Lab1b

Lexic.txt

Alphabet :

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

b. Underline character '_'

c. Decimal digits (0-9)

d. Special Characters

1.Lexic

a) Special symbols, representing:

-operators + - * / = <= == >=

-separators [] {} , space

-reserved words: array const if else for read print then execute

b) Identifiers

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

Identifier = sletter {letter | digit} letter = “A”
| “B” | .. |”Z” | “a” | “b” |… |”z” sletter =
“a”|….|”z” digit = “0”| “1”|….|”9”

c) Constants

1. Integer – rule :
Number_Const ::= 0 | <number> | <sign><number>
<number> ::= <nzDigit> | <nzDigit><digitSequence>
<digitSequence> ::= <digit> | <digit><digitSequence>
<nzDigit> ::= 1 | … | 9
<digit> ::= 0 | <nzDigit>
<sign> ::= - | +

2. Character
char_const = " ' " {letter | digit | special_symbol} " ' "
special_symbol = " "

3. String

String constant:
String_const = " {letter | digit | special_symbol} "

2.Syntax

File Syntax.in

program = decllist cmpstmt
decllist = declaration | declaration decllist
declaration = type listIDENT “ ;”
listIDENT = IDENTIFIER | IDENTIFIER “ , ”
listIDENT type1 = “number” | “string” | “char”
arraydecl = “array”” ””[“ type1 “]” ” “[“ “nr” “]”
type = type1|arraydecl
cmpdstmt = “{ “ stmtlist “}”
stmlist = stmt | stmt stmtlist
stmt = simplstmt” ;” | structstmt
simplstmt = assignstmt | iostmt
assignstmt = IDENTIFIER “ = ” expression

expression = expressionplus | espressionminus

expressionplus = expressionplus " + " term | term

expressionminus = expressionminus " - " term |

term term = termmul|termdiv|termmod

termmul = termmul " * " factor | factor

termdiv = termdiv " / " factor | factor

termmod = termod “ % ” factor |factor

factor ::= "(" expression ")" | list

iostmt = “read ” “(“ IDENTIFIER “)” | “print ” “(“ list “)”

list = IDENTIFER | const | array elem

structstmt = cmpdstmt | ifstmt |forstmt

ifstmt = “if ”(“ condition “) ” “then ” stmt “[“ “ else ” stmt ”]”
forstmt = “for”” ””(“” ”assignstmt “;” condition “;”” ”assignstmt” ””)”” ””execute”

stmt

condition = expression‘’ ‘’RELATION’’ ‘’expression

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

P1.

```
number a ;
number b ;
number c ;
{
read ( a ) ;
read ( b ) ;
read ( c ) ;
number max ;
max = a ;
 if ( max  <  b )  then
max = b ;
 if ( max < c ) then  max = c ;
print ( max ) ;
}
```

P2.

```
number a ;
number i ;
number divizori ;
{
read ( a ) ;
divizori = 0 ;
for ( i = 2 ; i < a; i = i + 1 )
execute  if ( a % i == 0 ) then
divizori = divizori + 1 ;
 if ( divizori > 0 ) then
print ( “a is not prime” ) ;
else print ( “a is prime) ;
}
```

P3.

```
number n ;
number a ;
number sum ;
{
```

```
read ( n ) ;
sum = 0 ;
for (number i = 1 ; i <= n ; i = i + 1 ) execute
{
read ( a ) ;
sum = sum + a ;
}
print ( sum );
}
```

P1err.

```
string s1 ;
string $s2 ; // identifiers must start with a leter
read ( s1 ) ;
Read ( s2 ) ;
print ( s1 + s2 ) ;
}
```

Token.in

+ - * / = <= == >= [] {} , array const if else for read print then execute number string char