Project 1 Documentation

Purpose of the Program

==================

This is a program that is meant to read in a certain number of names from an input file, sort them by alphabetical order, and write them out to the terminal and a separate output file. Specifically, it reads in the first 10 names of the input file using the input file stream as character arrays. Each of these names, which each represent one character array, is then stored into a two-dimensional character array, after which the program will sort the individual names based on character differences. Essentially, the program acts as a sorting algorithm that helps alphabetize names in an efficient manner.

Program Design

=============

The program initializes a two-dimensional character array to store the names being read in, with each row capable of storing a name character array, or string, while the columns represent the positions of specific characters within each name. First, the program asks for the user to enter the names of their desired input and output file names through prompts. The program then inputs the names entered into character arrays representing the file names and opens the input file specified as well as the output file to which sorted names are being written into. Using input file stream, the program reads in the first ten names of the input file by placing each name as a one-dimensional character array into the two-dimensional array storing all names. The ten names are then printed out in the order in which they were read before sorting occurs.

Through a bubble sort algorithm, the program sorts the names in the 2D array by alphabetical order, taking advantage of the strcmp() and strcpy() functions. The strcpy() function takes in two character arrays as arguments and copies the elements of the source array into the second destination array. Given two character arrays, the strcmp() function compares the letters of the two character arrays, or strings, by going over each position. If strcmp() finds a difference between corresponding characters in the two strings, it returns a value related to their alphabetical order (positive if the first string comes after the second alphabetically and negative if the first string comes before the second one). If the function finds that the two strings are exactly the same, it will return a value of zero.

The bubblesort() function itself goes through the list of unsorted names n(n-1)/2 times, where n is the number of elements in the list, and swaps two names (rows of the 2D array) if they are out of proper alphabetical order (i.e. strcmp() returns positive value for the two names). It swaps two names by having strcpy() place the first name into a temporary character array, copy the second name into the first name’s character array, and copy the first name’s characters from the temporary character array into the second name’s former character array space. Finally, after sorting is complete, the program prints out the student names in sorted order both to the terminal and to the output file from earlier. Once these tasks are complete, the program closes the input and output files and terminates.

Problems & Issues

===============

During the course of the assignment, I ran into some problems regarding proper printing of unsorted and sorted names due to issues with my code for how names were being read in using the input file stream. I originally had the program read in each character of each name individually and move onto each name after the current name being read only after it reached the null character of each name string. However, this conditional loop proved to be unnecessary, and I ended up just having the program read in each name as its own row in the two-dimensional array for storing the names, letting the program move on to each successive name when it read in whitespace or the null character automatically. Otherwise, there were no major problems that were encountered when testing the source code for the program.

Changes to Program

================

I would have probably implemented a better sorting algorithm rather than bubble sort in order to make my program more efficient in terms of passage time and memory allocation when it comes to sorting lists. More specifically, I would have been interested in implementing a version of a merge sort or qsort formula for sorting that could have reduced the time for needed to alphabetize a list of student names on a much more significantly smaller timescale. I would have even considered implementing an insertion sort formula into the sorting algorithm; although the insertion sort algorithm is still slower and less efficient than the merge sort and qsort algorithms, it probably would have been better than bubblesort.