Introducción a la Algorítmica y Programación (3300)

Prof. Ariel Ferreira Szpiniak - aferreira@exa.unrc.edu.ar
Departamento de Computación
Facultad de Cs. Exactas, Fco-Qcas y Naturales
Universidad Nacional de Río Cuarto

Teoría 17

Almacenamiento persistente. Archivos

Almacenamiento persistente
Archivos o Ficheros

Memoria Principal

- Existe un límite en la cantidad disponible.
  - 640 Kb a mediados de los 80.
  - 64 Mb a mediados de los 90 (aumenta unas 100 veces a costos muy inferiores).
- Volátil.
- Electrónica → velocidad elevada.

Almacenamiento persistente
Archivos o Ficheros

Memoria Secundaria

- Ubicados fuera de la memoria RAM.
- Mayor capacidad → menor costo.
- Retienen información después de finalizar el programa o al apagar la computadora.
- Más lentos que la RAM.

Almacenamiento persistente
Archivos o Ficheros

Tiempos de acceso

- RAM : nanosegundos (10^{-9} seg.)
- Disco: milisegundos (10^{-3} seg.)

Ejemplo:

Suponiendo que un disco tiene un tiempo de acceso promedio de 15 milisegundos para recuperar un carácter, si es necesario recuperar 10.000 caracteres tardaría:

150.000 milisegundos = 150 segundos = 2' 30"

Almacenamiento persistente
Archivos o Ficheros

Archivos

Concepto

Estructuras de almacenamiento que deben asociarse con un dispositivo de memoria auxiliar permanente.

Almacenamiento persistente
Archivos o Ficheros

Definiciones

Definición 1

Colección de registros semejantes, guardados en dispositivos de almacenamiento secundario.

Definición 2

Estructura que guarda, en un dispositivo de almacenamiento secundario de una computadora, una colección de elementos del mismo tipo.

Almacenamiento persistente
Archivos o Ficheros

Un **archivo** o fichero es un medio de almacenamiento persistente.

Hay varios tipos de archivos.

Nosotros usaremos **archivos de datos homogéneos**, consistentes en una secuencia de elementos del mismo tipo, que pueden ser de un tipo simple o compuesto (estos últimos son los más habituales).

Almacenamiento persistente
Archivos o Ficheros

Se podría pensar que los **archivos** son "parecidos a los arreglos", pero con la particularidad de que los datos almacenados en ellos no se pierden al finalizar el programa que los manipula.

Los **archivos** sirven para:
1) almacenar de manera permanente la información.
2) permitir tratar de manera fraccionada a dicha información.

El manejo de los archivos está muy atado a cada lenguaje de programación. Por ello nosotros analizaremos este tema desde un punto de vista muy cercano al lenguaje C.

Tipos de Archivos

Un archivo se almacena en un dispositivo auxiliar (discos, cintas, etc), de forma que los datos obtenidos antes, durante y después del procesamiento de los datos, no se pierden. Para declarar una variable archivo es necesario definir previamente la naturaleza de los datos a almacenar en él.

Existen dos tipos de archivos de datos en C:

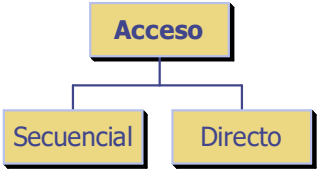
1. **Archivos de texto** (txt). Contienen texto "plano" (carácter ASCII).
2. **Archivos binarios**. Secuencia de bytes. Contienen datos de cualquier tipo (enteros, reales, registros, imágenes, texto con formatos), son archivos ejecutables, etc.

Tipos de Archivos

Nosotros utilizaremos archivos del primer y segundo tipo, es decir, **Archivos de texto y Archivos binarios**.

Para trabajar con archivos en Notación Algorítmica haremos un paralelismo directo con el manejo de archivos de C, aunque en general la mayoría de los lenguajes proveen operaciones similares.

