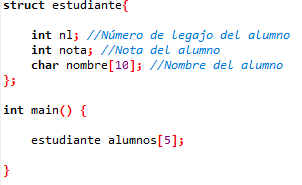
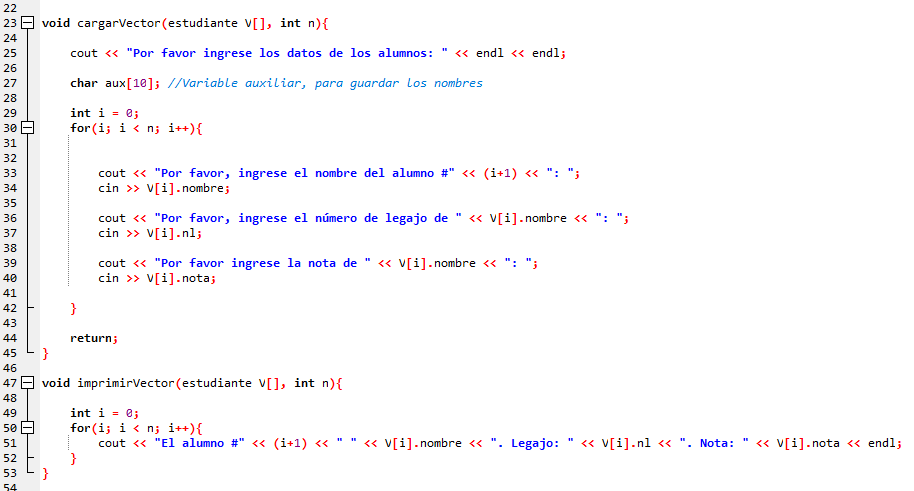
# Archivos

Para el siguiente resumen voy a hacer de cuenta que en mi programa tengo declarado un struct y un vector.



Un registro (struct) fuera del main, funciona de manera GLOBAL (lo puedo usar en cualquier función)

Tiene para cada estudiante: un número de legajo, una nota y un nombre.

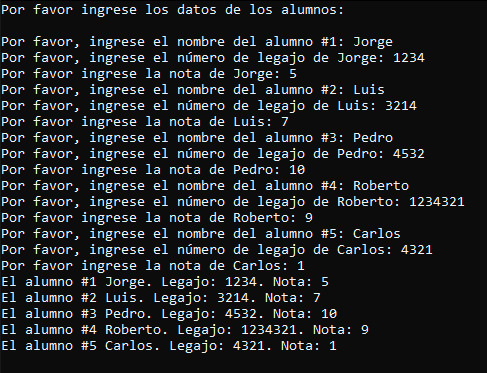
Voy a cargar el vector con datos para poder guardarlo. Voy a usar las siguientes funciones.

Recordar poner el prototipo

Se invocan de la siguiente manera:



Y el programa hasta ahora funciona así:



Recomiendo poner cout << endl al final de cada bloque de instrucciónes para separar las cosas que se imprimen y queda más prolijo.

Vamos a hacer un programa que cree un archivo nuevo y guarde los alumnos. Antes vamos a ver que modos hay:

|  |  |  |
| --- | --- | --- |
| w | Write (texto) | Archivo para escritura de texto |
| wt | Write (texto) | Archivo para escritura de texto |
| wb | Write (binario) | Archivo para escritura binario |
| wb+ | Write (binario) + read | Archivo para escritura binario y también se puede leer |
| r | Read (texto) | Archivo para lectura de texto |
| rt | Read (texto) | Archivo para lectura de texto |
| rb | Read(binario) | Archivo para lectura binario |
| rb+ | Read (binario) + write | Archivo para lectura binario, y también se puede escribir |

¿Diferencia entre rb+ y wb+?

En la función fopen si le decimos que el modo de apertura es:

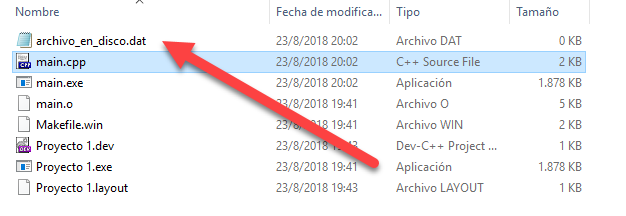
rb+: si existe el archivo, lo abre, sino no hace nada ( la variable queda NULL)

wb+: abre un archivo en BLANCO para escribir, exista o no (si no existe lo crea, si existe lo pisa).

Archivos:

Creamos un flujo de datos (FILE\*) que “contiene” el archivo y tiene un nombre lógico.

🡨 f es el nombre lógico, es el que vamos a usar dentro del programa, se suele usar f, puede llamarse de cualquier forma.



Ese es el nombre físico de un archivo.

Crear Archivo:

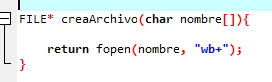
Usamos la función **fopen(nombre físico, modo de apertura)** para vincular el nombre lógico con el nombre físico. Una forma de hacerlo es la siguiente:



Yo prefiero hacerlo de esta manera:



Y la función creaArchivo es así:



Abrir archivo

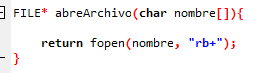
Si el archivo ya existe no hace falta crearlo, y tampoco queremos pisarlo.

