Juan Menduiña

Trabajo Práctico N° 7:
Repaso.

Ejercicio 1.

Una productora nacional realiza un casting de personas para la selección de actores extras de una nueva película, para ello se debe leer y almacenar la información de las personas que desean participar de dicho casting. De cada persona, se lee: DNI, apellido y nombre, edad y el código de género de actuación que prefiere (1: drama, 2: romántico, 3: acción, 4: suspenso, 5: terror). La lectura finaliza cuando llega una persona con DNI 33.555.444, la cual debe procesarse. Una vez finalizada la lectura de todas las personas, se pide:

(a) *Informar la cantidad de personas cuyo DNI contiene más dígitos pares que impares.*

(b) *Informar los dos códigos de género más elegidos.*

(c) *Realizar un módulo que reciba un DNI, lo busque y lo elimine de la estructura. El DNI puede no existir. Invocar dicho módulo en el programa principal.*

```
program TP7_E1;
{$codepage UTF8}
uses crt;
const
  genero_ini=1; genero_fin=5;
  dni_salida=33555444;
  digito_ini=0; digito_fin=9;
type
  t_genero=genero_ini..genero_fin;
  t_digito=digito_ini..digito_fin;
  t_registro_persona=record
    dni: int32;
    nombre: string;
    apellido: string;
    edad: int8;
    genero: t_genero;
  end;
  t_lista_personas=^t_nodo_personas;
  t_nodo_personas=record
    ele: t_registro_persona;
    sig: t_lista_personas;
  end;
  t_vector_cantidades=array[t_genero] of int16;
function random_string(length: int8): string;
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_persona(var registro_persona: t_registro_persona);
var
  i: int8;
begin
```

```
  i:=random(100);
  if (i=0) then
    registro_persona.dni:=dni_salida
  else
    registro_persona.dni:=10000000+random(40000001);
  registro_persona.nombre:=random_string(5+random(6));
  registro_persona.apellido:=random_string(5+random(6));
  registro_persona.edad:=1+random(high(int8));
  registro_persona.genero:=genero_ini+random(genero_fin);
end;
procedure agregar_adelante_lista_personas(var lista_personas: t_lista_personas;
registro_persona: t_registro_persona);
var
  nuevo: t_lista_personas;
begin
  new(nuevo);
  nuevo^.ele:=registro_persona;
  nuevo^.sig:=lista_personas;
  lista_personas:=nuevo;
end;
procedure cargar_lista_personas(var lista_personas: t_lista_personas);
var
  registro_persona: t_registro_persona;
begin
  repeat
    leer_persona(registro_persona);
    agregar_adelante_lista_personas(lista_personas,registro_persona);
  until (registro_persona.dni=dni_salida)
end;
procedure imprimir_registro_persona(registro_persona: t_registro_persona; persona: int16);
begin
  textcolor(green); write('El DNI de la persona '); textcolor(yellow); write(persona);
textcolor(green); write(' es '); textcolor(red); writeln(registro_persona.dni);
  textcolor(green); write('El nombre de la persona '); textcolor(yellow); write(persona);
textcolor(green); write(' es '); textcolor(red); writeln(registro_persona.nombre);
  textcolor(green); write('El apellido de la persona '); textcolor(yellow); write(persona);
textcolor(green); write(' es '); textcolor(red); writeln(registro_persona.apellido);
  textcolor(green); write('La edad de la persona '); textcolor(yellow); write(persona);
textcolor(green); write(' es '); textcolor(red); writeln(registro_persona.edad);
  textcolor(green); write('El código de género de actuación que prefiere la persona ');
textcolor(yellow); write(persona); textcolor(green); write(' es '); textcolor(red);
writeln(registro_persona.genero);
end;
procedure imprimir_lista_personas(lista_personas: t_lista_personas);
var
  i: int16;
begin
  i:=0;
  while (lista_personas<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información de la persona '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_persona(lista_personas^.ele,i);
    writeln();
    lista_personas:=lista_personas^.sig;
  end;
end;
procedure inicializar_vector_cantidades(var vector_cantidades: t_vector_cantidades);
var
  i: t_genero;
begin
  for i:= genero_ini to genero_fin do
    vector_cantidades[i]:=0;
end;
function contar_pares_impares(dni: int32): boolean;
```

```
var
  pares, impares: int8;
begin
  pares:=0; impares:=0;
  while (dni<>0) do
  begin
    if (dni mod 2=0) then
      pares:=pares+1
    else
      impares:=impares+1;
    dni:=dni div 10;
  end;
  contar_pares_impares:=(pares>impares);
end;
procedure actualizar_maximos(cantidad: int16; genero: t_genero; var cantidad_max1,
cantidad_max2: int16; var genero_max1, genero_max2: int8);
begin
  if (cantidad>cantidad_max1) then
  begin
    cantidad_max2:=cantidad_max1;
    genero_max2:=genero_max1;
    cantidad_max1:=cantidad;
    genero_max1:=genero;
  end
  else
    if (cantidad>cantidad_max2) then
    begin
      cantidad_max2:=cantidad;
      genero_max2:=genero;
    end;
end;
procedure procesar_vector_cantidades(vector_cantidades: t_vector_cantidades; var genero_max1,
genero_max2: int8);
var
  i: t_genero;
  cantidad_max1, cantidad_max2: int16;
begin
  cantidad_max1:=low(int16); cantidad_max2:=low(int16);
  for i:= genero_ini to genero_fin do
  begin
    actualizar_maximos(vector_cantidades[i],i,cantidad_max1,cantidad_max2,genero_max1,genero_m
ax2);
    textcolor(green); write('La cantidad de personas que prefieren el código de género de
actuación ',i, ' es '); textcolor(red); writeln(vector_cantidades[i]);
  end;
end;
procedure procesar_lista_personas(lista_personas: t_lista_personas; var personas: int16; var
genero_max1, genero_max2: int8);
var
  vector_cantidades: t_vector_cantidades;
begin
  inicializar_vector_cantidades(vector_cantidades);
  while (lista_personas<>nil) do
  begin
    if (contar_pares_impares(lista_personas^.ele.dni)=true) then
      personas:=personas+1;
    vector_cantidades[lista_personas^.ele.genero]:=vector_cantidades[lista_personas^.ele.gener
o]+1;
    lista_personas:=lista_personas^.sig;
  end;
  procesar_vector_cantidades(vector_cantidades,genero_max1,genero_max2);
end;
procedure eliminar_lista_personas(var lista_personas: t_lista_personas; var ok: boolean; dni:
int32);
var
  anterior, actual: t_lista_personas;
```

```
begin
  actual:=lista_personas;
  while ((actual<>nil) and (actual^.ele.dni<>dni)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual<>nil) then
  begin
    if (actual=lista_personas) then
      lista_personas:=lista_personas^.sig
    else
      anterior^.sig:=actual^.sig;
    dispose(actual);
    ok:=true;
  end
end;
var
  lista_personas: t_lista_personas;
  genero_max1, genero_max2: int8;
  personas: int16;
  dni: int32;
  ok: boolean;
begin
  randomize;
  lista_personas:=nil;
  personas:=0;
  genero_max1:=0; genero_max2:=0;
  ok:=false;
  cargar_lista_personas(lista_personas);
  imprimir_lista_personas(lista_personas);
  procesar_lista_personas(lista_personas,personas,genero_max1,genero_max2);
  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
  textcolor(green); write('La cantidad de personas cuyo DNI contiene más dígitos pares que
impares es '); textcolor(red); writeln(personas);
  writeln(); textcolor(red); writeln('INCISO (b):'); writeln();
  textcolor(green); write('Los dos códigos de género más elegidos son '); textcolor(red);
write(genero_max1); textcolor(green); write(' y '); textcolor(red); writeln(genero_max2);
  writeln(); textcolor(red); writeln('INCISO (c):'); writeln();
  dni:=10000000+random(40000001);
  eliminar_lista_personas(lista_personas,ok,dni);
  if (ok=true) then
  begin
    textcolor(green); write('El DNI '); textcolor(yellow); write(dni); textcolor(red); write('
SÍ'); textcolor(green); write(' fue encontrado y eliminado');
    imprimir_lista_personas(lista_personas);
  end
  else
  begin
    textcolor(green); write('El DNI '); textcolor(yellow); write(dni); textcolor(red); write('
NO'); textcolor(green); write(' fue encontrado');
  end;
end.
```

Ejercicio 2.

Implementar un programa que lea y almacene información de clientes de una empresa aseguradora automotriz. De cada cliente, se lee: código de cliente, DNI, apellido, nombre, código de póliza contratada (1..6) y monto básico que abona mensualmente. La lectura finaliza cuando llega el cliente con código 1122, el cual debe procesarse. La empresa dispone de una tabla donde guarda un valor que representa un monto adicional que el cliente debe abonar en la liquidación mensual de su seguro, de acuerdo al código de póliza que tiene contratada. Una vez finalizada la lectura de todos los clientes, se pide:

(a) *Informar, para cada cliente, DNI, apellido, nombre y el monto completo que paga mensualmente por su seguro automotriz (monto básico + monto adicional).*

(b) *Informar apellido y nombre de aquellos clientes cuyo DNI contiene, al menos, dos dígitos 9.*

(c) *Realizar un módulo que reciba un código de cliente, lo busque (seguro existe) y lo elimine de la estructura.*

```
program TP7_E2;
{$codepage UTF8}
uses crt;
const
  poliza_ini=1; poliza_fin=6;
  codigo_salida=1122;
  digito_corte=9; cantidad_digito_corte=2;
type
  t_poliza=poliza_ini..poliza_fin;
  t_registro_cliente=record
    codigo: int16;
    dni: int32;
    apellido: string;
    nombre: string;
    poliza: t_poliza;
    monto: real;
  end;
  t_lista_clientes=^t_nodo_clientes;
  t_nodo_clientes=record
    ele: t_registro_cliente;
    sig: t_lista_clientes;
  end;
  t_vector_montos=array[t_poliza] of real;
procedure cargar_vector_montos(var vector_montos: t_vector_montos);
var
  i: t_poliza;
begin
  for i:= poliza_ini to poliza_fin do
    vector_montos[i]:=1+random(991)/10;
end;
function random_string(length: int8): string;
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_cliente(var registro_cliente: t_registro_cliente);
var
  i: int8;
begin
  i:=random(10);
  if (i=0) then
    registro_cliente.codigo:=codigo_salida
  else
    registro_cliente.codigo:=1+random(high(int16));
  registro_cliente.dni:=10000000+random(40000001);
  registro_cliente.apellido:=random_string(5+random(6));
  registro_cliente.nombre:=random_string(5+random(6));
  registro_cliente.poliza:=poliza_ini+random(poliza_fin);
  registro_cliente.monto:=1+random(991)/10;
end;
procedure agregar_adelante_lista_clientes(var lista_clientes: t_lista_clientes;
registro_cliente: t_registro_cliente);
var
  nuevo: t_lista_clientes;
begin
  new(nuevo);
  nuevo^.ele:=registro_cliente;
  nuevo^.sig:=lista_clientes;
  lista_clientes:=nuevo;
end;
procedure cargar_lista_clientes(var lista_clientes: t_lista_clientes);
var
  registro_cliente: t_registro_cliente;
begin
  repeat
    leer_cliente(registro_cliente);
    agregar_adelante_lista_clientes(lista_clientes,registro_cliente);
  until (registro_cliente.codigo=codigo_salida)
end;
procedure imprimir_registro_cliente(registro_cliente: t_registro_cliente; cliente: int16;
vector_montos: t_vector_montos);
begin
  textcolor(green); write('El DNI del cliente '); textcolor(yellow); write(cliente);
textcolor(green); write(' es '); textcolor(red); writeln(registro_cliente.dni);
  textcolor(green); write('El apellido del cliente '); textcolor(yellow); write(cliente);
textcolor(green); write(' es '); textcolor(red); writeln(registro_cliente.apellido);
  textcolor(green); write('El nombre del cliente '); textcolor(yellow); write(cliente);
textcolor(green); write(' es '); textcolor(red); writeln(registro_cliente.nombre);
  textcolor(green); write('El monto completo que paga mensualmente el cliente ');
textcolor(yellow); write(cliente); textcolor(green); write(' es '); textcolor(red);
writeln(registro_cliente.monto+vector_montos[registro_cliente.poliza]:0:2);
end;
function contar_digitos(dni: int32): boolean;
var
  digitos: int8;
begin
  digitos:=0;
  while (dni<>0) do
  begin
    if (dni mod 10=digito_corte) then
      digitos:=digitos+1;
    dni:=dni div 10;
  end;
  contar_digitos:=(digitos>=cantidad_digito_corte);
end;
procedure procesar_lista_clientes(lista_clientes: t_lista_clientes; vector_montos:
t_vector_montos);
var
  i: int16;
```

```
begin
  i:=0;
  while (lista_clientes<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información del cliente '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_cliente(lista_clientes^.ele,i,vector_montos);
    if (contar_digitos(lista_clientes^.ele.dni)=true) then
    begin
      textcolor(green); write('El apellido y nombre de este cliente cuyo DNI contiene, al
menos, '); textcolor(yellow); write(cantidad_digito_corte); textcolor(green); write(' dígitos
'); textcolor(yellow); write(digito_corte); textcolor(green); write(' son '); textcolor(red);
writeln(lista_clientes^.ele.apellido,' ',lista_clientes^.ele.nombre);
    end;
    writeln();
    lista_clientes:=lista_clientes^.sig;
  end;
end;
procedure eliminar_lista_clientes(var lista_clientes: t_lista_clientes; codigo: int16);
var
  anterior, actual: t_lista_clientes;
begin
  actual:=lista_clientes;
  while (actual^.ele.codigo<>codigo) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=lista_clientes) then
    lista_clientes:=lista_clientes^.sig
  else
    anterior^.sig:=actual^.sig;
  dispose(actual);
end;
var
  vector_montos: t_vector_montos;
  lista_clientes: t_lista_clientes;
  codigo: int16;
begin
  randomize;
  lista_clientes:=nil;
  cargar_vector_montos(vector_montos);
  cargar_lista_clientes(lista_clientes);
  writeln(); textcolor(red); writeln('INCISOS (a) (b):'); writeln();
  procesar_lista_clientes(lista_clientes,vector_montos);
  writeln(); textcolor(red); writeln('INCISO (c):'); writeln();
  codigo:=codigo_salida;
  eliminar_lista_clientes(lista_clientes,codigo);
  procesar_lista_clientes(lista_clientes,vector_montos);
end.
```

Ejercicio 3.