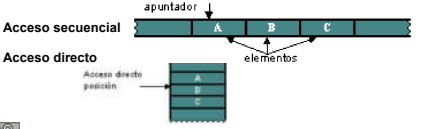
Tipos de Acceso a un Archivo



Tipos de Acceso a un Archivo

Existen dos modalidades para acceder a un archivo de datos: **acceso secuencial** y **acceso directo o aleatorio**.

- El **acceso secuencial** exige una exploración secuencial de los elementos, comenzando desde el primero. La más usada por nosotros.
- El **acceso directo** permite procesar o acceder a un elemento determinado haciendo una referencia directamente por su posición en el soporte de almacenamiento. Algo similar a los índices de los arreglos.



Tipos de Acceso a un Archivo

Los **Archivos de texto** solo pueden tratarse mediante **acceso secuencial**.

Los **Archivos binarios** pueden tratarse mediante **acceso secuencial o aleatorio**. Es decir, similar a lo habíamos visto para los arreglos.

Tipos de Acceso a un Archivo

A continuación analizaremos como definir los tipos de datos para manipular archivos y algunas de las operaciones para su tratamiento, bajo la modalidad de acceso secuencial y posteriormente bajo la modalidad de acceso directo.

Acceso secuencial:

NOTACION ALGORITMICA	C
Abrir	fopen
Cerrar	fclose
Leer	fgetc, fgets, fread, fscanf
Escribir	fputc, fputs, fwrite fprintf
EOF (fin de archivo)	feof
EOL (fin de línea). Solo para archivos de texto	no existe en C

Tipos de Acceso a un Archivo

Acceso directo:

NOTACION ALGORITMICA	C
PosicionActual	ftell
IrPos	fseek
TamañoArchivo	(usando las 2 anteriores)

Acceso secuencial - Tipo del Archivo

La declaración del archivo a que haremos referencia puede ser un archivo de texto (txt) o un archivo binario (en nuestro caso usados para manipular archivos de registros), por lo tanto debemos especificarlo.

- En notación algorítmica:

Para archivos de texto

f ∈ ARCHIVO de Texto
cad ∈ Cadena // para manipular el contenido del archivo

Para archivos binarios

TPersona = <nom ∈ Cadena, direccion ∈ Cadena, edad ∈ Z>
f ∈ ARCHIVO de TPersona
reg ∈ TPersona // para manipular el contenido del archivo

Siempre es necesario una variable de mismo tipo que los elementos del archivo para manipularlo más fácilmente.

Acceso secuencial - Tipo del Archivo

• En C esto se traduce:

```
typedef struct {
    char nombre[21];
    char direccion[41];
    int edad;
} TPersona;

FILE *f;  // FILE es la estructura que
          // define un descriptor de archivo
          //se puede usar para cualquier tipo de archivo

char cadena[100]; // para manipular el contenido del archivo

TPersona reg; // para manipular el contenido del archivo
```


Acceso secuencial - Abrir

Para saber con qué archivo deseo trabajar se hace uso de la primitiva **Abrir**, en Notación Algorítmica. Solo abre el archivo, y posiciona el dispositivo de lectura/escritura dentro del archivo (no lee ni escribe) .

Abrir(NomExt, NomInt, formaDeApertura)

nomExtArch es el nombre externo del archivo (cadena).
nomIntArch es el nombre interno del archivo.

FormaDeApertura puede ser:

- l (lectura). Abrir para lectura. El archivo debe existir.
- e (escritura). Abrir para escritura. Se crea un archivo nuevo o se sobreescribe si ya existe.
- a (agregar al final). Abrir para añadir datos al final. Si el archivo no existe, se crea.

Acceso secuencial – Abrir en C

- fopen
  - Abre un archivo para su uso

```
- FILE* fopen(char* nombre, char* modo);
```

Devuelve el descriptor del archivo para su uso **interno**. NULL en caso de error

Modo de apertura (lectura, escritura, etc.)

