Abraham Meza

Christos Papachristos

CS 202 – 1001

January 27, 2019

Project 1 Report

**Description:**For this project you will write a program to: a) **read-in** the **10 first names** from a **file** (the file is a priori given to have exactly 10 entries, of a maximum length of 8 letters each) into a **2-dimensional character array**, b) **output** the names **to the terminal** with each one preceded by a number indicating its **original order** in the list, c) **sort** the list of names, and then d) **output the sorted** names both **to the terminal and to a file**, again with each one preceded by its corresponding **original order** in the list. Although an example input file (Names.txt) is provided, for grading purposes your project will be tested against a file that we will supply but will not be provided to you beforehand. Our test file will be in the same format as the example input file.

As an introduction to Computer Science II (202), our instructor assigned us a project that demonstrates our abilities to access file contents and sort the information using multidimensional arrays. For the project, students had to write a program that extracts name entries from a given text document and sorts the order into a new document. In particular, we were explicitly told to sort the list of names in alphabetical order and from least to greatest in respect to string sizes. As someone who is familiar with coding through C, I was confident in my abilities to demonstrate the task efficiently; however, restrictions were placed in order to showcase the students’ way of thinking. To be more specific, the project asked us to create multiple functions to fulfill the roles of certain libraries. Students had to create their own algorithms in order to sort the information accordingly. As students tackle on a new programming language, they are advised to think differently as the code will rely on different sections of code for functionality.

For my design, I began with the given instructions, and devised my code piece by piece. Firstly, I declared all the given information such as prototypes, header files, and defined variables. From there, I began to break apart the prototypes and worked on them individually. Since names and sentences are made up of character arrays, I simply created a while loop to scan through the array elements and count the characters until a NULL-character was found. By doing so, I would able to see the length of varied names, and it would allow me to utilize my length function in other prototypes. From there, I used the same concept of going through each individual element to construct my copy function. In addition to counting the elements to keep track of the NULL-character, I also applied an equating line of code to copy the character from one string to the other. Finally, I began on my comparison function, which compares the alphabetical placement of each string. Similar to the other functions, I created a loop statement that went through each array element to check if the character was before the other. Depending on the condition/result, the function would return a value indicating the position of the string relative to the other.

After the prototypes were finished and compiled correctly, I began to work on the main section of the program. Before thinking about the algorithm, I started off with envisioning the different variables within the scenario. In order to receive a file input/output, the program must take in a user input. Therefore, I declared and initialized multiple one dimensional arrays to scan in the user inputs; in addition, I also declared a two dimensional array that would carry the information from file to file, as well as the file declaration. From there, I prompted the program to ask the user for the required information and placed file statements to open and read the text document. From there, I printed the text file to the screen and executed a nested “for” loop to sort the file based on string size. However, I ran into an issue when calling my length function due to the fact that it only provided a singular integer value when I needed a comparison condition between the size of two strings. In order to fix the issue, I created another function to compare the sizes of the two strings to indicate which one was larger. From there I implemented an “if” statement to execute a swap and copy algorithm. Once the information was sorted accordingly, I printed the results onto the screen and into the user’s output file, through a “for” loop and file statement, then closed the output file. Since the concept of sorting is almost identical between all variables, I simply copied the same loop from before and implemented the alphabetical comparison to create a new sorting. Once the program sorted the information correctly, I put the alphabetical information into another file indicated by the user then closed the file. After I had finished, I tested and debugged the code until it displayed the correct results. Though my sorting algorithms worked as intended, the order came out differently from the example output (e.g. Example Output: 5 Romeo, 6 Oscar, 0 Victor, 2 Juliet, 3 Hector, 9 Dannae – My Output: 5 Romeo, 6 Oscar, 2 Juliet, 3 Hector, 0 Victor, 9 Dannae). Even so, the results matched the required conditions (order by string size) and were sorted accordingly.

Generally speaking, my design is very simple as it revolves around the concept of analyzing each array element and checking if a condition is either true or false. As a result of the detailed approach, there is very little opportunity for mistakes. Even so, I ran into a couple of mapping problems since I was exulting unnecessary effort in trying to make the program work through only the three protocols given.

If I were to have more time with the project, I would improve the program by modifying the loop statements to be more efficient instead of nested sections within each of them. In addition, I would also modify my length/comparison function to be able to incorporate comparisons alongside integer values when comparing string size.

All in all, I feel as if this project efficiently allowed students to remember previous coding experiences by putting their function knowledge to the test. The code heavily relies on flexibility as the functions tie hand in hand to sort the information accordingly. I enjoyed programming this project since it allowed me to see the accuracy of loop statements, and introduced me to fundamental statements in C++ coding.