# Gramática Oberon-0

## Backus-Naur Form (BNF)

```
module ::= “module” ID “;” declaratios
                   “begin” statements “end” ID “.”

vardecl ::= “var” vardecl_list | /**/
vardecl_list ::= vardecl_list idlist “:” vartype “;” | idlist “:” vartype “;”

idlist ::= idlist ID | ID

vartype ::= “boolean” | “integer”

procdecl_list ::= procdecl_list  procdecl | /**/
procdecl ::= procheader procbody
procheader ::= “procedure” ID formalpars “;”
                  | “function” ID formalpars “:” vartype “;”
procbody ::= vardecl “begin” statements “end” ID “;”

formalpars ::= “(“ fpsection_opt_list “)”
fpsection_opt_list ::= fpsection_list | /**/
fpsection_list ::= fpsection_list “;” fpsection | fpsection
fpsection ::=  idlist “:” vartype

statements ::= statements statement “;” | /**/
declarations ::= vardecl procdecl_list | procdecl_list
expression ::= expression “or” andexp | andexp
andexp ::= andexp “and” relexp | relexp
relexp ::= aritexp REL_OP aritexp | aritexp
aritexp ::= aritexp ADD_OP term | term
term ::= term MULT_OP factor | factor
factor ::= UNARY_OP primary
primary ::= “(“ expression “)”
              | proccall
              | literal
              | ID

literal ::= BOOLEAN_LITERAL | INTEGER_LITERAL
proccall ::= ID actualpar
actualpar ::= “(“ expression_list “)” | “(“ “)”
expression_list ::= expression_list  “,” expression |  expression

REL_OP ::= ">" | "<" | ">=" | "<=" | "=" | "#"
ADD_OP ::= "+" | "-"
MULT_OP ::= "*" | "mod" | "/"
UNARY_OP ::= “+” | “-” | “not” | /**/

statement ::= assignment
                 | conditional
                 | repetition
                 | proccall
                 | io_statement
                 | "continue"
                 | "break"
                 | "return" expression

assignment ::= variable ":=" expression

conditional ::= "if" expression "then" statements
                           elseif_opt_list
                           else_opt
                     "end"

elseif_opt_list ::= elseif_opt_list "elseif" expression "then" statements | /**/

else_opt ::= "else" statements | /**/
repetition ::= "while" expression "do" statement "end"
                 | "repeat" statements "until" expression
                 | "for" ID "=" expression "to" expression
                            "do" statement "end"

io_statement ::= "write" "(" expression ")"
                 | "writeln"
                 | "writeln" "(" expression ")"
                 | "read" "(" expression ")"
```

## Convensões Léxicas

```
digit ::= [0-9]
letter ::= [A-Za-z]
printable ::= [^\"\n\r\t\0\xDD]
BOOLEAN_LITERALS ::= "TRUE" | "FALSE"
INTEGER_LITERALS ::= digit { digit }
STRING_LITERALS ::= "\"" { printable } "\""
ID ::= (letter | "_") { letter | digit | "_" }
```

## Palavras Chaves

module
begin
end
procedure
function
var
boolean
integer
and
or
not
mod
continue
break

return
if
then
elseif
else
while
do
repeat
until
for
to

## Símbolos

:=
<=
>=
:
;
.
,
+
-
*
/
=
#
<
>
(
)
{
}

## Palavras Reservadas

write
writeln
read