**Grammer without common left prefixes**

PROG -> GLOBAL\_VARS FUNC\_PREDEFS FUNC\_FULL\_DEFS

GLOBAL\_VARS -> VAR\_DEC GLOBAL\_VARS\_

GLOBAL\_VARS\_ -> VAR\_DEC GLOBAL\_VARS\_ | ɛ

VAR\_DEC -> TYPE id VAR\_DEC\_

VAR\_DEC\_ -> ; | [ DIM\_SIZES ] ;

TYPE -> int | float

DIM\_SIZES -> int\_num DIM\_SIZES\_

DIM\_SIZES\_ -> , int\_num DIM\_SIZES\_ | ɛ

FUNC\_PREDEFS -> FUNC\_PROTOTYPE ; FUNC\_PREDEFS\_

FUNC\_PREDEFS\_ -> FUNC\_PROTOTYPE ; FUNC\_PREDEFS\_ | ɛ

FUNC\_PROTOTYPE -> RETURNED\_TYPE id ( PARAMS )

FUNC\_FULL\_DEFS -> FUNC\_WITH\_BODY FUNC\_FULL\_DEFS\_

FUNC\_FULL\_DEFS\_ -> FUNC\_WITH\_BODY FUNC\_FULL\_DEFS\_ | ɛ

FUNC\_WITH\_BODY -> FUNC\_PROTOTYPE COMP\_STMT

RETURNED\_TYPE -> TYPE | void

PARAMS -> PARAM\_LIST | ɛ

PARAM\_LIST -> PARAM PARAM\_LIST\_

PARAM\_LIST\_ - > , PARAM PARAM\_LIST\_ | ɛ

PARAM -> TYPE id PARAM\_

PARAM\_ -> [ DIM\_SIZES ] | ɛ

COMP\_STMT -> { VAR\_DEC\_LIST\_ STMT\_LIST }

VAR\_DEC\_LIST\_ -> VAR\_DEC VAR\_DEC\_LIST\_| ɛ

STMT\_LIST -> STMT STMT\_LIST\_

STMT\_LIST\_ -> ; STMT STMT\_LIST\_ | ɛ

STMT -> id STMT\_| COMP\_STMT | IF\_STMT | RETURN\_STMT

STMT\_ -> VAR\_ = EXPR | ( ARGS )

IF\_STMT -> if ( CONDITION ) STMT

ARGS -> ARG\_LIST | ɛ

ARG\_LIST -> EXPR ARG\_LIST\_

ARG\_LIST\_ -> , EXPR ARG\_LIST\_ | ɛ

RETURN\_STMT -> return RETURN\_STMT\_

RETURN\_STMT\_ -> EXPR | ɛ

VAR\_ -> [ EXPR\_LIST ] | ɛ

EXPR\_LIST -> EXPR EXPR\_LIST\_

EXPR\_LIST\_ -> , EXPR EXPR\_LIST\_ | ɛ

CONDITION -> EXPR rel\_op EXPR

EXPR -> TERM EXPR\_

EXPR\_ -> + TERM EXPR\_ | ɛ

TERM -> FACTOR TERM\_

TERM\_ -> \* FACTOR TERM\_ | ɛ

FACTOR -> id MOMO | int\_num | float\_num | ( EXPR )

MOMO-> VAR\_ | ( ARGS )

0 1 2 3 4