Nombre **externo** del archivo a abrir

Acceso secuencial – Abrir en C

Modos de apertura

r	Abrir para lectura. El archivo debe existir.
w	Abrir para escritura. Se crea un archivo nuevo o se sobreescribe si ya existe.
a	Abrir para añadir datos al final. Si el archivo no existe, se crea.
rb	Abrir para lectura binaria.
wb	Abrir para escritura binaria.
ab	Abrir para añadir datos binarios.
r+	Abrir para lectura/escritura. El archivo debe existir.

Acceso secuencial – Abrir en C
Modos de apertura

w+	Crear archivo para lectura/escritura. se crea un archivo nuevo o se sobreescribe si ya existe.
a+	Añadir, lectura y escritura. El cursor se situa al final del archivo. Si el archivo no existe, se crea.
r+b	Abre para lectura/escritura binaria.
w+b	Crea para lectura/escritura binaria.
a+b	Abre o crea para añadir datos binarios.

Acceso secuencial - Cerrar

Una vez que se ha terminado de utilizar el archivo debe dejarse el mismo preparado para que otro programa lo pueda utilizar, para ello se hace uso de la primitiva **Cerrar**.

Cerrar(nomArch)

nomArch es el nombre **interno** del archivo.

Acceso secuencial – Cerrar en C

- fclose
  - Cierra un archivo previamente abierto, liberando los recursos asociados al programa.
  - `int fclose(FILE* f);`

Éxito de la operación (0 en caso de éxito)

Descriptor del archivo a cerrar (**interno).**

Manejo de errores en C

- En C, muchas funciones modifican una variable global cuando ocurre un error.
- Esta variable puede ser consultada para saber más acerca del error.
- La variable global se llama "`errno`".
  - Se define en `<errno.h>`
- La función "`strerror(int e)`" entrega una descripcion de un código de error.
  - Se define en `<string.h>`

Acceso secuencial
Ejemplo de apertura y cierre

```
#include <stdio.h>

int main(){

    FILE* archivo;
    archivo = fopen("test.txt","r");
    if(archivo!=NULL){
      printf("Apetura exitosa!\n");
      if (fclose(archivo)!=0) {
        printf("No se ha podido cerrar!\n");
      }
    }else
        printf("Error al abrir!\n");
}
```


Manejo de errores
Ejemplo

```
#include <stdio.h>

int main(){
    FILE* archivo;
    int errno;
    archivo = fopen("test.txt","r");
    if(archivo!=NULL){
      printf("Apetura exitosa!\n");
      if (fclose(archivo)!=0) {
        printf("No se ha podido cerrar!\n",strerror(errno));
      }
    }else
      printf("Error al abrir: %s\n",strerror(errno));
}
```


Acceso secuencial - Fin del Archivo

Para determinar el fin del archivo se utiliza la función **EOF(nomArch),** que significa **end of file** (fin de archivo).

A esta función se le pasa como parámetro el nombre interno del archivo **EOF(f)**.

En C se llama **feof**. C también posee una constante llamada **EOF** que es el retorno que envían distintas funciones de manejo de archivos al llegar a un final de archivo y no existir más datos.

- feof
  - Devuelve verdadero si se ha alcanzado el fin de archivo
  - `int feof(FILE* f);`

1: Fin de archivo
0: otro caso

Descriptor de archivo

Acceso secuencial - Leer

Leer(NomArch, variable)

nomArch es el nombre interno del archivo.

variable: identificador donde se almacena el elemento que se lee del archivo.

Leer es una primitiva que lee un elemento del archivo y avanza al siguiente elemento.

El uso repetido de la primitiva Leer permite ir extrayendo los datos de un archivo.

Acceso secuencial – archivos de texto

Ejemplo de lectura de datos de un archivo de texto, línea por línea, usando Leer

```
Algoritmo LeerArchivoDeTexto
Lexico
 f ∈ ARCHIVO de Texto
 cad ∈ Cadena
Inicio
  Abrir("test.txt",f,l)
  mientras not(EOF(f)) hacer
    Leer(f,cad)
    Salida:cad
  fmientras
  Cerrar (f)
Fin
```


Acceso secuencial – archivos de texto

- fgets
  - Lee desde un archivo abierto para lectura hasta un largo fijo o el fin de línea.

```
- fgets(char* cadena, int longitud, FILE* f);
```

Arreglo de caracteres donde guardar la cadena leída

Cantidad máxima de caracteres a leer

Descriptor de archivo

Acceso secuencial – archivos de texto

Ejemplo de lectura de datos de un archivo de texto, línea por línea, usando fgets

```
int main(){
  char cadena[128];
  FILE* archivo;
  archivo = fopen("test.txt","r");
  if(archivo!=NULL){
      while(!feof(archivo)){
        fgets(cadena,128,archivo);
        printf("%s",cadena);
      }
      fclose(archivo);
  }
}
```

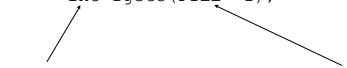

Acceso secuencial – archivos de texto

Ejemplo de lectura de datos de un archivo de texto, carácter por carácter, usando Leer

```
Algoritmo LeerArchivoDeTexto2
Lexico
 f ∈ ARCHIVO de Texto
 car ∈ Caracter
Inicio
  Abrir("test.txt",f,l)
  mientras not(EOF(f)) hacer
    mientras not(EOL(f)) hacer
      Leer(f,car)
    fmientras
  fmientras
  Cerrar (f)
Fin
```

Acceso secuencial – archivos de texto

- fgetc
  - Lee un carácter desde un archivo abierto para lectura.
  - `int fgetc(FILE* f);`



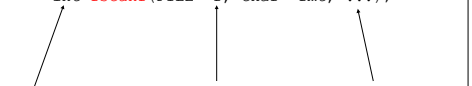
Acceso secuencial – archivos de texto

Ejemplo de lectura de datos de un archivo de texto, carácter por carácter, usando fgetc

```
int main(){
  char car;
  FILE* archivo;
  archivo = fopen("test.txt","r");
  if(archivo!=NULL){
      while(!feof(archivo)){
        car = fgetc(archivo)
        printf("%c",car);
      }
      fclose(archivo);
  }
}
```


Acceso secuencial – archivos de texto

- Existe una versión de scanf para archivos
- `int fscanf(FILE* f, char* fmt, ...);`



Acceso secuencial – archivos de texto

```
int main(){
  char c;
  FILE* archivo;
  archivo = fopen("test.txt","r");
  if(archivo!=NULL){
    while( fscanf(archivo,"%c",&c)==1 )
       printf("%c",c);
  fclose(archivo);
  }
}
```

Acceso secuencial – archivos de texto

Cada función de lectura (fgetc, fgets, fscanf) tiene su pareja para la escritura (fputc, fputs, fprintf)

- `int fputc(int c , FILE* f);`
  - EOF en caso de error
- `int fputs(char* cadena, FILE* f);`
  - EOF en caso de error
- `int fprintf(FILE* f, char* fmt, ...);`
  - Devuelve el numero de transformaciones realizadas con éxito.

Acceso secuencial - Escribir

Escribir(NomArch, variable)

nomArch es el nombre interno del archivo.

variable: identificador donde se almacena el elemento que se desea escribir en el archivo.

Escribir es una primitiva que escribe un elemento del archivo y avanza a la siguiente posición del archivo.

El uso repetido de la primitiva Escribir permite ir almacenando datos de un archivo.
El **Escribir** escribe donde esté posicionado el cabezal. Hay que tener cuidado de no sobreescribir datos existentes.

Acceso secuencial – archivos de texto

Ejemplo de escritura de datos de un archivo de texto, usando Escribir

```
Algoritmo EscribirArchivoDeTexto
Lexico
 f ∈ ARCHIVO de Texto
 c ∈ Caracter
Inicio
  Abrir("test.txt",f,a)
  para (i←1, i<=20, i←i+1) hacer
     Entrada:c
     Escribir(f,c)
  fpara
  Cerrar (f)
Fin
```


