Juan Menduiña

Trabajo Práctico N° 2:
Archivos Secuenciales Ordenados - Algorítmica Clásica.

Ejercicio 1.

Una empresa posee un archivo con información de los ingresos percibidos por diferentes empleados en concepto de comisión. De cada uno de ellos, se conoce: código de empleado, nombre y monto de la comisión. La información del archivo se encuentra ordenada por código de empleado y cada empleado puede aparecer más de una vez en el archivo de comisiones.

Realizar un procedimiento que reciba el archivo anteriormente descrito y lo compacte. En consecuencia, deberá generar un nuevo archivo en el cual cada empleado aparezca una única vez con el valor total de sus comisiones.

Nota: No se conoce, a priori, la cantidad de empleados. Además, el archivo debe ser recorrido una única vez.

```
program TP2_E1;
{$codepage UTF8}
uses crt, sysutils;
const
  codigo_salida=999;
type
  t_string10=string[10];
  t_registro_empleado=record
    codigo: int16;
    nombre: t_string10;
    comision: real;
  end;
  t_archivo_empleados=file of t_registro_empleado;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_empleados; var archivo_carga:
text);
var
  registro_empleado: t_registro_empleado;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga);
  while (not eof(archivo_carga)) do
    with registro_empleado do
    begin
      readln(archivo_carga,codigo,comision,nombre); nombre:=trim(nombre);
      write(archivo_detalle,registro_empleado);
    end;
  close(archivo_detalle);
  close(archivo_carga);
end;
procedure imprimir_registro_empleado(registro_empleado: t_registro_empleado);
begin
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_empleado.codigo);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_empleado.nombre);
  textcolor(green); write('; Comisión: '); textcolor(yellow);
writeln(registro_empleado.comision:0:2);
end;
procedure imprimir_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
  registro_empleado: t_registro_empleado;
begin
```

```
  reset(archivo_empleados);
  while (not eof(archivo_empleados)) do
  begin
    read(archivo_empleados,registro_empleado);
    imprimir_registro_empleado(registro_empleado);
  end;
  close(archivo_empleados);
end;
procedure leer_empleado(var archivo_detalle: t_archivo_empleados; var registro_empleado:
t_registro_empleado);
begin
  if (not eof(archivo_detalle)) then
    read(archivo_detalle,registro_empleado)
  else
    registro_empleado.codigo:=codigo_salida;
end;
procedure cargar_archivo_maestro(var archivo_maestro, archivo_detalle: t_archivo_empleados);
var
  registro_empleado_detalle, registro_empleado_maestro: t_registro_empleado;
  comision_total: real;
begin
  reset(archivo_detalle);
  rewrite(archivo_maestro);
  leer_empleado(archivo_detalle,registro_empleado_detalle);
  while (registro_empleado_detalle.codigo<>codigo_salida) do
  begin
    registro_empleado_maestro:=registro_empleado_detalle;
    comision_total:=0;
    while (registro_empleado_maestro.codigo=registro_empleado_detalle.codigo) do
    begin
      comision_total:=comision_total+registro_empleado_detalle.comision;
      leer_empleado(archivo_detalle,registro_empleado_detalle);
    end;
    registro_empleado_maestro.comision:=comision_total;
    write(archivo_maestro,registro_empleado_maestro);
  end;
  close(archivo_detalle);
  close(archivo_maestro);
end;
var
  archivo_detalle, archivo_maestro: t_archivo_empleados;
  archivo_carga: text;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE:'); writeln();
  assign(archivo_detalle,'empleadosDetalle'); assign(archivo_carga,'empleadosDetalle.txt');
  cargar_archivo_detalle(archivo_detalle,archivo_carga);
  imprimir_archivo_empleados(archivo_detalle);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'empleadosMaestro');
  cargar_archivo_maestro(archivo_maestro,archivo_detalle);
  imprimir_archivo_empleados(archivo_maestro);
end.
```

Ejercicio 2.

