Introducción a las Bases de Datos

Fundamentos de Organización de Datos

Práctica 1

Creación, consulta y mantenimiento de archivos secuenciales - Algorítmica Básica.

1. Realizar un algoritmo que cree un archivo de números enteros no ordenados y permita incorporar datos al archivo. Los números son ingresados desde teclado. La carga finaliza cuando se ingresa el número 30000, que no debe incorporarse al archivo. El nombre del archivo debe ser proporcionado por el usuario desde teclado.

2. Realizar un algoritmo, que utilizando el archivo de números enteros no ordenados creado en el ejercicio 1, informe por pantalla cantidad de números menores a 1500 y el promedio de los números ingresados. El nombre del archivo a procesar debe ser proporcionado por el usuario una única vez. Además, el algoritmo deberá listar el contenido del archivo en pantalla.

3. Realizar un programa que presente un menú con opciones para:

    a. Crear un archivo de registros **no ordenados** de empleados y completarlo con datos ingresados desde teclado. De cada empleado se registra: número de empleado, apellido, nombre, edad y DNI. Algunos empleados se ingresan con DNI 00. La carga finaliza cuando se ingresa el String 'fin' como apellido.

    b. Abrir el archivo anteriormente generado y

        i. Listar en pantalla los datos de empleados que tengan un nombre o apellido determinado, el cual se proporciona desde el teclado.

ii. Listar en pantalla los empleados de a uno por línea.

iii. Listar en pantalla los empleados mayores de 70 años, próximos a jubilarse.

NOTA: El nombre del archivo a crear o utilizar debe ser proporcionado por el usuario.

4. Agregar al menú del programa del ejercicio 3, opciones para:

   a. Añadir uno o más empleados al final del archivo con sus datos ingresados por teclado. Tener en cuenta que no se debe agregar al archivo un empleado con un número de empleado ya registrado (control de unicidad).

   b. Modificar la edad de un empleado dado.

   c. Exportar el contenido del archivo a un archivo de texto llamado "todos_empleados.txt".

   d. Exportar a un archivo de texto llamado: "faltaDNIEmpleado.txt", los empleados que no tengan cargado el DNI (DNI en 00).

   NOTA: Las búsquedas deben realizarse por número de empleado.

5. Realizar un programa para una tienda de celulares, que presente un menú con opciones para:

   a. Crear un archivo de registros no ordenados de celulares y cargarlo con datos ingresados desde un archivo de texto denominado "*celulares.txt*". Los registros correspondientes a los celulares deben contener: código de celular, nombre, descripción, marca, precio, stock mínimo y stock disponible.

   b. Listar en pantalla los datos de aquellos celulares que tengan un stock menor al stock mínimo.

   c. Listar en pantalla los celulares del archivo cuya descripción contenga una cadena de caracteres proporcionada por el usuario.

   d. Exportar el archivo creado en el inciso a) a un archivo de texto denominado "*celulares.txt*" con todos los celulares del mismo. El archivo de texto generado

podría ser utilizado en un futuro como archivo de carga (ver inciso a), por lo que debería respetar el formato dado para este tipo de archivos en la NOTA 2.

NOTA 1: El nombre del archivo binario de celulares debe ser proporcionado por el usuario.

NOTA 2: El archivo de carga debe editarse de manera que cada celular se especifique en tres líneas consecutivas. En la primera se especifica: código de celular, el precio y marca, en la segunda el stock disponible, stock mínimo y la descripción y en la tercera nombre en ese orden. Cada celular se carga leyendo tres líneas del archivo "*celulares.txt*".

6. Agregar al menú del programa del ejercicio 5, opciones para:
    a. Añadir uno o más celulares al final del archivo con sus datos ingresados por teclado.
    b. Modificar el stock de un celular dado.
    c. Exportar el contenido del archivo binario a un archivo de texto denominado: "*Sin*Stock.txt", con aquellos celulares que tengan stock 0.

NOTA: Las búsquedas deben realizarse por nombre de celular.

7. Realizar un programa que permita:
    a) Crear un archivo binario a partir de la información almacenada en un archivo de texto. El nombre del archivo de texto es: "novelas.txt". La información en el archivo de texto consiste en: código de novela, nombre, género y precio de diferentes novelas argentinas. Los datos de cada novela se almacenan en dos líneas en el archivo de texto. La primera línea contendrá la siguiente información: código novela, precio y género, y la segunda línea almacenará el nombre de la novela.

b) Abrir el archivo binario y permitir la actualización del mismo. Se debe poder agregar una novela y modificar una existente. Las búsquedas se realizan por código de novela.

NOTA: El nombre del archivo binario es proporcionado por el usuario desde el teclado.

IMPORTANTE: Se recomienda implementar los ejercicios prácticos en Dev-Pascal. El ejecutable puede descargarse desde la plataforma Moodle.

Juan Menduiña

Trabajo Práctico N° 1:
Creación, Consulta y Mantenimiento de Archivos Secuenciales - Algorítmica Básica.

Ejercicio 1.

Realizar un algoritmo que cree un archivo de números enteros no ordenados y permita incorporar datos al archivo. Los números son ingresados desde teclado. La carga finaliza cuando se ingresa el número 30000, que no debe incorporarse al archivo. El nombre del archivo debe ser proporcionado por el usuario desde teclado.

```
program TP1_E1;
{$codepage UTF8}
uses crt;
const
  num_salida=30000;
type
  t_archivo_enteros=file of int16;
procedure leer_numero(var num: int16);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    num:=num_salida
  else
    num:=random(high(int16));
end;
procedure cargar_archivo_enteros(var archivo_enteros: t_archivo_enteros);
var
  num: int16;
begin
  rewrite(archivo_enteros);
  textcolor(green); write('Los números ingresados son: ');
  leer_numero(num);
  while (num<>num_salida) do
  begin
    textcolor(yellow); write(num,' ');
    write(archivo_enteros,num);
    leer_numero(num);
  end;
  close(archivo_enteros);
end;
var
  archivo_enteros: t_archivo_enteros;
begin
  randomize;
  assign(archivo_enteros,'E1_enteros');
  cargar_archivo_enteros(archivo_enteros);
end.
```

Ejercicio 2.

Realizar un algoritmo que, utilizando el archivo de números enteros no ordenados creado en el Ejercicio 1, informe por pantalla cantidad de números menores a 1500 y el promedio de los números ingresados. El nombre del archivo a procesar debe ser proporcionado por el usuario una única vez. Además, el algoritmo deberá listar el contenido del archivo en pantalla.

```
program TP1_E2;
{$codepage UTF8}
uses crt;
const
  num_corte=1500;
type
  t_archivo_enteros=file of int16;
procedure procesar_archivo_enteros(var archivo_enteros: t_archivo_enteros; var nums_corte:
int16; var prom: real);
var
  num: int16;
  suma: real;
begin
  reset(archivo_enteros);
  suma:=0;
  textcolor(green); write('El contenido del archivo es: ');
  while (not eof(archivo_enteros)) do
  begin
    read(archivo_enteros,num);
    textcolor(yellow); write(num,' ');
    if (num<num_corte) then
      nums_corte:=nums_corte+1;
    suma:=suma+num;
  end;
  if (fileSize(archivo_enteros)>0) then
    prom:=suma/fileSize(archivo_enteros);
  writeln();
  close(archivo_enteros);
end;
var
  archivo_enteros: t_archivo_enteros;
  nums_corte: int16;
  prom: real;
begin
  nums_corte:=0; prom:=0;
  assign(archivo_enteros,'E1_enteros');
  procesar_archivo_enteros(archivo_enteros,nums_corte,prom);
  textcolor(green); write('La cantidad de números menores a '); textcolor(yellow);
write(num_corte); textcolor(green); write(' es '); textcolor(red); writeln(nums_corte);
  textcolor(green); write('El promedio de los números ingresados es '); textcolor(red);
write(prom:0:2);
end.
```

Ejercicio 3.

