Tema 1. Archivos.

Mezcla, actualización y enfrentamiento de archivos secuenciales

Mezcla

```
algoritmo mezcla
// Mezcla dos archivos ordenados por la misma clave
// en otro archivo también ordenado por la misma clave
tipos
   registro = tipoReg
      entero : clave
      //Resto de datos
   fin_registro
   archivo_s de tipoReg = tipoArch
var
   tipoArch : A,A1,A2
   tipoReg : R1,R2
inicio
   crear('NUEVO.DAT')
   abrir(A,'NUEVO.DAT',escritura)
   abrir(A1,'ARCHIVO1.DAT',lectura)
   abrir(A2,'ARCHIVO2.DAT',lectura)
   leer(A1,R1)
   leer(A2,R2)
   mientras no fda(A1) o no fda(A2) hacer
      si r1.clave < r2.clave entonces
         escribir(A,R1)
         leer(A1,R1)
      si_no
         escribir(A,R2)
         leer(A1,R2)
      fin_si
      //Para controlar el final de fichero de alguno de los archivos
      //y continuar la mezcla
      si fda(A1) entonces
         r1.clave ← +∞
      fin_si
      si fda(A2) entonces
         r1.clave ← +∞
      fin_si
   fin_mientras
   cerrar(A,A1,A2)
fin
```

Enfrentamiento de archivos

```
algoritmo EnfrentamientoArchivoMaestroConArchivoVentas
//Actualiza un archivo maestro de productos, restando las unidades vendidas
//al campo stock. Las unidades vendidas están almacenadas en un archivo de
movimientos
//Ambos archivos están ordenados por el campo código
tipos
   registro = RMaestro
      entero : código
      cadena :descripción
      entero : stock
   fin_registro
   archivo_s de RMaestro = AMaestro
   //El registro de movimiento contiene las ventas realizadas de los productos
   //Guarda información sobre el código del producto y el número de unidades vendidas
   registro = RMovimiento
      entero: código, unidades
```

```
   fin_registro
   archivo de RMovimiento : AMovimientos
var
  AMaestro: A, ANue
  AMovimientos: AMov
  RMaestro: R
  RMovimiento : RMov
inicio
   abrir(A,'PRODUCTOS.DAT',lectura)
   abrir(AMov,'VENTAS.DAT',lectura)
   crear('NUEVO.DAT')
   abrir(ANue,'NUEVO.DAT',escritura)
   leer(A,R)
   leer(AMov,RMov)
   mientras no fda(A) hacer
      si R.código = RMov.código entonces
          //Si el código es igual es que ese producto ha tenido ventas…
          //...se restan las ventas al stock
          R.stock ← R.stock – RMov.unidades
          //…y se lee la siguiente venta
          leer(AMov,RMov)
      fin_si
      //Siempre se escribe el resultado en el archivo nuevo de salida
      escribir(ANue,R)
      //En cualquier caso siempre se lee el siguiente producto
      leer(A,R)
   fin_mientras
   cerrar(A,AMov,ANue)
fin
```

Actualización de archivos

```
algoritmo ActualizaciónArchivoMaestroConArchivoDeMovimientos
//Actualiza un archivo maestro con otro de movimientos,
//dando altas, bajas o modificaciones según el valor del campo tipo
//Ambos archivos están ordenados por el campo código
tipos
   registro = RMaestro
      entero : código
      cadena :descripción
      entero : stock
   fin_registro
   archivo_s de RMaestro = AMaestro
   //El registro de movimiento contiene las los movimientos
   //en el archivo maestro
   //Además de la información del producto, también guarda información sobre
   //el tipo de movimiento (A) alta, (B) baja o (M) Modificación
   registro = RMaestro
      entero : código
      cadena :descripción
      entero : stock
      carácter : tipo
   fin_registro
   archivo de RMovimiento : AMovimientos
var
  AMaestro: A,ANue
  AMovimientos: AMov
  RMaestro: R, RNue
  RMovimiento: : RMov
inicio
   abrir(A,'PRIDUCTOS.DAT',lectura)
   abrir(ANue,'MOVIM.DAT',lectura)
   crear('NUEVO.DAT')
   abrir(ANue,'NUEVO.DAT',escritura)
   leer(A,R)
```

```
   leer(AMov,RMov)
   mientras no fda(A) o no fda(AMov) hacer
     si RMov.código < R.código entonces
        //RMov no existe en el maestro. Es un alta o un error
        //Primero se mueven los campos del registro de movimiento
        RNue.código ← RMov.código
        RNue.descripción ← RMov.descripción
        RNue.stock ← RMov.stock
        escribir(ANue,RNue)
        leer(AMov,RMov)
     si_no
        si R.código < RMov.dni entonces
           //R no tiene movimientos se graba el registro tal cual
           escribir(ANue,R)
        si_no
           //RMov existe en el maestro. Es una baja o modificación, o un error
           si RMov.tipo = 'M'
             //Primero se mueven los campos del registro de movimiento
                RNue.código ← RMov.código
                RNue.descripción ← RMov.descripción
                RNue.stock ← RMov.stock
                //...y se escribe el registro con los nuevos campos
                escribir(ANue,RNue)
           si_no
              si RMov.tipo = 'B' entonces
                 //Para dar una baja en un archivo secuencial,
                 //simplemente no se lleva nada al archivo de salida
              fin_si
           fin_si
           //Si hay movimientos, se lee el siguiente movimiento
           leer(AMov,Rmov)
         fin_si
         //Siempre que exista en el maestro, se lee del maestro
         leer(A,R)
     fin_si
     //Si se llega al final de los archivos se mueve el mayor valor a la clave
     si fda(AMov) entonces
        RMov.código ← +∞
     fin_si
     si fda(A) entonces
        R.código ← +∞
     fin_si
   fin_mientras
   cerrar(A,AMov,ANue)
fin
```

Ruptura de control

```
algoritmo ListadoTotalizandoLosSueldosPorDepartamento
//Hace un listado de un archivo de empleados ordenado por el campo departamento.
//Por cada departamento acumula el total de los sueldos
tipos
   //Definición del registro del archivo
   registro = REmpleado
      cadena: dni, nombre, dep
      real : suel
   fin_registro
   archivo_s de REmpleado = AEmpleado
var
   //Se supone que el archivo de empleados está ordenado por Departamento
   AEmpleado: A
   REmpleado: R
   cadena: depAux //Necesario para controlar el cambio de departamento
   real : TotalDep //Necesario para acumular los sueldos
```

```
inicio
   abrir(A,'EMPLEADOS.DAT',lectura)
   leer(A,R)
   mientras no fda(A) hacer
      //Por cada departamento se inicializa el total
      TotalDep ← 0
      //Por cada departamento se inicializa la variable auxiliar
      depAux ← R.dep
      mientras (R.dep = depAux) y no fda(A) hacer
          TotalDep ← TotalDep + R.suel
          leer(A,R)
      fin_mientras
      escribir(depAux, TotalDep)
   fin_mientras
   cerrar(A)
fin
```