```
PROGRAM -> ( DECL | ARRAY_DECL | ASSIGN | FUNC_IMPL )*

DECL -> TYPE ID ( '=' EXPR )? ';'
ASSIGN -> ID '=' ( EXPR ) ';'

ARRAY_DECL -> TYPE ID ( '[' ( EXPR ) ']' )+ ( '=' ARRAY_INIT )? ';'
ARRAY_INIT -> ID = ( STRING | '{' ( NUMBER ) ( ',' NUMBER )* '}' )
ARRAY_ELEM -> ID ( '[' ( EXPR ) ']' )+

FUNC_IMPL -> TYPE ID '(' PARAMS ')' '{' ( INSTR | RET )* ( RET )? '}'
FUNC_CALL -> ID '(' ARGS ')' ';'

PARAMS -> ( TYPE ID )? ( ',' TYPE ID )*
ARGS -> ( EXPR )? ( ',' EXPR )*

INSTR -> ( DECL | ARRAY_DECL | ASSIGN | FUNC_CALL | STAT )
JUMP -> ( 'break' | 'continue' ) ';'
RET -> 'return' ( EXPR )? ';'

STAT -> STAT_IF | STAT_WHILE | STAT_FOR
STAT_IF -> 'if' '(' LOGIC_EXPR ')' '{' ( INSTR )* '}' ( 'else' '{' ( INSTR )* '}' )?
STAT_WHILE -> 'while' '(' LOGIC_EXPR ')' '{' ( INSTR | JUMP )* '}'
STAT_FOR -> 'for' '(' ASSIGN ';' LOGIC_EXPR ';' EXPR ')' '{' ( INSTR | JUMP )* '}'

EXPR -> TERM ( ( '+' | '-' ) TERM )*
TERM -> FACT ( ( '*' | '/' | '%' ) FACT )*
FACT -> ( '-' )? ( CONST | ID | ARRAY_ELEM | FUNC_CALL | '(' EXPR ')' )

LOGIC_EXPR -> LOGIC_OR ( ( ( '==' | '!=' | '<' | '>' | '<=' | '>=' ) LOGIC_OR )*
LOGIC_OR -> LOGIC_AND ( '||' LOGIC_AND )*
LOGIC_AND -> LOGIC_FACT ( '&&' LOGIC_FACT )*
LOGIC_FACT -> ( '!' )? ( ID | '(' LOGIC_EXPR ')' )

TYPE -> ( 'int' | 'char' | 'void' )
NUMBER -> ( INT | CHAR )
CONST -> ( NUMBER | STRING )

ID, INT, CHAR, STRING -> . . .
```