Assignment Part 3 – Ada

# Program Testing Procedure

Test bubble sort procedure with differently sized/ordered arrays and observe output:

|  |  |  |
| --- | --- | --- |
| **Test name** | **Test array** | **Expected output** |
| Already sorted | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 |
| Reverse sorted | 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 |
| Random order | 7, 6, 3, 5, 9, 4, 8, 10, 1, 2 | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 |
| Single element | 42 | 42 |
| Empty array | <nothing> | <nothing> |

Make sure console output formatting is correct (we still haven’t gotten to languages that default to printing numbers without alignment!).

# Weekly Question

The general structure and control flow of the Ada and C versions are the same (or at least can easily be made the same). They both utilise nested control structures.  
The looping mechanisms differ slightly, however. For the outer loop, C will probably use a do…while, while Ada has loop and exit when. For the inner loop, C will probably use a for loop with an explicitly managed loop counter, while Ada has a slightly more expressive/abstract for…in that loops automatically over a range.

The overall syntax of the C version is somewhat more streamlined than Ada. C uses braces rather than Ada’s more verbose block keywords such as begin, end, loop, and then. Ada uses := for assignment, one more character than C’s =. Ada also has some constructs which may not strictly be necessary, for example using the range keyword before specifying start..end, and loop following for. It may be because I am far more used to C’s syntax, but I believe the extra verbosity in the Ada code makes it “noisier” and more difficult to read compared to the C version. However, this added verbosity likely improves reliability, a design goal for Ada.

Ada has a built-in array type, making the handling of arrays slightly easier, more abstract, and more reliable than the C version. Ada’s arrays can store their size, while C arrays are simply pointers with no additional information. Due to this, the C version, if defined as a procedure, must accept another parameter specifying the size of the array. Additionally, the C version is more prone to bugs as a result of the programmer providing the wrong array size. The Ada version has no such problem, as the size is handled by the language itself.

Finally, (modern) C allows variable definitions at any point in a procedure, whereas Ada requires them to be before the procedure body. Requiring the definitions to be before the body reduces readability, writability, and reliability, as it is possible that variables’ scope must be inflated beyond where they are used. For example, the Tmp variable in the Ada version is in scope for the entire procedure, even though it is only used within the inner if block. In C, this variable could be constrained to where it is used.

# Reflection

Ada’s compliance with the abstraction principle with respect to arrays and loops improves readability, writability, and reliability. Array types store the range of indices which are valid as the Range member, which can be used with the for…in… loop to automatically loop over the indices. Such functionality is very easy to use since there is no need to manually manage the loop variable. It is similarly easy to read, as the functionality is intuitive and concise. Finally, since the loop variable an exit condition is handled by the compiler, it is less prone to bugs as a result of human error.