

1

# Fundamentos de Organización de Datos

Clase 2

# Agenda

## Algoritmia

- Básica
- Clásica

## Básica

- Agregar nuevos elementos

## Clásica

- Actualización
- Merge
- Corte de Control

# Archivos - Algorítmica Clásica

## Operaciones usuales a resolver utilizando archivos

- Agregar nuevos elementos
- Actualizar un archivo maestro con uno o varios archivos detalles
- Corte de control
- Merge de varios archivos

Los discutiremos en las siguientes transparencias

# Archivos → Ej 4 Agregar Datos a un archivo existente

4

- Se procesa un solo archivo
- Ya tiene información
- Se le incorporan datos nuevos
- El proceso muestra como se hace

```
Procedure agregar (Var Emp: Empleados);  
  var E: registro;  
  begin  
    reset( Emp );  
    seek( Emp, filesize(Emp) );  
    leer( E );  
    while E.nombre <> ' ' do begin  
      write( Emp, E );  
      leer( E );  
    end;  
    close( Emp );  
  end;
```

# Archivos → Actualización Maestro Detalle

Este problema involucra utilizar al mismo tiempo varios archivos de datos.

- Se denomina maestro al archivo que resume un determinado conjunto de datos.
- Se denomina detalle al que agrupa información que se utilizará para modificar el contenido del archivo maestro.
- En general
  - Un maestro
  - N detalles.

## Consideraciones del proceso (precondiciones)

- 1) • Ambos archivos (maestro y detalle) están ordenados por el mismo criterio
- 2) • En el archivo detalle solo aparecen empleados que existen en el archivo maestro
- 3) • Cada empleado del archivo maestro a lo sumo puede aparecer una vez en el archivo detalle

# Archivos → Ej 5 Actualizar Un Maestro con Un detalle

6

```
program actualizar;
type emp = record
    nombre: string[30];
    direccion: string[30];
    cht: integer;
end;
detalle = file of e_diario; { archivo que contiene la información diaria }
maestro = file of emp;      { archivo que contiene la información completa }

var
    regm: emp;  regd: e_diario;  mael: maestro; det1: detalle;
begin
    assign (mael, 'maestro');
    assign (det1, 'detalle');
    {proceso principal}
    reset (mael);  reset (det1);
    while (not eof(det1)) do begin
        read(mael, regm);
        read(det1, regd);
        while (regm.nombre <> regd.nombre) do
            read (mael, regm);
        regm.cht := regm.cht + regd.cht;
        seek (mael, filepos(mael)-1);
        write(mael, regm);
    end;
end.
```

## Precondiciones del ejemplo

- Ambos archivos (maestro y detalle) están ordenados por código del producto
- En el archivo detalle solo aparecen productos que existen en el archivo maestro
- Cada producto del maestro puede ser, a lo largo del día, vendido más de una vez, por lo tanto, en el archivo detalle pueden existir varios registros correspondientes al mismo producto

# Archivos → Ej 6 Un Maestro Un detalle Nuevas condiciones (Cont)

8

```
program actualizar;
  const valoralto='9999';
  type str4 = string[4];
  prod = record
    cod: str4;
    descripcion:
      string[30];
    pu: real;
    cant: integer;
  end;
  v_prod = record
    cod: str4;
    cv: integer;
  {cantidad vendida}
  end;
  detalle = file of v_prod;
  maestro = file of prod;

  var
    regm: prod;
    regd: v_prod;
    mael: maestro;
    det1: detalle;
    total: integer;
```

```
begin
  assign (mael, 'maestro');
  assign (det1, 'detalle');
  {proceso principal}
  reset (mael); reset (det1);
  while (not eof(det1)) do begin
    read(mael, regm);
    read(det1, regd);
    while (regm.cod <> regd.cod) do
      read (mael, regm);
    while not eof(det1) and (regm.cod = regd.cod) do
      begin
        regm.cant := regm.cant - regd.cv;
        read (det1, regd);
      end;
    If not eof(det1)
      seek( det1, filepos(det1)-1)
    seek (mael, filepos(mael)-1);
    write(mael, regm);
  end;
end.
```

Esta solución planteada NO ES CORRECTA  
ya que genera ciertos inconvenientes (ver  
pág. 34 (pdf) del libro )



