
Lexic.txt

Alphabet:
a. uppercase and lowercase letters from the english alphabet: ([A-Za-z])
b. digits: [0-9]

Lexic:
a. Special symbols, representing:
- operators: + - * / < > <= = >= == != ! && ||
- separators: { } ( ) [ ] ; <space>
- reserved words: start, end, int, str, arr, char, bool, true, false, scan, print, if, elif, else, while, for

b. identifiers
- a sequence of letters and digits, such that the first character is a letter, followed only by letters and then only by digits.

identifier = letter | letter {letter}{digit}
letter = uppercase_letter | lowercase_letter
uppercase_letter = "A" | "B" | . ..| "Z"
lowercase_letter = "a" | "b" | ... | "z"
digit = "0" | non_zero_digit
non_zero_digit = "1" | ... | "9"

c. Constants
1. integer
integer_constant = "0" | ["+" | "-"] non_zero_digit {digit}

2. character
character_constant = 'character'
character = letter | digit

3. string
string_constant = "{character}"

4. boolean
boolean_constant = "true" | "false"

constant = integer_constant | character_constant | string_constant | boolean_constant

---
token.in

(
)
[
]
{
}
;
<space>
+

-
*
/
=
<
>
<=
>=
==
!=
!
&&
||
start
end
int
str
char
bool
true
false
arr
scan
print
if
elif
else
while
for

Syntax.in
---

program = "start" compound_statement "end"

simple_type = "int" | "str" | "char" | "bool"

array_type = simple_type " " "arr" "[" integer_constant "]"

type = simple_type | array_type

declaration = type " " identifier

statement_list = statement | statement statement_list

statement = simple_statement | structure_statement

compound_statement = "{" statement_list "}"

simple_statement = (assignment_statement | io_statement | declaration) ";"

structure_statement = compound_statement | if_statement | while_statement | for_statement

if_statement = "if" condition statement {"elif" condition statement} ["else" statement]

for_statement = "for" "(" "int" integer_assignment_statement ";" condition ";" assignment_statement ")" statement

while_statement = "while" condition statement

expression = [expression ("+" | "-")] term

assignment_statement = integer_assignment_statement | identifier "=" (identifier | character_constant | string_constant | boolean_constant)

integer_assignment_statement = identifier "=" expression

term = term("*" | "/") factor | factor

factor = "(" expression ")" | integer_constant | identifier | indexed_identifier

indexed_identifier = identifier "[" integer_constant "]"

io_statement = ("scan" "(" identifier ")") | ("print" "(" (identifier | Constant) ")")

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

condition = "(" expression relation expression ")"