Se puede hacer así:

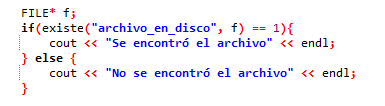


Yo prefiero hacerlo de la siguiente manera:  

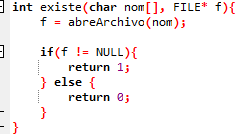

Y la función abreArchivo es así:



Para que esto funcione es necesario que el archivo exista previamente, si queremos validar la existencia yo hago lo siguiente (no se si está bien, o si es lo que los profes quieren, solo se que a mi me sirvió)



Y la función existe() la hago así:



Esto es porque si pudo abrir el archivo, entonces f seguro NO está vacío.

Si no pudo abrir el archivo porque no existe, entonces f es NULL.

Guardar archivo

Usamos la función **fwrite()** que lleva como parámetros, el dato a guardar, el tamaño en bytes del dato, la cantidad de bloques y el flujo.

Para tener en cuenta: la función **sizeof()** devuelve la cantidad de bytes de un dato, entre los paréntesis puede ir el tipo de dato, o el nombre de la variable ej:

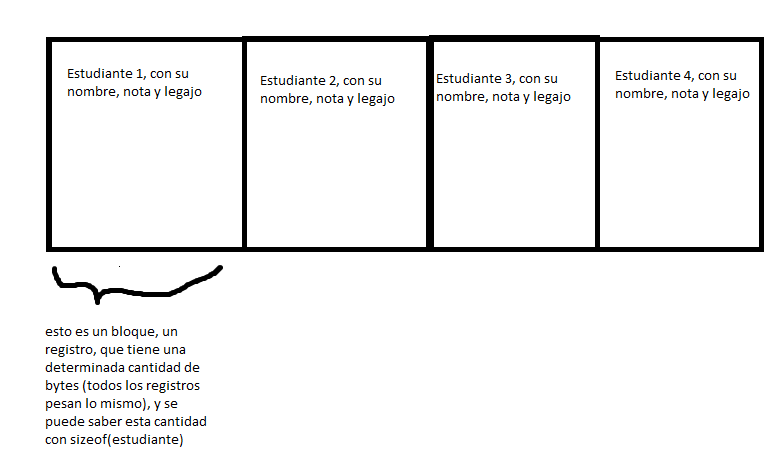
**int a;**

para saber la cantidad de bytes usamos **sizeof(int)** o **sizeof(a),** es lo mismo. En nuestro caso, el dato es de tipo estudiante, así que usamos **sizeof(estudiante)**

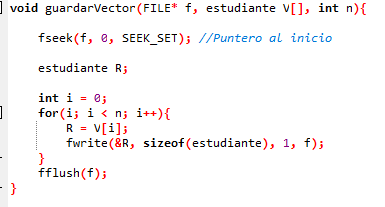
Para guardar un registro en un archivo usamos fwrite() de la siguiente manera;



Volviendo al vector que hicimos (alumnos) lo vamos a guardar en el archivo que creamos.



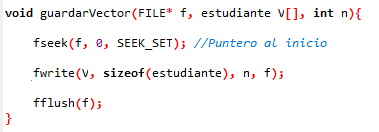
Sabiendo esto, tenemos dos formas de guardar los registros, de a uno por uno, o el vector entero.



* De a uno por uno:

(Este método nos puede servir si solo queremos guardar algunos de los registros del vector, pero no todos)

Es lo mismo que guardar un solo registro, pero al meterlo en un for, usamos esa misma estructura para un vector cualquiera.

* Todo el vector entero

**TENER EN CUENTA QUE ‘n’ ES LA CANTIDAD DE POSICIONES QUE TIENE EL VECTOR CON EL QUE ESTAMOS TRABAJANDO.**



Se invoca así:

NOTA: **fflush(f)** vacía el buffer, es recomendable ponerlo siempre que escribimos en un archivo.

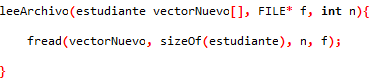
Leer un archivo

Igual que al escribir, la función **fread()** recibe los mismo parámetros que **fwrite()**

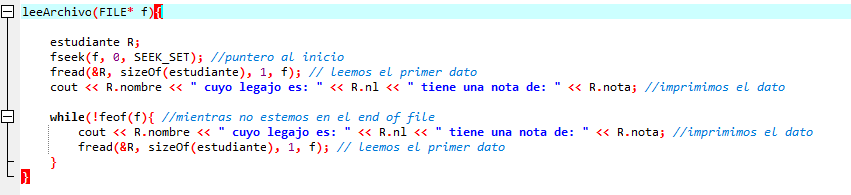


También podemos leer un solo registro:

O podemos leer de a varios (el vector entero, por ejemplo)

SI sabemos cuántos registros hay hacemos así:

Si NO sabemos cuántos registros hay, por ejemplo, para imprimir todos los registros de un archivo, podemos hacer así:



Cerrar flujo

Cuando terminamos de usar un flujo, hay que cerrarlo.

Yo lo hago así:

Recursos y cosas para tener en cuenta:

* sizeof(R) o sizeof(estudiante) es lo mismo. (suponiendo que R es un estudiante)
* &V[0] = V
* fread() devuelve un 1 si puede leer, devuelve un 0 si no puede leer
* la función ftell(f) dice cuántos bytes hay desde el principio del archivo hasta donde se encuentra el puntero
* para saber cuántos registros hay en un archivo podemos hacer lo siguiente :
* fseek() sirve para poner el puntero en un lugar en específico ej:

