Juan Menduiña

Trabajo Práctico N° 3:
Bajas en Archivos y Actualización Maestro/Detalle, Reportes y *Merge* con Archivos no Ordenados.

PARTE I: Bajas en Archivos.

Ejercicio 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.

```
program TP3_E1;
{$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,'E1_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,'E1_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 eliminar_empleado(var archivo_empleados: t_archivo_empleados);
var
  registro_empleado, registro_ultimo_empleado: t_registro_empleado;
  numero: int16;
begin
  numero:=1+random(1000);
  reset(archivo_empleados);
  seek(archivo_empleados,filesize(archivo_empleados)-1);
  read(archivo_empleados,registro_ultimo_empleado);
  seek(archivo_empleados,0);
  read(archivo_empleados,registro_empleado);
  while (not eof(archivo_empleados)) and (registro_empleado.numero<>numero) do
    read(archivo_empleados,registro_empleado);
  if (registro_empleado.numero=numero) then
  begin
    seek(archivo_empleados,filepos(archivo_empleados)-1);
    write(archivo_empleados,registro_ultimo_empleado);
    seek(archivo_empleados,filesize(archivo_empleados)-1);
    truncate(archivo_empleados);
```

```
    textcolor(green); write('Se ha eliminado el 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;
  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 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 9: '); textcolor(green); writeln('Eliminar un empleado del
archivo, copiando el último registro en la posición del registro a borrar y truncando el
archivo');
  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);
      9: eliminar_empleado(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,'E1_empleados');
  menu_opciones(archivo_empleados);
end.
```

Ejercicio 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 DNI. 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 elección. Ejemplo: "@Saldaño".

```
program TP3_E2;
{$codepage UTF8}
uses crt;
const
  numero_salida=0;
  numero_corte=1000;
type
  t_string20=string[20];
  t_registro_asistente=record
    numero: int16;
    apellido: t_string20;
    nombre: t_string20;
    email: t_string20;
    telefono: int32;
    dni: int32;
  end;
  t_archivo_asistentes=file of t_registro_asistente;
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_asistente(var registro_asistente: t_registro_asistente);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_asistente.numero:=numero_salida
  else
    registro_asistente.numero:=1+random(2000);
  if (registro_asistente.numero<>numero_salida) then
  begin
    registro_asistente.apellido:=random_string(5+random(5));
    registro_asistente.nombre:=random_string(5+random(5));
    registro_asistente.email:=random_string(5+random(5))+'@gmail.com';
    registro_asistente.telefono:=1000000000+random(1000000001);
    registro_asistente.dni:=10000000+random(40000001);
  end;
end;
procedure cargar_archivo_asistentes(var archivo_asistentes: t_archivo_asistentes);
var
  registro_asistente: t_registro_asistente;
begin
  rewrite(archivo_asistentes);
  leer_asistente(registro_asistente);
  while (registro_asistente.numero<>numero_salida) do
```

```
  begin
    write(archivo_asistentes,registro_asistente);
    leer_asistente(registro_asistente);
  end;
  close(archivo_asistentes);
end;
procedure imprimir_registro_asistente(registro_asistente: t_registro_asistente);
begin
  textcolor(green); write('Número: '); textcolor(yellow); write(registro_asistente.numero);
  textcolor(green); write('; Apellido: '); textcolor(yellow);
write(registro_asistente.apellido);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_asistente.nombre);
  textcolor(green); write('; Email: '); textcolor(yellow); write(registro_asistente.email);
  textcolor(green); write('; Teléfono: '); textcolor(yellow);
write(registro_asistente.telefono);
  textcolor(green); write('; DNI: '); textcolor(yellow); writeln(registro_asistente.dni);
end;
procedure imprimir_archivo_asistentes(var archivo_asistentes: t_archivo_asistentes);
var
  registro_asistente: t_registro_asistente;
begin
  reset(archivo_asistentes);
  while (not eof(archivo_asistentes)) do
  begin
    read(archivo_asistentes,registro_asistente);
    imprimir_registro_asistente(registro_asistente);
  end;
  close(archivo_asistentes);
end;
procedure eliminar_archivo_asistentes(var archivo_asistentes: t_archivo_asistentes);
var
  registro_asistente: t_registro_asistente;
begin
  reset(archivo_asistentes);
  while (not eof(archivo_asistentes)) do
  begin
    read(archivo_asistentes,registro_asistente);
    if (registro_asistente.numero<numero_corte) then
    begin
      registro_asistente.apellido:='@'+registro_asistente.apellido;
      seek(archivo_asistentes,filepos(archivo_asistentes)-1);
      write(archivo_asistentes,registro_asistente);
    end;
  end;
  close(archivo_asistentes);
end;
var
  archivo_asistentes: t_archivo_asistentes;
begin
  randomize;
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO ASISTENTES:'); writeln();
  assign(archivo_asistentes,'E2_asistentes');
  cargar_archivo_asistentes(archivo_asistentes);
  imprimir_archivo_asistentes(archivo_asistentes);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO ASISTENTES ACTUALIZADO:'); writeln();
  eliminar_archivo_asistentes(archivo_asistentes);
  imprimir_archivo_asistentes(archivo_asistentes);
end.
```

