Student: Condrea Adrian
Lab 1b
Lexic.txt

Specification of the programming language

Alphabet:
letter = “a” | “b” | … | “z” | “A” | “B” | … | “Z”
underline = "_"
nonZeroDigit = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
digit = “0” | nonZeroDigit

The lexic of the programming language contains:

Special symbols:
operator = “+” | “-“ | “*” | “/“ | “<=“ | “>=“ | “==“ | “!=“ | “<“ | “>” | “%” | “!”
separator = “[“ | “]” | “(“ | “)” | “<“ | “>” | “,” | "\"" | “’” | “ “ | “\n” | “.”

Reserved words:
reservedWords = “startProgram” | “input” | “declare” | “typeCheck” | “typeDefine” | “check” | “else” | “output” | “endProgram” | “assign” | “loop” | “breakLoop” | “integer” | “boolean” | “string” | “char” | “array”

Identifiers
Are a sequence of letters and digits.

Identifier rules:
An identifier consists of a letter or a sequence of any letters, digits, or underline
The first character is always a letter
The number of characters is <= 256
Identifiers can not be the same as reserved words

identifier = letter | letter {(letter, digit, underline)}

Constants
Integer - rule:
noConst = “0” | [“-“] nonZeroDigit {digit}

Boolean - rule: boolConst = “true” | “false”

Char - rule:
charConst = 'letter' | 'digit' | 'underline' | 'separator'

String - const:
stringConst = "{charConst}"

Array -const:
arrayConst = [const]

const = noConst | boolConst | charConst | stringConst | arrayConst

Student: Condrea Adrian
Lab 1b
syntax.in

program = “startProgram<” statements “>”

statements = {statement} [endProgramStatement]

statement = inputStatement | declarationStatement |
typeCheckStatement | typeDefineStatement |

checkStatement | outputStatement | assignStatement | loopStatement

endProgramStatement = “endProgram”

inputStatement = “input(“ [identifier] “)”

spaces = {“ “}

declarationStatement = “declare(“ type “,” spaces identifier “)”

type = “integer” | “boolean” | “string” | “char” | “array” |
userDefinedType

typeCheckStatement = “typeCheck(“ type “,” spaces identifier“)”

typeDefineStatement = “typeDefine(” identifier
“)<“ {declarationStatement} “>”

checkStatement = “check(“ condition “)<” statements “>”
[elseStatement]

elseStatement = “<“ statements “>”

condition = identifier | const relations identifier | const

relations = “==“ | “!=“ | “<=“ | “>=“ | “<“ | “>”

outputStatement = “output(“ {stringconst | identifier } “)”

assignStatement = “assign(“ identifier constant | identifier “)”

loopStatement = “loop(“ [condition] “)<“ {statements} [“breakLoop”]
“>”

Student: Condrea Adrian
Lab 1b
token.in

+
-
*
/
%
<=
>=
==
!=
!
<
>
[
]
(
)
<
>
,
"
'
.
\n
startProgram
input
declare
typeCheck
typeDefine
check
else
output
endProgram
assign
loop
breakLoop
integer
boolean
string
char
array