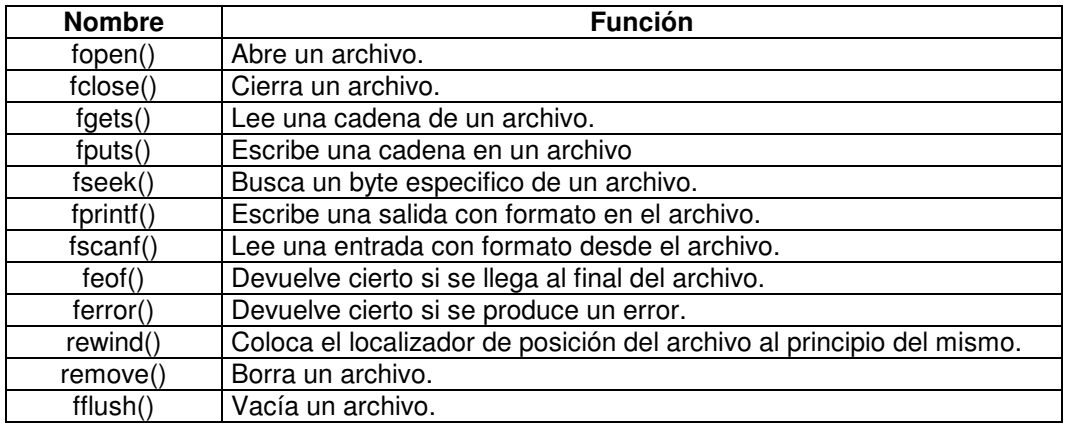
1. Archivos

Los archivos son colecciones de datos estructurados que se almacenan en la memoria secundaria de una computadora.

Hay dos tipos de archivos:

* + Archivos de texto: secuencia de caracteres organizadas y que terminan en carácter de fin de archivo (EOF). Es un archivo de texto plano que podrá ser leído y modificado por cualquier programa editor de textos.
  + Archivos binarios: secuencia de bytes cuya organización estará determinada por el programa que lo crea.
* ***Funciones de archivos de la biblioteca “stdio.h”***



* ***Puntero a archivo: FILE \*f***

Es un puntero a un tipo de estructura FILE que contiene información acerca de un archivo y permite realizar operaciones de lectura y escritura sobre archivos.

* ***Apertura de archivos: FILE \* open(const char \*Nombre\_archivo, const char \*modo)***

***Nombre:***

***a.txt***

***./a.txt***

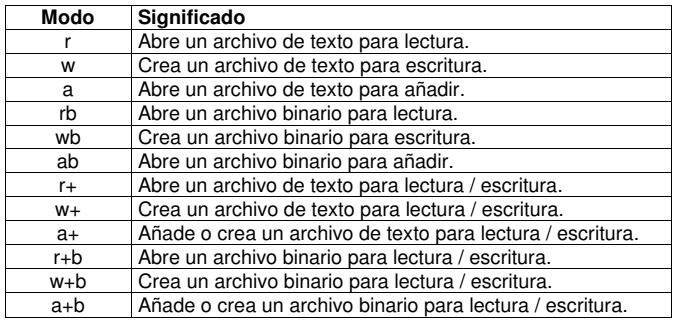
***/home/valentin/a.txt***

***C:\datos\a.txt***

Operaciones de archivo:

* Lectura (read, r)
* Escritura (write, w)
* Agregar información al final (append, a)

Modos de apertura:



La funcion fopen devuelve un puntero a archivo si la operación fue exitosa, NULL en caso contrario.

* ***Cierre de un archivo: int fclose(FILE \*f)***

Escribe toda la información pendiente en el archivo y realiza el cierre formal del mismo ante el sistema operativo.