Acceso secuencial – archivos de texto

Ejemplo de escritura de datos de un archivo de texto, usando Escribir

```
Algoritmo EscribirArchivoDeTexto
Lexico
 f ∈ ARCHIVO de Texto
 c ∈ Caracter
Inicio
  Abrir("test.txt",f,r)
  para (i←1, i<=20, i←i+1) hacer
     Entrada:c
     Escribir(f,c)
  fpara
  Cerrar (f)
Fin
```

Acceso secuencial – archivos de texto

Ejemplo de escritura de datos de un archivo de texto, usando Escribir

```
Algoritmo EscribirArchivoDeTexto
Lexico
 f ∈ ARCHIVO de Texto
 c ∈ Caracter
Inicio
  Abrir("test.txt",f,e)
  para (i←1, i<=20, i←i+1) hacer
    Entrada:c
    Escribir(f,c)
  fpara
  Cerrar (f)
Fin
```

Acceso secuencial – archivos de texto

Ejemplo de escritura de datos de un archivo de texto, usando Escribir

```
Algoritmo EscribirArchivoDeTextoDesdeOtroArchivo
Lexico
 f, g ∈ ARCHIVO de Texto
 c ∈ Caracter
Inicio
  Abrir("test.txt",f,l)
  Abrir("test2.txt",g,e)
  mientras not(EOF(f)) hacer
    Leer(f,c)
    Escribir(g,c)
  fmientras
  Cerrar (f)
Fin
```


Acceso secuencial – archivos de texto

Ejemplo de escritura de datos de un archivo de texto

```
archivo = fopen("test.txt","r");
archivo2 = fopen("test2.txt","w+");
do{
  c = fgetc(archivo);
  if(c!=EOF){
  fputc(c,archivo2);
  printf("%c",c);
  }
}while(c!=EOF);
```


Acceso secuencial – archivos de texto

Ejemplo de escritura de datos de un archivo de texto

```
do{
   res=fgets(cadena,128,archivo);
   if(res!=NULL){
     printf("%s",cadena);
     fputs(cadena,archivo2);
}while(res!=NULL);
```

Acceso secuencial – archivos de texto

Ejemplo de escritura de datos de un archivo de texto

```
while( fscanf(archivo,"%c",&c)==1 ){
   printf("%c",c);
   fprintf(archivo2,"%c",c);
}
```


Vaciando buffers

Un buffer es un área de almacenamiento temporal para datos leídos o escritos en un archivo. Estos buffers retienen datos en tránsito desde y hacia al archivo y tienen la finalidad de hacer más eficiente las operaciones de entrada/salida. El uso de buffers hace que no refleje inmediatamente los cambios en los archivos, sino hasta que se "vacía" el buffer.
Para ello se utiliza la función fflush. Los buffers también se vacían cuando se cierra el archivo.

- fflush
  - Vacía el buffer de escritura. Fuerza la salida del buffer hacia el disco.
  - `int fflush(FILE* f);`

Acceso secuencial

Los archivos con tipo nos permiten almacenar información estructurada en forma de registro.

En el caso de la **lectura**, la información es copiada automáticamente, campo por campo, en la variable definida en caso de uso del **Leer.**

En el caso de la **escritura**, la información debe estar cargada en la variable utilizada por el **Escribir**, antes de ser escritos en el archivo.

Acceso secuencial – archivos con tipo

Lectura de datos

```
Algoritmo LecturaDeArchivoBinario
Lexico
 TAlumno = <nombre ∈ Cadena, edad ∈ Z>
 f ∈ ARCHIVO de TAlumno
 alumno ∈ TAlumno
Inicio
  Abrir("alumnos.dat",f,l)
  mientras not(EOF(f)) hacer
    Leer(f,alumno)
    Salida:alumno
    // Salida: alumno.nombre alumno.edad
  fmientras
  Cerrar(f)
Fin
```

Acceso secuencial – archivos con tipo

Lectura de datos: fread

- size_t **fread**(void *ptr, size_t size, size_t nmemb, FILE *stream)

