**LEXIC**

**Alphabet:**

* Letters: a-z, A-Z
* Digits: 0-9
* Underline: \_

**Special symbols:**

1. **Operators:** +, -, \*, /, %, =, <, <=, >, >=, ==, and, or, not
2. **Separators:** {} [] ; space ( ) '

**3.Reserved words:** if else number while for boolean read print string start end

**Identifiers:**

identifier = letter{char}

letter = "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"

char = "0" | "1" | "2" | ... | "9" | \_ | "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"

**Constants:**

1. **Constant number:**

constant\_nr = "0" | [" + " | " - "] nonzero\_digit { "0" | nonzero\_digit }

nonzero\_digit = "1" | … | "9"

constant\_nr\_pozitiv = nonzero\_digit { "0" | nonzero\_digit }

1. **Constant string:**

constant\_string = " letter{char} "

letter = "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"

char = "0" | "1" | "2" | ... | "9" | \_ | "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"

1. **Constant boolean:**

constant\_boolean = true | false

**SYNTAX**

program = "start" {declaration} stmt\_list "end"

declaration = type IDENTIFIER ";"

type = basic\_type | array\_type

basic\_type = "number" | "boolean" | "string"

array\_type = basic\_type "[" constant\_nr\_pozitiv "]"

stmt\_list = stmt ";" { stmt ";"}

stmt = read\_stmt | print\_stmt | assign\_stmt | if\_stmt | while\_stmt | for\_stmt

read\_stmt = "read::" IDENTIFIER

print\_stmt = "print::" IDENTIFIER

assign\_stmt = IDENTIFIER "=" expression

expression = CONSTANT | IDENTIFIER | IDENTIFIER operation expression |

CONSTANT operation expression | IDENTIFIER "[" expression "]"

operation = "+" | "-" | "\*" | "/" | "%"

if\_stmt = "if" "(" condition ")" "{" statement\_list "}" "else" "{" stmt\_list "}"

while\_stmt = "while" "(" condition ")" "{" stmt\_list "}"

for\_stmt = "for" "(" assign\_stmt ";" condition ";" expression ")" "{" stmt\_list "}"

condition = expression relation expression

relation = "<" | "<=" | "==" | ">=" | ">"

**TOKENS**

**+**

**-**

**\***

**/**

%

**=**

**>**

**>=**

**<**

**<=**

**==**

true

false

not

and

or

number

boolean

string

if

else

while

for

[

]

{

}

(

)

;

read

print

start

end