

→ Operators :

arithmetic → $+, -, \times, \div, | |, \%, \ll, \gg$

logical : → $\&\&, ||$

comparators : $<, >, ==, >=, <=, !=$

assignment : $=$

delimiters : $(,), \{, \}, [,], ', ;, \langle \text{space} \rangle$

Identifiers :

- ★ start with letter
- ★ alphanumeric.

Literals :

- ★ Integer : sign → sequence of digits.
(edge case for zero)

- ★ float : sign → sequence of digits. → decimal
→ sequence of digits.

* string: sequence of chars, enclosed by double ? quotes which cannot be /n, etc.

Keywords :

int, while, bool, float, for?, if, else,
true, false, return, function., char,
main

BNF for this shit:

$\langle \text{Program} \rangle \rightarrow \langle \text{functions} \rangle \langle \text{main} \rangle$

$\langle \text{functions} \rangle \rightarrow \langle \text{functions} \rangle \langle \text{function} \rangle \mid \epsilon$

$\langle \text{function} \rangle \rightarrow \text{"function"} \langle \text{identifier} \rangle (\langle \text{parameter_list} \rangle) \{ \langle \text{statements} \rangle \} .$

$\langle \text{par_list} \rangle \rightarrow \epsilon \mid \langle \text{identifier} \rangle, \langle \text{par_list} \rangle \langle \text{identifier} \rangle$
 $\langle \text{identifier} \rangle$

$\langle \text{statements} \rangle \rightarrow \langle \text{statement} \rangle \langle \text{statements} \rangle \mid \epsilon$

$\langle \text{statement} \rangle \rightarrow \langle \text{conditional} \rangle \mid \langle \text{loop} \rangle \mid \langle \text{declaration} \rangle ; \mid$
 $\langle \text{assignment} \rangle ; \mid \langle \text{function_call} \rangle ; \mid \langle \text{return_st} \rangle ;$

$\langle \text{conditional} \rangle \rightarrow \text{"if"} (\langle \text{conditions} \rangle) \{ \langle \text{statements} \rangle \} . \mid$
 $\text{"if"} (\langle \text{conditions} \rangle) \{ \langle \text{statements} \rangle \} \text{"else"} \{ \langle \text{statements} \rangle \}$

$\langle \text{conditions} \rangle \rightarrow \langle \text{condition} \rangle \langle \text{logical_operator} \rangle \langle \text{condition} \rangle \mid$
 $\langle \text{condition} \rangle$

$\langle \text{condition} \rangle \rightarrow \langle \text{identifier} \rangle \langle \text{comparator} \rangle \langle \text{term} \rangle$

$\langle \text{loop} \rangle \rightarrow \text{"for"} (\langle \text{assignment} \rangle ; \langle \text{conditions} \rangle ; \langle \text{assignment} \rangle)$
 $\{ \langle \text{statements} \rangle \}$

$\langle \text{assignment} \rangle \rightarrow \langle \text{identifier} \rangle = \langle \text{expression} \rangle$ /
 $\langle \text{declaration} \rangle = \langle \text{expression} \rangle$

$\langle \text{declaration} \rangle \rightarrow \langle \text{datatype} \rangle \langle \text{identifier} \rangle$

$\langle \text{expression} \rangle \rightarrow \langle \text{expression} \rangle \langle \text{logical-operator} / \text{arithmetic-operator} \rangle \langle \text{expression} \rangle$

notice the brackets $\rightarrow (\langle \text{expression} \rangle) \mid \langle \text{term} \rangle \mid \langle \text{function-call} \rangle$
 $\mid \langle \text{unary-operator} \rangle \langle \text{identifier} \rangle$

$\langle \text{Term} \rangle \rightarrow \langle \text{identifier} \rangle \mid \langle \text{literal} \rangle$

$\langle \text{function-call} \rangle \rightarrow \langle \text{identifier} \rangle (\langle \text{arg-list} \rangle)$
(created arg list separately because we need literals in function calls)

$\langle \text{arg-list} \rangle \rightarrow \epsilon \mid \langle \text{Term} \rangle , \langle \text{arg-list} \rangle , \langle \text{Term} \rangle \mid \langle \text{term} \rangle$

$\langle \text{return-statement} \rangle \rightarrow \text{"return"} \mid \text{"return"} \langle \text{term} \rangle$

$\langle \text{identifier} \rangle \rightarrow [A-Z][A-Z, 0-9]^* - \{ \text{keywords} \}$
terminal (because lexer does not return lower than this)

$\langle \text{datatype} \rangle \rightarrow \text{int} \mid \text{char} \mid \text{float} \mid \text{bool} \mid \text{string}$

$\langle \text{arithmetic-operator} \rangle \rightarrow + \mid * \mid / \mid - \mid \% \mid \ll \mid \gg$

<logical-operator> \rightarrow && | ||

<comparator> \rightarrow < | > | == | >= | <= | !=

<unary-operator> \rightarrow ++ | -- | ! | - | +