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_string20=string[20];
  t_registro_empleado=record
    codigo: int16;
    nombre: t_string20;
    comision: real;
  end;
  t_archivo_empleados=file of t_registro_empleado;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_empleados; var
archivo_carga_detalle: text);
var
  registro_empleado: t_registro_empleado;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_empleado do
    begin
      readln(archivo_carga_detalle,codigo,comision,nombre); nombre:=trim(nombre);
      write(archivo_detalle,registro_empleado);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
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_detalle: text;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE:'); writeln();
  assign(archivo_detalle,'E1_empleadosDetalle');
assign(archivo_carga_detalle,'E1_empleadosDetalle.txt');
  cargar_archivo_detalle(archivo_detalle,archivo_carga_detalle);
  imprimir_archivo_empleados(archivo_detalle);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E1_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_string20=string[20];
  t_registro_producto=record
    codigo: int16;
    nombre: t_string20;
    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_maestro: text);
var
  registro_producto: t_registro_producto;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_producto do
    begin
      readln(archivo_carga_maestro,codigo,precio,stock_actual,stock_minimo,nombre);
nombre:=trim(nombre);
      write(archivo_maestro,registro_producto);
    end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
  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_detalle: text);
var
  registro_venta: t_registro_venta;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_venta do
    begin
      readln(archivo_carga_detalle,codigo,cantidad_vendida);
      write(archivo_detalle,registro_venta);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
  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,'E2_stock_minimo.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,'E2_productosMaestro');
assign(archivo_carga_maestro,'E2_productosMaestro.txt');
  assign(archivo_detalle,'E2_ventasDetalle');
assign(archivo_carga_detalle,'E2_ventasDetalle.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
  nombre_salida='ZZZ';
type
  t_string50=string[50];
  t_registro_provincia=record
    nombre: t_string50;
    alfabetizados: int16;
    encuestados: int16;
  end;
  t_registro_localidad=record
    nombre: 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_maestro: text);
var
  registro_provincia: t_registro_provincia;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_provincia do
    begin
      readln(archivo_carga_maestro,alfabetizados,encuestados,nombre); nombre:=trim(nombre);
      write(archivo_maestro,registro_provincia);
    end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var
archivo_carga_detalle: text);
var
  registro_localidad: t_registro_localidad;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_localidad do
    begin
      readln(archivo_carga_detalle,codigo,alfabetizados,encuestados,nombre);
nombre:=trim(nombre);
```

```
      write(archivo_detalle,registro_localidad);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
end;
procedure imprimir_registro_provincia(registro_provincia: t_registro_provincia);
begin
  textcolor(green); write('Provincia: '); textcolor(yellow); write(registro_provincia.nombre);
  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.nombre);
  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.nombre:=nombre_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.nombre<=registro_localidad2.nombre) 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.nombre<>nombre_salida) do
  begin
    read(archivo_maestro,registro_provincia);
    while (registro_provincia.nombre<>min.nombre) do
      read(archivo_maestro,registro_provincia);
    while (registro_provincia.nombre=min.nombre) 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,'E3_provinciasMaestro');
assign(archivo_carga_maestro,'E3_provinciasMaestro.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,'E3_localidadesDetalle1');
assign(archivo_carga_detalle1,'E3_localidadesDetalle1.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,'E3_localidadesDetalle2');
assign(archivo_carga_detalle2,'E3_localidadesDetalle2.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_maestro: text);
var
  registro_producto: t_registro_producto;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_producto do
    begin
      readln(archivo_carga_maestro,codigo,stock_disponible,stock_minimo,precio,nombre);
nombre:=trim(nombre);
      readln(archivo_carga_maestro,descripcion); descripcion:=trim(descripcion);
      write(archivo_maestro,registro_producto);
    end;
```