El encargado de ventas de un negocio de productos de limpieza desea administrar el stock de los productos que vende. Para ello, genera un archivo maestro donde figuran todos los productos que comercializa. De cada producto, se maneja la siguiente información: código de producto, nombre comercial, precio de venta, stock actual y stock mínimo. Diariamente, se genera un archivo detalle donde se registran todas las ventas de productos realizadas. De cada venta, se registran: código de producto y cantidad de unidades vendidas. Se pide realizar un programa con opciones para:

(a) *Actualizar el archivo maestro con el archivo detalle, sabiendo que:*

- *Ambos archivos están ordenados por código de producto.*
- *Cada registro del maestro puede ser actualizado por 0, 1 o más registros del archivo detalle.*
- *El archivo detalle sólo contiene registros que están en el archivo maestro.*

(b) *Listar en un archivo de texto llamado "stock_minimo.txt" aquellos productos cuyo stock actual esté por debajo del stock mínimo permitido.*

```
program TP2_E2;
{$codepage UTF8}
uses crt, sysutils;
const
  codigo_salida=999;
  opcion_salida=0;
type
  t_string10=string[10];
  t_registro_producto=record
    codigo: int16;
    nombre: t_string10;
    precio: real;
    stock_actual: int16;
    stock_minimo: int16;
  end;
  t_registro_venta=record
    codigo: int16;
    cantidad_vendida: int16;
  end;
  t_archivo_maestro=file of t_registro_producto;
  t_archivo_detalle=file of t_registro_venta;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var archivo_carga:
text);
var
  registro_producto: t_registro_producto;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga);
  while (not eof(archivo_carga)) do
    with registro_producto do
    begin
      readln(archivo_carga,codigo,precio,stock_actual,stock_minimo,nombre);
nombre:=trim(nombre);
      write(archivo_maestro,registro_producto);
    end;
  close(archivo_maestro);
  close(archivo_carga);
  textcolor(green); writeln('El archivo binario maestro fue creado y cargado con éxito');
```

```
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var archivo_carga:
text);
var
  registro_venta: t_registro_venta;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga);
  while (not eof(archivo_carga)) do
    with registro_venta do
    begin
      readln(archivo_carga,codigo,cantidad_vendida);
      write(archivo_detalle,registro_venta);
    end;
  close(archivo_detalle);
  close(archivo_carga);
  textcolor(green); writeln('El archivo binario detalle fue creado y cargado con éxito');
end;
procedure imprimir_registro_producto(registro_producto: t_registro_producto);
begin
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_producto.codigo);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_producto.nombre);
  textcolor(green); write('; Precio: '); textcolor(yellow);
write(registro_producto.precio:0:2);
  textcolor(green); write('; Stock actual: '); textcolor(yellow);
write(registro_producto.stock_actual);
  textcolor(green); write('; Stock mínimo: '); textcolor(yellow);
writeln(registro_producto.stock_minimo);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_producto: t_registro_producto;
begin
  reset(archivo_maestro);
  textcolor(green); writeln('Los productos del archivo maestro son: ');
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_producto);
    imprimir_registro_producto(registro_producto);
  end;
  close(archivo_maestro);
end;
procedure imprimir_registro_venta(registro_venta: t_registro_venta);
begin
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_venta.codigo);
  textcolor(green); write('; Cantidad vendida: '); textcolor(yellow);
writeln(registro_venta.cantidad_vendida);
end;
procedure imprimir_archivo_detalle(var archivo_detalle: t_archivo_detalle);
var
  registro_venta: t_registro_venta;
begin
  reset(archivo_detalle);
  textcolor(green); writeln('Las ventas del archivo detalle son: ');
  while (not eof(archivo_detalle)) do
  begin
    read(archivo_detalle,registro_venta);
    imprimir_registro_venta(registro_venta);
  end;
  close(archivo_detalle);
end;
procedure leer_venta(var archivo_detalle: t_archivo_detalle; var registro_venta:
t_registro_venta);
begin
  if (not eof(archivo_detalle)) then
    read(archivo_detalle,registro_venta)
```

```
  else
    registro_venta.codigo:=codigo_salida;
end;
procedure actualizar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_detalle: t_archivo_detalle);
var
  registro_producto: t_registro_producto;
  registro_venta: t_registro_venta;
begin
  reset(archivo_maestro);
  reset(archivo_detalle);
  leer_venta(archivo_detalle,registro_venta);
