Resumen FOD

Base de datos

Colección de archivos diseñados para servir a múltiples aplicaciones. Un dato representa hechos conocidos que pueden registrarse y que tienen un resultado implícito.

Propiedades implícitas:

- Representa aspectos del mundo real.
- Colección coherente de datos con significados incoherentes.
- Se diseña, construye y completa de datos para un propósito especifico.
- Está sustentada físicamente en archivos en dispositivos de almacenamiento persistente de datos.

Archivos

Colección de registros guardados en almacenamiento secundario.

Organización:

- Secuencia de Bytes: no se puede determinar fácilmente el comienzo y fin de los datos.
- Campo: unidad mas pequeña, lógicamente mas significativa de un archivo.
- Registros: conjunto de campos agrupados que definen un elemento de un archivo.

Acceso:

- Secuencial físico: acceso a los registros uno tras otro y en el orden físico en el que están guardados.
- Secuencial indizado: acceso a los registros de acuerdo al orden establecido por otra estructura.
- Directo: se accede directamente al registro.

Tipos:

De acuerdo al acceso podemos encontrar:

- Serie: cada registro es accesible luego de procesar su anterior.
- Secuencial: los registros son accesibles en orden de alguna clave.
- Directo: se accede al registro deseado.

Buffers:

Memoria intermedia entre un archivo y un programa, donde los datos residen temporalmente hasta ser almacenados definitivamente en memoria secundaria o donde residen una vez recuperados de ella.

Niveles:

- Físicos (almacenamiento secundario).
- Lógicos

Operaciones:

- Declaración:
  Type
  Archivo: file of tipo_de_dato;
  Var
  Arch: Archivo;
- Assign: establece la correspondencia entre el nombre físico y el nombre lógico.
- Rewrite: Solo de escritura (creación).
- Reset: Solo de lectura (apertura).
- Close: Cierre del archivo. Esta instrucción indica que no se va a trabajar mas con el archivo.
- Read (nombre_lógico, variable): Esta operación lee sobre los buffers relacionados a los archivos.
- Write (nombre_lógico, variable): Esta operación escribe sobre los buffers relacionados a los archivos.
- EOF: Fin de archivo.
- FileSize: Indica el tamaño del archivo.
- FilePos: Indica la posición dentro del archivo.
- Seek (nombre_lógico, posición):Nos permite ir a la posición del archivo.

Algoritmia Básica:

- Agregar nuevos elementos

```
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;
```

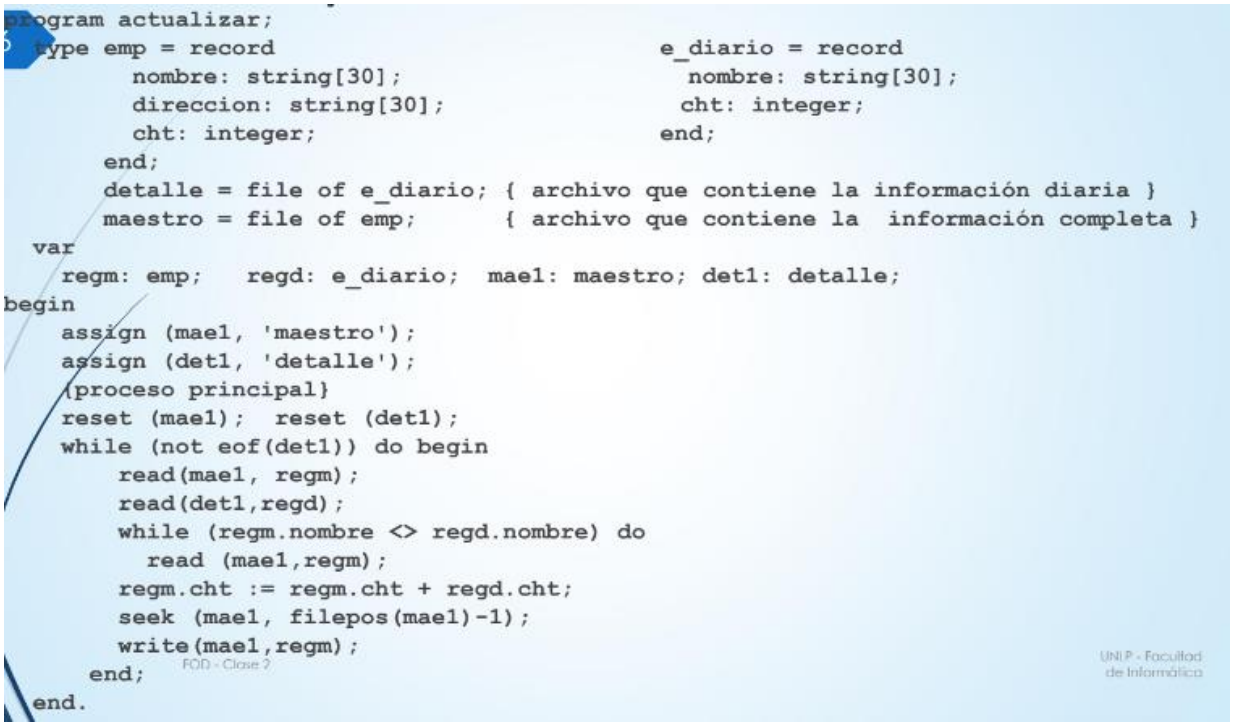
Algoritmia Clásica:

- Actualización maestro-detalle:
  - Maestro: archivo que resume un determinado conjunto de datos.
  - Detalle: archivo que agrupa información que se utilizará para modificar el contenido del archivo maestro.

Ej1: un maestro y un detalle.

Precondiciones:

- Ambos archivos están ordenados por el mismo criterio.
- En el archivo detalle solo aparecen empleados que existen en el archivo maestro.
- Cada empleado del archivo maestro a lo sumo puede aparecer una vez en el archivo detalle.



```
program actualizar;
  type emp = record                          e_diario = record
         nombre: string[30];                   nombre: string[30];
         direccion: string[30];                cht: integer;
         cht: integer;                       end;
       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;  mae1: maestro; det1: detalle;
begin
    assign (mae1, 'maestro');
    assign (det1, 'detalle');
    {proceso principal}
    reset (mae1);  reset (det1);
    while (not eof(det1)) do begin
        read(mae1, regm);
        read(det1,regd);
        while (regm.nombre <> regd.nombre) do
          read (mae1,regm);
        regm.cht := regm.cht + regd.cht;
        seek (mae1, filepos(mae1)-1);
        write(mae1,regm);
     end;
end.
```

Ej2: un maestro, un detalle (nuevas condiciones).

Precondiciones:

- 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.

```
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;
    mae1: maestro;
    det1: detalle;
    total: integer;
begin
    assign (mae1, 'maestro');
    assign (det1, 'detalle');
    {proceso principal}
    reset (mae1);  reset (det1);
    while (not eof(det1)) do begin
        read(mae1, regm);
        read(det1,regd);
        while (regm.cod <> regd.cod) do
            read (mae1,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 (mae1, filepos(mae1)-1);
        write(mae1,regm);
    end;
end.
```



```
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 (mae1, 'maestro');  assign (det1, 'detalle');
    reset (mae1);  reset (det1);
    leer(det1,regd);  {se procesan todos los registros del archivo det1}
    while (regd.cod <> valoralto) do begin
        read(mae1, regm);
        while (regm.cod <> regd.cod) do
            read (mae1,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 (mae1, filepos(mae1)-1);
        write(mae1,regm);
    end;
```


Ej3: un maestro, N detalles.

Precondiciones:

- 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.

```
var
 regm: prod;  min, regd1, regd2,regd3: v_prod;
 mae1: 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;
```

```
begin
  assign (mae1, 'maestro');   assign (det1, 'detalle1');
  assign (det2, 'detalle2');  assign (det3, 'detalle3');
  reset (mae1);  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(mae1,regm);
      while (regm.cod <> min.cod) do
         read(mae1,regm);
      while (regm.cod = min.cod ) do begin
         regm.cant:=regm.cant - min.cantvendida;
         minimo(regd1, regd2, regd3, min);
      end;
      seek (mae1, filepos(mae1)-1);
      write(mae1,regm);
  end;
end.
```