```
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var
archivo_carga_detalle: text);
var
  registro_venta: t_registro_venta;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_venta do
    begin
      readln(archivo_carga_detalle,codigo,cantidad_vendida);
      write(archivo_detalle,registro_venta);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
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,'E4_stock_minimo.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;
  min: t_registro_venta;
  vector_ventas: t_vector_ventas;
  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,'E4_productosMaestro');
assign(archivo_carga_maestro,'E4_productosMaestro.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],'E4_ventasDetalle'+intToStr(i));
assign(vector_carga_detalles[i],'E4_ventasDetalle'+intToStr(i)+'.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;
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_detalle: text);
var
  registro_sesion: t_registro_sesion;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_sesion do
    begin
      readln(archivo_carga_detalle,codigo,tiempo,fecha); fecha:=trim(fecha);
      write(archivo_detalle,registro_sesion);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
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;
  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, min: t_registro_sesion;
  vector_sesiones: t_vector_sesiones;
  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],'E5_sesionesDetalle'+inttoStr(i));
assign(vector_carga_detalles[i],'E5_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,'E5_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).

```
program TP2_E6;
{$codepage UTF8}
uses crt, sysutils;
const
  detalles_total=3; // detalles_total=10;
  codigo_salida=999;
  casos_activos_corte=50;
type
  t_detalle=1..detalles_total;
  t_string20=string[20];
  t_registro_localidad1=record
    codigo: int16;
    nombre: t_string20;
    codigo_cepa: int16;
    nombre_cepa: t_string20;
    casos_activos: int16;
    casos_nuevos: int16;
    casos_recuperados: int16;
    casos_fallecidos: int16;
  end;
  t_registro_localidad2=record
    codigo: int16;
    codigo_cepa: int16;
    casos_activos: int16;
    casos_nuevos: int16;
    casos_recuperados: int16;
```

```
    casos_fallecidos: int16;
  end;
  t_archivo_maestro=file of t_registro_localidad1;
  t_archivo_detalle=file of t_registro_localidad2;
  t_vector_localidades=array[t_detalle] of t_registro_localidad2;
  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_maestro: text);
var
  registro_localidad: t_registro_localidad1;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_localidad do
    begin
      readln(archivo_carga_maestro,codigo,codigo_cepa,casos_activos,casos_nuevos,casos_recuper
ados,casos_fallecidos,nombre_cepa); nombre_cepa:=trim(nombre_cepa);
      readln(archivo_carga_maestro,nombre); nombre:=trim(nombre);
      write(archivo_maestro,registro_localidad);
    end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var
archivo_carga_detalle: text);
var
  registro_localidad: t_registro_localidad2;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_localidad do
    begin
      readln(archivo_carga_detalle,codigo,codigo_cepa,casos_activos,casos_nuevos,casos_recuper
ados,casos_fallecidos);
      write(archivo_detalle,registro_localidad);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
end;
procedure imprimir_registro_localidad1(registro_localidad: t_registro_localidad1);
begin
  textcolor(green); write('Código de localidad: '); textcolor(yellow);
write(registro_localidad.codigo);
  textcolor(green); write('; Nombre de localidad: '); textcolor(yellow);
write(registro_localidad.nombre);
  textcolor(green); write('; Código de cepa: '); textcolor(yellow);
write(registro_localidad.codigo_cepa);
  textcolor(green); write('; Nombre de cepa: '); textcolor(yellow);
write(registro_localidad.nombre_cepa);
  textcolor(green); write('; Casos activos: '); textcolor(yellow);
write(registro_localidad.casos_activos);
  textcolor(green); write('; Casos nuevos: '); textcolor(yellow);
write(registro_localidad.casos_nuevos);
  textcolor(green); write('; Casos recuperados: '); textcolor(yellow);
write(registro_localidad.casos_recuperados);
  textcolor(green); write('; Casos fallecidos: '); textcolor(yellow);
writeln(registro_localidad.casos_fallecidos);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_localidad: t_registro_localidad1;
begin
  reset(archivo_maestro);
```

```
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_localidad);
    imprimir_registro_localidad1(registro_localidad);
  end;
  close(archivo_maestro);
end;
procedure imprimir_registro_localidad2(registro_localidad: t_registro_localidad2);
begin
  textcolor(green); write('Código de localidad: '); textcolor(yellow);
write(registro_localidad.codigo);
  textcolor(green); write('; Código de cepa: '); textcolor(yellow);
write(registro_localidad.codigo_cepa);
  textcolor(green); write('; Casos activos: '); textcolor(yellow);
write(registro_localidad.casos_activos);
  textcolor(green); write('; Casos nuevos: '); textcolor(yellow);
write(registro_localidad.casos_nuevos);
  textcolor(green); write('; Casos recuperados: '); textcolor(yellow);
write(registro_localidad.casos_recuperados);
  textcolor(green); write('; Casos fallecidos: '); textcolor(yellow);
writeln(registro_localidad.casos_fallecidos);
end;
procedure imprimir_archivo_detalles(var archivo_detalle: t_archivo_detalle);
var
  registro_localidad: t_registro_localidad2;
begin
  reset(archivo_detalle);
  while (not eof(archivo_detalle)) do
  begin
    read(archivo_detalle,registro_localidad);
    imprimir_registro_localidad2(registro_localidad);
  end;
  close(archivo_detalle);
end;
procedure leer_localidad(var archivo_detalle: t_archivo_detalle; var registro_localidad:
t_registro_localidad2);
begin
  if (not eof(archivo_detalle)) then
    read(archivo_detalle,registro_localidad)
  else
    registro_localidad.codigo:=codigo_salida;
end;
procedure minimo(var vector_detalles: t_vector_detalles; var vector_localidades:
t_vector_localidades; var min: t_registro_localidad2);
var
  i, pos: t_detalle;
begin
  min.codigo:=codigo_salida;
  for i:= 1 to detalles_total do
    if ((vector_localidades[i].codigo<min.codigo) or
((vector_localidades[i].codigo=min.codigo) and
(vector_localidades[i].codigo_cepa<min.codigo_cepa))) then
    begin
      min:=vector_localidades[i];
      pos:=i;
    end;
  if (min.codigo<codigo_salida) then
    leer_localidad(vector_detalles[pos],vector_localidades[pos]);
end;
procedure actualizar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
vector_detalles: t_vector_detalles);
var
  registro_localidad: t_registro_localidad1;
  min: t_registro_localidad2;
  vector_localidades: t_vector_localidades;
  i: t_detalle;
```

```
  casos_activos_localidad, localidades_corte: int16;
begin
  localidades_corte:=0;
  reset(archivo_maestro);
  for i:= 1 to detalles_total do
  begin
    reset(vector_detalles[i]);
    leer_localidad(vector_detalles[i],vector_localidades[i]);
  end;
  minimo(vector_detalles,vector_localidades,min);
  while (min.codigo<>codigo_salida) do
  begin
    casos_activos_localidad:=0;
    read(archivo_maestro,registro_localidad);
    while (registro_localidad.codigo<>min.codigo) do
      read(archivo_maestro,registro_localidad);
    while (registro_localidad.codigo=min.codigo) do
    begin
      while (registro_localidad.codigo_cepa<>min.codigo_cepa) do
        read(archivo_maestro,registro_localidad);
      while ((registro_localidad.codigo=min.codigo) and
(registro_localidad.codigo_cepa=min.codigo_cepa)) do
      begin
        registro_localidad.casos_fallecidos:=registro_localidad.casos_fallecidos+min.casos_fal
lecidos;
        registro_localidad.casos_recuperados:=registro_localidad.casos_recuperados+min.casos_r
ecuperados;
        registro_localidad.casos_activos:=min.casos_activos;
        registro_localidad.casos_nuevos:=min.casos_nuevos;
        casos_activos_localidad:=casos_activos_localidad+min.casos_activos;
        minimo(vector_detalles,vector_localidades,min);
      end;
      seek(archivo_maestro,filepos(archivo_maestro)-1);
      write(archivo_maestro,registro_localidad);
    end;
    if (casos_activos_localidad>casos_activos_corte) then
      localidades_corte:=localidades_corte+1;
  end;
  close(archivo_maestro);
  for i:= 1 to detalles_total do
    close(vector_detalles[i]);
  textcolor(green); write('La cantidad de localidades con más de '); textcolor(yellow);
write(casos_activos_corte); textcolor(green); write(' casos activos es: '); textcolor(red);
writeln(localidades_corte);
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,'E6_localidadesMaestro');
assign(archivo_carga_maestro,'E6_localidadesMaestro.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],'E6_localidadesDetalle'+inttoStr(i));
assign(vector_carga_detalles[i],'E6_localidadesDetalle'+inttoStr(i)+'.txt');
    cargar_archivo_detalle(vector_detalles[i],vector_carga_detalles[i]);
    imprimir_archivo_detalles(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 7.

Se dispone de un archivo maestro con información de los alumnos de la Facultad de Informática. Cada registro del archivo maestro contiene: código de alumno, apellido, nombre, cantidad de cursadas aprobadas y cantidad de materias con final aprobado. El archivo maestro está ordenado por código de alumno.

Además, se tienen dos archivos detalle con información sobre el desempeño académico de los alumnos: un archivo de cursadas y un archivo de exámenes finales. El archivo de cursadas contiene información sobre las materias cursadas por los alumnos. Cada registro incluye: código de alumno, código de materia, año de cursada y resultado (sólo interesa si la cursada fue aprobada o desaprobada). Por su parte, el archivo de exámenes finales contiene información sobre los exámenes finales rendidos. Cada registro incluye: código de alumno, código de materia, fecha del examen y nota obtenida. Ambos archivos detalle están ordenados por código de alumno y código de materia, y pueden contener 0, 1 o más registros por alumno en el archivo maestro. Un alumno podría cursar una materia muchas veces, así como también podría rendir el final de una materia en múltiples ocasiones.

Se debe desarrollar un programa que actualice el archivo maestro, ajustando la cantidad de cursadas aprobadas y la cantidad de materias con final aprobado, utilizando la información de los archivos detalle. Las reglas de actualización son las siguientes:

- *Si un alumno aprueba una cursada, se incrementa en uno la cantidad de cursadas aprobadas.*
- *Si un alumno aprueba un examen final (nota>=4), se incrementa en uno la cantidad de materias con final aprobado.*

Notas:

- *Los archivos deben procesarse en un único recorrido.*
- *No es necesario comprobar que no haya inconsistencias en la información de los archivos detalles. Esto es, no puede suceder que un alumno apruebe más de una vez la cursada de una misma materia (a lo sumo, la aprueba una vez), algo similar ocurre con los exámenes finales.*

