Grammar Definition

⟨*program*⟩ ::= ⟨*code block*⟩⟨*statement*⟩[⟨*code block*⟩ | ϵ] | ⟨*control flow construct*⟩[⟨*code block*⟩ | ϵ]
⟨*control flow construct*⟩ ::= ⟨*while*⟩ | ⟨*for*⟩ | ⟨*switch*⟩ | ⟨*if*⟩
⟨*while*⟩ ::= `BEGIN`⟨*operators*⟩`WHILE`⟨*code block*⟩`REPEAT`
⟨*for*⟩ ::= `DO`⟨*code block*⟩`LOOP`
⟨*if*⟩ ::= `IF`⟨*code block*⟩`ENDIF` | `IF`⟨*code block*⟩`ELSE`⟨*code block*⟩`ENDIF`
⟨*switch*⟩ ::= `CASE`{`literal OF`⟨*code block*⟩`ENDOF`}`ENDCASE`
⟨*statements*⟩ ::= ⟨*statement*⟩[⟨*statements*⟩ | ϵ]
⟨*function definition*⟩ ::=: `identificator`⟨*code block*⟩`;`
⟨*statement*⟩ ::= ⟨*arithmetic operator*⟩ | ⟨*stack operator*⟩ | ⟨*comparison operator*⟩ | ⟨*logical operator*⟩
| ⟨*address operator*⟩ | ⟨*input operator*⟩ | ⟨*output operator*⟩ | ⟨*control flow operator*⟩
| ⟨*new variable operator*⟩ | ⟨*size operator*⟩ | `literal` | `identificator` (variable or function call)
⟨*stack operator*⟩ ::= `dup` | `drop` | `swap` | `over` | `rot` | `pick` | `nip` | `tuck` | `roll`
⟨*arithmetic operator*⟩ ::= + | `s+` | ∗ | / | − | % | `negate` | `invert` | `lshift` | `rshift`
⟨*comparison operator*⟩ ::=<|>|≤|≥|=| `s=`
⟨*logical operator*⟩ ::= `and` | `or` | `xor` | `not`
⟨*address operator*⟩ ::=! | `f!` | `c!` | @ | `c@` | `f@`
⟨*input operator*⟩ ::= `sinput` | `finput` | `input`
⟨*output operator*⟩ ::= `type` | . | `.s` | `emit`
⟨*control flow operator*⟩ ::= `leave` | `continue`
⟨*new variable operator*⟩ ::= `VARIABLE identificator` | `CREATE identificator integer`⟨*size operator*⟩`allot`
⟨*size operator*⟩ ::= `chars` | `floats` | `cells`