PROGRAMA -> DECL PROGRAMA | PROCEDIMENT PROGRAMA | MAIN.

DECL -> DECL_TUPLA ';' | DECL_TAULA ';' | DECL_TIPUS ';' .
MODIF -> MODIF_TUPLA ';' | REM_TUPLA ';'| AFG_TUPLA ';'| MODIF_TAULA ';' |
REDEF_TAULA ';' | REDEF_TUPLA ';' | MODIF_TIPUS ';'.

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

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

RETURN -> return VALOR.

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

MAIN -> metode main '(' cad arguments '[' ']' ')' '{' M_1 '}' .
M_1 -> SENTENCIES M_1 | CRIDA_PROC M_1 | SENTENCIES.

ARGS -> ARGS_1 | .
ARGS_1 -> ARG ',' ARGS_1 | ARG.

ARG -> TIPUS ID.

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

IF -> if '(' VALOR ')' '{' SENTENCIES '}' IF_1.
IF_1 -> else if '(' VALOR ')' '{' SENTENCIES '}' IF_1 | else '{' SENTENCIES '}' | .

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

SWITCH -> switch '(' VALOR ')' '{' SWITCH_1 '}'.
SWITCH_1 -> case VALOR ':' SENTENCIES break ';' SWITCH_1 | default ':' SENTENCIES
break ';' | .

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_TAULA.

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

DECL_TAULA -> TIPUS ID '[' ']' '=' NOVA_TAULA | TIPUS ID '[' ']' .
NOVA_TAULA -> nou ID '[' enter ']' | '{' VALORS '}'.

VALORS -> VALOR | VALOR ',' VALORS.
VALOR -> 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 | vcaracter | '-' venter | '+' venter | '-' vdecimal | '+' vdecimal | '-' vcaracter | '+' vcaracter | cadena | boolea | id.

L -> EXPRESIO | E | E COMP_LOG L | EXPRESIO COMP_LOG L.

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 -> '<=' |