **ID (** ARG **)**
ARG → ARGL | eps

C → **ID ( ) | ID (**ARGL**)**

—------------------------

ARGL → E ARGL’
ARGL’ → , E ARGL’ | eps

ARGL → E (, E)*