```
ident  =  letter {letter | digit}.
number  =  integer | real.
integer  =  digit {digit} | digit {hexDigit} "H".
real  =  digit {digit} "." {digit} [ScaleFactor].
ScaleFactor  =  ("E" | "D") ["+" | "-"] digit {digit}.
hexDigit  =  digit | "A" | "B" | "C" | "D" | "E" | "F".
digit  =  "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".
CharConstant = "'" character "'" | digit {hexDigit} "X".
string  =  "'" {character} "'" .

identdef = ident ["*"].
qualident = [ident "."] ident.
ConstantDeclaration  =  identdef "=" ConstExpression.
ConstExpression  =  expression.
TypeDeclaration  =  identdef "=" type.
type  =  qualident | ArrayType | RecordType | PointerType | ProcedureType.
ArrayType  =  ARRAY length {"," length} OF type.
length  =  ConstExpression.
RecordType  =  RECORD ["(" BaseType ")"] FieldListSequence END.
BaseType  =  qualident.
FieldListSequence  =  FieldList {";" FieldList}.
FieldList  =  [IdentList ":" type].
IdentList  =  identdef {"," identdef}.
PointerType  =  POINTER TO type.
ProcedureType = PROCEDURE [FormalParameters].
VariableDeclaration  =  IdentList ":" type.

designator  =  qualident {"." ident | "[" ExpList "]" | "(" qualident ")" | "^" }.
ExpList  =  expression {"," expression}.
expression  =  SimpleExpression [relation SimpleExpression].
relation  =  "=" | "#" | "<" | "<=" | ">" | ">=" | IN | IS.
SimpleExpression  =  ["+"|"-"] term {AddOperator term}.
AddOperator  =  "+" | "-" | OR .
term  =  factor {MulOperator factor}.
MulOperator  =  "*" | "/" | DIV | MOD | "&" .
factor  =  number | CharConstant | string | NIL | set |
  designator [ActualParameters] | "(" expression ")" | "~" factor.
set  =  "{" [element {"," element}] "}".
element  =  expression [".." expression].
ActualParameters  =  "(" [ExpList] ")" .
statement  =  [assignment | ProcedureCall |
  IfStatement | CaseStatement | WhileStatement | RepeatStatement |
  LoopStatement | WithStatement | EXIT | RETURN [expression] ].
assignment  =  designator ":=" expression.
ProcedureCall  =  designator [ActualParameters].
StatementSequence  =  statement {";" statement}.
IfStatement  =  IF expression THEN StatementSequence
  {ELSIF expression THEN StatementSequence}
  [ELSE StatementSequence] END.
CaseStatement  =  CASE expression OF case {"|" case}
  [ELSE StatementSequence] END.
case  =  [CaseLabelList ":" StatementSequence].
CaseLabelList  =  CaseLabels {"," CaseLabels}.
CaseLabels  =  ConstExpression [".." ConstExpression].
WhileStatement  =  WHILE expression DO StatementSequence END.
RepeatStatement  =  REPEAT StatementSequence UNTIL expression.
LoopStatement  =  LOOP StatementSequence END.
WithStatement  =  WITH qualident ":" qualident DO StatementSequence END .

ProcedureDeclaration  =  ProcedureHeading ";" ProcedureBody ident.
ProcedureHeading  =  PROCEDURE ["*"] identdef [FormalParameters].
ProcedureBody  =  DeclarationSequence [BEGIN StatementSequence] END.
ForwardDeclaration  =  PROCEDURE "^" ident ["*"] [FormalParameters].
DeclarationSequence  =  {CONST {ConstantDeclaration ";"} |
   TYPE {TypeDeclaration ";"} | VAR {VariableDeclaration ";"}}
   {ProcedureDeclaration ";" | ForwardDeclaration ";"}.
FormalParameters  =  "(" [FPSection {";" FPSection}] ")" [":" qualident].
FPSection  =  [VAR] ident {"," ident} ":" FormalType.
FormalType  =  {ARRAY OF} (qualident | ProcedureType).
ImportList  =  IMPORT import {"," import} ";" .
import  =  ident [":=" ident].
module  =  MODULE ident ";"  [ImportList] DeclarationSequence
   [BEGIN StatementSequence] END ident "." .
```