program = program-heading ';' program-block '.'

program-heading = 'program' identifier ['(' program-parameter-list ')']$_{\text{opt}}$

program-parameter-list = identifier-list

identifier-list = identifier [',' identifier]*

program-block = block

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

constant-definition-part = 'const' const-definition ';' [const-definition ';']*

const-definition = identifier '=' constant

constant = sign$_{\text{opt}}$ (unsigned-number | constant-identifier)
| character-string

constant-identifier = identifier

type-definition-part = ...

variable-declaration-part = 'var' variable-declaration ';' [variable-declaration ';']*

variable-declaration = identifier-list ':' type-denoter

type-denoter = ...

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

statement-part = compound-statement

compound-statement = 'begin' statement-sequence 'end'

statement-sequence = statement [';' 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 = ...

structured-statement = compound-statement | conditional-statement
| repetitive-statement

conditional-statement = ...

repetitive-statement = ...