

type → int
type → char
type → real
type → bool
type → array of *type*₁
type → string
program → *declarations*
declarations → *declaration* [*declarations*₁]
declaration → *subprogram_decl* | var *var_declaration_list*
var_declaration_list → *var_declaration* ; [*var_declaration_list*₁]
var_declaration → *identifier_list* : *type*
identifier_list → *identifier* [, *identifier_list*₁]
subprogram_decl → *subprogram_header* ; | *subprogram_header* is *statement*
subprogram_header → *procedure_header* | *function_header*
procedure_header → *procedure name_and_formals*
function_header → *function name_and_formals* : *type*
name_and_formals → *identifier* ([*formal_parameters*])
formal_parameters → *formal_parameter* [; *formal_parameters*₁]
formal_parameter → [var] *var_declaration*
expression → (*expression*₁)
expression → integer_constant
expression → character_constant
expression → real_constant
expression → true | false
expression → string_constant
expression → true | false
expression → null
expression → identifier
expression → new array [*expression*₁] of *type*
expression → *expression*₁ [*expression*₂]
expression → *expression*₁ ([*argument_list*])
argument_list → *expression* [, *argument_list*₁]
expression → not *expression*₁
expression → size of *expression*₁
expression → - *expression*₁
expression → + *expression*₁
expression → *expression*₁ * *expression*₂
expression → *expression*₁ / *expression*₂
expression → *expression*₁ - *expression*₂
expression → *expression*₁ + *expression*₂
expression → *expression*₁ < *expression*₂
expression → *expression*₁ <= *expression*₂
expression → *expression*₁ > *expression*₂
expression → *expression*₁ >= *expression*₂
expression → *expression*₁ = *expression*₂
expression → *expression*₁ <> *expression*₂
expression → *expression*₁ and *expression*₂
expression → *expression*₁ or *expression*₂
statement → *expression*₁ := *expression*₂ ;
statement → *expression* ;
statement → delete *expression* ;
statement → if *expression* then *statement*₁ [else *statement*₂]
statement → for *expression*₁ := *expression*₂ to *expression*₃ [by *expression*₄]do *statement*₁
statement → repeat *statement*₁ until *expression* ;
statement → while *expression* do *statement*₁
statement → return [*expression*];
statement → [var *var_declaration_list*]begin [*statement_list*]end
statement_list → *statement* [*statement_list*₁]