Realizar un programa que presente un menú con opciones para:

(a) *Crear un archivo de registros no ordenados de empleados y completarlo con datos ingresados desde teclado. De cada empleado, se registra: número de empleado, apellido, nombre, edad y DNI. Algunos empleados se ingresan con DNI 00. La carga finaliza cuando se ingresa el String "fin" como apellido.*

(b) *Abrir el archivo anteriormente generado y:*

> **(i)** *Listar en pantalla los datos de empleados que tengan un nombre o apellido determinado, el cual se proporciona desde el teclado.*
> **(ii)** *Listar en pantalla los empleados de a uno por línea.*
> **(iii)** *Listar en pantalla los empleados mayores de 70 años, próximos a jubilarse.*

Nota: El nombre del archivo a crear o utilizar debe ser proporcionado por el usuario.

```
program TP1_E3;
{$codepage UTF8}
uses crt;
const
  apellido_salida='fin';
  edad_corte=70;
  opcion_salida=0;
type
  t_string10=string[10];
  t_registro_empleado=record
    numero: int16;
    apellido: t_string10;
    nombre: t_string10;
    edad: int8;
    dni: int32;
  end;
  t_archivo_empleados=file of t_registro_empleado;
function random_string(length: int8): t_string10;
var
  i: int8;
  string_aux: string;
begin
  string_aux:='';
  for i:= 1 to length do
    string_aux:=string_aux+chr(ord('A')+random(26));
  random_string:=string_aux;
end;
procedure leer_empleado(var registro_empleado: t_registro_empleado);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_empleado.apellido:=apellido_salida
  else
    registro_empleado.apellido:=random_string(5+random(5));
  if (registro_empleado.apellido<>apellido_salida) then
  begin
    registro_empleado.numero:=1+random(1000);
    registro_empleado.nombre:=random_string(5+random(5));
    registro_empleado.edad:=18+random(high(int8)-18);
```

```
    if (i<=10) then
      registro_empleado.dni:=0
    else
      registro_empleado.dni:=10000000+random(40000001);
  end;
end;
procedure cargar_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
  registro_empleado: t_registro_empleado;
begin
  rewrite(archivo_empleados);
  leer_empleado(registro_empleado);
  while (registro_empleado.apellido<>apellido_salida) do
  begin
    write(archivo_empleados,registro_empleado);
    leer_empleado(registro_empleado);
  end;
  close(archivo_empleados);
  textcolor(green); writeln('El archivo binario de empleados fue creado y cargado con éxito');
end;
procedure imprimir_registro_empleado(registro_empleado: t_registro_empleado);
begin
  textcolor(green); write('Número: '); textcolor(yellow); write(registro_empleado.numero);
  textcolor(green); write('; Apellido: '); textcolor(yellow);
write(registro_empleado.apellido);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_empleado.nombre);
  textcolor(green); write('; Edad: '); textcolor(yellow); write(registro_empleado.edad);
  textcolor(green); write('; DNI: '); textcolor(yellow); writeln(registro_empleado.dni);
end;
procedure imprimir1_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
  registro_empleado: t_registro_empleado;
  texto: t_string10;
begin
  texto:=random_string(5+random(5));
  reset(archivo_empleados);
  textcolor(green); write('Los datos de los empleados con nombre o apellido ');
textcolor(yellow); write(texto); textcolor(green); writeln(' son: ');
  while (not eof(archivo_empleados)) do
  begin
    read(archivo_empleados,registro_empleado);
    if ((registro_empleado.nombre=texto) or (registro_empleado.apellido=texto)) then
      imprimir_registro_empleado(registro_empleado);
  end;
  close(archivo_empleados);
end;
procedure imprimir2_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
  registro_empleado: t_registro_empleado;
begin
  reset(archivo_empleados);
  textcolor(green); writeln('Los empleados del archivo son: ');
  while (not eof(archivo_empleados)) do
  begin
    read(archivo_empleados,registro_empleado);
    imprimir_registro_empleado(registro_empleado);
  end;
  close(archivo_empleados);
end;
procedure imprimir3_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
  registro_empleado: t_registro_empleado;
begin
  reset(archivo_empleados);
  textcolor(green); write('Los empleados mayores a '); textcolor(yellow); write(edad_corte);
textcolor(green); writeln(' años son: ');
```

```
  while (not eof(archivo_empleados)) do
  begin
    read(archivo_empleados,registro_empleado);
    if (registro_empleado.edad>edad_corte) then
      imprimir_registro_empleado(registro_empleado);
  end;
  close(archivo_empleados);
end;
procedure leer_opcion(var opcion: int8);
begin
  textcolor(red); writeln('MENÚ DE OPCIONES');
  textcolor(yellow); write('OPCIÓN 1: '); textcolor(green); writeln('Crear un archivo de
registros no ordenados de empleados y completarlo con datos ingresados desde teclado');
  textcolor(yellow); write('OPCIÓN 2: '); textcolor(green); writeln('Listar en pantalla los
datos de empleados que tengan un nombre o apellido determinado');
  textcolor(yellow); write('OPCIÓN 3: '); textcolor(green); writeln('Listar en pantalla los
empleados de a uno por línea');
  textcolor(yellow); write('OPCIÓN 4: '); textcolor(green); writeln('Listar en pantalla los
empleados mayores a 70 años, próximos a jubilarse');
  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_empleados: t_archivo_empleados);
var
  opcion: int8;
begin
  leer_opcion(opcion);
  while (opcion<>opcion_salida) do
  begin
    case opcion of
      1: cargar_archivo_empleados(archivo_empleados);
      2: imprimir1_archivo_empleados(archivo_empleados);
      3: imprimir2_archivo_empleados(archivo_empleados);
      4: imprimir3_archivo_empleados(archivo_empleados);
    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_empleados: t_archivo_empleados;
begin
  randomize;
  assign(archivo_empleados,'E3_empleados');
  menu_opciones(archivo_empleados);
end.
```

Ejercicio 4.

Agregar al menú del programa del Ejercicio 3 opciones para:

(a) *Añadir uno o más empleados al final del archivo con sus datos ingresados por teclado. Tener en cuenta que no se debe agregar al archivo un empleado con un número de empleado ya registrado (control de unicidad).*

(b) *Modificar la edad de un empleado dado.*

(c) *Exportar el contenido del archivo a un archivo de texto llamado "todos_empleados.txt".*

(d) *Exportar a un archivo de texto llamado "faltaDNIEmpleado.txt" los empleados que no tengan cargado el DNI (DNI en 00).*

