Definiciones y especificación de requerimientos

a) Definición general del proyecto de software

Cuentapalabras es un programa que se encarga de leer todas las palabras dentro de uno o más archivos de texto en un directorio dado por el usuario y las ordena e imprime en 2 archivos .out, **cadauno.out** ordena las palabras de cada archivo de texto por separado y **totales.out** ordena todas las palabras juntos, en ambos casos las palabras son ordenadas por cantidad de apariciones y orden alfabético. El objetivo que intentamos cumplir con este programa es el de facilitar el análisis y ordenamiento de grandes cantidades de palabras. Este programa será útil para gente que necesite analizar grandes cantidades de palabras y necesite ordenarlas, pero requiere cierto conocimiento de cómo utilizar la consola de comandos y MinGW para ejecutar el programa y de GitHub para poder descargarlo.

b) Especificación de requerimientos del proyecto

El programa implementado (cuentapalabras) está conformado con la siguiente especificación al ser invocado desde la línea de comandos:

$ cuentapalabras [-h] [directorio de entrada]

Los parámetros entre corchetes denotan parámetros opcionales. El significado de las diversas opciones es el siguiente: En caso de no especificar parámetros o de especificar el parámetro -h como primer argumento en la línea de comandos, el programa debe mostrar una pequeña ayuda por pantalla, la cual consiste en un breve resumen del propósito del programa junto con una reseña de las diversas opciones disponibles. Si se especifica un directorio el programa debe procesar los archivos .txt que se encuentren en dicho directorio.

Para que el programa funcione, se requiere que el usuario le dé al programa un directorio valido, que separe los nombres de carpetas con "****" (ej: **C:\\Users**) y tener un compilador de C, como **MinGW**, instalado. Solo debe haber una sola palabra por renglón y esta debe de ser de menos de 40 caracteres. Además, el usuario no debe usar palabras con caracteres especiales (¡, ", $, etc.).

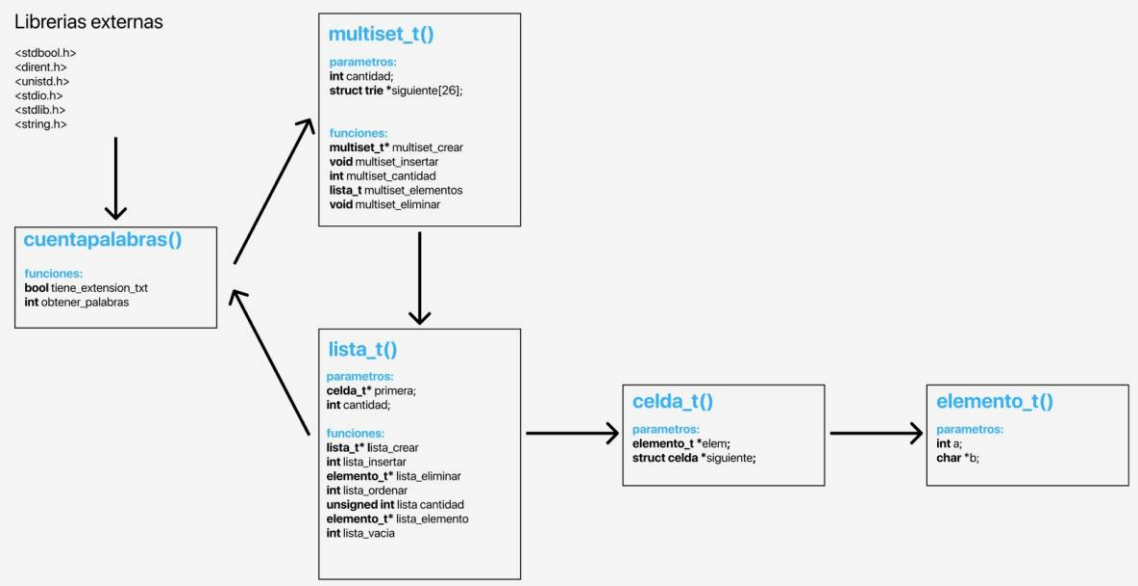
c) Procedimiento de instalación y prueba

Ingrese al repositorio de GitHub **https://github.com/valenpistonesi/Proyecto_Orga** y descargue los archivos necesarios desde ahí. Luego se puede ejecutar simplemente ejecutando **cuentapalabras.exe**

Descripción jerárquica:

El programa está organizado utilizando espacio de nombres para dividir el código en grupos lógicos y para evitar conflictos de nombres que puedan producirse.