# Archivos → Ej 6 Un Maestro Un detalle Nuevas condiciones (Cont)

9

```
procedure leer (var archivo:detalle; var dato:v_prod);
begin
    if (not eof(archivo))
    then read (archivo,dato)
    else dato.cod := valoralto;
end;
begin
    assign (mael, 'maestro'); assign (det1, 'detalle');
    reset (mael); reset (det1);
    leer(det1,regd); {se procesan todos los registros del archivo det1}
    while (regd.cod <> valoralto) do begin
        read(mael, regm);
        while (regm.cod <> regd.cod) do
            read (mael,regm);
        { se procesan códigos iguales }
        while (regm.cod = regd.cod) do begin
            regm.cant := regm.cant - regd.cv;
            leer(det1,regd);
        end;
        {reubica el puntero}
        seek (mael, filepos(mael)-1);
        write(mael,regm);
    end;
End;
```

FOD - Clase 2

# Archivos → Ej 7 Un Maestro N detalle

El problema siguiente generaliza aún más el problema anterior

El maestro se actualiza con tres archivos detalles

Los archivos detalle están ordenados de menor a mayor

Condiciones de archivos iguales, misma declaración de tipos del problema anterior

# Archivos → Ej 7 Un Maestro N detalle (cont)

11

```
var
  regm: prod;  min, regd1, regd2,regd3: v_prod;
  mael: maestro;  det1,det2,det3: detalle;

procedure leer (var archivo: detalle; var dato:v_prod);
begin
  if (not eof(archivo))
    then read (archivo,dato)
    else dato.cod := valoralto;
end;

procedure minimo (var r1,r2,r3: v_prod; var min:v_prod);
begin
  if (r1.cod<=r2.cod) and (r1.cod<=r3.cod) then begin
    min := r1;
    leer(det1,r1)
  end
  else if (r2.cod<=r3.cod) then begin
    min := r2;
    leer(det2,r2)
  end
  else begin
    min := r3;
    leer(det3,r3)
  end;
end;
```

FOD - Clase 2

# Archivos → Ej 7 Un Maestro N detalle (cont)

```
begin
  assign (mael, 'maestro');    assign (det1, 'detalle1');
  assign (det2, 'detalle2');  assign (det3, 'detalle3');
  reset (mael); reset (det1); reset (det2); reset (det3);
  leer(det1, regd1); leer(det2, regd2); leer(det3, regd3);
  minimo(regd1, regd2, regd3, min);
  while (min.cod <> valoralto) do begin
    read(mael, regm);
    while (regm.cod <> min.cod) do
      read(mael, regm);
    while (regm.cod = min.cod ) do begin
      regm.cant:=regm.cant - min.cantvendida;
      minimo(regd1, regd2, regd3, min);
    end;
    seek (mael, filepos(mael)-1);
    write(mael, regm);
  end;
end.
```

# Archivos → Corte de control

El problema consiste en la generación de reportes

- Es un problema clásico en el manejo de BD.
- Si bien los DBMS lo manejan diferente, veremos la algorítmica clásica de los mismos
- Precondiciones
  - El archivo se encuentra ordenado por provincia, partido y ciudad

Provincia: xxxx			
Partido: yyyy			
Ciudad	# Var.	# Muj.	Desocupados
aaa	.....	.....	.....
bbb	.....	.....	.....
ccc	.....	.....	.....
Total Partido .....			
Partido: zzzz			
Ciudad	# Var.	# Muj.	Desocupados
...	.....	.....	.....
Total Partido .....			
Total Provincia: .....			
Provincia: qqqq			

# Archivos → Ej 8 Corte de control

14

```
program Corte_de_Control;
  const valoralto='zzzz';
  type str10 = string[10];
  prov = record
    provincia, partido, ciudad: str10;
    cant_varones, cant_mujeres, cant_desocupados : integer;
  end;
  instituto = file of prov;

  var regm: prov;
  inst: instituto;
  t_varones, t_mujeres, t_desocupados: integer;
  t_prov_var, t_prov_muj, t_prov_des: integer;
  ant_prov, ant_partido : str10;

  procedure leer (var archivo:instituto;  var dato:prov);
  begin
    if (not eof( archivo ))
    then read (archivo,dato)
    else dato.provincia := valoralto;
    end;
```

# Archivos → Ej 8 Corte de control

15

begin

```
    assign (inst, 'censo' ); reset (inst); leer (inst, regm);
    writeln ('Provincia: ', regm.provincia); writeln ('Partido: ', regm.partido); writeln
    ('Ciudad','Varones','Mujeres','Desocupados');
    { se inician los contadores para el total del partido para varones, mujeres y desocupados }
    t_varones := 0;      t_mujeres := 0;      t_desocupados := 0;
    { se inician los contadores para el total de cada provincia }
    t_prov_var := 0;      t_prov_muj := 0;      t_prov_des := 0;
    while ( regm.provincia <> valoralto)do begin
        ant_prov := regm.provincia; ant_partido := regm.partido;
        while (ant_prov=regm.provincia) and (ant_partido=regm.partido) do begin
            write (regm.ciudad, regm.cant_varones, regm.cant_mujeres,regm.cant_desocupados);
            t_varones := t_varones + regm.cant_varones; t_mujeres := t_mujeres + regm.cant_mujeres;
            t_desocupados := t_desocupados + regm.cant_desocupados;
            leer (inst, regm);
        end;
        writeln ('Total Partido: ', t_varones,t_mujeres,t_desocupados);
        t_prov_var := t_prov_var + t_varones; t_prov_muj := t_prov_muj + t_mujeres;
        t_prov_des := t_prov_des + t_desocupados;
        t_varones := 0; t_mujeres := 0; t_desocupados := 0;
        ant_partido := regm.partido;
        if (ant_prov <> regm.provincia) then begin
            writeln ('Total Provincia',t_prov_var,t_prov_muj,t_prov_des);
            t_prov_var := 0; t_prov_muj := 0; t_prov_des := 0;
            writeln ('Provincia: ', regm.provincia);
        end;
        writeln ('Partido: ', regm.partido);
    end;
end.
```

# Archivos - Merge

- Involucra archivos con contenido similar, el cual debe resumirse en un único archivo.
- **Precondiciones:**
  - Todos los archivos detalle tienen igual estructura
  - Todos están ordenados por igual criterio
- **Primer ejemplo:**
  - CADP inscribe a los alumnos que cursarán la materia en tres computadoras separadas. C/U de ellas genera un archivo con los datos personales de los estudiantes, luego son ordenados físicamente por otro proceso. El problema que tienen los JTP es genera un archivo maestro de la asignatura
- **Precondiciones**
  - El proceso recibe tres archivos con igual estructura
  - Los archivos están ordenados por nombre de alumno
  - Un alumno solo aparece una vez en el archivo
- **Postcondición**
  - Se genera el archivo maestro de la asignatura ordenado por nombre del alumno



# Archivos – Ej 9: Merge 3 archivos

```
program union_de_archivos;  
17 const valoralto = 'zzzz';  
    type str30 = string[30];  
        str10 = string[10];  
        alumno = record  
            nombre: str30;  
            dni: str10;  
            direccion: str30;  
            carrera: str10;  
        end;  
        detalle = file of alumno;  
    var min,regd1,regd2,regd3: alumno;  
        det1,det2,det3,maestro : detalle;  
    procedure leer (var archivo:detalle; var  
        dato:alumno);  
    begin  
        if (not eof( archivo ))  
            then read (archivo, dato)  
            else dato.nombre := valoralto;  
    end;
```

FOD - Clase 2

```
procedure minimo (var r1,r2,r3:alumno; var  
    min:alumno);  
    begin  
        if (r1.nombre<r2.nombre) and  
            (r1.nombre<r3.nombre) then begin  
            min := r1;  
            leer(det1,r1)  
        end  
        else if (r2.nombre<r3.nombre) then  
            begin  
                min := r2;  
                leer(det2,r2)  
            end  
        else begin  
            min := r3;  
            leer(det3,r3)  
        end;  
    end;
```

# Archivos – Ej 9: Merge 3 archivos

```
begin
    assign (det1, 'det1');
    assign (det2, 'det2');
    assign (det3, 'det3');
    assign (maestro, 'maestro');
    rewrite (maestro);
    reset (det1);      reset (det2);      reset (det3);

    leer(det1, regd1); leer(det2, regd2); leer(det3, regd3);
    minimo(regd1, regd2, regd3, min);

    { se procesan los tres archivos }
    while (min.nombre <> valoralto) do
        begin
            write (maestro,min);
            minimo (regd1,regd2,regd3,min);
        end;
    close (maestro);

end.
```

# Archivos – Ej 10: Merge 3 archivos con repetición

- Los vendedores de cierto comercio asientan las ventas realizadas .....
- Precondiciones
  - Similar al anterior
  - Cada vendedor puede realizar varias ventas diarias

```

program union de archivos II;
const valor_alm = '9999';
type str4 = string[4];
    str10 = string[10];
    vendedor = record
        cod: str4;
        producto: str10;
        montoVenta: real;
    end;
    ventas = record
        cod: str4;
        total: real;
    end;
    detalle = file of vendedor;
    maestro = file of ventas;

```

```

var min, regd1, regd2, regd3: vendedor;
    det1, det2, det3: detalle;
    mael: maestro;
    regm: ventas;
    aux: str4;

```

# Archivos – Ej 10: Merge 3 archivos con repetición

20

begin

```
assign (det1, 'det1'); assign (det2, 'det2'); assign (det3, 'det3'); assign  
(mael, 'maestro');
```

```
reset (det1);      reset (det2);      reset (det3);
```

```
rewrite (mael);
```

```
leer (det1, regd1);      leer (det2, regd2);      leer (det3, regd3);
```

```
minimo (regd1, regd2, regd3,min);
```

```
{ se procesan los archivos de detalles }
```

```
while (min.cod <> valoralto) do begin
```

```
    {se asignan valores para registro del archivo maestro}
```

```
        regm.cod := min.cod;
```

```
        regm.total := 0;
```

```
        {se procesan todos los registros de un mismo vendedor}
```

```
        while (regm.cod = min.cod ) do begin
```

```
            regm.total := regm.total+ min.montoVenta;
```

```
            minimo (regd1, regd2, regd3, min);
```

```
        end;
```

```
        { se guarda en el archivo maestro}
```

```
        write(mael, regm);
```

```
    end;
```

```
End;
```

FOD - Clase 2

# Archivos – Ej 11: Merge N archivos con repetición

- Los vendedores de cierto comercio asientan las ventas realizadas.....
- Precondiciones
  - Similar al anterior
  - Cada vendedor puede realizar varias ventas diarias
- Idem anterior con N archivos....

# Archivos – Ej 11: Merge N archivos con repetición

22

```
program union_de_archivos_III;
  const valoralto = '9999';
  type vendedor = record
    cod: string[4];
    producto: string[10];
    montoVenta: real;
  end;
  ventas = record
    cod: string[4];
    total: real;
  end;
  maestro = file of ventas;
  arc_detalle=array[1..100] of file of vendedor;
  reg_detalle=array[1..100] of vendedor;
var min: vendedor;
  deta: arc_detalle;
  reg_det: reg_detalle;
  mael: maestro;
  regm: ventas;
  i,n: integer;
```

# Archivos – Ej 11: Merge N archivos con repetición

23

```
procedure leer (var archivo:detalle; var dato:vendedor);
begin
    if (not eof( archivo ))
    then read (archivo, dato)
    else dato.cod := valoralto;
end;

procedure minimo (var reg_det: reg_detalle; var min:vendedor; var deta:arc_detalle);
var i: integer;
begin
    { busco el mínimo elemento del
      vector reg_det en el campo cod,
      supongamos que es el índice i }
    min = reg_det[i];
    leer( deta[i], reg_det[i];
end;

begin
    Read(n)
    for i:= 1 to n do begin
        assign (deta[i], 'det'+i);
        { ojo lo anterior es incompatible en tipos}
        reset( deta[i] );
        leer( deta[i], reg_det[i] );
    end;
    assign (mael, 'maestro'); rewrite (mael);
    minimo (reg_det, min, deta);
```

# Archivos – Ej 11: Merge N archivos con repetición

24

```
{ se procesan los archivos de detalles }  
  while (min.cod <> valoralto) do  
    begin  
      {se asignan valores para registro del archivo maestro}  
      regm.cod := min.cod;  
      regm.total := 0;  
  
      {se procesan todos los registros de un mismo vendedor}  
      while (regm.cod = min.cod ) do begin  
        regm.total := regm.total+ min.montoVenta;  
        minimo (regd1, regd2, regd3, min);  
      end;  
  
      { se guarda en el archivo maestro}  
      write(mael, regm);  
  
    end;
```