Extended BNF Grammar for C-Algol

Metasymbols: The following metasymbols will be used for this grammar. { *statement* } means 0 or more repetitions of *statement*. [*statement*]$_{+}$ means that the statement is optional.

Bold face items are specific tokens. Uppercase boldface are tokens which should also have a specific value attached to the token (an attribute).

1. *program* → *declaration-list*

2. *declaration-list* → *declaration* { *declaration* }

3. *declaration* → *var-declaration* **|** *fun-declaration*

4. *var-declaration* → *type-specifier var-list* **;**

4a. var-list → **ID** | **ID [NUM] | ID ,** *var-list* | **ID [NUM] ,** *var-list*

5. *type-specifier* → **int | void | boolean**

6. *fun-declaration* →*type-specifier* ***ID*** **(** *params* **)** *compound-stmt*

7. *params* → **void |** *param-list*

8. *param-list* → *param* { **,** *param* }

9. *param* → *type-specifier* ***ID*** [**[]**]

10. *compound-stmt* → ***begin*** *local-declarations statement-list* ***end***

11. *local-declarations* → { *var-declarations* }

12. *statement-list* → { *statement* }

13. *statement* → *expression-stmt*
| *compound-stmt*
| *selection-stmt*
| *iteration-stmt*
| *assignment-stmt*
| *return-stmt*
| *read-stmt*
| *write-stmt*

14. *expression-stmt* → *expression* **;** | **;**

15. *selection-stmt* → **if** *expression* ***then*** *statement* [**else** *statement*] $_{+}$

16. *iteration-stmt* → **while** *expression* ***do*** *statement*

17. *return-stmt* → **return** [*expression*]$_+$ **;**

18. *read-stmt* → **read** *variable* **;**

19. *write-stmt* → **write** *expression* **;**

20. *assignment-stmt* → *var* **=** *simple-expression* **;**

21. *expression* → *simple-expression*

22. *var* → ***ID*** **[** [*expression*] **]** $_+$

23. *simple-expression* → *additive-expression* [*relop additive-expression*] $_+$

22. *relop* → **<=** | **<** | **>** | **>=** | **==** | **!=**

23. *additive-expression* → *term* { *addop term* }

24. *addop* → **+** | **-**

25. *term* → *factor* { *multop factor* }

26. *multop* → * | **/** | **and** | **or**

27. *factor* → **(** *expression* **)** | ***NUM*** | *var* | *call* | ***true*** | ***false*** **|** ***not*** *factor*

28. *call* → ***ID*** **(** *args* **)**

29. *args* → *arg-list* **|** *empty*

30. *arg-list* → *expression* { **,** *expression* }

Regular Expressions

1. ***ID*** = ***letter letter****
2. ***NUM*** = ***digit digit****
3. ***letter*** = ***a*** | ***b*** | … | ***z*** | ***A*** | ***B*** | … | ***Z***
4. ***digit*** = ***0*** | ***1*** | … | ***9***