

EBNF Grammar for Objective Modula-2 (Lexical Analysis)

Reserved Words

- | | |
|--|-------------------------------------|
| (1) AND = "AND" | (30) MODULE = "MODULE" |
| (2) ARRAY = "ARRAY" | (31) NOT = "NOT" |
| (3) BEGIN = "BEGIN" | (32) OF = "OF" |
| (4) BY = "BY" | (33) ON = "ON" |
| (5) BYCOPY = "BYCOPY" | (34) OPTIONAL = "OPTIONAL" |
| (6) BYREF = "BYREF" | (35) OR = "OR" |
| (7) CASE = "CASE" | (36) OUT = "OUT" |
| (8) CLASS = "CLASS" | (37) POINTER = "POINTER" |
| (9) CONST = "CONST" | (38) PRIVATE = "PRIVATE" |
| (10) CONTINUE = "CONTINUE" | (39) PROCEDURE = "PROCEDURE" |
| (11) CRITICAL = "CRITICAL" | (40) PROTECTED = "PROTECTED" |
| (12) DEFINITION = "DEFINITION" | (41) PROTOCOL = "PROTOCOL" |
| (13) DIV = "DIV" | (42) PUBLIC = "PUBLIC" |
| (14) DO = "DO" | (43) RECORD = "RECORD" |
| (15) ELSE = "ELSE" | (44) REFINES = "REFINES" |
| (16) ELSIF = "ELSIF" | (45) REPEAT = "REPEAT" |
| (17) END = "END" | (46) REQUIRED = "REQUIRED" |
| (18) EXIT = "EXIT" | (47) RETURN = "RETURN" |
| (19) FOR = "FOR" | (48) SELF = "SELF" |
| (20) FROM = "FROM" | (49) SET = "SET" |
| (21) IF = "IF" | (50) SUPER = "SUPER" |
| (22) IMPLEMENTATION =
"IMPLEMENTATION" | (51) THEN = "THEN" |
| (23) IMPORT = "IMPORT" | (52) TO = "TO" |
| (24) IN = "IN" | (53) TRY = "TRY" |
| (25) INOUT = "INOUT" | (54) TYPE = "TYPE" |
| (26) INSTANCE = "INSTANCE" | (55) UNTIL = "UNTIL" |
| (27) LOOP = "LOOP" | (56) VAR = "VAR" |
| (28) METHOD = "METHOD" | (57) WHILE = "WHILE" |
| (29) MOD = "MOD" | |

EBNF Grammar for Objective Modula-2 (Lexical Analysis)

Identifiers

- (58) `ident =`
 `("_" | "$" | letter) { "_" | "$" | letter | digit }`
- (59) `labeled-ident1 =`
 `ident ":"`
- (60) `letter =`
 `"A" .. "Z" | "a" .. "z"`

Numeric Literals

- (61) `octal-whole-number-literal =`
 `octal-digit { octal-digit } "B"`
- (62) `decimal-whole-number-literal =`
 `digit { digit }`
- (63) `sedecimal-whole-number-literal =`
 `digit { sedecimal-digit } "H"`
- (64) `7-bit-ascii-character-code-literal =`
 `{ "0" } ["1"] [octal-digit] octal-digit "C"`
- (65) `unicode-character-code-literal =`
 `{ "0" } (digit | ("0" non-decimal-digit))`
 `[sedecimal-digit] [sedecimal-digit] [sedecimal-digit] "U"`
- (66) `real-number-literal =`
 `(digit { digit } "." digit { digit }) |`
 `(digit "." digit { digit } "E" ["+" | "-"] digit { digit })`
- (67) `octal-digit =`
 `"0" | "1" | "2" | "3" | "4" | "5" | "6" | "7"`
- (68) `digit =`
 `octal-digit | "8" | "9"`
- (69) `non-decimal-digit =`
 `"A" | "B" | "C" | "D" | "E" | "F"`
- (70) `sedecimal-digit =`
 `digit | non-decimal-digit`

String Literals

- (71) `quoted-character-literal =`
 `('"' character '"') | ("'" character "'")`
- (72) `string-literal =`
 `('"' { character } '"') | ("'" { character } "'")`

¹ Labeled identifiers can only occur within method declarations and messages. Symbol resolution must take place during lexical analysis in order to be able to define an LL(1) grammar for syntax analysis.

EBNF Grammar for Objective Modula-2 (Lexical Analysis)

```
(73) character =  
    printable-character | escaped-character  
  
(74) printable-character =  
    " " | "!" | "'" | "#" | "$" | "%" | "&" | "'" | "(" | ")" | "*" |  
    "+" | "," | "-" | "." | "/" | ":" | ";" | "<" | "=" | ">" | "?" |  
    "@" | "[" | "]" | "^" | "_" | "`" | "{" | "|" | "}" | "~" |  
    letter | digit  
  
(75) escaped-character =  
    "\" ( "'" | '"' | "\" | "n" | "r" | "t" | "0" )
```

Operator Symbols

```
(76) assign-operator = " :="   
  
(77) logical-and-operator = "&"   
  
(78) logical-not-operator = "~"   
  
(79) equal-operator = "="   
  
(80) not-equal-operator = "#" | "<>" | "!="   
  
(81) greater-than-operator = ">"   
  
(82) greater-or-equal-operator = ">="   
  
(83) less-than-operator = "<"   
  
(84) less-or-equal-operator = "<="   
  
(85) plus-operator = "+"   
  
(86) postfix-increment-operator = "++"   
  
(87) minus-operator = "-"   
  
(88) postfix-decrement-operator = "--"   
  
(89) multiply-operator = "*"   
  
(90) divide-operator = "/"   
  
(91) pointer-dereference-operator = "^"
```

Punctuation

```
(92) message-prefix = "`"   
  
(93) opening-parenthesis = "("   
  
(94) closing-parenthesis = ")"   
  
(95) opening-bracket = "["
```

EBNF Grammar for Objective Modula-2 (Lexical Analysis)

```
(96) closing-bracket = "]"
(97) opening-brace = "{"
(98) closing-brace = "}"
(99) dot = "."
(100) double-dot = ".."
(101) comma = ","
(102) colon = ":"
(103) semicolon = ";"
(104) vertical-bar = "|"
```

Pragmas and Comments

```
(105) pragma =
    "<*" { printable-character } ">"
(106) c-comment =
    "/*" { printable-character | end-of-line-marker } "*/"
(107) m2-comment =
    "(*" { printable-character | end-of-line-marker } | { m2-comment } ")"
(108) cpp-comment =
    "/*" { printable-character } end-of-line-marker
```

Separators

```
(109) whitespace2 =
    ASCII-SP
(110) tabulator3 =
    ASCII-TAB
(111) end-of-line-marker4 =
    [ ASCII-CR ] ASCII-LF
(112) end-of-file-marker =
    operating system specific
```

² Whitespace separates symbols, with the exception of string literals, pragmas and comments

³ Tabulator is treated as whitespace

⁴ The end-of-line marker separates symbols with the exception of symbols #106 and #107