Pcl4

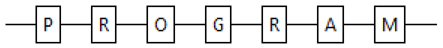
Rules

program Top

Text notation:

```
program : programHeader block '.' ;
```

Visual notation:

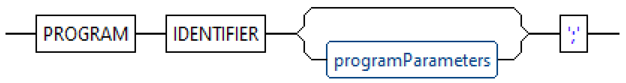


programHeader Top

Text notation:

```
programHeader : PROGRAM IDENTIFIER programParameters? ';' ;
```

Visual notation:

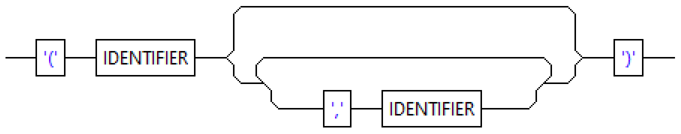


programParameters Top

Text notation:

```
programParameters : '(' IDENTIFIER ( ',' IDENTIFIER )* ')' ;
```

Visual notation:

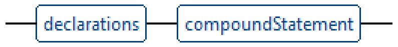


block Top

Text notation:

```
block : declarations compoundStatement ;
```

Visual notation:

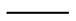


declarations Top

Text notation:

```
declarations : ;
```

Visual notation:

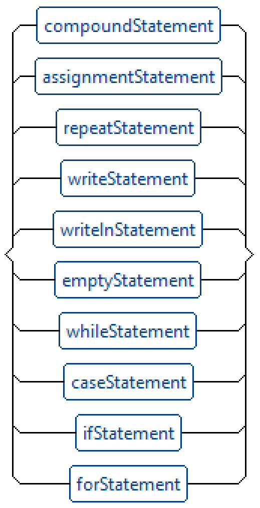


statement Top

Text notation:

```
statement : compoundStatement | assignmentStatement | repeatStatement | writeStatement | writelnStatement | emptyStatement | whileStatement
| caseStatement | ifStatement | forStatement ;
```

Visual notation:

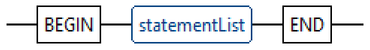


compoundStatement Top

Text notation:

```
compoundStatement : BEGIN statementList END ;
```

Visual notation:

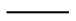


emptyStatement Top

Text notation:

```
emptyStatement : ;
```

Visual notation:

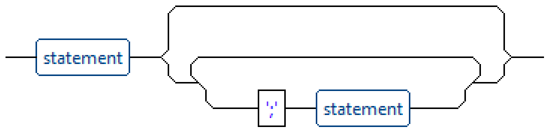


statementList Top

Text notation:

```
statementList : statement ( ';' statement )* ;
```

Visual notation:

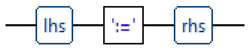


assignmentStatement Top

Text notation:

```
assignmentStatement : lhs ':=' rhs ;
```

Visual notation:

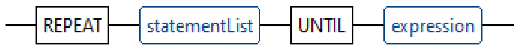


repeatStatement Top

Text notation:

```
repeatStatement : REPEAT statementList UNTIL expression ;
```

Visual notation:

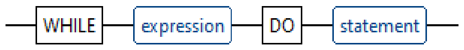


whileStatement Top

Text notation:

```
whileStatement : WHILE expression DO statement ;
```

Visual notation:

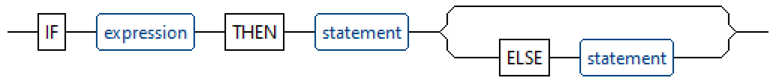


ifStatement Top

Text notation:

```
ifStatement : IF expression THEN statement (ELSE statement)? ;
```

Visual notation:

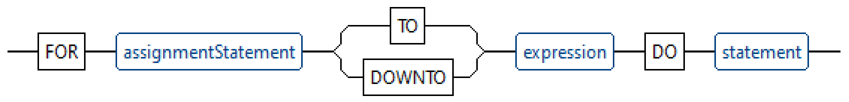


forStatement Top

Text notation:

```
forStatement : FOR assignmentStatement (TO | DOWNTO) expression DO statement ;
```

Visual notation:

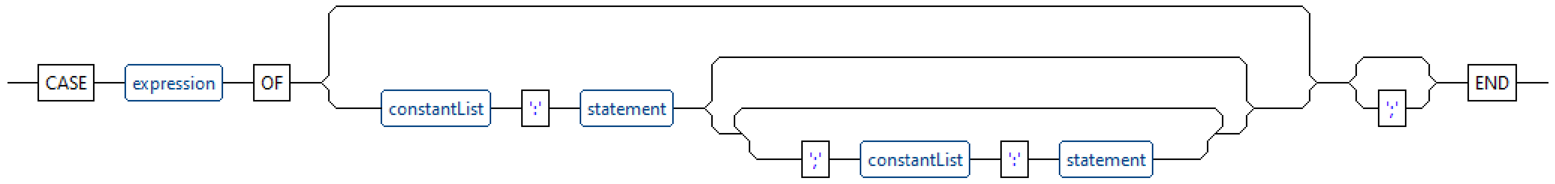


caseStatement Top

Text notation:

```
caseStatement : CASE expression OF (constantList ':' statement (';' constantList ':' statement)*)? ';'? END ;
```

Visual notation:

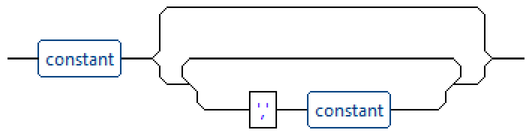


constantList Top

Text notation:

```
constantList : constant (',' constant)* ;
```

Visual notation:

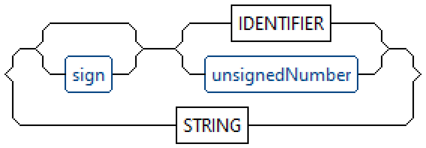


constant Top

Text notation:

```
constant : sign? (IDENTIFIER | unsignedNumber) | STRING ;
```

Visual notation:



lhs Top

Text notation:

```
lhs : variable ;
```

Visual notation:

rhs Top

Text notation:

```
rhs : expression ;
```

Visual notation:

expression

writeStatement Top

Text notation:

```
writeStatement : WRITE writeArgumentsOn ;
```

Visual notation:

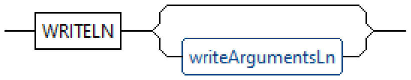
WRITE
writeArgumentsOn

writelnStatement Top

Text notation:

```
writelnStatement : WRITELN writeArgumentsLn? ;
```

Visual notation:

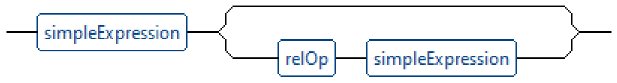


expression Top

Text notation:

```
expression : simpleExpression (relOp simpleExpression)? ;
```

Visual notation:

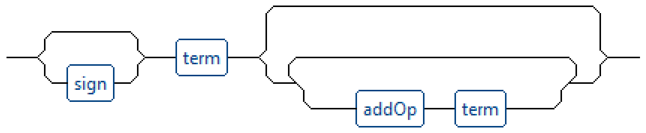


simpleExpression Top

Text notation:

```
simpleExpression : sign? term (addOp term)* ;
```

Visual notation:

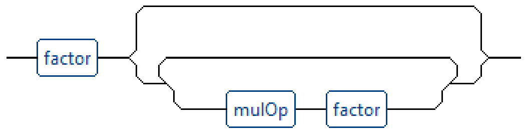


term Top

Text notation:

```
term : factor (mulOp factor)* ;
```

Visual notation:

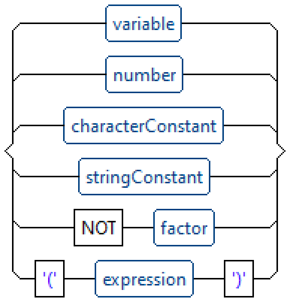


factor Top

Text notation:

```
factor : variable # variableExpression | number # numberExpression | characterConstant # characterFactor | stringConstant # stringFactor |
NOT factor # notFactor | '(' expression ')' # parenthesizedExpression ;
```

Visual notation:

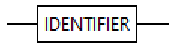


variable Top

Text notation:

```
variable : IDENTIFIER ;
```

Visual notation:

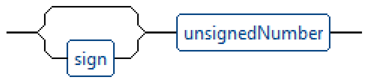


number Top

Text notation:

```
number : sign? unsignedNumber ;
```

Visual notation:

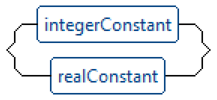


unsignedNumber Top

Text notation:

```
unsignedNumber : integerConstant | realConstant ;
```

Visual notation:



integerConstant Top

Text notation:

```
integerConstant : INTEGER ;
```

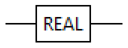
Visual notation:

realConstant Top

Text notation:

```
realConstant : REAL ;
```

Visual notation:

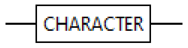


characterConstant Top

Text notation:

```
characterConstant : CHARACTER ;
```

Visual notation:

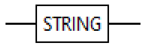


stringConstant Top

Text notation:

```
stringConstant : STRING ;
```

Visual notation:

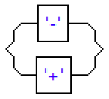


sign Top

Text notation:

```
sign : '-' | '+' ;
```

Visual notation:

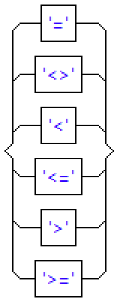


relOp Top

Text notation:

```
relOp : '=' | '<>' | '<' | '<=' | '>' | '>=' ;
```

Visual notation:

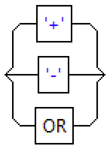


addOp Top

Text notation:

```
addOp : '+' | '-' | OR ;
```

Visual notation:

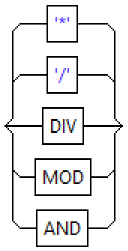


mulOp Top

Text notation:

```
mulOp : '*' | '/' | DIV | MOD | AND ;
```

Visual notation:

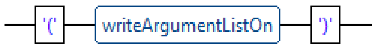


writeArgumentsOn Top

Text notation:

```
writeArgumentsOn : '(' writeArgumentListOn ')' ;
```

Visual notation:

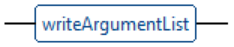


writeArgumentListOn Top

Text notation:

```
writeArgumentListOn : writeArgumentList ;
```

Visual notation:

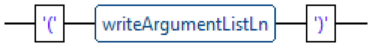


writeArgumentsLn Top

Text notation:

```
writeArgumentsLn : '(' writeArgumentListLn ')' ;
```

Visual notation:

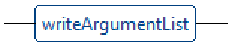


writeArgumentListLn Top

Text notation:

```
writeArgumentListLn : writeArgumentList ;
```

Visual notation:

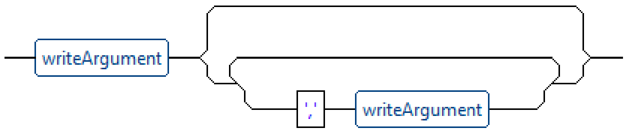


writeArgumentList Top

Text notation:

```
writeArgumentList : writeArgument (',' writeArgument)* ;
```

Visual notation:

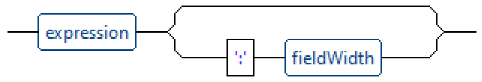


writeArgument Top

Text notation:

```
writeArgument : expression (':' fieldWidth)? ;
```

Visual notation:

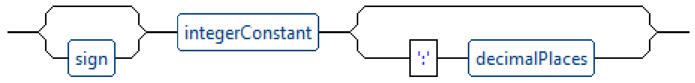


fieldWidth Top

Text notation:

```
fieldWidth : sign? integerConstant (':' decimalPlaces)? ;
```

Visual notation:

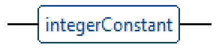


decimalPlaces Top

Text notation:

```
decimalPlaces : integerConstant ;
```

Visual notation:

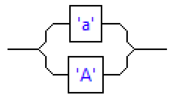


A Top

Text notation:

```
A : ('a' | 'A') ;
```

Visual notation:

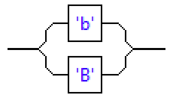


B Top

Text notation:

```
B : ('b' | 'B') ;
```

Visual notation:

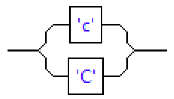


C Top

Text notation:

```
C : ('c' | 'C') ;
```

Visual notation:

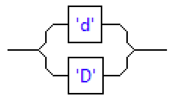


D Top

Text notation:

```
D : ('d' | 'D') ;
```

Visual notation:

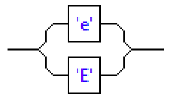


E Top

Text notation:

```
E : ('e' | 'E') ;
```

Visual notation:

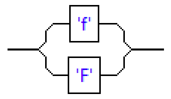


F Top

Text notation:

```
F : ('f' | 'F') ;
```

Visual notation:

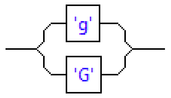


G Top

Text notation:

```
G : ('g' | 'G') ;
```

Visual notation:

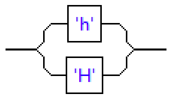


H Top

Text notation:

```
H : ('h' | 'H') ;
```

Visual notation:

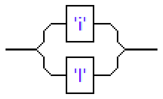


I Top

Text notation:

```
I : ('i' | 'I') ;
```

Visual notation:

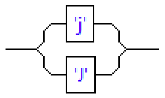


J Top

Text notation:

```
J : ('j' | 'J') ;
```

Visual notation:

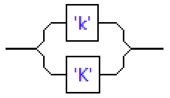


K Top

Text notation:

```
K : ('k' | 'K') ;
```

Visual notation:

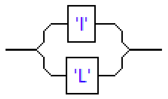


L Top

Text notation:

```
L : ('l' | 'L') ;
```

Visual notation:

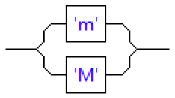


M Top

Text notation:

```
M : ('m' | 'M') ;
```

Visual notation:

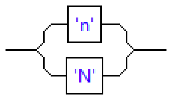


N Top

Text notation:

```
N : ('n' | 'N') ;
```

Visual notation:

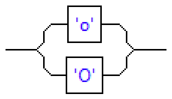


O Top

Text notation:

```
O : ('o' | 'O') ;
```

Visual notation:

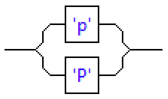


P Top

Text notation:

```
P : ('p' | 'P') ;
```

Visual notation:

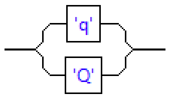


Q Top

Text notation:

```
Q : ('q' | 'Q') ;
```

Visual notation:

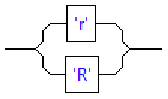


R Top

Text notation:

```
R : ('r' | 'R') ;
```

Visual notation:

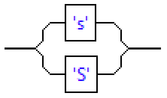


S Top

Text notation:

```
S : ('s' | 'S') ;
```

Visual notation:

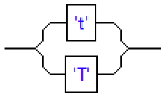


T Top

Text notation:

```
T : ('t' | 'T') ;
```

Visual notation:

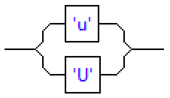


U Top

Text notation:

```
U : ('u' | 'U') ;
```

Visual notation:

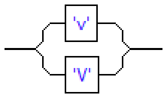


V Top

Text notation:

```
V : ('v' | 'V') ;
```

Visual notation:

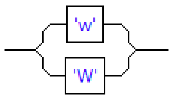


W Top

Text notation:

```
W : ('w' | 'W') ;
```

Visual notation:

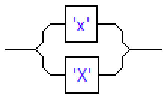


X Top

Text notation:

```
X : ('x' | 'X') ;
```

Visual notation:

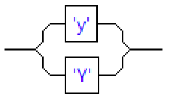


Y Top

Text notation:

```
Y : ('y' | 'Y') ;
```

Visual notation:

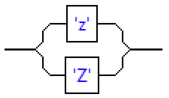


Z Top

Text notation:

```
Z : ('z' | 'Z') ;
```

Visual notation:

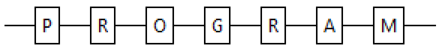


PROGRAM Top

Text notation:

```
PROGRAM : P R O G R A M ;
```

Visual notation:

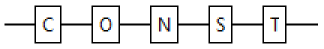


CONST Top

Text notation:

```
CONST : C O N S T ;
```

Visual notation:

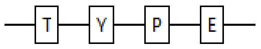


TYPE Top

Text notation:

```
TYPE : T Y P E ;
```

Visual notation:

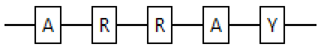


ARRAY Top

Text notation:

```
ARRAY : A R R A Y ;
```

Visual notation:

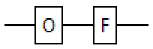


OF Top

Text notation:

```
OF : O F ;
```

Visual notation:

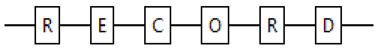


RECORD Top

Text notation:

```
RECORD : R E C O R D ;
```

Visual notation:

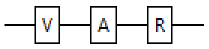


VAR Top

Text notation:

```
VAR : V A R ;
```

Visual notation:

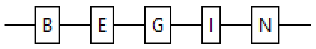


BEGIN Top

Text notation:

```
BEGIN : B E G I N ;
```

Visual notation:

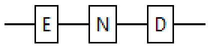


END Top

Text notation:

```
END : E N D ;
```

Visual notation:

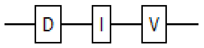


DIV Top

Text notation:

```
DIV : D I V ;
```

Visual notation:

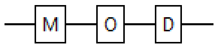


MOD Top

Text notation:

```
MOD : M O D ;
```

Visual notation:

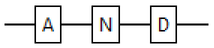


AND Top

Text notation:

```
AND : A N D ;
```

Visual notation:

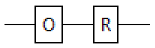


OR Top

Text notation:

```
OR : O R ;
```

Visual notation:

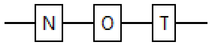


NOT Top

Text notation:

```
NOT : N O T ;
```

Visual notation:

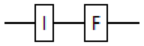


IF Top

Text notation:

```
IF : I F ;
```

Visual notation:

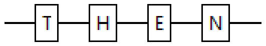


THEN Top

Text notation:

```
THEN : T H E N ;
```

Visual notation:

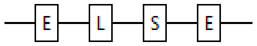


ELSE Top

Text notation:

```
ELSE : E L S E ;
```

Visual notation:

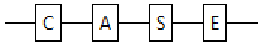


CASE Top

Text notation:

```
CASE : C A S E ;
```

Visual notation:

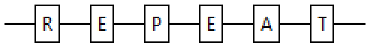


REPEAT Top

Text notation:

```
REPEAT : R E P E A T ;
```

Visual notation:

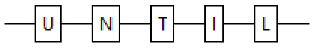


UNTIL Top

Text notation:

```
UNTIL : U N T I L ;
```

Visual notation:

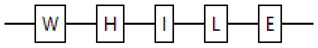


WHILE Top

Text notation:

```
WHILE : W H I L E ;
```

Visual notation:

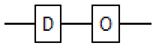


DO Top

Text notation:

```
DO : D O ;
```

Visual notation:

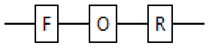


FOR Top

Text notation:

```
FOR : F O R ;
```

Visual notation:

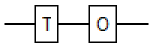


TO Top

Text notation:

```
TO : T O ;
```

Visual notation:

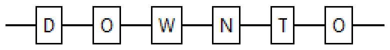


DOWNTO Top

Text notation:

```
DOWNTO : D O W N T O ;
```

Visual notation:

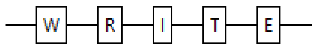


WRITE Top

Text notation:

```
WRITE : W R I T E ;
```

Visual notation:

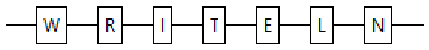


WRITELN Top

Text notation:

```
WRITELN : W R I T E L N ;
```

Visual notation:

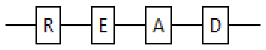


READ Top

Text notation:

```
READ : R E A D ;
```

Visual notation:

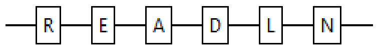


READLN Top

Text notation:

```
READLN : R E A D L N ;
```

Visual notation:

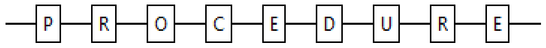


PROCEDURE Top

Text notation:

```
PROCEDURE : P R O C E D U R E ;
```

Visual notation:

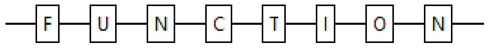


FUNCTION Top

Text notation:

```
FUNCTION : F U N C T I O N ;
```

Visual notation:

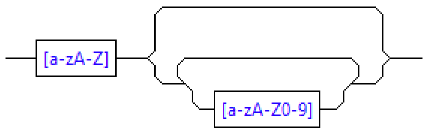


IDENTIFIER Top

Text notation:

```
IDENTIFIER : [a-zA-Z][a-zA-Z0-9]* ;
```

Visual notation:

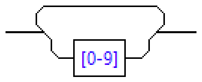


INTEGER Top

Text notation:

```
INTEGER : [0-9]+ ;
```

Visual notation:

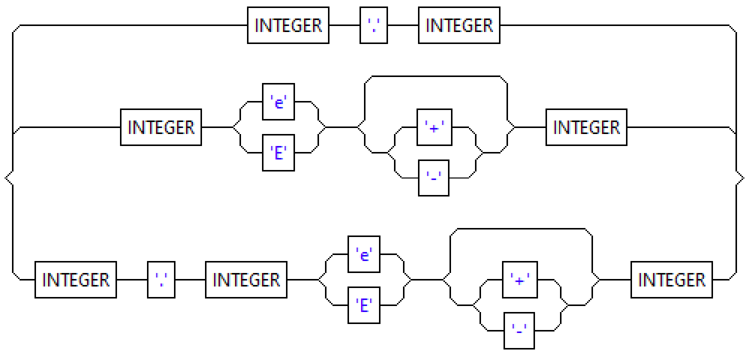


REAL Top

Text notation:

```
REAL : INTEGER '.' INTEGER | INTEGER ('e' | 'E') ('+' | '-')? INTEGER | INTEGER '.' INTEGER ('e' | 'E') ('+' | '-')? INTEGER ;
```

Visual notation:

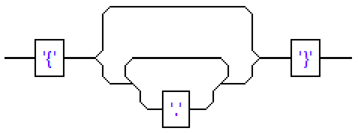


COMMENT Top

Text notation:

```
COMMENT : '{' .*? '}' -> skip ;
```

Visual notation:

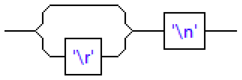


NEWLINE Top

Text notation:

```
NEWLINE : '\r'? '\n' -> skip ;
```

Visual notation:

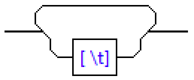


WS Top

Text notation:

```
WS : [ \t]+ -> skip ;
```

Visual notation:

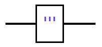


QUOTE Top

Text notation:

```
QUOTE : '\'' ;
```

Visual notation:

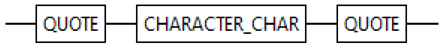


CHARACTER Top

Text notation:

```
CHARACTER : QUOTE CHARACTER_CHAR QUOTE ;
```

Visual notation:

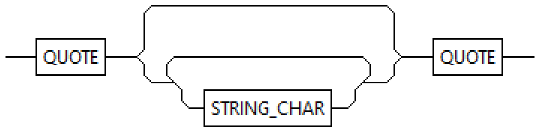


STRING Top

Text notation:

```
STRING : QUOTE STRING_CHAR* QUOTE ;
```

Visual notation:

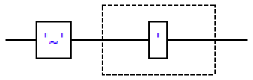


CHARACTER_CHAR Top

Text notation:

```
CHARACTER_CHAR : ~('\'') ;
```

Visual notation:

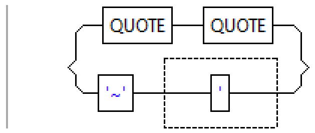


STRING_CHAR Top

Text notation:

```
STRING_CHAR : QUOTE QUOTE | ~('\'') ;
```

Visual notation:


QUOTE
QUOTE
'~'
'