  while (registro_venta.codigo<>codigo_salida) do
  begin
    read(archivo_maestro,registro_producto);
    while (registro_producto.codigo<>registro_venta.codigo) do
      read(archivo_maestro,registro_producto);
    while (registro_producto.codigo=registro_venta.codigo) do
    begin
      if (registro_venta.cantidad_vendida>=registro_producto.stock_actual) then
        registro_producto.stock_actual:=0
      else
        registro_producto.stock_actual:=registro_producto.stock_actual-
registro_venta.cantidad_vendida;
      leer_venta(archivo_detalle,registro_venta);
    end;
    seek(archivo_maestro,filepos(archivo_maestro)-1);
    write(archivo_maestro,registro_producto);
  end;
  close(archivo_maestro);
  close(archivo_detalle);
  textcolor(green); writeln('El archivo maestro fue actualizado con éxito');
end;
procedure exportar_archivo_txt(var archivo_maestro: t_archivo_maestro);
var
  registro_producto: t_registro_producto;
  archivo_txt: text;
begin
  reset(archivo_maestro);
  assign(archivo_txt,'stock_minimo_E2.txt'); rewrite(archivo_txt);
  textcolor(green); writeln('Los productos cuyo stock actual está por debajo del stock mínimo
son: ');
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_producto);
    if (registro_producto.stock_actual<registro_producto.stock_minimo) then
    begin
      imprimir_registro_producto(registro_producto);
      with registro_producto do
        writeln(archivo_txt,codigo,' ',nombre,' ',precio:0:2,' ',stock_actual,'
',stock_minimo);
    end;
  end;
  close(archivo_maestro);
  close(archivo_txt);
  textcolor(green); writeln('El archivo de texto "stock_minimo.txt" fue creado y cargado con
éxito');
end;
procedure leer_opcion(var opcion: int8);
begin
  textcolor(red); writeln('MENÚ DE OPCIONES');
  textcolor(yellow); write('OPCIÓN 1: '); textcolor(green); writeln('Crear archivos de
registros ordenados de productos y cargarlos con datos ingresados desde archivos de texto');
  textcolor(yellow); write('OPCIÓN 2: '); textcolor(green); writeln('Listar en pantalla los
datos de los productos del archivo maestro');
```

```
  textcolor(yellow); write('OPCIÓN 3: '); textcolor(green); writeln('Listar en pantalla los
datos de los productos del archivo detalle');
  textcolor(yellow); write('OPCIÓN 4: '); textcolor(green); writeln('Actualizar el archivo
maestro con el archivo detalle');
  textcolor(yellow); write('OPCIÓN 5: '); textcolor(green); writeln('Listar en un archivo de
texto llamado "stock_minimo.txt" aquellos productos cuyo stock actual esté por debajo del
stock mínimo permitido');
  textcolor(yellow); write('OPCIÓN 0: '); textcolor(green); writeln('Salir del menú de
opciones');
  textcolor(green); write('Introducir opción elegida: '); textcolor(yellow); readln(opcion);
  writeln();
end;
procedure menu_opciones(var archivo_maestro: t_archivo_maestro; var archivo_detalle:
t_archivo_detalle; var archivo_carga_maestro, archivo_carga_detalle: text);
var
  opcion: int8;
begin
  leer_opcion(opcion);
  while (opcion<>opcion_salida) do
  begin
    case opcion of
      1:
      begin
        cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
        cargar_archivo_detalle(archivo_detalle,archivo_carga_detalle);
      end;
      2: imprimir_archivo_maestro(archivo_maestro);
      3: imprimir_archivo_detalle(archivo_detalle);
      4: actualizar_archivo_maestro(archivo_maestro,archivo_detalle);
      5: exportar_archivo_txt(archivo_maestro);
    else
        textcolor(green); writeln('La opción ingresada no corresponde a ninguna de las
mostradas en el menú de opciones');
    end;
    writeln();
    leer_opcion(opcion);
  end;
end;
var
  archivo_maestro: t_archivo_maestro;
  archivo_detalle: t_archivo_detalle;
  archivo_carga_maestro, archivo_carga_detalle: text;
begin
  assign(archivo_maestro,'productosMaestro_E2');
assign(archivo_carga_maestro,'productosMaestro_E2.txt');
  assign(archivo_detalle,'ventasDetalle_E2');
assign(archivo_carga_detalle,'ventasDetalle_E2.txt');
  menu_opciones(archivo_maestro,archivo_detalle,archivo_carga_maestro,archivo_carga_detalle);
end.
```

Ejercicio 3.

A partir de información sobre la alfabetización en la Argentina, se necesita actualizar un archivo que contiene los siguientes datos: nombre de provincia, cantidad de personas alfabetizadas y total de encuestados. Se reciben dos archivos detalle provenientes de dos agencias de censo diferentes. Dichos archivos contienen: nombre de la provincia, código de localidad, cantidad de alfabetizados y cantidad de encuestados. Se pide realizar los módulos necesarios para actualizar el archivo maestro a partir de los dos archivos detalle.

