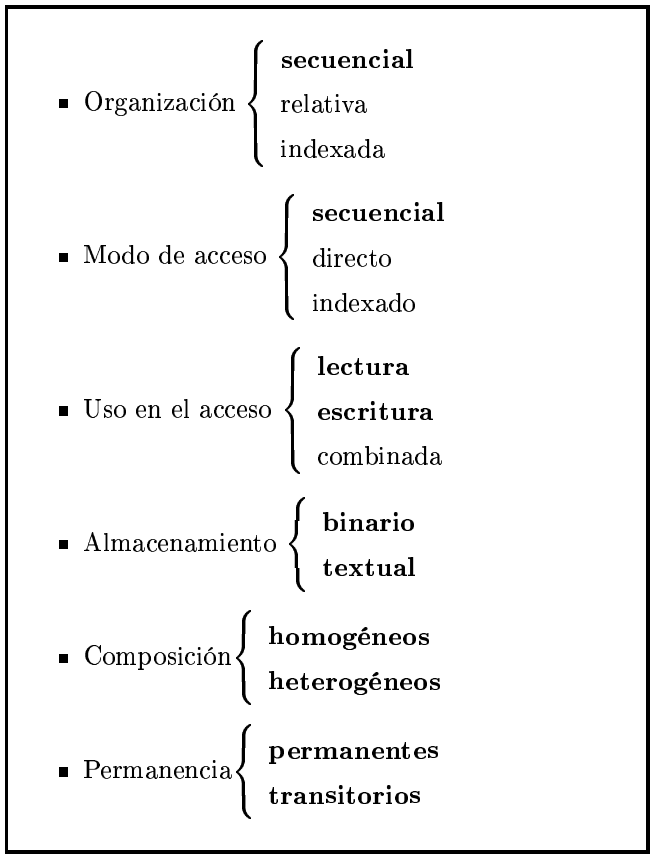
Slide 1

FICHEROS

Colección de datos almacenados en memoria externa

- Tamaño indeterminado (límite: dispositivo)
- Tamaño dinámico
- Pueden ser permanentes
- Acceso más lento

Slide 2


Organización
secuencial
relativa
indexada
Modo de acceso
secuencial
directo
indexado
Uso en el acceso
lectura
escritura
combinada
Almacenamiento
binario
textual
Composición
homogéneos
heterogéneos
Permanencia
permanentes
transitorios

Slide 3

Pascal: ficheros de datos

Secuencial, de acceso secuencial, binario, homogéneo.

Definición: file of <id-TBase>

```
type TFich = file of integer;
     TMat = array [1..10, -3..6] of real;
var f : Tfich;
    g : file of TMat;
    h : file of char;
```

Un fichero puede ser de lectura **o (exclusivo)** de escritura.

Operaciones:

1 Preparar, para lectura **o** escritura

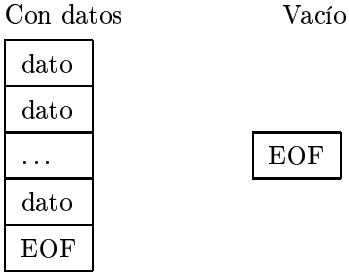
2 Leer el siguiente artículo $(0 - n$ veces)

ó 2 Escribir el siguiente artículo $(0 - n$ veces)

3 Cerrar el archivo

Slide 4

Composición de ficheros de datos:



Slide 5

Ficheros para escritura:

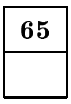
Preparación: **rewrite(**<id-fichero>)

```
rewrite(f);
```

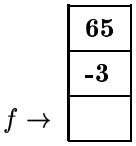
$f \rightarrow$ 

Escritura: **write(**<id-fichero>, <exp-TBase>)

```
write (f,65);
```

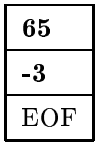
$f \rightarrow$ 


```
write (f,-3);
```

$f \rightarrow$ 

Cierre: **close**(<id-fichero>)

```
close(f)
```

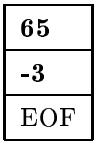


Slide 6

Ficheros para lectura:

Preparación: **reset(**<id-fichero>)

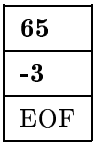
```
reset(f);
```

$f \rightarrow$ 

n 

Lectura: **read(**<id-fichero>, <id-var-TBase>)

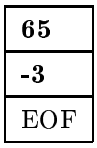
```
read (f, n)
```

$f \rightarrow$ 

n 

Cierre: **close**(<id-fichero>)

```
close(f)
```



Slide 7

Fichero para escritura: **rewrite**

- Se crea si no existe
- Se destruye el que hubiera si existía, para crear uno nuevo
- Cada nueva escritura se realiza al final (a continuación)

Fichero para lectura: **reset**

- Colocación en el primer dato del fichero
- Cada lectura se hace sobre el dato apuntado, y se avanza el visor
- Un nuevo **reset** vuelve a la primera posición
- No se puede leer el **EOF**
- La función **eof**(<id-fichero>) devuelve TRUE si en la ventana de lectura se encuentra sobre EOF y FALSE si no.

