LEXICAL

Alphabet:
	- upper (A-Z) and lower case letters (a-z) of the English alphabet
	- underline character '_'
	- decimal digits(0-9)

Lexic:
	- special symbols, representing:
		-> operators + - * / := < <= == > >= != !
		-> separators [] ; 'space'
		-> reserved words: number vector if else loop print while char string start_prg end_prg start stop read and or
	- identifiers:
		-> a sequence of letters and digits, the first character being a digit; the rule is:
			identifier ::= letter | letter{letter}{digit}{nonzerodigit}
			letter ::= "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"
			digit ::= "1" | ... | "9"
			nonzerodigit ::= "0"
	- constants:
		-> integer - rule:
			nrconst ::= "+" no | "-" no | no | "0"
			no ::= nonzerodigit digit{no}
		-> character
			character ::= 'letter' | 'digit'
		-> string
			strchar ::= "string"
			string ::= char{string}
			char ::= letter | digit

SYNTAX

program ::= "start_prg" declist ";" stmtlist "end_prg"
declist ::= declaration | declaration ";" declist
declaration ::= type IDENTIFIER
type ::= type1 | arraydecl
type1 ::= "NUMBER" | "CHAR" | "STRING"
arraydecl ::= "VECTOR" "[" type1 "]"
stmtlist ::= stmt | stmt ";" stmtlist
stmt ::= simplestmt | structstmt
simplestmt ::= assignstmt | iostmt
assignstmt ::= IDENTIFIER ":=" expression
expression ::= expression "+" term | expression "-" term | expression "*" term | | expression "/" term | term
term ::= "(" expression ")" | IDENTIFIER
iostmt ::= "READ" | "PRINT" IDENTIFIER
structstmt ::= ifstmt | whilestmt | loopstmt
ifstmt ::= "IF" condition ":" stmt ["ELSE" stmt]
whilestmt ::= "WHILE" condition ":" stmt ";"
loopstmt ::= "FOR" declaration "=" expression ";" condition ";" expression ":" stmt ";"
condition ::= expression relation expression
relation ::= "<" | "<=" | "==" | ">=" | ">" | "!=" | "and" | "or"

TOKEN

number
vector

```
if
else
loop
print
read
while
char
string
start_prg
end_prg
start
stop
and
or
<
:=
<=
>
>=
==
!
!=
```