GRAMÁTICA MEUPHORIA

Integrantes:

Caio Gomes	7239072
Diego Gomes	7656467
Fernando Cury Gorodscy	7152354
Roberty Manzini Bertolo	7573399
Vanessa Apolinário Lima	7239256

Documento:

Gramática

Nome:

MeuPhoria

1. alpha = "a" .. "z" | "A" .. "Z"
2. digit = "0" .. "9"
3. uscore = "_"
4. eol = "new_line_character"
5. name = (alpha | uscore) { alpha | digit | uscore }
6. spacename = name ":"
7. identifier = [spacename] name
8. atom = atom_integer_lit | atom_real_lit
9. atom_integer_lit = "-2^53" .. "+2^53" | "-9.007e15" .. "+9.007e15"
10. atom_real_lit = "-2^1024+1" .. "+2^1024-1" | "-1.798e308+1" .. "+1.798e308-1"
11. integer = integer_lit
12. integer_lit = "-2^30" .. "+2^30-1" | "-1_073_741_824" .. "+1_073_741_823"
13. sequence = "{" [object] { "," object } ["," "$"] "}"
14. object = atom | integer | sequence
15. expression = atomexpr | intexpr | strexpr | seqexpr | boolexpr
16. atomexpr = atom
17. intexpr = integer
18. strexpr = "an_expression_that_evaluates_to_a_string"

19. seqexpr = sequence
20. bool_lit = "0" | "1" | "0" | "!0"
21. boolexpr = "atom_zero_represents_falsehood_and_non-zero_represents_truth"
22. relationop = "<" | ">" | "<=" | ">=" | "=" | "!="
23. binaryexpr = { expression binop expression }
24. binop = "and" | "or" | "xor" | "+" | "-" | "*" | "/" | "&"
25. unaryexpr = { unaryop expression }
26. unaryop = "not" | "-" | "+"
27. seqenceop = "{" [item | { headitem } item] [lastitem] "}"
28. headitem = object ","
29. item = object
30. lastitem = "," "$"
31. statement = "complete_unit_of_code_executed_by_the_interpreter"
32. stmblk = statement { statement }
33. label = "label" stringlit
34. listdelim = ","
35. stringlit = simplestringlit | rawstringlit
36. simplestringlit = sslitstart { char | escchar } sslitend
37. sslitstart = "\""
38. sslitend = "\""
39. char = "any_byte_value"
40. escchar = esclead ("t" | "n" | "r" | " ")
41. esclead = " "
42. rawstringlit = dqrawstring | bqrawstring
43. dqrawstring = "\"\"\"" [marginstr] { char } "\"\"\""
44. bqrawstring = "`" [ marginstr ] { char } "`"
45. marginstr = "_"
46. scopemodifier = "global" | "public" | "export" | "override"
47. datatype = "atom" | "integer" | "sequence" | "object" | identifier
48. includestmt = "include" fileref ["as" namespaceid] eol
49. fileref = "file_path_that_may_be_enclosed_in_double-quotes"
50. namespaceid = identifier
51. slice = slicestart intexpr slicedelim (intexpr | "$") sliceend
52. slicestart = "["
53. slicedelim = ".."
54. sliceend = "]"
55. ifstmt = iftest { elsif } [else] endif
56. iftest = "if" atomexpr [label] "then" [stmblk]
57. elsif = "elsif" atomexpr "then" [stmblk]
58. else = "else" [stmblk]
59. endif = "end" "if"
60. ifdefstmt = ifdeftest [elsdefif { char }] [elsedef] enddefif
61. ifdeftest = "ifdef" defexpr "then" [stmblk]

62. elsdefif = "elsifedf" defexpr "then" [stmblk]
63. elsedef = "elsede" [stmblk]"
64. enddefif = "end" "ifdef"
65. defexpr = defterm [defop defterm]
66. defterm = ["not" identifier]
67. defop = "and" | "or"
68. switchstmt = switchtest case { case } [caseelse] [endswitch]
69. switchtest = "switch" expression [withfall] [label] "do"
70. withfall = ("with" | "without") "fallthru"
71. case = "case" caselist "then" [stmblk]
72. caselist = expression { listdelim expression }
73. caseelse = "case" "else"
74. endswitch = "end" "switch"
75. breakstmt = "break" [stringlit]
76. continuestmt = "continue" [stringlit]
77. retrystmt = "retry" [stringlit]
78. exitstmt = "exit" [stringlit]
79. fallthrustmt = "fallthru"
80. forstmt = "for" foridx [label] "do" [stmblk] "end" "for"
81. foridx = identifier "=" atomexpr "to" atomexpr ["by" atomexpr]
82. whilestmt = "while" boolexpr [withentry] [label] "do" stmblk [entry] "end" "while"
83. withentry = "with" "entry"
84. entry = "entry" [stmblk]
85. loopstmt = "loop" [withentry] [label] "do" stmblk [entry] "until" boolexpr "end" "loop"
86. gotostmt = "goto" label
87. vardeclare = [scopemodifier] datatype identlist
88. identlist = ident ["," identlist]
89. ident = identifier ["=" expression]
90. constdeclare = [scopemodifier] "constant" identlist
91. enumdeclare = [scopemodifier] [enumval | enumtype]
92. enumval = "enum" ["by" enumdelta] identlist
93. enumdelta = ["+" | "-" | "*" | "/"] atomexpr
94. enumtype = "enum" "type" ["by" enumdelta] identlist "end" "type"
95. call = identifier "(" [arglist] ")"
96. arglist = argument ["," arglist]
97. argument = expression
98. procdeclare = [scopemodifier] "procedure" identifier "(" [paramlist] ")" [stmblk] "end" "procedure"
99. paramlist = parameter ["," paramlist]
100. parameter = datatype identifier
101. funcdeclare = [scopemodifier] "function" identifier "(" [paramlist] ")" [stmblk] "end" "function"
102. typedeclar = [scopemodifier] "type" identifier "(" parameter ")" [stmblk] "end" "type"

103. return = "return" expression [{ "," expression }] **-- CARACTERÍSTICA MEUPHORIA**
104. namespace = "namespace" identifier eol
105. withstmt = ["with" | "without"] withoption
106. withoption = ["profile" | "profile_time" | "trace" | "batch" | "type_check" | "indirect_includes" | "inline" | withwarning]
107. withwarning = "warning" [warnopt]
108. warnopt = setwarn | addwarn? | savewarn | restorewarn | strictwarn
109. setwarn = ["+=" | "&="] "{" warnlist? "}"
110. savewarn = "save"
111. restorewarn = "restore"
112. strictwarn = "strict"
113. subscripting = identifier { index }
114. index = "[" intexpr "]"
115. assignmono = identifier "=" expression
116. assignmulti = "{" identifier [{ "," identifier }] "}" "=" expression | "{" expression [{ "," expression }] "}"
117. assignwithop = identifier ("+" | "-" | "/" | "*" | "&") "=" expression