Nota: Las búsquedas deben realizarse por número de empleado.

```
program TP1_E4;
{$codepage UTF8}
uses crt;
const
  apellido_salida='fin';
  edad_corte=70;
  dni_corte=0;
  opcion_salida=0;
type
  t_string10=string[10];
  t_registro_empleado=record
    numero: int16;
    apellido: t_string10;
    nombre: t_string10;
    edad: int8;
    dni: int32;
  end;
  t_archivo_empleados=file of t_registro_empleado;
function random_string(length: int8): t_string10;
var
  i: int8;
  string_aux: string;
begin
  string_aux:='';
  for i:= 1 to length do
    string_aux:=string_aux+chr(ord('A')+random(26));
  random_string:=string_aux;
end;
procedure leer_empleado(var registro_empleado: t_registro_empleado);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_empleado.apellido:=apellido_salida
  else
    registro_empleado.apellido:=random_string(5+random(5));
  if (registro_empleado.apellido<>apellido_salida) then
  begin
    registro_empleado.numero:=1+random(1000);
    registro_empleado.nombre:=random_string(5+random(5));
```

```
    registro_empleado.edad:=18+random(high(int8)-18);
    if (i<=10) then
      registro_empleado.dni:=0
    else
      registro_empleado.dni:=10000000+random(40000001);
  end;
end;
procedure cargar_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
  registro_empleado: t_registro_empleado;
begin
  rewrite(archivo_empleados);
  leer_empleado(registro_empleado);
  while (registro_empleado.apellido<>apellido_salida) do
  begin
    write(archivo_empleados,registro_empleado);
    leer_empleado(registro_empleado);
  end;
  close(archivo_empleados);
  textcolor(green); writeln('El archivo binario de empleados fue creado y cargado con éxito');
end;
procedure imprimir_registro_empleado(registro_empleado: t_registro_empleado);
begin
  textcolor(green); write('Número: '); textcolor(yellow); write(registro_empleado.numero);
  textcolor(green); write('; Apellido: '); textcolor(yellow);
write(registro_empleado.apellido);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_empleado.nombre);
  textcolor(green); write('; Edad: '); textcolor(yellow); write(registro_empleado.edad);
  textcolor(green); write('; DNI: '); textcolor(yellow); writeln(registro_empleado.dni);
end;
procedure imprimir1_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
  registro_empleado: t_registro_empleado;
  texto: t_string10;
begin
  texto:=random_string(5+random(5));
  reset(archivo_empleados);
  textcolor(green); write('Los datos de los empleados con nombre o apellido ');
textcolor(yellow); write(texto); textcolor(green); writeln(' son: ');
  while (not eof(archivo_empleados)) do
  begin
    read(archivo_empleados,registro_empleado);
    if ((registro_empleado.nombre=texto) or (registro_empleado.apellido=texto)) then
      imprimir_registro_empleado(registro_empleado);
  end;
  close(archivo_empleados);
end;
procedure imprimir2_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
  registro_empleado: t_registro_empleado;
begin
  reset(archivo_empleados);
  textcolor(green); writeln('Los empleados del archivo son: ');
  while (not eof(archivo_empleados)) do
  begin
    read(archivo_empleados,registro_empleado);
    imprimir_registro_empleado(registro_empleado);
  end;
  close(archivo_empleados);
end;
procedure imprimir3_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
  registro_empleado: t_registro_empleado;
begin
  reset(archivo_empleados);
```

```
  textcolor(green); write('Los empleados mayores a '); textcolor(yellow); write(edad_corte);
textcolor(green); writeln(' años son: ');
  while (not eof(archivo_empleados)) do
  begin
    read(archivo_empleados,registro_empleado);
    if (registro_empleado.edad>edad_corte) then
      imprimir_registro_empleado(registro_empleado);
  end;
  close(archivo_empleados);
end;
function control_unicidad(var archivo_empleados: t_archivo_empleados; numero: int16): boolean;
var
  registro_empleado: t_registro_empleado;
  ok: boolean;
begin
  ok:=false;
  while ((not eof(archivo_empleados)) and (ok=false)) do
  begin
    read(archivo_empleados,registro_empleado);
    if (registro_empleado.numero=numero) then
      ok:=true;
  end;
  control_unicidad:=ok;
end;
procedure agregar_empleado(var archivo_empleados: t_archivo_empleados);
var
  registro_empleado: t_registro_empleado;
  empleados: int16;
begin
  empleados:=0;
  reset(archivo_empleados);
  leer_empleado(registro_empleado);
  while (registro_empleado.apellido<>apellido_salida) do
  begin
    if (control_unicidad(archivo_empleados,registro_empleado.numero)=false) then
    begin
      seek(archivo_empleados,filesize(archivo_empleados));
      write(archivo_empleados,registro_empleado);
      empleados:=empleados+1;
    end;
    leer_empleado(registro_empleado);
  end;
  close(archivo_empleados);
  textcolor(green); write('Se han agregado '); textcolor(yellow); write(empleados);
textcolor(green); writeln(' empleados al final del archivo');
end;
procedure modificar_edad_empleado(var archivo_empleados: t_archivo_empleados);
var
  registro_empleado: t_registro_empleado;
  numero: int16;
  ok: boolean;
begin
  numero:=1+random(1000);
  ok:=false;
  reset(archivo_empleados);
  while ((not eof(archivo_empleados)) and (ok=false)) do
  begin
    read(archivo_empleados,registro_empleado);
    if (registro_empleado.numero=numero) then
    begin
      registro_empleado.edad:=18+random(high(int8)-18);
      seek(archivo_empleados,filepos(archivo_empleados)-1);
      write(archivo_empleados,registro_empleado);
      ok:=true;
    end;
  end;
```

```
  close(archivo_empleados);
  if (ok=true) then
  begin
    textcolor(green); write('Se ha modificado la edad del empleado con número ');
textcolor(yellow); write(numero); textcolor(green); writeln(' en el archivo');
  end
  else
  begin
    textcolor(green); write('No se ha encontrado el empleado con número '); textcolor(yellow);
write(numero); textcolor(green); writeln(' en el archivo');
  end;
end;
procedure exportar_archivo_txt1(var archivo_empleados: t_archivo_empleados);
var
  registro_empleado: t_registro_empleado;
  archivo_txt: text;
begin
  reset(archivo_empleados);
  assign(archivo_txt,'E4_todos_empleados.txt'); rewrite(archivo_txt);
  while (not eof(archivo_empleados)) do
  begin
    read(archivo_empleados,registro_empleado);
    with registro_empleado do
      writeln(archivo_txt,'Número: ',numero,'; Apellido: ',apellido,'; Nombre: ',nombre,';
Edad: ',edad,'; DNI: ',dni);
  end;
  close(archivo_empleados);
  close(archivo_txt);
  textcolor(green); write('Se ha exportado el contenido del archivo al archivo de texto
llamado '); textcolor(yellow); writeln('"todos_empleados.txt"');
end;
procedure exportar_archivo_txt2(var archivo_empleados: t_archivo_empleados);
var
  registro_empleado: t_registro_empleado;
  archivo_txt: text;
begin
  reset(archivo_empleados);
  assign(archivo_txt,'E4_faltaDNIEmpleado.txt'); rewrite(archivo_txt);
  while (not eof(archivo_empleados)) do
  begin
    read(archivo_empleados,registro_empleado);
    if (registro_empleado.dni=dni_corte) then
      with registro_empleado do
        writeln(archivo_txt,'Número: ',numero,'; Apellido: ',apellido,'; Nombre: ',nombre,';
Edad: ',edad,'; DNI: ',dni);
  end;
  close(archivo_empleados);
  close(archivo_txt);
  textcolor(green); write('Se ha exportado a un archivo de texto llamado ');
textcolor(yellow); write('"faltaDNIEmpleado.txt"'); textcolor(green); writeln(' los empleados
que no tienen cargado el DNI (DNI en 00)');
end;
procedure leer_opcion(var opcion: int8);
begin
  textcolor(red); writeln('MENÚ DE OPCIONES');
  textcolor(yellow); write('OPCIÓN 1: '); textcolor(green); writeln('Crear un archivo de
registros no ordenados de empleados y completarlo con datos ingresados desde teclado');
  textcolor(yellow); write('OPCIÓN 2: '); textcolor(green); writeln('Listar en pantalla los
datos de empleados que tengan un nombre o apellido determinado');
  textcolor(yellow); write('OPCIÓN 3: '); textcolor(green); writeln('Listar en pantalla los
empleados de a uno por línea');
  textcolor(yellow); write('OPCIÓN 4: '); textcolor(green); writeln('Listar en pantalla los
empleados mayores a 70 años, próximos a jubilarse');
  textcolor(yellow); write('OPCIÓN 5: '); textcolor(green); writeln('Añadir uno o más
empleados al final del archivo con sus datos ingresados por teclado');
```

```
  textcolor(yellow); write('OPCIÓN 6: '); textcolor(green); writeln('Modificar la edad de un
empleado dado');
  textcolor(yellow); write('OPCIÓN 7: '); textcolor(green); writeln('Exportar el contenido del
archivo a un archivo de texto llamado “todos_empleados.txt”');
  textcolor(yellow); write('OPCIÓN 8: '); textcolor(green); writeln('Exportar a un archivo de
texto llamado “faltaDNIEmpleado.txt” los empleados que no tengan cargado el DNI (DNI en 00)');
  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_empleados: t_archivo_empleados);
var
  opcion: int8;
begin
  leer_opcion(opcion);
  while (opcion<>opcion_salida) do
  begin
    case opcion of
      1: cargar_archivo_empleados(archivo_empleados);
      2: imprimir1_archivo_empleados(archivo_empleados);
      3: imprimir2_archivo_empleados(archivo_empleados);
      4: imprimir3_archivo_empleados(archivo_empleados);
      5: agregar_empleado(archivo_empleados);
      6: modificar_edad_empleado(archivo_empleados);
      7: exportar_archivo_txt1(archivo_empleados);
      8: exportar_archivo_txt2(archivo_empleados);
    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_empleados: t_archivo_empleados;
begin
  randomize;
  assign(archivo_empleados,'E4_empleados');
  menu_opciones(archivo_empleados);
end.
```

Ejercicio 5.

Realizar un programa para una tienda de celulares, que presente un menú con opciones para:

(a) *Crear un archivo de registros no ordenados de celulares y cargarlo con datos ingresados desde un archivo de texto denominado "celulares.txt". Los registros correspondientes a los celulares deben contener: código de celular, nombre, descripción, marca, precio, stock mínimo y stock disponible.*

(b) *Listar en pantalla los datos de aquellos celulares que tengan un stock menor al stock mínimo.*

(c) *Listar en pantalla los celulares del archivo cuya descripción contenga una cadena de caracteres proporcionada por el usuario.*

(d) *Exportar el archivo creado en el inciso (a) a un archivo de texto denominado "celulares.txt" con todos los celulares del mismo. El archivo de texto generado podría ser utilizado en un futuro como archivo de carga (ver inciso (a)), por lo que debería respetar el formato dado para este tipo de archivos en la Nota 2.*

Nota 1: El nombre del archivo binario de celulares debe ser proporcionado por el usuario.