```
program TP2_E7;
{$codepage UTF8}
uses crt, sysutils;
const
  codigo_salida=999;
  resultado_corte='Aprobada'; nota_corte=4.0;
  anio_ini=2000; anio_fin=2025;
type
  t_string20=string[20];
  t_anio=anio_ini..anio_fin;
  t_registro_alumno=record
    codigo: int16;
    apellido: t_string20;
    nombre: t_string20;
    cursadas_aprobadas: int16;
    finales_aprobados: int16;
  end;
```

```
  t_registro_cursada=record
    codigo: int16;
    codigo_materia: int16;
    anio: t_anio;
    resultado: t_string20;
  end;
  t_registro_final=record
    codigo: int16;
    codigo_materia: int16;
    fecha: t_string20;
    nota: real;
  end;
  t_registro_cursada_final=record
    codigo: int16;
    codigo_materia: int16;
    resultado: t_string20;
    nota: real;
  end;
  t_archivo_maestro=file of t_registro_alumno;
  t_archivo_detalle1=file of t_registro_cursada;
  t_archivo_detalle2=file of t_registro_final;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_carga_maestro: text);
var
  registro_alumno: t_registro_alumno;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_alumno do
    begin
      readln(archivo_carga_maestro,codigo,cursadas_aprobadas,finales_aprobados,apellido);
apellido:=trim(apellido);
      readln(archivo_carga_maestro,nombre); nombre:=trim(nombre);
      write(archivo_maestro,registro_alumno);
    end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure cargar_archivo_detalle1(var archivo_detalle1: t_archivo_detalle1; var
archivo_carga_detalle1: text);
var
  registro_cursada: t_registro_cursada;
begin
  rewrite(archivo_detalle1);
  reset(archivo_carga_detalle1);
  while (not eof(archivo_carga_detalle1)) do
    with registro_cursada do
    begin
      readln(archivo_carga_detalle1,codigo,codigo_materia,anio,resultado);
resultado:=trim(resultado);
      write(archivo_detalle1,registro_cursada);
    end;
  close(archivo_detalle1);
  close(archivo_carga_detalle1);
end;
procedure cargar_archivo_detalle2(var archivo_detalle2: t_archivo_detalle2; var
archivo_carga_detalle2: text);
var
  registro_final: t_registro_final;
begin
  rewrite(archivo_detalle2);
  reset(archivo_carga_detalle2);
  while (not eof(archivo_carga_detalle2)) do
    with registro_final do
    begin
```

```
      readln(archivo_carga_detalle2,codigo,codigo_materia,nota,fecha); fecha:=trim(fecha);
      write(archivo_detalle2,registro_final);
    end;
  close(archivo_detalle2);
  close(archivo_carga_detalle2);
end;
procedure imprimir_registro_alumno(registro_alumno: t_registro_alumno);
begin
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_alumno.codigo);
  textcolor(green); write('; Apellido: '); textcolor(yellow); write(registro_alumno.apellido);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_alumno.nombre);
  textcolor(green); write('; Cursadas aprobadas: '); textcolor(yellow);
write(registro_alumno.cursadas_aprobadas);
  textcolor(green); write('; Finales aprobados: '); textcolor(yellow);
writeln(registro_alumno.finales_aprobados);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_alumno: t_registro_alumno;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_alumno);
    imprimir_registro_alumno(registro_alumno);
  end;
  close(archivo_maestro);
end;
procedure imprimir_registro_cursada(registro_cursada: t_registro_cursada);
begin
  textcolor(green); write('Código de alumno: '); textcolor(yellow);
write(registro_cursada.codigo);
  textcolor(green); write('; Código de materia: '); textcolor(yellow);
write(registro_cursada.codigo_materia);
  textcolor(green); write('; Año: '); textcolor(yellow); write(registro_cursada.anio);
  textcolor(green); write('; Resultado: '); textcolor(yellow);
writeln(registro_cursada.resultado);
end;
procedure imprimir_archivo_detalle1(var archivo_detalle1: t_archivo_detalle1);
var
  registro_cursada: t_registro_cursada;
begin
  reset(archivo_detalle1);
  while (not eof(archivo_detalle1)) do
  begin
    read(archivo_detalle1,registro_cursada);
    imprimir_registro_cursada(registro_cursada);
  end;
  close(archivo_detalle1);
end;
procedure imprimir_registro_final(registro_final: t_registro_final);
begin
  textcolor(green); write('Código de alumno: '); textcolor(yellow);
write(registro_final.codigo);
  textcolor(green); write('; Código de materia: '); textcolor(yellow);
write(registro_final.codigo_materia);
  textcolor(green); write('; Fecha: '); textcolor(yellow); write(registro_final.fecha);
  textcolor(green); write('; Nota: '); textcolor(yellow); writeln(registro_final.nota:0:2);
end;
procedure imprimir_archivo_detalle2(var archivo_detalle2: t_archivo_detalle2);
var
  registro_final: t_registro_final;
begin
  reset(archivo_detalle2);
  while (not eof(archivo_detalle2)) do
  begin
```

```
    read(archivo_detalle2,registro_final);
    imprimir_registro_final(registro_final);
  end;
  close(archivo_detalle2);
end;
procedure leer_cursada(var archivo_detalle1: t_archivo_detalle1; var registro_cursada:
t_registro_cursada);
begin
  if (not eof(archivo_detalle1)) then
    read(archivo_detalle1,registro_cursada)
  else
    registro_cursada.codigo:=codigo_salida;
end;
procedure leer_final(var archivo_detalle2: t_archivo_detalle2; var registro_final:
t_registro_final);
begin
  if (not eof(archivo_detalle2)) then
    read(archivo_detalle2,registro_final)
  else
    registro_final.codigo:=codigo_salida;
end;
procedure minimo(var archivo_detalle1: t_archivo_detalle1; var archivo_detalle2:
t_archivo_detalle2; var registro_cursada: t_registro_cursada; var registro_final:
t_registro_final; var min: t_registro_cursada_final);
begin
  if (registro_cursada.codigo<=registro_final.codigo) then
  begin
    min.codigo:=registro_cursada.codigo;
    min.codigo_materia:=registro_cursada.codigo_materia;
    min.resultado:=registro_cursada.resultado;
    min.nota:=0;
    leer_cursada(archivo_detalle1,registro_cursada);
  end
  else
  begin
    min.codigo:=registro_final.codigo;
    min.codigo_materia:=registro_final.codigo_materia;
    min.resultado:='';
    min.nota:=registro_final.nota;
    leer_final(archivo_detalle2,registro_final);
  end;
end;
procedure actualizar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_detalle1: t_archivo_detalle1; var archivo_detalle2: t_archivo_detalle2);
var
  registro_alumno: t_registro_alumno;
  registro_cursada: t_registro_cursada;
  registro_final: t_registro_final;
  min: t_registro_cursada_final;
begin
  reset(archivo_maestro);
  reset(archivo_detalle1); reset(archivo_detalle2);
  leer_cursada(archivo_detalle1,registro_cursada);
  leer_final(archivo_detalle2,registro_final);
  minimo(archivo_detalle1,archivo_detalle2,registro_cursada,registro_final,min);
  while (min.codigo<>codigo_salida) do
  begin
    read(archivo_maestro,registro_alumno);
    while (registro_alumno.codigo<>min.codigo) do
      read(archivo_maestro,registro_alumno);
    while (registro_alumno.codigo=min.codigo) do
    begin
      if (min.resultado=resultado_corte) then
        registro_alumno.cursadas_aprobadas:=registro_alumno.cursadas_aprobadas+1;
      if (min.nota>=nota_corte) then
        registro_alumno.finales_aprobados:=registro_alumno.finales_aprobados+1;
```

