"Clase 4: ARCHIVOS

Los archivos permiten el almacenamiento en memoria secundaria de grandes volúmenes de datos en un medio que prevalece después que termina de correr un programa y se apaga la computadora.
El archivo no determina a priori su dimensión física (mientras haya espacio en disco, pueden agregarse componentes).

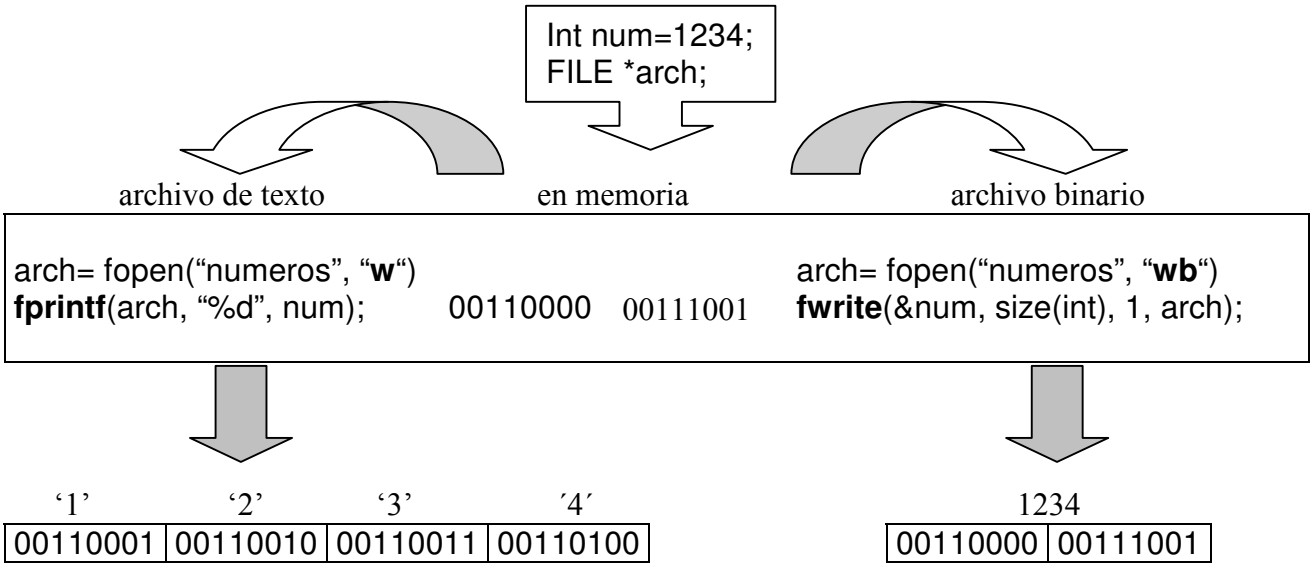
Las principales características son:
- Independencia de la información respecto a los programas
- Distintos programas pueden acceder a un mismo archivo
- La información es permanente
- Gran capacidad de almacenamiento

Los archivos se clasifican según :

☞ LA FORMA EN LA QUE ALMACENA LA INFORMACION

El archivo de **texto**, consiste en una colección de caracteres (puede estar dividido en líneas), puede ser grabado, inspeccionado o modificado mediante el uso de un editor, también un programa Pascal o C puede escribir o leer del archivo. Los valores numéricos son convertidos en carácter al grabarlos y reconvertidos en la lectura. Se abre para operaciones de lectura o escritura. Son estandar, por lo tanto interpretables por cualquier programa.

El archivo **binario** almacena información a imagen de memoria, son creados por un programa (Pascal, C, etc.) que graba componentes de longitud fija o variable, de tipo simple o estructurado. Los valores almacenados deben recuperarse por programas que utilicen el mismo tipo de componentes (en el mismo lenguaje). Salvo la información alfanumérica, no se pueden interpretar con un procesador de texto. No son estandar, por lo tanto no trasladables a otros ámbitos.



