**(C++ subset) Minilanguage Specification**

Language Definition

Alphabet:

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

b. Decimal digits (0-9);

Lexic:

a. special symbols, representing:

- operators + - \* / = < <= == >= << >>

- separators [ ] { } , : ; space

- reserved words: int char if else while

b. identifiers

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

<identifier> ::= <letter> | <letter><sequence>

<letter> ::= a| b | … | z | A | B | . ..| Z

<digit> ::= 0 | 1 |...| 9

<sequence> ::= <letter>|<digit>|<letter><sequence>|<digit><sequence>

c. constants

**1.integer**

<integer> ::= 0 | <number> | <sign><number>

<number> ::= <nz\_digit> | <nz\_digit><digit\_seq>

<digit\_seq> ::= <digit> | <digit><digit\_seq>

<nz\_digit> ::= 1 | 2 | … | 9

<sign> ::= + | -

**2.character**

<character> ::= '<letter>' | ’<digit>’ | ‘<symbol>’

<symbol> ::= ?|!|.|,|:|;

**3.array**

<array> ::= {<set\_char>} | {set\_int}

<set\_int> ::= <integer> | <integer>,<set\_int>

<set\_char> ::= <char> | <char>,<set\_char>

Syntax:

a) Sintactical rules:

<program> ::= int main() <cmpdstmt>

<declaration> ::= <decl> | <decl> = <constant>

<constant> ::= <integer> | <char> | <array>

<type> ::= char | int

<decl> ::= <type> <identifier> | <arraydecl>

<arraydecl> ::= <type> <identifier>[ <number> ]

<cmpdstmt> ::= { <stmtlist> }

<stmtlist> ::= <simplestmt> | <simplestmt><stmtlist>

<simplestmt> ::= <assignstmt>; | <iostmt>; | <declaration>; | <whilestmt> | <ifstmt>

<stmt> ::= <simplestmt> | <cmpdstmt>

<iostmt> ::= cin >> <IDENTIFIER> | cout << <IDENTIFIER> | cout << endl

<ifstmt> ::= if( <condition> ) <stmt> | if( <condition> ) <stmt> else <stmt>

<assignstmt> ::= <IDENTIFIER> = <expression>

<expression> ::= <expression> <arith\_op2> <expression> |<term>

<term> ::= <factor> | <term> <arith\_op1> <factor>

<factor> ::= (<expression>) | <integer> | <identifier> | <identifier> [<index>]

<index> ::= <number> | 0

<arith\_op1> ::= \* | /

<arith\_op2> ::= + | -

<whilestmt> ::= while (<condition>) <stmt>

<condition> ::= <expression> <RELATION> <expression>

b) lexical rules:

<IDENTIFIER> ::= <letter> | <letter> <sequence>

<letter> ::= a | b | … | z | A | B | … | Z

<digit> ::= 0| 1 |...| 9

<RELATION> ::= < | == | != | >

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:

-------------------------

| Token type | code |

-------------------------

| identifier | 0 |

-------------------------

| constant | 1 |

-------------------------

| int | 2 |

-------------------------

| char | 3 |

-------------------------

| cin | 4 |

-------------------------

| cout | 5 |

-------------------------

| endl | 6 |

-------------------------

| while | 7 |

-------------------------

| if | 8 |

-------------------------

| else | 9 |

-------------------------

| : | 10 |

-------------------------

| ; | 11 |

-------------------------

| , | 12 |

-------------------------

| + | 13 |

-------------------------

| \* | 14 |

-------------------------

| - | 15 |

-------------------------

| / | 16 |

-------------------------

| { | 17 |

-------------------------

| } | 18 |

-------------------------

| ( | 19 |

-------------------------

| ) | 20 |

-------------------------

| [ | 21 |

-------------------------

| ] | 22 |

-------------------------

| < | 23 |

-------------------------

| > | 24 |

-------------------------

| == | 25 |

-------------------------

| = | 26 |

-------------------------

| != | 27 |

-------------------------

| >> | 28 |

-------------------------

| << | 29 |

-------------------------