L1b

Student: Iulia Groza

Lexic.txt

```
Alphabet
a. Lowercase letters of the English alphabet: a-z;
b. Uppercase letters of the English alphabet: A-Z;
c. Decimal digits: 0-9;
d. Underscore character: _.

Lexic
a. Special symbols, representing:
    - operators: +, -, *, /, ==, <, >, <=, >=, =;
    - separators: {, }, (, ), ;, comma, ", space, newline;
    - reserved words: var, readInt, readString, print, if, else,
while, getPos, setPos.
b. Identifiers:
    BNF:
        <identifier> ::=
<letter_or_underline>|<letter_or_underline><char_without_space_seq
>
        <char_without_space__seq> ::=
<char_without_space>|<char_without_space><char_without_space_seq>
        <char_without_space> ::= <letter_or_underline>|<digit>
        <letter_or_underline> ::= a|b|...|z|A|B|...|Z|_
        <digit> ::= 0|1|...|9
c. Constants:
    BNF:
        <int_constant> ::= 0|<abs_val>|<sign><abs_val>
        <sign> ::= +|-
        <abs_val> ::= <non_zero_digit>|<non_zero_digit><digit_seq>
        <digit_seq> ::= <digit>|<digit><digit_seq>
        <non_zero_digit> ::= 1|2|...|9

        <string_constant> ::= "<char_seq>"
        <char_seq> ::= <char>|<char><char_seq>
        <char> ::= <letter_or_underline>| |<digit>
```

token.in

```
+
-
*
/
==
<
>
<=
>=
=
{
}
(
)
;
,
"
space
newline
var
readInt
readString
print
if
else
while
getPos
setPos
```

Syntax.in

```
BNF:
    <program> ::= <statement>|<statement><program>
    <statement> ::=
<var_statement>;|<arr_statement>;|<assign_statement>;|<if_statemen
t>;|<while_statement>;|
                |<return_statement>;|<function_call_statement>;

    <var_statement> ::= var <identifier_list>
    <arr_statement> ::= arr[<positive_number>]
<pure_identifier_list>
    <assign_statement> ::= <identifier> = <expression>
    <if_statement> ::= if(<condition>) { <program> }|if(condition)
{ <program> } else { <program> }
    <while_statement> ::= while(<condition>) { <program> }
    <return_statement> ::= return <expression>
    <function_call_statement> ::=
<function_name>(<expression_list>)

    <pure_identifier_list> ::= <identifier>|<identifier>,
<pure_identifier_list>
    <identifier_list> ::=
<composed_identifier>|<composed_identifier>, <identifier_list>
    <composed_identifier> ::= <identifier>|<identifier> =
<expression>

    <positive_number> ::= +<abs_val>|<abs_val>

    <expression_list> ::= <expression>|<expression>,
<expression_list>
    <expression> ::= <int_expression>|<string_expression>
    <int_expression> ::=
<int_constant>|<identifier>|<operation>|(<operation>)
    <operation> ::= <int_expression> + <int_expression> |
<int_expression> - <int_expression> |
                    | <int_expression> * <int_expression> |
<int_expression> / <int_expression>

    <string_expression> ::=
<string_constant>|<identifier>|<string_expression> +
<string_expression>

    <condition> ::= <expression> == <expression>|<expression> <
<expression>|<expression> > <expression>|
                    |<expression> <= <expression>|<expression> >=
<expression>

    <function_name> ::= readInt|readString|print|getPos|setPos
```