Algol 60 grammar in BNF

<program> ::= <block> | <compound statement>
<block> ::= <unlabelled block> | <label>: <block>
<unlabelled block> ::= <block head> ; <compound tail>
<block head> ::= **begin** <declaration> | <block head> ; <declaration>
<compound statement> ::= <unlabelled compound> | <label>: <compound statement>
<unlabelled compound> ::= **begin** <compound tail>
<compound tail> ::= <statement> **end** | <statement> ; <compound tail>

<declaration> ::= <type declaration> | <array declaration> | <switch declaration> | <procedure declaration>
<type declaration> ::= <local or own type> <type list>
<local or own type> ::= <type> | **own** <type>
<type> ::= **real** | **integer** | **boolean**
<type list> ::= <simple variable> | <simple variable> , <type list>
<array declaration> ::= **array** <array list> | <local or own type> **array** <array list>
<array list> ::= <array segment> | <array list> , <array segment>
<array segment> ::= <array identifier> [<bound pair list>] | <array identifier> , <array segment>
<array identifier> ::= <identifier>
<bound pair list> ::= <bound pair> | <bound pair list> , <bound pair>
<bound pair> ::= <lower bound> : <upper bound>
<upper bound> ::= <arithmetic expression>
<lower bound> ::= <arithmetic expression>
<switch declaration> ::= **switch** <switch identifier> := <switch list>
<switch identifier> ::= <identifier>
<switch list> ::= <designational expression> | <switch list> , <designational expression>
<procedure declaration> ::= **procedure** <procedure heading> <procedure body> | <type> **procedure** <procedure heading> <procedure body>
<procedure heading> ::= <procedure identifier> <formal parameter part> ; <value part> <specification part>
<procedure identifier> ::= <identifier>
<formal parameter part> ::= <empty> | (<formal parameter list>)
<formal parameter list> ::= <formal parameter> | <formal parameter list> <parameter delimiter> <formal parameter>
<formal parameter> ::= <identifier>
<value part> ::= **value** <identifier list> ; | <empty>
<specification part> ::= <empty> | <specifier> <identifier list> ; | <specification part> <specifier> <identifier list>
<specifier> ::= **string** | <type> | **array** | <type> **array** | **label** | **switch** | **procedure** | <type> **procedure**
<identifier list> ::= <identifier> | <identifier list> , <identifier>
<procedure body> ::= <statement> | <code>

<statement> ::= <unconditional statement> | <conditional statement> | <for statement>
<unconditional statement> ::= <basic statement> | <compound statement> | <block>
<basic statement> ::= <unlabelled basic statement> | <label>: <basic statement>
<label> ::= <identifier> | <unsigned integer>
<unlabelled basic statement> ::= <assignment statement> | <go to statement> | <dummy statement> | <procedure statement>
<assignment statement> ::= <left part list> <arithmetic expression> | <left part list> <Boolean expression>
<left part list> ::= <left part> | <left part list> <left part>
<left part> ::= <variable> := | <procedure identifier> :=
<go to statement> ::= **goto** <designational expression>
<designational expression> ::= <simple designational expression> | <if clause> <simple designational expression> **else** <designational expression>
<simple designational expression> ::= <label> | <switch designator> | (<designational expression>)
<switch designator> ::= <switch identifier> [<subscript expression>]
<dummy statement> ::= <empty>
<procedure statement> ::= <procedure identifier> <actual parameter part>
<actual parameter part> ::= <empty> | (<actual parameter list>)
<actual parameter list> ::= <actual parameter> | <actual parameter list> <parameter delimiter> <actual parameter>
<parameter delimiter> ::= , |) <letter string> : (
<actual parameter> ::= <string> | <expression> | <array identifier> | <switch identifier> | <procedure identifier>
<conditional statement> ::= <if statement> | <if statement> **else** <statement> | <if clause> <for statement> | <label>: <conditional statement>
<if statement> ::= <if clause> <unconditional statement>
<if clause> ::= **if** <Boolean expression> **then**
<for statement> ::= <for clause> <statement> | <label>: <for statement>
<for clause> ::= **for** <variable> := <for list> **do**
<for list> ::= <for list element> | <for list> , <for list element>
<for list element> ::= <arithmetic expression> |
<arithmetic expression> **step** <arithmetic expression> **until** <arithmetic expression> | <arithmetic expression> **while** <Boolean expression>