Una remisería dispone de información acerca de los viajes realizados durante el mes de mayo de 2020. De cada viaje, se conoce: número de viaje, código de auto, dirección de origen, dirección de destino y kilómetros recorridos durante el viaje. Esta información se encuentra ordenada por código de auto y, para un mismo código de auto, pueden existir 1 o más viajes. Se pide:

(a) *Informar los dos códigos de auto que más kilómetros recorrieron.*

(b) *Generar una lista nueva con los viajes de más de 5 kilómetros recorridos, ordenada por número de viaje.*

```
program TP7_E3;
{$codepage UTF8}
uses crt;
const
  km_corte=5;
  numero_salida=0;
type
  t_registro_viaje=record
    numero: int16;
    auto: int8;
    origen: string;
    destino: string;
    km: real;
  end;
  t_lista_viajes=^t_nodo_viajes;
  t_nodo_viajes=record
    ele: t_registro_viaje;
    sig: t_lista_viajes;
  end;
function random_string(length: int8): string;
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_viaje(var registro_viaje: t_registro_viaje);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_viaje.numero:=numero_salida
  else
    registro_viaje.numero:=1+random(high(int16));
  if (registro_viaje.numero<>numero_salida) then
  begin
    registro_viaje.auto:=1+random(high(int8));
    registro_viaje.origen:=random_string(5+random(6));
    registro_viaje.destino:=random_string(5+random(6));
    registro_viaje.km:=1+random(991)/10;
  end;
end;
procedure agregar_ordenado_lista_viajes1(var lista_viajes1: t_lista_viajes; registro_viaje:
t_registro_viaje);
```

```
var
  anterior, actual, nuevo: t_lista_viajes;
begin
  new(nuevo);
  nuevo^.ele:=registro_viaje;
  nuevo^.sig:=nil;
  actual:=lista_viajes1;
  while ((actual<>nil) and (actual^.ele.auto<nuevo^.ele.auto)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=lista_viajes1) then
    lista_viajes1:=nuevo
  else
    anterior^.sig:=nuevo;
  nuevo^.sig:=actual;
end;
procedure cargar_lista_viajes1(var lista_viajes1: t_lista_viajes);
var
  registro_viaje: t_registro_viaje;
begin
  leer_viaje(registro_viaje);
  while (registro_viaje.numero<>numero_salida) do
  begin
    agregar_ordenado_lista_viajes1(lista_viajes1,registro_viaje);
    leer_viaje(registro_viaje);
  end;
end;
procedure imprimir_registro_viaje(registro_viaje: t_registro_viaje; viaje: int16);
begin
  textcolor(green); write('El número de viaje del viaje '); textcolor(yellow); write(viaje);
textcolor(green); write(' es '); textcolor(red); writeln(registro_viaje.numero);
  textcolor(green); write('El código de auto del viaje '); textcolor(yellow); write(viaje);
textcolor(green); write(' es '); textcolor(red); writeln(registro_viaje.auto);
  textcolor(green); write('La dirección de origen del viaje '); textcolor(yellow);
write(viaje); textcolor(green); write(' es '); textcolor(red); writeln(registro_viaje.origen);
  textcolor(green); write('La dirección de destino del viaje '); textcolor(yellow);
write(viaje); textcolor(green); write(' es '); textcolor(red);
writeln(registro_viaje.destino);
  textcolor(green); write('Los kilómetros recorridos durante el viaje '); textcolor(yellow);
write(viaje); textcolor(green); write(' son '); textcolor(red);
writeln(registro_viaje.km:0:2);
end;
procedure imprimir_lista_viajes(lista_viajes1: t_lista_viajes);
var
  i: int16;
begin
  i:=0;
  while (lista_viajes1<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información del viaje '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_viaje(lista_viajes1^.ele,i);
    writeln();
    lista_viajes1:=lista_viajes1^.sig;
  end;
end;
procedure agregar_ordenado_lista_viajes2(var lista_viajes2: t_lista_viajes; registro_viaje:
t_registro_viaje);
var
  anterior, actual, nuevo: t_lista_viajes;
begin
  new(nuevo);
  nuevo^.ele:=registro_viaje;
```

```
  nuevo^.sig:=nil;
  actual:=lista_viajes2;
  while ((actual<>nil) and (actual^.ele.numero<nuevo^.ele.numero)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=lista_viajes2) then
    lista_viajes2:=nuevo
  else
    anterior^.sig:=nuevo;
  nuevo^.sig:=actual;
end;
procedure actualizar_maximos(km_auto: real; auto: int8; var km_max1, km_max2: real; var
auto_max1, auto_max2: int8);
begin
  if (km_auto>km_max1) then
  begin
    km_max2:=km_max1;
    auto_max2:=auto_max1;
    km_max1:=km_auto;
    auto_max1:=auto;
  end
  else
    if (km_auto>km_max2) then
    begin
      km_max2:=km_auto;
      auto_max2:=auto;
    end;
end;
procedure procesar_lista_viajes1(lista_viajes1: t_lista_viajes; var auto_max1, auto_max2:
int8; var lista_viajes2: t_lista_viajes);
var
  auto: int8;
  km_auto, km_max1, km_max2: real;
begin
  km_max1:=-9999999; km_max2:=-9999999;
  while (lista_viajes1<>nil) do
  begin
    auto:=lista_viajes1^.ele.auto;
    km_auto:=0;
    while ((lista_viajes1<>nil) and (lista_viajes1^.ele.auto=auto)) do
    begin
      km_auto:=km_auto+lista_viajes1^.ele.km;
      if (lista_viajes1^.ele.km>km_corte) then
        agregar_ordenado_lista_viajes2(lista_viajes2,lista_viajes1^.ele);
      lista_viajes1:=lista_viajes1^.sig;
    end;
    actualizar_maximos(km_auto,auto,km_max1,km_max2,auto_max1,auto_max2);
  end;
end;
var
  lista_viajes1, lista_viajes2: t_lista_viajes;
  auto_max1, auto_max2: int8;
begin
  randomize;
  lista_viajes1:=nil;
  auto_max1:=0; auto_max2:=0;
  lista_viajes2:=nil;
  cargar_lista_viajes1(lista_viajes1);
  if (lista_viajes1<>nil) then
  begin
    imprimir_lista_viajes(lista_viajes1);
    procesar_lista_viajes1(lista_viajes1,auto_max1,auto_max2,lista_viajes2);
    writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
```

```
    textcolor(green); write('Los dos códigos de auto que más kilómetros recorrieron son ');
textcolor(red); write(auto_max1); textcolor(green); write(' y '); textcolor(red);
writeln(auto_max2);
    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();
    imprimir_lista_viajes(lista_viajes2);
  end;
end.
```

Ejercicio 4.

Una maternidad dispone información sobre sus pacientes. De cada una, se conoce: nombre, apellido y peso registrado el primer día de cada semana de embarazo (a lo sumo, 42). La maternidad necesita un programa que analice esta información, determine e informe:

(a) *Para cada embarazada, la semana con mayor aumento de peso.*

(b) *El aumento de peso total de cada embarazada durante el embarazo.*

```
program TP7_E4;
{$codepage UTF8}
uses crt;
const
  semanas_total=42;
  nombre_salida='XXX';
type
  t_semana=1..semanas_total;
  t_vector_pesos=array[t_semana] of real;
  t_registro_paciente=record
    nombre: string;
    apellido: string;
    pesos: t_vector_pesos;
    semanas: int8;
  end;
  t_lista_pacientes=^t_nodo_pacientes;
  t_nodo_pacientes=record
    ele: t_registro_paciente;
    sig: t_lista_pacientes;
  end;
function random_string(length: int8): string;
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 cargar_vector_pesos(var vector_pesos: t_vector_pesos; var semanas: int8);
var
  peso: real;
begin
  semanas:=semanas+1;
  vector_pesos[semanas]:=50+random(11)/10;
  peso:=random(11)/10;
  while ((peso<>0) and (semanas<semanas_total)) do
  begin
    semanas:=semanas+1;
    vector_pesos[semanas]:=vector_pesos[semanas-1]+peso;
    if (semanas<semanas_total) then
      peso:=random(11)/10;
  end;
end;
procedure leer_paciente(var registro_paciente: t_registro_paciente);
var
  i: int8;
begin
  i:=random(100);
```

```
  if (i=0) then
    registro_paciente.nombre:=nombre_salida
  else
    registro_paciente.nombre:=random_string(5+random(6));
  if (registro_paciente.nombre<>nombre_salida) then
  begin
    registro_paciente.apellido:=random_string(5+random(6));
    registro_paciente.semanas:=0;
    cargar_vector_pesos(registro_paciente.pesos,registro_paciente.semanas);
  end;
end;
procedure agregar_adelante_lista_pacientes(var lista_pacientes: t_lista_pacientes;
registro_paciente: t_registro_paciente);
var
  nuevo: t_lista_pacientes;
begin
  new(nuevo);
  nuevo^.ele:=registro_paciente;
  nuevo^.sig:=lista_pacientes;
  lista_pacientes:=nuevo;
end;
procedure cargar_lista_pacientes(var lista_pacientes: t_lista_pacientes);
var
  registro_paciente: t_registro_paciente;
begin
  leer_paciente(registro_paciente);
  while (registro_paciente.nombre<>nombre_salida) do
  begin
    agregar_adelante_lista_pacientes(lista_pacientes,registro_paciente);
    leer_paciente(registro_paciente);
  end;
end;
procedure imprimir_registro_paciente(registro_paciente: t_registro_paciente; paciente: int16);
begin
  textcolor(green); write('El nombre del paciente '); textcolor(yellow); write(paciente);
textcolor(green); write(' es '); textcolor(red); writeln(registro_paciente.nombre);
  textcolor(green); write('El apellido del paciente '); textcolor(yellow); write(paciente);
textcolor(green); write(' es '); textcolor(red); writeln(registro_paciente.apellido);
  textcolor(green); write('La cantidad de semanas del paciente '); textcolor(yellow);
write(paciente); textcolor(green); write(' es '); textcolor(red);
writeln(registro_paciente.semanas);
end;
procedure imprimir_lista_pacientes(lista_pacientes: t_lista_pacientes);
var
  i: int16;
begin
  i:=0;
  while (lista_pacientes<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información del paciente '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_paciente(lista_pacientes^.ele,i);
    writeln();
    lista_pacientes:=lista_pacientes^.sig;
  end;
end;
procedure actualizar_maximo(aumento: real; semana: t_semana; var aumento_max: real; var
semana_max: t_semana);
begin
  if (aumento>aumento_max) then
  begin
    aumento_max:=aumento;
    semana_max:=semana;
  end;
end;
```

```
procedure procesar_vector_pesos(vector_pesos: t_vector_pesos; semanas: int8; var aumento_max:
real; var semana_max: t_semana);
var
  i: t_semana;
  aumento: real;
begin
  for i:= 2 to semanas do
  begin
    aumento:=vector_pesos[i]-vector_pesos[i-1];
    actualizar_maximo(aumento,i,aumento_max,semana_max);
  end;
end;
procedure procesar_lista_pacientes(lista_pacientes: t_lista_pacientes);
var
  semana_max: t_semana;
  i: int16;
  aumento_max: real;
begin
  i:=0;
  while (lista_pacientes<>nil) do
  begin
    i:=i+1;
    aumento_max:=-9999999; semana_max:=1;
    procesar_vector_pesos(lista_pacientes^.ele.pesos,lista_pacientes^.ele.semanas,aumento_max,
semana_max);
    textcolor(green); write('La semana con mayor aumento de peso de la embarazada ');
textcolor(yellow); write(i); textcolor(green); write(' fue la semana '); textcolor(red);
write(semana_max); textcolor(green); write(', con un aumento de peso de '); textcolor(red);
write(aumento_max:0:2); textcolor(green); writeln(' kilos');
    textcolor(green); write('El aumento de peso total de la embarazada '); textcolor(yellow);
write(i); textcolor(green); write(' fue de '); textcolor(red);
write(lista_pacientes^.ele.pesos[lista_pacientes^.ele.semanas]-
lista_pacientes^.ele.pesos[1]:0:2); textcolor(green); writeln(' kilos');
    writeln();
    lista_pacientes:=lista_pacientes^.sig;
  end;
end;
var
  lista_pacientes: t_lista_pacientes;
begin
  randomize;
  lista_pacientes:=nil;
  cargar_lista_pacientes(lista_pacientes);
  if (lista_pacientes<>nil) then
  begin
    imprimir_lista_pacientes(lista_pacientes);
    writeln(); textcolor(red); writeln('INCISOS (a) (b):'); writeln();
    procesar_lista_pacientes(lista_pacientes);
  end;
end.
```

Ejercicio 5.

Una empresa de transporte de cargas dispone de la información de su flota compuesta por 100 camiones. De cada camión, se tiene: patente, año de fabricación y capacidad (peso máximo en toneladas que puede transportar). Realizar un programa que lea y almacene la información de los viajes realizados por la empresa. De cada viaje, se lee: código de viaje, código del camión que lo realizó (1..100), distancia en kilómetros recorrida, ciudad de destino, año en que se realizó el viaje y DNI del chofer. La lectura finaliza cuando se lee el código de viaje -1. Una vez leída y almacenada la información, se pide:

(a) *Informar la patente del camión que más kilómetros recorridos posee y la patente del camión que menos kilómetros recorridos posee.*

(b) *Informar la cantidad de viajes que se han realizado en camiones con capacidad mayor a 30,5 toneladas y que posean una antigüedad mayor a 5 años al momento de realizar el viaje (año en que se realizó el viaje).*

(c) *Informar los códigos de los viajes realizados por choferes cuyo DNI tenga sólo dígitos impares.*

Nota: Los códigos de viaje no se repiten.

```
program TP7_E5;
{$codepage UTF8}
uses crt;
const
  anio_ini_camion=2000; anio_fin_camion=2010;
  camion_ini=1; camion_fin=100;
  anio_ini_viaje=2010; anio_fin_viaje=2020;
  codigo_viaje_salida=-1;
  toneladas_corte=30.5; antiguedad_corte=5;
type
  t_camion=camion_ini..camion_fin;
  t_registro_camion=record
    patente: string;
    anio: int16;
    capacidad: real;
  end;
  t_registro_viaje=record
    codigo_viaje: int16;
    codigo_camion: t_camion;
    distancia: real;
    destino: string;
    anio: int16;
    dni: int32;
  end;
  t_vector_camiones=array[t_camion] of t_registro_camion;
  t_lista_viajes=^t_nodo_viajes;
  t_nodo_viajes=record
    ele: t_registro_viaje;
    sig: t_lista_viajes;
  end;
function random_string(length: int8): string;
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_camion(var registro_camion: t_registro_camion);
begin
  registro_camion.patente:=random_string(2);
  registro_camion.anio:=anio_ini_camion+random(anio_fin_camion-anio_ini_camion+1);
  registro_camion.capacidad:=1+random(991)/10;
end;
procedure cargar_vector_camiones(var vector_camiones: t_vector_camiones);
var
  registro_camion: t_registro_camion;
  i: t_camion;
begin
  for i:= camion_ini to camion_fin do
  begin
    leer_camion(registro_camion);
    vector_camiones[i]:=registro_camion;
  end;
end;
procedure leer_viaje(var registro_viaje: t_registro_viaje);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_viaje.codigo_viaje:=codigo_viaje_salida
  else
    registro_viaje.codigo_viaje:=1+random(high(int16));
  if (registro_viaje.codigo_viaje<>codigo_viaje_salida) then
  begin
    registro_viaje.codigo_camion:=camion_ini+random(camion_fin);
    registro_viaje.distancia:=1+random(9991)/10;
    registro_viaje.destino:=random_string(5+random(6));
    registro_viaje.anio:=anio_ini_viaje+random(anio_fin_viaje-anio_ini_viaje+1);
    registro_viaje.dni:=10000000+random(40000001);
  end;
end;
procedure agregar_adelante_lista_viajes(var lista_viajes: t_lista_viajes; registro_viaje:
t_registro_viaje);
var
  nuevo: t_lista_viajes;
begin
  new(nuevo);
  nuevo^.ele:=registro_viaje;
  nuevo^.sig:=lista_viajes;
  lista_viajes:=nuevo;
end;
procedure cargar_lista_viajes(var lista_viajes: t_lista_viajes);
var
  registro_viaje: t_registro_viaje;
begin
  leer_viaje(registro_viaje);
  while (registro_viaje.codigo_viaje<>codigo_viaje_salida) do
  begin
    agregar_adelante_lista_viajes(lista_viajes,registro_viaje);
    leer_viaje(registro_viaje);
  end;
end;
procedure imprimir_registro_viaje(registro_viaje: t_registro_viaje; viaje: int16);
begin
  textcolor(green); write('El código de viaje del viaje '); textcolor(yellow); write(viaje);
textcolor(green); write(' es '); textcolor(red); writeln(registro_viaje.codigo_viaje);
```

```
  textcolor(green); write('El código de camión del viaje '); textcolor(yellow); write(viaje);
textcolor(green); write(' es '); textcolor(red); writeln(registro_viaje.codigo_camion);
  textcolor(green); write('La distancia del viaje '); textcolor(yellow); write(viaje);
textcolor(green); write(' es '); textcolor(red); writeln(registro_viaje.distancia:0:2);
  textcolor(green); write('El destino del viaje '); textcolor(yellow); write(viaje);
textcolor(green); write(' es '); textcolor(red); writeln(registro_viaje.destino);
  textcolor(green); write('El año del viaje '); textcolor(yellow); write(viaje);
textcolor(green); write(' es '); textcolor(red); writeln(registro_viaje.anio);
  textcolor(green); write('El DNI del chofer del viaje '); textcolor(yellow); write(viaje);
textcolor(green); write(' es '); textcolor(red); writeln(registro_viaje.dni);
end;
procedure imprimir_lista_viajes(lista_viajes: t_lista_viajes);
var
  i: int16;
begin
  i:=0;
  while (lista_viajes<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información del viaje '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_viaje(lista_viajes^.ele,i);
    writeln();
    lista_viajes:=lista_viajes^.sig;
  end;
end;
procedure actualizar_maximo(distancia: real; patente: string; var distancia_max: real; var
patente_max: string);
begin
  if (distancia>distancia_max) then
  begin
    distancia_max:=distancia;
    patente_max:=patente;
  end;
end;
procedure actualizar_minimo(distancia: real; patente: string; var distancia_min: real; var
patente_min: string);
begin
  if (distancia<distancia_min) then
  begin
    distancia_min:=distancia;
    patente_min:=patente;
  end;
end;
function contar_pares(dni: int32): boolean;
var
  pares: int8;
begin
  pares:=0;
  while ((dni<>0) and (pares=0)) do
  begin
    if (dni mod 2=0) then
      pares:=pares+1;
    dni:=dni div 10;
  end;
  contar_pares:=(pares=0);
end;
procedure procesar_lista_viajes(lista_viajes: t_lista_viajes; vector_camiones:
t_vector_camiones; var patente_max, patente_min: string; var viajes_corte: int16);
var
  distancia_max, distancia_min: real;
begin
  distancia_max:=-9999999; distancia_min:=9999999;
  while (lista_viajes<>nil) do
  begin
```

```
    actualizar_maximo(lista_viajes^.ele.distancia,vector_camiones[lista_viajes^.ele.codigo_cam
ion].patente,distancia_max,patente_max);
    actualizar_minimo(lista_viajes^.ele.distancia,vector_camiones[lista_viajes^.ele.codigo_cam
ion].patente,distancia_min,patente_min);
    if ((vector_camiones[lista_viajes^.ele.codigo_camion].capacidad>toneladas_corte) and
(lista_viajes^.ele.anio-
vector_camiones[lista_viajes^.ele.codigo_camion].anio>antiguedad_corte)) then
      viajes_corte:=viajes_corte+1;
    if (contar_pares(lista_viajes^.ele.dni)=true) then
    begin
      textcolor(green); write('El código de viaje de este viaje realizado por un chofer cuyo
DNI tiene sólo dígitos impares es '); textcolor(red); writeln(lista_viajes^.ele.codigo_viaje);
    end;
    lista_viajes:=lista_viajes^.sig;
  end;
end;
var
  vector_camiones: t_vector_camiones;
  lista_viajes: t_lista_viajes;
  viajes_corte: int16;
  patente_max, patente_min: string;
begin
  randomize;
  lista_viajes:=nil;
  patente_max:=''; patente_min:='';
  viajes_corte:=0;
  cargar_vector_camiones(vector_camiones);
  cargar_lista_viajes(lista_viajes);
  if (lista_viajes<>nil) then
  begin
    imprimir_lista_viajes(lista_viajes);
    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();
    procesar_lista_viajes(lista_viajes,vector_camiones,patente_max,patente_min,viajes_corte);
    writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
    textcolor(green); write('La patente del camión que más kilómetros recorridos posee es ');
textcolor(red); writeln(patente_max);
    textcolor(green); write('La patente del camión que menos kilómetros recorridos posee es
'); textcolor(red); writeln(patente_min);
    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();
    textcolor(green); write('La cantidad de viajes que se han realizado en camiones con
capacidad mayor a '); textcolor(yellow); write(toneladas_corte:0:2); textcolor(green); write('
toneladas y que posean una antigüedad mayor a '); textcolor(yellow); write(antiguedad_corte);
textcolor(green); write(' años al momento de realizar el viaje es '); textcolor(red);
write(viajes_corte);
  end;
end.
```

