```
program
   : ( structDeclaration  | variableDeclaration | functionDeclaration )* EOF
   ;
structDeclaration
   : 'struct' s=ID '{' (ID ':' type ';')* '}'
   ;

variableDeclaration
   : 'var' ID ':' type ';'
   ;

variables
   : (variableDeclaration)*
   ;

functionDeclaration
   : ID '(' parameterList ')' '{' variables statements '}'
   | ID '(' parameterList ')' ':' type '{' variables statements '}'
   ;

parameterList
   : (ID ':' type (',' ID ':' type)*)?

statement
   : 'print' expressionList ';'
   | 'printsp' expressionList ';’
   | 'println' expressionList ';’
   | 'read' expression ';'
```

```
  | 'if' '(' expression ')' '{' statements '}'
  | 'if' '(' expression ')' '{' s1 = statements '}' 'else' '{' s2= statements '}'
  | 'while' '(' expression ')' '{' statements '}'
  | 'return' expression ';'
  | 'return' ';'
  | e1=expression '=' e2=expression ';'
  | ID '(' expressionList ')' ';'
  ;

statements
  : (statement)*
  ;

expression
  : CHAR_LITERAL
  | ID
  | LITENT
  | LITREAL
  |'<' type '>' '(' expression ')'
  | e=expression '.' ID
  | e1=expression '[' e2=expression ']'
  | ID '(' expressionList ')'
  | '(' expression ')'
  | '!' expression
  | e1=expression op=('*' | '/' | '%') e2=expression
  | e1=expression op=('+' | '-') e2=expression
  | e1=expression op=('<' | '>' | '<=' | '>=') e2=expression
  | e1=expression op=('==' | '!=') e2=expression
```

```
    | e1=expression op='&&' e2=expression
    | e1=expression op='||' e2=expression
    ;

expressionList
    : (expression (',' expression)*)?
    ;

type
    : 'int'
    | 'float'
    | 'char'
    | '[' LITENT ']' type
    | ID
    ;
```