Nota: Los archivos están ordenados por nombre de provincia y, en los archivos detalle, pueden venir 0, 1 o más registros por cada provincia.

```
program TP2_E3;
{$codepage UTF8}
uses crt, sysutils;
const
  provincia_salida='ZZZ';
type
  t_string50=string[50];
  t_registro_provincia=record
    provincia: t_string50;
    alfabetizados: int16;
    encuestados: int16;
  end;
  t_registro_localidad=record
    provincia: t_string50;
    codigo: int16;
    alfabetizados: int16;
    encuestados: int16;
  end;
  t_archivo_maestro=file of t_registro_provincia;
  t_archivo_detalle=file of t_registro_localidad;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var archivo_carga:
text);
var
  registro_provincia: t_registro_provincia;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga);
  while (not eof(archivo_carga)) do
    with registro_provincia do
    begin
      readln(archivo_carga,alfabetizados,encuestados,provincia); provincia:=trim(provincia);
      write(archivo_maestro,registro_provincia);
    end;
  close(archivo_maestro);
  close(archivo_carga);
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var archivo_carga:
text);
var
  registro_localidad: t_registro_localidad;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga);
  while (not eof(archivo_carga)) do
    with registro_localidad do
    begin
      readln(archivo_carga,codigo,alfabetizados,encuestados,provincia);
provincia:=trim(provincia);
```

```
      write(archivo_detalle,registro_localidad);
    end;
  close(archivo_detalle);
  close(archivo_carga);
end;
procedure imprimir_registro_provincia(registro_provincia: t_registro_provincia);
begin
  textcolor(green); write('Provincia: '); textcolor(yellow);
write(registro_provincia.provincia);
  textcolor(green); write('; Alfabetizados: '); textcolor(yellow);
write(registro_provincia.alfabetizados);
  textcolor(green); write('; Encuestados: '); textcolor(yellow);
writeln(registro_provincia.encuestados);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_provincia: t_registro_provincia;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_provincia);
    imprimir_registro_provincia(registro_provincia);
  end;
  close(archivo_maestro);
end;
procedure imprimir_registro_localidad(registro_localidad: t_registro_localidad);
begin
  textcolor(green); write('Provincia: '); textcolor(yellow);
write(registro_localidad.provincia);
  textcolor(green); write('; Código de localidad: '); textcolor(yellow);
write(registro_localidad.codigo);
  textcolor(green); write('; Alfabetizados: '); textcolor(yellow);
write(registro_localidad.alfabetizados);
  textcolor(green); write('; Encuestados: '); textcolor(yellow);
writeln(registro_localidad.encuestados);
end;
procedure imprimir_archivo_detalle(var archivo_detalle: t_archivo_detalle);
var
  registro_localidad: t_registro_localidad;
begin
  reset(archivo_detalle);
  while (not eof(archivo_detalle)) do
  begin
    read(archivo_detalle,registro_localidad);
    imprimir_registro_localidad(registro_localidad);
  end;
  close(archivo_detalle);
end;
procedure leer_localidad(var archivo_detalle: t_archivo_detalle; var registro_localidad:
t_registro_localidad);
begin
  if (not eof(archivo_detalle)) then
    read(archivo_detalle,registro_localidad)
  else
    registro_localidad.provincia:=provincia_salida;
end;
procedure minimo(var archivo_detalle1, archivo_detalle2: t_archivo_detalle; var
registro_localidad1, registro_localidad2, min: t_registro_localidad);
begin
  if (registro_localidad1.provincia<=registro_localidad2.provincia) then
  begin
    min:=registro_localidad1;
    leer_localidad(archivo_detalle1,registro_localidad1);
  end
  else
```

```
  begin
    min:=registro_localidad2;
    leer_localidad(archivo_detalle2,registro_localidad2);
  end;
end;
procedure actualizar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_detalle1, archivo_detalle2: t_archivo_detalle);
var
  registro_provincia: t_registro_provincia;
  registro_localidad1, registro_localidad2, min: t_registro_localidad;
begin
  reset(archivo_maestro);
  reset(archivo_detalle1); reset(archivo_detalle2);
  leer_localidad(archivo_detalle1,registro_localidad1);
  leer_localidad(archivo_detalle2,registro_localidad2);
  minimo(archivo_detalle1,archivo_detalle2,registro_localidad1,registro_localidad2,min);
  while (min.provincia<>provincia_salida) do
  begin
    read(archivo_maestro,registro_provincia);
    while (registro_provincia.provincia<>min.provincia) do
      read(archivo_maestro,registro_provincia);
    while (registro_provincia.provincia=min.provincia) do
    begin
      registro_provincia.alfabetizados:=registro_provincia.alfabetizados+min.alfabetizados;
      registro_provincia.encuestados:=registro_provincia.encuestados+min.encuestados;
      minimo(archivo_detalle1,archivo_detalle2,registro_localidad1,registro_localidad2,min);
    end;
    seek(archivo_maestro,filepos(archivo_maestro)-1);
    write(archivo_maestro,registro_provincia);
  end;
  close(archivo_maestro);
  close(archivo_detalle1); close(archivo_detalle2);
end;
var
  archivo_maestro: t_archivo_maestro;
  archivo_detalle1, archivo_detalle2: t_archivo_detalle;
  archivo_carga_maestro, archivo_carga_detalle1, archivo_carga_detalle2: text;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'provinciaMaestro');
assign(archivo_carga_maestro,'provinciaMaestro.txt');
  cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
  imprimir_archivo_maestro(archivo_maestro);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE 1:'); writeln();
  assign(archivo_detalle1,'localidadDetalle1');
assign(archivo_carga_detalle1,'localidadDetalle1.txt');
  cargar_archivo_detalle(archivo_detalle1,archivo_carga_detalle1);
  imprimir_archivo_detalle(archivo_detalle1);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE 2:'); writeln();
  assign(archivo_detalle2,'localidadDetalle2');
assign(archivo_carga_detalle2,'localidadDetalle2.txt');
  cargar_archivo_detalle(archivo_detalle2,archivo_carga_detalle2);
  imprimir_archivo_detalle(archivo_detalle2);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO ACTUALIZADO:'); writeln();
  actualizar_archivo_maestro(archivo_maestro,archivo_detalle1,archivo_detalle2);
  imprimir_archivo_maestro(archivo_maestro);
end.
```

