Language Specification

1. Language definition:

1.1 Alphabet:

1.1 a. Upper(A-Z) and lower case letters(a-z) of the English alphabet

b. Unerline character '_'

c. Decimal digits (0-9)

Lexic:

a.Special symbols, representing:

- operators:

arithmetic: +,-,*,/,%

assignment: =

bitwise logic: ~,&,|,^

bitwise shifts: <<, >>

boolean logic: !,&&,||

conditional evaluation: ?:

increment and decrement: ++, --

equality testing: ==, !=

order relations: <, <=, >, >=

sequencing: ,

-separators:

; : [] {}

-reserved words:

integer

float

char

string

boolean

true

false

if

then

else

for

range

while

do

read

write

begin

end

b. Identifiers

-a sequence of letters and digits, such that the first charater is a letter; the rule is:

identifier ::= letter | _letter{letter}{digit}

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

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

c. Constants

1. Integer:

integer := "0" | ["+"|"-"] non_zero_digit{digit}

non_zero_digit := "1"|"2"|...|"9"

2. Float:

float := integer [, digit {digit}]

3. Character:

char := 'letter'|'digit'

4. String:

string := " character{character} "

5. Boolean :

boolean := "true" | "false"

true := "1"

false := "0"

2.2 Syntax:

The words - predefined tokens are specified between " and ":

a) Sintactical rules:

program ::= "begin" statement_list "end"

statement_list ::= statement ";" {statement}

statement ::= declaration | simple_statement | struct_statement

declaration ::= type identifier

type ::= primary_type | array_type

primary_type ::= "integer" | "float" | "char" | "string" | "boolean"

array_type ::= primary_type "["nr"]"

operation ::= "+" | "-" | "*" | "/" | "%" | "^" | "&"

relation ::= "<" | "<=" | "==" | "!=" | ">=" | ">" | "&&" | "||"

condition ::= expression relation expression

simple_statement ::= assignment | io_statement

assignment ::= identifier "=" expression

expression ::= term | expression operation term

term ::= identifier | nr

io_statement ::= "read" | "write" "(" identifier ")"

compound_statement ::= "{" statement_list "}"

struct_statement ::= if_statement | while_statement | for_statement | switch_statement

if_statement ::= "if" condition ":" "then:" statement "else:" statement

while_statement ::= "while" condition ":"

for_statement ::= "for" assignment; condition; assignment; ":" statement

switch_statement ::= "switch" condition case_statement{case_statement} "default" ":' statement_list

case_statement ::= "case" ":" statement_list "break;"

arrayToString ::= identifier "." "toString" "(" ")" ";"

arrayAdd ::= identifier "." "add" "(" "term" ")" ";"

arrayDelete ::= identifier "." "delete" "(" "term" ")" ";"

arrayLength ::= identifier "." "length" "(" ")" ";"

b) lexical rules:

identifier ::= letter | _letter{letter}{digit}

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

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