----------Lexic.txt----------

Alphabet:
    Lower case letters: a-z
    Upper case letters: A-Z
    Decimal digits: 0-9
    Underline: _

Operators:
    + - * / % < <= > >= == != && || ! =

Separators:
    ( ) { } ; , space newline

Reserved words:
    begin end if else while readInt readChar print true false int bool char

Identifiers:
    - A sequence of letters, digits and underlines, starting with a letter or an underline (_),
    but if it starts with an underline, it must be followed by a letter or a digit.

    identifier ::= letter | letter {letter}{digit}{_} | _ (letter | digit) {letter}{digit}{_}
    letter ::= "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"
    digit ::= "0" | "1" | ... | "9"

 Constants:
    - Same rule us identifiers, but is must start with "@".

    constant ::= "@" identifier

   1.Integer constant:
        int_constant ::= 0 | [sign] nonzero_digit {digit}
        sign ::= "+" | "-"
        nonzero_digit ::= "1" | ... | "9"
        digit ::= "0" | "1" | ... | "9"

   2.Bool constant:
        bool_constant ::= "true" | "false"

   3.Char constant:
        char_constant ::= letter | digit

----------Syntax.in----------

type ::= "int" | "char" | "bool"
relation ::= "<" | "<=" | "==" | ">=" | ">" | "!=" | "&&" | "||"
operator ::= "+" | "-" | "*" | "/" | "%"

program ::= "begin" "{" decllist stmtlist "}" "end" ";"

decllist ::= declaration | declaration decllist

declaration ::= type IDENTIFIER ";" | type IDENTIFIER "[" int_constant "]" ";"

stmtlist ::= stmt | stmt stmtlist

stmt ::= simplstmt | structstmt

simplstmt ::= assignstmt ";" | arrayassignstmt ";" | iostmt ";"

assignstmt ::= IDENTIFIER "=" expression ";"

arrayassignstmt ::= IDENTIFIER "[" expression "]" "=" expression ";"

iostmt ::= "readInt" "(" IDENTIFIER ")" ";" | "readChar" "(" IDENTIFIER ")" ";" | "print" "(" IDENTIFIER ")" ";"

structstmt ::= ifstmt | whilestmt

ifstmt ::= "if" "(" condition ")" "{" stmtlist "}" ["else" "{" stmtlist "}"]

whilestmt ::= "while" "(" condition ")" "{" stmtlist "}"

factor ::= IDENTIFIER | int_constant | "(" expression ")"

term ::= factor | term operator factor

expression ::= term | expression operator term

condition ::= expression relation expression

----------token.in----------

+
-
*
/
%
>
<
>=
<=
==
!=
==
||
&&

[
]
{
}
(

)
,
;
space
newline

begin
end
if
else
while
readInt
readChar
print
true
false
int
bool
char