```
1   //    Pseudo-code Syntax Definitions
2   //
3   //    This file is designed to showcase the pseudo-code library for Notepad++. It
    has been built with the following syllabuses in mind.
4   //        * Cambridge International AS & A Level Computer Science 9608
5   //        * Cambridge IGCSE Computer Science 0478
6   //
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 23:00 09-07-2019
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
    Level Computer Science
22  // and Cambridge IGCSE Computer Science. They are used here for explanation purposes
    only
23
24
25  /* 2 Variables, constants and data types */
26  // 2.1 Atomic type names
27  INTEGER
28  REAL
29  CHAR
30  STRING
31  BOOLEAN
32  DATE
33
34  // 2.2 Literals
35  -5
36  3.141
37  'H'
38  "Hello"
39  FALSE
40  09/07/2019
41
42  // 2.3 Identifiers
43  VariableName
44
45  // 2.5 Variable declarations
46  DECLARE <<identifier>> : <<data type>>
47
48  // 2.6 Constants
49  CONSTANT <<identifier>> : <<value>>
50
51  // 2.7 Assignments
52  <<identifier>> ← <<value>>
53
54
55  /* 3 Arrays */
56  // 3.1 Declaring arrays
57  DECLARE <<identifier>> : ARRAY[<<l>>:<<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
```

```
/* 4 Abstract data types */

// 4.1 Defining custom types
TYPE <<identifier>>
    DECLARE <<SubIdentifier1>> : <<data type>>
    DECLARE <<SubIdentifier2>> : <<data type>>
    ...
ENDTYPE

// 4.2 Using custom types
<<identifier>>.<<SubIdentifier>>


/* 5 Common operations */

// 5.1 Input and output
INPUT <<identifier>>
OUTPUT <<identifier 1>> [, <<identifier 2>> ... <<identifier n>>]

// 5.2 Arithmetic operations
<<addend 1>> + <<addend 2>>
<<minuend>> - <<subtrahend>>
<<multiplicand>> * <<multiplier>>
<<dividend>> / <<divisor>>

// 5.3 Relational operations
<<value 1>>  > <<value 2>>
<<value 1>>  < <<value 2>>
<<value 1>>  >= <<value 2>>
<<value 1>>  <= <<value 2>>
<<value 1>>  = <<value 2>>
<<value 1>>  <> <<value 2>>

// 5.4 Logic operations
<<value 1>> AND <<value 2>>
<<value 1>> OR <<value 2>>
NOT <<value>>

// 5.6 Random number generation
RANDOMBETWEEN (<<minimum>>, <<maximum>>)
RND()


/* 6 Selection */

// 6.1 IF statements
IF <<condition>>
  THEN
    <<statements>>
ENDIF

IF <<condition>>
  THEN
    <<statements>>
  ELSE
    <<statements>>
ENDIF

// 6.2 Case statements
CASE OF <<identifier>>
    <<value 1>> : <<statement>>
    <<value 2>> : <<statement>>
    ...
ENDCASE
```

```
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>>...)
      RETURNS <<data type>>
188       <<statements>>
189       RETURN <<value>>
190   ENDFUNCTION
191
192   <<identifier>>
193   <<identifier>> (<<value 1>>, <<value 2>>...)
194
195   // 8.3 Passing parameters by value or by reference
196   PROCEDURE <<identifier>> (BYREF <<param1>>:<<data-type>>, <<param2>>:<<data-type>>...)
197       <<statements>>
198   ENDPROCEDURE
199
200   PROCEDURE <<identifier>> (BYVALUE <<param1>>:<<data-type>>, <<param2>>:<<data-type>>
      ...)
201       <<statements>>
202   ENDPROCEDURE
```

```
/* 9 File handling */

// 9.1 Handling text files
OPENFILE <<file identifier>> FOR <<file mode>>

READFILE <<file identifier>>, <<variable>>

EOF (<<file identifier>>)

WRITEFILE <<file identifier>>, <<string>>

CLOSEFILE <<file identifier>>

// 9.2 Handling random files
OPENFILE <<file identifier>> FOR RANDOM

SEEK <<file identifier>>, <<address>>

GETRECORD <<file identifier>>, <<variable>>

PUTRECORD <<file identifier>>, <<variable>>
```