CANDET ANDREI GABRIEL

932

LAB 1b

Alphabet:

1. Upper (A-Z) and lower case letters (a-z) of the English alphabet
2. Underline character "\_"
3. Decimal digits (0-9)

1. Lexic
2. Special symbols, representing:

* Operators + - \* / := < <= = >= === ~
* Separators [ ] { } : ; *space*
* Reserved words: **array, map, const, do, else, if, int, elif while, for, range, class, struct, string, float, char, boolean, input, print, return, fun, key, value, main, entry**

1. Identifiers

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

Identifier ::= letter | letter{letter}{digit}

Letter ::= "A" | "B" | … | "Z" | "a" | "b" | … | "z"

Digit ::= "0" | "1" | … | "9"

1. Constants
   1. Integer - rule:

integer\_const = "0" | [" + " | " - "] nonzerodigit { "0" | nonzerodigit }

nonzero\_digit = "1" | … | "9"

* 1. Character

character:= 'letter | digit | SPACE '

1. String

String\_const ::= " char{str} "

1. Syntax

**Predefined tokens are emphasized.**

Program ::= **entry** cmpdstmt **;**

Type ::= **bool** | **int** | **char** | **string**

Assignstmt ::= IDENTIFIER **=** expression

decl ::= declstmt | declasgnstmt

Declstmt ::= type IDENTIFIER

Declasgnstmt ::= type IDENTIFIER **=** expression

Cmpdstmt ::= **{** stmtlist **}**

Stmtlist ::= stmt | stmt **;** stmtlist

Stmt ::= simplstmt | structstmt

Simplstmt ::= assignstmt | decl | iostmt

Iostmt ::= **INPUT(** IDENTIFIER **) | WRITE(** IDENTIFIER **)**

Value ::= integer\_const | character | string\_const | IDENTIFIER | arrayAccess | expression

Expression ::= value arithmetic\_ops **(** expression **)**

Arithmetic\_ops ::= + | - | / | \*

Term ::= term **\*** factor | factor | arrayAccess

Factor ::= **(** expression **)** | IDENTIFIER

Structstmt ::= cmpdstmt | ifstmt | whilestmt

Whilestmt ::= **while (** condition **)** cmpdstmt

Ifstmt:**:= if(** CONDITION **)** cmpdstmt **else** cmpdstmt

Condition ::= expression RELATION expression

Relation ::= **<** | **<=** | **==** | **>=** | **===** | **>**

Arraydecl ::= type IDENTIFIER **[** number **]**

arrayAccess::= IDENTIFIER[ IDENTIFIER ]

Mapdecl ::= **map{** type : type **}**

mapAccess ::= IDENTIFIER[ IDENTIFIER ]

**TOKEN LIST:**

+ - \* / := < <= = >= === ~ % & ^ **array, map, const, do, else, if, int, elif while, for, range, class, struct, string, float, char, boolean, input, print, return, fun, key, value, main, entry**