```
      minimo(archivo_detalle1,archivo_detalle2,registro_cursada,registro_final,min);
    end;
    seek(archivo_maestro,filepos(archivo_maestro)-1);
    write(archivo_maestro,registro_alumno);
  end;
  close(archivo_maestro);
  close(archivo_detalle1); close(archivo_detalle2);
end;
var
  archivo_maestro: t_archivo_maestro;
  archivo_detalle1: t_archivo_detalle1;
  archivo_detalle2: t_archivo_detalle2;
  archivo_carga_maestro, archivo_carga_detalle1, archivo_carga_detalle2: text;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E7_alumnosMaestro');
assign(archivo_carga_maestro,'E7_alumnosMaestro.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,'E7_cursadasDetalle');
assign(archivo_carga_detalle1,'E7_cursadasDetalle.txt');
  cargar_archivo_detalle1(archivo_detalle1,archivo_carga_detalle1);
  imprimir_archivo_detalle1(archivo_detalle1);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE 2:'); writeln();
  assign(archivo_detalle2,'E7_finalesDetalle');
assign(archivo_carga_detalle2,'E7_finalesDetalle.txt');
  cargar_archivo_detalle2(archivo_detalle2,archivo_carga_detalle2);
  imprimir_archivo_detalle2(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 8.

Se quiere optimizar la gestión del consumo de yerba mate en distintas provincias de Argentina. Para ello, se cuenta con un archivo maestro que contiene la siguiente información: código de provincia, nombre de la provincia, cantidad de habitantes y cantidad total de kilos de yerba consumidos históricamente.

Cada mes, se reciben 16 archivos de relevamiento con información sobre el consumo de yerba en los distintos puntos del país. Cada archivo contiene: código de provincia y cantidad de kilos de yerba consumidos en ese relevamiento. Un archivo de relevamiento puede contener información de una o varias provincias, y una misma provincia puede aparecer cero, una o más veces en distintos archivos de relevamiento.

Tanto el archivo maestro como los archivos de relevamiento están ordenados por código de provincia.

Se desea realizar un programa que actualice el archivo maestro en base a la nueva información de consumo de yerba. Además, se debe informar en pantalla aquellas provincias (código y nombre) donde la cantidad total de yerba consumida supere los 10.000 kilos históricamente, junto con el promedio consumido de yerba por habitante. Es importante tener en cuenta tanto las provincias actualizadas como las que no fueron actualizadas.

Nota: Cada archivo debe recorrerse una única vez.

```
program TP2_E8;
{$codepage UTF8}
uses crt, sysutils;
const
  detalles_total=3; // detalles_total=16;
  codigo_salida=999;
  kilos_corte=10000;
type
  t_detalle=1..detalles_total;
  t_string20=string[20];
  t_registro_provincia1=record
    codigo: int16;
    nombre: t_string20;
    habitantes: int16;
    kilos: int32;
  end;
  t_registro_provincia2=record
    codigo: int16;
    kilos: int32;
  end;
  t_archivo_maestro=file of t_registro_provincia1;
  t_archivo_detalle=file of t_registro_provincia2;
  t_vector_provincias=array[t_detalle] of t_registro_provincia2;
  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_maestro: text);
var
  registro_provincia: t_registro_provincia1;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
```

```
  while (not eof(archivo_carga_maestro)) do
    with registro_provincia do
    begin
      readln(archivo_carga_maestro,codigo,habitantes,kilos,nombre); nombre:=trim(nombre);
      write(archivo_maestro,registro_provincia);
    end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var
archivo_carga_detalle: text);
var
  registro_provincia: t_registro_provincia2;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_provincia do
    begin
      readln(archivo_carga_detalle,codigo,kilos);
      write(archivo_detalle,registro_provincia);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
end;
procedure imprimir_registro_provincia1(registro_provincia: t_registro_provincia1);
begin
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_provincia.codigo);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_provincia.nombre);
  textcolor(green); write('; Habitantes: '); textcolor(yellow);
write(registro_provincia.habitantes);
  textcolor(green); write('; Kilos de yerba consumidos: '); textcolor(yellow);
writeln(registro_provincia.kilos);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_provincia: t_registro_provincia1;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_provincia);
    imprimir_registro_provincia1(registro_provincia);
  end;
  close(archivo_maestro);
end;
procedure imprimir_registro_provincia2(registro_provincia: t_registro_provincia2);
begin
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_provincia.codigo);
  textcolor(green); write('; Kilos de yerba consumidos: '); textcolor(yellow);
writeln(registro_provincia.kilos);
end;
procedure imprimir_archivo_detalle(var archivo_detalle: t_archivo_detalle);
var
  registro_provincia: t_registro_provincia2;
begin
  reset(archivo_detalle);
  while (not eof(archivo_detalle)) do
  begin
    read(archivo_detalle,registro_provincia);
    imprimir_registro_provincia2(registro_provincia);
  end;
  close(archivo_detalle);
end;
procedure leer_provincia(var archivo_detalle: t_archivo_detalle; var registro_provincia:
t_registro_provincia2);
```

```
begin
  if (not eof(archivo_detalle)) then
    read(archivo_detalle,registro_provincia)
  else
    registro_provincia.codigo:=codigo_salida;
end;
procedure minimo(var vector_detalles: t_vector_detalles; var vector_provincias:
t_vector_provincias; var min: t_registro_provincia2);
var
  i, pos: t_detalle;
begin
  min.codigo:=codigo_salida;
  for i:= 1 to detalles_total do
    if (vector_provincias[i].codigo<min.codigo) then
    begin
      min:=vector_provincias[i];
      pos:=i;
    end;
  if (min.codigo<codigo_salida) then
    leer_provincia(vector_detalles[pos],vector_provincias[pos]);
end;
procedure imprimir_texto(registro_provincia: t_registro_provincia1);
begin
  if (registro_provincia.kilos>kilos_corte) then
  begin
    textcolor(green); write('El nombre y el código de esta provincia donde la cantidad total
de yerba consumida supera los '); textcolor(yellow); write(kilos_corte); textcolor(green);
write(' kilos son '); textcolor(red); write(registro_provincia.nombre); textcolor(green);
write(' y '); textcolor(red); write(registro_provincia.codigo); textcolor(green); write(',
respectivamente, mientras que el promedio consumido de yerba por habitante es ');
textcolor(red); writeln(registro_provincia.kilos/registro_provincia.habitantes:0:2);
  end;
end;
procedure actualizar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
vector_detalles: t_vector_detalles);
var
  registro_provincia: t_registro_provincia1;
  min: t_registro_provincia2;
  vector_provincias: t_vector_provincias;
  i: t_detalle;
begin
  reset(archivo_maestro);
  for i:= 1 to detalles_total do
  begin
    reset(vector_detalles[i]);
    leer_provincia(vector_detalles[i],vector_provincias[i]);
  end;
  minimo(vector_detalles,vector_provincias,min);
  while (min.codigo<>codigo_salida) do
  begin
    read(archivo_maestro,registro_provincia);
    while (registro_provincia.codigo<>min.codigo) do
    begin
      imprimir_texto(registro_provincia);
      read(archivo_maestro,registro_provincia);
    end;
    while (registro_provincia.codigo=min.codigo) do
    begin
      registro_provincia.kilos:=registro_provincia.kilos+min.kilos;
      minimo(vector_detalles,vector_provincias,min);
    end;
    seek(archivo_maestro,filepos(archivo_maestro)-1);
    write(archivo_maestro,registro_provincia);
    imprimir_texto(registro_provincia);
  end;
  while (not eof(archivo_maestro)) do
```

```
  begin
    read(archivo_maestro,registro_provincia);
    imprimir_texto(registro_provincia);
  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_maestro;
  archivo_carga_maestro: text;
  i: t_detalle;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E8_provinciasMaestro');
assign(archivo_carga_maestro,'E8_provinciasMaestro.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],'E8_provinciasDetalle'+intToStr(i));
assign(vector_carga_detalles[i],'E8_provinciasDetalle'+intToStr(i)+'.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 9.

Se cuenta con un archivo que posee información de las ventas que realiza una empresa a los diferentes clientes. Se necesita obtener un reporte con las ventas organizadas por cliente. Para ello, se deberá informar por pantalla: los datos personales del cliente, el total mensual (mes por mes cuánto compró) y, finalmente, el monto total comprado en el año por el cliente. Además, al finalizar el reporte, se debe informar el monto total de ventas obtenido por la empresa.

El formato del archivo maestro está dado por: cliente (código cliente, nombre y apellido), año, mes, día y monto de la venta. El orden del archivo está dado por: código cliente, año y mes.

Nota: Tener en cuenta que puede haber meses en los que los clientes no realizaron compras. No es necesario informar tales meses en el reporte.