Nota 2: El archivo de carga debe editarse de manera que cada celular se especifique en tres líneas consecutivas. En la primera, se especifica: código de celular, precio y marca; en la segunda, stock disponible, stock mínimo y descripción; y, en la tercera, nombre en ese orden. Cada celular se carga leyendo tres líneas del archivo "celulares.txt".

```
program TP1_E5;
{$codepage UTF8}
uses crt, sysutils;
const
  codigo_salida=0;
  opcion_salida=0;
type
  t_string20=string[20];
  t_registro_celular=record
    codigo: int16;
    nombre: t_string20;
    descripcion: t_string20;
    marca: t_string20;
    precio: real;
    stock_minimo: int16;
    stock_disponible: int16;
  end;
  t_archivo_celulares=file of t_registro_celular;
function random_string(length: int8): t_string20;
var
  i: int8;
  string_aux: string;
begin
  string_aux:='';
  for i:= 1 to length do
    string_aux:=string_aux+chr(ord('A')+random(26));
  random_string:=string_aux;
```

```
end;
procedure leer_celular(var registro_celular: t_registro_celular);
var
  vector_marcas: array[1..10] of t_string20=('Alcatel', 'Apple', 'Huawei', 'Lenovo', 'LG',
'Motorola', 'Nokia', 'Samsung', 'Sony', 'Xiaomi');
  vector_descripciones: array[1..5] of t_string20=('Gama baja', 'Gama media baja', 'Gama
media', 'Gama media alta', 'Gama alta');
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_celular.codigo:=codigo_salida
  else
    registro_celular.codigo:=1+random(1000);
  if (registro_celular.codigo<>codigo_salida) then
  begin
    registro_celular.nombre:=random_string(5+random(5));
    registro_celular.descripcion:=vector_descripciones[1+random(5)];
    registro_celular.marca:=vector_marcas[1+random(10)];
    registro_celular.precio:=100+random(9001)/10;
    registro_celular.stock_minimo:=1+random(10);
    registro_celular.stock_disponible:=random(101);
  end;
end;
procedure cargar_archivo_carga(var archivo_carga: text);
var
  registro_celular: t_registro_celular;
begin
  rewrite(archivo_carga);
  leer_celular(registro_celular);
  while (registro_celular.codigo<>codigo_salida) do
  begin
    with registro_celular do
    begin
      writeln(archivo_carga,codigo,' ',precio:0:2,' ',marca);
      writeln(archivo_carga,stock_disponible,' ',stock_minimo,' ',descripcion);
      writeln(archivo_carga,nombre);
    end;
    leer_celular(registro_celular);
  end;
  close(archivo_carga);
end;
procedure cargar_archivo_celulares(var archivo_celulares: t_archivo_celulares; var
archivo_carga: text);
var
  registro_celular: t_registro_celular;
begin
  rewrite(archivo_celulares);
  reset(archivo_carga);
  while (not eof(archivo_carga)) do
    with registro_celular do
    begin
      readln(archivo_carga,codigo,precio,marca); marca:=trim(marca);
      readln(archivo_carga,stock_disponible,stock_minimo,descripcion);
descripcion:=trim(descripcion);
      readln(archivo_carga,nombre);
      write(archivo_celulares,registro_celular);
    end;
  close(archivo_celulares);
  close(archivo_carga);
  textcolor(green); writeln('El archivo binario de celulares fue creado y cargado con éxito');
end;
procedure imprimir_registro_celular(registro_celular: t_registro_celular);
begin
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_celular.codigo);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_celular.nombre);
```

```
  textcolor(green); write('; Descripción: '); textcolor(yellow);
write(registro_celular.descripcion);
  textcolor(green); write('; Marca: '); textcolor(yellow); write(registro_celular.marca);
  textcolor(green); write('; Precio: '); textcolor(yellow);
write(registro_celular.precio:0:2);
  textcolor(green); write('; Stock mínimo: '); textcolor(yellow);
write(registro_celular.stock_minimo);
  textcolor(green); write('; Stock disponible: '); textcolor(yellow);
writeln(registro_celular.stock_disponible);
end;
procedure imprimir1_archivo_celulares(var archivo_celulares: t_archivo_celulares);
var
  registro_celular: t_registro_celular;
begin
  reset(archivo_celulares);
  textcolor(green); writeln('Los datos de aquellos celulares que tienen un stock menor al
stock mínimo son: ');
  while (not eof(archivo_celulares)) do
  begin
    read(archivo_celulares,registro_celular);
    if (registro_celular.stock_disponible<registro_celular.stock_minimo) then
      imprimir_registro_celular(registro_celular);
  end;
  close(archivo_celulares);
end;
procedure imprimir2_archivo_celulares(var archivo_celulares: t_archivo_celulares);
var
  registro_celular: t_registro_celular;
  vector_descripciones: array[1..5] of t_string20=('Gama baja', 'Gama media baja', 'Gama
media', 'Gama media alta', 'Gama alta');
  descripcion: t_string20;
begin
  reset(archivo_celulares);
  descripcion:=vector_descripciones[1+random(5)];
  textcolor(green); write('Los celulares del archivo cuya descripción contiene la cadena de
caracteres '); textcolor(yellow); write(descripcion); textcolor(green); writeln(' son: ');
  while (not eof(archivo_celulares)) do
  begin
    read(archivo_celulares,registro_celular);
    if (registro_celular.descripcion=descripcion) then
      imprimir_registro_celular(registro_celular);
  end;
  close(archivo_celulares);
end;
procedure exportar_archivo_txt(var archivo_celulares: t_archivo_celulares);
var
  registro_celular: t_registro_celular;
  archivo_txt: text;
begin
  reset(archivo_celulares);
  assign(archivo_txt,'E5_celulares2.txt'); rewrite(archivo_txt);
  while (not eof(archivo_celulares)) do
  begin
    read(archivo_celulares,registro_celular);
    with registro_celular do
    begin
      writeln(archivo_txt,codigo,' ',precio:0:2,' ',marca);
      writeln(archivo_txt,stock_disponible,' ',stock_minimo,' ',descripcion);
      writeln(archivo_txt,nombre);
    end;
  end;
  close(archivo_celulares);
  close(archivo_txt);
  textcolor(green); write('Se ha exportado el archivo creado en el inciso (a) a un archivo de
texto denominado '); textcolor(yellow); write('"celulares2.txt"'); textcolor(green); writeln('
con todos los celulares del mismo');
```

```
end;
procedure leer_opcion(var opcion: int8);
begin
  textcolor(red); writeln('MENÚ DE OPCIONES');
  textcolor(yellow); write('OPCIÓN 1: '); textcolor(green); writeln('Crear un archivo de
registros no ordenados de celulares y cargarlo con datos ingresados desde un archivo de texto
denominado "celulares1.txt"');
  textcolor(yellow); write('OPCIÓN 2: '); textcolor(green); writeln('Listar en pantalla los
datos de aquellos celulares que tengan un stock menor al stock mínimo');
  textcolor(yellow); write('OPCIÓN 3: '); textcolor(green); writeln('Listar en pantalla los
celulares del archivo cuya descripción contenga una cadena de caracteres proporcionada por el
usuario');
  textcolor(yellow); write('OPCIÓN 4: '); textcolor(green); writeln('Exportar el archivo
creado en el inciso (a) a un archivo de texto denominado "celulares2.txt" con todos los
celulares del mismo');
  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_celulares: t_archivo_celulares; var archivo_carga: text);
var
  opcion: int8;
begin
  leer_opcion(opcion);
  while (opcion<>opcion_salida) do
  begin
    case opcion of
      1: cargar_archivo_celulares(archivo_celulares,archivo_carga);
      2: imprimir1_archivo_celulares(archivo_celulares);
      3: imprimir2_archivo_celulares(archivo_celulares);
      4: exportar_archivo_txt(archivo_celulares);
    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_celulares: t_archivo_celulares;
  archivo_carga: text;
begin
  randomize;
  assign(archivo_carga,'E5_celulares1.txt');
  assign(archivo_celulares,'E5_celulares2');
  cargar_archivo_carga(archivo_carga);
  menu_opciones(archivo_celulares,archivo_carga);
end.
```

Ejercicio 6.

Agregar al menú del programa del Ejercicio 5 opciones para:

(a) *Añadir uno o más celulares al final del archivo con sus datos ingresados por teclado.*

