```
Winzig     -> 'program' Name ':' Consts Types Dclns
                 SubProgs Body Name '.'                  => "program";

Consts     -> 'const' Const list ',' ';'                 => "consts"
           ->                                            => "consts";

Const      -> Name '=' ConstValue                        => "const";

ConstValue -> '<integer>'
           -> '<char>'
           -> Name;

Types      -> 'type' (Type ';')+                         => "types"
           ->                                            => "types";

Type       -> Name '=' LitList                           => "type";

LitList    -> '(' Name list ',' ')'                      => "lit";

SubProgs   -> Fcn*                                       => "subprogs";

Fcn        -> 'function' Name '(' Params ')' ':' Name
              ';' Consts Types Dclns Body Name ';'       => "fcn";

Params     -> Dcln list ';'                              => "params";

Dclns      -> 'var' (Dcln ';')+                          => "dclns"
           ->                                            => "dclns";

Dcln       -> Name list ',' ':' Name                     => "var";

Body       -> 'begin' Statement list ';' 'end'           => "block";

Statement  -> Assignment
           -> 'output' '(' OutExp list ',' ')'           => "output"
           -> 'if' Expression 'then' Statement
                          ('else' Statement)?            => "if"
           -> 'while' Expression 'do' Statement          => "while"
           -> 'repeat' Statement list ';' 'until'
                Expression                               => "repeat"
           -> 'for' '(' ForStat ';' ForExp ';'
                ForStat ')' Statement                    => "for"
           -> 'loop' Statement list ';' 'pool'           => "loop"
           -> 'case' Expression 'of' Caseclauses
                   OtherwiseClause 'end'                 => "case"
           -> 'read' '(' Name list ',' ')'               => "read"
           -> 'exit'                                     => "exit"
           -> 'return' Expression                        => "return"
           -> Body
           ->                                            => "<null>";

OutExp     -> Expression                                 => "integer"
           -> StringNode                                 => "string";
```

```
StringNode -> '<string>';

Caseclauses-> (Caseclause ';')+;

Caseclause -> CaseExpression list ',' ':' Statement => "case_clause";

CaseExpression
           -> ConstValue
           -> ConstValue '..' ConstValue                => "..";

OtherwiseClause
           -> 'otherwise' Statement                     => "otherwise"
           -> ;

Assignment -> Name ':=' Expression                      => "assign"
           -> Name ':=:' Name                           => "swap";

ForStat    -> Assignment
           ->                                           => "<null>";

ForExp     -> Expression
           ->                                           => "true";

Expression -> Term
           -> Term '<=' Term                            => "<="
           -> Term '<' Term                             => "<"
           -> Term '>=' Term                            => ">="
           -> Term '>' Term                             => ">"
           -> Term '=' Term                             => "="
           -> Term '<>' Term                            => "<>";

Term       -> Factor
           -> Term '+' Factor                           => "+"
           -> Term '-' Factor                           => "-"
           -> Term 'or' Factor                          => "or";

Factor     -> Factor '*' Primary                        => "*"
           -> Factor '/' Primary                        => "/"
           -> Factor 'and' Primary                      => "and"
           -> Factor 'mod' Primary                      => "mod"
           -> Primary;

Primary    -> '-' Primary                               => "-"
           -> '+' Primary
           -> 'not' Primary                             => "not"
           -> 'eof'                                     => "eof"
           -> Name
           -> '<integer>'
           -> '<char>'
           -> Name '(' Expression list ',' ')'          => "call"
           -> '(' Expression ')'
           -> 'succ' '(' Expression ')'                 => "succ"
           -> 'pred' '(' Expression ')'                 => "pred"
           -> 'chr' '(' Expression ')'                  => "chr"
```

```
            -> 'ord' '(' Expression ')'                    => "ord";

Name        -> '<identifier>';
```