CNF Grammar of our language

Mobin Razavi, Ali Ahmadvand, Mohammad Davoodabadi

Our grammar is based on C-minus grammar which you can find in http://www.csci-snc.com/resources/ExamplesX/C-Syntax.pdfhere.

1. program → declaration-list
2. declaration-list → declaration-list declaration | declaration
3. declaration → var-declaration | fun-declaration
4. var-declaration → type-spec ID; | type-spec ID [NUM];
5. type-spec → int | float | string
6. fun-declaration → type-spec ID (params) compound-stmnt
7. params → param-list | ϵ
8. param-list → param-list, param | param
9. param → type-spec ID | type-spec ID [ ]
10. compound-stmnt → {stmnt-list}
11. stmnt-list → stmnt-list stmnt | ϵ
12. stmnt → compound-stmnt | cond-stmnt | iter-stmnt | return-stmnt | expression-stmnt | var-declaration
13. cond-stmnt → if (expression) stmnt | if (expression) compound-stmnt else compound-stmnt
14. iter-stmnt → while (expression) compound-stmnt
15. return-stmnt → return; | return expression;
16. expression-stmnt → expression; | ;
17. expression → var = expression | simple-expression
18. var → ID | ID [expression]
19. simple-expression → logical-expression rel-op logical-expression | logical-expression

20. rel-op $\rightarrow$ $>$ | $<$ | $>=$ | $<=$ | $==$ | $!=$

21. logical-expression $\rightarrow$ logical-expression log-op additive-expression | additive-expression

22. log-op $\rightarrow$ && | ||

23. additive-expression $\rightarrow$ additive-expression add-op mul-expression | mul-expression

24. add-op $\rightarrow$ $+$ | $-$

25. mul-expression $\rightarrow$ mul-expression mul-op term | term

26. mul-op $\rightarrow$ $*$ | $/$

27. term $\rightarrow$ (expression) | var | call | NUM | FNUM | string-word

28. call $\rightarrow$ ID (args) | print(print-full-args)

29. print-full-args $\rightarrow$ string args

30. args $\rightarrow$ arg-list | ϵ

31. arg-list $\rightarrow$ arg-list, expression | expression

keywords: if else int float string return while print
Special symbols: $+ - * /$ & | $> < >= <= == != , ;$
ID = $letter\ (letter + digit)^*$
NUM = $digit^+$
FNUM = NUM.NUM
string-word = "$(letter + digit + whitespace)^*$"