LEXIC

program = "{"cmpdstmt "}"

cmpdstmt = declist statementlist

declist = declaration | declaration ";" decllist

declaration = ("var" type id)

type = "int" | "character" | "string" | arraytype

arraytype = "array""("type")"

statementlist = "[" statement | stuctstat "]"

statement = simplestat | structstat

simplestat = assing | iostat

iostat = readStatement | writeStatement

assing = id "=" exp

exp = term {("+" | "-") exp} | constant

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

factor = "(" exp ")" | IDENTIFIER | int

readStatement = "read" "(" id ")"

writeStatement = "write" "("id")"

ifstmt = "if" condition ":" statementlist["else" statementlist]

do_while= "do_while" condition ":" statementlist

do_steps= "do_steps" forcondition ":" statementlist

forcondition = assign"," id "," number

condition = exp RELATION exp{comprel condition}

RELATION = "<" | "<=" | "==" | "!=" | ">=" | ">"

comprel = "and" | "or"

SYNTAX

program = "{"cmpdstmt "}"

cmpdstmt = declist statementlist

declist = declaration | declaration ";" decllist

declaration = ("var" type id)

type = "int" | "character" | "string" | arraytype

arraytype = "array""("type")"

statementlist = "[" statement | stuctstat "]"

statement = simplestat | structstat

simplestat = assing | iostat

iostat = readStatement | writeStatement

assing = id "=" exp

exp = term {("+" | "-") exp} | constant

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

factor = "(" exp ")" | IDENTIFIER | int

readStatement = "read" "(" id ")"

writeStatement = "write" "("id")"

ifstmt = "if" condition ":" statementlist["else" statementlist]

do_while= "do_while" condition ":" statementlist

do_steps= "do_steps" forcondition ":" statementlist

forcondition = assign"," id "," number

condition = exp RELATION exp{comprel condition}

RELATION = "<" | "<=" | "==" | "!=" | ">=" | ">"

comprel = "and" | "or"

TOKENS

=

+

-

*

/

%

&&

||

<

>

<=s

=>

!=

==

id

var

int

string

char

test

else

then

read

write

do_while

do_steps

array