Ejercicio 4.

Se cuenta con un archivo de productos de una cadena de venta de alimentos congelados. De cada producto, se almacena: código del producto, nombre, descripción, stock disponible, stock mínimo y precio del producto.

Se recibe, diariamente, un archivo detalle de cada una de las 30 sucursales de la cadena. Se debe realizar el procedimiento que recibe los 30 detalles y actualiza el stock del archivo maestro. La información que se recibe en los detalles es: código de producto y cantidad vendida. Además, se deberá informar en un archivo de texto: nombre de producto, descripción, stock disponible y precio de aquellos productos que tengan stock disponible por debajo del stock mínimo. Pensar alternativas sobre realizar el informe en el mismo procedimiento de actualización o realizarlo en un procedimiento separado (analizar ventajas/desventajas en cada caso).

Nota: Todos los archivos se encuentran ordenados por código de productos. En cada detalle, puede venir 0 o N registros de un determinado producto.

```
program TP2_E4;
{$codepage UTF8}
uses crt, sysutils;
const
  detalles_total=3; // detalles_total=30;
  codigo_salida=999;
type
  t_detalle=1..detalles_total;
  t_string20=string[20];
  t_registro_producto=record
    codigo: int16;
    nombre: t_string20;
    descripcion: t_string20;
    stock_disponible: int16;
    stock_minimo: int16;
    precio: real;
  end;
  t_registro_venta=record
    codigo: int16;
    cantidad_vendida: int16;
  end;
  t_archivo_maestro=file of t_registro_producto;
  t_archivo_detalle=file of t_registro_venta;
  t_vector_ventas=array[t_detalle] of t_registro_venta;
  t_vector_detalles=array[t_detalle] of t_archivo_detalle;
  t_vector_carga_detalles=array[t_detalle] of text;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var archivo_carga:
text);
var
  registro_producto: t_registro_producto;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga);
  while (not eof(archivo_carga)) do
    with registro_producto do
    begin
      readln(archivo_carga,codigo,stock_disponible,stock_minimo,precio,nombre);
nombre:=trim(nombre);
      readln(archivo_carga,descripcion); descripcion:=trim(descripcion);
      write(archivo_maestro,registro_producto);
    end;
```

```
  close(archivo_maestro);
  close(archivo_carga);
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var archivo_carga:
text);
var
  registro_venta: t_registro_venta;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga);
  while (not eof(archivo_carga)) do
    with registro_venta do
    begin
      readln(archivo_carga,codigo,cantidad_vendida);
      write(archivo_detalle,registro_venta);
    end;
  close(archivo_detalle);
  close(archivo_carga);
end;
procedure imprimir_registro_producto(registro_producto: t_registro_producto);
begin
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_producto.codigo);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_producto.nombre);
  textcolor(green); write('; Descripción: '); textcolor(yellow);
write(registro_producto.descripcion);
  textcolor(green); write('; Stock disponible: '); textcolor(yellow);
write(registro_producto.stock_disponible);
  textcolor(green); write('; Stock mínimo: '); textcolor(yellow);
write(registro_producto.stock_minimo);
  textcolor(green); write('; Precio: '); textcolor(yellow);
writeln(registro_producto.precio:0:2);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_producto: t_registro_producto;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_producto);
    imprimir_registro_producto(registro_producto);
  end;
  close(archivo_maestro);
end;
procedure imprimir_registro_venta(registro_venta: t_registro_venta);
begin
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_venta.codigo);
  textcolor(green); write('; Cantidad vendida: '); textcolor(yellow);