Ejercicio 6.

El Observatorio Astronómico de La Plata ha realizado un relevamiento sobre los distintos objetos astronómicos observados durante el año 2015. Los objetos se clasifican en 7 categorías: 1: estrellas, 2: planetas, 3: satélites, 4: galaxias, 5: asteroides, 6: cometas y 7: nebulosas. Al observar un objeto, se registran los siguientes datos: código del objeto, categoría del objeto (1..7), nombre del objeto, distancia a la Tierra (medida en años luz), nombre del descubridor y año de su descubrimiento.

(a) *Desarrollar un programa que lea y almacene la información de los objetos que han sido observados. Dicha información se lee hasta encontrar un objeto con código -1 (el cual no debe procesarse). La estructura generada debe mantener el orden en que fueron leídos los datos.*

(b) *Una vez leídos y almacenados todos los datos, se pide realizar un reporte con la siguiente información:*

> **(i)** *Los códigos de los dos objetos más lejanos de la tierra que se hayan observado.*
> **(ii)** *La cantidad de planetas descubiertos por "Galileo Galilei" antes del año 1600.*
> **(iii)** *La cantidad de objetos observados por cada categoría.*
> **(iv)** *Los nombres de las estrellas cuyos códigos de objeto poseen más dígitos pares que impares.*

```
program TP7_E6;
{$codepage UTF8}
uses crt;
const
  categoria_ini=1; categoria_fin=7;
  anio_ini=1500; anio_fin=2020;
  codigo_salida=-1;
  categoria_corte1=2; descubridor_corte='Galileo Galilei'; anio_corte=1600;
  categoria_corte2=1;
  vector_categorias: array[categoria_ini..categoria_fin] of string=('estrella', 'planeta',
'satelite', 'galaxia', 'asteroide', 'cometa', 'nebulosa');
type
  t_categoria=categoria_ini..categoria_fin;
  t_registro_objeto=record
    codigo: int16;
    categoria: t_categoria;
    nombre: string;
    distancia: real;
    descubridor: string;
    anio: int16;
  end;
  t_lista_objetos=^t_nodo_objetos;
  t_nodo_objetos=record
    ele: t_registro_objeto;
    sig: t_lista_objetos;
  end;
  t_vector_cantidades=array[t_categoria] of int16;
procedure inicializar_vector_cantidades(var vector_cantidades: t_vector_cantidades);
var
  i: t_categoria;
begin
  for i:= categoria_ini to categoria_fin do
    vector_cantidades[i]:=0;
```

```
end;
function random_string(length: int8): string;
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_objeto(var registro_objeto: t_registro_objeto);
var
  i: int8;
begin
  i:=random(101);
  if (i=0) then
    registro_objeto.codigo:=codigo_salida
  else
    registro_objeto.codigo:=1+random(high(int16));
  if (registro_objeto.codigo<>codigo_salida) then
  begin
    registro_objeto.categoria:=categoria_ini+random(categoria_fin);
    registro_objeto.nombre:=random_string(5+random(6));
    registro_objeto.distancia:=1+random(991)/10;
    if (i<=50) then
      registro_objeto.descubridor:=descubridor_corte
    else
      registro_objeto.descubridor:=random_string(5+random(6));
    registro_objeto.anio:=anio_ini+random(anio_fin-anio_ini+1);
  end;
end;
procedure agregar_atras_lista_objetos(var lista_objetos: t_lista_objetos; registro_objeto:
t_registro_objeto);
var
  nuevo, ult: t_lista_objetos;
begin
  new(nuevo);
  nuevo^.ele:=registro_objeto;
  nuevo^.sig:=nil;
  if (lista_objetos=nil) then
    lista_objetos:=nuevo
  else
  begin
    ult:=lista_objetos;
    while (ult^.sig<>nil) do
      ult:=ult^.sig;
    ult^.sig:=nuevo;
  end;
end;
procedure cargar_lista_objetos(var lista_objetos: t_lista_objetos);
var
  registro_objeto: t_registro_objeto;
begin
  leer_objeto(registro_objeto);
  while (registro_objeto.codigo<>codigo_salida) do
  begin
    agregar_atras_lista_objetos(lista_objetos,registro_objeto);
    leer_objeto(registro_objeto);
  end;
end;
procedure imprimir_registro_objeto(registro_objeto: t_registro_objeto; objeto: int16);
begin
  textcolor(green); write('El código de objeto del objeto '); textcolor(yellow);
write(objeto); textcolor(green); write(' es '); textcolor(red);
writeln(registro_objeto.codigo);
```

```
  textcolor(green); write('La categoría del objeto '); textcolor(yellow); write(objeto);
textcolor(green); write(' es '); textcolor(red); writeln(registro_objeto.categoria);
  textcolor(green); write('El nombre del objeto '); textcolor(yellow); write(objeto);
textcolor(green); write(' es '); textcolor(red); writeln(registro_objeto.nombre);
  textcolor(green); write('La distancia a la Tierra (medida en años luz) del objeto ');
textcolor(yellow); write(objeto); textcolor(green); write(' es '); textcolor(red);
writeln(registro_objeto.distancia:0:2);
  textcolor(green); write('El nombre del descubridor del objeto '); textcolor(yellow);
write(objeto); textcolor(green); write(' es '); textcolor(red);
writeln(registro_objeto.descubridor);
  textcolor(green); write('El año de descubrimiento del objeto '); textcolor(yellow);
write(objeto); textcolor(green); write(' es '); textcolor(red); writeln(registro_objeto.anio);
end;
procedure imprimir_lista_objetos(lista_objetos: t_lista_objetos);
var
  i: int16;
begin
  i:=0;
  while (lista_objetos<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información del objeto '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_objeto(lista_objetos^.ele,i);
    writeln();
    lista_objetos:=lista_objetos^.sig;
  end;
end;
procedure actualizar_maximos(distancia: real; codigo: int16; var distancia_max1,
distancia_max2: real; var codigo_max1, codigo_max2: int16);
begin
  if (distancia>distancia_max1) then
  begin
    distancia_max2:=distancia_max1;
    codigo_max2:=codigo_max1;
    distancia_max1:=distancia;
    codigo_max1:=codigo;
  end
  else
    if (distancia>distancia_max2) then
    begin
      distancia_max2:=distancia;
      codigo_max2:=codigo;
    end;
end;
function contar_pares_impares(codigo: int16): boolean;
var
  pares, impares: int8;
begin
  pares:=0; impares:=0;
  while (codigo<>0) do
  begin
    if (codigo mod 2=0) then
      pares:=pares+1
    else
      impares:=impares+1;
    codigo:=codigo div 10;
  end;
  contar_pares_impares:=(pares>impares);
end;
procedure procesar_lista_objetos(lista_objetos: t_lista_objetos; var codigo_max1, codigo_max2,
objetos_corte: int16; var vector_cantidades: t_vector_cantidades);
var
  distancia_max1, distancia_max2: real;
begin
  distancia_max1:=-9999999; distancia_max2:=-9999999;
```

```
  while (lista_objetos<>nil) do
  begin
    actualizar_maximos(lista_objetos^.ele.distancia,lista_objetos^.ele.codigo,distancia_max1,d
istancia_max2,codigo_max1,codigo_max2);
    if ((lista_objetos^.ele.categoria=categoria_corte1) and
(lista_objetos^.ele.descubridor=descubridor_corte) and (lista_objetos^.ele.anio<anio_corte))
then
      objetos_corte:=objetos_corte+1;
    vector_cantidades[lista_objetos^.ele.categoria]:=vector_cantidades[lista_objetos^.ele.cate
goria]+1;
    if ((lista_objetos^.ele.categoria=categoria_corte2) and
(contar_pares_impares(lista_objetos^.ele.codigo)=true)) then
    begin
      textcolor(green); write('El nombre de este objeto '); textcolor(yellow);
write(vector_categorias[categoria_corte2]); textcolor(green); write(' cuyo código de objeto
posee más dígitos pares que impares es '); textcolor(red); writeln(lista_objetos^.ele.nombre);
    end;
    lista_objetos:=lista_objetos^.sig;
  end;
end;
procedure imprimir_vector_cantidades(vector_cantidades: t_vector_cantidades);
var
  i: t_categoria;
begin
  for i:= categoria_ini to categoria_fin do
  begin
    textcolor(green); write('La cantidad de objetos observados por la categoría ');
textcolor(yellow); write(i); textcolor(green); write(' es '); textcolor(red);
writeln(vector_cantidades[i]);
  end;
end;
var
  vector_cantidades: t_vector_cantidades;
  lista_objetos: t_lista_objetos;
  codigo_max1, codigo_max2, objetos_corte: int16;
begin
  randomize;
  lista_objetos:=nil;
  codigo_max1:=0; codigo_max2:=0;
  objetos_corte:=0;
  inicializar_vector_cantidades(vector_cantidades);
  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
  cargar_lista_objetos(lista_objetos);
  if (lista_objetos<>nil) then
  begin
    imprimir_lista_objetos(lista_objetos);
    writeln(); textcolor(red); writeln('INCISO (b) (iv):'); writeln();
    procesar_lista_objetos(lista_objetos,codigo_max1,codigo_max2,objetos_corte,vector_cantidad
es);
    writeln(); textcolor(red); writeln('INCISO (b) (i):'); writeln();
    textcolor(green); write('Los códigos de los dos objetos más lejanos de la tierra que se
hayan observado son '); textcolor(red); write(codigo_max1); textcolor(green); write(' y ');
textcolor(red); writeln(codigo_max2);
    writeln(); textcolor(red); writeln('INCISO (b) (ii):'); writeln();
    textcolor(green); write('La cantidad de '); textcolor(yellow);
write(vector_categorias[categoria_corte1]); write('s'); textcolor(green); write(' descubiertos
por '); textcolor(yellow); write(descubridor_corte); textcolor(green); write(' antes del año
'); textcolor(yellow); write(anio_corte); textcolor(green); write(' es '); textcolor(red);
writeln(objetos_corte);
    writeln(); textcolor(red); writeln('INCISO (b) (iii):'); writeln();
    imprimir_vector_cantidades(vector_cantidades);
  end;
end.
```

Ejercicio 7.

La Facultad de Informática desea procesar la información de los alumnos que finalizaron la carrera de Analista Programador Universitario. Para ello, se deberá leer la información de cada alumno, a saber: número de alumno, apellido, nombres, dirección de correo electrónico, año de ingreso, año de egreso y las notas obtenidas en cada una de las 24 materias que aprobó (los aplazos no se registran).