Ejercicio 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 (utilizar 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.

Ejercicio 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) *Implementar 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) *Listar el contenido del archivo omitiendo las flores eliminadas. Modificar lo que se considere necesario para obtener el listado.*

```
program TP3_E4;
{$codepage UTF8}
uses crt;
const
  codigo_salida=-1;
  codigo_cabecera=0;
type
  t_string45=string[45];
  t_registro_flor=record
    nombre: t_string45;
    codigo: int16;
  end;
  t_archivo_flores=file of t_registro_flor;
function random_string(length: int8): t_string45;
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_flor(var registro_flor: t_registro_flor);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_flor.codigo:=codigo_salida
  else
    registro_flor.codigo:=1+random(1000);
```

```
  if (registro_flor.codigo<>codigo_salida) then
    registro_flor.nombre:=random_string(5+random(6));
end;
procedure cargar_archivo_flores(var archivo_flores: t_archivo_flores);
var
  registro_flor: t_registro_flor;
begin
  rewrite(archivo_flores);
  registro_flor.codigo:=codigo_cabecera;
  registro_flor.nombre:='Cabecera';
  while (registro_flor.codigo<>codigo_salida) do
  begin
    write(archivo_flores,registro_flor);
    leer_flor(registro_flor);
  end;
  close(archivo_flores);
end;
procedure imprimir_registro_flor(registro_flor: t_registro_flor);
begin
  textcolor(green); write('Nombre: '); textcolor(yellow); write(registro_flor.nombre);
  textcolor(green); write('; Código: '); textcolor(yellow); writeln(registro_flor.codigo);
end;
procedure imprimir_archivo_flores(var archivo_flores: t_archivo_flores);
var
  registro_flor: t_registro_flor;
begin
  reset(archivo_flores);
  while (not eof(archivo_flores)) do
  begin
    read(archivo_flores,registro_flor);
    if (registro_flor.codigo>codigo_cabecera) then
      imprimir_registro_flor(registro_flor);
  end;
  close(archivo_flores);
end;
procedure agregar_flor(var archivo_flores: t_archivo_flores; nombre: t_string45; codigo:
int16);
var
  registro_flor, cabecera: t_registro_flor;
begin
  reset(archivo_flores);
  read(archivo_flores,cabecera);
  registro_flor.nombre:=nombre;
  registro_flor.codigo:=codigo;
  if (cabecera.codigo=codigo_cabecera) then
  begin
    seek(archivo_flores,filesize(archivo_flores));
    write(archivo_flores,registro_flor);
  end
  else
  begin
    seek(archivo_flores,cabecera.codigo*(-1));
    read(archivo_flores,cabecera);
    seek(archivo_flores,filepos(archivo_flores)-1);
    write(archivo_flores,registro_flor);
    seek(archivo_flores,0);
    write(archivo_flores,cabecera);
  end;
  close(archivo_flores);
  textcolor(green); write('Se ha realizado el alta de la flor '); textcolor(yellow);
write(nombre); textcolor(green); write(' con código '); textcolor(yellow); writeln(codigo);
  writeln();
end;
var
  archivo_flores: t_archivo_flores;
  codigo: int16;
```

```
  nombre: t_string45;
begin
  randomize;
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO FLORES:'); writeln();
  assign(archivo_flores,'flores');
  cargar_archivo_flores(archivo_flores);
  imprimir_archivo_flores(archivo_flores);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO FLORES AL AGREGAR FLOR:'); writeln();
  codigo:=1+random(1000);
  nombre:=random_string(5+random(6));
  agregar_flor(archivo_flores,nombre,codigo);
  imprimir_archivo_flores(archivo_flores);
end.
```