Slide 8

Lectura completa de un fichero de datos:

```
type tmatriz = ...
var f : file of tmatriz;
    m : tmatriz;
Begin
...
   reset (f);
   while not eof (f) do
         begin
             read (f, m);
             (* Tratar m *)
         end;
   close (f)
End.
```

Slide 9

Permanencia:

Asignación (temporal) de un "*nombre lógico*" (identificador de variable de tipo fichero) a un "*nombre físico*" (dependiente del sistema operativo).

En Turbo Pascal:

assign (<id-var-fichero>, <exp-string-nombre-físico>);

assign (f, 'mifich.dat');

Si es para lectura: debe existir antes de abrirlo (reset)

Si es para escritura: existirá después de terminar el programa

En ambos casos: parámetro del programa (como **input** y **output**)

Slide 10

```
program creaf (input, output, f);
var f : file of integer; n : integer; mas : char;
function mayus(c:char):char; ...
Begin
    assign (f, 'fichEnt.dat'); rewrite (f);
    repeat
        readln (input, n);
        write (f, n);
        repeat
            writeln (ouput, '¿Más? (S/N)');
            readln (input, mas); mas := mayus(mas);
        until mas in ['S', 'N'];
    until mas = 'N';
    close (f)
End.
```

Slide 11

```
program copiaPositivos (f, g);
var f, g : file of integer; n : integer;
Begin
    assign (f, 'fichEnt.dat'); reset (f);
    assign (g, 'fichPos.dat'); rewrite (g);
    while not eof (f) do
        begin ;
          read (f, n);
          if n>0 then write (g, n)
        end;
    close (f);
    close (g)
End.
```

Slide 12

Pascal: ficheros textuales

Secuencial organizado en líneas, acceso secuencial, (heterogéneo), de caracteres.

Se transforman los datos entre representación interna y textual.

Definición: **text**

Ejemplos: **input**, **output**

Lectura : **read** y **readln**:

read(<id-fich>, <lista de variables>);

readln(<id-fich>, <lista de variables>);

write(<id-fich>, <lista de expresiones>);

writeln (<id-fich>, <lista de expresiones>);

Slide 13

No se puede leer en el fin de línea: función **eoln**

```
program PasoAMay (f, g);
(* Pasa a mayúsculas todo un texto *)
var f, g : text;
    c : char;
function mayus (c: char): char;
...
Begin
    assign (f, 'UnTexto.txt'); reset (f);
    assign (g, 'EnMayus.txt'); rewrite (g);
    while not eof (f) do
        begin
            while not eoln (f) do
                begin
                    read (f, c);
                    write (g, mayus(c));
                end;
            readln (f); writeln (g);
        end;
    close (f); close (g)
End.
```

Entrada texto, por teclado (hasta fin de entrada: ctrl-Z en DOS, ctrl-D en Unix)

Salida número de mayúsculas que contiene y cuáles

Ejemplo 'Esta Es Una Prueba' —>3, EPU

```
program MayusUsadas (input, output);
type TMayus = 'A'..'Z';
var  conjunto : set of TMayus;
     c : char;

BEGIN
   conjunto := [ ];
   while not eof(input) do
         begin
             while not eoln(input) do
                   begin
                        read (input, c);
                        if c in ['A' ..'Z'] then
                             conjunto := conjunto + [c]
                   end;
             readln (input)
         end;
   write (output, 'Letras usadas: ');
   c := 'A';
   while conjunto <> [ ] do
         begin
             if c in conjunto then write (c);
             conjunto := conjunto - [c];
             if c<'Z' then c := succ(c)
         end;
   writeln (output)
END.
```