(a) *Realizar un módulo que lea y almacene la información de los alumnos hasta que se ingrese el alumno con número de alumno -1, el cual no debe procesarse. Las 24 notas correspondientes a cada alumno deben quedar ordenadas de forma descendente.*

(b) *Una vez leída y almacenada la información del inciso (a), se solicita calcular e informar:*

> **(i)** *El promedio de notas obtenido por cada alumno.*
> **(ii)** *La cantidad de alumnos ingresantes 2012 cuyo número de alumno está compuesto, únicamente, por dígitos impares.*
> **(iii)** *El apellido, nombres y dirección de correo electrónico de los dos alumnos que más rápido se recibieron (o sea, que tardaron menos años).*

(c) *Realizar un módulo que, dado un número de alumno leído desde teclado, lo busque y elimine de la estructura generada en el inciso (a). El alumno puede no existir.*

```
program TP7_E7;
{$codepage UTF8}
uses crt;
const
  anio_ini=2000; anio_fin=2015;
  notas_total=24;
  numero_salida=-1;
  ingreso_corte=2012;
type
  t_nota=1..notas_total;
  t_vector_notas=array[t_nota] of int8;
  t_registro_alumno=record
    numero: int16;
    apellido: string;
    nombres: string;
    email: string;
    ingreso: int16;
    egreso: int16;
    notas: t_vector_notas;
  end;
  t_lista_alumnos=^t_nodo_alumnos;
  t_nodo_alumnos=record
    ele: t_registro_alumno;
    sig: t_lista_alumnos;
  end;
function random_string(length: int8): string;
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 inicializar_vector_notas(var vector_notas: t_vector_notas);
var
  i: t_nota;
begin
  for i:= 1 to notas_total do
    vector_notas[i]:=0;
end;
procedure agregar_ordenado_vector_notas(var vector_notas: t_vector_notas; notas, nota: int8);
var
  i, j: int8;
begin
  i:=1;
  while ((i<=notas) and (vector_notas[i]>nota)) do
    i:=i+1;
  for j:= notas downto i do
    vector_notas[j+1]:=vector_notas[j];
  vector_notas[i]:=nota;
end;
procedure cargar_vector_notas(var vector_notas: t_vector_notas);
var
  i, nota, notas: int8;
begin
  notas:=0;
  for i:= 1 to notas_total do
  begin
    nota:=4+random(7);
    agregar_ordenado_vector_notas(vector_notas,notas,nota);
    notas:=notas+1;
  end;
end;
procedure leer_alumno(var registro_alumno: t_registro_alumno);
var
  vector_emails: array[1..3] of string=('@gmail.com', '@hotmail.com', '@yahoo.com');
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_alumno.numero:=numero_salida
  else
    registro_alumno.numero:=1+random(high(int16));
  if (registro_alumno.numero<>numero_salida) then
  begin
    registro_alumno.apellido:=random_string(5+random(6));
    registro_alumno.nombres:=random_string(5+random(6))+' '+random_string(5+random(6));
    registro_alumno.email:=random_string(5+random(6))+'@'+vector_emails[1+random(3)];
    registro_alumno.ingreso:=anio_ini+random(anio_fin-anio_ini+1);
    registro_alumno.egreso:=registro_alumno.ingreso+5+random(6);
    inicializar_vector_notas(registro_alumno.notas);
    cargar_vector_notas(registro_alumno.notas);
  end;
end;
procedure agregar_adelante_lista_alumnos(var lista_alumnos: t_lista_alumnos; registro_alumno:
t_registro_alumno);
var
  nuevo: t_lista_alumnos;
begin
  new(nuevo);
  nuevo^.ele:=registro_alumno;
  nuevo^.sig:=lista_alumnos;
  lista_alumnos:=nuevo;
end;
procedure cargar_lista_alumnos(var lista_alumnos: t_lista_alumnos);
var
  registro_alumno: t_registro_alumno;
```

```
begin
  leer_alumno(registro_alumno);
  while (registro_alumno.numero<>numero_salida) do
  begin
    agregar_adelante_lista_alumnos(lista_alumnos,registro_alumno);
    leer_alumno(registro_alumno);
  end;
end;
procedure imprimir_registro_alumno(registro_alumno: t_registro_alumno; alumno: int16);
begin
  textcolor(green); write('El número de alumno del alumno '); textcolor(yellow);
write(alumno); textcolor(green); write(' es '); textcolor(red);
writeln(registro_alumno.numero);
  textcolor(green); write('El apellido del alumno '); textcolor(yellow); write(alumno);
textcolor(green); write(' es '); textcolor(red); writeln(registro_alumno.apellido);
  textcolor(green); write('Los nombres del alumno '); textcolor(yellow); write(alumno);
textcolor(green); write(' son '); textcolor(red); writeln(registro_alumno.nombres);
  textcolor(green); write('La dirección de correo electrónico del alumno ');
textcolor(yellow); write(alumno); textcolor(green); write(' es '); textcolor(red);
writeln(registro_alumno.email);
  textcolor(green); write('El año de ingreso del alumno '); textcolor(yellow); write(alumno);
textcolor(green); write(' es '); textcolor(red); writeln(registro_alumno.ingreso);
  textcolor(green); write('El año de egreso del alumno '); textcolor(yellow); write(alumno);
textcolor(green); write(' es '); textcolor(red); writeln(registro_alumno.egreso);
end;
procedure imprimir_lista_alumnos(lista_alumnos: t_lista_alumnos);
var
  i: int16;
begin
  i:=0;
  while (lista_alumnos<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información del alumno '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_alumno(lista_alumnos^.ele,i);
    writeln();
    lista_alumnos:=lista_alumnos^.sig;
  end;
end;
function calcular_promedio_notas(vector_notas: t_vector_notas): real;
var
  i: t_nota;
  suma: int16;
begin
  suma:=0;
  for i:= 1 to notas_total do
    suma:=suma+vector_notas[i];
  calcular_promedio_notas:=suma/notas_total;
end;
function contar_pares(numero: int16): boolean;
var
  pares: int8;
begin
  pares:=0;
  while ((numero<>0) and (pares=0)) do
  begin
    if (numero mod 2=0) then
      pares:=pares+1;
    numero:=numero div 10;
  end;
  contar_pares:=(pares=0);
end;
procedure actualizar_maximos(anios: int8; apellido, nombres, email: string; var anios_max1,
anios_max2: int8; var apellido_max1, apellido_max2, nombres_max1, nombres_max2, email_max1,
email_max2: string);
```

```
begin
  if (anios>anios_max1) then
  begin
    anios_max2:=anios_max1;
    apellido_max2:=apellido_max1;
    nombres_max2:=nombres_max1;
    email_max2:=email_max1;
    anios_max1:=anios;
    apellido_max1:=apellido;
    nombres_max1:=nombres;
    email_max1:=email;
  end
  else
    if (anios>anios_max2) then
    begin
      anios_max2:=anios;
      apellido_max2:=apellido;
      nombres_max2:=nombres;
      email_max2:=email;
    end;
end;
procedure procesar_lista_alumnos(lista_alumnos: t_lista_alumnos; var alumnos_corte: int16; var
apellido_max1, apellido_max2, nombres_max1, nombres_max2, email_max1, email_max2: string);
var
  anios, anios_max1, anios_max2: int8;
begin
  anios_max1:=low(int8); anios_max2:=low(int8);
  while (lista_alumnos<>nil) do
  begin
    textcolor(green); write('El promedio de notas obtenido por este alumno es ');
textcolor(red); writeln(calcular_promedio_notas(lista_alumnos^.ele.notas):0:2);
    if ((lista_alumnos^.ele.ingreso=ingreso_corte) and
(contar_pares(lista_alumnos^.ele.numero)=true)) then
      alumnos_corte:=alumnos_corte+1;
    anios:=lista_alumnos^.ele.egreso-lista_alumnos^.ele.ingreso;
    actualizar_maximos(anios,lista_alumnos^.ele.apellido,lista_alumnos^.ele.nombres,lista_alum
nos^.ele.email,anios_max1,anios_max2,apellido_max1,apellido_max2,nombres_max1,nombres_max2,ema
il_max1,email_max2);
    lista_alumnos:=lista_alumnos^.sig;
  end;
end;
procedure eliminar_lista_alumnos(var lista_alumnos: t_lista_alumnos; var ok: boolean; numero:
int16);
var
  anterior, actual: t_lista_alumnos;
begin
  actual:=lista_alumnos;
  while ((actual<>nil) and (actual^.ele.numero<>numero)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual<>nil) then
  begin
    if (actual=lista_alumnos) then
      lista_alumnos:=lista_alumnos^.sig
    else
      anterior^.sig:=actual^.sig;
    dispose(actual);
    ok:=true;
  end
end;
var
  lista_alumnos: t_lista_alumnos;
  alumnos_corte, numero: int16;
  ok: boolean;
```

```
  apellido_max1, apellido_max2, nombres_max1, nombres_max2, email_max1, email_max2: string;
begin
  randomize;
  lista_alumnos:=nil;
  alumnos_corte:=0;
  apellido_max1:=''; apellido_max2:=''; nombres_max1:=''; nombres_max2:=''; email_max1:='';
email_max2:='';
  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
  cargar_lista_alumnos(lista_alumnos);
  if (lista_alumnos<>nil) then
  begin
    imprimir_lista_alumnos(lista_alumnos);
    writeln(); textcolor(red); writeln('INCISO (b) (i):'); writeln();
    procesar_lista_alumnos(lista_alumnos,alumnos_corte,apellido_max1,apellido_max2,nombres_max
1,nombres_max2,email_max1,email_max2);
    writeln(); textcolor(red); writeln('INCISO (b) (ii):'); writeln();
    textcolor(green); write('La cantidad de alumnos ingresantes '); textcolor(yellow);
write(ingreso_corte); textcolor(green); write(' cuyo número de alumno está compuesto,
únicamente, por dígitos impares es '); textcolor(red); writeln(alumnos_corte);
    writeln(); textcolor(red); writeln('INCISO (b) (iii):'); writeln();
    textcolor(green); write('El apellido, nombres y dirección de correo electrónico del alumno
que más rápido se recibió son '); textcolor(red); write(apellido_max1); textcolor(green);
write(', '); textcolor(red); write(nombres_max1); textcolor(green); write(' y ');
textcolor(red); write(email_max1); textcolor(green); writeln(', respectivamente');
    textcolor(green); write('El apellido, nombres y dirección de correo electrónico del
segundo alumno que más rápido se recibió son '); textcolor(red); write(apellido_max2);
textcolor(green); write(', '); textcolor(red); write(nombres_max2); textcolor(green); write('
y '); textcolor(red); write(email_max2); textcolor(green); writeln(', respectivamente');
    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();
    numero:=1+random(high(int16));
    eliminar_lista_alumnos(lista_alumnos,ok,numero);
    if (ok=true) then
    begin
      textcolor(green); write('El número de alumno '); textcolor(yellow); write(numero);
textcolor(red); write(' SÍ'); textcolor(green); write(' fue encontrado y eliminado');
      imprimir_lista_alumnos(lista_alumnos);
    end
    else
    begin
      textcolor(green); write('El número de alumno '); textcolor(yellow); write(numero);
textcolor(red); write(' NO'); textcolor(green); write(' fue encontrado');
    end;
  end;
end.
```

Ejercicio 8.

Una entidad bancaria de la ciudad de La Plata solicita realizar un programa destinado a la administración de transferencias de dinero entre cuentas bancarias, efectuadas entre los meses de Enero y Noviembre del año 2018. El banco dispone de una lista de transferencias realizadas entre Enero y Noviembre del 2018. De cada transferencia, se conoce: número de cuenta origen, DNI de titular de cuenta origen, número de cuenta destino, DNI de titular de cuenta destino, fecha, hora, monto y el código del motivo de la transferencia (1: alquiler, 2: expensas, 3: facturas, 4: préstamo, 5: seguro, 6: honorarios y 7: varios). Esta estructura no posee orden alguno. Se pide:

(a) *Generar una nueva estructura que contenga sólo las transferencias a terceros (son aquellas en las que las cuentas origen y destino no pertenecen al mismo titular). Esta nueva estructura debe estar ordenada por número de cuenta origen.*

(b) *Una vez generada la estructura del inciso (a), utilizar dicha estructura para:*

> **(i)** *Calcular e informar, para cada cuenta de origen, el monto total transferido a terceros.*
> **(ii)** *Calcular e informar cuál es el código de motivo que más transferencias a terceros tuvo.*
> **(iii)** *Calcular e informar la cantidad de transferencias a terceros realizadas en el mes de Junio en las cuales el número de cuenta destino posea menos dígitos pares que impares.*

```
program TP7_E8;
{$codepage UTF8}
uses crt;
const
  dia_ini=1; dia_fin=31;
  mes_ini=1; mes_fin=11;
  motivo_ini=1; motivo_fin=7;
  numero_origen_salida=-1;
  mes_corte=6;
  vector_meses: array[mes_ini..mes_fin] of string=('Enero', 'Febrero', 'Marzo', 'Abril',
'Mayo', 'Junio', 'Julio', 'Agosto', 'Septiembre', 'Octubre', 'Noviembre');
type
  t_dia=dia_ini..dia_fin;
  t_mes=mes_ini..mes_fin;
  t_motivo=motivo_ini..motivo_fin;
  t_registro_fecha=record
    dia: t_dia;
    mes: t_mes;
  end;
  t_registro_hora=record
    hora: int8;
    minuto: int8;
  end;
  t_registro_transferencia=record
    numero_origen: int16;
    dni_origen: int32;
    numero_destino: int16;
    dni_destino: int32;
    fecha: t_registro_fecha;
    hora: t_registro_hora;
    monto: real;
    motivo: t_motivo;
```

```
  end;
  t_lista_transferencias=^t_nodo_transferencias;
  t_nodo_transferencias=record
    ele: t_registro_transferencia;
    sig: t_lista_transferencias;
  end;
  t_vector_cantidades=array[t_motivo] of int16;
function random_string(length: int8): string;
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_transferencia(var registro_transferencia: t_registro_transferencia);
var
  i: int8;
begin
  i:=random(101);
  if (i=0) then
    registro_transferencia.numero_origen:=numero_origen_salida
  else
    registro_transferencia.numero_origen:=1+random(high(int16));
  if (registro_transferencia.numero_origen<>numero_origen_salida) then
  begin
    registro_transferencia.dni_origen:=10000000+random(40000001);
    if (i<=50) then
    begin
      registro_transferencia.numero_destino:=registro_transferencia.numero_origen;
      registro_transferencia.dni_destino:=registro_transferencia.dni_origen;
    end
    else
    begin
      registro_transferencia.numero_destino:=1+random(high(int16));
      registro_transferencia.dni_destino:=10000000+random(40000001);
    end;
    registro_transferencia.fecha.dia:=dia_ini+random(dia_fin);
    registro_transferencia.fecha.mes:=mes_ini+random(mes_fin);
    registro_transferencia.hora.hora:=random(24);
    registro_transferencia.hora.minuto:=random(60);
    registro_transferencia.monto:=1+random(991)/10;
    registro_transferencia.motivo:=motivo_ini+random(motivo_fin);
  end;
end;
procedure agregar_adelante_lista_transferencias1(var lista_transferencias1:
t_lista_transferencias; registro_transferencia: t_registro_transferencia);
var
  nuevo: t_lista_transferencias;
begin
  new(nuevo);
  nuevo^.ele:=registro_transferencia;
  nuevo^.sig:=lista_transferencias1;
  lista_transferencias1:=nuevo;
end;
procedure cargar_lista_transferencias1(var lista_transferencias1: t_lista_transferencias);
var
  registro_transferencia: t_registro_transferencia;
begin
  leer_transferencia(registro_transferencia);
  while (registro_transferencia.numero_origen<>numero_origen_salida) do
  begin
    agregar_adelante_lista_transferencias1(lista_transferencias1,registro_transferencia);
    leer_transferencia(registro_transferencia);
```

```
  end;
end;
procedure imprimir_registro_transferencia(registro_transferencia: t_registro_transferencia;
transferencia: int16);
begin
  textcolor(green); write('El número de cuenta origen de la transferencia ');
textcolor(yellow); write(transferencia); textcolor(green); write(' es '); textcolor(red);
writeln(registro_transferencia.numero_origen);
  textcolor(green); write('El DNI de titular de la cuenta origen de la transferencia ');
textcolor(yellow); write(transferencia); textcolor(green); write(' es '); textcolor(red);
writeln(registro_transferencia.dni_origen);
  textcolor(green); write('El número de cuenta destino de la transferencia ');
textcolor(yellow); write(transferencia); textcolor(green); write(' es '); textcolor(red);
writeln(registro_transferencia.numero_destino);
  textcolor(green); write('El DNI de titular de la cuenta destino de la transferencia ');
textcolor(yellow); write(transferencia); textcolor(green); write(' es '); textcolor(red);
writeln(registro_transferencia.dni_destino);
  textcolor(green); write('La fecha de la transferencia '); textcolor(yellow);
write(transferencia); textcolor(green); write(' es '); textcolor(red);
write(registro_transferencia.fecha.dia); textcolor(green); write('/'); textcolor(red);
writeln(registro_transferencia.fecha.mes);
  textcolor(green); write('La hora de la transferencia '); textcolor(yellow);
write(transferencia); textcolor(green); write(' es '); textcolor(red);
write(registro_transferencia.hora.hora); textcolor(green); write(':'); textcolor(red);
writeln(registro_transferencia.hora.minuto);
  textcolor(green); write('El monto de la transferencia '); textcolor(yellow);
write(transferencia); textcolor(green); write(' es '); textcolor(red);
writeln(registro_transferencia.monto:0:2);
  textcolor(green); write('El código del motivo de la transferencia '); textcolor(yellow);
write(transferencia); textcolor(green); write(' es '); textcolor(red);
writeln(registro_transferencia.motivo);
end;
procedure imprimir_lista_transferencias(lista_transferencias: t_lista_transferencias);
var
  i: int16;
begin
  i:=0;
  while (lista_transferencias<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información de la transferencia '); textcolor(yellow);
write(i); textcolor(green); writeln(' es:');
    imprimir_registro_transferencia(lista_transferencias^.ele,i);
    writeln();
    lista_transferencias:=lista_transferencias^.sig;
  end;
end;
procedure agregar_ordenado_lista_transferencias2(var lista_transferencias2:
t_lista_transferencias; registro_transferencia: t_registro_transferencia);
var
  anterior, actual, nuevo: t_lista_transferencias;
begin
  new(nuevo);
  nuevo^.ele:=registro_transferencia;
  nuevo^.sig:=nil;
  actual:=lista_transferencias2;
  while ((actual<>nil) and (actual^.ele.numero_origen<nuevo^.ele.numero_origen)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=lista_transferencias2) then
    lista_transferencias2:=nuevo
  else
    anterior^.sig:=nuevo;
  nuevo^.sig:=actual;
```

```
end;
procedure procesar_lista_transferencias1(lista_transferencias1: t_lista_transferencias; var
lista_transferencias2: t_lista_transferencias);
begin
  while (lista_transferencias1<>nil) do
  begin
    if (lista_transferencias1^.ele.numero_origen<>lista_transferencias1^.ele.numero_destino)
then
      agregar_ordenado_lista_transferencias2(lista_transferencias2,lista_transferencias1^.ele)
;
    lista_transferencias1:=lista_transferencias1^.sig;
  end;
end;
procedure inicializar_vector_cantidades(var vector_cantidades: t_vector_cantidades);
var
  i: t_motivo;
begin
  for i:= motivo_ini to motivo_fin do
    vector_cantidades[i]:=0;
end;
function contar_pares_impares(numero: int16): boolean;
var
  pares, impares: int8;
begin
  pares:=0; impares:=0;
  while (numero<>0) do
  begin
    if (numero mod 2=0) then
      pares:=pares+1
    else
      impares:=impares+1;
    numero:=numero div 10;
  end;
  contar_pares_impares:=(pares<impares);
end;
procedure actualizar_maximo(cantidad: int16; motivo: t_motivo; var cantidad_max: int16; var
motivo_max: int8);
begin
  if (cantidad>cantidad_max) then
  begin
    cantidad_max:=cantidad;
    motivo_max:=motivo;
  end;
end;
procedure procesar_vector_cantidades(vector_cantidades: t_vector_cantidades; var motivo_max:
int8);
var
  i: t_motivo;
  cantidad_max: int16;
begin
  cantidad_max:=low(int16);
  for i:= motivo_ini to motivo_fin do
  begin
    actualizar_maximo(vector_cantidades[i],i,cantidad_max,motivo_max);
    textcolor(green); write('La cantidad de transferencias a terceros con motivo de la
transferencia ',i,' es '); textcolor(red); writeln(vector_cantidades[i]);
  end;
end;
procedure procesar_lista_transferencias2(lista_transferencias2: t_lista_transferencias; var
motivo_max: int8; var transferencias_corte: int16);
var
  vector_cantidades: t_vector_cantidades;
  numero_origen: int16;
  monto_total: real;
begin
  inicializar_vector_cantidades(vector_cantidades);
```

```
  while (lista_transferencias2<>nil) do
  begin
    numero_origen:=lista_transferencias2^.ele.numero_origen;
    monto_total:=0;
    while ((lista_transferencias2<>nil) and
(lista_transferencias2^.ele.numero_origen=numero_origen)) do
    begin
      monto_total:=monto_total+lista_transferencias2^.ele.monto;
      vector_cantidades[lista_transferencias2^.ele.motivo]:=vector_cantidades[lista_transferen
cias2^.ele.motivo]+1;
      if ((lista_transferencias2^.ele.fecha.mes=mes_corte) and
(contar_pares_impares(lista_transferencias2^.ele.numero_destino)=true)) then
        transferencias_corte:=transferencias_corte+1;
      lista_transferencias2:=lista_transferencias2^.sig;
    end;
    textcolor(green); write('El monto total transferido a terceros por la cuenta origen ');
textcolor(yellow); write(numero_origen); textcolor(green); write(' es '); textcolor(red);
writeln(monto_total:0:2);
  end;
  writeln();
  procesar_vector_cantidades(vector_cantidades,motivo_max);
end;
var
  lista_transferencias1, lista_transferencias2: t_lista_transferencias;
  motivo_max: int8;
  transferencias_corte: int16;
begin
  randomize;
  lista_transferencias1:=nil;
  lista_transferencias2:=nil;
  motivo_max:=0;
  transferencias_corte:=0;
  cargar_lista_transferencias1(lista_transferencias1);
  if (lista_transferencias1<>nil) then
  begin
    imprimir_lista_transferencias(lista_transferencias1);
    writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
    procesar_lista_transferencias1(lista_transferencias1,lista_transferencias2);
    if (lista_transferencias2<>nil) then
    begin
      imprimir_lista_transferencias(lista_transferencias2);
      writeln(); textcolor(red); writeln('INCISO (b) (i):'); writeln();
      procesar_lista_transferencias2(lista_transferencias2,motivo_max,transferencias_corte);
      writeln(); textcolor(red); writeln('INCISO (b) (ii):'); writeln();
      textcolor(green); write('El código de motivo que más transferencias a terceros tuvo es
'); textcolor(red); writeln(motivo_max);
      writeln(); textcolor(red); writeln('INCISO (b) (iii):'); writeln();
      textcolor(green); write('La cantidad de transferencias a terceros realizadas en el mes
de '); textcolor(yellow); write(vector_meses[mes_corte]); textcolor(green); write(' en las
cuales el número de cuenta destino posee menos dígitos pares que impares es ');
textcolor(red); write(transferencias_corte);
    end;
  end;
end.
```

