Program:

program = program-heading ';' program-block '.'
program-heading = 'program' identifier ['(' program-parameter-list ')']$_{opt}$
program-parameter-list = identifier-list
program-block = block

Block:

block = constant-definition-part$_{opt}$ type-definition-part$_{opt}$
variable-declaration-part$_{opt}$
procedure-and-function-declaration-part$_{opt}$
statement-part

Constant definition part:

constant-definition-part = 'const' const-definition ';' [const-definition ';']*
const-definition = identifier '=' constant
constant = sign$_{opt}$ (unsigned-number | constant-identifier)
| character-string
constant-identifier = identifier

Type definition part:

type-definition-part = 'type' type-definition ';' [type-definition ';']*
type-definition = identifier '=' type-denoter

About type:

type-denoter = type-identifier | array-type
type-identifier = identifier
array-type = 'array' index-type-list 'of' component-type
index-type-list = index-type [',' index-type]*
index-type = type-identifier
component-type = type-denoter

Variable declaration part:

variable-declaration-part = 'var' variable-declaration ';' [variable-declaration ';']*
variable-declaration = identifier-list ':' type-denoter

Procedure and function declaration part:

procedure-and-function-declaration-part = [(procedure-declaration | function-declaration) ';']$^{+}$
procedure-declaration = procedure-heading ';' procedure-block
procedure-heading = 'procedure' identifier [formal-parameter-list]$_{\text{opt}}$
procedure-block = block
function-declaration = function-heading ';' function-block
function-heading = 'function' identifier [formal-parameter-list]$_{\text{opt}}$ ':' result-type
function-block = block
result-type = type-identifier
formal-parameter-list = '(' formal-parameter-section [';' formal-parameter-section]* ')'
formal-parameter-section = value-parameter-specification | variable-parameter-specification
value-parameter-specification = identifier-list ':' type-identifier
variable-parameter-specification = 'var' identifier-list ':' type-identifier

Statement part:

statement-part = compound-statement
compound-statement = 'begin' statement-sequence 'end'
statement-sequence = statement [';' statement]*

About statement:

statement = simple-statement | structured-statement
simple-statement = empty-statement | assignment-statement | procedure-statement
empty-statement = ϵ
assignment-statement = (variable-access | function-identifier) ':=' expression
procedure-statement = procedure-identifier (actual-parameter-list$_{\text{opt}}$ | read-parameter-list | write-parameter-list)
structured-statement = compound-statement | conditional-statement | repetitive-statement
conditional-statement = if-statement
repetitive-statement = while-statement | for-statement
if-statement = 'if' Boolean-expression 'then' statement else-part$_{\text{opt}}$
else-part = 'else' statement
while-statement = 'while' Boolean-expression 'do' statement
for-statement = 'for' control-variable ':=' initial-value ('to' | 'downto') final-value 'do' statement

variable-access = entire-variable | component-variable
entire-variable = variable-identifier
variable-identifier = identifier
component-variable = indexed-variable
indexed-variable = array-variable
'[' index-expression [',' index-expression]* ']'
array-variable = variable-access
index-expression = expression
function-identifier = identifier
procedure-identifier = identifier
actual-parameter-list = '(' actual-parameter [',' actual-parameter]* ')'
actual-parameter = expression | variable-access
read-parameter-list = '(' variable-access [',' variable-access]* ')'
write-parameter-list = '(' write-parameter [',' write-parameter]* ')'
write-parameter = expression
control-variable = entire-variable
initial-value = expression
final-value = expression
Boolean-expression = expression

Expression

expression = simple-expression [relational-operator simple-expression]$_{\text{opt}}$
simple-expression = sign$_{\text{opt}}$ term [adding-operator term]*
term = factor [multiplying-operator factor]*
factor = variable-access | unsigned-constant | function-designator
| '(' expression ')' | 'not' factor
function-designator = function-identifier actual-parameter-list$_{\text{opt}}$

Other

identifier-list = identifier [',' identifier]*
relational-operator = '<' | '=' | '>' | '<=' | '>=' | '<>'
adding-operator = '+' | '−' | 'or'
multiplying-operator = '∗' | '/' | 'div' | 'mod' | 'and'