John Carrabino
carrabij@oregonstate.edu
July 10th, 2016

Lab C
File I/O & Recursion

DISCOVERING REQUIREMENTS:

For this lab we were asked to write a program that reads in the contents of a file, uses a recursive function to reverse the characters read in, and writes the results to a separate input file. We need to prompt the user for the names of the input and output file, using exceptions, and prompting the user to alert them of any file errors.

For the contents of the input file we need to read it into a dynamic array that starts at size 50. If the array is full and you are not at the end of the file then you will create a new array that is twice the size of the old array, copying into it the contents of the old array. Once the file has been read in, we need to create a recursive function that will reverse the contents of the array. For this lab we can use a single array and pass it by reference to the function and write it to the output file.

After reviewing the guidelines for this lab, I was able to look back on past labs and assignments for reference. In particular I thoroughly reviewed my grocery list assignment and its use of dynamic memory allocation for arrays, as well as some file adding programs I made for CS 161.

DESIGN:

main.cpp

- do-while loop with a switch statement to loop displayMenu();
- Ask user if they want to reverse a file or exit the program
- input validation, open input file and use conditional statement to see if it exists
- prompt user for valid file name in directory
- asks user to name an output file to write reverse file to
- display reversed file to user
- end of loop, show displayMenu() again

Reverse.h

- Reverse Class
    - Public:
        - Reverse() constructor that initializes the character array
        - ~Reverse() destructor that frees the memory in the heap
        - getSize() returns size of the array
        - getEnd() returns ending index of the array

    - expandArray() doubles the size of the array
    - reverseArray() reverses the contents of the array
    - printArray() prints reversed array to screen
    - toOutput() writes contents of the array to the output file
    - addChar() adds characters to the array

- Private:
    - array size
    - array end index
    - *arrayItem; a pointer to chars in the char array

TESTING:

After I got my program together and working I wanted to make sure that my program could handle most types of file contents that I could think of. This meant testing for white space, special characters, and making sure that all memory was cleared from the heap upon exit. I also used my getEnd() and getSize() functions to help keep track of the contents of my array and to test the edge cases to see how it would act after dynamically expanding. Lastly, I also added an itExists() function to my main file in order to test if the file names for the input file were valid.

Test Case	Input	Test Function	Expected Outcomes	Observed Outcomes
Input a file that does not exist.	fileName = “help!.fml”	switch(choice){ case 1:prompt user to enter the file to be reversed	Loop back to question prompting user for valid input	Loop back to question prompting user for valid input
Input an invalid file name	fileName = “ ”	switch(choice){ case 1:prompt user to enter the file to be reversed	Loop back to question prompting user for valid input	Loop back to question prompting user for valid input
Using files that began and ended with white space	Ex) “ t-rex arms “	reverseArray()	“ smra xer-t “	“ smra xer-t “
Using files that did not begin or end with white space	Ex) “wow lol omg”	reverseArray()	“gmo lol wow”	“gmo lol wow”
Reversing the array previously reversed.	Already reversed: “STAC”	reverseArray() … reverseArray()	Reversed a second time: “CATS”	Reversed a second time: “CATS”

It took a lot of trial and error to get things working properly in this program. Most of my issues actually came from validating user input and making sure I had cleared the keyboard buffer so the user input would not be adulterated by buffer overrun. By the end once I had all of my test cases working as needed, I was able to paste entire word documents and have them be reversed through my program.

REFLECTIONS:

Recursion was one of the more interesting topics we covered last semester in Discrete Math. Soon after we began the topic I was able to make a near immediate connection between recursion and programming. In discrete we used recursive functions can help us generate large sequences of numbers, but the fact that this assignment had us using it to reverse the contents of an array was a great twist on a not-so-old topic! Overall I really enjoyed this lab, and unlike the doodlebugs assignment, I actually had fun drafting the recursive function out and trying to implement it on my own.