Ejercicio 9.

Un cine posee la lista de películas que proyectará durante el mes de Febrero. De cada película, se tiene: código de película, título de la película, código de género (1: acción, 2: aventura, 3: drama, 4: suspenso, 5: comedia, 6: bélica, 7: documental y 8: terror) y puntaje promedio otorgado por las críticas. Dicha estructura no posee orden alguno. Se pide:

(a) *Actualizar (en la lista que se dispone) el puntaje promedio otorgado por las críticas. Para ello, se debe leer desde teclado las críticas que han realizado críticos de cine. De cada crítica, se lee: DNI del crítico, apellido y nombre del crítico, código de película y el puntaje otorgado. La lectura finaliza cuando se lee el código de película -1 y la información viene ordenada por código de película.*

(b) *Informar el código de género que más puntaje obtuvo entre todas las críticas.*

(c) *Informar el apellido y nombre de aquellos críticos que posean la misma cantidad de dígitos pares que impares en su DNI.*

(d) *Realizar un módulo que elimine de la lista que se dispone una película cuyo código se recibe como parámetro (el mismo puede no existir).*

```
program TP7_E9;
{$codepage UTF8}
uses crt;
const
  genero_ini=1; genero_fin=8;
  codigo_salida=-1;
type
  t_genero=genero_ini..genero_fin;
  t_registro_pelicula=record
    codigo: int16;
    titulo: string;
    genero: t_genero;
    puntaje: real;
  end;
  t_lista_peliculas=^t_nodo_peliculas;
  t_nodo_peliculas=record
    ele: t_registro_pelicula;
    sig: t_lista_peliculas;
  end;
  t_registro_critica=record
    dni: int32;
    apellido: string;
    nombre: string;
    codigo: int16;
    puntaje: real;
  end;
  t_vector_puntajes=array[t_genero] of real;
function random_string(length: int8): string;
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_pelicula(var registro_pelicula: t_registro_pelicula);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_pelicula.codigo:=codigo_salida
  else
    registro_pelicula.codigo:=1+random(high(int16));
  if (registro_pelicula.codigo<>codigo_salida) then
  begin
    registro_pelicula.titulo:=random_string(5+random(6));
    registro_pelicula.genero:=genero_ini+random(genero_fin);
    registro_pelicula.puntaje:=1+random(91)/10;
  end;
end;
procedure agregar_adelante_lista_peliculas(var lista_peliculas: t_lista_peliculas;
registro_pelicula: t_registro_pelicula);
var
  nuevo: t_lista_peliculas;
begin
  new(nuevo);
  nuevo^.ele:=registro_pelicula;
  nuevo^.sig:=lista_peliculas;
  lista_peliculas:=nuevo;
end;
procedure cargar_lista_peliculas(var lista_peliculas: t_lista_peliculas);
var
  registro_pelicula: t_registro_pelicula;
begin
  leer_pelicula(registro_pelicula);
  while (registro_pelicula.codigo<>codigo_salida) do
  begin
    agregar_adelante_lista_peliculas(lista_peliculas,registro_pelicula);
    leer_pelicula(registro_pelicula);
  end;
end;
procedure imprimir_registro_pelicula(registro_pelicula: t_registro_pelicula; pelicula: int16);
begin
  textcolor(green); write('El código de película de la película '); textcolor(yellow);
write(pelicula); textcolor(green); write(' es '); textcolor(red);
writeln(registro_pelicula.codigo);
  textcolor(green); write('El título de la película '); textcolor(yellow); write(pelicula);
textcolor(green); write(' es '); textcolor(red); writeln(registro_pelicula.titulo);
  textcolor(green); write('El código de género de la película '); textcolor(yellow);
write(pelicula); textcolor(green); write(' es '); textcolor(red);
writeln(registro_pelicula.genero);
  textcolor(green); write('El puntaje promedio otorgado por las críticas de la película ');
textcolor(yellow); write(pelicula); textcolor(green); write(' es '); textcolor(red);
writeln(registro_pelicula.puntaje:0:2);
end;
procedure imprimir_lista_peliculas(lista_peliculas: t_lista_peliculas);
var
  i: int16;
begin
  i:=0;
  while (lista_peliculas<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información de la película '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_pelicula(lista_peliculas^.ele,i);
    writeln();
    lista_peliculas:=lista_peliculas^.sig;
  end;
```

```
end;
procedure leer_critica(var registro_critica: t_registro_critica; codigo: int16);
var
  i: int8;
begin
  i:=random(101);
  if (i=0) then
    registro_critica.codigo:=codigo_salida
  else if (i<=50) then
    registro_critica.codigo:=codigo
  else
    registro_critica.codigo:=codigo+random(high(int16)-(codigo-1));
  if (registro_critica.codigo<>codigo_salida) then
  begin
    registro_critica.dni:=10000000+random(40000001);
    registro_critica.apellido:=random_string(5+random(6));
    registro_critica.nombre:=random_string(5+random(6));
    registro_critica.puntaje:=1+random(91)/10;
  end;
end;
function contar_pares_impares(dni: int32): boolean;
var
  pares, impares: int8;
begin
  pares:=0; impares:=0;
  while (dni<>0) do
  begin
    if (dni mod 2=0) then
      pares:=pares+1
    else
      impares:=impares+1;
    dni:=dni div 10;
  end;
  contar_pares_impares:=(pares=impares);
end;
procedure buscar_lista_peliculas(lista_peliculas: t_lista_peliculas; codigo: int16; puntaje:
real);
begin
  while ((lista_peliculas<>nil) and (lista_peliculas^.ele.codigo<>codigo)) do
    lista_peliculas:=lista_peliculas^.sig;
  if (lista_peliculas<>nil) then
    lista_peliculas^.ele.puntaje:=puntaje;
end;
procedure actualizar_lista_peliculas(var lista_peliculas: t_lista_peliculas);
var
  registro_critica: t_registro_critica;
  codigo, criticas: int16;
  puntaje: real;
begin
  codigo:=1;
  leer_critica(registro_critica,codigo);
  while (registro_critica.codigo<>codigo_salida) do
  begin
    codigo:=registro_critica.codigo;
    puntaje:=0; criticas:=0;
    while ((registro_critica.codigo<>codigo_salida) and (registro_critica.codigo=codigo)) do
    begin
      puntaje:=puntaje+registro_critica.puntaje;
      criticas:=criticas+1;
      if (contar_pares_impares(registro_critica.dni)=true) then
      begin
        textcolor(green); write('El apellido y nombre de este crítico que posee la misma
cantidad de dígitos pares e impares en su DNI son '); textcolor(red);
writeln(registro_critica.apellido,' ',registro_critica.nombre);
      end;
      leer_critica(registro_critica,codigo);
```

```
    end;
    buscar_lista_peliculas(lista_peliculas,codigo,puntaje/criticas);
  end;
end;
procedure inicializar_vector_puntajes(var vector_puntajes: t_vector_puntajes);
var
  i: t_genero;
begin
  for i:= genero_ini to genero_fin do
    vector_puntajes[i]:=0;
end;
procedure actualizar_maximo(puntaje: real; genero: t_genero; var puntaje_max: real; var
genero_max: int8);
begin
  if (puntaje>puntaje_max) then
  begin
    puntaje_max:=puntaje;
    genero_max:=genero;
  end;
end;
procedure procesar_vector_puntajes(vector_puntajes: t_vector_puntajes; var genero_max: int8);
var
  i: t_genero;
  puntaje_max: real;
begin
  puntaje_max:=-9999999;
  for i:= genero_ini+1 to genero_fin do
  begin
    actualizar_maximo(vector_puntajes[i],i,puntaje_max,genero_max);
    textcolor(green); write('El puntaje que obtuvo entre todas las críticas el código de
género ',i,' es '); textcolor(red); writeln(vector_puntajes[i]:0:2);
  end;
end;
procedure procesar_lista_peliculas(lista_peliculas: t_lista_peliculas; var genero_max: int8);
var
  vector_puntajes: t_vector_puntajes;
begin
  inicializar_vector_puntajes(vector_puntajes);
  while (lista_peliculas<>nil) do
  begin
    vector_puntajes[lista_peliculas^.ele.genero]:=vector_puntajes[lista_peliculas^.ele.genero]
+lista_peliculas^.ele.puntaje;
    lista_peliculas:=lista_peliculas^.sig;
  end;
  procesar_vector_puntajes(vector_puntajes,genero_max);
end;
procedure eliminar_lista_peliculas(var lista_peliculas: t_lista_peliculas; var ok: boolean;
codigo: int16);
var
  anterior, actual: t_lista_peliculas;
begin
  actual:=lista_peliculas;
  while ((actual<>nil) and (actual^.ele.codigo<>codigo)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual<>nil) then
  begin
    if (actual=lista_peliculas) then
      lista_peliculas:=lista_peliculas^.sig
    else
      anterior^.sig:=actual^.sig;
    dispose(actual);
    ok:=true;
  end
```

```
end;
var
  lista_peliculas: t_lista_peliculas;
  genero_max: int8;
  codigo: int16;
  ok: boolean;
begin
  randomize;
  lista_peliculas:=nil;
  genero_max:=0;
  cargar_lista_peliculas(lista_peliculas);
  if (lista_peliculas<>nil) then
  begin
    imprimir_lista_peliculas(lista_peliculas);
    writeln(); textcolor(red); writeln('INCISO (a) (c):'); writeln();
    actualizar_lista_peliculas(lista_peliculas);
    imprimir_lista_peliculas(lista_peliculas);
    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();
    procesar_lista_peliculas(lista_peliculas,genero_max);
    writeln(); textcolor(green); write('El código de género que más puntaje obtuvo entre todas
las críticas es '); textcolor(red); writeln(genero_max);
    writeln(); textcolor(red); writeln('INCISO (d):'); writeln();
    codigo:=1+random(high(int16));
    eliminar_lista_peliculas(lista_peliculas,ok,codigo);
    if (ok=true) then
    begin
      textcolor(green); write('El código de película '); textcolor(yellow); write(codigo);
textcolor(red); write(' SÍ'); textcolor(green); write(' fue encontrado y eliminado');
      imprimir_lista_peliculas(lista_peliculas);
    end
    else
    begin
      textcolor(green); write('El código de película '); textcolor(yellow); write(codigo);
textcolor(red); write(' NO'); textcolor(green); write(' fue encontrado');
    end;
  end;
end.
```

Ejercicio 10.