```
program TP2_E9;
{$codepage UTF8}
uses crt, sysutils;
const
  codigo_salida=999;
type
  t_string20=string[20];
  t_registro_cliente=record
    codigo: int16;
    nombre: t_string20;
    apellido: t_string20;
  end;
  t_registro_fecha=record
    anio: int16;
    mes: int8;
    dia: int8;
  end;
  t_registro_venta=record
    cliente: t_registro_cliente;
    fecha: t_registro_fecha;
    monto: real;
  end;
  t_archivo_maestro=file of t_registro_venta;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_carga_maestro: text);
var
  registro_venta: t_registro_venta;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_venta do
    begin
      readln(archivo_carga_maestro,cliente.codigo,fecha.anio,fecha.mes,fecha.dia,monto,cliente
.nombre); cliente.nombre:=trim(cliente.nombre);
      readln(archivo_carga_maestro,cliente.apellido);
cliente.apellido:=trim(cliente.apellido);
      write(archivo_maestro,registro_venta);
    end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure imprimir_registro_cliente(registro_cliente: t_registro_cliente);
begin
```

```
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_cliente.codigo);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_cliente.nombre);
  textcolor(green); write('; Apellido: '); textcolor(yellow);
write(registro_cliente.apellido);
end;
procedure imprimir_registro_fecha(registro_fecha: t_registro_fecha);
begin
  textcolor(green); write('; Fecha: '); textcolor(yellow); write(registro_fecha.anio);
  textcolor(green); write('/'); textcolor(yellow); write(registro_fecha.mes);
  textcolor(green); write('/'); textcolor(yellow); write(registro_fecha.dia);
end;
procedure imprimir_registro_venta(registro_venta: t_registro_venta);
begin
  imprimir_registro_cliente(registro_venta.cliente);
  imprimir_registro_fecha(registro_venta.fecha);
  textcolor(green); write('; Monto: '); textcolor(yellow); writeln(registro_venta.monto:0:2);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_venta: t_registro_venta;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_venta);
    imprimir_registro_venta(registro_venta);
  end;
  close(archivo_maestro);
end;
procedure leer_venta(var archivo_maestro: t_archivo_maestro; var registro_venta:
t_registro_venta);
begin
  if (not eof(archivo_maestro)) then
    read(archivo_maestro,registro_venta)
  else
    registro_venta.cliente.codigo:=codigo_salida;
end;
procedure procesar_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_venta: t_registro_venta;
  monto_total, monto_anio, monto_mes: real;
  codigo, anio, mes: int16;
begin
  monto_total:=0;
  reset(archivo_maestro);
  leer_venta(archivo_maestro,registro_venta);
  while (registro_venta.cliente.codigo<>codigo_salida) do
  begin
    codigo:=registro_venta.cliente.codigo;
    textcolor(green); write('CLIENTE: '); imprimir_registro_cliente(registro_venta.cliente);
writeln(); writeln();
    while (registro_venta.cliente.codigo=codigo) do
    begin
      anio:=registro_venta.fecha.anio;
      monto_anio:=0;
      textcolor(green); write('AÑO '); textcolor(yellow); writeln(anio);
      while ((registro_venta.cliente.codigo=codigo) and (registro_venta.fecha.anio=anio)) do
      begin
        mes:=registro_venta.fecha.mes;
        monto_mes:=0;
        while ((registro_venta.cliente.codigo=codigo) and (registro_venta.fecha.anio=anio) and
(registro_venta.fecha.mes=mes)) do
        begin
          monto_mes:=monto_mes+registro_venta.monto;
          leer_venta(archivo_maestro,registro_venta);
        end;
```

```
        textcolor(green); write('Monto total del mes '); textcolor(yellow); write(mes);
textcolor(green); write(': $'); textcolor(red); writeln(monto_mes:0:2);
        monto_anio:=monto_anio+monto_mes;
      end;
      textcolor(green); write('Monto total del año '); textcolor(yellow); write(anio);
textcolor(green); write(': $'); textcolor(red); writeln(monto_anio:0:2); writeln();
      monto_total:=monto_total+monto_anio;
    end;
  end;
  textcolor(green); write('Monto total de ventas obtenido por la empresa: $'); textcolor(red);
writeln(monto_total:0:2);
  close(archivo_maestro);
end;
var
  archivo_maestro: t_archivo_maestro;
  archivo_carga_maestro: text;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E9_ventasMaestro');
assign(archivo_carga_maestro,'E9_ventasMaestro.txt');
  cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
  imprimir_archivo_maestro(archivo_maestro);
  writeln(); textcolor(red); writeln('PROCESAMIENTO ARCHIVO MAESTRO:'); writeln();
  procesar_archivo_maestro(archivo_maestro);
end.
```

Ejercicio 10.

Se necesita contabilizar los votos de las diferentes mesas electorales registradas por provincia y localidad. Para ello, se posee un archivo con la siguiente información: código de provincia, código de localidad, número de mesa y cantidad de votos en dicha mesa. Presentar en pantalla un listado como se muestra a continuación:

Código de Provincia

Código de Localidad Total de Votos

....

....

Total de Votos Provincia: ____

Código de Provincia

Código de Localidad Total de Votos

....

Total de Votos Provincia: ___

.....

Total General de Votos: ___

Nota: La información está ordenada por código de provincia y código de localidad.