<expression> ::= <arithmetic expression> | <Boolean expression> | <designational expression>
<arithmetic expression> ::= <simple arithmetic expression> | <if clause> <simple arithmetic expression> **else** <arithmetic expression>
<simple arithmetic expression> ::= <term> | <adding operator> <term> | <simple arithmetic expression> <adding operator> <term>
<adding operator> ::= + | –
<term> ::= <factor> | <term> <multiplying operator> <factor>
<multiplying operator> ::= × | / | ÷
<factor> ::= <primary> | <factor> | <factor> ↑ <primary>
<primary> ::= <unsigned number> | <variable> | <function designator> | (<arithmetic expression>)
<unsigned number> ::= <decimal number> | <exponential part> | <decimal number> <exponential part>

<decimal number> ::= <unsigned integer> | <decimal fraction> | <unsigned integer> <decimal fraction>
<unsigned integer> ::= <digit> | <unsigned integer> <digit>
<decimal fraction> ::= . <unsigned integer>
<exponential part> ::= $_{10}$ <integer>
<integer> ::= <unsigned integer> | + <unsigned integer> | – <unsigned integer>
<Boolean expression> ::= <simple Boolean> | <if clause> <simple Boolean> **else** <Boolean expression>
<simple Boolean> ::= <implication> | <simple Boolean> ≡ <implication>
<implication> ::= <Boolean term> | <implication> ⊃ <Boolean term>
<Boolean term> ::= <Boolean factor> | <Boolean term> ∨ <Boolean factor>
<Boolean factor> ::= <Boolean secondary> | <Boolean factor> ∧ <Boolean secondary>
<Boolean secondary> ::= <Boolean primary> | ¬ <Boolean primary>
<Boolean primary> ::= <logical value> | <variable> | <function designator> | <relation> | (<Boolean expression>)
<relation> ::= <simple arithmetic expression> <relational operator> <simple arithmetic expression>
<relational operator> ::= < | ≤ | = | ≠ | > | ≥
<function designator> ::= <procedure identifier> <actual parameter part>

<variable> ::= <simple variable> | <subscripted variable>
<simple variable> ::= <variable identifier>
<variable identifier> ::= <identifier>
<subscripted variable> ::= <array identifier> [<subscript list>]
<subscript list> ::= <subscript expression> | <subscript list> , <subscript expression>
<subscript expression> ::= <arithmetic expression>

<string> ::= "<open string>"
<open string> ::= <proper string> "<open string>" | <open string><open string>
<proper string> ::= <any sequence of symbols not containing " > | <empty>

<letter string> ::= <letter> | <letter string> <letter>

<identifier> ::= letter> | <identifier> <letter> | <identifier> <digit>

<basic symbol> ::= <letter> | <digit> | <logical value> | <delimiter>

<letter> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<logical value> ::= **true** | **false**

<delimiter> ::= <operator> | <separator> | <bracket> | <declarator> | <specificator>

<operator> ::= <arithmetic operator> | <relational operator> | <logical operator> | <sequential operator>
<arithmetic operator> ::= + | – | × | / | ÷ | ↑
<relational operator> ::= < | ≤ | = | ≠ | > | ≥
<logical operator> ::= ≡ | ⊃ | ∨ | ∧ | ¬
<sequential operator> ::= **goto** | **if** | **then** | **else** | **for** | **do**

<separator> ::= , | . | $_{10}$ | : | ; | := | _ | **step** | **until** | **while** | **comment**
<bracket> ::= (|) | [|] | ` | ' | **begin** | **end**
<declarator> ::= **own** | **boolean** | **integer** | **real** | **array** | **switch** | **procedure**
<specificator> ::= **string** | **label** | **value**