ARCHIVOS SECUENCIALES

¿Qué son los archivos?

¿Cuándo y para qué uso un archivo?

Organización de Archivos

Modo de Acceso

Operaciones con archivos: abrir, cerrar, leer y grabar

En siguientes documentos trabajaremos con:

Patrones de resolución: corte de control; apareo; actualización

Acceso directo a los registros

¿Qué son los archivos?

Es un grupo de datos estructurados almacenados en un medio externo y puede ser accedido para su uso por medio de una computadora en todo momento.

Mientras que los vectores -que son datos almacenados en memoria- "desaparecen" al terminar de ejecutar el programa, los datos de los archivos permanecen hasta que otro programa decide "eliminarlos".

Hablamos de grupo de datos estructurado. A los componentes de este grupo de datos se los denomina *registro*.

Un archivo es una sucesión secuencial de registros.

A su vez un registro está formado por *campos* y constituye una unidad de información (referida por ejemplo a un cliente, a un alumno, etc.).

Estos campos son la unidad mínima de información del registro.

¿Qué son los archivos?

Si prestamos atención, al párrafo anterior, podríamos decir que un archivo se compone de datos organizados de forma similar aun vector.

Como veremos más adelante se pueden realizar las mismas operaciones que con un vector.

Entonces, ¿dónde está la diferencia?

Los datos de un vector están en memoria mientras se ejecuta el programa.

Los datos de un archivo están en un periférico y se acceden toda vez que se necesita

¿Para qué se utiliza un archivo?

Generalmente, un archivo es usado para acceder a información que se usa frecuentemente, o para almacenar un gran volumen de información que va a ser usada por distintas personas o distintas áreas de una organización.

Encontramos a nuestro alrededor varios ejemplos de utilización de archivos: la información de un padrón electoral, se encuentra en un archivo, donde cada registro corresponde a una persona con todos sus datos (documento de identidad, domicilio, ocupación, etc.); los datos de los estudiantes de una facultad también se encuentra en un archivo; los artículos de un comercio, etc.

¡¡¡Nuevamente podríamos pensar en qué se diferencian los archivos de los vectores.!!!

¿Para qué se utiliza un archivo?

Hasta ahora mencionamos que los **vectores** son "efímeros" , por lo cual los datos que contienen no pueden pasar de un programa a otro. Recordemos que los vectores son **estructuras estáticas** (se define su dimensión en el diseño).

Vemos ahora que los archivos permiten agregar registros casi (digamos) sin límites.

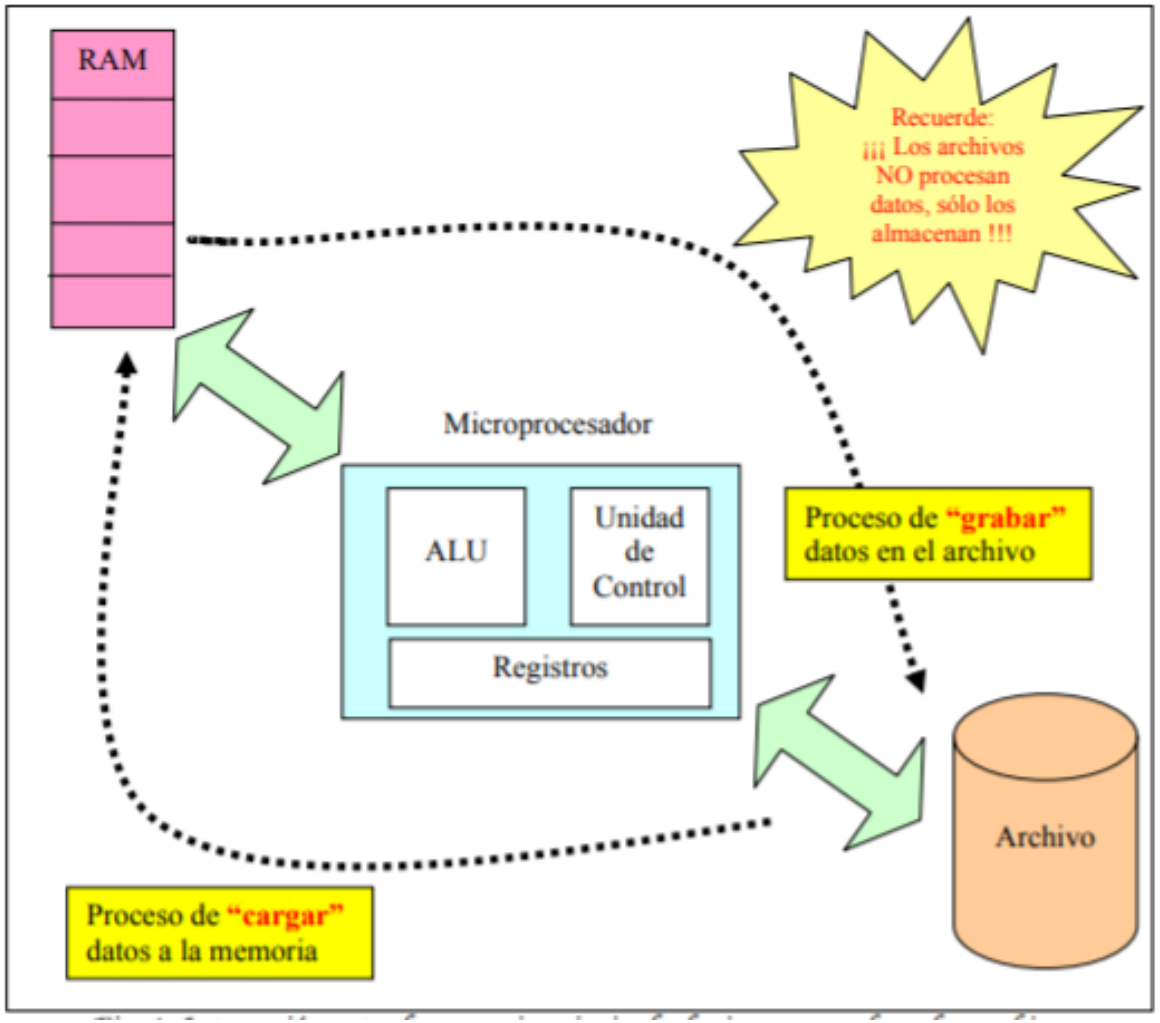
Recordemos que los **vectores** son estructura de **acceso muy rápidas** debido a estar en memoria.

Los archivos son estructuras que requieren el intercambio memoria-periférico, por lo cual son estructuras más lentas.

Los archivos permiten diversas formas de organización y métodos de acceso (por ejemplo base de datos) los cuales permiten que se acceda a un dato muy rápidamente. Los vectores también.

Breve síntesis

Los datos en memoria (vectores) no son permanentes, requieren que el programa esté funcionando para acceder a ellos.
No pueden ser "compartidos" entre distintos programas (cada programa accede a su propio espacio de memoria).
No pueden ser "transportados" entre diversos dispositivos.
Son procesados "directamente" en el programa, quiere decir que se acceden sin ninguna intermediación.



Interacción entre el procesador, la memoria y el archivo (ver cita)

Cita: "Manejo de Archivos en Lenguaje C++" Ing. Bruno López Takeyas, M.C. INSTITUTO TECNOLÓGICO DE NUEVO LAREDO DEPTO. DE SISTEMAS Y COMPUTACIÓN
http://www.itnuevolaredo.edu.mx/takeyas/apuntes/Administracion_Archivos/Apuntes/Manejo%20de%20Archivos%20en%20Lenguaje%20C++/Manejo%20de%20Archivos%20en%20Lenguaje%20C++.pdf

Los datos en los archivos no requieren estar activos y con energía eléctrica (pensemos en un pendrive). Se pueden usar en distintos dispositivos y por diferentes programas.

Se pueden acceder en forma simultanea (por ejemplo varias aplicaciones pueden agregar, eliminar, consultar, modificar) registros en un archivo.

Se debe tener en cuenta que no se realizan operaciones sobre el dispositivo externo (salvo leer y grabar).

Para procesar registros primero se deber "trasportar" a la memoria. Requieren intermediación.

El autor mencionado en la filmina anterior presenta un esquema clásico de una computadora: microprocesador (unidad aritmética y lógica; unidad de control; registros); memoria; periféricos y describe la interacción entre los distintos componentes. En el documento citado expresa con claridad cómo es el proceso, el cual trascribo casi textual.

Existe una estrecha relación entre la memoria principal, el microprocesador y los dispositivos de almacenamiento secundario ya que el procesamiento que realiza una computadora es tarea absoluta del microprocesador en conjunción con la memoria principal; es decir, los dispositivos de almacenamiento secundario (discos, flash drives, etc.) no procesan datos, solo los almacenan.

En estos dispositivos sólo se reflejan los datos previamente procesados y funcionan exclusivamente como un medio de almacenamiento. Por tal motivo, desarrollar algoritmos con uso de datos en archivos requiere nuevas técnicas de resolución.

Por ejemplo si se requiere modificar datos de un registro del archivo es necesario "cargarlo" en la memoria principal, es decir, localizar el registro en el archivo y leerlo para colocar sus datos en la memoria RAM, ahí modificarlo y posteriormente grabarlo en la misma posición en la que se encontraba.

Para modificar datos en una celda de un vector, basta con buscar y cambiar el valor.

Es importante destacar que los archivos no reemplazan a los vectores, sino que los complementan.

Se debe determinar en cada caso cuál es la estructura de datos más adecuada.

Organización de Archivos

El término organización de archivos se aplica a la forma en que se estructuran los registros sobre el soporte magnético o digital (disco, pendrive, etc. durante su grabación.

Podemos hacer una breve enumeración de las formas básicas de organización de archivos: secuencial, relativa y secuencial indexada.

Las base de datos son estructuras más complejas que combinan datos con funciones.

Las diferentes organizaciones mencionadas se deben a los diversos momentos que fueron pensadas y guardan relación estrecha con los dispositivos existentes en cada etapa.

Al principio los dispositivos tenían un comportamiento secuencial (por ejemplo una cinta magnética), luego con la aparición de los discos (magnéticos, laser, flash) se pudo acceder a la información en forma directa (sin necesidad de recorrer todo la superficie).

El paso de dispositivos secuenciales a dispositivos de acceso directo, permitió describir algoritmos más elaborados y eficientes, toda vez que brindó la posibilidad de acceder a los datos en forma directa (tal como los vectores en memoria).

Estos nuevos desarrollos se fueron complejizando con el tiempo, hasta permitir organizar (guardar y recuperar) los datos mediante diferentes identificadores (claves), permitiendo así que la información pueda estar organizada por más de un campo.

Modo de acceso

La forma en que se organiza un archivo determina el modo de acceso al mismo. Se refiere al procedimiento que se debe seguir para situarse en un registro determinado para realizar una operación de lectura o grabación del mismo.

Puede ser secuencial o directo.

En el modo de acceso secuencial para llegar a un registro es necesario pasar por todos los anteriores, mientras que en el modo de acceso directo se puede llegar directamente a un registro conociendo únicamente el número de registro (ubicación en el archivo relativa al inicio del mismo).

Al momento de diseñar un sistema informático se debe decidir qué tipo de archivos utilizar. Es una tarea compleja debido a los múltiples criterios a tener en cuenta: rápido acceso, fácil actualización, economía de almacenamiento, fácil mantenimiento.

El modo de acceso directo se puede llevar a cabo de varias formas:

1. La posición que ocupa el registro (número de registro).
2. Calculando la posición que ocupa el registro en el archivo a partir del contenido de un campo (clave o identificador de registro).
3. Mediante el uso de tablas de índices. La localización de un registro se hace buscando en la tabla de índices el valor del campo clave, de donde se obtiene la posición en que está grabado el registro dentro del archivo (acceso indexado).

Archivos secuenciales

La organización secuencial es la forma más simple de almacenamiento de información en un medio magnético o laser. Los registros están organizados de manera secuencial, es decir uno detrás del otro y con una marca a continuación del último que indica el final del archivo. Esta marca se la conoce con el nombre de End Of File (EOF).

La lectura se realiza desde el primer registro, pasando por cada uno de los restantes hasta llegar al final.

Las primeras aplicaciones con archivos secuenciales eran muy limitadas (debido a los dispositivos del momento), permitiendo solamente este tipo de recorrido, grabar registros uno a continuación del otro y leer con igual criterio.

No era posible tener accesos directos (leer o grabar en cualquier lugar del archivo), limitando mucho su uso.

Con el paso del tiempo y la aparición de dispositivos de acceso directo, se agregaron nuevas operaciones permitiendo de esta manera acceder a cualquier registro del archivo sin necesidad de pasar por todos los anteriores, leer un registro (transferir su contenido a memoria), modificar el contenido (en memoria) y volver a grabar el registro (de memoria a disco).

Pero, qué pasaba si un registro estaba de más (necesidad de borrarlo). Esta es la única limitación para este tipo de organización de archivos.

Esto se debe a que el acceso a los registros es físico, lo cual quiere decir que el registro 10 está en una posición única a partir del inicio del archivo, la cual se calcula matemáticamente conociendo el tamaño del registro (todos los registros son del mismo tipo, por lo tanto tienen en mismo tamaño).

Vamos a trabajar con archivos secuenciales

Repasemos:

Un archivo secuencial es un conjunto de datos agrupados en registros en un medio de almacenamiento permanente. Estos datos pueden ser utilizados por varios programas, permitiendo así que la información "persista".

Se conocen con el nombre de "archivos secuenciales" debido a que los registros se encuentran "físicamente" uno detrás de otro. Todos los registros tienen el mismo "tamaño" (recordar cómo se define un registro en C++) por lo cual no hay separador entre registros. Al final del archivo hay una "marca de fin de archivo".

Al "principio de los tiempos" estos archivos estaban en dispositivos de acceso secuencial, por lo tanto las únicas operaciones que se podían realizar eran leer o grabar. Para ser más claro, un programa grababa (un registro detrás de otro desde el inicio) y otro programa leía (un registro detrás de otro). No se podía escribir un programa que lea y grabe (ambas operaciones en el mismo programa). Tampoco se podía modificar el contenido de los registros.

Al aparecer los dispositivos de acceso directo, los lenguajes de programación agregaron funciones que permiten acceder en forma directa a cualquier registro, por lo tanto un programa puede leer y/o grabar en cualquier posición del archivo.

Esto permite, entre otras operaciones, leer un registro (pasar los datos a memoria), modificar los datos en memoria y luego volver a grabar (con algún cuidado en el algoritmo).

Esta característica permitió que se desarrollen "administradores" complejos de datos: archivos secuenciales indexados (VISAM, ISAM), y bases de datos.

Características de trabajo con archivos secuenciales

Como describimos anteriormente, en los archivos secuenciales los registros están organizados de manera secuencial, es decir uno detrás del otro y con una marca a continuación del último que indica el final del archivo. Esta marca se la conoce con el nombre de End Of File (EOF).

Por lo tanto, la lectura se realiza desde el primer registro, pasando por cada uno de los restantes hasta llegar al final, sin posibilidad de retroceder en el recorrido, o de "saltar" aleatoriamente a cualquier registro.

Recordemos nuevamente que la posibilidad de acceso directo se agregó mucho tiempo después del inicio del uso de archivos.

Hasta el momento debemos tener bien claro que este tipo de archivos **no permite la eliminación de registros ni la inserción en cualquier lugar, salvo al final.** Entonces, para poder resolver la inserción de registros se utilizan algoritmos llamados **apareo de archivos** que a partir del archivo original y uno o varios archivos de novedades generan un nuevo archivo actualizado que luego reemplazará al original.

Hagamos una analogía con los vectores. Recordemos que un vector se puede recorrer secuencialmente (pensemos por un momento que no permite el acceso directo).

Supongamos que tenemos un vector al que llamamos **"Maestro"**, y otro vector al que llamamos **"Novedades"**. La única restricción a tener en cuenta es que ambos vectores deben tener un campo en común (clave) y deben estar ordenados por ese campo.

Entonces, mediante un algoritmo de apareo de vectores podemos obtener un nuevo vector al que llamaremos **"Maestro actualizado".** BINGO, es el mismo algoritmo para vectores y archivos.

Pero, a no asustarse, como se mencionó anteriormente los archivos secuenciales permiten el acceso directo, así que también trabajaremos con algoritmos para modificar datos en modo directo.

Operaciones con archivos secuenciales

- abrir archivo
- grabar registro
- leer registro
- detectar fin de archivo
- posicionamiento directo
- cerrar archivo

Operaciones con archivos secuenciales

Antes de escribir o leer datos de un archivo es necesario **abrirlo**. Esta operación establece la comunicación entre el programa y el sistema operativo

La instrucción para abrir el archivo debe proporcionar al "Sistema Operativo" el nombre completo del archivo y cómo se va a utilizar : leer o escribir datos.

Una vez abierto el archivo, el "Sistema Operativo" devuelve información al programa (en una variable llamada puntero), la cual se utiliza para el resto de las operaciones.

¿Qué es un puntero? Es una variable que hace referencia a una región de memoria; en otras palabras es una variable cuyo contenido es una dirección de memoria.

Hay que tener en cuenta:

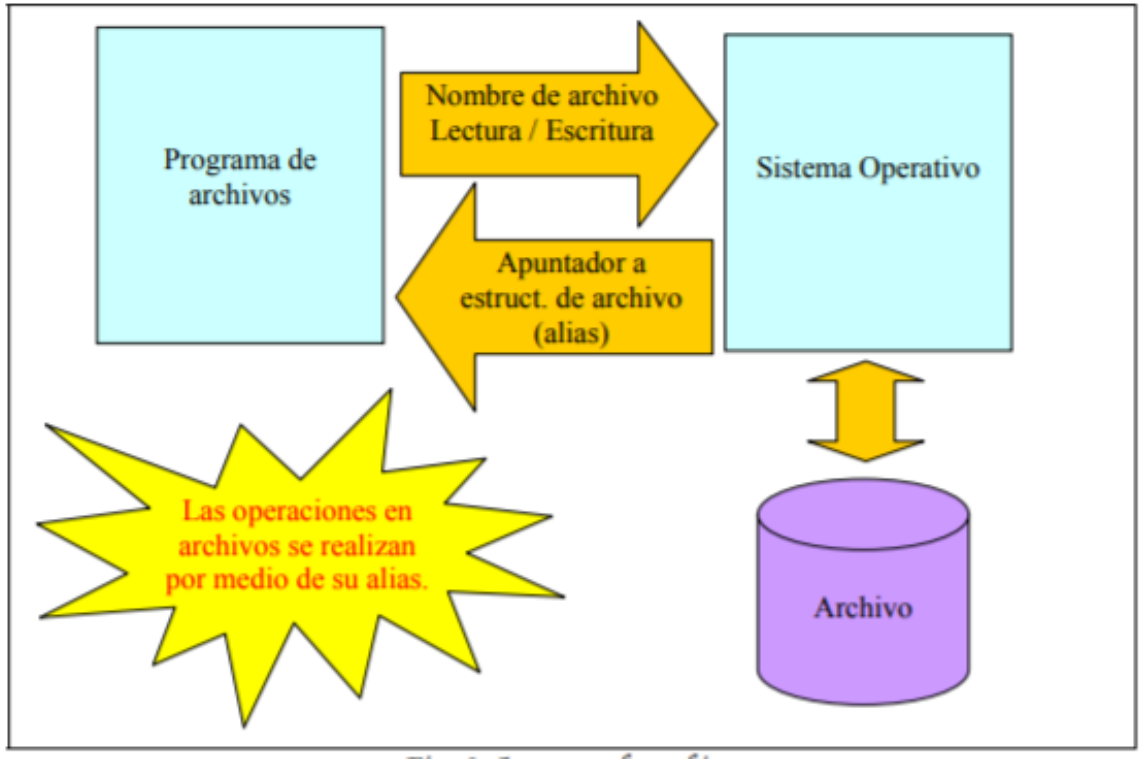
- un archivo que se va a leer (modo de apertura lectura) debe existir.
- un archivo que se va a grabar (modo de apertura grabación) no debe existir, caso contrario se borra.

Operaciones con archivos secuenciales

El siguiente esquema (fuente Ing. Bruno López Takeyas, M.C.) nos permite observar gráficamente cómo es la operación de apertura de archivo:

El programa indica el nombre físico del archivo (incluye carpeta), el modo de apertura, y recibe en una variable de tipo puntero (llamada alias) información respecto a la operación realizada.

Esta variable se usa en el resto de las operaciones con archivos.



Variables a utilizar en las operaciones con archivos

Para trabajar con archivos en el lenguaje de programación C++, debemos contar con 2 variables.

En la primera de ellas se almacenará la información acerca de la ubicación del archivo en el dispositivo externo (llamada alias en la filmina anterior) y la segunda variable deberá ser de tipo estructura o registro, que es en donde se almacenarán los datos del registro del archivo.

Como ya se mencionó anteriormente, mediante la variable de tipo "puntero" llamada alias se relaciona el nombre físico del archivo (ubicación en el disco) con el programa.

Para ser más claro, a partir de la operación de apertura del archivo, el resto de las operaciones se realizan a través de la variable alias (puntero al archivo).

En el caso de lectura y grabación de archivos son necesarias dos variables:

- El puntero (alias) que indica la información "física" del archivo (donde se va a realizar la operación de lectura o de escritura),
- y la variable de tipo registro desde o hacia donde se van a realizar las transferencias entre el archivo y la memoria.

Variables a utilizar en las operaciones con archivos

Intentemos aclarar un poco más cómo funcionan las operaciones de lectura y grabación:

Cada vez que se invoca a la función de lectura, se transfiere un registro a memoria y se avanza para la siguiente lectura.

Cada vez que se invoca a la función de grabación, se transfiere un registro desde la memoria y se avanza para la próxima grabación.

La información referida a dónde se debe leer o grabar la contiene la variable puntero o alias, que es "modificada" por cada operación.

Operaciones básicas con archivos secuenciales

Abrir archivo Se usa la función **fopen** para abrir un archivo, determinar el modo de apertura y establecer la vía de comunicación mediante su alias correspondiente. Además determinar el tipo de contenido del archivo (texto o binario). Esta función tiene dos argumentos: el nombre del archivo y su modo

Cerrar archivo. Antes de dejar de utilizar un archivo es necesario cerrarlo. Esto se logra mediante las funciones **fclose** o **fcloseall**.

Leer de archivo. La función **fread** permite "cargar" todos los campos de un registro desde un archivo, es decir, lee un registro y lo copia en la memoria

Grabar en archivo. La función **fwrite** proporciona el mecanismo para almacenar todos los campos de un registro en memoria en un archivo.

Reposicionar el puntero de un archivo. La función **fseek**, permite posicionar el puntero o alias de un archivo para realizar lecturas y grabaciones NO SECUENCIALES.

Detección del fin de archivo. Mediante la función **feof()** se detecta si se ha encontrado el final de un archivo. Si se encuentra el final de un archivo, devuelve un valor diferente de cero y cero en caso contrario.

Operaciones básicas: ABRIR ARCHIVO

Se usa la función **fopen** para abrir un archivo, se debe indicar el modo de apertura y establecer la vía de comunicación mediante su alias (variable de tipo **FILE ***) correspondiente. También se define el tipo de contenido del archivo (texto o binario). Esta función tiene dos argumentos: el nombre del archivo y su modo .

Una variable de tipo **FILE *** define un "puntero" a un objeto de tipo **ARCHIVO**. mediante esta variable se establece el vínculo (en la función abrir archivo) entre el nombre físico del archivo (como se lo define en las carpetas) y el nombre lógico.

Modo Texto	Modo binarios	Operación
“r”	“rb”	Apertura en modo de solo lectura. El archivo debe existir
“w”	“wb”	Apertura en modo de solo escritura. Si el archivo existe, se reescribirá (pierde el contenido anterior). Si el archivo no existe, lo crea.
“a”	“ab”	Apertura en modo de agregar. Si el archivo existe, los datos se agregan al final del archivo, en caso contrario, el archivo se crea.
“r+”	“rb+”	Apertura en modo de lectura/escritura. El archivo debe existir
“w+”	“wb+”	Apertura en modo de lectura/escritura. Si el archivo existe, se reescribirá (pierde el contenido anterior).
“a+”	“ab+”	Apertura en modo de lectura/agregar. Si el archivo no existe lo crea.

Ejemplos:

```
#include  <stdio.h>
Int main() {
  FILE *arch1, *arch2, *arch3,* arch4 ,* arch5 ,* arch6 ;
  arch1 = fopen("arch1.dat", "rb") ; // leer archivo binário
  arch2 = fopen("arch2.dat", "wb") ;// grabar archivo binario
  arch3 = fopen("arch3.dat", "rb+");// lectura/grabación

  arch4 = fopen("arch4.dat", "ab");// agregar al final
  arch5 = fopen("arch5.dat", "wb+");// lectura/escritura
  arch6 = fopen("arch6.dat", "ab+");// lectura/grabación/final
```

Todos los ejemplos presentados son para archivos binários (com registros).
Para utilizar archivos de texto trabajaremos mediante las funcioens iostream

De acuerdo al tipo de operación a realizar con el archivo, es el modo que se elige al abrirlo (conectar archivo físico con el programa en memoria).

Es importante tener en cuenta que un archivo se puede (y debe) abrir en diferentes modos de acuerdo al programa. Para generar un archivo, se abre en modo escritura, para procesar (leer) un archivo en modo lectura.

La función fopen "devuelve" un valor: NULL en caso de no completar la operación (por ejemplo archivo inexistente (en apertura para lectura). SI termina correctamente devuelve un valor distinto a NULL que no se debe modificar en el programa.

Operaciones básicas: GRABAR ARCHIVO

Mediante la función **fwrite** (esta es una de varias posibilidades que ofrece el lenguaje) se graban registros en el archivo.

La operación de grabar quiere decir que se "copian" datos desde la memoria al archivo.

Los datos a grabar están agrupados en una estructura o registro. Es importante que quede claro que se transfiere de a registro (no de a campo).

Esta función tiene cuatro argumentos: la variable que se desea grabar, su tamaño en bytes, la cantidad de variables y el alias del archivo donde se desea almacenar.

Para ser más claros y precisos: por cada operación de grabación se pueden "enviar" desde memoria al archivo 1 o varios registros juntos. **EN ESTE DOCUMENTO TRABAJAREMOS ENVIANDO DE A UN SOLO REGISTRO**

Los datos se graban "a continuación" del último registro grabado. Para ser más precisos, los datos se graban en el lugar físico del archivo donde quedó el puntero.

Sintaxis:

```
fwrite(&Registro, sizeof(Registro), cantidadRegistros, aliasArchivo);
```

Donde &registro es la referencia al registro en memoria a grabar; sizeof(registro) mide el tamaño del registro; cantidadRegistros refiere a cuántos registros se graban (siempre 1) y aliasArchivo es el puntero al archivo utilizado en la función fopen

Ejemplo: grabamos en un archivo

El siguiente programa genera un archivo de estudiantes a partir de datos obtenidos desde el teclado.

Con los datos de cada estudiante (legajo, DNI, apellido y mail) arma un registro y lo graba en el archivo.

El orden de grabación del archivo coincide con el orden de ingreso de los datos

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
using namespace std;

typedef char str60[60];
struct tREstudiante {
  int legajo;
  int DNI;
  str60 apellido;
  str60 mail;
};

void pedirEstudiante(tREstudiante & rEstudiante, bool & fin);
```

Modo grabación archivo binario. El archivo se puede abrir en modo "ab", esto permite agregar al final

```
int main(){
  FILE * fEstudiante;
  tREstudiante rEstudiante;
  bool fin;
  fEstudiante = fopen("Estudiante.dat", "wb");
  pedirEstudiante(rEstudiante, fin);
  while (!fin) {
    fwrite(&rEstudiante, sizeof(rEstudiante), 1, fEstudiante);
    pedirEstudiante(rEstudiante, fin);
}
  fclose(fEstudiante) ;
}
```

```
void pedirEstudiante(tREstudiante & rEstudiante, bool & fin) {
  printf("\nInresar datos de un estudiante\n");
  printf("Ingresar legajo (0=fin)=");
  cin>>rEstudiante.legajo;
  cin.ignore() ;
  if (rEstudiante.legajo!=0) {
    fin = false;
    printf("\nIngresar DNI=");
    cin>>rEstudiante.DNI;
    cin.ignore() ;
    printf("\nIngresar apellido=");
    cin.getline(rEstudiante.apellido,sizeof(rEstudiante.apellido));
    printf("\nIngresar mail=");
    cin.getline(rEstudiante.mail,sizeof(rEstudiante.mail));
  } else fin = true;
}
```

Operaciones básicas: LEER ARCHIVO

Mediante la función **fread** (esta es una de varias posibilidades que ofrece el lenguaje) se leen registros del archivo.

La operación de leer quiere decir que se "copian" datos desde el archivo a la memoria .

Los datos a leer están agrupados en una estructura o registro.

Es importante que quede claro que se transfiere de a registro (no de a campo).

Esta función tiene cuatro argumentos: la variable que se desea leer, su tamaño en bytes, la cantidad de variables y el alias del archivo desde donde se desea leer.

Para ser más claros y precisos: por cada operación de lectura se pueden "enviar" desde el archivo a la memoria 1 o varios registros juntos. **EN ESTE DOCUMENTO TRABAJAREMOS ENVIANDO DE A UN SOLO REGISTRO**

Los datos se leen uno a continuación del otro en el mismo orden en que fueron grabados los registros. Para ser más precisos, los datos se leen desde el lugar físico del archivo donde quedó el puntero. **MÁS ADELANTE VEREMOS QUE SE PUEDE ALTERAR ESTE ORDEN**

Sintaxis:

```
fread(&Registro, sizeof(Registro), cantidadRegistros, aliasArchivo);
```

Donde &registro es la referencia al registro en memoria para la lectura; sizeof(registro) mide el tamaño del registro; cantidadRegistros refiere a cuántos registros se leer (siempre 1) y aliasArchivo es el puntero al archivo utilizado en la función fopen

Ejemplo: leemos desde un archivo

El siguiente programa lee desde un archivo de estudiantes la información grabada previamente. La lectura es en el orden en que fue grabada

Los datos de cada estudiante (legajo, DNI, apellido y mail) se muestran en forma e tabla.

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
using namespace std;

typedef char str60[60];
struct tREstudiante {
  int legajo;
  int DNI;
  str60 apellido;
  str60 mail;
};

void leerEstudiante(FILE * &fEstudiante, tREstudiante & rEstudiante, bool & fin);
```

```
Legajo DNI      Apellido                       Mail
    10 12345678 Ana                            ana@mail.com
    20 87654321 Pedro                          pedro@mail.com
    30 78787878 Carla                          carla@mail.com
```

Modo lectura archivo binario

```
int main(){
  FILE * fEstudiante;
  tREstudiante rEstudiante;
  bool fin;
  fEstudiante = fopen("Estudiante.dat", "rb");
  if (fEstudiante != NULL){
     printf("%6s %-8s %-30s %-30s\n", "Legajo", "DNI","Apellido", "Mail");
     leerEstudiante(fEstudiante, rEstudiante, fin);
     while (!fin) {
       printf("%6d %8d %-30s %-30s\n", rEstudiante.legajo, rEstudiante.DNI,
                  rEstudiante.apellido, rEstudiante.mail);
       leerEstudiante(fEstudiante, rEstudiante, fin);
     }
    fclose(fEstudiante)                ;
  } else    cout<<"No existe archivo Estudiante.dat"<<endl;
}
```

```
void leerEstudiante(FILE * & fEstudiante, tREstudiante &
rEstudiante, bool & fin){
   fread(&rEstudiante, sizeof(rEstudiante), 1,
fEstudiante);
   if (feof(fEstudiante) )
      fin = true;
   else fin = false ;
}
```

Resumiendo…..