Lee desde de un archivo uno o varios registros de la misma longitud

```
typedef struct { char nombre[30]; int edad;} TAlumno;
TAlumno alumno;
void main() {
 FILE *f;
 f = fopen("alumnos.dat","r");
 while ((!feof(f)) {
  fread(&alumno,sizeof(alumno),1,f);
  printf(" NOMBRE = %s ",alumno.nombre);
  printf(" EDAD = %d ",alumno.edad);
  printf("\n");
 };
 fclose(f);
```


Acceso secuencial – archivos con tipo

Escritura de datos

```
Algoritmo EscrituraDeArchivoBinario
Lexico
 TPersona = <nombre ∈ Cadena, edad ∈ Z>
 f ∈ ARCHIVO de TPersona
 reg ∈ TPersona
Inicio
  Abrir("personas.dat",f,a)
  Entrada:reg.nombre
  Entrada:reg.edad
  Escribir(f,reg)
  Cerrar(f)
Fin
```


Acceso secuencial – archivos con tipo

Escritura de datos

```
Algoritmo EscrituraDeArchivoBinario2
Lexico
 TPersona = <nombre ∈ Cadena, edad ∈ Z>
 f ∈ ARCHIVO de TPersona
 reg ∈ Tpersona
 i ∈ Z
Inicio
  Abrir("personas.dat",f,a)
  para (i←1, i<=20, i←i+1) hacer
    Entrada:reg.nombre
    Entrada:reg.edad
    Escribir(f,reg)
  fpara
  Cerrar(f)
Fin
```


Acceso secuencial – archivos con tipo

Escritura de datos

```
Algoritmo EscrituraDeArchivoBinario2bis
Lexico
 TPersona = <nombre ∈ Cadena, edad ∈ Z>
 f ∈ ARCHIVO de TPersona
 reg ∈ Tpersona
 i ∈ Z
Inicio
  Abrir("personas.dat",f,r)
  para (i←1, i<=20, i←i+1) hacer
    Entrada:reg.nombre
    Entrada:reg.edad
    Escribir(f,reg)
  fpara
  Cerrar(f)
Fin
```

Acceso secuencial – archivos con tipo

Escritura de datos: fwrite

- size_t **fwrite**(void *ptr, size_t size, size_t nmemb, FILE *stream)

Escribe en un fichero uno o varios registros de la misma longitud

```
typedef struct { char nombre[30]; int edad;} TPersona;
TPersona reg;
FILE *f;
void main() {
 f = fopen("personas.dat","a+");
 printf("ingrese nombre:");
 gets(reg.nombre);
 printf("ingrese edad:");
 scanf("%d",&reg.edad);
 fwrite(&reg,sizeof(reg),1,f);
 fclose(f);
 getchar();
```


Acceso secuencial - Ejemplos

1) Construya un algoritmo que dado un archivo de texto (sopaDeLetras.txt), muestre por pantalla todas las vocales contenidas en dicho archivo y la cantidad de cada una.

2) Construya un algoritmo que dado un archivo de relojes (relojes.dat), informe el modelo, marca y precio de los relojes cuyo costo está entre 50 y 100 pesos. Defina el tipo reloj con 3 campos: modelo, marca y precio.

2a) Agregue al algoritmo anterior una acción para cargar el archivo de relojes desde 0, es decir, sin importar lo que contenía interiormente relojes.dat.

2b) Agregue al algoritmo anterior una acción para cargar el archivo de relojes, manteniendo los datos contenidos anteriormente en relojes.dat.

Nota: Recuerde que para agregar relojes al archivo debe posicionarse al final del mismo.

2c) Modifique el tipo reloj, y el algoritmo donde corresponda, para agregar un nuevo campo llamado **pulsera** que puede contener los siguientes valores: metal, goma, plástica, o cuero.

Acceso Directo en archivos con tipo

También existe la posibilidad de acceder a los registros de los archivos de manera directa. Para ello existen las siguientes operaciones:

- **PosicionActual(NomArch):** Devuelve la posición actual del dispositivo de lectura/escritura en el archivo, da el número del registro en forma de un numero entero. *Los registros comienzan desde la posición 0, es decir que en esa posición está el primer registro.*
  En C es ftell.

