Digital Solutions 2019 v1.2

Supporting resource: Representing algorithms with pseudocode

Purpose

The purpose of this resource is to provide supporting information to the syllabus requirements for *Digital Solutions 2019*.

Syllabus subject matter

Algorithmic design method

Pseudocode will be used as the formal method of representing algorithms in this syllabus. Pseudocode is a descriptive method used to represent an algorithm and is a mixture of everyday language and programming conventions.

Pseudocode implements the basic control structures of assignment, sequence, selection, condition, iteration and modularisation through the use of keywords associated with the constructs, and textual indentation. Used to show how a computing algorithm should and could work, it is often an intermediate step in programming between the planning stage and writing executable code.

Pseudocode can also be useful for:

- demonstrating thinking that later can become comments in the final program
- describing how an algorithm should work
- explaining a computing process to less technical people
- generating code in collaboration with others.

Pseudocode does not have a standard format and varies from programmer to programmer. However, a number of conventions are generally used.

Conventions for writing pseudocode

KEYWORDS are written in bold capitals and are often words taken directly from programming languages. For example, **IF**, **THEN** and **ELSE** are all words that can be validly used in most languages. **OUTPUT** and **COMPUTE** are from the language COBOL and **WRITE** is from the language Pascal.
Keywords do not have to be valid programming language words as long as they clearly convey the intent of the line of pseudocode.

Statements that form part of a **REPETITION LOOP** are indented by the same amount to indicate that they form a logical grouping.

In a similar way, **IF**, **THEN** and **ELSE** statements are indented to clearly distinguish the alternative processing paths.

The end of **REPETITION LOOPS** and **IF**, **THEN** and **ELSE** statements are explicitly indicated by the use of **ENDWHILE** and **ENDIF** at the appropriate points.

Pseudocode should clearly indicate what is happening at each step, including formulas of calculations.
For example:
CALCULATE net is not as clear as **CALCULATE** net = gross − tax.

Programmers prefer to use a more abbreviated version in which memory cells used to store the input are given program-like names.
For example:
INPUT num1
INPUT num2
is preferable to
INPUT first number
INPUT second number

See: Subject matter in the *Digital Solutions 2019 syllabus*
www.qcaa.qld.edu.au/senior/senior-subjects/technologies/digital-solutions/syllabus

Further considerations

Digital Solutions 2019 subject matter describes conventions for writing pseudocode (above). While these are not exhaustive, additional information outlined in the tables that follow is used when providing students with learning opportunities.

Additional considerations for writing pseudocode

Language

Common keywords are written in bold capitals. Keywords do not have to be valid programming language words as long as they clearly convey the intent of the line of pseudocode.

Statements in a block are indented by the same amount to show hierarchy.

Naming convention

Use camel case naming convention for variables, subroutines, methods and functions.

Modularisation

Pseudocode always starts and ends with the **BEGIN** and **END** keywords.

Main algorithm:

```
BEGIN
   statements
END
```

Procedures, subroutines, methods or functions:

```
BEGIN name
   statements
END name
```

Variables

Programmers use names without spaces for variables. In pseudocode, this will make the algorithm.

```
INPUT num1    is preferable to  INPUT FirstNumber
INPUT num2    is preferable to  INPUT SecondNumber
```

To input, assign or output values, common words can be used as keywords.

For example:

```
INPUT mark                WRITE "the total is" count      PRINT x, y
DISPLAY name, result      READ name from list.txt         OUTPUT average
```

Assignment

Pseudocode should clearly indicate what is happening at each step. For example:

`CALCULATE net = gross - tax` is clearer than `CALCULATE net`

Selection

A control structure used for decisions or branching and choosing alternate paths.

The beginning and end of these structures are indicated with keywords (for multiple branches).

```
IF condition THEN          IF condition THEN
     statements               statements
ENDIF                     ELSE
                             statements
                          ENDIF
```

Iterations (loops)

Control structures to provide repetitions. There are three main types of loops.

Each has a clear start and end, with the statements within the loop indented.

For post-test loops:

```
REPEAT
   statements
UNTIL condition
```

For pre-test loops:

```
WHILE condition
   statements
ENDWHILE
```

For counted loops:

```
FOR count = startVal TO endVal
   statements
NEXT count
```

Other statement types and other constructs can be represented in similar ways.

Font

A mono-space typeface, such as *Courier New*, is recommended when writing algorithms on computer.

Vertical quotation marks should also be used. e.g. " and .

Supplementary explanations

The following explanations also provide support for teaching and learning.

Term	Explanation
efficiency	a situation in which a system or machine uses minimal resources such as time and processing power while still achieving its goals. There are two types algorithmic and code efficiency. Algorithmic efficiency refers to the reliability, speed and programming methodology for developing succinct structures within an application. Code efficiency is directly linked with algorithmic efficiency and the speed of runtime execution for software. It is the key element in ensuring high performance. The goal of code efficiency is to reduce resource consumption and completion time as much as possible with minimum risk to the business or operating environment. The software product quality can be evaluated using algorithm or code used efficiency.
maintainability	easy to read code, that is easy to dissect so parts relating to a required change is easy to modify without risking a chain reaction of errors in dependant modules
reliability of software or hardware	attribute of software (data and algorithms) or hardware that consistently performs without failure and according to its' specifications
technical specification	a set of requirements that a product must meet or exceed. In Digital Solutions: Program specifications describe what the software is required to achieve. Functional specifications describe the manner in which the program specifications are achieved. These specifications may be regarded as prescribed criteria.