**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. Underline characters ‘\_’;

c. Decimal digits (0-9);

Lexic:

a.Special symbol, representing:

**-** operators + - \* / < <= = >= >> << == && || % ! != ^

- separators [ ] { }( ) : ; space \n.

- reserved words: int, char, bool, array, float, struct, if, else, for, while, string

b.Identifiers:

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

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

letter = “a” | “b” | … “z“ | "A" | "B" | . ..| "Z"

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

c.Constants:

1.integer - rule:

nonzero\_digit = “1” | … | “9”

no\_constant = 0 | [ “+” | “-” ] nonzero\_digit { no }

no = digit { no }

2.character:

character:=” ' ”letter” ' ”|” ' “digit” ' “

3.string:

string = “char { char }”

char = letter| digit

2.2 Syntax:

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

1. Syntactical rules:

program = “begin” cmpdstmt “end”

cmpdstmt = {line} cmpdstmt

line = decllist | stmt

decllist = decl | decl “;” decllist

decl = type identifier | type identifier “=” expression

type = “int” | “bool” | “float” | “char” | “string”

arraydecl = type identifier “[“ no “]”

no = digit { no }

stmtlist= stmt “;” stmtlist | stmt

stmt = simplstmt | structstmt

simplstmt = assignstmt | iostmt

assignstmt = identifier “ = ” expression

expression = [“!”] ( term | expression operation expression | “(“ expression op expression “)” )

operation = “ + ” | “ - ” | “ \* ” | “ / ” | “ % ” | “ ^ ”

relation = “<” | “>” | “<=” | “>=” | “==” | “&&” | “||” | “!” | “!=”

term = identifier| no

iostmt = “READ” ‘(“identifierlist“)” | “WRITE” “(”expression“)”

identifierlist = identifier | identifier “,” identifierlist

structstmt = ifstmt | whilestmt | forstmt | cmpdstmt

Ifstmt = “if” condition cmpdstmt [ “else” cmpdstmt ]

condition = “(“ expression relation expression “)”

whilestmt = “while” condition cmpdstmt

forstmt = “for” “(“ assignstmt “;” condition “;” assignstmt “)” cmpdstmt

1. lexical rules:

Identifier = letter{letter | digit | “\_”}

letter = “a” | “b” | … | “z” | "A" | "B" |...| "Z"

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

relation = "<" | "<=" | "==" | "!=" | ">=" | ">" | “&&” | “||”

string = char { string }

char = letter | digit

The tokens are codified according to the following table:

- identifiers - code 0

- constants - code 1

- reserved words: each word has its own code

- operators: each operator has its own code

- separators: each separator has its own code

Codification table:

|  |  |
| --- | --- |
| token type | code |
| identifier | 0 |
| constant | 1 |
| int | 2 |
| char | 3 |
| bool | 4 |
| array | 5 |
| float | 6 |
| struct | 7 |
| if | 8 |
| else | 9 |
| for | 10 |
| while | 11 |
| begin | 12 |
| end | 13 |
| read | 14 |
| write | 15 |
| [ | 16 |
| ] | 17 |
| { | 18 |
| } | 19 |
| ( | 20 |
| ) | 21 |
| : | 22 |
| ; | 23 |
|  | 24 |
| , | 25 |
| . | 26 |
| + | 27 |
| / | 30 |
| < | 31 |
| > | 32 |
| <= | 33 |
| = | 34 |
| >= | 35 |
| == | 36 |
| && | 37 |
| || | 38 |
| % | 39 |
| ! | 40 |
| != | 41 |
| ^ | 42 |
| string | 43 |
| - | 44 |
| \* | 45 |