LEXIC
 alphabet:
  upper case (A-Z)
  lower case (a-z)
  digits (0-9)
  underscore(_)

 operators:
  unaryoperation ::= "!"
  binaryoperation ::= ">" | "<" | "=" | "+" | "-" | "/" | "*" | "%" | "!=" | "==" | ">=" | "<="
  separators: [], {}, ;, space

 reserved words:
  defINT, main, defSTRING, input, output, if, while, for, else, defBOOl, false, true, defFLOAT

 identifiers:
  a sequence of letters, digits and underscore, but the first character is a letter
  identifier ::= letter{letter|digit|underscore}
  letter ::= "A" | "B" | ... | "Z" | "a" | "b" | ... | "z"
  digit ::= "0" | "1" | ... | "9"
  underscore = "_"

 constants:
  integer:
  constant ::= ("+"|"-")non_zero_digit{digit}|0
  non_zero_digit ::= "1" | "2" | ... | "9"
  digit ::= 0 | non_zero_digit

 number:
  number ::= non_zero_digit{digit}|0
  non_zero_digit ::= "1" | ... | "9"
  digit ::= 0 | non_zero_digit

 character:
  character ::= letter | digit | symbol
  letter ::= upper | lower
  upper ::= "A" | "B" | ... | "Z"
  lower ::= "a" | "b" | ... | "z"
  digit ::= "0" | "1" | ... | "9"
  symbol ::= "#" | "!" | "?" | "*" | "^" | "space" | "," | ";" | ":" | "@" | "$" | "%" | "&" | "(" | ")

 string:
  string ::= "{string_char}"
  string_char ::= letter | digit | symbol
  letter ::= upper | lower
  upper ::= "A" | "B" | ... | "Z"
  lower ::= "a" | "b" | ... | "z"
  digit ::= "0" | "1" | ... | "9"
  symbol ::= "#" | "!" | "?" | "*" | "^" | "space" | "," | ";" | ":" | "@" | "$" | "%" | "&" | "(" | ")"

 bool:
  bool ::= "false" | "true"

SYNTAX:

predefined tokens are specified between " and "
program ::= "main" "(" ")" "{" stmtlist "}"
stmtlist ::= stmt | stmt stmtlist
stmt ::= simplestmt | compstmt
simplestmt ::= assignstmt | iostmt | declstmt
assignstmt ::= identifier "=" expression
expression ::= constant | identifier | identifier"["number"]" | expression binaryoperation expression | unaryoperation expression
iostmt ::= "output" "(" identifier ")" | "input" "(" identifier ")"
declstmt ::= type identifier | type identifier "=" expression
type ::= simpletype | arraytype
simpletype ::= "defINT" | "defBOOL" | "defSTRING" | "defFLOAT"
arraytype ::= simpletype"[]"
compstmt ::= whilestmt | ifstmt
ifstmt ::= "if" "(" expression ")" "{" stmtlist "}" ["else" "{" stmtlist "}"]
whilestmt ::= "while" "(" expression ")" "{" stmtlist "}"