**ARCHIVOS**

ES una estructura de datos pero que no va a esta almacenada en memoria sino almacenada de manera en algún dispositivo externo

Ventaja es que esa estructura de datos, de acuerdo a las necesidades que tengamos de registrar información, puede crecer o decrecer e incluso un mismo conjunto de información puede ser este utilizado por varios programas si fuera necesario.

Ahora nosotros vamos a poder crear un tipo especial específico de archivo que es un **archivo de**

**datos** que son archivos que se crean para una aplicación y por lo tanto vamos a poder leer y escribir vamos a poder hacer que nuestros programas lean y escriban en esos en esos archivos.

**Características de los Archivos:**

1. Primero solo va a guardar información de un solo tipo de dato
2. Todos los registros tienen la misma las mismas propiedades o atributos o campos. (son registros de longitud fija).
3. Los registros permanecen mientras sean útiles mientras sean el archivo puede crecer o decrecer
4. No existen registros repetidos como norma general (cada registro se tiene que y se tiene que se tiene que diferenciar de otro en al menos un atributo o una propiedad)

**Operaciones Básicas con archivos:**

|  |  |  |
| --- | --- | --- |
| **Operación** | **Función** | **Observación** |
| Crear/Abrir | **fopen** | Se solicita autorización al SO |
| Leer | **fread** | Va desde el disco a la memoria (para poder leerlo primero tengo que haberlo creado y haberle escrito algo) el procesador trabaja contra la memoria, por eso voy a ir al disco voy a leer lo que está en el disco y lo voy a guardar en una variable de memoria |
| Escribir | **fwrite** | Va desde la memoria al disco (guardo en una variable de memoria y después lo llevo al disco) |
| Cerrar | **fclose** | Se pide al SO que clausure la conexión entre el programa y el archivo. |

**Crear o abrir un archivo**

**Función fopen**

FILE \* fopen (“ruta al nombre del archivo”, “modo de apertura”)

FILE un tipo de datos puntero file que nosotros vamos a usar para establecer la conexión entre el programa y el archivo físico.

Ejemplo:

Como es un puntero, primero declaramos una variable de tipo puntero y después le guardamos adentro lo que queremos que se guarde en el archivo:

***FILE \* = pAlum;***

***pAlum = fopen(“alumnos.dt”,”wb)***

**Modos de apertura**

* WB: este modo que crea un archivo vacío. Si yo pongo WB lo que obtengo acá es un archivo vacío sin ningún registro, Pero ojo le importa un comino si el archivo existe o no existe Si yo tengo un archivo que se llama alumnos.dat y le guardé registros y hablo y lo abro con WB lo sobreescribo y se pierde la info que ya tenía guardada el archivo.
* AB: appen Bueno es para agregar registros Si el archivo existe y tiene registros me deja preparado para que le agregue Si no existe crea el archivo Crea el archivo
* RB: read es de solo lectura. El archivo tiene que existir para ser

Como lo que se está haciendo es una petición al sistema operativo que me deje modificar dentro de un archivo que está en un dispositivo externo a la memoria puede ser que haya obtenido o no

el acceso a ese archivo Así que siempre después de intentar la apertura

tengo que preguntarme qué pasó con ese puntero:

***if(pAlum ===nullptr){***

***cout << “no se pudo abrir el archivo”***

***return -1,***

***}***

Una vez que ya tengo la memoria lo que quiero cargar, tengo que escribir en el disco con la función

***fwrite (&obj, sizeof obj,1,pAlum)***

Los parámetros son :

1. En primer lugar, la dirección de memoria de la variable que contiene la información que yo quiero llevar al disco y como es dirección de memoria va con el
2. El segundo es el tamaño que tiene esa variable 🡺 sizeof: me da el tamaño de una variable o de un tipo de datos
3. Tercer parámetro ¿cuántos registros quiero grabar?
4. Cuarto Parámetro cuál es el archivo sobre el cual quiero escribir y el archivo se representa mediante el puntero file que sobre el que hicimos la apertura

**Agregar otro registro al mismo archivo**

Tener en cuenta que, si yo le quisiera agregar otro registro, en el fopen, le tengo que cambiar el modo de apertura a AB, para que no me sobreescriba el archivo:

***FILE \* = pAlum;***

***pAlum = fopen(“alumnos.dt”,”ab”) 🡺 cambio a append***

***if(pAlum ===nullptr){ 🡺 verifico que se haya podido abrir***

***cout << “no se pudo abrir el archivo”***

***return -1,***

***}***

***fwrite (&obj, sizeof obj,1,pAlum) 🡺 escribo***

**Leer un Archivo**

**Función fread 🡺 traer desde el disco a la memoria**

***pAlum = fopen(“alumnos.dt”,”rb”) 🡺 se cambia el modo de apertura a read***

***if(pAlum ===nullptr){ 🡺 se chequea que se haya podido abrir***

***cout << “no se pudo abrir el archivo”***

***return -1***

***}***

***fread (&obj, sizeof obj,1,pAlum)🡺 son los mismo parámetros pero entendidos de otra manera:***

* &obj, es la dirección de memoria en donde se va a escribir una cantidad de bytes equivalente a multiplicar sizeof obj (tamaño del objeto)\* 1(cantidad de objetos). Y en la dirección de memoria en donde esté el puntero pAlum, se va a asignar esa cantidad de memoria y a partir de la dirección de &obj se va a escribir. Como la operación de lectura trae información del disco a la memoria, la tengo que guardar en algún lado par después mostrarla.

En obj ya tengo almacenado lo que leí, entonces para mostrarlo, simplemente llamo a la función del objeto.

***obj.Mostrar()***

Si en el archivo tengo más de un objeto, cada vez que le de la instrucción de leer, me va a leer el siguiente objeto. Eso sucede por lo siguiente:

* Cuando abrimos el archivo en modo RW entonces hay un indicador del puntero file que se queda al inicio el archivo. Cuando hace la operación de lectura, le digo que lea, desde donde está el puntero, una cantidad de bytes (sizeof de obj y un registro) eso me da un número. Entonces la acción de lectura hace dos cosas: Por un lado agarra el registro que está en el archivo en el disco y lo copia en la memoria y después corre el puntero y lo deja apuntando al registro que sigue. Por eso son registros de longitud fija.

**LEER MAS DE UN ARCHIVO**

Si quiero leer más de un registro se puede hacer, porque la función lee por fila. Para ello en la función fread le tengo indicar que quiero leer 2:

***fread (&obj, sizeof obj,1,pAlum)***

Pero ojo que yo te ojo que siempre te tiene que coincidir el recipiente con lo que vos querés guardar en el recipiente. Si quiero guardar más de un registro, necesiaria guaradarlo en un array y en el ejemplo, obj solo puede guardar un elemento, por lo que daría error.

La función fread devuelve la cantidad de registros que leyó (devuelve 0 cuando no hay más archivos para leer), con esa información se puede hacer un ciclo while (inexacto, ya que no sé cuántos registros tengo) para recorrer todo el archivo y leer lo que tengo adentro.

**Cerrar un Archivo:**

Nunca hay que olvidarse de cerrar la conexión una vez que se terminó de trabajar en el archivo

***fclose(pAlum) 🡺 es como liberar la memoria***