# Codigos

## Leer Detalle

```
procedure LeerD (var archivo:detalle;var dato:licencia);
begin
  if(not EOF(archivo)) then
        read(archivo,dato)
  else
     dato.codigo_E:=numeroAlto;
    end;
```

## Minimo y declaracion de archivos

```
detalle= file of licencia;
maestro= file of empleado;
vecdet=array [1..cantD] of detalle;
veclic=array [1..cantD] of licencia;
procedure Minimo(var archivo:vecdet; var rdet:veclic; var min:licencia);
var
  pos,i:integer;
begin
  pos:=1;
  min:=rdet[pos];
  for i:= 2 to cantD do begin
     if(rdet[i].codigo_E<min.codigo_E)then begin
       min:=rdet[pos];
       pos:=i;
     end;
  end;
  leerD(archivo[pos],rdet[pos]);
end;
```

## Creación de Archivo Binario desde Texto

```
program CrearArchivoMateriales;
type
  tMaterial = string[50];
  tArchivo = file of tMaterial;

var
  arch: tArchivo;
  nombreArch: string;
  material: tMaterial;
begin
  Write('Nombre del archivo binario: ');
  ReadLn(nombreArch);
  Assign(arch, nombreArch);
  Rewrite(arch); // Crear archivo nuevo

  Write('Material (fin con "cemento"): ');
  ReadLn(material);
  while (material <> 'cemento') do begin
    Write(arch, material); // Escribir en binario
    ReadLn(material);
  end;

  Close(arch);
end.
```

## Actualización Maestro-Detalle

```
procedure ActualizarMaestro(var mae: File of tProducto; var det: File of tVenta
var
  regM: tProducto;
  regD: tVenta;
  codActual: string;
  totalVendido: integer;
begin
  Reset(mae);
  Reset(det);
```

```
  LeerDetalle(det, regD); // Procedimiento personalizado para leer detalles
  while (regD.cod <> 'ZZZZ') do begin
    codActual := regD.cod;
    totalVendido := 0;

    // Acumular ventas del mismo producto
    while (regD.cod = codActual) do begin
      totalVendido := totalVendido + regD.cant;
      LeerDetalle(det, regD);
    end;

    // Buscar producto en el maestro
    BuscarEnMaestro(mae, codActual, regM);
    regM.stock := regM.stock - totalVendido;

    // Sobrescribir registro actualizado
    Seek(mae, FilePos(mae) - 1);
    Write(mae, regM);
  end;

  Close(mae);
  Close(det);
end;
```

## Merge de Archivos

```
procedure MergeDetalles(var mae: File of tProducto; dets: array of File of tPro
var
  min: tProducto;
  regD: array[1..3] of tProducto;
  i: integer;
begin
  Rewrite(mae); // Crear nuevo maestro
  for i := 1 to 3 do Reset(dets[i]); // Abrir detalles

  // Inicializar registros
  for i := 1 to 3 do LeerDetalle(dets[i], regD[i]);
```

```
  // Encontrar mínimo código
  ObtenerMinimo(regD, min);

  while (min.cod <> 'ZZZZ') do begin
    Write(mae, min); // Escribir en maestro
    ObtenerMinimo(regD, min); // Actualizar mínimo
  end;

  for i := 1 to 3 do Close(dets[i]);
  Close(mae);
end;
```

## Cortes de Control

```
procedure ProcesarHogares(var arch: File of tHogar);
var
  reg: tHogar;
  provinciaAct, localidadAct: integer;
  totalProvincia, totalLocalidad: integer;
begin
  Reset(arch);
  LeerHogar(arch, reg);

  while (reg.codProv <> 999) do begin
    provinciaAct := reg.codProv;
    totalProvincia := 0;
    WriteLn('Provincia: ', provinciaAct);

    while (reg.codProv = provinciaAct) do begin
      localidadAct := reg.codLoc;
      totalLocalidad := 0;

      while (reg.codProv = provinciaAct) and (reg.codLoc = localidadAct) do beg
        totalLocalidad := totalLocalidad + reg.cantidad;
        LeerHogar(arch, reg);
      end;
```

```
      WriteLn('  Localidad ', localidadAct, ': ', totalLocalidad);
      totalProvincia := totalProvincia + totalLocalidad;
    end;

    WriteLn('Total Provincia: ', totalProvincia);
  end;

  Close(arch);
end;
```

## Baja Fisica

```
procedure BajaFisica(var arch: File of tVehiculo; codigo: integer);
var
  reg, ultimo: tVehiculo;
  posBorrar: integer;
begin
  Reset(arch);
  Seek(arch, FileSize(arch) - 1); // Ir al último registro
  Read(arch, ultimo); // Guardar último registro
  Seek(arch, 0);

  // Buscar registro a borrar
  while (not Eof(arch)) and (reg.codigo <> codigo) do
    Read(arch, reg);

  if (reg.codigo = codigo) then begin
    posBorrar := FilePos(arch) - 1;
    Seek(arch, posBorrar);
    Write(arch, ultimo); // Sobrescribir con último
    Seek(arch, FileSize(arch) - 1);
    Truncate(arch); // Eliminar último duplicado
  end;

  Close(arch);
end;
```