ORGANIZACIÓN DE COMPUTADORAS
PROYECTO N°1
Programación en Lenguaje C

Integrantes:

Enrique, Cristian Facundo

Sepulveda, Lucia Belen

Arroyo,María Sol

TDAs IMPLEMENTADOS: prototipos de funciones y sus comportamientos.

TDA LISTA

1. void crear_lista(tLista * l): Inicializa una lista vacía. En *L se referencia la lista creada. LST_ERROR_MEMORIA si no es posible reservar memoria correspondientemente.

2. void l_insertar(tLista l, tPosicion p, tElemento e): inserta un elemento en la posición anterior a una dada. Recibe como argumentos a la lista que se desea modificar, la posición próxima siguiente a la cual se va a realizar la inserción, y el nuevo elemento. Se busca el nodo recibido como parámetro, se crea un nuevo nodo con el elemento recibido, y se inserta inmediatamente antes a pos. Finalmente, se actualiza la estructura.

3. void l_eliminar(tLista l, tPosicion p, void (*fEliminar)(tElemento)): Elimina el elemento de una posición dada de la lista. Se reciben como argumentos la lista y la posición que se desea remover. Se busca en la lista el nodo equivalente al recibido, se elimina mediante la función fEliminar y se libera el espacio de memoria de dicho nodo. Finalmente se actualiza la estructura.

4. void l_destruir(tLista * l, void (*fEliminar)(tElemento)): Destruye la lista L, elimininando cada una de sus celdas. Los elementos almacenados en las celdas son eliminados mediante la función fEliminar.

5. tElemento l_recuperar(tLista l, tPosicion p): Recupera y retorna el elemento en la posición P. Finaliza indicando LST_POSICION_INVALIDA si P es fin(L).

6. tPosicion l_primera(tLista l): Retorna la primera posición de la lista. Se recibe como parámetro la lista lista. Se devuelve el primer elemento. Si la lista está vacía, entonces dicho puntero será nulo (POS_NULA).

7. tPosicion l_siguiente(tLista l, tPosicion p): Retorna la posición siguiente de p. Éste método recibe una lista y la posición de la cual se desea conocer el siguiente nodo. Devuelve un puntero nulo si el nodo recibido no tiene un elemento siguiente. En caso contrario, el método retorna la posición correspondiente.

8. tPosicion l_anterior(tLista l, tPosicion p): Retorna la posición anterior de p en L.Éste método recibe una lista y la posición de la cual se desea conocer el anterior nodo. Devuelve puntero nulo si el nodo recibido no tiene elemento anterior. En caso contrario, el método retorna la posición correspondiente.

9. tPosicion l_ultima(tLista l): Retorna la últimaposición de la lista. Se recibe como parámetro la lista l. Se devuelve el último elemento. Si la lista está vacía, entonces dicho puntero será nulo (POS_NULA).

10. tPosicion l_fin(tLista l): Recupera y retorna la posición fin de L. Si la lista L es vacía entonces , primera(L) = ultima(L) = fin(L).

11. int l_longitud(tLista l): Retorna la longitud actual de la lista.

TDA Mapeo

1. crear_mapeo(tMapeo m, int ci,int (fHash)(void),int (Comparacion)(void,void)): Inicializa un mapeo vacío con referencia al mapeo creado, con capacidad inicial igual a ci,el valor hash de las claves que se comparan mediante la función de comparación

2. tValor m_insertar (tMapeo m, tClave c, tValor v): Inserta en el Mapeo M una entrada con clave c y valor v. Si ya existe una entrada con clave c y valor vi se modifica vi por v. Retornar NULL si la clave c no existe en M o vi en caso contrario.

3. void m_eliminar(tMapeo m, tClave c, void (*fEliminarC)(void *), void (*fEliminarV)(void *)): Si la entrada con clave c ya existe en M, se elimina la entrada. La clave y el valor de la entrada son eliminados mediante las funciones fEliminarC y fEliminarV.

4. void m_destruir(tMapeo * m, void (*fEliminarC)(void *), void (*fEliminarV)(void *)): Se destruye el mapeo M, elimininando cada una de sus entradas. Las claves y valores almacenados en las entradas son eliminados mediante las funciones fEliminarC y fEliminarV.

5. tValor m_recuperar(tMapeo m, tClave c): Recupera el valor correspondiente a la entrada con clave C en M, si esta existe y retorna el valor correspondiente, o NULL en caso contrario.

DECISIONES DE DISEÑO

TDA Lista

Se siguen las pautas de diseño establecidas en el enunciado del proyecto: cada uno de los elementos de la lista son punteros tipo genéricos y la lista es simplemente enlazada con nodo centinela. Las celdas mantienen un elemento y un enlace a la posición anterior. Una lista se considera vacía si el valor que indica el tamaño es un cero.

TDA Mapeo

Siguiendo las pautas de diseño establecidas en el enunciado del proyecto: cada una de las claves y valores son punteros genéricos. El mapeo es implementado mediante una tabla hash abierta usando el TDA Lista.