writeln(registro_venta.cantidad_vendida);
end;
procedure imprimir_archivo_detalle(var archivo_detalle: t_archivo_detalle);
var
  registro_venta: t_registro_venta;
begin
  reset(archivo_detalle);
  while (not eof(archivo_detalle)) do
  begin
    read(archivo_detalle,registro_venta);
    imprimir_registro_venta(registro_venta);
  end;
  close(archivo_detalle);
end;
procedure leer_venta(var archivo_detalle: t_archivo_detalle; var registro_venta:
t_registro_venta);
begin
  if (not eof(archivo_detalle)) then
```

```
    read(archivo_detalle,registro_venta)
  else
    registro_venta.codigo:=codigo_salida;
end;
procedure minimo(var vector_detalles: t_vector_detalles; var vector_ventas: t_vector_ventas;
var min: t_registro_venta);
var
  i, pos: t_detalle;
begin
  min.codigo:=codigo_salida;
  for i:= 1 to detalles_total do
    if (vector_ventas[i].codigo<min.codigo) then
    begin
      min:=vector_ventas[i];
      pos:=i;
    end;
  if (min.codigo<codigo_salida) then
    leer_venta(vector_detalles[pos],vector_ventas[pos]);
end;
procedure exportar_archivo_txt(var archivo_maestro: t_archivo_maestro);
var
  registro_producto: t_registro_producto;
  archivo_txt: text;
begin
  reset(archivo_maestro);
  assign(archivo_txt,'stock_minimo_E4.txt'); rewrite(archivo_txt);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_producto);
    if (registro_producto.stock_disponible<registro_producto.stock_minimo) then
      with registro_producto do
        writeln(archivo_txt,nombre,' ',descripcion,' ',stock_disponible,' ',precio:0:2);
  end;
  close(archivo_txt);
end;
procedure actualizar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
vector_detalles: t_vector_detalles);
var
  registro_producto: t_registro_producto;
  vector_ventas: t_vector_ventas;
  min: t_registro_venta;
  i: t_detalle;
begin
  reset(archivo_maestro);
  for i:= 1 to detalles_total do
  begin
    reset(vector_detalles[i]);
    leer_venta(vector_detalles[i],vector_ventas[i]);
  end;
  minimo(vector_detalles,vector_ventas,min);
  while (min.codigo<>codigo_salida) do
  begin
    read(archivo_maestro,registro_producto);
    while (registro_producto.codigo<>min.codigo) do
      read(archivo_maestro,registro_producto);
    while (registro_producto.codigo=min.codigo) do
    begin
      if (min.cantidad_vendida>=registro_producto.stock_disponible) then
        registro_producto.stock_disponible:=0
      else
        registro_producto.stock_disponible:=registro_producto.stock_disponible-
min.cantidad_vendida;
      minimo(vector_detalles,vector_ventas,min);
    end;
    seek(archivo_maestro,filepos(archivo_maestro)-1);
    write(archivo_maestro,registro_producto);
```

```
  end;
  exportar_archivo_txt(archivo_maestro);
  close(archivo_maestro);
  for i:= 1 to detalles_total do
    close(vector_detalles[i]);
end;
var
  vector_detalles: t_vector_detalles;
  vector_carga_detalles: t_vector_carga_detalles;
  archivo_maestro: t_archivo_maestro;
  archivo_carga_maestro: text;
  i: t_detalle;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'productosMaestro_E4');
assign(archivo_carga_maestro,'productosMaestro_E4.txt');
  cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
  imprimir_archivo_maestro(archivo_maestro);
  for i:= 1 to detalles_total do
  begin
    writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE ',i,':'); writeln();
    assign(vector_detalles[i],'ventasDetalle'+intToStr(i)+'_E4');
assign(vector_carga_detalles[i],'ventasDetalle'+intToStr(i)+'_E4.txt');
    cargar_archivo_detalle(vector_detalles[i],vector_carga_detalles[i]);
    imprimir_archivo_detalle(vector_detalles[i]);
  end;
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO ACTUALIZADO:'); writeln();
  actualizar_archivo_maestro(archivo_maestro,vector_detalles);
  imprimir_archivo_maestro(archivo_maestro);
end.
```

