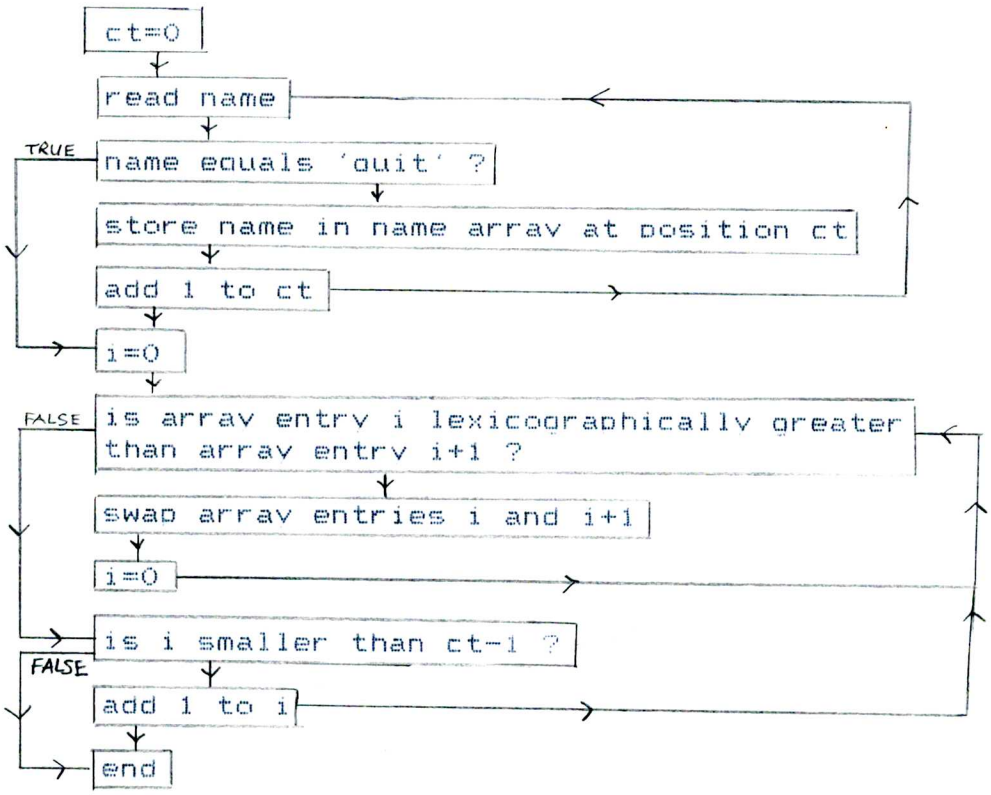
Unit 7: Linear relationships (2) Process
Consolidation: C
Title: The bubblesort algorithm
Author: Erik van Evkelen, E12A
Date: sep-3-89



The bubblesort algorithm can be used in many programming languages to sort small lists of (alpha)numerical data. It's name is derived from the fact that the lexicographically smaller entries come floating upwards in the list. Floating (in nature) is usually a slow process and this applies also to the bubblesort sorting algorithm. Other algorithms do lesser swapping and copying of data as this costs a lot of processing time. The advantage is that it is easy to grasp and therefore easy to covert to working code.

Firstly, the variable ct (count) is put to zero. Then the user is asked to enter a name. After the user has pressed the <ENTER> key, the program checks if the word 'quit' was typed as this word ends the name entry part of the program. If this is not true, the name is stored in an array (a piece of memory). The array locations can be accessed by giving a number. Array location 0 contains the first name, array location 1 the next etc. Before the program asks for the next name, ct is incremented by one because the next name should be stored in the next array location.

When the user types 'quit', the actual bubblesort part comes into action. A new variable (i) is put to zero. Then a lexicographical comparison is carried out between array entry n (initially 0) and array entry n+1 (initially containing a 1).

If the comparison judges that the first entry is greater than the second, these two entries are swapped. The whole process begins again after this swap so i is set to 0 again.

If the outcome of the comparison is false, the program should evaluate if all entries have been checked. This is done by checking if the number of entered names (ct) is one less than the number of sorted elements. The -1 in 'is i smaller than ct-1 ?' is added because the algorithm checks the entries in pairs of two. If the evaluation of 'is i smaller than ct-1' is true, the program increments i and jumps to the array element comparing part.

If all conditions are met, the program ends holding a sorted list into memory.

Very interesting + well done

9