Conclusiones del proyecto: Diego Orozco

En este proyecto, se implementaron las funciones que nosotros creemos que eran las más importantes para el correcto funcionamiento del programa. Si bien pudimos haber hecho ciertos pasos de *main,* tener todo bien organizado, función por función, nos pareció más adecuado. Considerando también que solamente dos funciones eran requeridas según las instrucciones que recibimos de parte del profesor.

Primero, se definió la estructura de nodo, recibiendo un apuntador a una palabra, y la estructura en sí con otro apuntador a lo que sería la siguiente palabra de la frase a verificar.

La primera función devuelve un apuntador al archivo con el que se va a trabajar. Se inicializa un apuntador a archivo como nulo, y después se abre. En caso de que siga siendo nulo, hacerle saber al usuario y retornar el archivo. La función “ListFillIn” devuelve un apuntador a un nodo de la lista. Primero se inicializa como nulo, y se reserva memoria para cada palabra de la frase a revisar. Con la función fgets(), se obtiene la primera línea del archivo. Si es nula, se retorna lo mismo. Finalmente, la variable head se iguala a la función de WordSep(), y retorna la cabeza de la lista. La función de WordSep(), también devuelve un apuntador a la cabeza de la lista, y se encarga básicamente de saber cuando distinguir entre las palabras, buscando el carácter terminador ‘\0’ para saber que la palabra ya terminó. Se implementó con un ciclo for para iterar sobre la cadena de caracteres, y un ciclo do while, para que busque las palabras mientras sea verdadero. La cabeza es igualada a ListAddWord(). Después, la función de ListAddWord() agrega palabra por palabra a la lista, agregando los nodos con la función NewNode(). Se usa la técnica de apuntador a current, para que la cabeza no se vea modificada, y mientras current en next sea diferente de nulo, va a agregando los nodos. Para la función de NewNode(), simplemente se reserva memoria para el mismo, y el apuntador que declaramos de new en la palabra se iguala a word_to_add, que es el argumento de la función, y en next se iguala a nulo. Se retorna el apuntador de nuevo. PrintCadenita() solamente va imprimiendo los nodos que encuentre en el archivo. WriteResult() escribe el resultado de las funciones en un nuevo archivo llamado

result.txt. Se usa la función de stract() que añade un bloque de memoria a otro, los va uniendo. Estas son las funciones auxiliares de nuestro programa.

La primera función importante es la de WordReverse(), que analiza todas las palabras que encuentre en el archivo, y las voltea de orden. Es decir:

- Input: hola esto es un string
- Output: string un es esto hola

Primero, se verifica que la cabeza no sea nula para poder proceder con la función. Con el nodo, se declara un apuntador a next, uno a before y uno para una nueva cabeza. Mientras la cabeza sea diferente de null, next se iguala a head en next, head en next se iguala a before, before se iguala a head y head a next. Fuera del ciclo, la nueva cabeza se iguala a before, y se retorna la nueva cabeza.

La segunda función importante en nuestro programa es la de ReverseSomeWords(), que analiza todas las palabras que encuentre en el archivo, y si su longitud es mayor o igual a 5, entonces las voltea. Es decir:

- Input: hola esto es un string
- Output: hola esto es un gnirts

Se crea un apuntador a nodo current, y se declara un apuntador a una nueva palabra de char. Haciendo uso de la librería <string.h>, usamos dos funciones, strlen() y strrev(). La primera puede checar la longitud del string, y la segunda escribe de reversa el string seleccionado. Mientras current sea diferente de null, para iterar sobre las cadenas que encuentre. Si la longitud de current en word (usando strlen()) es mayor o igual a 5, entonces la nueva palabra es igualada a strrev de current en word, y después se iguala current en word a la nueva palabra. Ya fuera de la condición, pero aun sobre el ciclo while, current se iguala a current en next para que pase al siguiente nodo. Se retorna la cabeza.

Finalmente, la última función, abre el archivo de input para leerlo y abre también el de result para escribir. Se abre un do while para escoger la función a utilizar, se

escribe el resultado y también se imprimen las palabras en la consola. Para cerrar, se libera la memoria utilizada en el archivo y en el resultado de archivo.

Me parece que este proyecto integrador fue muy bueno para repasar los temas vistos hasta ahora en un ambiente más aterrizado a la realidad, con un caso de prueba que pueda ser útil para un usuario, en vez de ejemplos genéricos como los que se ven en la clase. Tuvo sus dificultades, pero creo que hicimos un buen trabajo.