Mihalcea Leoanrd 935

```
Alphabet:
    a. Upper (A-Z) and lower case letters (a-z) of the English alphabet
    b. Underline character "_";
    c. Decimal digits (0-9);

Lexic:
     a. operator:
        arithmetic: +, -, *, /, %;
        assignment: =;
        boolean logic: and, or, not;
        comparison: ==, !=, >=, <=, >, <;
    b. separators:
        [], {}, :, ;, space
    c. reserved words:
        break continue pass do if else while return start print read
        Integer String Float Boolean
    d. identifiers:
      a sequence of letters and  digits, such that the first character is a letter;
      the rule is:
            identifier ::= letter | letter{letter}{digit}
            letter ::= "A" | "B" | . ..| "Z" | "a" ... | "z"
            digit ::= "0" | "1" |...| "9"
    e. constants:
        integer:
            integer ::= "0" | ["+"|"-"] ("1"|"2"|...|"9") {"0"|"1"|"2"|...|"9"}
        char:
            char ::= """ alphanumeric """
        float:
            float ::=
                ["+" | "-"] ("1"|"2"|...|"9") rest_of_the_float |
                ("1"|"2"|...|"9") rest_of_the_float |
                "0"
            rest_of_the_float ::= {digit} ["." digit {digit}]
            alphanumeric = letter | digit
        string:
            string ::= {("""alphanumeric""" | " ")}
```

Mihalcea Leoanrd 935

Syntax:

The words - predefined tokens are specified between " and ":

```
program ::= "start" declaration_list ";" compound_statement "."

declaration_list ::= declaration | declaration ";" declaration_list

declaration ::= type identifier

compound_type ::= "Integer" | "Char" | "Float" | "String"

array_declaration ::= type1 "[" number_of_elements "]"

number_of_elements ::= integer

array_access ::= identifier "[" number | identifier "]"

type  ::= type1 | array_declaration

compound_statement ::= "{" statement_list "}"

statement_list ::= statement | statement statement_list

statement ::= simple_statement ";" | structured_statement

simple_statement ::= assignment_statement | IO_statement

assignment_statement ::= identifier "=" expression

expression ::= expression "+" term | term

term ::= term operand [factor | array_access]  | factor

factor ::= "(" expression ")" | identifier

IO_statement ::= ["read" | "write"] "(" identifier ")"

structured_statement ::= compound_statement | if_statement | while_statement

if_statement ::= "if" condition compound_statement "else" compound_statement

while_statement ::= "while" condition compound_statement

condition ::= "(" expression relation expression ")"

relation ::= "<" | "<=" | "==" | "!=" | ">=" | ">"

operand ::= "+" | "-" | "*" | "/"
```

Mihalcea Leoanrd 935

Tokens:

Token type	code
identifier	0
constant	1
[	2
]	3
{	4
}	5
;	6
:	7
,	8
<	9
<=	10
>	11
>=	12
=	13
!	14
not	15
and	16
or	17
read	18
write	19
+	20
-	21
*	22
%	23
/	24
break	25
start	26
Char	27
continue	28
do	29
else	30
float	31
for	32
while	33
if	34
pass	35
return	36
Integer	37
String	38
Float	39
Boolean	40