☞ LA ORGANIZACION:

Secuencial para procesos secuenciales sobre todos o un gran porcentaje de los datos que almacena el archivo.
El orden o secuencia en el que se acceden los registros coincide con la secuencia física (items contiguos físicamente)
Para acceder a un determinado dato deben haberse leído todos los previos a él.

La actualización del archivo (incorporación, eliminación o modificación de sus elementos) requiere crear uno nuevo.
El archivo se procesa para lectura o escritura, si se está leyendo no se graba y viceversa.
Es de gran utilidad en archivos de gran tamaño que no requieren actualizarse con excesiva frecuencia o prontitud.

Indexada, permite el acceso a registros en forma individual, sin necesidad de buscar en secuencia.
Permite leer y/o escribir en cualquier lugar

Secuencial Indexada, proporciona el tipo de acceso "casi" directo y también el secuencial

- **EL TIPO DE ACCESO:**

secuencial: el acceso es a partir del comienzo del archivo en forma físicamente progresiva

aleatorio: permiten acceder (en forma directa o indirecta) a cualquier lugar del archivo

- **EL FLUJO DE DATOS:**

De entrada: los datos se leen por el programa desde el archivo.

De salida: los datos se escriben por el programa hacia el archivo.

De entrada/salida: los datos pueden se escritos o leídos.

Utilizaremos los archivos binarios en forma secuencial para entrada de datos y en algunos casos salida de información.
A continuación se desarrollan algunos ejemplos sobre archivos binarios que almacenan structs.

Ej1.- Cargar desde teclado un archivo y listar su contenido

```
#include <stdio.h>
struct persona{
char nombre[25];
int  edad;
float  peso;
float  altura; };

void main(){
/*Variables*/
struct persona  r, v[20];
FILE *a;
int n;

/*Cargar desde teclado un archivo*/
a=fopen("archper","wb");
scanf("%d",&n);
```

```
while(n--) {
  fflush(stdin);
  gets(r.nombre);
  scanf("%d %f %f",&r.edad, &r.peso, &r.altura);
  fwrite(&r, sizeof(struct persona), 1,a);
      }
fclose(a);

/*Mostrar por pantalla el contenido del archivo*/
a=fopen("archper","rb");
fread(&r,sizeof(struct persona), 1,a);
while(!feof(a)) {
  puts(r.nombre);
  printf("%d %f %f\n",r.edad, r.peso, r.altura );
  fread(&r,sizeof(struct persona), 1,a);
  }
fclose(a);
}
```

Para leer de teclado debe hacerlo campo a campo

graba la componente struct completa

Ej2.-Leer un vector de registros (tabla) desde un archivo

```
#include <stdio.h>
struct persona{
char nombre[25];
int  edad;
float  peso;
float  altura; };
void main(){
/*Variables*/
struct persona   v[10];
FILE *a;
int n;
n = 0;
a=fopen("archper","rb");
/*se lee una componente completa sobre un elemento del arreglo,
no es necesaria una variable auxiliar*/
fread(&v[0],sizeof(struct persona), 1, a);
while (! feof(a) && n<10){
    n++;
    fread(&v[n],sizeof(struct persona),1,a);
    }
fclose(a);
}
```

Ej3.- Considerar los siguientes archivos:

ARCHIVO-ALUMNOS	**ARCHIVO-CARRERAS**
-REGISTRO-ALUMNO .NUMEROdeMATRICULA (ANU8) .APELLIDOyNOMBRE (ANU25) .CODIGOdeCARRERA (ANU6) .AÑOdeINGRESO .PROMEDIO	.CODIGOdeCARRERA

Grabar los archivos ingresando la información desde teclado (no está ordenado por ningún ítem)

A partir de los archivos ALUMNOS y CARRERAS:

a. Calcular e imprimir total de alumnos por carrera. (son 30 carreras como máximo)
b. Informar a quien le correspondería ser el abanderado, el 1ro y el 2do escolta, recorriendo una sola vez el archivo. ¿cómo lo solucionaría si hay más de tres alumnos con promedio máximo y se quieren conservar sus datos?