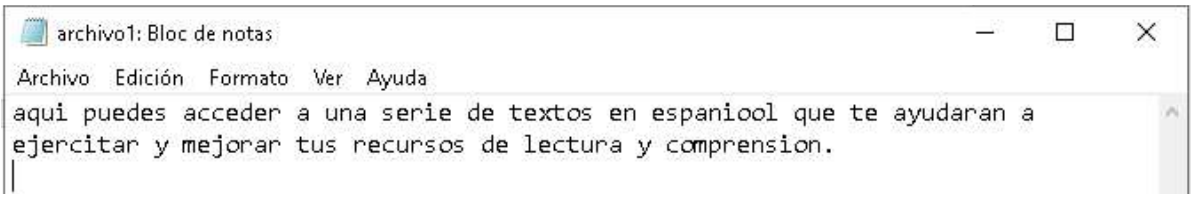
Programa Principal:

Para el programa principal se usó un mapeo, donde se insertan las palabras usando un codigo hash .Se hace uso de la liberación de memoria que se reserva . Se hace uso de las estructuras antes mencionadas de forma indirecta.

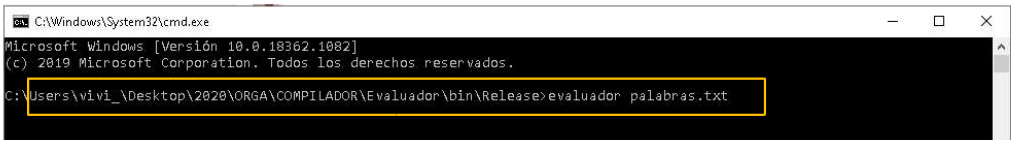
La lectura del archivo de realiza por renglones, luego estos se dividen en palabras y se insertan en el mapeo antes mencionado.

Ejemplo de aplicación del Programa

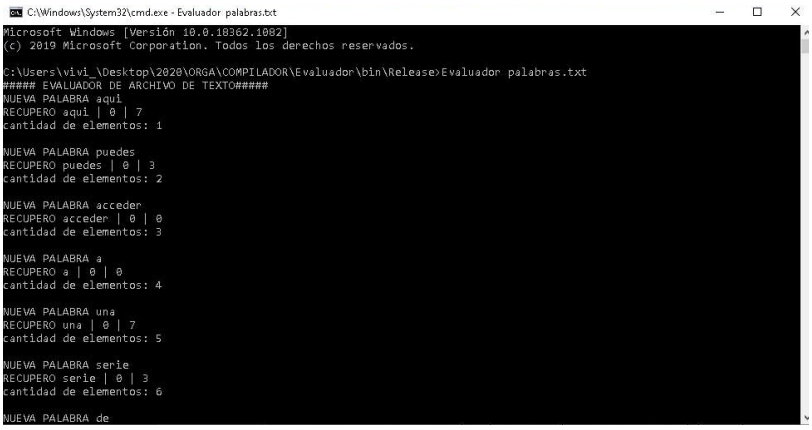
Supongamos que tenemos un archivo de texto "palabras.txt" que contiene:



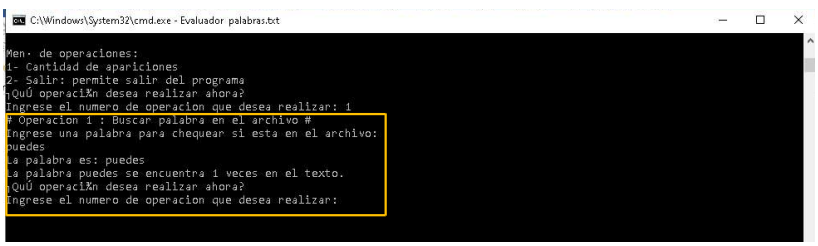
Para ejecutar el programa hacemos:



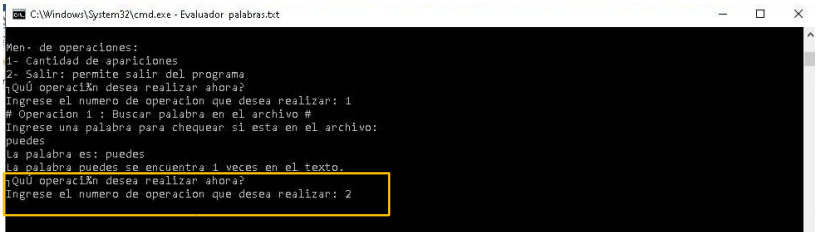
Una vez ejecutado el programa se cargan las palabras en un mapeo y se muestra el siguiente menú de operación:



Para saber la cantidad de veces que aparece una palabra en el archivo(opción 1)



Para salir del programa (opcion 2):



Si el archivo se encuentra vacío no se cargarán elementos por lo tanto al elegir la primer operación siempre retornara que la palabra no se encuentra en el archivo:

En el caso de que se ingresaran en medio signos tales como “ ,” el programa no los toma y solo cuenta las palabras , en el caso de ser repetidas solo cuenta una a la hora de mostrar la cantidad de palabras