# 8. Trabajo Práctico 8 - Archivos de Texto

## 8.1. Ejercicio 1

Escribir un programa llamado mi_cat que tenga una funcionalidad similar a la del programa cat de Linux. Es decir, debe recibir el nombre de un archivo de texto como argumento por línea de comandos (argumento del main), y debe imprimir a continuación su contenido en la consola.

## 8.2. Ejercicio 2

Escriba un programa que lea el archivo de texto "diodo.txt" y determine:

- Cantidad total de palabras.
- Cantidad de veces que aparece la palabra "diodo". Para ello se debe escribir una función que permita buscar cualquier palabra e indique cuántas veces la encontró. **Nota: La función no debe distinguir entre mayúsculas y minúsculas.**

## 8.3. Ejercicio 3

Se desea disponer de una función que permita depurar o "debuggear" un programa, volcando la información sobre un archivo de texto histórico (log file), de acuerdo al siguiente prototipo:

**void log_msg(NivelLog nivel, const char* archivo, int linea, const char* mensaje)**

Donde:

- nivel: Tipo enumerativo que representa el nivel de debug al que pertenece el mensaje (DEBUG, INFO, WARN, ERROR, FATAL).
- archivo: Archivo donde se genera el log (siempre vale __FILE__).
- linea: Linea donde se genera el log (siempre vale __LINE__).
- mensaje: Mensaje de log.

**Ayuda**

La línea de debug debe contener el timestamp (en formato Unix Epoch Timestamp), el nivel de debug, el nombre del archivo, el número de línea y el mensaje. Se debe abrir y cerrar el archivo de log en la función. Por ejemplo, si se invoca la función de la siguiente manera:
**log_msg(INFO, __FILE__, __LINE__, "Registrando usuario...");**
Se debería obtener la siguiente línea de debug:
**"1565474700 INFO main.c:35: Registrando usuario..."**

## 8.4. Ejercicio 4

Para simplificar el desarrollo de una aplicación de login de usuarios, nos piden implementar las siguientes funciones (deben replicar la misma funcionalidad que en libinfo.a):

- **int leer_linea(FILE* fp, char* linea);** - Lee una línea del archivo (remueve el caracter \n y coloca \0). Retorna EXITO o ERROR en función del resultado obtenido.
- **int escribir_linea(FILE* fp, const char* linea);** - Escribe una línea en el archivo al final del mismo (remueve el caracter \0 y coloca \n). Retorna EXITO o ERROR según el resultado.