Laboratory 1

Mini-Language: Pascal

Identifiers: length at most 8 characters

Symbol Table: separate tables for identifiers, respectively constants

Symbol Table Organization: lexicographically sorted table

Language Specification:

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

* Operators + - \* / := <= >= = <>
* Separators : ; [ ] { } space .
* Reserved words: program, var, integer, char, array, if, then, else, while, do, boolean, end, begin, read, write, div, mod, type, and, or, of, not

1. Identifiers:

* Sequence of letters and digits, with the first character being a letter

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

Letter ::= “A” | … | “Z”

Digit ::= “0” | … | “9”

1. Constants:
2. Integer:

int ::= ([ + | - ] non\_zero{digit}) | “0”

non\_zero ::= “1” | “2” | … | “9”

digit ::= non\_zero | “0”

1. Character

Character ::= ‘letter’ | ‘digit’

1. Syntax:

Predefined tokens are specified between “ ”.

1. Sintactical rules:

Program ::= “VAR” decllist “;” cmpdstmt “.”

Decllist ::= declaration | declaration “;” decllist

Declaration ::= IDENTIFIER “:” type

Type ::= sngtype | arraytype

Sngtype ::= “INTEGER” | “CHAR” | “BOOLEAN” | “STRING”

Arraytype ::= “ARRAY” “[“ nr “]” “OF” sngtype

Cmpdstmt ::= “BEGIN” stmtlist “END”

Stmtlist ::= stmt | stmt “;” stmtlist

Stmt ::= simplestmt | structstmt

Simplestmt ::= assignstmt | iostmt

Assignstmt ::= IDENTIFIER “:=” expression

Expression ::= term | expression “+” term

Term ::= factor | term “\*” factor

Factor ::= “(“ expression “)” | IDENTIFIER

Iostmt ::= (“READ” | “WRITE”) “(“ IDENTIFIER “)”

Structstmt ::= cmpdstmt | ifstmt | whilestmt

Ifstmt ::= “IF” condition “THEN” cmpdstmt [ “ELSE” cmpdstmt ]

Whilestmt ::= “WHILE” condition “DO” compdstmt

Condition ::= expression RELATION expression

1. Lexical rules:

IDENTIFIER ::= letter | letter{letter}{digit}

Letter ::= “a” | “b” | … | “z”

Digit ::= “0” | “1” | … | “9”

Relation ::= “<” | “>” | “<=” | “>=” | “=” | “<>”

The tokens are codified according to the following table:

|  |  |
| --- | --- |
| Token Type | Code |
| identifier | 0 |
| Constant | 1 |
| : | 2 |
| ; | 3( |
| . | 4 |
| , | 5 |
| ( | 6 |
| ) | 7 |
| [ | 8 |
| ] | 9 |
| + | 10 |
| - | 11 |
| \* | 12 |
| = | 13 |
| < | 14 |
| > | 15 |
| := | 16 |
| <= | 17 |
| >= | 18 |
| <> | 19 |
| program | 20 |
| array | 21 |
| of | 22 |
| var | 23 |
| integer | 24 |
| char | 25 |
| boolean | 26 |
| begin | 27 |
| end | 28 |
| read | 29 |
| write | 30 |
| if | 31 |
| then | 32 |
| else | 33 |
| while | 34 |
| do | 35 |
| and | 36 |
| or | 37 |
| not | 38 |
| div | 39 |
| mod | 40 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |