Southern New Hampshire University

# IT 511 How to Write Pseudocode

Pseudocode is an intermediary step between reading a problem statement and writing the code to solve the problem. It serves as a blueprint for your program in order to guide you through it, in much the same way that contractors start with a blueprint before building a house. Use it as a tool to begin thinking about your program, but keep in mind it might not be the final solution to the problem. Pseudocode is written in a natural language using some programming keywords. Consider this example:

```
INCREMENT the number of apples in the basket by one
```

Notice how the example fully describes, in natural language, what needs to be done in the program. When writing pseudocode, start at the beginning of what you need the program to do and then work through step by step until reaching the end of what is required by the program in the problem statement. This is putting the problem in sequence. For example, making a peanut butter sandwich could be written as follows:

```
OBTAIN a plate
OBTAIN two slices of bread
OBTAIN a jar of peanut butter
OBTAIN a knife
Place the slices of bread on the plate
Open the jar of peanut butter
Spread peanut butter on one bread slice with the knife
Place the empty slice of bread on top of the slice with peanut butter
Serve
```

There are several common keywords that get capitalized because they refer to actions taken in the program. Those words include READ, WRITE, PRINT, DISPLAY, CALCULATE, SET, and INCREMENT. Choices and loops can be shown in pseudocode. When an item is nested inside another item, indent that line of pseudocode, just like coding. Below are three generic examples:

```
IF condition THEN
        Include the first sequence
ELSE
        Include the second sequence
ENDIF

WHILE condition
        Include the sequence
ENDWHILE

FOR loop parameters
        Include the sequence
ENDFOR
```

You can also use the keyword CALL to reference another algorithm written separately. Now look at a more complete example of both good and bad pseudocode to get a general feel of how to write it:

**Bad Example—Vague and Incomplete**

```
function doProgrammingHomework():
        Get things for homework
        Write the code correctly
        Finish the homework
```

**Bad Example—Too Technical, Does Not Follow Natural Language Usage**

```
function doProgrammingHomework():
        getComputer();
        openBlackboard();
        for (var count = 0; count < problems.length(); count++)
                solve();
                while (!compile)
                        debug();
                submit()
        shutDownComputer();
```

**Good Example–Follows Steps One at a Time Through the End of the Algorithm**

```
function doProgrammingHomework():
        GET a computer
        OPEN the Blackboard module
        FOR each of the problems in the module
                Complete problem
                WHILE the problem does not compile
                        Debug
                ENDWHILE
                Submit the assignment
        ENDFOR
        Shut down the computer
```

Remember not to make your pseudocode too technical. You are not trying to write the code itself, just a plan to be used as a stepping stone after the initial problem to get your creative juices flowing.