Ejercicio 5.

Dada la estructura planteada en el ejercicio anterior, implementar 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);

```
program TP3_E5;
{$codepage UTF8}
uses crt;
const
  codigo_salida=-1;
  codigo_cabecera=0;
type
  t_string45=string[45];
  t_registro_flor=record
    nombre: t_string45;
    codigo: int16;
  end;
  t_archivo_flores=file of t_registro_flor;
function random_string(length: int8): t_string45;
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_flor(var registro_flor: t_registro_flor);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_flor.codigo:=codigo_salida
  else
    registro_flor.codigo:=1+random(100);
  if (registro_flor.codigo<>codigo_salida) then
    registro_flor.nombre:=random_string(5+random(6));
end;
procedure cargar_archivo_flores(var archivo_flores: t_archivo_flores);
var
  registro_flor: t_registro_flor;
begin
  rewrite(archivo_flores);
  registro_flor.codigo:=codigo_cabecera;
  registro_flor.nombre:='Cabecera';
  while (registro_flor.codigo<>codigo_salida) do
  begin
    write(archivo_flores,registro_flor);
    leer_flor(registro_flor);
  end;
  close(archivo_flores);
end;
procedure imprimir_registro_flor(registro_flor: t_registro_flor);
begin
  textcolor(green); write('Nombre: '); textcolor(yellow); write(registro_flor.nombre);
  textcolor(green); write('; Código: '); textcolor(yellow); writeln(registro_flor.codigo);
end;
```

```
procedure imprimir_archivo_flores(var archivo_flores: t_archivo_flores);
var
  registro_flor: t_registro_flor;
begin
  reset(archivo_flores);
  while (not eof(archivo_flores)) do
  begin
    read(archivo_flores,registro_flor);
    if (registro_flor.codigo>codigo_cabecera) then
      imprimir_registro_flor(registro_flor);
  end;
  close(archivo_flores);
end;
procedure agregar_flor(var archivo_flores: t_archivo_flores; nombre: t_string45; codigo:
int16);
var
  registro_flor, cabecera: t_registro_flor;
begin
  reset(archivo_flores);
  read(archivo_flores,cabecera);
  registro_flor.nombre:=nombre;
  registro_flor.codigo:=codigo;
  if (cabecera.codigo=codigo_cabecera) then
  begin
    seek(archivo_flores,filesize(archivo_flores));
    write(archivo_flores,registro_flor);
  end
  else
  begin
    seek(archivo_flores,cabecera.codigo*(-1));
    read(archivo_flores,cabecera);
    seek(archivo_flores,filepos(archivo_flores)-1);
    write(archivo_flores,registro_flor);
    seek(archivo_flores,0);
    write(archivo_flores,cabecera);
  end;
  close(archivo_flores);
  textcolor(green); write('Se ha realizado el alta de la flor '); textcolor(yellow);
write(nombre); textcolor(green); write(' con código '); textcolor(yellow); writeln(codigo);
  writeln();
end;
procedure eliminar_flor(var archivo_flores: t_archivo_flores; flor: t_registro_flor);
var
  registro_flor, cabecera: t_registro_flor;
  ok: boolean;
begin
  ok:=false;
  reset(archivo_flores);
  read(archivo_flores,cabecera);
  while ((not eof(archivo_flores)) and (ok=false)) do
  begin
    read(archivo_flores,registro_flor);
    if (registro_flor.codigo=flor.codigo) then
    begin
      ok:=true;
      seek(archivo_flores,filepos(archivo_flores)-1);
      write(archivo_flores,cabecera);
      cabecera.codigo:=(filepos(archivo_flores)-1)*(-1);
      seek(archivo_flores,0);
      write(archivo_flores,cabecera);
    end;
  end;
  close(archivo_flores);
  if (ok=true) then
  begin
```

```
    textcolor(green); write('Se ha realizado la baja de la flor '); textcolor(yellow);
write(flor.nombre); textcolor(green); write(' con código '); textcolor(yellow);
writeln(flor.codigo);
  end
  else
  begin
    textcolor(green); write('No se ha encontrado la flor '); textcolor(yellow);
write(flor.nombre); textcolor(green); write(' con código '); textcolor(yellow);
writeln(flor.codigo);
  end;
  writeln();
end;
var
  archivo_flores: t_archivo_flores;
  registro_flor: t_registro_flor;
begin
  randomize;
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO FLORES:'); writeln();
  assign(archivo_flores,'flores');
  cargar_archivo_flores(archivo_flores);
  imprimir_archivo_flores(archivo_flores);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO FLORES AL AGREGAR FLOR ANTES DE LA
BAJA:'); writeln();
  registro_flor.codigo:=1+random(100);
  registro_flor.nombre:=random_string(5+random(6));
  agregar_flor(archivo_flores,registro_flor.nombre,registro_flor.codigo);
  imprimir_archivo_flores(archivo_flores);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO FLORES AL ELIMINAR FLOR:'); writeln();
  registro_flor.codigo:=1+random(100);
  eliminar_flor(archivo_flores,registro_flor);
  imprimir_archivo_flores(archivo_flores);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO FLORES AL AGREGAR FLOR DESPUÉS DE LA
BAJA:'); writeln();
  registro_flor.codigo:=1+random(100);
  registro_flor.nombre:=random_string(5+random(6));
  agregar_flor(archivo_flores,registro_flor.nombre,registro_flor.codigo);
  imprimir_archivo_flores(archivo_flores);
end.
```

