Lab 1a+b LFTC

Popa Alex Ovidiu

936/1

LEXIC

Alphabet:

a. [A-Za-z]

b. [0-9]

c. Underscore ('_')

d. All characters which are used in operators, separators etc (<,=,{ a.s.o.)

Lexic:

a.Special symbols, representing:

- operators: + - * / = < <= = >= == >> << [ ] ! != and or xor

- separators { } ( ) . , : ; <space>

- reserved words:

  number, array, std::cin, std::cout, if, else, for, while, go, string

b.identifiers

-a sequence of letters and digits, such that the first character is a letter, and intertwining is allowed (I.e. a2a,ba3a)

identifier = letter {letter|digit}

letter = capital_letter | small_letter

capital_letter = "A" | "B" | . ..| "Z"

small_letter = "a" | "b" | ... | "z"

digit = "0" | non_zero_digit

non_zero_digit = "1" | ... | "9"

c.constants

1.integer - rule: doesn't allow things like -0, 001 etc

    integer = "0" | ["+" | "-"] non_zero_digit{digit}

2.character

  character=”letter”|”digit”

3.string

    string=’{letter|digit}’

CONSTANT = integer | character | string

Tokens list

(

)

[

]

{

}

;

:

<space>

.

,

+

-

*

/

=

<

>

<=

>=

==

!=

!

>>

<<

and

array

else

for

go

if

number

or

std::cin

std::cout

string

while

xor

Syntax

program = "go" cmpdstmt

declaration = type " " IDENTIFIER

simpletype = "number" | "string"

arraydecl = simpletype " " "array" "[" integer "]"

type = simpletype|arraydecl

cmpdstmt = "{" stmtlist "}"

stmtlist = stmt | stmt ";" stmtlist

stmt = simplstmt | structstmt

simplstmt = (assignstmt | iostmt | declaration) “;”

structstmt = cmpdstmt | ifstmt | whilestmt | forstmt

ifstmt = "if" condition stmt ["else" stmt]

forstmt = "for" forheader stmt

forheader = "(" "number" assignstmt ";" condition ";" assignstmt ")"

whilestmt = "while" condition stmt

assignstmt = IDENTIFIER "=" expression

expression = [expression("+"|"-")] term

term = term("*" | "/") factor | factor

factor = "(" expression ")" | int | IDENTIFIER | Indexedidentifier

Indexedidentifier = IDENTIFIER “[“ integer “]”

iostmt = ("std::cin" ">>" IDENTIFIER) | ("std::cout" "<<" (IDENTIFIER | CONSTANT))

condition = "(" expression RELATION expression ")"

RELATION = "<" | "<=" | "==" | "!=" | ">=" | ">"

Lab1a Updated

P1. Max of 3 numbers

```
go{
        number a;
        number b;
        number c;
        Std::cin>>a;
        Std::cin>>b;
        Std::cin>>c;
        number max ;
        If(a>b and a>c){
                Max = a;
        }
        If(b>a and b>c){
                Max=b;
        }
        If(c>a and c>b){
                Max=c;
        }
        Std::cout<<max;
}
```

P1err. Max of 3 numbers- lexical error at number 5$a, lexical error at message (unclosed apostrophe)

```
go{
        number 5$a;
        number b;
        number c;
        Std::cin>> 5$a;
        Std::cin>>b;
        Std::cin>>c;
        number max ;
        If(5$a >b and 5$a >c){
```

```
        Max = 5$a;
    }
    If(b> 5$a and b>c){
        Max=b;
    }
    If(c> 5$a and c>b){
        Max=c;
    }
    string message;
    message='number is;
    Std::cout<<message;
    Std::cout<<max;
}
```

P2. Sum of positive numbers in an array

```
go{
    number array[10] arr;
    number size;
    Std::cin>>size;
    number sum;
    sum=0;
    For (I=0,I<size;I=I+1){
        Std::cin>>arr[I];
        If (arr[I]>0){
            sum = sum + arr[I];
        }
    }
    Std::cout<<sum;
}
```

P3. Check if a number is prime or not

```
go{
    Number n;
    Std::cin>>n;
    Number ok;
    ok=1;
    If (n<2 or n>2 and n%2==0){
        ok=0;
    }
    For(d=3;d*d<=n;d=d+2){
        If (n%d==0){
            ok=0;
        }
    }
    If(ok==1){
        Std::cout<<'prime';
    }
    Else {
        Std::cout<<'not prime';
}
```