**Pop Ruxandra -936**

**Specification**

**Alphabet:**

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

b. Decimal digits (0-9);

c. Operators, separators, underscore(”\_”)

**1. Lexic:**

**a. Special symbols, representing:**

- operators + - \* / = < > <= >= != == && || %

- separators [ ] { } ; space

- reserved words:

**int string const do else if program readInt readString while print**

**b. Identifiers**

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

IDENTIFIER = LETTER { LETTER | DIGIT}

LETTER = "A" | "B" | ... | "Z" | ”a” | ”b” | ... | ”z”

DIGIT = "0" | "1" |...| "9"

**c. constants**

**1. integer - rule:**

INT = “0” | [“+” | “-“] NON\_ZERO\_DIGIT {DIGIT}

DIGIT = “0” | NON\_ZERO\_DIGIT

NON\_ZERO\_DIGIT = “1” | … | “9”

**2. string**

      CONSTCHAR = "STRING"

      STRING = {CHAR}

      CHAR = LETTER | DIGIT

**2. Syntax:**

The words - predefined tokens are specified between " and ":

**Sintactical rules:**

PROGRAM = ”START” ”{” STMTLIST ”}”

DECLARATION = TYPE IDENTIFIER

TYPE1 = "INT" | "STRING"

ARRAYDECL = TYPE1 ”[” NR ”]”

TYPE  = TYPE1 | ARRAYDECL

STMTLIST = STMT ”;” {STMT ”;”} // scotem ; - punem doar dupa assign sau dupa io

STMT = SIMPLSTMT | STRUCTSTMT | DECLARATION

SIMPLSTMT = ASSIGNSTMT | IOSTMT

ASSIGNSTMT = IDENTIFIER "=" EXPRESSION

EXPRESSION = EXPRESSION ("+" | ”-”) TERM | TERM

TERM = TERM ("\*" | ”/”) FACTOR | FACTOR

FACTOR = "(" EXPRESSION ")" | IDENTIFIER | CONST

IOSTMT = "READINT" ”(” ”)” | ”READSTRING” ”(” ”)” | "PRINT" "(" IDENTIFIER ")"

STRUCTSTMT = CMPDSTMT | IFSTMT | WHILESTMT

CMPDSTMT = ”{” STMTLIST ”}”

IFSTMT = "IF" ”(” CONDITION ”)” STMT ["ELSE" STMT]

FORSTMT = "FOR" ”(” ASSIGNSTMT ”;” CONDITION ”;” ASSIGNSTMT ”)” STMT

CONDITION = EXPRESSION RELATION EXPRESSION

RELATION = "<" | "<=" | "==" | "!=" | ">=" | ">" | ”&&” | ”||” // problem

**Tokens**

**int string const do else if program readInt readString while print [ ] { } ; space + - \* / % = > < <= >= == != && ||**

**Lab1 – revised**

**Lab 1 – FLCD**

Pop Ruxandra Paula

936

**P1 – max of three numbers**

int num1;

num1 = readInt();

int num2;

num2 = readInt();

int num3;

num3 = readInt();

if (num1 >= num2 && num1 >= num3) {

return num1;

}

if (num2 >= num1 && num2 >= num3) {

return num2;

}

if (num3 >= num1 && num3 >= num2) {

return num3;

}

**P2 – check if number is prime**

int nr ;

nr = readInt();

string isPrime;

isPrime = “Number is prime”;

string isNotPrime;

isNotPrime = “Number is not prime”;

int d;

for(d = 2; d <nr/2; d=d+1) {

if (nr % 2 == 0) {

print(isNotPrime);

}

}

print(isPrime);

**P2err – check if number is prime**

int nr ;

nr = readInt();

string isPrime ;

isPrime = “Number is prime; // lexical error

string isNotPrime ;

isNotPrime = “Number is not prime”;

int d#; // lexical error

for(d = 2; d <nr/2; d=d+1) {

if (nr % 2 == 0) {

print(isNotPrime);

}

}

print(isPrime);

**P3 - compute the sum of n numbers**

int n ;

n = readInt();

int i;

int[100] array;

int sum;

sum = 0;

// read numbers

for (i=0; i<n; i=i+1) {

int m;

m = readInt();

array[i] = m;

}

for (i=0; i<n; i=i+1) {

sum += array[i];

}

return sum;