(b) *Modificar el stock de un celular dado.*

(c) *Exportar el contenido del archivo binario a un archivo de texto denominado "SinStock.txt" con aquellos celulares que tengan stock 0.*

```
program TP1_E6;
{$codepage UTF8}
uses crt, sysutils;
const
  codigo_salida=0;
  stock_disponible_corte=0;
  opcion_salida=0;
type
  t_string20=string[20];
  t_registro_celular=record
    codigo: int16;
    nombre: t_string20;
    descripcion: t_string20;
    marca: t_string20;
    precio: real;
    stock_minimo: int16;
    stock_disponible: int16;
  end;
  t_archivo_celulares=file of t_registro_celular;
function random_string(length: int8): t_string20;
var
  i: int8;
  string_aux: string;
begin
  string_aux:='';
  for i:= 1 to length do
    string_aux:=string_aux+chr(ord('A')+random(26));
  random_string:=string_aux;
end;
procedure leer_celular(var registro_celular: t_registro_celular);
var
  vector_marcas: array[1..10] of t_string20=('Alcatel', 'Apple', 'Huawei', 'Lenovo', 'LG',
'Motorola', 'Nokia', 'Samsung', 'Sony', 'Xiaomi');
  vector_descripciones: array[1..5] of t_string20=('Gama baja', 'Gama media baja', 'Gama
media', 'Gama media alta', 'Gama alta');
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_celular.codigo:=codigo_salida
  else
    registro_celular.codigo:=1+random(1000);
  if (registro_celular.codigo<>codigo_salida) then
  begin
    registro_celular.nombre:=random_string(5+random(5));
    registro_celular.descripcion:=vector_descripciones[1+random(5)];
    registro_celular.marca:=vector_marcas[1+random(10)];
    registro_celular.precio:=100+random(9001)/10;
    registro_celular.stock_minimo:=1+random(10);
    registro_celular.stock_disponible:=random(101);
```

```
  end;
end;
procedure cargar_archivo_carga(var archivo_carga: text);
var
  registro_celular: t_registro_celular;
begin
  rewrite(archivo_carga);
  leer_celular(registro_celular);
  while (registro_celular.codigo<>codigo_salida) do
  begin
    with registro_celular do
    begin
      writeln(archivo_carga,codigo,' ',precio:0:2,' ',marca);
      writeln(archivo_carga,stock_disponible,' ',stock_minimo,' ',descripcion);
      writeln(archivo_carga,nombre);
    end;
    leer_celular(registro_celular);
  end;
  close(archivo_carga);
end;
procedure cargar_archivo_celulares(var archivo_celulares: t_archivo_celulares; var
archivo_carga: text);
var
  registro_celular: t_registro_celular;
begin
  rewrite(archivo_celulares);
  reset(archivo_carga);
  while (not eof(archivo_carga)) do
    with registro_celular do
    begin
      readln(archivo_carga,codigo,precio,marca); marca:=trim(marca);
      readln(archivo_carga,stock_disponible,stock_minimo,descripcion);
descripcion:=trim(descripcion);
      readln(archivo_carga,nombre);
      write(archivo_celulares,registro_celular);
    end;
  close(archivo_celulares);
  close(archivo_carga);
  textcolor(green); writeln('El archivo binario de celulares fue creado y cargado con éxito');
end;
procedure imprimir_registro_celular(registro_celular: t_registro_celular);
begin
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_celular.codigo);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_celular.nombre);
  textcolor(green); write('; Descripción: '); textcolor(yellow);
write(registro_celular.descripcion);
  textcolor(green); write('; Marca: '); textcolor(yellow); write(registro_celular.marca);
  textcolor(green); write('; Precio: '); textcolor(yellow);
write(registro_celular.precio:0:2);
  textcolor(green); write('; Stock mínimo: '); textcolor(yellow);
write(registro_celular.stock_minimo);
  textcolor(green); write('; Stock disponible: '); textcolor(yellow);
writeln(registro_celular.stock_disponible);
end;
procedure imprimir1_archivo_celulares(var archivo_celulares: t_archivo_celulares);
var
  registro_celular: t_registro_celular;
begin
  reset(archivo_celulares);
  textcolor(green); writeln('Los datos de aquellos celulares que tienen un stock menor al
stock mínimo son: ');
  while (not eof(archivo_celulares)) do
  begin
    read(archivo_celulares,registro_celular);
    if (registro_celular.stock_disponible<registro_celular.stock_minimo) then
      imprimir_registro_celular(registro_celular);
```

```
  end;
  close(archivo_celulares);
end;
procedure imprimir2_archivo_celulares(var archivo_celulares: t_archivo_celulares);
var
  registro_celular: t_registro_celular;
  vector_descripciones: array[1..5] of t_string20=('Gama baja', 'Gama media baja', 'Gama
media', 'Gama media alta', 'Gama alta');
  descripcion: t_string20;
begin
  reset(archivo_celulares);
  descripcion:=vector_descripciones[1+random(5)];
  textcolor(green); write('Los celulares del archivo cuya descripción contiene la cadena de
caracteres '); textcolor(yellow); write(descripcion); textcolor(green); writeln(' son: ');
  while (not eof(archivo_celulares)) do
  begin
    read(archivo_celulares,registro_celular);
    if (registro_celular.descripcion=descripcion) then
      imprimir_registro_celular(registro_celular);
  end;
  close(archivo_celulares);
end;
procedure exportar_archivo_txt1(var archivo_celulares: t_archivo_celulares);
var
  registro_celular: t_registro_celular;
  archivo_txt: text;
begin
  reset(archivo_celulares);
  assign(archivo_txt,'E6_celulares2.txt'); rewrite(archivo_txt);
  while (not eof(archivo_celulares)) do
  begin
    read(archivo_celulares,registro_celular);
    with registro_celular do
    begin
      writeln(archivo_txt,codigo,' ',precio:0:2,' ',marca);
      writeln(archivo_txt,stock_disponible,' ',stock_minimo,' ',descripcion);
      writeln(archivo_txt,nombre);
    end;
  end;
  close(archivo_celulares);
  close(archivo_txt);
  textcolor(green); write('Se ha exportado el archivo creado en el inciso (a) a un archivo de
texto denominado '); textcolor(yellow); write('"celulares2.txt"'); textcolor(green); writeln('
con todos los celulares del mismo');
end;
function control_unicidad(var archivo_celulares: t_archivo_celulares; codigo: int16): boolean;
var
  registro_celular: t_registro_celular;
  ok: boolean;
begin
  ok:=false;
  while ((not eof(archivo_celulares)) and (ok=false)) do
  begin
    read(archivo_celulares,registro_celular);
    if (registro_celular.codigo=codigo) then
      ok:=true;
  end;
  control_unicidad:=ok;
end;
procedure agregar_celular(var archivo_celulares: t_archivo_celulares);
var
  registro_celular: t_registro_celular;
  celulares: int16;
begin
  celulares:=0;
  reset(archivo_celulares);
```

```
  leer_celular(registro_celular);
  while (registro_celular.codigo<>codigo_salida) do
  begin
    if (control_unicidad(archivo_celulares,registro_celular.codigo)=false) then
    begin
      seek(archivo_celulares,filesize(archivo_celulares));
      write(archivo_celulares,registro_celular);
      celulares:=celulares+1;
    end;
    leer_celular(registro_celular);
  end;
  close(archivo_celulares);
  textcolor(green); write('Se han agregado '); textcolor(yellow); write(celulares);
textcolor(green); writeln(' celulares al final del archivo');
end;
procedure modificar_stock_celular(var archivo_celulares: t_archivo_celulares);
var
  registro_celular: t_registro_celular;
  codigo: int16;
  ok: boolean;
begin
  codigo:=1+random(1000);
  ok:=false;
  reset(archivo_celulares);
  while ((not eof(archivo_celulares)) and (ok=false)) do
  begin
    read(archivo_celulares,registro_celular);
    if (registro_celular.codigo=codigo) then
    begin
      registro_celular.stock_disponible:=random(101);
      seek(archivo_celulares,filepos(archivo_celulares)-1);
      write(archivo_celulares,registro_celular);
      ok:=true;
    end;
  end;
  close(archivo_celulares);
  if (ok=true) then
  begin
    textcolor(green); write('Se ha modificado el stock del celular con código ');
textcolor(yellow); write(codigo); textcolor(green); writeln(' en el archivo');
  end
  else
  begin
    textcolor(green); write('No se ha encontrado el celular con código '); textcolor(yellow);
write(codigo); textcolor(green); writeln(' en el archivo');
  end;
end;
procedure exportar_archivo_txt2(var archivo_celulares: t_archivo_celulares);
var
  registro_celular: t_registro_celular;
  archivo_txt: text;
begin
  reset(archivo_celulares);
  assign(archivo_txt,'E6_SinStock.txt'); rewrite(archivo_txt);
  while (not eof(archivo_celulares)) do
  begin
    read(archivo_celulares,registro_celular);
    if (registro_celular.stock_disponible=stock_disponible_corte) then
      with registro_celular do
      begin
        writeln(archivo_txt,codigo,' ',precio:0:2,' ',marca);
        writeln(archivo_txt,stock_disponible,' ',stock_minimo,' ',descripcion);
        writeln(archivo_txt,nombre);
      end;
  end;
  close(archivo_celulares);
```

```
  close(archivo_txt);
  textcolor(green); write('Se ha exportado el contenido del archivo binario a un archivo de
texto denominado '); textcolor(yellow); write('"SinStock.txt"'); textcolor(green); writeln('
con aquellos celulares que tienen stock 0');
end;
procedure leer_opcion(var opcion: int8);
begin
  textcolor(red); writeln('MENÚ DE OPCIONES');
  textcolor(yellow); write('OPCIÓN 1: '); textcolor(green); writeln('Crear un archivo de
registros no ordenados de celulares y cargarlo con datos ingresados desde un archivo de texto
denominado "celulares1.txt"');
  textcolor(yellow); write('OPCIÓN 2: '); textcolor(green); writeln('Listar en pantalla los
datos de aquellos celulares que tengan un stock menor al stock mínimo');
  textcolor(yellow); write('OPCIÓN 3: '); textcolor(green); writeln('Listar en pantalla los
celulares del archivo cuya descripción contenga una cadena de caracteres proporcionada por el
usuario');
  textcolor(yellow); write('OPCIÓN 4: '); textcolor(green); writeln('Exportar el archivo
creado en el inciso (a) a un archivo de texto denominado "celulares2.txt" con todos los
celulares del mismo');
  textcolor(yellow); write('OPCIÓN 5: '); textcolor(green); writeln('Añadir uno o más
celulares al final del archivo con sus datos ingresados por teclado');
  textcolor(yellow); write('OPCIÓN 6: '); textcolor(green); writeln('Modificar el stock de un
celular dado');
  textcolor(yellow); write('OPCIÓN 7: '); textcolor(green); writeln('Exportar el contenido del
archivo binario a un archivo de texto denominado "SinStock.txt" con aquellos celulares que
tengan stock 0');
  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_celulares: t_archivo_celulares; var archivo_carga: text);
var
  opcion: int8;
begin
  leer_opcion(opcion);
  while (opcion<>opcion_salida) do
  begin
    case opcion of
      1: cargar_archivo_celulares(archivo_celulares,archivo_carga);
      2: imprimir1_archivo_celulares(archivo_celulares);
      3: imprimir2_archivo_celulares(archivo_celulares);
      4: exportar_archivo_txt1(archivo_celulares);
      5: agregar_celular(archivo_celulares);
      6: modificar_stock_celular(archivo_celulares);
      7: exportar_archivo_txt2(archivo_celulares);
    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_celulares: t_archivo_celulares;
  archivo_carga: text;
begin
  randomize;
  assign(archivo_carga,'E6_celulares1.txt');
  assign(archivo_celulares,'E6_celulares2');
  cargar_archivo_carga(archivo_carga);
  menu_opciones(archivo_celulares,archivo_carga);
end.
```

