Algoritmos y Estructuras de Datos

Trabajo Práctico: Diccionario de Datos

Los diccionarios de datos, también conocidos como tablas de hash, hashtables o hashmaps, son una estructura de datos que asocian claves con un valores para realizar búsquedas de una manera eficiente. Funcionan transformando la clave con una función de hash y obteniendo la posición en la que el valor está guardado.
Las funciones de hash no son a prueba de colisiones, es decir, es posible que haya 2 claves distintas que tengan el mismo hash. En ese caso los 2 o más elementos deben compartir la misma posición en la tabla.
Lo atrayente de estas estructuras de datos es que alcanzan un tiempo constante de búsqueda promedio O(1) sin importar el número de elementos en la tabla, aunque en casos particularmente malos esta complejidad puede llegar a O(n).

Parte 1

Implementar el TDA t_diccionario genérico que contiene claves de cadenas de caracteres de tamaño variable y valores de cualquier tipo y tamaño.
El vector principal del diccionario es de tamaño fijo, pero se define al momento de la creación del diccionario.
Para manejar las colisiones debe utilizar listas simplemente enlazadas, cuya implementación debe estar separada de la del diccionario y debe utilizar las primitivas vistas en clase.

Debe implementar las siguientes primitivas para el TDA diccionario:

- crear_dic: debe recibir por parámetro una capacidad máxima y definir los valores necesarios para inicializar el diccionario.
- poner_dic: agrega un nuevo elemento al diccionario. Recibe la clave y el valor a agregar al diccionario, calcula la posición correcta para insertarlo y maneja las colisiones. En el caso de que la clave a insertar ya exista, debe reemplazar el antiguo valor por el nuevo.
- obtener_dic: busca un elemento en el diccionario por clave y devuelve el valor.
- sacar_dic: Elimina un elemento del diccionario buscando por clave.
- recorrer_dic: Recorre el diccionario de datos ejecutando una acción para cada elemento presente.
- destruir_dic: Vacía el diccionario y libera toda la memoria.

Se espera que la organización del proyecto esté bien estructurada con los correspondientes archivos de headers (.h) y de código (.c).

Parte 2

Crear un procesador de textos que a partir de un archivo de texto que ingresa, calcule la cantidad de palabras, espacios y signos de puntuación que contiene y la cantidad de apariciones de una misma palabra.

Construya una interfaz de usuario donde primero el usuario ingrese el archivo de texto a procesar. Luego muestre un menú donde se pueda seleccionar mostrar las estadísticas generales (cantidad de palabras y signos), el listado de apariciones por palabra o la cantidad de apariciones de una palabra en particular que ingrese un usuario.

Debe valerse del TDA Diccionario desarrollado en la primera parte.

Debe proporcionar al menos 4 lotes de pruebas, donde se pueda notar cómo funciona el diccionario y donde se vean colisiones en los hashes.

Entrega y defensa

Este trabajo práctico se realiza en equipos de hasta 4 personas, sin excepción. Los equipos deben ser informados por la plataforma MIeL en un mensaje en el foro “Trabajo Práctico” con plazo máximo hasta el día **13/10/2025 a las 23:59.**
La entrega del trabajo práctico se realiza de manera individual. Cada integrante del grupo debe enviar desde la sección prácticas de MIeL en la práctica “Trabajo Práctico” a todos los docentes, una copia del programa informando nuevamente en el mensaje los integrantes de su grupo.
El plazo **máximo** de entrega del trabajo práctico será el día **03/11/2025 a las 13:59** a través de la plataforma MIeL y habrá una defensa del mismo en la clase de los días 03/11/2025 y 17/11/2025, contando con una evaluación grupal e individual.
Los trabajos entregados después de dicha fecha, no serán tenidos en cuenta, por lo que dicho equipo tendrá un ausente en la entrega y en la defensa, teniendo que utilizar la fecha de recuperatorio para regularizar su situación.