EBNF Grammar for Objective Modula-2

Typography

• Production rules are shown in fixed width
• Alias rules are shown in italic fixed width
• Reserved words are shown in UPPERCASE BOLDFACE
• Other terminal symbols are shown in lowercase boldface
• Reserved symbols are shown in ‘single quotes’ or “double quotes”

EBNF Notation

• Alternatives are separated by a vertical bar |
• Groups of entities are enclosed in parentheses ()
• One or more occurrences are indicated by a trailing raised plus sign +
• Zero or more occurrences are indicated by a trailing raised asterisk *
• Zero or one occurrences are indicated by a trailing raised question mark ?

Compilation Units

(1) compilationUnit
    programModule | definitionOfModule | implementationOfModule | protocolModule

(2) programModule
    MODULE moduleId ( "[" priority "]" )? "";
    importList* block moduleId "."

(3) definitionOfModule
    DEFINITION MODULE moduleId "";
    importList* definition*
    END moduleId "."

(4) implementationOfModule
    IMPLEMENTATION programModule

(5) protocolModule
    PROTOCOL protocolId ( "(" adoptedProtocols ")" )? "";
    importList* ( OPTIONAL? methodHeader )* 
    END protocolId "."

(6) moduleId = ident

(7) priority = constExpression

(8) protocolId = ident

(9) adoptedProtocols = identList
EBNF Grammar for Objective Modula-2

Import Lists, Blocks, Declarations, Definitions

(10) importList
    ( FROM moduleId IMPORT ( identList | "*" ) | IMPORT identList ) ";"

(11) block
    declaration*
    ( BEGIN statementSequence )? END

(12) declaration
    CONST ( constantDeclaration ";" )* | TYPE ( typeDeclaration ";" )* | VAR ( variableDeclaration ";" )* | procedureDeclaration ";" | methodDeclaration ";"

(13) definition
    CONST ( constantDeclaration ";" )* | TYPE ( ident ( "=" ( type | OPAQUE ) | IS namedType ) ";" )" | VAR ( variableDeclaration ";" )* | procedureHeader ";" | methodHeader ";"

Constant Declarations

(14) constantDeclaration
    ident "=" ( constExpression | structuredValue )

Type Declarations

(15) typeDeclaration
    ident ( "=" type | IS namedType )

(16) type
    namedType | anonymousType | enumerationType | setType | classType

(17) namedType = qualident

(18) anonymousType
    arrayType | recordType | pointerType | procedureType

(19) enumerationType
    ENUM ( "(" baseType ")" )? identList? END | "(" identList ")"

(20) baseType = qualident
EBNF Grammar for Objective Modula-2

(21) arrayType
   ARRAY arrayIndex ( "," arrayIndex )* OF ( namedType | recordType | procedureType )

(22) arrayIndex = ordinalConstExpression

(23) ordinalConstExpression = constExpression

(24) recordType
   RECORD ( "(" baseType ")" )? fieldListSequence? END

(25) fieldListSequence
   fieldList ( ";" fieldList )*

(26) fieldList
   identList ":"
   ( namedType | arrayType | recordType | procedureType )

(27) classType
   "<QUALIFIED*>" CLASS "(" superClass ( "," adoptedProtocols )? ")"
   ( ( PUBLIC | MODULE | PROTECTED | PRIVATE )? fieldListSequence )*
   END

(28) superClass = qualident

(29) setType
   SET OF ( namedType | "(" identList ")" )

(30) pointerType
   POINTER TO IMMUTABLE? namedType

(31) procedureType
   PROCEDURE
   ( "(" formalTypeList ")" )?
   ( ":" returnedType )?

(32) formalTypeList
   attributedFormalType ( "," attributedFormalType )*

(33) attributedFormalType
   IMMUTABLE? VAR? formalType

(34) formalType
   ( ARRAY OF )? namedType

(35) returnedType = namedType
EBNF Grammar for Objective Modula-2

Variable Declarations

(36) variableDeclaration
    ident ( "[" machineAddress "]" | "," identList )? "::" ( namedType | anonymousType )

(37) machineAddress = constExpression

Procedure Declarations

(38) procedureDeclaration
    procedureHeader "::" block ident

(39) procedureHeader
    PROCEDURE ( "(" ident "::" receiverType ")" )? ident ( "(" formalParamList ")" )? ( "::" returnedType )?

(40) receiverType = ident

(41) formalParamList
    formalParams ( ";" ( formalParams | variadicParams ) )*

(42) formalParams
    IMMUTABLE? VAR? identList ";" formalType

(43) variadicParams
    VARIADIC handle ( "[" indexParam "]" )? OF
    IMMUTABLE? VAR? ( ident ( ( "." ident )* | ( "," ident )* ";" returnedType )

(44) handle = ident

(45) indexParam = ident

Method Declarations

(46) methodDeclaration
    methodHeader "::" block ident

(47) methodHeader
    CLASS? METHOD ("(" ident "::" ( receiverClass | "*" ) ")")
    ( ident | methodArg ) methodArg*
    ( ";" returnedType )?

(48) receiverClass = qualident
EBNF Grammar for Objective Modula-2

(49) methodArg
    colonIdent "( IMMUTABLE? VAR? ident ":" formalType ")"

Statements

(50) statement
    ( assignmentOrProcedureCall | methodInvocation |
      ifStatement | caseStatement | whileStatement | repeatStatement |
      loopStatement | forStatement | tryStatement | criticalStatement |
      RETURN expression? | EXIT )?

(51) statementSequence
    statement ( ";" statement )*

(52) methodInvocation
    "[" receiver message "]"

(53) receiver
    ident | methodInvocation

(54) message
    ident ( colonIdent expression )* |
    ( colonIdent expression )+

(55) assignmentOrProcedureCall
    designator
    ( ":=" ( expression | structuredValue ) | 
      ++ " " -- " |
      actualParameters )?

(56) ifStatement
    IF expression THEN statementSequence
    ( ELSEIF expression THEN statementSequence )* 
    ( ELSE statementSequence )* 
    END

(57) caseStatement
    CASE expression OF case ( "|" case )* 
    ( ELSE statementSequence )* 
    END

(58) case
    caseLabelList ":" statementSequence

(59) caseLabelList
    caseLabels ( "," caseLabels )*
EBNF Grammar for Objective Modula-2

(60) caseLabels
    constExpression ( ".." constExpression )?

(61) whileStatement
    WHILE expression DO statementSequence END

(62) repeatStatement
    REPEAT statementSequence UNTIL expression

(63) loopStatement
    LOOP statementSequence END

(64) forStatement
    FOR ident ":=" expression TO expression ( BY constExpression )?
    DO statementSequence END

(65) tryStatement
    TRY statementSequence
    ON ident DO statementSequence
    CONTINUE statementSequence
    END

(66) criticalStatement
    CRITICAL "(" classInstance ")" statementSequence
    END

(67) classInstance = qualident

Expressions

(68) constExpression
    simpleConstExpr ( relation simpleConstExpr | "::" namedType )?

(69) relation
    "=" | "#" | "<" | "<=" | "=" | ">" | ">=" | IN | IS

(70) simpleConstExpr
    ( "+" | "-" )? constTerm ( addOperator constTerm )*

(71) addOperator
    "+" | "-" | OR

(72) constTerm
    constFactor ( mulOperator constFactor )*

(73) mulOperator
    "*" | "/" | DIV | MOD | AND | "&"
EBNF Grammar for Objective Modula-2

(74) constFactor
    number | string | qualident | "(" constExpression ")" | ( NOT | "~" ) constFactor

(75) designator
    qualident ( designatorTail )?

(76) designatorTail
    ( ( "[" expressionList "]" | "^" ) ( "." ident )* )+

(77) expressionList
    expression ( "," expression )*

(78) expression
    simpleExpression ( relation simpleExpression | "::" namedType )?

(79) simpleExpression
    ( "+" | "-" )? term ( addOperator term )*

(80) term
    factor ( mulOperator factor )*

(81) factor
    number | string | designatorOrProcedureCall | methodInvocation
    "(" constExpression ")" | ( NOT | "~" ) factor |

(82) designatorOrProcedureCall
    qualident designatorTail? actualParameters?

(83) actualParameters
    "(" expressionList? ")"

Value Constructors

(84) structuredValue
    "{" ( valueComponent ( "," valueComponent )* )? "}"  

(85) valueComponent
    constExpression ( ( BY | ".." ) constExpression )? | structuredValue

Identifiers

(86) qualident
    ident ( "." ident )*

Copyright © 2009 The Objective Modula-2 Project
EBNF Grammar for Objective Modula-2

(87) identList
    ident ( ",” ident )* 

Pragmas

(88) pragma
    “<*” ( IF | ELSIF ) constExpression | ELSE | ENDIF |
    (INFO | WARN | ERROR | FATAL ) compileTimeMessage |
    INLINE | NOINLINE | FRAMEWORK | IBACTION | IBOUTLET | QUALIFIED |
    implementationDefinedPragma ( “+” | “-” | “=” ( ident | number ) )?
    “*>” 

(89) compileTimeMessage = string 

(90) implementationDefinedPragma = ident 

Terminal Symbols

(91) ident 
    ( “_” | “$” | LETTER ) ( “_” | “$” | LETTER | DIGIT )* 

(92) colonIdent 
    ident “:” 

(93) number 
    DIGIT+ | 
    BINARY-DIGIT+ “B” | 
    DIGIT SEDECIMAL-DIGIT* ( “C” | “H” ) | 
    DIGIT+ “.” DIGIT+ ( “E” ( “+” | “-” )? DIGIT+ )? 

(94) string 
    “’” ( CHARACTER | “’” )* “’” | 
    “’” ( CHARACTER | “’” )* “’” 

(95) DIGIT 
    “A” .. “Z” | “a” .. “z” 

(96) DIGIT 
    BINARY-DIGIT | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9” 

(97) BINARY-DIGIT 
    “0” | “1” 

(98) SEDECIMAL-DIGIT 
    DIGIT | “A” | “B” | “C” | “D” | “E” | “F”
EBNF Grammar for Objective Modula-2

(99) CHARACTER

DIGIT | LETTER |
" " | "!" | "#" | "$" | "%" | "&" | "(" | ")" | "+" | "-" |
"," | "-" | "." | "," | "<" | "=" | ">" | "?" | "@" |
"{" | "|" | "}" | "~" | ESCAPE-SEQUENCE

(100) ESCAPE-SEQUENCE

"\" ( "0" | "n" | "r" | "t" | "\" | "'" | "" | "'" )

Ignore Symbols

(101) WHITESPACE

" " | ASCII-TAB

(102) COMMENT

NESTABLE-COMMENT | NON-NESTABLE-COMMENT | SINGLE-LINE-COMMENT

(103) NESTABLE-COMMENT

"(*) ( ANY-CHAR | END-OF-LINE )* NESTABLE-COMMENT* ")"

(104) NON-NESTABLE-COMMENT

"/" ( ANY-CHAR | END-OF-LINE ) "/"

(105) SINGLE-LINE-COMMENT

"//" ANY-CHAR END-OF-LINE

(106) ANY-CHAR

ASCII(8) | ASCII(32) .. ASCII(127) | ANY-UNICODE

(107) END-OF-LINE

ASCII-LF ASCII-CR | ASCII-CR ASCII-LF
Reserved Words

AND ARRAY BEGIN BY BYCOPY BYREF CASE CLASS CONST CONTINUE CRITICAL DEFINITION DIV DO ELSE ELSIF END ENUM EXIT FOR FROM IF IMMUTABLE IMPLEMENTATION IMPORT IN INOUT IS LOOP METHOD MOD MODULE NOT OF ON OPAQUE OPTIONAL OR OUT POINTER PRIVATE PROCEDURE PROTECTED PROTOCOL PUBLIC RECORD REPEAT RETURN SET SUPER THEN TO TRY TYPE UNTIL VAR VARIADIC WHILE

Reserved Symbols

:= + - * / ++ -- & ~ = < <= > >= " ( ) [ ] { } ^ . , : ; .. :: <* *>

Pragma Identifiers

IF ELSIF ELSE ENDIF INFO WARN ERROR FATAL INLINE NOINLINE FRAMEWORK IBAction IBOutlet QUALIFIED
## Cross Reference

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Rule</th>
<th>Referenced from</th>
</tr>
</thead>
<tbody>
<tr>
<td>actualParameters</td>
<td>80</td>
<td>52, 79</td>
</tr>
<tr>
<td>addOperator</td>
<td>68</td>
<td>67, 76</td>
</tr>
<tr>
<td>adoptedProtocols</td>
<td>9</td>
<td>5, 27</td>
</tr>
<tr>
<td>anonymousType</td>
<td>18</td>
<td>16, 18, 36</td>
</tr>
<tr>
<td>ANY-CHAR</td>
<td>103</td>
<td>100, 101, 102</td>
</tr>
<tr>
<td>arrayIndex</td>
<td>22</td>
<td>21</td>
</tr>
<tr>
<td>arrayType</td>
<td>21</td>
<td>18, 26</td>
</tr>
<tr>
<td>assignmentOrProcedureCall</td>
<td>52</td>
<td>47</td>
</tr>
<tr>
<td>attributedFormalType</td>
<td>33</td>
<td>32</td>
</tr>
<tr>
<td>baseType</td>
<td>20</td>
<td>19, 24</td>
</tr>
<tr>
<td>BINARY-DIGIT</td>
<td>94</td>
<td>90, 93</td>
</tr>
<tr>
<td>block</td>
<td>11</td>
<td>2, 38, 43</td>
</tr>
<tr>
<td>case</td>
<td>55</td>
<td>54</td>
</tr>
<tr>
<td>caseLabelList</td>
<td>56</td>
<td>55</td>
</tr>
<tr>
<td>caseLabels</td>
<td>57</td>
<td>56</td>
</tr>
<tr>
<td>caseStatement</td>
<td>54</td>
<td>47</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>96</td>
<td>91</td>
</tr>
<tr>
<td>classInstance</td>
<td>64</td>
<td>63</td>
</tr>
<tr>
<td>classType</td>
<td>27</td>
<td>16</td>
</tr>
<tr>
<td>colonIdent</td>
<td>89</td>
<td>46, 51</td>
</tr>
<tr>
<td>COMMENT</td>
<td>99</td>
<td>-</td>
</tr>
<tr>
<td>compilationUnit</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>compileTimeMessage</td>
<td>86</td>
<td>85</td>
</tr>
<tr>
<td>constantDeclaration</td>
<td>14</td>
<td>12, 13</td>
</tr>
<tr>
<td>constExpression</td>
<td>65</td>
<td>7, 14, 23, 37, 57, 61, 71, 82, 85</td>
</tr>
<tr>
<td>constFactor</td>
<td>71</td>
<td>69, 71</td>
</tr>
<tr>
<td>constTerm</td>
<td>69</td>
<td>67</td>
</tr>
<tr>
<td>Symbol</td>
<td>Rule</td>
<td>Referenced from</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------</td>
<td>-----------------</td>
</tr>
<tr>
<td>criticalStatement</td>
<td>63</td>
<td>47</td>
</tr>
<tr>
<td>declaration</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>definition</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>definitionOfModule</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>designator</td>
<td>72</td>
<td>52</td>
</tr>
<tr>
<td>designatorOrProcedureCall</td>
<td>79</td>
<td>78</td>
</tr>
<tr>
<td>designatorTail</td>
<td>73</td>
<td>72, 79</td>
</tr>
<tr>
<td>DIGIT</td>
<td>93</td>
<td>88, 90, 95, 96</td>
</tr>
<tr>
<td>END-OF-LINE</td>
<td>104</td>
<td>100, 101, 102</td>
</tr>
<tr>
<td>enumerationType</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>ESCAPE-SEQUENCE</td>
<td>97</td>
<td>96</td>
</tr>
<tr>
<td>expression</td>
<td>75</td>
<td>47, 51, 52, 53, 54, 58, 59, 61, 74, 78</td>
</tr>
<tr>
<td>expressionList</td>
<td>74</td>
<td>73, 80</td>
</tr>
<tr>
<td>factor</td>
<td>78</td>
<td>77, 78</td>
</tr>
<tr>
<td>fieldList</td>
<td>26</td>
<td>25</td>
</tr>
<tr>
<td>fieldListSequence</td>
<td>25</td>
<td>24, 27</td>
</tr>
<tr>
<td>formalParamList</td>
<td>41</td>
<td>39</td>
</tr>
<tr>
<td>formalParams</td>
<td>42</td>
<td>41</td>
</tr>
<tr>
<td>formalType</td>
<td>34</td>
<td>33, 42, 46</td>
</tr>
<tr>
<td>formalTypeList</td>
<td>32</td>
<td>31</td>
</tr>
<tr>
<td>forStatement</td>
<td>61</td>
<td>47</td>
</tr>
<tr>
<td>handle</td>
<td>44</td>
<td>43</td>
</tr>
<tr>
<td>ident</td>
<td>88</td>
<td>6, 8, 13, 14, 15, 36, 38, 39, 40, 43, 44, 46, 50, 51, 61, 62, 73, 83, 84, 85, 87, 89</td>
</tr>
<tr>
<td>identList</td>
<td>84</td>
<td>9, 10, 19, 26, 29, 36, 42</td>
</tr>
<tr>
<td>ifStatement</td>
<td>53</td>
<td>47</td>
</tr>
<tr>
<td>implementationDefinedPragma</td>
<td>87</td>
<td>85</td>
</tr>
<tr>
<td>implementationOfModule</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
## EBNF Grammar for Objective Modula-2

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Rule</th>
<th>Referenced from</th>
</tr>
</thead>
<tbody>
<tr>
<td>importList</td>
<td>10</td>
<td>2, 3, 5</td>
</tr>
<tr>
<td>indexParam</td>
<td>45</td>
<td>43</td>
</tr>
<tr>
<td>LETTER</td>
<td>92</td>
<td>88, 96</td>
</tr>
<tr>
<td>loopStatement</td>
<td>60</td>
<td>47</td>
</tr>
<tr>
<td>machineAddress</td>
<td>37</td>
<td>36</td>
</tr>
<tr>
<td>message</td>
<td>51</td>
<td>49</td>
</tr>
<tr>
<td>methodArg</td>
<td>46</td>
<td>44</td>
</tr>
<tr>
<td>methodDeclaration</td>
<td>43</td>
<td>12</td>
</tr>
<tr>
<td>methodHeader</td>
<td>44</td>
<td>5, 13, 43</td>
</tr>
<tr>
<td>methodInvocation</td>
<td>49</td>
<td>47, 50, 78</td>
</tr>
<tr>
<td>moduleId</td>
<td>6</td>
<td>2, 3</td>
</tr>
<tr>
<td>mulOperator</td>
<td>70</td>
<td>69, 77</td>
</tr>
<tr>
<td>namedType</td>
<td>17</td>
<td>16, 21, 26, 29, 30, 34, 35</td>
</tr>
<tr>
<td>NESTABLE-COMMENT</td>
<td>100</td>
<td>99, 100</td>
</tr>
<tr>
<td>NON-NESTABLE-COMMENT</td>
<td>101</td>
<td>99</td>
</tr>
<tr>
<td>number</td>
<td>90</td>
<td>71, 78, 85</td>
</tr>
<tr>
<td>ordinalConstExpression</td>
<td>23</td>
<td>22</td>
</tr>
<tr>
<td>PointerType</td>
<td>30</td>
<td>18</td>
</tr>
<tr>
<td>pragma</td>
<td>85</td>
<td>–</td>
</tr>
<tr>
<td>priority</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>procedureDeclaration</td>
<td>38</td>
<td>12</td>
</tr>
<tr>
<td>procedureHeader</td>
<td>39</td>
<td>13, 38</td>
</tr>
<tr>
<td>procedureType</td>
<td>32</td>
<td>18, 21, 26</td>
</tr>
<tr>
<td>programModule</td>
<td>2</td>
<td>1, 4</td>
</tr>
<tr>
<td>protocolId</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>protocolModule</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>qualident</td>
<td>83</td>
<td>17, 20, 28, 45, 64, 71, 72, 79</td>
</tr>
<tr>
<td>receiver</td>
<td>50</td>
<td>49</td>
</tr>
<tr>
<td>Symbol</td>
<td>Rule</td>
<td>Referenced from</td>
</tr>
<tr>
<td>---------------------</td>
<td>------</td>
<td>-----------------</td>
</tr>
<tr>
<td>receiverClass</td>
<td>45</td>
<td>44</td>
</tr>
<tr>
<td>receiverType</td>
<td>40</td>
<td>39</td>
</tr>
<tr>
<td>recordType</td>
<td>24</td>
<td>18, 21, 26</td>
</tr>
<tr>
<td>relation</td>
<td>66</td>
<td>65, 75</td>
</tr>
<tr>
<td>repeatStatement</td>
<td>59</td>
<td>47</td>
</tr>
<tr>
<td>returnedType</td>
<td>35</td>
<td>31, 39, 44</td>
</tr>
<tr>
<td>SEDECIMAL-DIGIT</td>
<td>95</td>
<td>90</td>
</tr>
<tr>
<td>setType</td>
<td>29</td>
<td>16</td>
</tr>
<tr>
<td>simpleConstExpr</td>
<td>67</td>
<td>65</td>
</tr>
<tr>
<td>simpleExpression</td>
<td>76</td>
<td>75</td>
</tr>
<tr>
<td>SINGLE-LINE-COMMENT</td>
<td>102</td>
<td>99</td>
</tr>
<tr>
<td>statement</td>
<td>47</td>
<td>48</td>
</tr>
<tr>
<td>statementSequence</td>
<td>48</td>
<td>11, 53, 54, 55, 58, 59, 60, 61, 62, 63</td>
</tr>
<tr>
<td>string</td>
<td>91</td>
<td>71, 78, 86</td>
</tr>
<tr>
<td>structuredValue</td>
<td>81</td>
<td>14, 52, 82</td>
</tr>
<tr>
<td>superClass</td>
<td>28</td>
<td>27</td>
</tr>
<tr>
<td>term</td>
<td>77</td>
<td>76</td>
</tr>
<tr>
<td>tryStatement</td>
<td>62</td>
<td>47</td>
</tr>
<tr>
<td>type</td>
<td>16</td>
<td>13, 15</td>
</tr>
<tr>
<td>typeDeclaration</td>
<td>15</td>
<td>12</td>
</tr>
<tr>
<td>valueComponent</td>
<td>82</td>
<td>81</td>
</tr>
<tr>
<td>variableDeclaration</td>
<td>36</td>
<td>12, 13</td>
</tr>
<tr>
<td>variadicParams</td>
<td>43</td>
<td>41</td>
</tr>
<tr>
<td>whileStatement</td>
<td>58</td>
<td>47</td>
</tr>
<tr>
<td>WHITESPACE</td>
<td>98</td>
<td>–</td>
</tr>
</tbody>
</table>

**Further Reading**

http://objective.modula2.net
http://objective.modula2.net/grammar.shtml

Status: October 4, 2009