¿Se podrían realizar ambos procesos simultáneamente y por lo tanto recorriendo una sola vez el archivo?

Ejemplo Archivo Alumnos

Matricula	Apellido	Carrera	Año	Promedio
384	Perez	Sist01	2000	4
251	Gomez	Mark03	2002	8
932	Ruiz	Mark03	2004	5
125	Garcia	Dise01	2001	6
343	Gonzalez	Sist01	2001	8
127	Fernandez	Dise01	2002	9
834	Gimenez	Sist01	2000	10
915	Alvarez	Sist01	2004	7

Resultado

Carrera Cant.. Alumnos
Dise01 2
Mark03 2
Sist01 4

Abanderado: Gimenez 10
Escolta1: Fernandez 9
Escolta2: Gomez 8 (tomamos el más antiguo, por eso no va Gonzalez)

Solución

Para contabilizar la cantidad de alumnos por carrera se utiliza un arreglo ***v*** (de enteros), cada componente corresponde a una carrera. Cada registro ***r***, leído del archivo ***alumn***, representa a un alumno y determina con su campo **r*.codigo***, la posición de la componente del arreglo **v** a ser incrementada . Dicha posición es el resultado de la búsqueda de la carrera ***r.codigo*** en la tabla ***tabla*** (contiene códigos de carrera) que se carga, al comienzo, a partir del archivo ***carre***. El arreglo ***v*** es paralelo a dicha tabla. Los datos de los tres máximos se almacenan en un arreglo de tres registros.

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
struct alumno{
      char matri[4], apel[10], codigo[7];
       int anio;
       float prom;};

char   tabla[30][7]; /*tabla con los codigos de las carreras*/
int n, pos, v[30]={0}; /*acumula alumnos por carrera*/
FILE *alum;
struct alumno r, mejor[3];
/*carga los codigos en la tabla desde el archivo*/
void cargatabla(char tabla[][7], int *n);
int obtieneindice(char tabla[][7], char cod[]);
void TresMax(struct alumno r, struct alumno mejor[]);
void EscribirMax(struct alumno mejor[]);
void EscribirTotales(char tabla[][7],int v[],int n);
```

```
void main(){
   clrscr();
  /*carga los codigos en la tabla desde el archivo*/
   cargatabla(tabla, &n);
   /*inicia los mejores promedios*/
   mejor[0].prom= mejor[1].prom=mejor[2].prom= -1;
   alum= fopen("archalu","rb");
   fread(&r,sizeof(struct alumno), 1,alum);
   while (! feof(alum)) {
   /*posicion de la componente a ser incrementada
      pos = obtieneindice(tabla, r.codigo);
      v[pos]++;
      TresMax(r, mejor);
      fread(&r,sizeof(struct alumno), 1,alum);
 }
   EscribirMax(mejor);
   EscribirTotales(tabla,v,n);
   fclose( alum );
 }
void cargatabla(char tabla[][7], int *n){
  FILE  *carre;
  *n=0;
  carre=fopen("archcarr","rb");
  fread(tabla[0],7, 1, carre);
  while (! feof(carre)) {
    (*n)++;
    fread(tabla[*n],7,1,carre);
    }
fclose(carre); }

int obtieneindice(char tabla[][7], char cod[]){
   int i=0;
   /*supone que existe el codigo en la tabla*/
   while (strcmp(cod,tabla[i]))
  i++;
   return i; }

void TresMax(struct alumno r, struct alumno mejor[]){
    if (r.prom>mejor[2].prom)
   if (r.prom>mejor[1].prom)
       if (r.prom>mejor[0].prom){
       mejor[2] = mejor[1];
       mejor[1] = mejor[0];
       mejor[0] = r;}
       else  {
       mejor[2] = mejor[1];
       mejor[1] = r;}
    else
       mejor[2] = r;
       }

void EscribirTotales(char tabla[][7],int v[],int n){
   printf("  Codigo de Carrera  _  Total de Alumnos\n");
   for(int i= 0; i< n; i++)
```

