Ομαδική Εργασία – Προηγμένα Θέματα Γλωσσών Προγραμματισμού 2018

Παναγιώτης Κεραμίδης 321/2016067
Χρήστος Λουκόπουλος 321/2016093
Άννα Δερβέναγα 321/2016033

Για ευκολία ανάγνωσης οι απαντήσεις των ερωτημάτων 1 και 2 διατίθενται και σε txt αρχεία.

Ερώτηση 1:

Αρχικά, τα keywords είναι case sensitive εν αντιθέσει με τους identifiers. Επίσης ένας identifier δύναται να ξεκινάει και με (_) εν αντιθέσει με τα keywords που ξεκινάνε μόνο με κεφαλαία γράμματα. Επίσης, καθότι τα keywords είναι σαφώς ορισμένα, μοναδικά και δεσμευμένα από τη γλώσσα μια υλοποίηση θα απαγόρευε σε κάποιον identifier να ονομαστεί ακριβώς όπως ένα keyword.

Ακολουθεί η δήλωση Κανονικών Εκφράσεων:

~~~~~~

CHARACTER_SET -> [!"'#$%^&*()~+\.-,.<>/?\|;[]{}:=`0-9a-zA-Z_] ;

WHITESPACES -> [\t\n] ;

COMMENTS -> '{' .*? ^'\n' '}' ;

IDENTIFIERS -> [a-zA-Z_][a-zA-Z_0-9]+ ;

CONSTANTS -> [-]?[0-9'TRUE''FALSE'ALPHARITHMETIC] ;

ALPHARITHMETIC -> '"' + [a-zA-Z'\”''\\'] + '"' ;

OPERATORS -> [UNARY_MINUS, MULTIPLICATIVE, ADDITIVE, RELATIONAL, LOGICAL, CONCATENATION, ASSIGNMENT] ;

UNARY_MINUS -> '-' ;

MULTIPLICATIVE -> '*' | '/' | '%' ;

ADDITIVE -> '+' | '-' ;

RELATIONAL -> '=', '<>', '<', '>', '<=', '>=' ;

LOGICAL -> 'AND' | 'OR' | 'NOT' ;

CONCATENATION -> '|' ;

ASSIGNMENT -> ':=' ;
~~~~~~

PUNCTUATORS -> [GROUPING, ARRAY, SEP, ID_TYPE, SEP_LIST] ;

GROUPING -> [L_PARENTHESES, R_PARENTHESES] ;

L_PARENTHESES -> '(' ;

R_PARENTHESES -> ')' ;

ARRAY -> [L_BRACK, R_BRACK] ;

L_BRACK -> '[' ;

R_BRACK -> ']' ;

SEP -> ';' ;

ID_TYPE -> ':' ;

SEP_LIST -> ',' ;

Ερώτηση 2:

Απαραίτητη η απαλοιφή κάθε αριστερής αναδρομής:

stmtlist -> stmtlist statement '

stmtlist statement ' -> SEMICOLON stmtlist statement '
| statement
| ε

statement -> lvalue expr '
| expr statement '

lvalue expr ' -> ASSIGN lvalue expr '
| EXIT
| ε

lvalue -> ID
| ID LBRACK index RBRACK

expr statement ' -> IF expr THEN statement
| IF expr THEN statement ELSE statement
| WHILE expr DO statement

| EXIT

Ψευδοκώδικες:

```
function statement1(TOKEN_TYPE token): void
	begin
		lvalue() := expr;
	end

function statement2(TOKEN_TYPE token): void
	begin
		if (expr) then begin
			statement();
		end
	end

function statement3(TOKEN_TYPE token): void
	begin
		if (expr) then begin
			statement();
		end
		else begin
			statement();
		end
	end

function statement4(TOKEN_TYPE token): void
	begin
		while (expr) then begin
			statement();
		end
```

```
        end


function lvalue1(): int
        begin
                return ID;
        end


function lvalue2(): int
        begin
                return ID[index];
        end
```