PROG -> **FUNC PROG** | ε

FUNC -> function identifier ; beginparams **DECLARATION_CYCLE** endparams beginlocals **DECLARATION_CYCLE** endlocals beginbody **STATEMENT_CYCLE** endbody

DECLARATION_CYCLE -> **DECLARATION** ; **DECLARATION_CYCLE** | ε

DECLARATION -> **IDENTIFIER_CYCLE** : enum (**IDENTIFIER_CYCLE**)
| IDENTIFIER_CYCLE : integer
| IDENTIFIER_CYCLE : array [number] of integer

IDENTIFIER_CYCLE -> identifier **|** identifier, **IDENTIFIER_CYCLE**

STATEMENT_CYCLE -> **STATEMENT** ; **| STATEMENT** ; **STATEMENT_CYCLE**

STATEMENT -> VAR := **EXPRESSION**
| if **BOOL-EXPR** then **STATEMENT_CYCLE ELSE** endif
| while **BOOL-EXPR** beginloop **STATEMENT_CYCLE** endloop
| do beginloop **STATEMENT_CYCLE** endloop while **BOOL-EXPR**
| read **VAR_CYCLE**
| write **VAR_CYCLE**
| continue
| return **EXPRESSION**

ELSE -> else **STATEMENT_CYCLE |** ε

VAR_CYCLE -> VAR , **VAR_CYCLE | VAR**

BOOL-EXPR -> RELATION-AND-EXPR
| RELATION-AND-EXPR or **BOOL-EXPR**

RELATION-AND-EXPR -> RELATION-EXPR
| RELATION-EXPR and **RELATION-AND-EXPR**

RELATION-EXPR -> not **RELATION-EXPR-CASES | RELATION-EXPR-CASES**

```
RELATION-EXPR-CASES -> EXPRESSION COMP EXPRESSION
                                          | true
                                          | false
                                          | ( BOOL-EXPR )

COMP -> = | <> | < | > | <= | >=

EXPRESSION -> MULTIPLICATIVE-EXPR
                         | MULTIPLICATIVE-EXPR + EXPRESSION
                         | MULTIPLICATIVE-EXPR -  EXPRESSION

MULTIPLICATIVE-EXPR -> TERM
                                          | TERM * MULTIPLICATIVE-EXPR
                                          | TERM / MULTIPLICATIVE-EXPR
                                          | TERM % MULTIPLICATIVE-EXPR

TERM -> - VAR
             | - number
             | - ( EXPRESSION )
             | VAR
             | number
             | EXPRESSION
             | identifier ( EXPRESSION_CYCLE )

EXPRESSION_CYCLE -> EXPRESSION , EXPRESSION_CYCLE
                                    | EXPRESSION

VAR -> identifier | identifier [ EXPRESSION ]
```