```
     printf("%10s  %20d  \n",tabla[i], v[i]);
   }
void EscribirMax(struct alumno mejor[]){
   printf("Abanderado y promedio: %s  %.2f \n ",mejor[0].apel,
          mejor[0].prom);
   printf("Primer Escolta y promedio: %s  %.2f \n ",mejor[1].apel,
          mejor[1].prom);
   printf("Segundo Escolta y promedio: %s  %.2f \n ", mejor[2].apel,
          mejor[2].prom);
   }
```

Ej4.-Una empresa almacena en distintos depósitos (son 30 como máximo) la existencia de sus artículos y mantiene en un archivo la siguiente información:

ARCHIVO_STOCK
- -REGISTRO_STOCK
  - .DEPOSITO (ANU6)
  - .CODIGOdeARTICULO (ANU3)
  - .CANTIDADdeUNIDADES

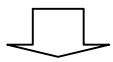
A partir del archivo STOCK, generar el siguiente listado:

INFORME de STOCK

Depósito	Cantidad de Artículos	Stock Total de unidades
xxxxxxx	9999	9999999

Solución
Es necesario una tabla con los códigos de depósito para trabajar paralelamente a los arreglos ***vcant*** y ***vstock*** que acumulan cantidad de artículos y unidades por depósito respectivamente. Como dichos códigos no se encuentran grabados en un archivo independiente, la tabla se va cargando a partir de los códigos del archivo ***arch*** (Archivo_Stock). Por cada registro leído debe convertir el código de depósito en índice, para ello busca en la tabla (que comienza vacía) dicho código de depósito y si no lo encuentra lo inserta.

Archivo Stock		
Deposito	Código de artículo	Cantidad de unidades
MDP001	A22	10
BSAS02	B11	15
MDP001	B11	20
MEND03	C45	5
MEND03	A22	10
MDP001	D55	30



Informe de Stock

Deposito	Cantidad de artículos	Stock total de unidades
MDP001	3	60
BSAS02	1	15
MEND03	2	15

vcant vstock

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
```

```
struct stock  {
     char depos[7];
     char art[4];
     int cant;} ;
void buscainserta(char tabla[][7], int *n, char depos[7], int *pos);
void Listado(char tabla[][7],int vcant[], int vstock[], int n);

void main(){
   struct stock r;
   FILE *archstock;
   char tabla[30][7];
   int vcant[30]={0}, vstock[30]={0}, n=0, pos;
  clrscr();
  archstock= fopen("archstock","rb");
  fread(&r,sizeof(struct stock), 1,archstock);

  while (! feof(archstock)) {
    //posicion de la componente a ser incrementada
    buscainserta(tabla, &n, r.depos, &pos);
    vcant[pos]++;   vstock[pos] += r.cant;
    fread(&r,sizeof(struct stock), 1,archstock);
 }
 Listado(tabla, vcant, vstock, n);
 fclose( archstock);
 }

void buscainserta(char tabla[][7], int *n, char depos[7], int *pos ){
  *pos=0;
  while (*pos<*n && strcmp(depos, tabla[*pos]))
     (*pos)++;
  if  (*pos ==*n){         /*insertar al final, indice n */
    strcpy(tabla[*n], depos);
    (*n)++;}  /*copia en tabla y actualiza n*/
  }
void Listado(char tabla[][7],int vcant[], int vstock[], int n){
  printf("Codigo de Dep  - Cant de Art  -  Stock");
  for(int i= 0; i< n; i++)
  printf("%10s  %10d  %10d",tabla[i], vcant[i],vstock[i]);
  }
```

Ej5.-Una compañía de electricidad factura mensualmente a sus clientes de acuerdo con la siguiente tarifa: hasta 500 kw a 10 centavos el kw, hasta 1000 kw a 18 ctvos. y consumo mayor a 1000 kw, 24 ctvos.
Se pretende emitir la factura correspondiente a cada abonado, calculando además cantidad de clientes, monto total facturado y gasto promedio por cliente.
Para ello se cuenta con el siguiente archivo:
ARCHIVO-CLIENTES
 -REG-CLIENTE
    .CODIGO-CLIENTE (ANU5)
    .DIRECCION
.   CONSUMO-KW

Por fin de proceso, se desea saber cuáles fueron los 5 clientes de mayor consumo indicando además la cantidad de kw consumidos.