Un archivo es una estructura en un medio de almacenamiento permanente (persiste al finalizar el programa), que permite guardar información para ser recuperada en otro momento.

Podemos pensar en ejemplos cotidianos: nuestra información en el padrón electoral, los artículos del supermercado, la información de nuestras llamadas telefónicas. Y así una innumerable cantidad de aplicaciones necesitan utilizar archivos.

Los archivos tienen diversas formas de organización, la más "sencilla" (para el desarrollo) es la secuencial y es sobre la cual está desarrollado este documento. Esta forma de organizar archivos permite también el acceso directo a través de funciones provistas por el lenguaje, temas que iremos desarrollando más adelante.

Un archivo se puede abrir (para ser usado por el programa) en diversos modos: solo lectura, solo escritura, para agregar al final y de lectura/ escritura.

Las operaciones con archivos se realizan "transfiriendo" entre el archivo físico y la memoria (en variables de tipo registro). Para ser más preciso. Una lectura transfiere un conjunto de datos del archivo a un registro en memoria. Una grabación transfiere los valores de un registro en memoria al archivo físico.

Cada vez que se realiza una operación de lectura o grabación, el "puntero" del archivo queda posicionado al final de los datos transferidos. Para ser más claros, al abrir el archivo el puntero queda posicionado al inicio. Al realizar la primera grabación, se transfiere uno (o más) registros de memoria al archivo y el puntero "se va corriendo" hasta posicionarse al final de los datos transferidos. Esto permite que la próxima grabación "no pise" los datos de la grabación actual.

Ahora, una vez leído el material, es momento de comenzar a trabajar.

En primer lugar les proponemos que creen un archivo binario que contenga los datos de los artículos de un supermercado, agreguen un campo rubro (entero de 1 a 10). Para facilitar la tarea tomen el programa publicado y realicen las modificaciones necesarias para el nuevo archivo.

Recuerden que cada vez que ejecuten ese programa, el archivo se crea nuevamente en caso que hayan utilizado el modo "wb" (write binary), por lo tanto abran (fopen) el archivo en modo "ab" (append binary). De esta manera podrán agregar al final del archivo (si no existe lo crea).

Escriban un programa que recorra todo el archivo y muestre los datos en pantalla.

Al final de este programa deben mostrar la cantidad de artículos por rubro

Sugerencia: definan un vector de 10 valores enteros y cuenten de acuerdo al rubro del artículo).

Al finalizar los programas no olviden cerrar el archivo.

Para los que quieran ir adelantando,
veamos cómo posicionarse en un registro de un archivo.
(En el próximo documento desarrollaremos las funciones y algoritmos con acceso directo)

Los archivos secuenciales permiten **regrabar registros**, por ejemplo cuando necesitamos modificar un dato de un campo de un registro.

Para esto debemos tener presente que el acceso es a nivel de byte, esto es calcular dónde comienza un registro y ejecutar una instrucción que posicione el puntero en ese byte.

Luego se debe leer el registro **(el posicionamiento no lee)**.

Una vez leído hay que tener en cuenta que el puntero avanza al siguiente registro.

Una vez posicionado y leído, se puede modificar los datos en memoria y volver a grabar en el archivo.

Para esto hay que tener en cuenta que se debe volver a posicionar antes de grabar, caso contrario se “pisa” el siguiente registro.

Ejemplo:

Posicionarse en el registro 20 y cambiar el DNI, teniendo en cuenta que el registro 0 (primer registro) comienza en el byte 0, se debe avanzar hasta al primer byte del registro 20, para lo cual se debe multiplicar 20 por el tamaño del registro

```
typedef char str60[60] ;
struct tRPer{
    long dni;
    str60 apeyNbe;
}

int main() {
  FILE *  f;
  int pos;
  tRPer r;
  f = fopen("Persona.dat", "wb+");
  pos = 20  ;   // para posicionarse en el registro 20,
  fseek(f, pos * sizeof(registro), SEEK_SET);   // posiciona desde el primer byte
  fread(&r, sizeof(r), 1, f);
  r.dni = 92887788;
  fseek(f, pos * sizeof(registro), SEEK_SET);
  fwrite(&r, sizeof(r), 1, f);
  fclose(f);
}
```