Ejercicio 5.

Suponer que se trabaja en una oficina donde está montada una LAN (red local). La misma fue construida sobre una topología de red que conecta 5 máquinas entre sí y todas las máquinas se conectan con un servidor central. Semanalmente, cada máquina genera un archivo de logs informando las sesiones abiertas por cada usuario en cada terminal y por cuánto tiempo estuvo abierta. Cada archivo detalle contiene los siguientes campos: cod_usuario, fecha, tiempo_sesion. Se debe realizar un procedimiento que reciba los archivos detalle y genere un archivo maestro con los siguientes datos: cod_usuario, fecha, tiempo_total_de_sesiones_abiertas.

Notas:

- *Cada archivo detalle está ordenado por cod_usuario y fecha.*
- *Un usuario puede iniciar más de una sesión el mismo día en la misma máquina o, inclusive, en diferentes máquinas.*
- *El archivo maestro debe crearse en la siguiente ubicación física: /var/log.*

```
program TP2_E5;
{$codepage UTF8}
uses crt, sysutils;
const
  detalles_total=3; // detalles_total=5;
  codigo_salida=999; fecha_salida='ZZZ';
type
  t_detalle=1..detalles_total;
  t_string20=string[20];
  t_registro_sesion=record
    codigo: int16;
    fecha: t_string20;
    tiempo: int16;
  end;
  t_archivo_sesiones=file of t_registro_sesion;
  t_vector_sesiones=array[t_detalle] of t_registro_sesion;
  t_vector_detalles=array[t_detalle] of t_archivo_sesiones;
  t_vector_carga_detalles=array[t_detalle] of text;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_sesiones; var archivo_carga:
text);
var
  registro_sesion: t_registro_sesion;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga);
  while (not eof(archivo_carga)) do
    with registro_sesion do
    begin
      readln(archivo_carga,codigo,tiempo,fecha); fecha:=trim(fecha);
      write(archivo_detalle,registro_sesion);
    end;
  close(archivo_detalle);
  close(archivo_carga);
end;
procedure imprimir_registro_sesion(registro_sesion: t_registro_sesion);
begin
  textcolor(green); write('Código de usuario: '); textcolor(yellow);
write(registro_sesion.codigo);
  textcolor(green); write('; Fecha: '); textcolor(yellow); write(registro_sesion.fecha);
  textcolor(green); write('; Tiempo de sesión: '); textcolor(yellow);
writeln(registro_sesion.tiempo);
end;
```

```
procedure imprimir_archivo_sesiones(var archivo_sesiones: t_archivo_sesiones);
var
  registro_sesion: t_registro_sesion;
begin
  reset(archivo_sesiones);
  while (not eof(archivo_sesiones)) do
  begin
    read(archivo_sesiones,registro_sesion);
    imprimir_registro_sesion(registro_sesion);
  end;
  close(archivo_sesiones);
end;
procedure leer_sesion(var archivo_detalle: t_archivo_sesiones; var registro_sesion:
t_registro_sesion);
begin
  if (not eof(archivo_detalle)) then
    read(archivo_detalle,registro_sesion)
  else
    registro_sesion.codigo:=codigo_salida;
end;
procedure minimo(var vector_detalles: t_vector_detalles; var vector_sesiones:
t_vector_sesiones; var min: t_registro_sesion);
var
  i, pos: t_detalle;
begin
  min.codigo:=codigo_salida; min.fecha:=fecha_salida;
  for i:= 1 to detalles_total do
    if ((vector_sesiones[i].codigo<min.codigo) or ((vector_sesiones[i].codigo=min.codigo) and
(vector_sesiones[i].fecha<min.fecha))) then
    begin
      min:=vector_sesiones[i];
      pos:=i;
    end;
  if (min.codigo<codigo_salida) then
    leer_sesion(vector_detalles[pos],vector_sesiones[pos]);