Ejercicio 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. Se deberá implementar un procedimiento que reciba ambos archivos y realice la baja lógica de las prendas. Para ello, se deberá modificar el stock de la prenda correspondiente a valor negativo.

Adicionalmente, se 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.

Ejercicio 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. Realizar 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) *Implementar una variante de este procedimiento de compactación del archivo (baja física) donde el archivo se trunque una sola vez.*

Ejercicio 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. Escribir 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 se 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, se debe utilizar el módulo BuscarDistribucion. En caso de no existir, se debe informar "Distribución no existente".*

PARTE II: 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.

Ejercicio 9.

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. Resolver los siguientes puntos:

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

- *Los archivos no están ordenados por ningún criterio.*
- *Cada registro del maestro puede ser actualizado por 0, 1 o más registros del archivo detalle.*

(b) *¿Qué cambios se realizarían en el procedimiento del inciso anterior si se sabe que cada registro del archivo maestro puede ser actualizado por 0 o 1 registro del archivo detalle?*

Ejercicio 10.

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.*
- *Tratar de resolver el problema sin modificar el contenido del archivo dado.*
- *Se puede utilizar una estructura auxiliar, como por ejemplo otro archivo, para llevar el control de las localidades que han sido procesadas.*

Ejercicio 11.

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:

- *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.*