LL grammar

<prog> -> <decl_list> Scope EOL <stat_list> End Scope EOF
<decl_list> -> <var_decl> <decl_list>
<decl_list> -> <func_decl> <decl_list>
<decl_list> -> <func_def> <decl_list>
<decl_list> -> <empty_statement>
<decl_list> -> ε
<func_decl> -> Declare Function id (<param_list>) As <type> EOL
<func_def> -> Function id (<param_list>) As <type> EOL <stat_list> End Function EOL
<var_decl> -> Dim <opt_scope_modifier> id As <type> <opt_initialiser> EOL
<var_decl> -> Static id as <type> <opt_initialiser> EOL
<param_list> -> <param> <param_list_cont>
<param_list> -> ε
<param> -> id As <type>
<param_list_cont> -> , <param> <param_list_cont>
<param_list_cont> -> ε
<opt_scope_modifier> -> Shared
<opt_scope_modifier> -> ε
<opt_initialiser> -> = <expr>
<opt_initialiser> -> ε
<type> -> Integer
<type> -> Double
<type> -> String
<cycle_stat_list> -> <statement> <cycle_stat_list>
<cycle_stat_list> -> <iteration_control_statement> <cycle_stat_list>
<cycle_stat_list> -> ε
<stat_list> -> <statement> <stat_list>
<stat_list> -> ε
<statement> -> <assignment>
<statement> -> <read_statement>
<statement> -> <print_statement>
<statement> -> <scope_statement>
<statement> -> <selection_statement>
<statement> -> <iteration_statement>
<statement> -> <return_statement>
<statement> -> <empty_statement>
<statement> -> <var_decl>
<assignment> -> id = <expr> EOL
<read_statement> -> Input id EOL
<print_statement> -> print <expr> ; <expr_list> EOL
<expr_list> -> <expr> ; <expr_list>
<expr_list> -> ε
<scope_statement> -> Scope EOL <stat_list> End Scope EOL
<selection_statement> -> If <expr> then EOL <stat_list> <alternative_statement> End If EOL
<alternative_statement> -> elseif <expr> then EOL <stat_list> <alternative_statement>
<alternative_statement> -> else EOL <stat_list>
<alternative_statement> -> ε

<iteration_statement> -> for id <opt_type_decl> = <expr> to <expr> <opt_step> EOL <cycle_stat_list> Next <opt_id> EOL
<opt_type_decl> -> as <type>
<opt_type_decl> -> ε
<opt_step> -> step <expr>
<opt_step> -> ε
<opt_id> -> id
<opt_id> -> ε
<iteration_statement> -> do <do_cycle> EOL
<do_cycle> -> EOL <cycle_stat_list> Loop <opt_cond>
<do_cycle> -> <cond> EOL <cycle_stat_list> Loop
<opt_cond> -> <cond>
<opt_cond> -> ε
<cond> -> <do_mode> <expr>
<do_mode> -> until
<do_mode> -> while
<return_statement> -> return <expr> EOL
<iteration_control_statement> -> <control_statement> <cycle_type> <cycle_type_list> EOL
<control_statement> -> Exit
<control_statement> -> Continue
<cycle_type_list> -> , <cycle_type> <cycle_type_list>
<cycle_type_list> -> ε
<cycle_type> -> for
<cycle_type> -> do
<empty_statement> -> EOL