Abstract Syntax of SOL

program → *func-decl*
func-decl → **id** *decl-list-opt domain type-sect-opt var-sect-opt const-sect-opt func-list-opt func-body*
decl-list-opt → { *decl* }
decl → *id-list domain*
id-list → { **id** }$^+$
domain → **atomic-domain** | *struct-domain* | *vector-domain* | **id**
where **atomic-domain** qualified as: `CHAR` | `INT` | `REAL` | `STRING` | `BOOL`
struct-domain → { *decl* }$^+$
vector-domain → **intconst** *domain*
type-sect-opt → { *decl* }
var-sect-opt → { *decl* }
const-sect-opt → { *decl* EXPR }
func-list-opt → { *func-decl* }
func-body → **id** *stat-list* **id**
stat-list → { *stat* }$^+$
stat → *assign-stat* | *if-stat* | *while-stat* | *for-stat* | *foreach-stat* | *return-stat* | *read-stat* | *write-stat*
assign-stat → *left-hand-side* EXPR
left-hand-side → **id** | *fielding* | *indexing*
fielding → *left-hand-side* **id**
indexing → *left-hand-side* EXPR
if-stat → EXPR *stat-list elsif-stat-list-opt* [*stat-list*]
elsif-stat-list-opt → { EXPR *stat-list* }
while-stat → EXPR *stat-list*
for-stat → **id** EXPR EXPR *stat-list*
foreach-stat → **id** EXPR *stat-list*
return-stat → EXPR
read-stat → *specifier-opt* **id**
specifier-opt → [EXPR]
write-stat → *specifier-opt* EXPR

define EXPR = (*logic-expr* | *rel-expr* | *math-expr* | *neg-expr* | *rd-expr* | *left-hand-side* |
charconst | **intconst** | **realconst** | **strconst** | **boolconst** |
instance-expr | *func-call* | *wr-expr* | *cond-expr* | *built-in-call*)

where qualification is:

logic-expr : `AND` | `OR`
rel-expr : `EQUAL` | `NEQ` | `'>'` | `GEQ` | `'<'` | `LEQ` | `IN`
math-expr : `'+'` | `'-'` | `'*'` | `'/'`
neg-expr : `'-'` | `NOT`
instance-expr : `STRUCT` | `VECTOR`
built-in-call : `TOINT` | `TOREAL`

logic-expr → EXPR EXPR
rel-expr → EXPR EXPR
neg-expr → EXPR
wr-expr → *specifier-opt* EXPR
rd-expr → *specifier-opt domain*
instance-expr → { EXPR }$^+$
func-call → **id** { EXPR }
cond-expr → EXPR EXPR *elsif-expr-list-opt* EXPR
elsif-expr-list-opt → { EXPR EXPR }
built-in-call → EXPR