Una compañía de venta de insumos agrícolas desea procesar la información de las empresas a las que les provee sus productos. De cada empresa, se conoce su código, nombre, si es estatal o privada, nombre de la ciudad donde está radicada y los cultivos que realiza (a lo sumo, 20). Para cada cultivo de la empresa, se registra: tipo de cultivo (trigo, maíz, soja, girasol, etc.), cantidad de hectáreas dedicadas y la cantidad de meses que lleva el ciclo de cultivo.

(a) *Realizar un programa que lea la información de las empresas y sus cultivos. Dicha información se ingresa hasta que llegue una empresa con código -1 (la cual no debe procesarse). Para cada empresa se leen todos sus cultivos, hasta que se ingrese un cultivo con 0 hectáreas (que no debe procesarse).*

(b) *Una vez leída y almacenada la información, calcular e informar:*

> **(i)** *Nombres de las empresas radicadas en "San Miguel del Monte" que cultivan trigo y cuyo código de empresa posee, al menos, dos ceros.*
> **(ii)** *La cantidad de hectáreas dedicadas al cultivo de soja y qué porcentaje representa con respecto al total de hectáreas.*
> **(iii)** *La empresa que dedica más tiempo al cultivo de maíz.*

(c) *Realizar un módulo que incremente en un mes los tiempos de cultivos de girasol de menos de 5 hectáreas de todas las empresas que no son estatales.*

```
program TP7_E10;
{$codepage UTF8}
uses crt;
const
  cultivos_total=20;
  codigo_salida=-1; hectareas_salida=0;
  ciudad_corte='San Miguel del Monte'; tipo_corte1='trigo'; digito_corte=0;
cantidad_digito_corte=2;
  tipo_corte2='soja';
  tipo_corte3='maiz';
  tipo_corte4='girasol'; hectareas_corte4=5;
type
  t_cultivo=1..cultivos_total;
  t_registro_cultivo=record
    tipo: string;
    hectareas: real;
    meses: int8;
  end;
  t_vector_cultivos=array[t_cultivo] of t_registro_cultivo;
  t_registro_empresa=record
    codigo: int16;
    nombre: string;
    estatal: boolean;
    ciudad: string;
    cultivos: t_vector_cultivos;
    cantidad_cultivos: int8;
  end;
  t_lista_empresas=^t_nodo_empresas;
  t_nodo_empresas=record
    ele: t_registro_empresa;
    sig: t_lista_empresas;
  end;
```

```
function random_string(length: int8): string;
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_cultivo(var registro_cultivo: t_registro_cultivo);
var
  vector_tipo: array[1..4] of string=('trigo', 'maiz', 'soja', 'girasol');
  i: int8;
begin
  i:=random(cultivos_total);
  if (i=0) then
    registro_cultivo.hectareas:=hectareas_salida
  else
    registro_cultivo.hectareas:=1+random(991)/10;
  if (registro_cultivo.hectareas<>hectareas_salida) then
  begin
    registro_cultivo.tipo:=vector_tipo[1+random(4)];
    registro_cultivo.meses:=1+random(12);
  end;
end;
procedure cargar_vector_cultivos(var vector_cultivos: t_vector_cultivos; var cultivos: int8);
var
  registro_cultivo: t_registro_cultivo;
begin
  leer_cultivo(registro_cultivo);
  while ((registro_cultivo.hectareas<>hectareas_salida) and (cultivos<cultivos_total)) do
  begin
    cultivos:=cultivos+1;
    vector_cultivos[cultivos]:=registro_cultivo;
    if (cultivos<cultivos_total) then
      leer_cultivo(registro_cultivo);
  end;
end;
procedure leer_empresa(var registro_empresa: t_registro_empresa);
var
  i: int8;
begin
  i:=random(101);
  if (i=0) then
    registro_empresa.codigo:=codigo_salida
  else
    registro_empresa.codigo:=1+random(high(int16));
  if (registro_empresa.codigo<>codigo_salida) then
  begin
    registro_empresa.nombre:=random_string(5+random(6));
    registro_empresa.estatal:=(random(2)=0);
    if (i<=50) then
      registro_empresa.ciudad:=ciudad_corte
    else
      registro_empresa.ciudad:=random_string(5+random(6));
    registro_empresa.cantidad_cultivos:=0;
    cargar_vector_cultivos(registro_empresa.cultivos,registro_empresa.cantidad_cultivos);
  end;
end;
procedure agregar_adelante_lista_empresas(var lista_empresas: t_lista_empresas;
registro_empresa: t_registro_empresa);
var
  nuevo: t_lista_empresas;
begin
  new(nuevo);
```

```
  nuevo^.ele:=registro_empresa;
  nuevo^.sig:=lista_empresas;
  lista_empresas:=nuevo;
end;
procedure cargar_lista_empresas(var lista_empresas: t_lista_empresas);
var
  registro_empresa: t_registro_empresa;
begin
  leer_empresa(registro_empresa);
  while (registro_empresa.codigo<>codigo_salida) do
  begin
    agregar_adelante_lista_empresas(lista_empresas,registro_empresa);
    leer_empresa(registro_empresa);
  end;
end;
procedure imprimir_registro_cultivo(registro_cultivo: t_registro_cultivo; cultivo: int8;
empresa: int16);
begin
  textcolor(green); write('El tipo de cultivo del cultivo '); textcolor(yellow);
write(cultivo); textcolor(green); write(' de la empresa '); textcolor(yellow); write(empresa);
textcolor(green); write(' es '); textcolor(red); writeln(registro_cultivo.tipo);
  textcolor(green); write('La cantidad de hectáreas del cultivo '); textcolor(yellow);
write(cultivo); textcolor(green); write(' de la empresa '); textcolor(yellow); write(empresa);
textcolor(green); write(' es '); textcolor(red); writeln(registro_cultivo.hectareas:0:2);
  textcolor(green); write('La cantidad de meses del cultivo '); textcolor(yellow);
write(cultivo); textcolor(green); write(' de la empresa '); textcolor(yellow); write(empresa);
textcolor(green); write(' es '); textcolor(red); writeln(registro_cultivo.meses);
end;
procedure imprimir_vector_cultivos(vector_cultivos: t_vector_cultivos; cultivos: int8;
empresa: int16);
var
  i: int8;
begin
  for i:= 1 to cultivos do
    imprimir_registro_cultivo(vector_cultivos[i],i,empresa);
end;
procedure imprimir_registro_empresa(registro_empresa: t_registro_empresa; empresa: int16);
begin
  textcolor(green); write('El código de la empresa '); textcolor(yellow); write(empresa);
textcolor(green); write(' es '); textcolor(red); writeln(registro_empresa.codigo);
  textcolor(green); write('El nombre de la empresa '); textcolor(yellow); write(empresa);
textcolor(green); write(' es '); textcolor(red); writeln(registro_empresa.nombre);
  textcolor(green); write('La empresa '); textcolor(yellow); write(empresa); textcolor(green);
write(' es '); textcolor(red); if (registro_empresa.estatal=true) then writeln('estatal') else
writeln('privada');
  textcolor(green); write('La ciudad de la empresa '); textcolor(yellow); write(empresa);
textcolor(green); write(' es '); textcolor(red); writeln(registro_empresa.ciudad);
  textcolor(green); write('La cantidad de cultivos de la empresa '); textcolor(yellow);
write(empresa); textcolor(green); write(' es '); textcolor(red);
writeln(registro_empresa.cantidad_cultivos);
  imprimir_vector_cultivos(registro_empresa.cultivos,registro_empresa.cantidad_cultivos,empres
a);
end;
procedure imprimir_lista_empresas(lista_empresas: t_lista_empresas);
var
  i: int16;
begin
  i:=0;
  while (lista_empresas<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información de la empresa '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_empresa(lista_empresas^.ele,i);
    writeln();
    lista_empresas:=lista_empresas^.sig;
```

```
  end;
end;
procedure procesar_vector_cultivos(var vector_cultivos: t_vector_cultivos; cultivos: int8; var
cumple_tipo_corte1: boolean; var hectareas_corte2, hectareas_total: real; var meses_corte3:
int8; estatal: boolean);
var
  i: int8;
begin
  for i:= 1 to cultivos do
  begin
    hectareas_total:=hectareas_total+vector_cultivos[i].hectareas;
    if ((vector_cultivos[i].tipo=tipo_corte1) and (cumple_tipo_corte1=false)) then
      cumple_tipo_corte1:=true
    else if (vector_cultivos[i].tipo=tipo_corte2) then
      hectareas_corte2:=hectareas_corte2+vector_cultivos[i].hectareas
    else if (vector_cultivos[i].tipo=tipo_corte3) then
      meses_corte3:=meses_corte3+vector_cultivos[i].meses
    else if ((vector_cultivos[i].tipo=tipo_corte4) and
(vector_cultivos[i].hectareas<hectareas_corte4) and (estatal=false)) then
      vector_cultivos[i].meses:=vector_cultivos[i].meses+1;
  end;
end;
function contar_digitos(codigo: int16): boolean;
var
  digitos: int8;
begin
  digitos:=0;
  while (codigo<>0) do
  begin
    if (codigo mod 10=digito_corte) then
      digitos:=digitos+1;
    codigo:=codigo div 10;
  end;
  contar_digitos:=(digitos>=cantidad_digito_corte);
end;
procedure actualizar_maximo(meses_corte3: int8; nombre: string; var meses_corte3_max: int8;
var nombre_corte3: string);
begin
  if (meses_corte3>meses_corte3_max) then
  begin
    meses_corte3_max:=meses_corte3;
    nombre_corte3:=nombre;
  end;
end;
procedure procesar_lista_empresas(lista_empresas: t_lista_empresas; var hectareas_corte2,
porcentaje_corte2: real; var nombre_corte3: string);
var
  meses_corte3, meses_corte3_max: int8;
  hectareas_total: real;
  cumple_tipo_corte1: boolean;
begin
  cumple_tipo_corte1:=false;
  hectareas_total:=0;
  meses_corte3_max:=low(int8);
  while (lista_empresas<>nil) do
  begin
    meses_corte3:=0;
    procesar_vector_cultivos(lista_empresas^.ele.cultivos,lista_empresas^.ele.cantidad_cultivo
s,cumple_tipo_corte1,hectareas_corte2,hectareas_total,meses_corte3,lista_empresas^.ele.estatal
);
    if ((lista_empresas^.ele.ciudad=ciudad_corte) and (cumple_tipo_corte1=true) and
(contar_digitos(lista_empresas^.ele.codigo)=true)) then
    begin
      textcolor(green); write('El nombre de esta empresa radicada en '); textcolor(yellow);
write(ciudad_corte); textcolor(green); write(' que cultiva '); textcolor(yellow);
write(tipo_corte1); textcolor(green); write(' y cuyo código de empresa posee, al menos, ');
```

```
textcolor(yellow); write(cantidad_digito_corte); textcolor(green); write(' dígitos ');
textcolor(yellow); write(digito_corte); textcolor(green); write(' es '); textcolor(red);
writeln(lista_empresas^.ele.nombre);
    end;
    actualizar_maximo(meses_corte3,lista_empresas^.ele.nombre,meses_corte3_max,nombre_corte3);
    lista_empresas:=lista_empresas^.sig;
  end;
  if (hectareas_total>0) then
    porcentaje_corte2:=hectareas_corte2/hectareas_total*100;
end;
var
  lista_empresas: t_lista_empresas;
  hectareas_corte2, porcentaje_corte2: real;
  nombre_corte3: string;
begin
  randomize;
  lista_empresas:=nil;
  hectareas_corte2:=0; porcentaje_corte2:=0;
  nombre_corte3:='';
  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
  cargar_lista_empresas(lista_empresas);
  if (lista_empresas<>nil) then
  begin
    imprimir_lista_empresas(lista_empresas);
    writeln(); textcolor(red); writeln('INCISO (b) (i):'); writeln();
    procesar_lista_empresas(lista_empresas,hectareas_corte2,porcentaje_corte2,nombre_corte3);
    writeln(); textcolor(red); writeln('INCISO (b) (ii):'); writeln();
    textcolor(green); write('La cantidad de hectáreas dedicadas al cultivo de ');
textcolor(yellow); write(tipo_corte2); textcolor(green); write(' y el que porcentaje
representa con respecto al total de hectáreas son '); textcolor(red);
write(hectareas_corte2:0:2); textcolor(green); write(' y '); textcolor(red);
write(porcentaje_corte2:0:2); textcolor(green); writeln('%, respectivamente');
    writeln(); textcolor(red); writeln('INCISO (b) (iii):'); writeln();
    textcolor(green); write('La empresa que dedica más tiempo al cultivo de ');
textcolor(yellow); write(tipo_corte3); textcolor(green); write(' es '); textcolor(red);
writeln(nombre_corte3);
    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();
    imprimir_lista_empresas(lista_empresas);
  end;
end.
```

Ejercicio 11.

Realizar un programa para una empresa productora que necesita organizar 100 eventos culturales. De cada evento, se dispone la siguiente información: nombre del evento, tipo de evento (1: música, 2: cine, 3: obra de teatro, 4: unipersonal y 5: monólogo), lugar del evento, cantidad máxima de personas permitidas para el evento y costo de la entrada. Se pide:

(a) *Generar una estructura con las ventas de entradas para tales eventos culturales. De cada venta, se debe guardar: código de venta, número de evento (1..100), DNI del comprador y cantidad de entradas adquiridas. La lectura de las ventas finaliza con código de venta -1.*

(b) *Una vez leída y almacenada la información de las ventas, calcular e informar:*

