1. program -> decl { decl }
2. decl -> void ID fun-decl | int ID decl’
3. decl’ -> [ “[“ NUM “]” ] ; | fun-decl
4. var-decl -> int ID [ “[“ NUM “]” ] ;
5. fun-decl -> “(“ params “)” compound-stmt
6. params -> param { , param } | void
7. param -> int ID [ “[“ “]” ]
8. compound-stmt -> “{“ local-decl statement-list “}”
9. local-decl -> [ { var-decl } ]
10. statement-list -> [ { statement } ]
11. statement -> expression-stmt | compound-stmt | selection-stmt | iteration-stmt | return-stmt
12. expression-stmt -> [ expression ] ;
13. selection-stmt -> if ( expression ) statement [ else statement ]
14. iteration-stmt -> while “(“ expression “)” statement
15. return-stmt -> return [ expression ] ;
16. expression -> “(“ expression “)” simple-expression’ | NUM simple-expression’ | ID expression’
17. expression’ -> = expression | simple-expression’ | “(“ args “)” simple-expression’ | “[“ expression “]” expression’’
18. expression’’ -> = expression | simple-expression’
19. relop -> <= | < | > | >= | == | !=
20. additive-expression -> term { additive-expression term }
21. term -> factor { mulop factor }
22. factor -> “(“ expression “)” | var-call’ | NUM
23. var-call’ -> “(“ args “)” | [ “[“ expression “]” ]
24. simple-expression’ -> additive-expression’ [ relop additive-expression ]
25. additive-expression’ -> term’ { addop term }
26. term’ -> { mulop factor }
27. addop -> +|-
28. mulop -> \*|/
29. args -> [ expression {, expression} ]