Lab 1b

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 [ ] { } ( ) " ; , space newline

- reserved words: int, char, string, array, read, print, if, else, while, for

b.Identifiers

-a sequence of letters and digits and _ such that the first character is either _ or a letter

identifier = (letter | "_"){ letter | digit | "_"}

letter = "A" | "B" | ... | "Z" | "a" | "b"... | "z"

digit = "0" | "1" | ... | "9"

c. constants

1.integer

int_constant = "0" | [" + " | " - "] nonzerodigit { "0" | nonzerodigit }

nonzerodigit = "1" | … | "9"

2.character

char_constant="' " letter " ' " | " ' " digit " ' " | " ' " "_" " ' "

3.string

string_constant = " " "{ character } " " "

Tokens list:

+

-

*

/

%

=

==

!=

<

>

<=

>=

&&

||

;

,

{

}

(

)

[

]

space

newline

int

char

string

array

read

print

if

else

while

for

Syntax

program = "{" { statement} "}"

statement = simple_stmt ";" | struct_stmt

simple_stmt = declaration_stmt | array_decl_stmt | assign_ stmt | io _stmt

struct_stmt = if_stmt | while_stmt | for_stmt

declaration_stmt = type comp_identifier_list

array_decl_stmt = "array" type "[" positive_number "]" simple_identifier_list

comp_identifier_list = identifier ["=" expression] {"," identifier ["=" expression]}

simple_identifier_list = identifier {"," identifier}

type = "int" | "char" | "string"

positive_number = [" + "] nonzerodigit { "0" | nonzerodigit }

assign_stmt = identifier "=" expression

expression = int_expression | string_expression

int_expression = int_constant | identifier | ["("] int_expression ("+" | "-" | "*" | "/" | "%") int_expression [")"]

string_expression = string_constant | identifier | string_expression "+" string_expression

io_stmt = "read" "(" identifier ")" | "print" "(" identifier | constant | expression ")"

if_stmt = "if" "(" condition ")" "{" { statement } "}" ["else" "{" { statement} "}"]

condition = expression ("==" | "<" | "<=" | ">" | ">=") expression | ["("] condition ("||" | "&&") condition [")"]

while_stmt = "while" "(" condition ")" "{" { statement } "}"

for_stmt = "for" "(" assign_stmt ";" condition ";" assign_stmt ")" "{" { statement } "}"