Language Specification

1. **Language Definition**
   1. Alphabet:
      1. Upper (A-Z) and lower case letters (a-z) of the English alphabet;
      2. Underline character: ‘\_’;
      3. Decimal digits: (0-9).
   2. Lexic:
      1. Special Symbols:

* Operators:
  + Arithmetic:
    - Additive: +, -
    - Multiplicative: \*, /, %
  + Relational:
    - Comparison:<, >, <=, >=
    - Equality: ==, !=
  + Logical: &&, ||
  + Assignment: =
  + Increment: ++
  + Decrement: --
  + Sequence: ,
* Separators: [ ] { } ( ) : ; space
* Reserved words: array, integer, string, if, do, else, while, write, read, then, for, true, false, char, new, const, begin, end, of, var.
  + 1. Identifiers
    2. Constants:
* Integer:
  + <integer> ::= 0|<sign><non\_zero\_digit><digits>|<sign><non\_zero\_digit>|<non\_zero\_digit><digits>|<non\_zero\_digit>
  + <sign> ::= +|-
  + <digits> ::= <digit>|<digit><digits>
  + <non\_zero\_digit> ::= 1|2|..|9
  + <digit> ::= 0|<non\_zero\_digit>
* String:
  + <string> ::= <empty\_string>|<char>|<char><string>
  + <empty\_string> ::= ‘‘
  + <char> ::= a|b|c|..|z|A|B|..|Z|0|1|2|..|9
* Boolean:
  + <boolean> ::= true|false

1. **Syntax:**
   1. Syntactical rules:
      * <program> ::= <compound\_stmt>.
      * <array> ::= new Array[<integer>] of <type>
      * <array\_declaration> ::= var <identifier> : <array>
      * <type> ::= integer|string|boolean|char
      * <compound\_stmt> ::=

begin {<statement\_list>}end

* + - <statement\_list> ::= <statement>;| <statement> ; <statement\_list>
    - <statement> ::= <assign\_stmt>|<io\_stmt>|<if\_stmt>|<while\_stmt>|<for\_stmt>|<declaration>
    - <declaration> ::= var <identifier> : <type>|<array\_declaration>
    - <assign\_stmt> ::= <identifier> = <expression>
    - <expression> ::= <term>|<identifier>|<term><op1><expression>
    - <op1> ::= +|-
    - <term> ::= <term><op2><factor>|<factor>
    - <op2> ::= \*|/
    - <factor> ::= (<expression>)|<identifier>
    - <io\_stmt> ::= <read>|<write>
    - <read> ::= read(<identifier>)
    - <write> ::= write(<identifier>) | write(<exression>)
    - <if\_stmt> ::= if (<condition>) then {<statement\_list>}|if (<condition>)then {<statement\_list>} else {<statement\_list>}
    - <condition> ::= <expression><relation><expression>
    - <while\_stmt> ::= while (<condition>) do {<statement\_list>}
    - <do\_while> ::= do {<statement\_list>} while (<condition>)
    - <for\_stmt> ::= for (<assign\_stmt>; <condition>; <expression>) begin <statement\_list> end

* 1. Lexical rules:
     + <identifier> ::= \_<letters>|\_<letters><digits>|<letters><digits>|<letters>
     + <letters> ::= <letter>|<letter><letters>
     + <letter> ::= a|b|c|..|z|A|B|..|Z
     + <relation> ::= <|>|=|<=|>=|<>|!=|==

1. **Codification table:**

|  |  |
| --- | --- |
| **Token Type** | **Code** |
| **identifier** | **0** |
| **const** | **1** |
| **var** | **2** |
| **begin** | **3** |
| **end** | **4** |
| **new** | **5** |
| **Array** | **6** |
| **of** | **7** |
| **integer** | **8** |
| **char** | **9** |
| **string** | **10** |
| **boolean** | **11** |
| **while** | **12** |
| **do** | **13** |
| **write** | **14** |
| **read** | **15** |
| **if** | **16** |
| **then** | **17** |
| **else** | **18** |
| **for** | **19** |
| **:** | **20** |
| **,** | **21** |
| **<** | **22** |
| **>** | **23** |
| **<=** | **24** |
| **>=** | **25** |
| **<>** | **26** |
| **==** | **27** |
| **+** | **28** |
| **-** | **28** |
| **++** | **30** |
| **--** | **31** |
| **=** | **32** |
| **\*** | **33** |
| **/** | **34** |
| **&&** | **35** |
| **;** | **36** |
| **!** | **37** |
| **||** | **38** |
| **&** | **39** |
| **:=** | **40** |
| **!=** | **41** |
| **%** | **42** |
| **{** | **43** |
| **}** | **44** |
| **(** | **45** |
| **)** | **46** |
| **[** | **47** |
| **]** | **48** |
| **\_** | **49** |
| **.** | **50** |