```
program TP2_E10;
{$codepage UTF8}
uses crt, sysutils;
const
  provincia_salida=999;
type
  t_registro_mesa=record
    provincia: int16;
    localidad: int16;
    mesa: int16;
    votos: int32;
  end;
  t_archivo_maestro=file of t_registro_mesa;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_carga_maestro: text);
var
  registro_mesa: t_registro_mesa;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_mesa do
    begin
      readln(archivo_carga_maestro,provincia,localidad,mesa,votos);
      write(archivo_maestro,registro_mesa);
    end;
```

```
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure imprimir_registro_mesa(registro_mesa: t_registro_mesa);
begin
  textcolor(green); write('Código de provincia: '); textcolor(yellow);
write(registro_mesa.provincia);
  textcolor(green); write('; Código de localidad: '); textcolor(yellow);
write(registro_mesa.localidad);
  textcolor(green); write('; Número de mesa: '); textcolor(yellow); write(registro_mesa.mesa);
  textcolor(green); write('; Votos: '); textcolor(yellow); writeln(registro_mesa.votos);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_mesa: t_registro_mesa;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_mesa);
    imprimir_registro_mesa(registro_mesa);
  end;
  close(archivo_maestro);
end;
procedure leer_votos(var archivo_maestro: t_archivo_maestro; var registro_mesa:
t_registro_mesa);
begin
  if (not eof(archivo_maestro)) then
    read(archivo_maestro,registro_mesa)
  else
    registro_mesa.provincia:=provincia_salida;
end;
procedure procesar_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_mesa: t_registro_mesa;
  provincia, localidad: int16;
  votos_total, votos_provincia, votos_localidad: int32;
begin
  votos_total:=0;
  reset(archivo_maestro);
  leer_votos(archivo_maestro,registro_mesa);
  while (registro_mesa.provincia<>provincia_salida) do
  begin
    provincia:=registro_mesa.provincia;
    votos_provincia:=0;
    textcolor(green); write('Código de Provincia: '); textcolor(yellow); writeln(provincia);
    textcolor(green); writeln('Código de Localidad          Total de Votos');
    while (registro_mesa.provincia=provincia) do
    begin
      localidad:=registro_mesa.localidad;
      votos_localidad:=0;
      while ((registro_mesa.provincia=provincia) and (registro_mesa.localidad=localidad)) do
      begin
        votos_localidad:=votos_localidad+registro_mesa.votos;
        leer_votos(archivo_maestro,registro_mesa);
      end;
      textcolor(yellow); write(localidad); textcolor(green);
write('                            '); textcolor(red); writeln(votos_localidad);
      votos_provincia:=votos_provincia+votos_localidad;
    end;
    textcolor(green); write('Total de Votos Provincia: '); textcolor(red);
writeln(votos_provincia); writeln();
    votos_total:=votos_total+votos_provincia;
  end;
  textcolor(green); write('Total General de Votos: '); textcolor(red); writeln(votos_total);
  close(archivo_maestro);
```

```
end;
var
  archivo_maestro: t_archivo_maestro;
  archivo_carga_maestro: text;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E10_mesasMaestro');
assign(archivo_carga_maestro,'E10_mesasMaestro.txt');
  cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
  imprimir_archivo_maestro(archivo_maestro);
  writeln(); textcolor(red); writeln('PROCESAMIENTO ARCHIVO MAESTRO:'); writeln();
  procesar_archivo_maestro(archivo_maestro);
end.
```

Ejercicio 11.

Se tiene información en un archivo de las horas extras realizadas por los empleados de una empresa en un mes. Para cada empleado, se tiene la siguiente información: departamento, división, número de empleado, categoría y cantidad de horas extras realizadas por el empleado. Se sabe que el archivo se encuentra ordenado por departamento, luego por división y, por último, por número de empleado. Presentar en pantalla un listado con el siguiente formato:

Departamento

División

Número de Empleado Total de Hs. Importe a cobrar

....

....

Total de horas división: ____

Monto total por división: ____

División

.....

Total horas departamento: ____

Monto total departamento: ____

Para obtener el valor de la hora, se debe cargar un arreglo desde un archivo de texto al iniciar el programa con el valor de la hora extra para cada categoría. La categoría varía de 1 a 15. En el archivo de texto, debe haber una línea para cada categoría con el número de categoría y el valor de la hora, pero el arreglo debe ser de valores de horas, con la posición del valor coincidente con el número de categoría.

```
program TP2_E11;
{$codepage UTF8}
uses crt;
const
  departamento_salida=999;
  categoria_ini=1; categoria_fin=15;
type
  t_categoria=categoria_ini..categoria_fin;
  t_registro_empleado=record
    departamento: int16;
    division: int16;
    empleado: int16;
    categoria: t_categoria;
    horas: int16;
  end;
  t_vector_horas=array[t_categoria] of real;
  t_archivo_maestro=file of t_registro_empleado;
```

```
procedure cargar_vector_horas(var vector_horas: t_vector_horas; var archivo_carga_vector:
text);
var
  categoria: t_categoria;
  valor_hora: real;
begin
  reset(archivo_carga_vector);
  while (not eof(archivo_carga_vector)) do
  begin
    readln(archivo_carga_vector,categoria,valor_hora);
    vector_horas[categoria]:=valor_hora;
  end;
  close(archivo_carga_vector);
end;
procedure imprimir_vector_horas(vector_horas: t_vector_horas);
var
  i: t_categoria;
begin
  for i:= categoria_ini to categoria_fin do
  begin
    textcolor(green); write('El valor de la hora de la categoría ',i,' es ');
textcolor(yellow); writeln(vector_horas[i]:0:2);
  end;
end;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_carga_maestro: text);
var
  registro_empleado: t_registro_empleado;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
  begin
    with registro_empleado do
    begin
      readln(archivo_carga_maestro,departamento,division,empleado,categoria,horas);
      write(archivo_maestro,registro_empleado);
    end;
  end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure imprimir_registro_empleado(registro_empleado: t_registro_empleado);
begin
  textcolor(green); write('Departamento: '); textcolor(yellow);
write(registro_empleado.departamento);
  textcolor(green); write('; División: '); textcolor(yellow);
write(registro_empleado.division);
  textcolor(green); write('; Número de empleado: '); textcolor(yellow);
write(registro_empleado.empleado);
  textcolor(green); write('; Categoría: '); textcolor(yellow);
write(registro_empleado.categoria);
  textcolor(green); write('; Horas extras: '); textcolor(yellow);
writeln(registro_empleado.horas);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_empleado: t_registro_empleado;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_empleado);
    imprimir_registro_empleado(registro_empleado);
  end;
  close(archivo_maestro);
```

