Alphabet:
    a. Upper (A-Z) and lower case letters (a-z) of the English alphabet
    b. Underline character '_';
    c. Decimal digits (0-9);
1. Lexic:
    a. Special symbols representing:
        operators: = + ++ += - -- -= * *= / /= == != < <= > >= ! & | &= |=
        separators: \n ( ) { } [ ] space "
        reserved words:
            int, bool, string, input, print, when, otherwise, in, while, each
    b. identifiers:
        - a sequence of letters, digits and underscores, such that the identifier starts with a letter:
            identifier = letter | letter{letterDigitOrUnderscore}
            letterDigitOrUnderscore = lowerOrUpperLetter | digit | "_"
            digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
            lowerOrUpperLetter = lowerLetter | upperLetter
            lowerLetter = "a" | "b" | ... | "z"
            upperLetter = "A" | "B" | ... | "Z"
    c. constants:
        1. int
            intConst = "0" | ["+" | "-"]digit{digitOrZero}
            digitOrZero = "0" | digit
            digit = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
        2. bool
            boolConst = "false" | "true"
        3. string
            stringConst = """{character}"""
            character = lowerOrUpperLetter | symbol
            symbol = "\n" | " " | "." | "!" | "?" | "\"" | "'" | ":" | ";" | "|" | "/" | "\" | "=" | "+" | "(" | ")" | "&" | "*" | "-" | "_" | "=" | "+" | "[" | "]"
                    | "{" | "}" | "<" | ">" | "\t"
            lowerOrUpperLetter = lowerLetter | upperLetter
            lowerLetter = "a" | "b" | ... | "z"
            upperLetter = "A" | "B" | ... | "Z"

```
declaration = type identifier "=" expression "\n"
type = ("bool" | "int" | "string")["[]"]
const = intConst | boolConst | stringConst

expression = "(" expression operator expression ")" | expression operator
expression | expression operator term | "(" expression operator term ")"
| term | "input()"
term = const | identifier
operator = "+" | "-" | "*" | "/" | "==" | "!=" | "<" | "<=" | ">" | ">="
| "!" | "&" | "|"

assignment = ((identifier inplaceOperator expression) | (identifier
selfOperator) | (selfOperator identifier))  "\n"
inplaceOperator = "=" | "+=" | "-=" | "*=" | "/=" | "&=" | "|="
selfOperator = "++" | "--";

compund = {compund} simple "\n"
simple = declaration | assignment | "print(" expression ")" | block

block = when | while | for

when = "when(" expression ")" "{" compund "}" [otherwise]
otherwise = "otherwise"["(" expression ")"] "{" compund "}"
while = "while(" expression ")" "{" compund "}"
for = "(" identifier " in " identifier ")" "{" compund "}"
```

=
+
++
+=
-
--
-=
*
*=
/
/=
==
!=
<
<=
>
>=
!
&
|
&=
|=
\n
space
(
)
[
]
{
}
"
int
bool
string
input
print
when
otherwise
in
while
each