Ejercicio 7.

Realizar un programa que permita:

(a) *Crear un archivo binario a partir de la información almacenada en un archivo de texto. El nombre del archivo de texto es: "novelas.txt". La información en el archivo de texto consiste en: código de novela, nombre, género y precio de diferentes novelas argentinas. Los datos de cada novela se almacenan en dos líneas en el archivo de texto. La primera línea contendrá la siguiente información: código novela, precio y género; y la segunda línea almacenará el nombre de la novela.*

(b) *Abrir el archivo binario y permitir la actualización del mismo. Se debe poder agregar una novela y modificar una existente. Las búsquedas se realizan por código de novela.*

Nota: El nombre del archivo binario es proporcionado por el usuario desde el teclado.

```
program TP1_E7;
{$codepage UTF8}
uses crt, sysutils;
const
  codigo_salida=0;
  opcion_salida=0;
type
  t_string20=string[20];
  t_registro_novela=record
    codigo: int16;
    nombre: t_string20;
    genero: t_string20;
    precio: real;
  end;
  t_archivo_novelas=file of t_registro_novela;
function random_string(length: int8): t_string20;
var
  i: int8;
  string_aux: string;
begin
  string_aux:='';
  for i:= 1 to length do
    string_aux:=string_aux+chr(ord('A')+random(26));
  random_string:=string_aux;
end;
procedure leer_novela(var registro_novela: t_registro_novela; ok: boolean);
var
  i: int8;
begin
  if (ok=true) then
  begin
    i:=random(100);
    if (i=0) then
      registro_novela.codigo:=codigo_salida
    else
      registro_novela.codigo:=1+random(1000);
  end;
  if (registro_novela.codigo<>codigo_salida) then
  begin
    registro_novela.nombre:=random_string(5+random(15));
    registro_novela.genero:=random_string(5+random(15));
    registro_novela.precio:=100+random(9001)/10;
  end;
end;
```

```
procedure cargar_archivo_carga(var archivo_carga: text);
var
  registro_novela: t_registro_novela;
begin
  rewrite(archivo_carga);
  leer_novela(registro_novela,true);
  while (registro_novela.codigo<>codigo_salida) do
  begin
    with registro_novela do
    begin
      writeln(archivo_carga,codigo,' ',precio:0:2,' ',genero);
      writeln(archivo_carga,nombre);
    end;
    leer_novela(registro_novela,true);
  end;
  close(archivo_carga);
end;
procedure cargar_archivo_novelas(var archivo_novelas: t_archivo_novelas; var archivo_carga:
text);
var
  registro_novela: t_registro_novela;
begin
  rewrite(archivo_novelas);
  reset(archivo_carga);
  while (not eof(archivo_carga)) do
  begin
    with registro_novela do
    begin
      readln(archivo_carga,codigo,precio,genero); genero:=trim(genero);
      readln(archivo_carga,nombre);
    end;
    write(archivo_novelas,registro_novela);
  end;
  close(archivo_novelas);
  close(archivo_carga);
  textcolor(green); writeln('El archivo binario de novelas fue creado y cargado con éxito');
end;
function control_unicidad(var archivo_novelas: t_archivo_novelas; codigo: int16): boolean;
var
  registro_novela: t_registro_novela;
  ok: boolean;
begin
  ok:=false;
  while ((not eof(archivo_novelas)) and (ok=false)) do
  begin
    read(archivo_novelas,registro_novela);
    if (registro_novela.codigo=codigo) then
      ok:=true;
  end;
  control_unicidad:=ok;
end;
procedure agregar_novela(var archivo_novelas: t_archivo_novelas);
var
  registro_novela: t_registro_novela;
  novelas: int16;
begin
  novelas:=0;
  reset(archivo_novelas);
  leer_novela(registro_novela,true);
  while (registro_novela.codigo<>codigo_salida) do
  begin
    if (control_unicidad(archivo_novelas,registro_novela.codigo)=false) then
    begin
      seek(archivo_novelas,filesize(archivo_novelas));
      write(archivo_novelas,registro_novela);
      novelas:=novelas+1;
```

```
    end;
    leer_novela(registro_novela,true);
  end;
  close(archivo_novelas);
  textcolor(green); write('Se han agregado '); textcolor(yellow); write(novelas);
textcolor(green); writeln(' novelas al final del archivo');
end;
procedure modificar_novela(var archivo_novelas: t_archivo_novelas);
var
  registro_novela: t_registro_novela;
  codigo: int16;
  ok: boolean;
begin
  codigo:=1+random(1000);
  ok:=false;
  reset(archivo_novelas);
  while ((not eof(archivo_novelas)) and (ok=false)) do
  begin
    read(archivo_novelas,registro_novela);
    if (registro_novela.codigo=codigo) then
    begin
      leer_novela(registro_novela,false);
      seek(archivo_novelas,filepos(archivo_novelas)-1);
      write(archivo_novelas,registro_novela);
      ok:=true;
    end;
  end;
  close(archivo_novelas);
  if (ok=true) then
  begin
    textcolor(green); write('Se ha modificado la novela con código '); textcolor(yellow);
write(codigo); textcolor(green); writeln(' en el archivo');
  end
  else
  begin
    textcolor(green); write('No se ha encontrado la novela con código '); textcolor(yellow);
write(codigo); textcolor(green); writeln(' en el archivo');
  end;
end;
procedure exportar_archivo_txt(var archivo_novelas: t_archivo_novelas);
var
  registro_novela: t_registro_novela;
  archivo_txt: text;
begin
  reset(archivo_novelas);
  assign(archivo_txt,'E7_novelas2.txt'); rewrite(archivo_txt);
  while (not eof(archivo_novelas)) do
  begin
    read(archivo_novelas,registro_novela);
    with registro_novela do
    begin
      writeln(archivo_txt,codigo,' ',precio:0:2,' ',genero);
      writeln(archivo_txt,nombre);
    end;
  end;
  close(archivo_novelas);
  close(archivo_txt);
  textcolor(green); write('Se ha exportado el archivo creado en el inciso (a) a un archivo de
texto denominado '); textcolor(yellow); write('"novelas2.txt"'); textcolor(green); writeln('
con todas las novelas del mismo');
end;
procedure leer_opcion(var opcion: int8);
begin
  textcolor(red); writeln('MENÚ DE OPCIONES');
  textcolor(yellow); write('OPCIÓN 1: '); textcolor(green); writeln('Crear un archivo binario
a partir de la información almacenada en un archivo de texto denominado "novelas.txt"');
```

```
  textcolor(yellow); write('OPCIÓN 2: '); textcolor(green); writeln('Añadir una o más novelas
al final del archivo con sus datos ingresados por teclado');
  textcolor(yellow); write('OPCIÓN 3: '); textcolor(green); writeln('Modificar una novela
existente');
  textcolor(yellow); write('OPCIÓN 4: '); textcolor(green); writeln('Exportar el archivo
creado en el inciso (a) a un archivo de texto denominado "novelas2.txt" con todas las novelas
del mismo');
  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_novelas: t_archivo_novelas; var archivo_carga: text);
var
  opcion: int8;
begin
  leer_opcion(opcion);
  while (opcion<>opcion_salida) do
  begin
    case opcion of
      1: cargar_archivo_novelas(archivo_novelas,archivo_carga);
      2: agregar_novela(archivo_novelas);
      3: modificar_novela(archivo_novelas);
      4: exportar_archivo_txt(archivo_novelas);
    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_novelas: t_archivo_novelas;
  archivo_carga: text;
begin
  randomize;
  assign(archivo_carga,'E7_novelas1.txt');
  assign(archivo_novelas,'E7_novelas2');
  cargar_archivo_carga(archivo_carga);
  menu_opciones(archivo_novelas,archivo_carga);
end.
```

Introducción a las Bases de Datos

Fundamentos de Organización de Datos

Práctica 2

Archivos Secuenciales ordenados - Algorítmica Clásica

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.

   Realice 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.

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 ó 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.

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 ó más registros por cada provincia.

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.

5. Suponga que 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. 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.

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:

1. Al número de fallecidos se le suman el valor de fallecidos recibido del detalle.
2. Idem anterior para los recuperados.
3. Los casos activos se actualizan con el valor recibido en el detalle.
4. Idem anterior para los casos nuevos hallados.

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

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 (solo 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.

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.

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

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

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.

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.

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

13. Suponga que usted 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 encuentra 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. Realice el procedimiento necesario para actualizar la información del log en un día particular. Defina las estructuras de datos que utilice su procedimiento.
    b. Genere 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. Considere la implementación de esta opción de las siguientes maneras:

            i- Como un procedimiento separado del punto a).

            ii- En el mismo procedimiento de actualización del punto a). Qué cambios se requieren en el procedimiento del punto a) para realizar el informe en el mismo recorrido?

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 ó 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 y 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.

15. Se desea modelar la información de una ONG dedicada a la asistencia de personas con carencias habitacionales. La ONG cuenta con un archivo maestro conteniendo información como se indica a continuación: Código pcia, nombre provincia, código de localidad, nombre de localidad, #viviendas sin luz, #viviendas sin gas, #viviendas de chapa, #viviendas sin agua, # viviendas sin sanitarios.

    Mensualmente reciben detalles de las diferentes provincias indicando avances en las obras de ayuda en la edificación y equipamientos de viviendas en cada provincia. La información de los detalles es la siguiente: Código pcia, código localidad, #viviendas con luz, #viviendas construidas, #viviendas con agua, #viviendas con gas, #entrega sanitarios.

    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 provincia y código de localidad.**

    Para la actualización del archivo maestro, se debe proceder de la siguiente manera:

    - Al valor de viviendas sin luz se le resta el valor recibido en el detalle.
    - Idem para viviendas sin agua, sin gas y sin sanitarios.
    - A las viviendas de chapa se le resta el valor recibido de viviendas construidas

    La misma combinación de provincia y localidad aparecen a lo sumo una única vez.

    Realice las declaraciones necesarias, el programa principal y los procedimientos que requiera para la actualización solicitada e informe cantidad de localidades sin viviendas de chapa (las localidades pueden o no haber sido actualizadas).

16. La editorial X, autora de diversos semanarios, posee un archivo maestro con la información correspondiente a las diferentes emisiones de los mismos. De cada emisión se registra: fecha, código de semanario, nombre del semanario, descripción, precio, total de ejemplares y total de ejemplares vendidos.

    Mensualmente se reciben 100 archivos detalles con las ventas de los semanarios en todo el país. La información que poseen los detalles es la siguiente: fecha, código de semanario y cantidad de ejemplares vendidos. Realice las declaraciones necesarias, la llamada al procedimiento y el procedimiento que recibe el archivo maestro y los 100 detalles y realice la actualización del archivo maestro en función de las ventas registradas. Además deberá informar fecha y semanario que tuvo más ventas y la misma información del semanario con menos ventas.

    Nota: Todos los archivos están ordenados por fecha y código de semanario. No se realizan ventas de semanarios si no hay ejemplares para hacerlo

17. Una concesionaria de motos de la Ciudad de Chascomús, posee un archivo con información de las motos que posee a la venta. De cada moto se registra: código, nombre, descripción, modelo, marca y stock actual. Mensualmente se reciben 10 archivos detalles con información de las ventas de cada uno de los 10 empleados que trabajan. De cada archivo detalle se dispone de la siguiente información: código de moto, precio y fecha de la venta. Se debe realizar un proceso que actualice el stock del archivo maestro desde los archivos detalles. Además se debe informar cuál fue la moto más vendida.

    NOTA: Todos los archivos están ordenados por código de la moto y el archivo maestro debe ser recorrido sólo una vez y en forma simultánea con los detalles.

18. Se cuenta con un archivo con información de los casos de COVID-19 registrados en los diferentes hospitales de la Provincia de Buenos Aires cada día. Dicho archivo contiene: código de localidad, nombre de localidad, código de municipio, nombre de municipio, código de hospital, nombre de hospital, fecha y cantidad de casos positivos detectados. El archivo está ordenado por localidad, luego por municipio y luego por hospital.

    Escriba la definición de las estructuras de datos necesarias y un procedimiento que haga un listado con el siguiente formato:

    Nombre: Localidad 1
        Nombre: Municipio 1
            Nombre Hospital 1……..Cantidad de casos Hospital 1
            ……..
            Nombre Hospital N…….Cantidad de casos Hospital N
        Cantidad de casos Municipio 1
        …….
        Nombre Municipio N
            Nombre Hospital 1……..Cantidad de casos Hospital 1

……..
Nombre Hospital N…….Cantidad de casos Hospital N
Cantidad de casos Municipio N
Cantidad de casos Localidad 1

Nombre Localidad N
Nombre Municipio 1
Nombre Hospital 1……..Cantidad de casos Hospital 1
……..
Nombre Hospital N**…….**Cantidad de casos Hospital N
Cantidad de casos Municipio 1
…….
Nombre Municipio N
Nombre Hospital 1……..Cantidad de casos Hospital 1
……..
Nombre Hospital N…….Cantidad de casos Hospital N
Cantidad de casos Municipio N
Cantidad de casos Localidad N

Cantidad de casos Totales en la Provincia

Además del informe en pantalla anterior, es necesario exportar a un archivo de texto la siguiente información: nombre de localidad, nombre de municipio y cantidad de casos del municipio, para aquellos municipios cuya cantidad de casos supere los 1500. El formato del archivo de texto deberá ser el adecuado para recuperar la información con la menor cantidad de lecturas posibles.

NOTA: El archivo debe recorrerse solo una vez.

19. A partir de un siniestro ocurrido se perdieron las actas de nacimiento y fallecimientos de toda la provincia de buenos aires de los últimos diez años. En pos de recuperar dicha información, se deberá procesar 2 archivos por cada una de las 50 delegaciones distribuidas en la provincia, un archivo de nacimientos y otro de fallecimientos y crear el archivo maestro reuniendo dicha información.

    Los archivos detalles con nacimientos, contendrán la siguiente información: nro partida nacimiento, nombre, apellido, dirección detallada (calle, nro, piso, depto, ciudad), matrícula del médico, nombre y apellido de la madre, DNI madre, nombre y apellido del padre, DNI del padre.

    En cambio, los 50 archivos de fallecimientos tendrán: nro partida nacimiento, DNI, nombre y apellido del fallecido, matrícula del médico que firma el deceso, fecha y hora del deceso y lugar.

    Realizar un programa que cree el archivo maestro a partir de toda la información de los archivos detalles. Se debe almacenar en el maestro: nro partida nacimiento, nombre, apellido, dirección detallada (calle, nro, piso, depto, ciudad), matrícula del médico, nombre y apellido de la madre, DNI madre, nombre y apellido del padre, DNI del padre y si falleció,

además matrícula del médico que firma el deceso, fecha y hora del deceso y lugar. Se deberá, además, listar en un archivo de texto la información recolectada de cada persona.

Nota: Todos los archivos están ordenados por nro partida de nacimiento que es única. Tenga en cuenta que no necesariamente va a fallecer en el distrito donde nació la persona y además puede no haber fallecido.

IMPORTANTE: Se recomienda implementar los ejercicios prácticos en Dev-Pascal. El ejecutable puede descargarse desde la plataforma moodle.

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.

Introducción a las Bases de Datos

Fundamentos de Organización de Datos

Práctica 3

Parte 1: Bajas en archivos

1. Modificar el ejercicio 4 de la práctica 1 (programa de gestión de empleados), agregándole una opción para realizar bajas copiando el último registro del archivo en la posición del registro a borrar y luego truncando el archivo en la posición del último registro de forma tal de evitar duplicados.

2. Definir un programa que genere un archivo con registros de longitud fija conteniendo información de asistentes a un congreso a partir de la información obtenida por teclado. Se deberá almacenar la siguiente información: nro de asistente, apellido y nombre, email, teléfono y D.N.I. Implementar un procedimiento que, a partir del archivo de datos generado, elimine de forma lógica todos los asistentes con nro de asistente inferior a 1000.

   Para ello se podrá utilizar algún carácter especial situándolo delante de algún campo String a su elección. Ejemplo: '@Saldaño'.