```
end;
procedure leer_empleado(var archivo_maestro: t_archivo_maestro; var registro_empleado:
t_registro_empleado);
begin
  if (not eof(archivo_maestro)) then
    read(archivo_maestro,registro_empleado)
  else
    registro_empleado.departamento:=departamento_salida;
end;
procedure procesar_archivo_maestro(var archivo_maestro: t_archivo_maestro; vector_horas:
t_vector_horas);
var
  registro_empleado: t_registro_empleado;
  categoria: t_categoria;
  departamento, division, empleado, horas_departamento, horas_division, horas_empleado: int16;
  monto_departamento, monto_division, monto_empleado: real;
begin
  reset(archivo_maestro);
  leer_empleado(archivo_maestro,registro_empleado);
  while (registro_empleado.departamento<>departamento_salida) do
  begin
    departamento:=registro_empleado.departamento;
    horas_departamento:=0; monto_departamento:=0;
    textcolor(green); write('Departamento: '); textcolor(yellow); writeln(departamento);
    while (registro_empleado.departamento=departamento) do
    begin
      division:=registro_empleado.division;
      horas_division:=0; monto_division:=0;
      textcolor(green); write('División: '); textcolor(yellow); writeln(division);
      textcolor(green); writeln('Número de Empleado          Total de Hs.          Importe a
cobrar');
      while ((registro_empleado.departamento=departamento) and
(registro_empleado.division=division)) do
      begin
        empleado:=registro_empleado.empleado; categoria:=registro_empleado.categoria;
        horas_empleado:=0; monto_empleado:=0;
        while ((registro_empleado.departamento=departamento) and
(registro_empleado.division=division) and (registro_empleado.empleado=empleado)) do
        begin
          horas_empleado:=horas_empleado+registro_empleado.horas;
          leer_empleado(archivo_maestro,registro_empleado);
        end;
        monto_empleado:=horas_empleado*vector_horas[categoria];
        textcolor(yellow); write(empleado); textcolor(green);
write('                          '); textcolor(red); write(horas_empleado); textcolor(green);
write('                    $'); textcolor(red); writeln(monto_empleado:0:2);
        horas_division:=horas_division+horas_empleado;
        monto_division:=monto_division+monto_empleado;
      end;
      textcolor(green); write('Total horas división: '); textcolor(red);
writeln(horas_division);
      textcolor(green); write('Monto total división: $'); textcolor(red);
writeln(monto_division:0:2);
      horas_departamento:=horas_departamento+horas_division;
      monto_departamento:=monto_departamento+monto_division;
    end;
    textcolor(green); write('Total horas departamento: '); textcolor(red);
writeln(horas_departamento);
    textcolor(green); write('Monto total departamento: $'); textcolor(red);
writeln(monto_departamento:0:2); writeln();
  end;
end;
var
  vector_horas: t_vector_horas;
  archivo_maestro: t_archivo_maestro;
  archivo_carga_vector, archivo_carga_maestro: text;
```

```
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN VECTOR HORAS:'); writeln();
  assign(archivo_carga_vector,'E11_horasVector.txt');
  cargar_vector_horas(vector_horas,archivo_carga_vector);
  imprimir_vector_horas(vector_horas);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E11_empleadosMaestro');
assign(archivo_carga_maestro,'E11_empleadosMaestro.txt');
  cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
  imprimir_archivo_maestro(archivo_maestro);
  writeln(); textcolor(red); writeln('PROCESAMIENTO ARCHIVO MAESTRO:'); writeln();
  procesar_archivo_maestro(archivo_maestro,vector_horas);
end.
```

Ejercicio 12.

La empresa de software "X" posee un servidor web donde se encuentra alojado el sitio web de la organización. En dicho servidor, se almacenan, en un archivo, todos los accesos que se realizan al sitio. La información que se almacena en el archivo es la siguiente: año, mes, día, idUsuario y tiempo de acceso al sitio de la organización. El archivo se encuentra ordenado por los siguientes criterios: año, mes, día e idUsuario.

Se debe realizar un procedimiento que genere un informe en pantalla. Para ello, se indicará el año calendario sobre el cual debe realizar el informe. El mismo debe respetar el formato mostrado a continuación:

```
Año : ---
    Mes:-- 1
        día:-- 1
              idUsuario 1   Tiempo Total de acceso en el dia 1 mes 1
              --------
              idUsuario N    Tiempo total de acceso en el dia 1 mes 1
           Tiempo total acceso dia 1 mes 1
     -------------
     día N
              idUsuario 1   Tiempo Total de acceso en el dia N mes 1
              --------
              idUsuario N    Tiempo total de acceso en el dia N mes 1
           Tiempo total acceso dia N mes 1
     Total tiempo de acceso mes 1
     ------
     Mes 12
     día 1
              idUsuario 1   Tiempo Total de acceso en el dia 1 mes 12
              --------
              idUsuario N    Tiempo total de acceso en el dia 1 mes 12
           Tiempo total acceso dia 1 mes 12
     -------------

    día N
             idUsuario 1   Tiempo Total de acceso en el dia N mes 12
                --------
             idUsuario N    Tiempo total de acceso en el dia N mes 12
          Tiempo total acceso dia N mes 12
    Total tiempo de acceso mes 12
Total tiempo de acceso año
```

Se deberá tener en cuenta las siguientes aclaraciones:

- *El año sobre el cual realizará el informe de accesos debe leerse desde el teclado.*
- *El año puede no existir en el archivo, en tal caso debe informarse en pantalla "Año no encontrado".*

- *Se debe definir las estructuras de datos necesarias.*
- *El recorrido del archivo debe realizarse una única vez, procesando sólo la información necesaria.*

```
program TP2_E12;
{$codepage UTF8}
uses crt;
const
  usuario_salida=999;
  anio_ini=2020; anio_fin=2025;
type
  t_anio=anio_ini..anio_fin;
  t_registro_fecha=record
    anio: int16;
    mes: int8;
    dia: int8;
  end;
  t_registro_usuario=record
    fecha: t_registro_fecha;
    usuario: int16;
    tiempo: real;
  end;
  t_archivo_maestro=file of t_registro_usuario;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_carga_maestro: text);
var
  registro_usuario: t_registro_usuario;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_usuario do
    begin
      readln(archivo_carga_maestro,fecha.anio,fecha.mes,fecha.dia,usuario,tiempo);
      write(archivo_maestro,registro_usuario);
    end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure imprimir_registro_fecha(registro_fecha: t_registro_fecha);
begin
  textcolor(green); write('Fecha: '); textcolor(yellow); write(registro_fecha.anio);
  textcolor(green); write('/'); textcolor(yellow); write(registro_fecha.mes);
  textcolor(green); write('/'); textcolor(yellow); write(registro_fecha.dia);
end;
procedure imprimir_registro_usuario(registro_usuario: t_registro_usuario);
begin
  imprimir_registro_fecha(registro_usuario.fecha);
  textcolor(green); write('; ID usuario: '); textcolor(yellow);
write(registro_usuario.usuario);
  textcolor(green); write('; Tiempo de acceso: '); textcolor(yellow);
writeln(registro_usuario.tiempo:0:2);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_usuario: t_registro_usuario;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_usuario);
    imprimir_registro_usuario(registro_usuario);
  end;
```

