CSC 122 001 Computer Science II
Julius Ranoa

Documentation for Program 16-19.

OVERVIEW.

The program processes two words that have the same letter counts for each type of letter (e.g. ABBA and BABA; these two words have two A's and two B's) and shows how the first one can become the second word by moving a letter at a time.

Note. *This program does not test whether the words given have the same letter counts. The example is hard-coded and is guaranteed to satisfy that requirement. If, however, two words that do not have the same letter counts had been given, the program will display a false positive.*

GENERAL ALGORITHM.

The program does this by sorting letters of two words in ascending order and recording the indexes involved in the swaps taken to accomplish the sorting in an integer vector. Since both words have the same numbers for each letter type, the two words will converge into the same output. That same output is used to connect the two transformations by presenting the transformation of the first word to the output and then the transformation of the output to the second word (the second process is reversed).

THE SORT FUNCTION.

The recording of indexes in the integer vector is done by the *sort()* function in the program. It accepts three arguments: (1) the pointer to the character array for the word, (2) the size of the word, and (3) a reference to the vector where the indexes will be saved.

The algorithm used is a variation of the bubble sort algorithm. In this variation of the bubble sort algorithm, the current index in the loop is tested against the object next to it (index + 1). If the object at the current index is larger than the object next to it, the objects are swapped and the current index is saved. The consistency of swapping the object to the one next to it makes recording the steps easy. One consequence of this function is that it loses the original word after sorting since it does not make a copy of it. A copy has to be explicitly made before sorting so as not to lose the data.

THE MAIN FUNCTION.

There are five variables declared and defined first in this function: (1) *str1[]* – a character array containing the first word; (2) *str1Copy[]* – a copy of the first word to be used later to present the transformation; (3) *str2[]* – a character array containing the second word; (4) *transpose* – an integer vector containing the swap indexes (see *sort()* function) of the first word; and (5) *reverse_transpose* – an integer vector containing the swap indexes of the second word.

The program first displays the first two words and calls the *sort()* function of *str1[]* and *str2[]*. After the function call, the original words stored in those character arrays are not lost. This is where *str1Copy[]* comes into play.

In simulating the transformation, the program basically does the bubble sorting again with the swap indexes saved in the integer vectors. In the first loop, it starts with *str1Copy[]* and starts the swapping all over again, displaying the word after each swap so as to show it transforming a character at a time. After that loop, the letters in *str1Copy[]* should be the same as those of *str1[]* and *str2[]* after the *sort()* function call. In the second loop, it does the transformation of the second word in reverse. It just starts with the last member of *reverse_transpose* and does the swapping as usual (i.e. swap object at this index with the next object), again printing the word as it swaps each letter.

The main function is now done.