> **(i)** *El nombre y lugar de los dos eventos que han tenido menos recaudación.*
> **(ii)** *La cantidad de entradas vendidas cuyo comprador contiene, en su DNI, más dígitos pares que impares y que sean para el evento de tipo "obra de teatro".*
> **(iii)** *Si la cantidad de entradas vendidas para el evento número 50 alcanzó la cantidad máxima de personas permitidas.*

```
program TP7_E11;
{$codepage UTF8}
uses crt;
const
  eventos_total=100;
  tipo_ini=1; tipo_fin=5;
  codigo_salida=-1;
  tipo_corte=3;
  evento_corte=50;
  vector_tipos: array[tipo_ini..tipo_fin] of string=('musica', 'cine', 'obra de teatro',
'unipersonal', 'monologo');
type
  t_evento=1..eventos_total;
  t_tipo=tipo_ini..tipo_fin;
  t_registro_evento=record
    nombre: string;
    tipo: t_tipo;
    lugar: string;
    personas: int16;
    costo: real;
  end;
  t_registro_venta=record
    codigo: int16;
    evento: t_evento;
    dni: int32;
    entradas: int16;
  end;
  t_vector_eventos=array[t_evento] of t_registro_evento;
  t_vector_entradas=array[t_evento] of int16;
  t_lista_ventas=^t_nodo_ventas;
  t_nodo_ventas=record
    ele: t_registro_venta;
    sig: t_lista_ventas;
  end;
function random_string(length: int8): string;
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_evento(var registro_evento: t_registro_evento);
begin
  registro_evento.nombre:=random_string(5+random(6));
  registro_evento.tipo:=tipo_ini+random(tipo_fin);
  registro_evento.lugar:=random_string(5+random(6));
  registro_evento.personas:=10+random(11);
  registro_evento.costo:=1+random(991)/10;
end;
procedure cargar_vector_eventos(var vector_eventos: t_vector_eventos);
var
  i: t_evento;
begin
  for i:= 1 to eventos_total do
    leer_evento(vector_eventos[i]);
end;
procedure leer_venta(var registro_venta: t_registro_venta);
var
  i: int8;
begin
  i:=random(1000);
  if (i=0) then
    registro_venta.codigo:=codigo_salida
  else
    registro_venta.codigo:=1+random(high(int16));
  if (registro_venta.codigo<>codigo_salida) then
  begin
    registro_venta.evento:=1+random(eventos_total);
    registro_venta.dni:=10000000+random(40000001);
    registro_venta.entradas:=1+random(10);
  end;
end;
procedure agregar_ordenado_lista_ventas(var lista_ventas: t_lista_ventas; registro_venta:
t_registro_venta);
var
  anterior, actual, nuevo: t_lista_ventas;
begin
  new(nuevo);
  nuevo^.ele:=registro_venta;
  nuevo^.sig:=nil;
  actual:=lista_ventas;
  while ((actual<>nil) and (actual^.ele.evento<nuevo^.ele.evento)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=lista_ventas) then
    lista_ventas:=nuevo
  else
    anterior^.sig:=nuevo;
  nuevo^.sig:=actual;
end;
procedure inicializar_vector_entradas(var vector_entradas: t_vector_entradas);
var
  i: t_evento;
begin
  for i:= 1 to eventos_total do
    vector_entradas[i]:=0;
end;
```

```
procedure cargar_lista_ventas(var lista_ventas: t_lista_ventas; vector_eventos:
t_vector_eventos);
var
  registro_venta: t_registro_venta;
  vector_entradas: t_vector_entradas;
begin
  inicializar_vector_entradas(vector_entradas);
  leer_venta(registro_venta);
  while (registro_venta.codigo<>codigo_salida) do
  begin
    if
(vector_entradas[registro_venta.evento]+registro_venta.entradas<=vector_eventos[registro_venta
.evento].personas) then
    begin
      vector_entradas[registro_venta.evento]:=vector_entradas[registro_venta.evento]+registro_
venta.entradas;
      agregar_ordenado_lista_ventas(lista_ventas,registro_venta);
    end
    else if
(vector_entradas[registro_venta.evento]<vector_eventos[registro_venta.evento].personas) then
    begin
      registro_venta.entradas:=vector_eventos[registro_venta.evento].personas-
vector_entradas[registro_venta.evento];
      vector_entradas[registro_venta.evento]:=vector_entradas[registro_venta.evento]+registro_
venta.entradas;
      agregar_ordenado_lista_ventas(lista_ventas,registro_venta);
    end;
    leer_venta(registro_venta);
  end;
end;
procedure imprimir_registro_venta(registro_venta: t_registro_venta; venta: int16);
begin
  textcolor(green); write('El código de venta de la venta '); textcolor(yellow); write(venta);
textcolor(green); write(' es '); textcolor(red); writeln(registro_venta.codigo);
  textcolor(green); write('El número de evento de la venta '); textcolor(yellow);
write(venta); textcolor(green); write(' es '); textcolor(red); writeln(registro_venta.evento);
  textcolor(green); write('El DNI del comprador de la venta '); textcolor(yellow);
write(venta); textcolor(green); write(' es '); textcolor(red); writeln(registro_venta.dni);
  textcolor(green); write('La cantidad de entradas adquiridas de la venta ');
textcolor(yellow); write(venta); textcolor(green); write(' es '); textcolor(red);
writeln(registro_venta.entradas);
end;
procedure imprimir_lista_ventas(lista_ventas: t_lista_ventas);
var
  i: int16;
begin
  i:=0;
  while (lista_ventas<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información de la venta '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_venta(lista_ventas^.ele,i);
    writeln();
    lista_ventas:=lista_ventas^.sig;
  end;
end;
function contar_pares_impares(dni: int32): boolean;
var
  pares, impares: int8;
begin
  pares:=0; impares:=0;
  while (dni<>0) do
  begin
    if (dni mod 2=0) then
      pares:=pares+1
```

```
    else
      impares:=impares+1;
    dni:=dni div 10;
  end;
  contar_pares_impares:=(pares>impares);
end;
procedure actualizar_minimos(recaudacion_evento: real; nombre, lugar: string; var
recaudacion_min1, recaudacion_min2: real; var nombre_min1, nombre_min2, lugar_min1,
lugar_min2: string);
begin
  if (recaudacion_evento<recaudacion_min1) then
  begin
    recaudacion_min2:=recaudacion_min1;
    nombre_min2:=nombre_min1;
    lugar_min2:=lugar_min1;
    recaudacion_min1:=recaudacion_evento;
    nombre_min1:=nombre;
    lugar_min1:=lugar;
  end
  else
    if (recaudacion_evento<recaudacion_min2) then
    begin
      recaudacion_min2:=recaudacion_evento;
      nombre_min2:=nombre;
      lugar_min2:=lugar;
    end;
end;
procedure procesar_lista_ventas(lista_ventas: t_lista_ventas; vector_eventos:
t_vector_eventos; var nombre_min1, nombre_min2, lugar_min1, lugar_min2: string; var
entradas_corte1: int16; var cumple_evento_corte: boolean);
var
  evento: t_evento;
  entradas_corte2: int8;
  recaudacion_evento, recaudacion_min1, recaudacion_min2: real;
begin
  recaudacion_min1:=9999999; recaudacion_min2:=9999999;
  entradas_corte2:=0;
  while (lista_ventas<>nil) do
  begin
    evento:=lista_ventas^.ele.evento;
    recaudacion_evento:=0;
    while ((lista_ventas<>nil) and (lista_ventas^.ele.evento=evento)) do
    begin
      recaudacion_evento:=recaudacion_evento+vector_eventos[evento].costo*lista_ventas^.ele.en
tradas;
      if ((contar_pares_impares(lista_ventas^.ele.dni)=true) and
(vector_eventos[evento].tipo=tipo_corte)) then
        entradas_corte1:=entradas_corte1+lista_ventas^.ele.entradas;
      if (evento=evento_corte) then
        entradas_corte2:=entradas_corte2+lista_ventas^.ele.entradas;
      lista_ventas:=lista_ventas^.sig;
    end;
    actualizar_minimos(recaudacion_evento,vector_eventos[evento].nombre,vector_eventos[evento]
.lugar,recaudacion_min1,recaudacion_min2,nombre_min1,nombre_min2,lugar_min1,lugar_min2);
  end;
  cumple_evento_corte:=(entradas_corte2=vector_eventos[evento_corte].personas);
end;
var
  vector_eventos: t_vector_eventos;
  lista_ventas: t_lista_ventas;
  entradas_corte1: int16;
  cumple_evento_corte: boolean;
  nombre_min1, nombre_min2, lugar_min1, lugar_min2: string;
begin
  randomize;
  lista_ventas:=nil;
```

```
  nombre_min1:=''; nombre_min2:=''; lugar_min1:=''; lugar_min2:='';
  entradas_corte1:=0;
  cumple_evento_corte:=false;
  cargar_vector_eventos(vector_eventos);
  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
  cargar_lista_ventas(lista_ventas,vector_eventos);
  if (lista_ventas<>nil) then
  begin
    imprimir_lista_ventas(lista_ventas);
    writeln(); textcolor(red); writeln('INCISO (b) (i):'); writeln();
    procesar_lista_ventas(lista_ventas,vector_eventos,nombre_min1,nombre_min2,lugar_min1,lugar
_min2,entradas_corte1,cumple_evento_corte);
    textcolor(green); write('El nombre y el lugar del evento que ha tenido menos recaudación
son '); textcolor(red); write(nombre_min1); textcolor(green); write(' y '); textcolor(red);
write(lugar_min1); textcolor(green); writeln(', respectivamente');
    textcolor(green); write('El nombre y el lugar del segundo evento que ha tenido menos
recaudación son '); textcolor(red); write(nombre_min2); textcolor(green); write(' y ');
textcolor(red); write(lugar_min2); textcolor(green); writeln(', respectivamente');
    writeln(); textcolor(red); writeln('INCISO (b) (ii):'); writeln();
    textcolor(green); write('La cantidad de entradas vendidas cuyo comprador contiene, en su
DNI, más dígitos pares que impares y que son para el evento de tipo '); textcolor(yellow);
write(vector_tipos[tipo_corte]); textcolor(green); write(' es '); textcolor(red);
writeln(entradas_corte1);
    writeln(); textcolor(red); writeln('INCISO (b) (iii):'); writeln();
    textcolor(green); write('La cantidad de entradas vendidas para el evento número ');
textcolor(yellow); write(evento_corte); textcolor(red); if (cumple_evento_corte=true) then
write(' SÍ') else write(' NO'); textcolor(green); write(' alcanzó la cantidad máxima de
personas permitidas');
  end;
end.
```

Ejercicio 12.

El centro de deportes Fortaco's quiere procesar la información de los 4 tipos de suscripciones que ofrece: 1) Musculación, 2) Spinning, 3) CrossFit, 4) Todas las clases. Para ello, el centro dispone de una tabla con información sobre el costo mensual de cada tipo de suscripción.

(a) *Realizar un programa que lea y almacene la información de los clientes del centro. De cada cliente, se conoce el nombre, DNI, edad y tipo de suscripción contratada (valor entre 1 y 4). Cada cliente tiene una sola suscripción. La lectura finaliza cuando se lee el cliente con DNI 0, el cual no debe procesarse.*

(b) *Una vez almacenados todos los datos, procesar la estructura de datos generada, calcular e informar:*

(i) *La ganancia total de Fortaco's.*
(ii) *Las 2 suscripciones con más clientes.*

(c) *Generar una lista con nombre y DNI de los clientes de más de 40 años que están suscritos a CrossFit o a Todas las clases. Esta lista debe estar ordenada por DNI.*

```
program TP7_E12;
{$codepage UTF8}
uses crt;
const
  suscripcion_ini=1; suscripcion_fin=4;
  dni_salida=0;
  edad_corte=40; suscripcion_corte1=3; suscripcion_corte2=4;
type
  t_suscripcion=suscripcion_ini..suscripcion_fin;
  t_registro_cliente1=record
    nombre: string;
    dni: int32;
    edad: int8;
    suscripcion: t_suscripcion;
  end;
  t_vector_costos=array[t_suscripcion] of real;
  t_lista_clientes1=^t_nodo_clientes1;
  t_nodo_clientes1=record
    ele: t_registro_cliente1;
    sig: t_lista_clientes1;
  end;
  t_vector_cantidades=array[t_suscripcion] of int16;
  t_registro_cliente2=record
    nombre: string;
    dni: int32;
  end;
  t_lista_clientes2=^t_nodo_clientes2;
  t_nodo_clientes2=record
    ele: t_registro_cliente2;
    sig: t_lista_clientes2;
  end;
procedure cargar_vector_costos(var vector_costos: t_vector_costos);
var
  i: t_suscripcion;
begin
  for i:= suscripcion_ini to suscripcion_fin do
    vector_costos[i]:=1+random(991)/10;
```

```
end;
function random_string(length: int8): string;
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_cliente(var registro_cliente1: t_registro_cliente1);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_cliente1.dni:=dni_salida
  else
    registro_cliente1.dni:=10000000+random(40000001);
  if (registro_cliente1.dni<>dni_salida) then
  begin
    registro_cliente1.nombre:=random_string(5+random(6));
    registro_cliente1.edad:=18+random(83);
    registro_cliente1.suscripcion:=suscripcion_ini+random(suscripcion_fin);
  end;
end;
procedure agregar_adelante_lista_clientes1(var lista_clientes1: t_lista_clientes1;
registro_cliente1: t_registro_cliente1);
var
  nuevo: t_lista_clientes1;
begin
  new(nuevo);
  nuevo^.ele:=registro_cliente1;
  nuevo^.sig:=lista_clientes1;
  lista_clientes1:=nuevo;
end;
procedure cargar_lista_clientes1(var lista_clientes1: t_lista_clientes1);
var
  registro_cliente1: t_registro_cliente1;
begin
  leer_cliente(registro_cliente1);
  while (registro_cliente1.dni<>dni_salida) do
  begin
    agregar_adelante_lista_clientes1(lista_clientes1,registro_cliente1);
    leer_cliente(registro_cliente1);
  end;
end;
procedure imprimir_registro_cliente1(registro_cliente1: t_registro_cliente1; cliente: int16);
begin
  textcolor(green); write('El nombre del cliente '); textcolor(yellow); write(cliente);
textcolor(green); write(' es '); textcolor(red); writeln(registro_cliente1.nombre);
  textcolor(green); write('El DNI del cliente '); textcolor(yellow); write(cliente);
textcolor(green); write(' es '); textcolor(red); writeln(registro_cliente1.dni);
  textcolor(green); write('La edad del cliente '); textcolor(yellow); write(cliente);
textcolor(green); write(' es '); textcolor(red); writeln(registro_cliente1.edad);
  textcolor(green); write('El tipo de suscripción contratada por el cliente ');
textcolor(yellow); write(cliente); textcolor(green); write(' es '); textcolor(red);
writeln(registro_cliente1.suscripcion);
end;
procedure imprimir_lista_clientes1(lista_clientes1: t_lista_clientes1);
var
  i: int16;
begin
  i:=0;
  while (lista_clientes1<>nil) do
```

```
  begin
    i:=i+1;
    textcolor(green); write('La información del cliente '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_cliente1(lista_clientes1^.ele,i);
    writeln();
    lista_clientes1:=lista_clientes1^.sig;
  end;
end;
procedure inicializar_vector_cantidades(var vector_cantidades: t_vector_cantidades);
var
  i: t_suscripcion;
begin
  for i:= suscripcion_ini to suscripcion_fin do
    vector_cantidades[i]:=0;
end;
procedure cargar_registro_cliente2(var registro_cliente2: t_registro_cliente2;
registro_cliente1: t_registro_cliente1);
begin
  registro_cliente2.nombre:=registro_cliente1.nombre;
  registro_cliente2.dni:=registro_cliente1.dni;
end;
procedure agregar_ordenado_lista_clientes2(var lista_clientes2: t_lista_clientes2;
registro_cliente1: t_registro_cliente1);
var
  anterior, actual, nuevo: t_lista_clientes2;
begin
  new(nuevo);
  cargar_registro_cliente2(nuevo^.ele,registro_cliente1);
  nuevo^.sig:=nil;
  actual:=lista_clientes2;
  while ((actual<>nil) and (actual^.ele.dni<nuevo^.ele.dni)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=lista_clientes2) then
    lista_clientes2:=nuevo
  else
    anterior^.sig:=nuevo;
  nuevo^.sig:=actual;
end;
procedure actualizar_maximos(cantidad: int16; suscripcion: t_suscripcion; var cantidad_max1,
cantidad_max2: int16; var suscripcion_max1, suscripcion_max2: int8);
begin
  if (cantidad>cantidad_max1) then
  begin
    cantidad_max2:=cantidad_max1;
    suscripcion_max2:=suscripcion_max1;
    cantidad_max1:=cantidad;
    suscripcion_max1:=suscripcion;
  end
  else
    if (cantidad>cantidad_max2) then
    begin
      cantidad_max2:=cantidad;
      suscripcion_max2:=suscripcion;
    end;
end;
procedure procesar_vector_cantidades(vector_cantidades: t_vector_cantidades; var
suscripcion_max1, suscripcion_max2: int8);
var
  i: t_suscripcion;
  cantidad_max1, cantidad_max2: int16;
begin
  cantidad_max1:=low(int16); cantidad_max2:=low(int16);
```

```
  for i:= suscripcion_ini to suscripcion_fin do
  begin
    actualizar_maximos(vector_cantidades[i],i,cantidad_max1,cantidad_max2,suscripcion_max1,sus
cripcion_max2);
    textcolor(green); write('La cantidad de clientes con tipo de suscripción contratada ',i,'
es '); textcolor(red); writeln(vector_cantidades[i]);
  end;
end;
procedure procesar_lista_clientes1(lista_clientes1: t_lista_clientes1; vector_costos:
t_vector_costos; var ganancia: real; var suscripcion_max1, suscripcion_max2: int8; var
lista_clientes2: t_lista_clientes2);
var
  vector_cantidades: t_vector_cantidades;
begin
  inicializar_vector_cantidades(vector_cantidades);
  while (lista_clientes1<>nil) do
  begin
    ganancia:=ganancia+vector_costos[lista_clientes1^.ele.suscripcion];
    vector_cantidades[lista_clientes1^.ele.suscripcion]:=vector_cantidades[lista_clientes1^.el
e.suscripcion]+1;
    if ((lista_clientes1^.ele.edad>edad_corte) and
((lista_clientes1^.ele.suscripcion=suscripcion_corte1) or
(lista_clientes1^.ele.suscripcion=suscripcion_corte2))) then
      agregar_ordenado_lista_clientes2(lista_clientes2,lista_clientes1^.ele);
    lista_clientes1:=lista_clientes1^.sig;
  end;
  procesar_vector_cantidades(vector_cantidades,suscripcion_max1,suscripcion_max2);
end;
procedure imprimir_registro_cliente2(registro_cliente2: t_registro_cliente2; cliente: int16);
begin
  textcolor(green); write('El nombre del cliente '); textcolor(yellow); write(cliente);
textcolor(green); write(' es '); textcolor(red); writeln(registro_cliente2.nombre);
  textcolor(green); write('El DNI del cliente '); textcolor(yellow); write(cliente);
textcolor(green); write(' es '); textcolor(red); writeln(registro_cliente2.dni);
end;
procedure imprimir_lista_clientes2(lista_clientes2: t_lista_clientes2);
var
  i: int16;
begin
  i:=0;
  while (lista_clientes2<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información del cliente '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_cliente2(lista_clientes2^.ele,i);
    writeln();
    lista_clientes2:=lista_clientes2^.sig;
  end;
end;
var
  vector_costos: t_vector_costos;
  lista_clientes1: t_lista_clientes1;
  lista_clientes2: t_lista_clientes2;
  suscripcion_max1, suscripcion_max2: int8;
  ganancia: real;
begin
  randomize;
  lista_clientes1:=nil;
  ganancia:=0;
  suscripcion_max1:=0; suscripcion_max2:=0;
  lista_clientes2:=nil;
  cargar_vector_costos(vector_costos);
  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
  cargar_lista_clientes1(lista_clientes1);
  if (lista_clientes1<>nil) then
```

```
  begin
    imprimir_lista_clientes1(lista_clientes1);
    writeln(); textcolor(red); writeln('INCISO (b) (i):'); writeln();
    procesar_lista_clientes1(lista_clientes1,vector_costos,ganancia,suscripcion_max1,suscripci
on_max2,lista_clientes2);
    writeln(); textcolor(green); write('La ganancia total de Fortaco's es $'); textcolor(red);
writeln(ganancia:0:2);
    writeln(); textcolor(red); writeln('INCISO (b) (ii):'); writeln();
    textcolor(green); write('Las 2 suscripciones con más clientes son '); textcolor(red);
write(suscripcion_max1); textcolor(green); write(' y '); textcolor(red);
writeln(suscripcion_max2);
    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();
    imprimir_lista_clientes2(lista_clientes2);
  end;
end.
```

Ejercicio 13.

