```
//    Pseudo-code Syntax Definitions
//
//    This file is designed to showcase the pseudo-code library for Notepad++. It has been built with the following syllabuses in mind.
//        * Cambridge International AS & A Level Computer Science 9608
//        * Cambridge IGCSE Computer Science 0478
//
//    This document closely follows the requirements of the syllabus.
//
//
//    Designed and maintained by
//    Anuj Verma,
//    Certified LabVIEW Associate Developer
//
//    Last updated 21:56 05-10-2020 AD


/* 1 Pseudo-code in examined components */
// 1.5 Comments
// This is a comment

// The C++ style comments /* */ are not allowed in Cambridge International AS & A Level Computer Science
// and Cambridge IGCSE Computer Science. They are used here for explanation purposes only


/* 2 Variables, constants and data types */
// 2.1 Atomic type names
INTEGER
REAL
CHAR
STRING
BOOLEAN
DATE

// 2.2 Literals
-5
3.141
'H'
"Hello"
FALSE
09/07/2019

// 2.3 Identifiers
VariableName

// 2.5 Variable declarations
DECLARE <<identifier>> : <<data type>>

// 2.6 Constants
CONSTANT <<identifier>> : <<value>>

// 2.7 Assignments
<<identifier>> ← <<value>>


/* 3 Arrays */
// 3.1 Declaring arrays
DECLARE <<identifier>> : ARRAY[<<l>>:<<u>>] OF <<data type>>
DECLARE <<identifier>> : ARRAY[<<l1>>:<<u1>>, <<l2>>:<<u2>>] OF <<data type>>

// 3.2 Using arrays
<<array 1>> ← <<array 2>>
```

```
/* 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
```

```
CASE OF <<identifier>>
    <<value 1>> : <<statement>>
    <<value 2>> : <<statement>>
    ...
    OTHERWISE <<statement>>
ENDCASE


/* 7 Iteration */

// 7.1 Count-controlled (FOR) loops
FOR <<identifier>> ← <<value 1>> TO <<value 2>>
    <<statements>>
ENDFOR

FOR <<identifier>> ← <<value 1>> TO <<value 2>> STEP <<increment>>
    <<statements>>
ENDFOR

// 7.2 Post-condition (REPEAT UNTIL) loops
REPEAT
    <<statements>>
UNTIL <<condition>>

// 7.3 Pre-condition (WHILE) loops
WHILE <<condition>> DO
    <<statements>>
ENDWHILE


/* 8 Procedures and functions */

// 8.1 Defining and calling procedures
PROCEDURE <<identifier>>
    <<statements>>
ENDPROCEDURE

PROCEDURE <<identifier>> (<<param1>>:<<data-type>>, <<param2>>:<<data-type>>...)
    <<statements>>
ENDPROCEDURE

CALL <<identifier>>

CALL <<identifier>> (<<value 1>>, <<value 2>>...)

//8.2 Defining and calling functions
FUNCTION <<identifier>> RETURNS <<data type>>
    <<statements>>
    RETURN <<value>>
ENDFUNCTION

FUNCTION <<identifier>> (<<param1>>:<<data-type>>, <<param2>>:<<data-type>>...)
RETURNS <<data type>>
    <<statements>>
    RETURN <<value>>
ENDFUNCTION

<<identifier>>
<<identifier>> (<<value 1>>, <<value 2>>...)

// 8.3 Passing parameters by value or by reference
PROCEDURE <<identifier>> (BYREF <<param1>>:<<data-type>>, <<param2>>:<<data-type>>...)
    <<statements>>
ENDPROCEDURE

PROCEDURE <<identifier>> (BYVALUE <<param1>>:<<data-type>>, <<param2>>:<<data-type>>
...)
    <<statements>>
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>>

CLOSEFILE <<file identifier>>
```