```
  close(archivo_maestro);
end;
procedure leer_acceso(var archivo_maestro: t_archivo_maestro; var registro_usuario:
t_registro_usuario);
begin
  if (not eof(archivo_maestro)) then
    read(archivo_maestro,registro_usuario)
  else
    registro_usuario.usuario:=usuario_salida;
end;
procedure procesar_archivo_maestro(var archivo_maestro: t_archivo_maestro; anio: t_anio);
var
  registro_usuario: t_registro_usuario;
  mes, dia, usuario: int16;
  tiempo_anio, tiempo_mes, tiempo_dia, tiempo_usuario: real;
begin
  reset(archivo_maestro);
  leer_acceso(archivo_maestro,registro_usuario);
  while ((registro_usuario.usuario<>usuario_salida) and (registro_usuario.fecha.anio<>anio))
do
    leer_acceso(archivo_maestro,registro_usuario);
  if (registro_usuario.usuario<>usuario_salida) then
  begin
    tiempo_anio:=0;
    textcolor(green); write('Año: '); textcolor(yellow); writeln(anio); writeln();
    while (registro_usuario.fecha.anio=anio) do
    begin
      mes:=registro_usuario.fecha.mes;
      tiempo_mes:=0;
      textcolor(green); write('  Mes: '); textcolor(yellow); writeln(mes);
      while ((registro_usuario.fecha.anio=anio) and (registro_usuario.fecha.mes=mes)) do
      begin
        dia:=registro_usuario.fecha.dia;
        tiempo_dia:=0;
        textcolor(green); write('    Día: '); textcolor(yellow); writeln(dia);
        textcolor(green); write('      idUsuario          Tiempo Total de acceso en el día ');
textcolor(yellow); write(dia); textcolor(green); write(' mes '); textcolor(yellow);
writeln(mes);
        while ((registro_usuario.fecha.anio=anio) and (registro_usuario.fecha.mes=mes) and
(registro_usuario.fecha.dia=dia)) do
        begin
          usuario:=registro_usuario.usuario;
          tiempo_usuario:=0;
          while ((registro_usuario.fecha.anio=anio) and (registro_usuario.fecha.mes=mes) and
(registro_usuario.fecha.dia=dia) and (registro_usuario.usuario=usuario)) do
          begin
            tiempo_usuario:=tiempo_usuario+registro_usuario.tiempo;
            leer_acceso(archivo_maestro,registro_usuario);
          end;
          textcolor(green); write('      '); textcolor(yellow); write(usuario);
textcolor(green); write('                  '); textcolor(red); writeln(tiempo_usuario:0:2);
          tiempo_dia:=tiempo_dia+tiempo_usuario;
        end;
        textcolor(green); write('    Tiempo total acceso día '); textcolor(yellow);
write(dia); textcolor(green); write(' mes '); textcolor(yellow); write(mes); textcolor(green);
write(': '); textcolor(red); writeln(tiempo_dia:0:2);
        tiempo_mes:=tiempo_mes+tiempo_dia;
      end;
      textcolor(green); write('  Tiempo total acceso mes '); textcolor(yellow); write(mes);
textcolor(green); write(': '); textcolor(red); writeln(tiempo_mes:0:2); writeln();
      tiempo_anio:=tiempo_anio+tiempo_mes;
    end;
    textcolor(green); write('Tiempo total acceso año '); textcolor(yellow); write(anio);
textcolor(green); write(': '); textcolor(red); writeln(tiempo_anio:0:2);
  end
  else
```

```
  begin
    textcolor(green); write('Año '); textcolor(yellow); write(anio); textcolor(green);
writeln(' no encontrado');
  end;
  close(archivo_maestro);
end;
var
  archivo_maestro: t_archivo_maestro;
  archivo_carga_maestro: text;
  anio: t_anio;
begin
  randomize;
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E12_usuariosMaestro');
assign(archivo_carga_maestro,'E12_usuariosMaestro.txt');
  cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
  imprimir_archivo_maestro(archivo_maestro);
  writeln(); textcolor(red); writeln('PROCESAMIENTO ARCHIVO MAESTRO:'); writeln();
  anio:=anio_ini+random(anio_fin-anio_ini+1);
  procesar_archivo_maestro(archivo_maestro,anio);
end.
```

Ejercicio 13.

Suponer que se es administrador de un servidor de correo electrónico. En los logs del mismo (información guardada acerca de los movimientos que ocurren en el server) que se encuentran en la siguiente ruta: /var/log/logmail.dat se guarda la siguiente información: nro_usuario, nombreUsuario, nombre, apellido, cantidadMailEnviados. Diariamente, el servidor de correo genera un archivo con la siguiente información: nro_usuario, cuentaDestino, cuerpoMensaje. Este archivo representa todos los correos enviados por los usuarios en un día determinado. Ambos archivos están ordenados por nro_usuario y se sabe que un usuario puede enviar cero, uno o más mails por día.

(a) *Realizar el procedimiento necesario para actualizar la información del log en un día particular. Definir las estructuras de datos que utilice el procedimiento.*

(b) *Generar un archivo de texto que contenga el siguiente informe dado un archivo detalle de un día determinado:*

nro_usuarioX……..cantidadMensajesEnviados

……

nro_usuarioX+n……..cantidadMensajesEnviados

Nota: Tener en cuenta que, en el listado, deberán aparecer todos los usuarios que existen en el sistema. Considerar la implementación de esta opción de las siguientes maneras:

- *Como un procedimiento separado del inciso (a).*
- *En el mismo procedimiento de actualización del inciso (a). ¿Qué cambios se requieren en el procedimiento del inciso (a) para realizar el informe en el mismo recorrido?*

Ejercicio 14.

Una compañía aérea dispone de un archivo maestro donde guarda información sobre sus próximos vuelos. En dicho archivo, se tiene almacenado el destino, fecha, hora de salida y la cantidad de asientos disponibles. La empresa recibe todos los días dos archivos detalles para actualizar el archivo maestro. En dichos archivos, se tiene destino, fecha, hora de salida y cantidad de asientos comprados. Se sabe que los archivos están ordenados por destino más fecha y hora de salida, y que, en los detalles, pueden venir 0, 1 o más registros por cada uno del maestro. Se pide realizar los módulos necesarios para:

(a) *Actualizar el archivo maestro sabiendo que no se registró ninguna venta de pasaje sin asiento disponible.*

(b) *Generar una lista con aquellos vuelos (destino, fecha y hora de salida) que tengan menos de una cantidad específica de asientos disponibles. La misma debe ser ingresada por teclado.*

Nota: El archivo maestro y los archivos detalles sólo pueden recorrerse una vez.

Ejercicio 15.

Ejercicio 16.

Ejercicio 17.

Ejercicio 18.

Ejercicio 19.