3. Realizar un programa que genere un archivo de novelas filmadas durante el presente año. De cada novela se registra: código, género, nombre, duración, director y precio. El programa debe presentar un menú con las siguientes opciones:

a. Crear el archivo y cargarlo a partir de datos ingresados por teclado. Se utiliza la técnica de lista invertida para recuperar espacio libre en el archivo. Para ello, durante la creación del archivo, en el primer registro del mismo se debe almacenar la cabecera de la lista. Es decir un registro ficticio, inicializando con el valor cero (0) el campo correspondiente al código de novela, el cual indica que no hay espacio libre dentro del archivo.

b. Abrir el archivo existente y permitir su mantenimiento teniendo en cuenta el inciso a), se utiliza lista invertida para recuperación de espacio. En particular, para el campo de "enlace" de la lista (utilice el código de novela como enlace), se debe especificar los números de registro referenciados con signo negativo, . Una vez abierto el archivo, brindar operaciones para:

    i. Dar de alta una novela leyendo la información desde teclado. Para esta operación, en caso de ser posible, deberá recuperarse el espacio libre. Es decir, si en el campo correspondiente al código de novela del registro cabecera hay un valor negativo, por ejemplo -5, se debe leer el registro en la posición 5, copiarlo en la posición 0 (actualizar la lista de espacio libre) y grabar el nuevo registro en la posición 5. Con el valor 0 (cero) en el registro cabecera se indica que no hay espacio libre.

    ii. Modificar los datos de una novela leyendo la información desde teclado. El código de novela no puede ser modificado.

    iii. Eliminar una novela cuyo código es ingresado por teclado. Por ejemplo, si se da de baja un registro en la posición 8, en el campo código de novela del registro cabecera deberá figurar -8, y en el registro en la posición 8 debe copiarse el antiguo registro cabecera.

c. Listar en un archivo de texto todas las novelas, incluyendo las borradas, que representan la lista de espacio libre. El archivo debe llamarse "*novelas.txt*".

NOTA: Tanto en la creación como en la apertura el nombre del archivo debe ser proporcionado por el usuario.

4. Dada la siguiente estructura:

```
type

     reg_flor = record

        nombre: String[45];
        codigo: integer;
     end;

     tArchFlores = file of reg_flor;
```

Las bajas se realizan apilando registros borrados y las altas reutilizando registros borrados. El registro 0 se usa como cabecera de la pila de registros borrados: el número 0 en el campo código implica que no hay registros borrados y -N indica que el próximo registro a reutilizar es el N, siendo éste un número relativo de registro válido.

a. Implemente el siguiente módulo:

{Abre el archivo y agrega una flor, recibida como parámetro manteniendo la política descrita anteriormente}

```
procedure agregarFlor (var a: tArchFlores ; nombre: string; codigo:integer);
```

b. Liste el contenido del archivo omitiendo las flores eliminadas. Modifique lo que considere necesario para obtener el listado.

5. Dada la estructura planteada en el ejercicio anterior, implemente el siguiente módulo:

{Abre el archivo y elimina la flor recibida como parámetro manteniendo la política descripta anteriormente}

```
procedure eliminarFlor (var a: tArchFlores; flor:reg_flor);
```

6. Una cadena de tiendas de indumentaria posee un archivo maestro no ordenado con la información correspondiente a las prendas que se encuentran a la venta. De cada prenda se registra: cod_prenda, descripción, colores, tipo_prenda, stock y precio_unitario. Ante un eventual cambio de temporada, se deben actualizar las prendas a la venta. Para ello reciben un archivo conteniendo: cod_prenda de las

prendas que quedarán obsoletas. Deberá implementar un procedimiento que reciba ambos archivos y realice la baja lógica de las prendas, para ello deberá modificar el stock de la prenda correspondiente a valor negativo.

Adicionalmente, deberá implementar otro procedimiento que se encargue de efectivizar las bajas lógicas que se realizaron sobre el archivo maestro con la información de las prendas a la venta. Para ello se deberá utilizar una estructura auxiliar (esto es, un archivo nuevo), en el cual se copien únicamente aquellas prendas que no están marcadas como borradas. Al finalizar este proceso de compactación del archivo, se deberá renombrar el archivo nuevo con el nombre del archivo maestro original.

7. Se cuenta con un archivo que almacena información sobre especies de aves en vía de extinción, para ello se almacena: código, nombre de la especie, familia de ave, descripción y zona geográfica. El archivo no está ordenado por ningún criterio. Realice un programa que permita borrar especies de aves extintas. Este programa debe disponer de dos procedimientos:
    a. Un procedimiento que dada una especie de ave (su código) marque la misma como borrada (en caso de querer borrar múltiples especies de aves, se podría invocar este procedimiento repetidamente).
    b. Un procedimiento que compacte el archivo, quitando definitivamente las especies de aves marcadas como borradas. Para quitar los registros se deberá copiar el último registro del archivo en la posición del registro a borrar y luego eliminar del archivo el último registro de forma tal de evitar registros duplicados.
        i. Implemente una variante de este procedimiento de compactación del archivo (baja física) donde el archivo se trunque una sola vez.

8. Se cuenta con un archivo con información de las diferentes distribuciones de linux existentes. De cada distribución se conoce: nombre, año de lanzamiento, número de

versión del kernel, cantidad de desarrolladores y descripción. El nombre de las distribuciones no puede repetirse. Este archivo debe ser mantenido realizando bajas lógicas y utilizando la técnica de reutilización de espacio libre llamada **lista invertida**. Escriba la definición de las estructuras de datos necesarias y los siguientes procedimientos:

a. **BuscarDistribucion**: módulo que recibe por parámetro el archivo, un nombre de distribución y devuelve la posición dentro del archivo donde se encuentra el registro correspondiente a la distribución dada (si existe) o devuelve -1 en caso de que no exista..
b. **AltaDistribucion**: módulo que recibe como parámetro el archivo y el registro que contiene los datos de una nueva distribución, y se encarga de agregar la distribución al archivo reutilizando espacio disponible en caso de que exista. (El control de unicidad lo debe realizar utilizando el módulo anterior). En caso de que la distribución que se quiere agregar ya exista se debe informar "ya existe la distribución".
c. **BajaDistribucion**: módulo que recibe como parámetro el archivo y el nombre de una distribución, y se encarga de dar de baja lógicamente la distribución dada. Para marcar una distribución como borrada se debe utilizar el campo cantidad de desarrolladores para mantener actualizada la lista invertida. Para verificar que la distribución a borrar exista debe utilizar el módulo BuscarDistribucion. En caso de no existir se debe informar "Distribución no existente".

Parte 2: Actualización maestro/detalle, reportes y merge con archivos no ordenados

Para los ejercicios de esta parte de la práctica, teniendo en cuenta que los archivos no están ordenados por ningún criterio, puede resultar necesario recorrer los archivos más de una vez. La idea es resolver los ejercicios sin ordenar los archivos dados, y comparar la eficiencia (en cuanto al número de lecturas/escrituras) de la solución brindada en esta práctica respecto a la solución para el mismo problema considerando los archivos ordenados.

1. 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. Resuelve los siguientes puntos:

    a. Se pide realizar un procedimiento que actualice el archivo maestro con el archivo detalle, teniendo en cuenta que:

        i. Los archivos no están ordenados por ningún criterio.
        ii. Cada registro del maestro puede ser actualizado por 0, 1 ó más registros del archivo detalle.

    b. ¿Qué cambios realizaría en el procedimiento del punto anterior si se sabe que cada registro del archivo maestro puede ser actualizado por 0 o 1 registro del archivo detalle?

2. Se necesita contabilizar los votos de las diferentes mesas electorales registradas por localidad en la provincia de Buenos Aires. Para ello, se posee un archivo con la siguiente información: 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 Localidad	Total de Votos
.....	.....
.....	.....
Total General de Votos:	.....

    NOTAS:

    - La información en el archivo no está ordenada por ningún criterio.
    - Trate de resolver el problema sin modificar el contenido del archivo dado.
    - Puede utilizar una estructura auxiliar, como por ejemplo otro archivo, para llevar el control de las localidades que han sido procesadas.

3. Suponga que 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. 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:

   - Los archivos detalle no están ordenados por ningún criterio.
   - Un usuario puede iniciar más de una sesión el mismo día en la misma máquina, o inclusive, en diferentes máquinas.

Juan Menduiña

Trabajo Práctico N° 3

PARTE I: Bajas en Archivos.

Ejercicio 1.

Ejercicio 2.

Ejercicio 3.

Ejercicio 4.

Ejercicio 5.

Ejercicio 6.

Ejercicio 7.

Ejercicio 8.

PARTE II: Actualización Maestro/Detalle, Reportes y *Merge* con Archivos no Ordenados.

Ejercicio 9.

Ejercicio 10.

Ejercicio 11.

Juan Menduiña

Trabajo Práctico N° 4:

.

Ejercicio 1.

Juan Menduiña

Trabajo Práctico N° 5:

.

Ejercicio 1.