Specification (file 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);

Lexic:

a.Special symbols, representing:

- operators & @ $ # {} <<< >>>

- separators [ ] : . space

- reserved words:

    arry chr cnst do 3153 17 dec in out ret -> loop flo

b.identifiers

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

    identifier ::= letter | letter{letter}{digit}

    letter ::= "A" | "B" | . ..| "Z"

    digit ::= "0" | "1" |...| "9"

c.constants

1.integer - rule:

      noconst:="+"no|"-"no|no

      no:=digit{no}

2.character

    character:='letter'|'digit'

3.string

      constchar:="string"

      string:=char{string}

      char:=letter|digit

2. Syntax:

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

Sintactical rules: (file Syntax.in)

program ::= decllist "." cmpdstmt "."

decllist ::= declaration | declaration "." decllist

declaration ::= type IDENTIFIER "."

type1 ::= "dec" | "chr" | "flo"

arraydecl ::= "arry" "[" nr "]" "OF" type1

type ::= type1|arraydecl

cmpdstmt ::= "[" stmtlist "]"

stmtlist ::= stmt | stmt "." stmtlist

stmt ::= simplstmt | structstmt

simplstmt ::= assignstmt | iostmt

assignstmt ::= IDENTIFIER "{}" expression

expression ::= expression "&" term | expression "@" term | term

term ::= term "$" factor | term "#" factor | factor

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

iostmt ::= "in" | "out" "<" IDENTIFIER ">"

structstmt ::= cmpdstmt | ifstmt | whilestmt | loopstmt

ifstmt ::= "17" condition ":" stmt ["3153" stmt]

whilestmt ::= "woah" condition "DO" stmt

loopstmt ::= "loop" IDENTIFIER "->" IDENTIFIER ":"

condition ::= expression RELATION expression

RELATION ::= "<<<" | "<<<{}" | "{}" | ">>>{}" | ">>>"

.
{}
dec
flo
chr
in
out
loop
woah
[ ]
<
>
ret
&
@
$
#
^x
<<<
>>>
17
3153
->
arry
cnst
0
1
2
3
4
5
6
7
8
9
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t

v
u
w
x
y
z
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
V
U
W
X
Y
Z