Entrada texto, en fichero de nombre `texto1.txt` del directorio por defecto

Salida número de mayúsculas que contiene y cuáles

```
program MayusUsadas (input, output, f);
type TMayus = 'A'..'Z';
var  f : text;
     conjunto : set of TMayus;
     c : char;

BEGIN
   assign (f, 'texto1.txt');
   reset (f);
   conjunto := [ ];
   while not eof(f) do
         begin
             while not eoln(f) do
                   begin
                        read (f, c);
                        if c in ['A' ..'Z'] then
                             conjunto := conjunto + [c]
                   end;
             readln (f)
         end;
   write (output, 'Letras usadas: ');
   c := 'A';
   while conjunto <> [ ] do
         begin
             if c in conjunto then write (c);
             conjunto := conjunto - [c];
             if c<'Z' then c := succ(c)
         end;
   writeln (output);
   close (f)
END.
```

Paso como parámetro: siempre por variable

Entrada
- opción para la salida: 'F' o 'P'
- si la opción fué 'F', un nombre para fichero, cadena de caracteres
- sucesión de enteros no nulos, por teclado. 0 significa el fin de la entrada.

Salida texto en fichero con los números leídos, a uno por línea, si la opción de entrada fué 'F', con el nombre especificado, o la misma sucesión de números por pantalla si la opción fué 'P'.

Ejemplo F numeros.txt 12 -12 0 —>fichero numeros.txt con dos líneas, cuyos contenidos son las cadenas 12 y -12 respectivamente.

Ejemplo P 12 -12 0 —>por pantalla las cadenas 12 y -12

```
program EscFichero (input, f, output);
 (* crea o no un fichero textual con enteros no nulos leídos de teclado *)

procedure escribeEntero (var f: text; n: integer);
begin
    write (f, n)
end;

var  f   : text;
     via : char;
     nombre : string[10];

     n : integer;

BEGIN
    repeat
       writeln ('Salida a pantalla (P) o crear fichero (F): ');
       readln (via)
    until via in ['P', 'F'];

    if via = 'F' then
       begin
           write ('Nombre: '); readln (nombre);
           assign (f, nombre); rewrite (f)
       end
    else
       begin
            assign (f, ''); rewrite (f)
       end;

    write ('Número: (fin = 0) '); readln (n);
    while n<>0 do
          begin
               escribeEntero (f, n); writeln (f);
               write ('Número: (fin = 0) '); readln (n);
          end;
    if via = 'F' then close (f)
END.
```

Entrada
- opción para la entrada posterior: 'F' o 'P'
- si la opción fué 'F', un nombre para fichero, cadena de caracteres.
- si la entrada fué 'P', sucesión de enteros no nulos, por teclado, hasta fin de fichero (un entero por línea).

Salida media de los enteros leídos

Observaciones Si la opción fué 'F', debe existir un fichero con el nombre proporcionado a continuación, debe ser de tipo `text` y contener un entero por línea (o más, pero sólo se considerará el primero de cada línea).

```
program MediaDeEnteros (input, f, output);
 (* calcula la media de una serie de enteros, de fichero o teclado *)
var  via : char;
     f   : text;
     nombre : string[40];

     suma: real;
     cantidad, n: integer;

BEGIN
    repeat
         write ('Entrada por fichero (F) o por teclado (T): ');
         readln (via)
    until via in ['F', 'f', 'T', 't'];

    if via in ['f', 'F'] then
       begin
         write ('Nombre del fichero (existente): ');
         readln (nombre);
         assign (f, nombre); reset (f)
       end
    else
       begin
         assign (f, ''); reset (f)
       end;

   cantidad := 0; suma := 0;
   while not eof (f) do
         begin
              readln (f, n);
              cantidad := cantidad +1;
              suma:= suma + n
         end;

   if cantidad > 0 then
        writeln ('Media = ', suma/cantidad)
   else
        writeln ('No hay datos');

   if via in ['f', 'F'] then close (f)
END.
```

Slide 14

Ejercicio: :

Entrada: Fichero de matrices de 10×10 enteros, `fichmat.dat`

Salida:

1. fichero de vectores de 10 enteros con las dos diagonales de cada matriz, `diag.dat`
2. texto (en fichero) con los mismos datos que el anterior, a dos diagonales por línea, `diag.txt`
3. número de matrices en las que las dos diagonales coincidan, por pantalla

Slide 15

```
program ej124 (fm, fv, t, output);
const TAM = 10;
type tmatriz = array [1..TAM, 1..TAM] of integer;
       tvector = array [1..TAM] of integer;
       tfm = file of tmatriz;
       tfv = file of tvector;
var fm : tfm ;
     fv : tfv ;
     t : text ;
     n : integer;
     m : tmatriz ;
     diag1, diag2 : tvector;
```

Slide 16

```
Begin
    assign (fm, 'fichmat.dat'); reset (fm);
    assign (fv, 'diag.dat'); rewrite (fv);
    assign (t, 'diag.txt'); rewrite (t);
    n := 0;
    while not eof (fm) do begin
        read (fm, m);
        extraediagonales (m, TAM, diag1, diag2);
        write (fv, diag1); write (fv, diag2);
        esctext (t, diag1, TAM); esctext (t, diag2, TAM); writeln (t);
        if coinciden (diag1, diag2, TAM) then n := n+1;
      end;
    writeln ('Número de matrices con diagonales iguales', n);
    close (fm); close (fv); close (t)
End.
```

Slide 17

```
procedure extraediagonales (var m: tmatriz; n : integer;
                                    var d1, d2: tvector);
var i : integer;
begin
       for i := 1 to n do
           begin
               d1[i]:= m[i, i];
               d2[i] := m[i, n+1-i];
           end
end;
procedure esctext (var t: text; var v: tvector; n: integer);
var i : integer;
begin
      for i := 1 to n do write (t, v[i]:5)
end;
```

Slide 18

```
function coinciden (var v1, v2: tvector; n: integer): boolean;
var i : integer; prov : boolean;
begin
    prov := true;(* de momento coinciden *)
    i := 1;
    while prov and (i<=n) do
            begin
                prov := prov and (v1[i]=v2[i]);
                i := i+1
            end;
    (* prov = false ó i>n *)
    coinciden := prov
end;
```

Slide 19

Ejercicio: localizar un entero en un fichero de enteros.

Entrada: un fichero de enteros `enteros.dat`
un entero `n`

Salida: si `n` está en el fichero: la posición que ocupa
si no : mensaje
(en ambos casos por pantalla)

Slide 20

```
program buscaEnF (input, f, output);
var f : file of integer; n : integer ;
    pos : integer;      i: integer;
Begin
    assign (f, 'enteros.dat'); reset (f);
    writeln ('Número '); readln (n); pos := 0;
    if not eof (f) do begin
                repeat read (f, i); pos := pos +1
                    until eof (f) or (i=n);
                if i=n then writeln ('Posición', pos)
                else writeln ('No está')
            end
    else writeln ('Fichero vacío')
    close (f)
End.
```

Slide 21

```
program buscaEnF (input, f, output);
var f : file of integer; n : integer ;
    pos : integer; i: integer; esta : boolean;
Begin
    assign (f, 'enteros.dat'); reset (f);
    writeln ('Número '); readln (n); pos := 0; esta ;= FALSE;
    while (not esta) and (not eof (f)) do begin
                        read (f, i); pos := pos +1;
                        esta := i=n
            end;
    if esta then writeln ('Posición', pos)
    else writeln ('No está');
    close (f)
End.
```

Slide 22

Añadir un elemento al final:

1. Copiar en un nuevo fichero los elementos del fichero original
2. Añadir el elemento nuevo

Borrado de un elemento:

1. Copiar en un nuevo fichero los elementos anteriores al elemento a borrar
2. Copiar en el nuevo fichero los elementos posteriores al elemento a borrar

La implantación suele proporcionar procedimientos para borrar y renombrar ficheros con arreglo a las normas del sistema operativo. En Turbo Pascal, en la unidad Dos.

Slide 23

Inserción de un elemento

1. Identificar un criterio que permita obtener el lugar de inserción
2. Copiar en un nuevo fichero todos los elementos del fichero original anteriores al lugar de inserción
3. Escribir en el nuevo fichero el elemento a insertar
4. Copiar en el nuevo fichero el resto de los elementos del fichero original

Comentarios generales:

- Los ficheros deben ser siempre pasados por variable
- Debe usarse un fichero cuando sea necesario
- No debe usarse un fichero cuando sea innecesario
- No debe copiarse el contenido total de un fichero a una variable de programa de otro tipo (`array` por ejemplo) prácticamente bajo ninguna circunstancia
- Los ficheros deben recorrerse el número de veces estrictamente necesario (o sea, el mínimo posible)
- La lectura de un archivo textual (`text`) se realiza bien como tal (por líneas, o sea cadenas de caracteres de un tamaño suficiente, y `readln (st)`, o carácter a carácter) o bien, sabiendo la disposición de los datos, con sentencias read o readln sobre variables adecuadas.
- La lectura de un fichero de tipo `file of T`, se hace elemento a elemento de tipo `T`.
- El nombre de un fichero puede ser variable, de tipo `string`
- Una organización muy frecuente es la de fichero de registros.