* **Lectura y escritura de archivos de texto:**

En un archivo de texto se almacenan caracteres (o secuencias). Por ende se utilizan funciones similares a las utilizadas para manejar texto en consola:

* + Fprintf() y fscanf(): para cadenas de texto con formato
  + Fputs() y fgets(): para cadenas de texto terminadas en fin de cadena
* **Lectura y escritura de archivos binarios:**

En los archivos binarios se puede almacenar cualquier tipo de variable de C. Las mismas se van a leer y escribir en bloques según el tamaño de la variable.

size\_t fwrite(void \*puntero, size\_t tamano, size\_t cantidad, FILE \*archivo);

size\_t fread ( void \* ptr, size\_t size, size\_t count, FILE \* stream );

Fread:

Esta función lee un bloque de una "stream" de datos. Efectúa la lectura de un arreglo de elementos "count", cada uno de los cuales tiene un tamaño definido por "size". Luego los guarda en el bloque de memoria especificado por "ptr". El indicador de posición de la cadena de caracteres avanza hasta leer la totalidad de bytes. Si esto es exitoso la cantidad de bytes leídos es (size\*count).

Fwrite:

Esta función está pensada para trabajar con registros de longitud constante y forma pareja con fread. Es capaz de escribir hacia un fichero uno o varios registros de la misma longitud almacenados a partir de una dirección de memoria determinada. El valor de retorno es el número de registros escritos, no el número de bytes. Los parámetros son: un puntero a la zona de memoria de donde se obtendrán los datos a escribir, el tamaño de cada registro, el número de registros a escribir y un puntero a la estructura FILE del fichero al que se hará la escritura.

* ***Estructura y exploración de un archivo:***

En los archivos se almacena la información de manera secuencial (un dato a continuación del otro) y pueden tener cualquier tamaño siempre y cuando haya memoria disponible. En la estructura FILE existe un apuntador o cursor que indica a las funciones de lectura y escritura de archivos la próxima posición a leer o escribir dentro del archivo. Cada operación de lectura y/o escritura reposiciona ese cursor justo en la posición siguiente a los datos ya procesados. Es posible mover ese cursor para acceder a una posición deseada del archivo. Al final del archivo, justo a continuación del último dato almacenado, se agrega un carácter de fin de archivo EOF.

Funciones para recorrer archivos:

***Fin de archivo: int feof(FILE \*F):*** *devuelve 1 si el cursor se encuentra en el final del archivo. 0 en caso contrario*

***Posicionarme dentro del archivo: void fseek(FILE \* f, int posición, int modo)***

Esta función sitúa el cursor del archivo *apuntado por* ***f*** en una posición indicada por ***posición*** con respecto al ***modo***.

Modos:

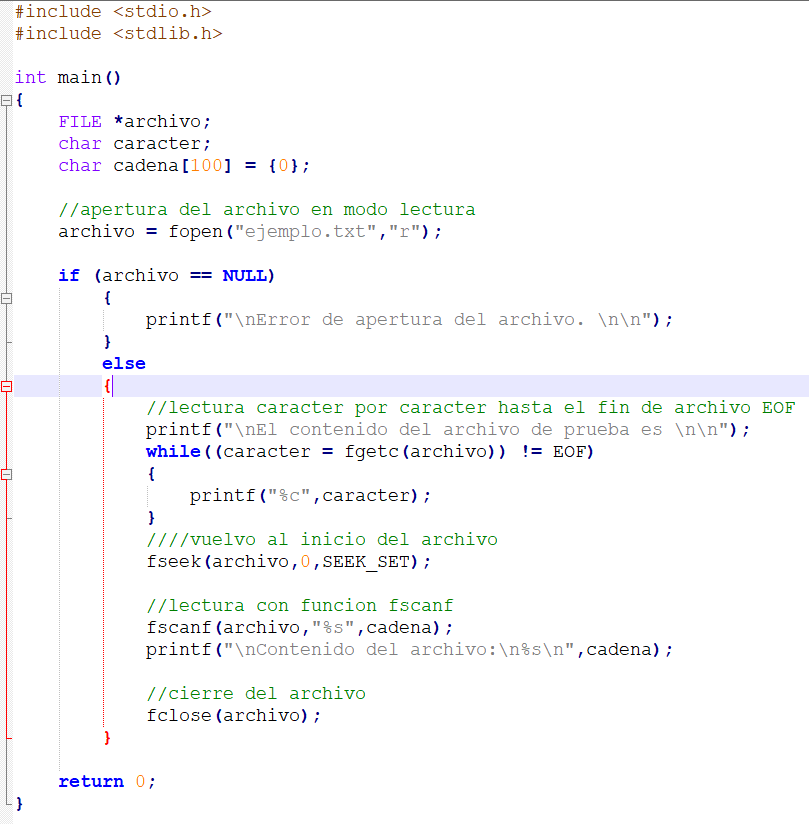
**SEEK\_SET** // Posición respecto al inicio del archivo (0)

