→ Operators :

arithmetic → +, −, ×, ÷, | |, %, <<, >>

logical :→ &&, ||

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

assignment : =

delimiters : (,), {, }, [,], ', ;, <space>

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 : $\boxed{\text{int, while, bool, float, for?; if, else, true, false, return, function., char, main}}$

BNF for this shit :

<Program> → <functions> <main>

<functions> → <functions> <function> | e

<function> → "function" <identifier> (<parameter_list>)
{ <statements> } .

<par_list> → e | <identifier>, <par_list> <identifier>

<identifier>

<statements> → <statement> <statements> | e

<statement> → <conditional> | <loop> | <declaration>; |
<assignment>; | <function_call> ; | <return_st>;

<conditional> → "if" (<conditions>) {<statiments>} |

"if" (<conditions>) {<statiments>} "else" {<statements>}

<conditions> → <condition> <logical_operator> <condition> |

<condition>

<condition> → <identifier> <comparator> <term>

```
<loop> → "for" (<assignment> ;<conditions> ;<assignment>)
            { <statements> }

<assignment> → <identifier> = <expression> |
               <declaration> = <expression>

<declaration> → <datatype> <identifier>

                                 logical_operator/
<expression> → <expression> <arithmetic_operator> <expression> |
notice the brackets → (<expression>) | <term> | <function_call>
               | <unary_operator> <identifier>

<Term> → <identifier> | <literal>

                  (created arg list seperately because we need literals in function calls)
<function_call> → <identifier> ( <arg_list> )

<arg_list> → e |<Term>, <arg_list>, <Term> |<term>


<return_statement> → "return" | "return" <term>
                 terminal (because lexer does not return lower than this)
<identifier> → [A-Z] [A-Z, 0-9]* - {keywords}.

<datatype> → int | char | float | bool | string
<arithmetic_operator> → + | * | / | - | % | << | >>
```

```
<logical_operator> → && | ||

<comparator> → < | > | == | >= | <= | !=

<unary_operator> → ++ | -- | ! | - | +
```