

PROGRAMA -> DECL PROGRAMA | PROCEDIMENT PROGRAMA | MAIN.

DECL -> DECL_TUPLA ';' | DECL_TAUOLA ';' | DECL_TIPUS ';' .

MODIF -> MODIF_TUPLA ';' | REM_TUPLA ';' | AFG_TUPLA ';' | MODIF_TAUOLA ';' |
REDEF_TAUOLA ';' | REDEF_TUPLA ';' | MODIF_TIPUS ';' .

SENT -> SENTENCIES SENT | .

SENTENCIES -> DECL | MODIF | MENTRE | IF | SWITCH | FMENTRE | CRIDA_PROC.

PROCEDIMENT -> metode ID '(' ARGS ')' '{' SENT '}' | metode TIPUS ID '(' ARGS ')' '{' SENT
RETURN'}' .

RETURN -> retorna VALOR.

CRIDA_PROC -> ID '(' ARGS ')' ';' .

MAIN -> metode main '(' cad arguments '[' ']' ')' '{' SENT '}' .

ARGS -> ARGS_1 | .

ARGS_1 -> ARG ',' ARGS_1 | ARG.

ARG -> TIPUS ID.

MENTRE -> mentre '(' L ')' fer '{' SENT '}' .

IF -> si '(' VALOR ')' '{' SENT '}' IF_1.

IF_1 -> sino si '(' VALOR ')' '{' SENT '}' IF_1 | sino '{' SENT '}' | .

FMENTRE -> fer '(' SENT ')' mentre '(' L ')' ';' .

SWITCH -> seleccio '(' VALOR ')' '{' SWITCH_1 '}' .

SWITCH_1 -> cas VALOR ':' SENT acaba ';' SWITCH_1 | pdefecte ':' SENT acaba ';' | .

DECL_TIPUS -> TIPUS A | TIPUS A_1 .

MODIF_TIPUS -> ID I_1 .

TIPUS -> ent | logic | decimal | cadena | car .

A -> A_1 I.

A_1 -> ID ',' A_1 | ID.

I -> '=' VALOR | '=' MODIF_TIPUS | '=' MODIF_TUPLA | '=' MODIF_TAUOLA.

I_1 -> '++' | '--' | '+=' VALOR | '-=' VALOR | I.

DECL_TAUOLA -> TIPUS ID '[' ']' '=' NOVA_TAUOLA | TIPUS ID '[' ']' .

NOVA_TAUOLA -> nou ID '[' enter ']' '{' VALORS '}' .

VALORS -> VALOR | VALOR ';' VALORS.

VALOR -> L | E | EXPRESIO | '(' L ')' | NOT '(' L ')' .

MODIF_TAULA -> ID '[' enter ']' I_1 .
REDEF_TAULA -> ID '=' NOVA_TAULA.

DECL_TUPLA -> tupla ID '(' ')' NOVA_TUPLA | tupla ID '(' ')' .
NOVA_TUPLA -> '=' '(' VALORS ') ' .

MODIF_TUPLA -> ID '(' enter ')' I_1.
REM_TUPLA -> ID '.' remove('VALOR') .
AFG_TUPLA -> ID '.' append('VALOR') .
REDEF_TUPLA -> ID '=' NOVA_TUPLA.

E -> F OP.
OP -> '+' F OP | '-' F OP | '*' F OP | '/' F OP | '%' F OP | .

F -> '(' E ')' | venter | vdecimal | vcharacter | '-' venter | '+' venter | '-' vdecimal | '+' vdecimal | '-' vcharacter | '+' vcharacter | vcadena | boolea | id.

L -> F_1 OPP.
OPP -> COMP_LOG F_1 OPP | COMP_LOG F_1.

F_1 -> '(' L ')' | NOT '(' L ')' | E | EXPRESIO.

COMP_LOG -> or | and .
EXPRESIO -> NOT '(' EXPRESIO ')' | '(' EXPRESIO ')' | EXPRESIO_1.

EXPRESIO_1 -> E COMP E | NOT E COMP E | E COMP NOT E | NOT E COMP NOT E .

COMP -> '<=' | '==' | '>=' | '!=',
NOT -> '!' .