<relation>  :: = < | <= | == | != | >= | > | && |  | |
<operation> :: = + | - | * | / | % | ^
<primary_type> :: = number | string | bool
<array_type> :: = <primary_type> [ ]
<type> :: = <primary_type> | <array_type>
<declaration> :: = let <id> : <type> ;
<declaration_list> :: = <declaration> | <declaration><declaration_list>

<constant> :: = <number> | <string> | <bool>
<term> :: = <id> | <constant>
<term_list> :: = <term> | <term><term_list>;
<expression> ::= <term> |  <expression><operation><expression> |
(<expression><operation><expression>)

<inputstmt> :: = read ( <id> );
<outputstmt> :: = printf ( <term_list> );
<ios_statement> :: = <inputstmt> | <outputstmt>
<assignment> :: = <id> = <expression>; | <id> ++; | <id> - -;
<simple_statement> :: = <assignment> | <ios_statement>

<statement> :: = <declaration> | <simple_statement> | <struct_statement>
<statement_list> :: = <statement> | <statement><statement_list>
<compstmt> :: = { <statement_list> }

<whilestmt> :: = while ( <condition> ) <compstmt>
<forstmt> :: = for ( <assignment>; <condition>; <assignment> ) { <statement> }
<ifstmt> :: = if ( <condition> ) { <statement> } | if ( <condition> ) { <statement> }
else { <statement> } | if ( <condition> ) { <statement> } else <ifstmt>

<struct_statement> :: = <whilestmt> | <forstmt> | <ifstmt> | <compstmt>

<condition> :: = <expression><relation><expression> |
!(<expression><relation><expression>)

<program> :: = @ <declaration_list> ; <compstmt> @