- **TamañoArchivo(NomArch):** Devuelve el tamaño actual del archivo es decir el número de registros que contiene.
  En C es no existe.

Acceso Directo

- **irPos(NomArch, numReg):** Sitúa el dispositivo de lectura/escritura sobre el archivo NomArch en la posición del registro numReg.

  irPos(NomArch, 0) se posiciona en el primer registro,

  irPos(NomArch, 1) se posiciona en el segundo registro,

  irPos(NomArch, 2) se posiciona en el tercer registro,

  irPos(NomArch,TamañoArchivo(NomArch)-1) en el último.

  irPos(NomArch,TamañoArchivo(NomArch)) al final.

En C es fseek.

Atención! Tanto en Notación Algorítmica como en lenguaje C, se considera al primer registro del archivo como en la posición cero (0).

Acceso Directo

- long int **ftell**(FILE *fichero): devuelve la posición actual del cursor de lectura/escritura, o -1 si hay algún error.
- int **fseek**(FILE *fichero, long int desplazamiento, int origen): situa el cursor del archivo para leer o escribir en el lugar deseado. Retorna cero si la función tuvo éxito, y un valor distinto de cero si hubo algún error.

El parámetro origen:

- SEEK_SET el desplazamiento se cuenta desde el principio del archivo. **El primer byte del archivo tiene un desplazamiento cero**.
- SEEK_CUR el desplazamiento se cuenta desde la posición actual del cursor.
- SEEK_END el desplazamiento se cuenta desde el final del archivo.

- void **rewind**(FILE *f): sitúa el cursor de lectura/escritura al principio del archivo.

Acceso directo – archivos con tipo
Como determinar el tamaño de un archivo

int tamanio; // declaro la variable que recibira el tamanio.

FILE* arch; // declaro un puntero de tipo FILE.

arch=**fopen**("miArchivo.dat", "rb"); // abro el archivo de solo lectura.

fseek(arch, SEEK_END); // me ubico en el final del archivo.

tamanio=**ftell**(arch); // obtengo su tamaño en BYTES.

fclose(arch); // cierro el archivo.

Acceso secuencial – archivos con tipo
Acceso directo

```
Algoritmo EscrituraDeArchivoBinario3
Lexico
 TPersona = <nombre ∈ Cadena, edad ∈ Z>
 f ∈ ARCHIVO de TPersona
reg ∈ TPersona
Inicio
  Abrir("personas.dat",f,a)
  Entrada:reg.nombre reg.edad
  Escribir(f,reg)
  irPos(f,TamañoArchivo(f) div 2)
  Entrada:reg.nombre reg.edad
  Escribir(f,reg)
 Cerrar(f)
Fin
```


Acceso secuencial – archivos con tipo
Acceso directo

```
Algoritmo EscrituraDeArchivoBinario4
Lexico
 TPersona = <nombre ∈ Cadena, edad ∈ Z>
 f ∈ ARCHIVO de TPersona
reg ∈ TPersona
Inicio
  Abrir("personas.dat",f,a)
  Entrada:reg.nombre reg.edad
  Escribir(f,reg)
  irPos(f,TamañoArchivo(f)-1)
  Entrada:reg.nombre
  Escribir(f,reg)
 Cerrar(f)
Fin
```

Borrar en un archivo con tipo

Existen varias alternativas:

1. Borrar de manera virtual o lógica el registro: Agregar en el registro un campo lógico y adoptar la convención: si ese campo está en verdadero significa que el registro ha sido borrado o viceversa.

2. Eliminar físicamente el registro: Para implementar esta solución no hay otra forma que crear un archivo nuevo (auxiliar) con un nombre distinto con todos los registros que se quieren conservar y no incluir el registro que se borra. Luego borrar el archivo original y renombrar el auxiliar con el nombre original.

3. Combinar las dos anteriores: A los fines del tratamiento de los registros se implementa el campo de borrado y para la eliminación física se implementa la opción 2 antes de cerrar el programa.