

```
1  //      Pseudo-code Syntax Definitions
2  //
3  //      This file is designed to showcase the pseudo-code library for Notepad++. It
4  //      has been built with the following syllabuses in mind.
5  //          * Cambridge International AS & A Level Computer Science 9608
6  //          * Cambridge IGCSE Computer Science 0478
7  //      This document closely follows the requirements of the syllabus.
8  //
9  //
10 //      Designed and maintained by
11 //      Anuj Verma,
12 //      Certified LabVIEW Associate Developer
13 //
14 //      Last updated 02:56 01-04-2020
15
16
17 /* 1 Pseudo-code in examined components */
18 // 1.5 Comments
19 // This is a comment
20
21 // The C++ style comments /* */ are not allowed in Cambridge International AS & A
22 // Level Computer Science
23 // and Cambridge IGCSE Computer Science. They are used here for explanation purposes
24 // only
25
26 /* 2 Variables, constants and data types */
27 // 2.1 Atomic type names
28 INTEGER
29 REAL
30 CHAR
31 STRING
32 BOOLEAN
33 DATE
34
35 // 2.2 Literals
36 -5
37 3.141
38 'H'
39 "Hello"
40 FALSE
41 09/07/2019
42
43 // 2.3 Identifiers
44 VariableName
45
46 // 2.5 Variable declarations
47 DECLARE <<identifier>> : <<data type>>
48
49 // 2.6 Constants
50 CONSTANT <<identifier>> : <<value>>
51
52 // 2.7 Assignments
53 <<identifier>> ← <<value>>
54
55 /* 3 Arrays */
56 // 3.1 Declaring arrays
57 DECLARE <<identifier>> : ARRAY[<<l1>>:<<u>>] OF <<data type>>
58 DECLARE <<identifier>> : ARRAY[<<l1>>:<<u1>>, <<l2>>:<<u2>>] OF <<data type>>
59
60 // 3.2 Using arrays
61 <<array 1>> ← <<array 2>>
62
63
64
65
66
```

```

67  /* 4 Abstract data types */
68
69  // 4.1 Defining custom types
70  TYPE <<identifier>>
71      DECLARE <<SubIdentifier1>> : <<data type>>
72      DECLARE <<SubIdentifier2>> : <<data type>>
73      ...
74  ENDTYPE
75
76  // 4.2 Using custom types
77  <<identifier>>. <<SubIdentifier>>
78
79
80  /* 5 Common operations */
81
82  // 5.1 Input and output
83  INPUT <<identifier>>
84  OUTPUT <<identifier 1>> [, <<identifier 2>> ... <<identifier n>>]
85
86  // 5.2 Arithmetic operations
87  <<addend 1>> + <<addend 2>>
88  <<minuend>> - <<subtrahend>>
89  <<multiplicand>> * <<multiplier>>
90  <<dividend>> / <<divisor>>
91
92  // 5.3 Relational operations
93  <<value 1>> > <<value 2>>
94  <<value 1>> < <<value 2>>
95  <<value 1>> >= <<value 2>>
96  <<value 1>> <= <<value 2>>
97  <<value 1>> = <<value 2>>
98  <<value 1>> <> <<value 2>>
99
100 // 5.4 Logic operations
101 <<value 1>> AND <<value 2>>
102 <<value 1>> OR <<value 2>>
103 NOT <<value>>
104
105 // 5.6 Random number generation
106 RANDOMBETWEEN (<<minimum>>, <<maximum>>)
107 RND()
108
109
110 /* 6 Selection */
111
112 // 6.1 IF statements
113 IF <<condition>>
114     THEN
115         <<statements>>
116     ENDIF
117
118 IF <<condition>>
119     THEN
120         <<statements>>
121     ELSE
122         <<statements>>
123     ENDIF
124
125 // 6.2 Case statements
126 CASE OF <<identifier>>
127     <<value 1>> : <<statement>>
128     <<value 2>> : <<statement>>
129     ...
130 ENDCASE
131
132
133
134
135

```

```

136 CASE OF <<identifier>>
137     <<value 1>> : <<statement>>
138     <<value 2>> : <<statement>>
139     ...
140 OTHERWISE <<statement>>
141 ENDCASE
142
143
144 /* 7 Iteration */
145
146 // 7.1 Count-controlled (FOR) loops
147 FOR <<identifier>> ← <<value 1>> TO <<value 2>>
148     <<statements>>
149 ENDFOR
150
151 FOR <<identifier>> ← <<value 1>> TO <<value 2>> STEP <<increment>>
152     <<statements>>
153 ENDFOR
154
155 // 7.2 Post-condition (REPEAT UNTIL) loops
156 REPEAT
157     <<statements>>
158 UNTIL <<condition>>
159
160 // 7.3 Pre-condition (WHILE) loops
161 WHILE <<condition>> DO
162     <<statements>>
163 ENDWHILE
164
165
166 /* 8 Procedures and functions */
167
168 // 8.1 Defining and calling procedures
169 PROCEDURE <<identifier>>
170     <<statements>>
171 ENDPROCEDURE
172
173 PROCEDURE <<identifier>> (<<param1>>:<<data-type>>, <<param2>>:<<data-type>>...)
174     <<statements>>
175 ENDPROCEDURE
176
177 CALL <<identifier>>
178
179 CALL <<identifier>> (<<value 1>>, <<value 2>>...)
180
181 //8.2 Defining and calling functions
182 FUNCTION <<identifier>> RETURNS <<data type>>
183     <<statements>>
184     RETURN <<value>>
185 ENDFUNCTION
186
187 FUNCTION <<identifier>> (<<param1>>:<<data-type>>, <<param2>>:<<data-type>>...)
188     RETURNS <<data type>>
189     <<statements>>
190     RETURN <<value>>
191 ENDFUNCTION
192
193 <<identifier>>
194 <<identifier>> (<<value 1>>, <<value 2>>...)
195
196 // 8.3 Passing parameters by value or by reference
197 PROCEDURE <<identifier>> (BYREF <<param1>>:<<data-type>>, <<param2>>:<<data-type>>...)
198     <<statements>>
199 ENDPROCEDURE
200
201 PROCEDURE <<identifier>> (BYVALUE <<param1>>:<<data-type>>, <<param2>>:<<data-type>>
202     ...)
203     <<statements>>
204 ENDPROCEDURE

```

```
203  /* 9 File handling */
204
205  // 9.1 Handling text files
206  OPENFILE <<file identifier>> FOR <<file mode>>
207
208  READFILE <<file identifier>>, <<variable>>
209
210  EOF (<<file identifier>>)
211
212  WRITEFILE <<file identifier>>, <<string>>
213
214  CLOSEFILE <<file identifier>>
215
216  // 9.2 Handling random files
217  OPENFILE <<file identifier>> FOR RANDOM
218
219  SEEK <<file identifier>>, <<address>>
220
221  GETRECORD <<file identifier>>, <<variable>>
222
223  PUTRECORD <<file identifier>>, <<variable>>
224
```