Diagrama de módulos:



Descripción individual de los módulos:

cuentapalabras(): en el main se leen los archivos de texto y se los pasa al multiset, luego recibe una lista_t con las palabras y las escribe en los .txts. No se encarga del ordenamiento de las palabras ni de la creación de la lista_t. Requiere una dirección de archivos valida con archivos .txts para poder inicializarse y recibir una lista_t() válida para poder imprimir las palabras en los .txts. Se encuentra en **main.c.**

multiset_t(): tiene como parámetros un int que indica la cantidad y un pointer a un multiset_t que conecta con la siguiente letra. En el multiset_t se reciben las palabras de main y se encarga de crear los multisets necesarios para guardar las palabras y crear la lista_t. Requiere de palabras de validas sin signos especiales y de menos de 40 caracteres poder inicializarse. No se encarga del ordenamiento de las palabras. Se encuentra en **multiset.c y multiset.h.**

lista_t(): tiene como parámetros un int que indica cantidad de palabras distintas en la lista_t y un pointer a una celda_t que hace referencia a la primera palabra de la lista. En la lista_t se reciben palabras del multiset_t y la lista la agrega a la lista (o aumenta la cantidad de repeticiones si la palabra es repetida) y luego las ordena con lista_ordernar. La lista al ser simplemente enlazada solo se puede acceder al siguiente y no se puede volver atrás.

celda_t(): tiene como parámetros un elemento_t que contiene la palabra y sus apariciones y un pointer a la siguiente celda_t. Este componente solo almacena estos 2 datos y no hace nada con ellos, ya que no tiene funciones propias.

celda_t(): tiene como parámetros un int que indica la cantidad de apariciones de la palabra y un pointer a un char que contiene la palabra. Este componente solo almacena estos 2 datos y no hace nada con ellos, ya que no tiene funciones propias.

Funciones cuentapalabras

bool tiene_extension_txt (char const *nombre)
Revisa si el archivo nombre es un documento de texto o no

void llenar_totales (multiset_t *m, char *filename)
Llena un multiset m con las palabras del archivo de nombre filename

void imprimir_lista(lista_t *l, FILE *filename)
Imprime la lista l en el archivo filename

Funciones multiset_t

Multiset_t *multiset_crear()
Crea un multiset vacío de palabras y lo devuelve.

void multiset_insertar(multiset_t *m, char *s)
Inserta la palabra s al multiset m.

int multiset_cantidad(multiset_t *m, char *s)
Devuelve la cantidad de repeticiones de la palabra s en el multiset m.

lista t multiset_elementos(multiset t *m, int (*f)(elemento_t,elemento_t))
Devuelve una lista de tipo lista t con todos los elementos del multiset m y la cantidad de apariciones de cada uno.

void multiset_eliminar(multiset_t **m)
Elimina el multiset m liberando el espacio de memoria reservado. Luego de la invocación m debe valer NULL.

Funciones lista_t

Lista_t *lista_crear()
Crea una lista vacía y la devuelve.

int lista_insertar(lista_t *l, elemento_t elem, unsigned int pos)
Inserta el elemento elem en la posición pos de la lista.

Elemento_t *lista_eliminar(lista_t *l, unsigned int pos)
Elimina el elemento de la posición pos de la lista.

Elemento_t *lista_elemento(lista_t *l, unsigned int pos)
Devuelve un puntero al elemento que ocupa la posición pos de la lista.

int lista_ordenar(lista_t *l, funcion comparación_t comparar)
Dada la lista l y la función comparar ordena la lista de acuerdo al criterio de dicha función.

unsigned int lista_cantidad(lista_t *l)
Devuelve la cantidad de elementos de la lista l.

int lista_vacia(lista_t lista)
Devuelve verdadero (= 0) si la lista está vacía, y falso (= 0) si la lista contiene al menos un elemento.

Diseño del modelo de datos