**SEEK\_CUR** // Incremento relativo a la posición actual (1)

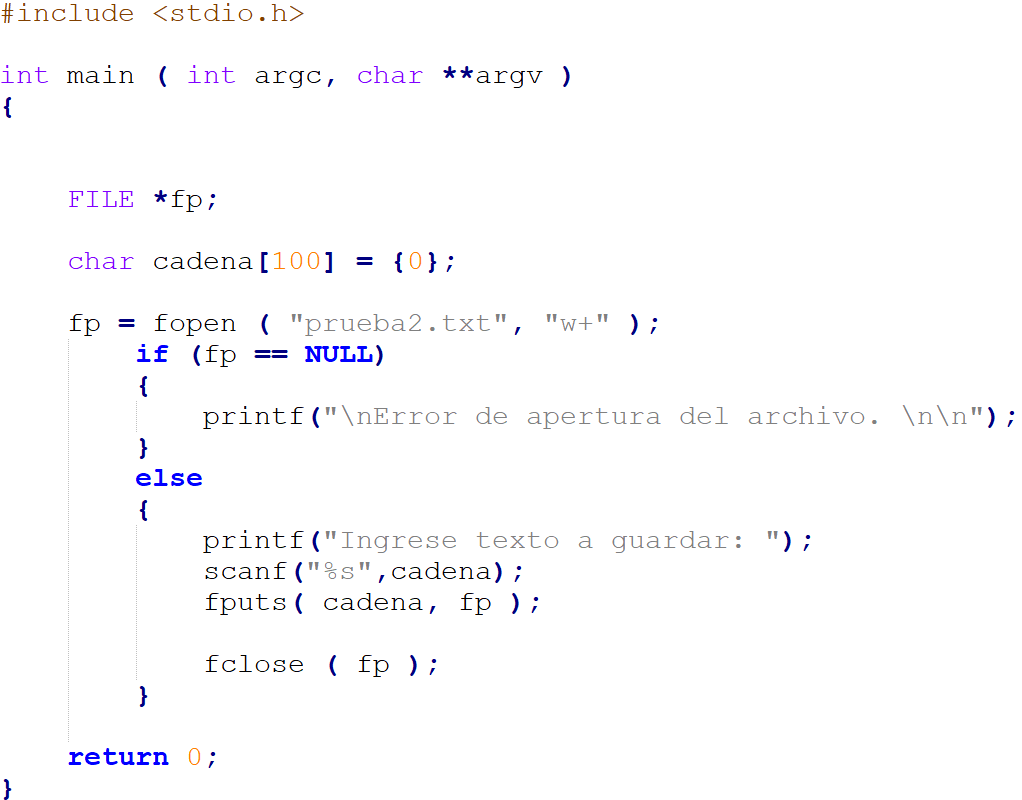
**SEEK\_END** // Posición respecto al final del archivo (2)

Ejercicios:

1. Abrir el archivo de texto ejemplo.txt en modo lectura e imprimir su contenido en consola:

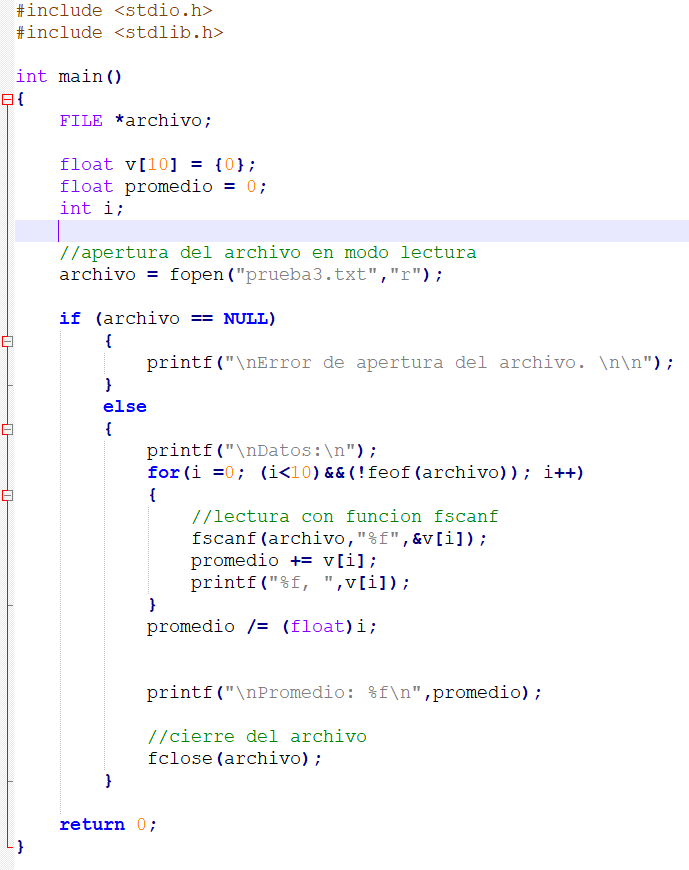


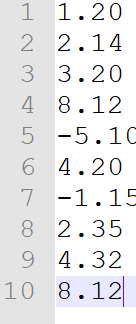
1. Crear el archivo de texto prueba2.txt y guardar en el mismo un texto introducido por consola:



1. Abrir el archivo prueba3.txt que contiene 10 números con decimales, uno en cada renglón, calcular el promedio y mostrar todo en pantalla:

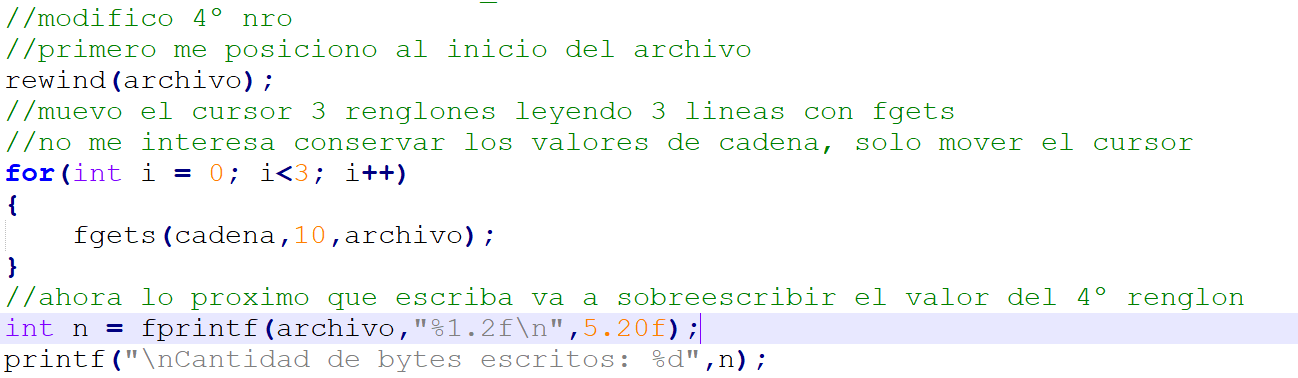
Archivo prueba3.txt: Programa:



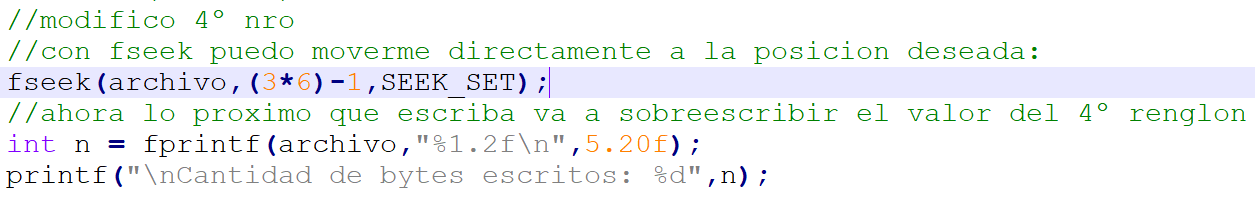


1. En el archivo anterior prueba3.txt modifique el 4º numero:

Hay varias formas de realizar esto, por ejemplo:

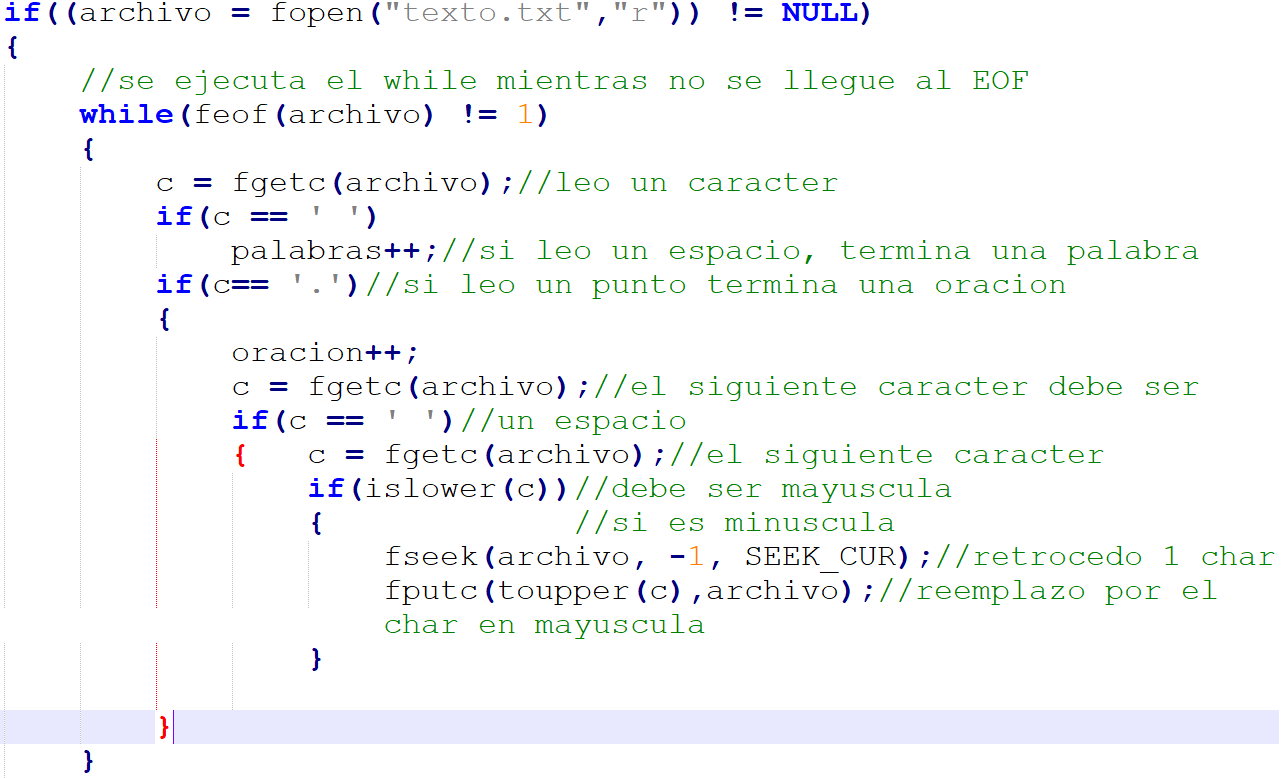


Otra opción si conozco cuantos caracteres hay en cada línea:



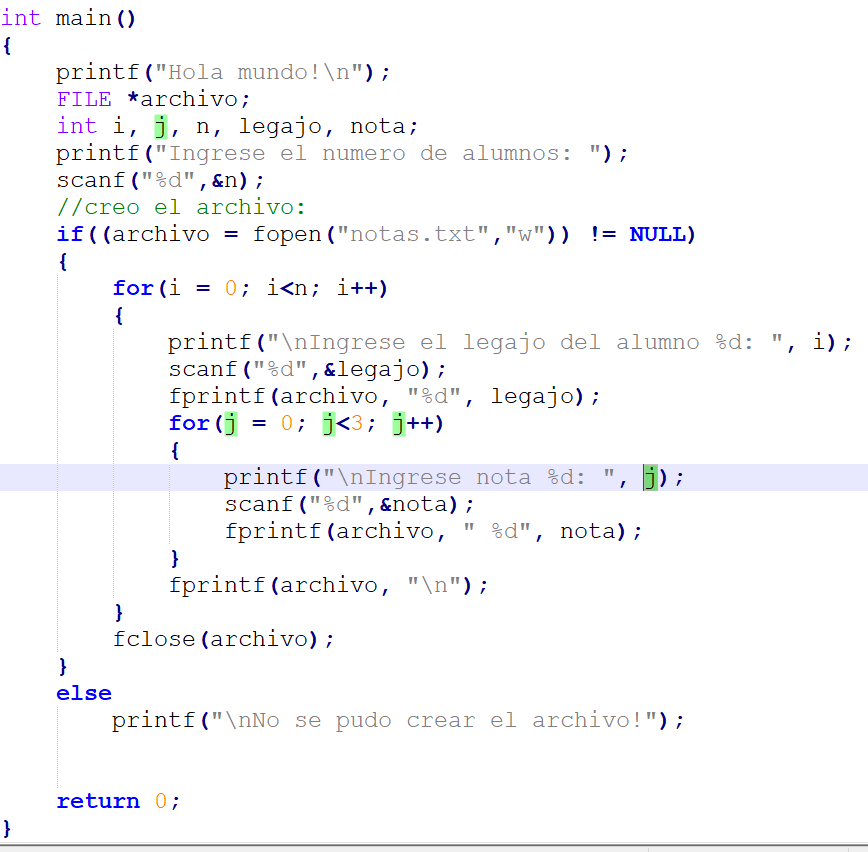
1. Escriba un programa en C que lea un archivo de texto, cuente cuantas palabras y oraciones contiene y verifique que cada oración comience con letra mayúscula, en caso contrario que la corrija.

Ayuda:

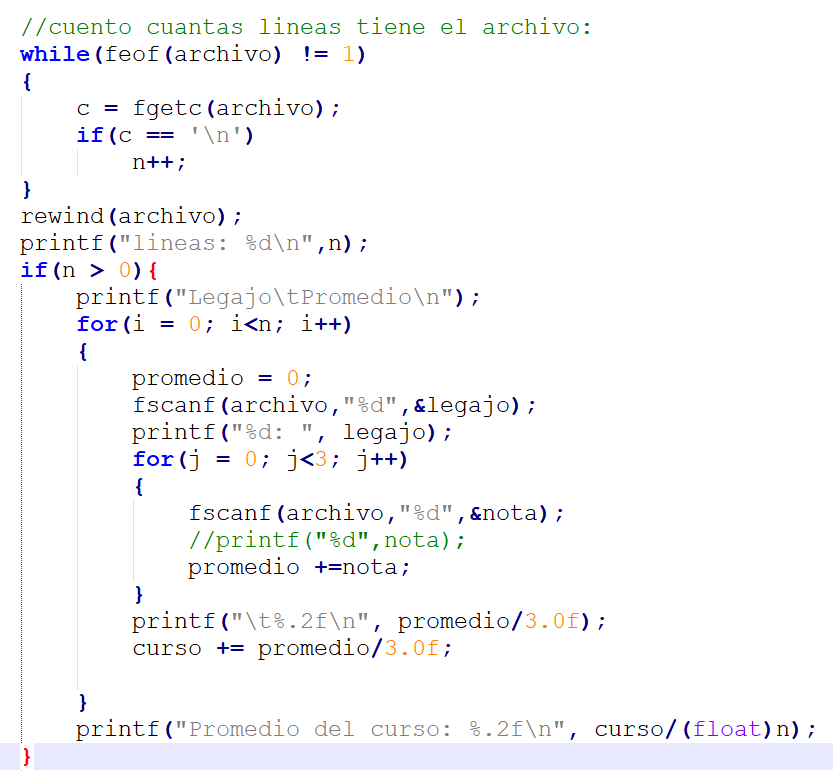


1. Escriba un programa en C que reciba por consola la información de numero de legajo y 3 calificaciones (0 a 10) de N alumnos y almacene esta información en un archivo de texto “notas.txt”. En cada línea del archivo se guardará la información de cada alumno en el siguiente orden:

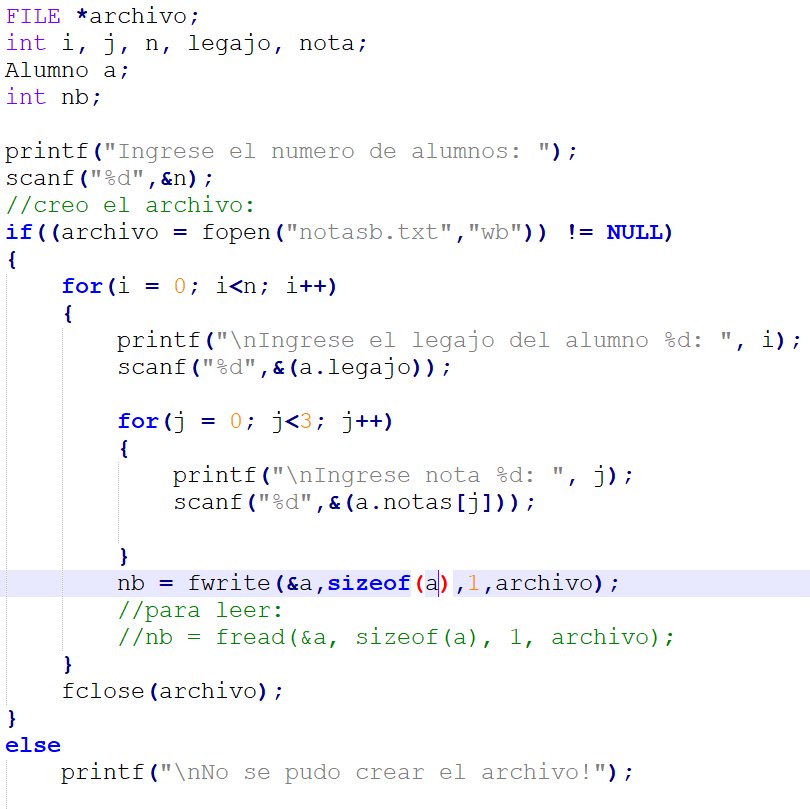
Legajo nota1 nota2 nota3



1. Escriba un programa en C que abra en modo lectura el archivo creado por el programa anterior y muestre en pantalla el legajo y promedio de cada alumno así como el promedio general de todos los alumnos.



1. Modifique el programa del ejercicio 6 para solicitar además el nombre del alumno y almacene la información de cada alumno en una variable de tipo struct. Luego almacene la información de todos los alumnos directamente en un archivo de tipo binario.



1. Escriba un programa en C que abra en modo lectura el archivo creado por el programa anterior y muestre en pantalla el legajo, nombre y promedio de cada alumno así como el promedio general de todos los alumnos.
2. Escriba un programa en C que abra un archivo de audio en formato WAVE, lea los datos de audio y calcule el valor RMS de la señal del mismo.