end;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_sesiones; var vector_detalles:
t_vector_detalles);
var
  registro_sesion: t_registro_sesion;
  vector_sesiones: t_vector_sesiones;
  min: t_registro_sesion;
  i: t_detalle;
begin
  rewrite(archivo_maestro);
  for i:= 1 to detalles_total do
  begin
    reset(vector_detalles[i]);
    leer_sesion(vector_detalles[i],vector_sesiones[i]);
  end;
  minimo(vector_detalles,vector_sesiones,min);
  while (min.codigo<>codigo_salida) do
  begin
    registro_sesion.codigo:=min.codigo;
    while (registro_sesion.codigo=min.codigo) do
    begin
      registro_sesion.fecha:=min.fecha;
      registro_sesion.tiempo:=0;
      while ((registro_sesion.codigo=min.codigo) and (registro_sesion.fecha=min.fecha)) do
      begin
        registro_sesion.tiempo:=registro_sesion.tiempo+min.tiempo;
        minimo(vector_detalles,vector_sesiones,min);
      end;
      write(archivo_maestro,registro_sesion);
    end;
  end;
```

```
  close(archivo_maestro);
  for i:= 1 to detalles_total do
    close(vector_detalles[i]);
end;
var
  vector_detalles: t_vector_detalles;
  vector_carga_detalles: t_vector_carga_detalles;
  archivo_maestro: t_archivo_sesiones;
  i: t_detalle;
begin
  for i:= 1 to detalles_total do
  begin
    writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE ',i,':'); writeln();
    assign(vector_detalles[i],'sesionesDetalle'+inttoStr(i));
assign(vector_carga_detalles[i],'sesionesDetalle'+inttoStr(i)+'.txt');
    cargar_archivo_detalle(vector_detalles[i],vector_carga_detalles[i]);
    imprimir_archivo_sesiones(vector_detalles[i]);
  end;
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'sesionesMaestro');
  cargar_archivo_maestro(archivo_maestro,vector_detalles);
  imprimir_archivo_sesiones(archivo_maestro);
end.
```

Ejercicio 6.

Se desea modelar la información necesaria para un sistema de recuentos de casos de COVID para el ministerio de salud de la provincia de Buenos Aires.

Diariamente, se reciben archivos provenientes de los distintos municipios. La información contenida en los mismos es la siguiente: código de localidad, código cepa, cantidad de casos activos, cantidad de casos nuevos, cantidad de casos recuperados, cantidad de casos fallecidos.

El ministerio cuenta con un archivo maestro con la siguiente información: código localidad, nombre localidad, código cepa, nombre cepa, cantidad de casos activos, cantidad de casos nuevos, cantidad de recuperados y cantidad de fallecidos.

Se debe realizar el procedimiento que permita actualizar el maestro con los detalles recibidos, se reciben 10 detalles. Todos los archivos están ordenados por código de localidad y código de cepa.

Para la actualización, se debe proceder de la siguiente manera:

- *Al número de fallecidos se le suman el valor de fallecidos recibido del detalle.*
- *Ídem anterior para los recuperados.*
- *Los casos activos se actualizan con el valor recibido en el detalle.*
- *Ídem anterior para los casos nuevos hallados.*

Realizar las declaraciones necesarias, el programa principal y los procedimientos que requiera para la actualización solicitada e informar cantidad de localidades con más de 50 casos activos (las localidades pueden o no haber sido actualizadas).

Ejercicio 7.

Ejercicio 8.

Ejercicio 9.

Ejercicio 10.

Ejercicio 11.

Ejercicio 12.

Ejercicio 13.

Ejercicio 14.

Ejercicio 15.

Ejercicio 16.

Ejercicio 17.

Ejercicio 18.

Ejercicio 19.