

## Grammar description

$G = \{ T, N, S, P \}$  , where

T = terminal symbols

N = non-terminal symbols

S = start symbol

P = derivation rules

$T = \{ a..z, A..Z, , , . , ' ' , " " , \{ , \} , ( , ) , : , ; , 0..9 , + , - , * , / , < , > , = , <= , >= , != , \&\& , || , \# ,$   
return, print, func, var, if, else, while, true, false }

$N = \{ \langle \text{program} \rangle , \langle \text{function} \rangle , \langle \text{id} \rangle , \langle \text{expression} \rangle , \langle \text{operation} \rangle , \langle \text{expression} \rangle , \langle \text{operation} \rangle ,$   
 $\langle \text{declaration} \rangle , \langle \text{function-call} \rangle , \langle \text{display} \rangle , \langle \text{flow-control} \rangle , \langle \text{arithmetic-operation} \rangle , \langle \text{arithmetic-}$   
 $\text{operator} \rangle , \langle \text{logic-operation} \rangle , \langle \text{logic-operator} \rangle , \langle \text{letters} \rangle , \langle \text{digits} \rangle , \langle \text{string} \rangle , \langle \text{char} \rangle , \langle \text{characters} \rangle ,$   
 $\langle \text{number} \rangle , \langle \text{boolean} \rangle \}$

$P = \{$

$\langle \text{program} \rangle \rightarrow \langle \text{function} \rangle$

$\langle \text{function} \rangle \rightarrow \text{func } \langle \text{id} \rangle ( \langle \text{id} \rangle^* ) \{ \langle \text{expression} \rangle^+ [ \text{return } \langle \text{id} \rangle \mid \langle \text{expression} \rangle \langle \text{operation} \rangle ] \}$

$\langle \text{id} \rangle \rightarrow \langle \text{letters} \rangle [ \langle \text{letters} \rangle^* \mid \langle \text{digits} \rangle^* \mid \_ ]$

$\langle \text{expression} \rangle \rightarrow \langle \text{declaration} \rangle \mid$

$\langle \text{function-call} \rangle \mid$

$\langle \text{flow-control} \rangle$

$\langle \text{operation} \rangle \rightarrow \langle \text{arithmetic-operation} \rangle \mid$

$\langle \text{logic-operation} \rangle$

$\langle \text{declaration} \rangle \rightarrow \text{var } \langle \text{id} \rangle [ = \langle \text{id} \rangle \mid \langle \text{operation} \rangle \mid \langle \text{function-call} \rangle \mid \langle \text{string} \rangle \mid \langle \text{number} \rangle ] \mid$

$\text{const } \langle \text{id} \rangle [ = \langle \text{id} \rangle \mid \langle \text{operation} \rangle \mid \langle \text{function-call} \rangle \mid \langle \text{string} \rangle \mid \langle \text{number} \rangle ]$

$\langle \text{function-call} \rangle \rightarrow \langle \text{display} \rangle ( [ \langle \text{id} \rangle^* \mid \langle \text{numbers} \rangle^* \mid \langle \text{char} \rangle^* \mid \langle \text{string} \rangle^* ] ) \mid$

$\langle \text{id} \rangle ( [ \langle \text{id} \rangle^* \mid \langle \text{numbers} \rangle^* \mid \langle \text{char} \rangle^* \mid \langle \text{string} \rangle^* ] )$

$\langle \text{display} \rangle \rightarrow \text{print}$

$\langle \text{flow-control} \rangle \rightarrow \text{if } \langle \text{logic-operation} \rangle \{ \langle \text{expression} \rangle^+ \} [ \text{else } \{ \langle \text{expression} \rangle^+ \} ] \mid$

$\text{while } \langle \text{operation} \rangle :$

$\text{for } ( \text{var } \langle \text{id} \rangle = \langle \text{number} \rangle ; \langle \text{id} \rangle \langle \text{logic-operator} \rangle \langle \text{id} \rangle ;$

$\langle \text{id} \rangle \langle \text{arithmetic-operator} \rangle ) \{ \langle \text{expression} \rangle \}$

$\langle \text{arithmetic-operation} \rangle \rightarrow \langle \text{number} \rangle \langle \text{arithm-operator} \rangle \langle \text{number} \rangle \mid$

$\langle \text{id} \rangle \langle \text{arithm-operator} \rangle \langle \text{id} \rangle$

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

$\langle \text{logic-operation} \rangle \rightarrow \langle \text{number} \rangle \langle \text{logic-operator} \rangle \langle \text{number} \rangle \mid$

$\langle \text{id} \rangle \langle \text{logic-operator} \rangle \langle \text{id} \rangle$

<logic-operator> → = | > | < | >= | <= | != | && | ||

<number> → <digits>+ [. <digits>+]

<letters> → a..z, A...z

<digits> → 0..9

<string> → " <characters>\* "

<char> → ' <characters>\* '

<characters>→

0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#\$%&\'()\*+,-./:;<=>?[\]^\_`{|}~ \t\n

<boolean> → true | false

}