La tienda de libros Amazon Books está analizando información de algunas editoriales. Para ello, Amazon cuenta con una tabla con las 35 áreas temáticas utilizadas para clasificar los libros (Arte y Cultura, Historia, Literatura, etc.). De cada libro, se conoce su título, nombre de la editorial, cantidad de páginas, año de edición, cantidad de veces que fue vendido y código del área temática (1..35). Realizar un programa que:

(a) *Invoque a un módulo que lea la información de los libros hasta ingresar el título "Relato de un náufrago" (que debe procesarse) y devuelva, en una estructura de datos adecuada para la editorial "Planeta Libros", la siguiente información:*

- *Nombre de la editorial.*
- *Año de edición del libro más antiguo.*
- *Cantidad de libros editados.*
- *Cantidad total de ventas entre todos los libros.*
- *Detalle con título, nombre del área temática y cantidad de páginas de todos los libros con más de 250 ventas.*

(b) *Invoque a un módulo que reciba la estructura generada en el inciso (a) e imprima el nombre de la editorial y el título de cada libro con más de 250 ventas.*

```
program TP7_E13;
{$codepage UTF8}
uses crt;
const
  anio_ini=2000; anio_fin=2020;
  area_ini=1; area_fin=35;
  titulo_salida='Relato de un naufrago';
  editorial_corte='Planeta Libros';
  ventas_corte=250;
  vector_areas: array[area_ini..area_fin] of string=('Arte y Cultura', 'Historia',
'Literatura', 'Ciencia Ficcion', 'Policial', 'Romantica', 'Aventura', 'Infantil', 'Juvenil',
'Terror', 'Fantasia', 'Biografia', 'Autoayuda', 'Cocina', 'Viajes', 'Deportes', 'Salud',
'Economia', 'Politica', 'Sociedad', 'Filosofia', 'Religiin', 'Ciencia', 'Tecnologia',
'Matematicas', 'Fisica', 'Quimica', 'Biologia', 'Geografia', 'Ecologia', 'Astronomia',
'Medicina', 'Derecho', 'Arquitectura', 'Musica');
type
  t_area=area_ini..area_fin;
  t_registro_libro1=record
    titulo: string;
    editorial: string;
    paginas: int16;
    anio: int16;
    ventas: int16;
    area: t_area;
  end;
  t_registro_libro2=record
    titulo: string;
    area: string;
    paginas: int16;
  end;
  t_lista_libros=^t_nodo_libros;
  t_nodo_libros=record
    ele: t_registro_libro2;
    sig: t_lista_libros;
  end;
```

```
  t_registro_editorial=record
    editorial: string;
    anio: int16;
    libros: int16;
    ventas: int16;
    lista_libros: t_lista_libros;
  end;
procedure inicializar_registro_editorial(var registro_editorial: t_registro_editorial);
begin
  registro_editorial.editorial:=editorial_corte;
  registro_editorial.anio:=high(int16);
  registro_editorial.libros:=0;
  registro_editorial.ventas:=0;
  registro_editorial.lista_libros:=nil;
end;
function random_string(length: int8): string;
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_libro(var registro_libro1: t_registro_libro1);
var
  i: int8;
begin
  i:=random(101);
  if (i=0) then
    registro_libro1.titulo:=titulo_salida
  else
    registro_libro1.titulo:=random_string(5+random(6));
  if (i<=50) then
    registro_libro1.editorial:=editorial_corte
  else
    registro_libro1.editorial:=random_string(5+random(6));
  registro_libro1.paginas:=10+random(491);
  registro_libro1.anio:=anio_ini+random(anio_fin-anio_ini+1);
  registro_libro1.ventas:=1+random(1000);
  registro_libro1.area:=area_ini+random(area_fin);
end;
procedure cargar_registro_libro2(var registro_libro2: t_registro_libro2; registro_libro1:
t_registro_libro1);
begin
  registro_libro2.titulo:=registro_libro1.titulo;
  registro_libro2.area:=vector_areas[registro_libro1.area];
  registro_libro2.paginas:=registro_libro1.paginas;
end;
procedure agregar_adelante_lista_libros(var lista_libros: t_lista_libros; registro_libro1:
t_registro_libro1);
var
  nuevo: t_lista_libros;
begin
  new(nuevo);
  cargar_registro_libro2(nuevo^.ele,registro_libro1);
  nuevo^.sig:=lista_libros;
  lista_libros:=nuevo;
end;
procedure cargar_registro_editorial(var registro_editorial: t_registro_editorial);
var
  registro_libro1: t_registro_libro1;
begin
  repeat
    leer_libro(registro_libro1);
```

```
    if (registro_libro1.editorial=editorial_corte) then
    begin
      if (registro_libro1.anio<registro_editorial.anio) then
        registro_editorial.anio:=registro_libro1.anio;
      registro_editorial.libros:=registro_editorial.libros+1;
      registro_editorial.ventas:=registro_editorial.ventas+registro_libro1.ventas;
      if (registro_libro1.ventas>ventas_corte) then
        agregar_adelante_lista_libros(registro_editorial.lista_libros,registro_libro1);
    end;
  until (registro_libro1.titulo=titulo_salida);
end;
procedure imprimir_registro_libro2(registro_libro2: t_registro_libro2; libro: int16);
begin
  textcolor(green); write('El título del libro '); textcolor(yellow); write(libro);
textcolor(green); write(' es '); textcolor(red); writeln(registro_libro2.titulo);
  textcolor(green); write('El nombre del área temática del libro '); textcolor(yellow);
write(libro); textcolor(green); write(' es '); textcolor(red); writeln(registro_libro2.area);
  textcolor(green); write('La cantidad de páginas del libro '); textcolor(yellow);
write(libro); textcolor(green); write(' es '); textcolor(red);
writeln(registro_libro2.paginas);
end;
procedure imprimir_lista_libros(lista_libros: t_lista_libros);
var
  i: int16;
begin
  i:=0;
  while (lista_libros<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información del libro '); textcolor(yellow); write(i);
textcolor(green); write(' con más de '); textcolor(yellow); write(ventas_corte);
textcolor(green); writeln(' ventas es:');
    imprimir_registro_libro2(lista_libros^.ele,i);
    writeln();
    lista_libros:=lista_libros^.sig;
  end;
end;
procedure imprimir_registro_editorial(registro_editorial: t_registro_editorial);
begin
  textcolor(green); write('El nombre de la editorial es '); textcolor(red);
writeln(registro_editorial.editorial);
  textcolor(green); write('El año de edición del libro más antiguo es '); textcolor(red);
writeln(registro_editorial.anio);
  textcolor(green); write('La cantidad de libros editados es '); textcolor(red);
writeln(registro_editorial.libros);
  textcolor(green); write('La cantidad total de ventas entre todos los libros es ');
textcolor(red); writeln(registro_editorial.ventas);
  writeln();
  imprimir_lista_libros(registro_editorial.lista_libros);
end;
var
  registro_editorial: t_registro_editorial;
begin
  randomize;
  inicializar_registro_editorial(registro_editorial);
  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
  cargar_registro_editorial(registro_editorial);
  writeln(); textcolor(red); writeln('INCISO (b):'); writeln();
  imprimir_registro_editorial(registro_editorial);
end.
```

Ejercicio 14.

La biblioteca de la Universidad Nacional de La Plata necesita un programa para administrar información de préstamos de libros efectuados en marzo de 2020. Para ello, se debe leer la información de los préstamos realizados. De cada préstamo, se lee: nro. de préstamo, ISBN del libro prestado, nro. de socio al que se prestó el libro, día del préstamo (1..31). La información de los préstamos se lee de manera ordenada por ISBN y finaliza cuando se ingresa el ISBN -1 (que no debe procesarse). Se pide:

(a) *Generar una estructura que contenga, para cada ISBN de libro, la cantidad de veces que fue prestado. Esta estructura debe quedar ordenada por ISBN de libro.*

(b) *Calcular e informar el día del mes en que se realizaron menos préstamos.*

(c) *Calcular e informar el porcentaje de préstamos que poseen nro. de préstamo impar y nro. de socio par.*

```
program TP7_E14;
{$codepage UTF8}
uses crt;
const
  dia_ini=1; dia_fin=31;
  isbn_salida=-1;
type
  t_dia=dia_ini..dia_fin;
  t_registro_prestamo=record
    numero: int16;
    isbn: int32;
    socio: int16;
    dia: t_dia;
  end;
  t_registro_isbn=record
    isbn: int32;
    prestamos: int16;
  end;
  t_lista_isbns=^t_nodo_isbns;
  t_nodo_isbns=record
    ele: t_registro_isbn;
    sig: t_lista_isbns;
  end;
  t_vector_cantidades=array[t_dia] of int16;
procedure inicializar_vector_cantidades(var vector_cantidades: t_vector_cantidades);
var
  i: t_dia;
begin
  for i:= dia_ini to dia_fin do
    vector_cantidades[i]:=0;
end;
procedure leer_prestamo(var registro_prestamo: t_registro_prestamo; isbn: int32);
var
  i: int8;
begin
  i:=random(101);
  if (i=0) then
    registro_prestamo.isbn:=isbn_salida
  else if (i<=50) then
    registro_prestamo.isbn:=isbn
  else
    registro_prestamo.isbn:=isbn+random(high(int32)-(isbn-1));
```

```
  if (registro_prestamo.isbn<>isbn_salida) then
  begin
    registro_prestamo.numero:=1+random(high(int16));
    registro_prestamo.socio:=1+random(high(int16));
    registro_prestamo.dia:=dia_ini+random(dia_fin);
  end;
end;
procedure agregar_atras_lista_isbns(var lista_isbns: t_lista_isbns; registro_isbn:
t_registro_isbn);
var
  nuevo, ult: t_lista_isbns;
begin
  new(nuevo);
  nuevo^.ele:=registro_isbn;
  nuevo^.sig:=nil;
  if (lista_isbns=nil) then
    lista_isbns:=nuevo
  else
  begin
    ult:=lista_isbns;
    while (ult^.sig<>nil) do
      ult:=ult^.sig;
    ult^.sig:=nuevo;
  end;
end;
procedure cargar_lista_isbns(var lista_isbns: t_lista_isbns; var vector_cantidades:
t_vector_cantidades; var porcentaje: real);
var
  registro_prestamo: t_registro_prestamo;
  registro_isbn: t_registro_isbn;
  prestamos_corte, prestamos_total: int16;
begin
  prestamos_corte:=0; prestamos_total:=0;
  leer_prestamo(registro_prestamo,1+random(high(int32)));
  while (registro_prestamo.isbn<>isbn_salida) do
  begin
    registro_isbn.isbn:=registro_prestamo.isbn;
    registro_isbn.prestamos:=0;
    while ((registro_prestamo.isbn<>isbn_salida) and
(registro_prestamo.isbn=registro_isbn.isbn)) do
    begin
      registro_isbn.prestamos:=registro_isbn.prestamos+1;
      vector_cantidades[registro_prestamo.dia]:=vector_cantidades[registro_prestamo.dia]+1;
      if ((registro_prestamo.numero mod 2<>0) and (registro_prestamo.socio mod 2=0)) then
        prestamos_corte:=prestamos_corte+1;
      prestamos_total:=prestamos_total+1;
      leer_prestamo(registro_prestamo,registro_isbn.isbn);
    end;
    agregar_atras_lista_isbns(lista_isbns,registro_isbn);
  end;
  if (prestamos_total>0) then
    porcentaje:=prestamos_corte/prestamos_total*100;
end;
procedure imprimir_registro_isbn(registro_isbn: t_registro_isbn; isbn: int32);
begin
  textcolor(green); write('El ISBN del ISBN '); textcolor(yellow); write(isbn);
textcolor(green); write(' es '); textcolor(red); writeln(registro_isbn.isbn);
  textcolor(green); write('La cantidad de préstamos del ISBN '); textcolor(yellow);
write(isbn); textcolor(green); write(' es '); textcolor(red);
writeln(registro_isbn.prestamos);
end;
procedure imprimir_lista_isbns(lista_isbns: t_lista_isbns);
var
  i: int8;
begin
  i:=0;
```

```
  while (lista_isbns<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información del ISBN '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_isbn(lista_isbns^.ele,i);
    writeln();
    lista_isbns:=lista_isbns^.sig;
  end;
end;
procedure actualizar_minimo(cantidad: int16; dia: t_dia; var cantidad_min: int16; var dia_min:
int8);
begin
  if (cantidad<cantidad_min) then
  begin
    cantidad_min:=cantidad;
    dia_min:=dia;
  end;
end;
procedure procesar_vector_cantidades(var vector_cantidades: t_vector_cantidades; var dia_min:
int8);
var
  i: t_dia;
  cantidad_min: int16;
begin
  cantidad_min:=high(int16);
  for i:= dia_ini to dia_fin do
  begin
    actualizar_minimo(vector_cantidades[i],i,cantidad_min,dia_min);
    textcolor(green); write('La cantidad de préstamos que se realizaron el día ',i,' es ');
textcolor(red); writeln(vector_cantidades[i]);
  end;
end;
var
  vector_cantidades: t_vector_cantidades;
  lista_isbns: t_lista_isbns;
  dia_min: int8;
  porcentaje: real;
begin
  randomize;
  inicializar_vector_cantidades(vector_cantidades);
  lista_isbns:=nil;
  dia_min:=0;
  porcentaje:=0;
  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
  cargar_lista_isbns(lista_isbns,vector_cantidades,porcentaje);
  if (lista_isbns<>nil) then
  begin
    imprimir_lista_isbns(lista_isbns);
    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();
    procesar_vector_cantidades(vector_cantidades,dia_min);
    writeln(); textcolor(green); write('El día del mes en que se realizaron menos préstamos
fue el día '); textcolor(red); writeln(dia_min);
    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();
    textcolor(green); write('El porcentaje de préstamos que poseen nro. de préstamo impar y
nro. de socio par es '); textcolor(red); write(porcentaje:0:2); textcolor(green); write('%');
  end;
end.
```