Solución

Se define un arreglo de structs para los 5 clientes de mayor consumo. Una función calcula el precio del KW según el consumo

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
   struct direccion{
     char calle[10];
     char nro [4];
     };
   struct cliente  {
     char codigo[5];
     struct direccion direc;
     int cons;} ;
   struct maycons  {
     char codigo[5];
     int cons;} ;

  void tarifa(int cons,float *precio);
  void inserta (struct maycons v[], struct cliente r);
  void listado(struct maycons v[]);

  void main(){
  FILE * aclientes;
  struct cliente r;
  struct maycons v[5]={{"",0}};
  int cantcli = 0, i;
  float imptot = 0, imp, precio;

  clrscr();
  aclientes =  fopen("clientes","rb");
  printf("Listado de clientes\n");
  printf("C¢digo     Direcci¢n    Consumo    Importe\n");
  fread(&r,sizeof(struct cliente), 1, aclientes);
  while (! feof(aclientes)) {
    tarifa(r.cons, &precio);
    imp= precio * r.cons;
    printf("%6s  %10s %3s  %6d   %8.2f\n",r.codigo, r.direc.calle,
           r.direc.nro, r.cons, imp);
    imptot += imp;        cantcli ++;
    if  (r.cons >v[4].cons)
      inserta (v, r);
    fread(&r,sizeof(struct cliente), 1, aclientes);
    }
  printf("Cantidad de clientes= %5d\n",cantcli);
  printf("Total Facturado = %.2f\n", imptot);
  printf("Gasto promedio por cliente= %.2f\n\n", imptot/cantcli );
  listado(v);
  fclose(aclientes);}
```

```
void tarifa(int cons,float *precio){
  if  (cons < 500)
     *precio = 0.10;
  else
    if  (cons < 1000)
      *precio = 0.18;
    else
      *precio = 0.24;
  };
void inserta (struct maycons v[], struct cliente r){
  int i=   3;
  while( i>=0 && v[i].cons < r.cons){
    v[i+1] =v[i];
     i--;
    };
  strcpy(v[i+1].codigo,r.codigo);
    v[i+1].cons = r.cons;
  }
void  listado(struct maycons v[]){
  printf("Clientes maximo consumo \n Codigo   Consumo\n");
  for(int i=0; i<5; i++)
    printf("%8s  %6d\n", v[i].codigo,v[i].cons);
  }
```

Ej5b.- Modificar el algoritmo para que las condiciones de venta sean variables.
Se incorpora una tabla donde cada elemento almacene un rango y la respectiva tarifa, de forma de permitir variación en los topes y precios como así también en la cantidad de categorías a considerar. La tabla podria declararse constante o cargarse desde un archivo al comienzo del proceso, dicha tabla sería parámetro de entrada de la función *Tarifa.*

```
  struct categoria{
       float costo;
       nt tope;};

typedef struct categoria CATEGORIAS[5];

CATEGORIAS vcat={{0.1,500},{0.18,1000},{0.24, }};

void tarifa(CATEGORIAS vcat, int cons,float *precio);
.................
  tarifa(vcat,r.cons, &precio);
.................

void tarifa(CATEGORIAS vcat,int cons,float *precio){
  int i=0;
  while (i<2 && cons >vcat[i].tope)
       i++;
  *precio = vcat[i].costo;
  };
```

// podría cargarse la tabla desde un archivo y flexibilizar sus valores