Trabajo Práctico N° 5:
Módulo Imperativo (Adicionales).

Ejercicio 1.

El administrador de un edificio de oficinas cuenta, en papel, con la información del pago de las expensas de dichas oficinas. Implementar un programa con:

(a) *Un módulo que retorne un vector, sin orden, con, a lo sumo, las 300 oficinas que administra. Se debe leer, para cada oficina, el código de identificación, DNI del propietario y valor de la expensa. La lectura finaliza cuando llega el código de identificación -1.*

(b) *Un módulo que reciba el vector retornado en (a) y retorne dicho vector ordenado por código de identificación de la oficina. Ordenar el vector aplicando uno de los métodos vistos en la cursada.*

(c) *Un módulo que realice una búsqueda dicotómica. Este módulo debe recibir el vector generado en (b) y un código de identificación de oficina. En el caso de encontrarlo, debe retornar la posición del vector donde se encuentra y, en caso contrario, debe retornar 0. Luego, el programa debe informar el DNI del propietario o un cartel indicando que no se encontró la oficina.*

(d) *Un módulo recursivo que retorne el monto total de las expensas.*

```
program TP5_E1;
{$codepage UTF8}
uses crt;
const
  oficinas_total=300;
  codigo_salida=-1;
type
  t_oficina=1..oficinas_total;
  t_registro_oficina=record
    codigo: int16;
    dni: int32;
    expensa: real;
  end;
  t_vector_oficinas=array[t_oficina] of t_registro_oficina;
procedure leer_oficina(var registro_oficina: t_registro_oficina);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_oficina.codigo:=codigo_salida
  else
    registro_oficina.codigo:=random(high(int16));
  if (registro_oficina.codigo<>codigo_salida) then
  begin
    registro_oficina.dni:=10000000+random(40000001);
    registro_oficina.expensa:=1+random(100);
  end;
end;
procedure cargar_vector_oficinas(var vector_oficinas: t_vector_oficinas; var oficinas: int16);
var
```

```
  registro_oficina: t_registro_oficina;
begin
  leer_oficina(registro_oficina);
  while ((registro_oficina.codigo<>codigo_salida) and (oficinas<oficinas_total)) do
  begin
    oficinas:=oficinas+1;
    vector_oficinas[oficinas]:=registro_oficina;
    leer_oficina(registro_oficina);
  end;
end;
procedure imprimir_registro_oficina(registro_oficina: t_registro_oficina; oficina: t_oficina);
begin
  textcolor(green); write('El código de identificación de la oficina '); textcolor(yellow);
write(oficina); textcolor(green); write(' es '); textcolor(red);
writeln(registro_oficina.codigo);
  textcolor(green); write('El DNI del propietario de la oficina '); textcolor(yellow);
write(oficina); textcolor(green); write(' es '); textcolor(red);
writeln(registro_oficina.dni);
  textcolor(green); write('El valor de la expensa de la oficina '); textcolor(yellow);
write(oficina); textcolor(green); write(' es $'); textcolor(red);
writeln(registro_oficina.expensa:0:2);
end;
procedure imprimir_vector_oficinas(vector_oficinas: t_vector_oficinas; oficinas: int16);
var
  i: t_oficina;
begin
  for i:= 1 to oficinas do
  begin
    textcolor(green); write('La información de la oficina '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_oficina(vector_oficinas[i],i);
    writeln();
  end;
end;
procedure ordenar_vector_oficinas(var vector_oficinas: t_vector_oficinas; oficinas: int16);
var
  item: t_registro_oficina;
  i, j, k: t_oficina;
begin
  for i:= 1 to (oficinas-1) do
  begin
    k:=i;
    for j:= (i+1) to oficinas do
      if (vector_oficinas[j].codigo<vector_oficinas[k].codigo) then
        k:=j;
    item:=vector_oficinas[k];
    vector_oficinas[k]:=vector_oficinas[i];
    vector_oficinas[i]:=item;
  end;
end;
function buscar_vector_oficinas(vector_oficinas: t_vector_oficinas; codigo, pri, ult: int16):
int16;
var
  medio: int8;
begin
  if (pri<=ult) then
  begin
    medio:=(pri+ult) div 2;
    if (codigo=vector_oficinas[medio].codigo) then
      buscar_vector_oficinas:=medio
    else if (codigo<vector_oficinas[medio].codigo) then
      buscar_vector_oficinas:=buscar_vector_oficinas(vector_oficinas,codigo,pri,medio-1)
    else
      buscar_vector_oficinas:=buscar_vector_oficinas(vector_oficinas,codigo,medio+1,ult)
  end
  else
```

```
    buscar_vector_oficinas:=0;
end;
function sumar_vector_oficinas(vector_oficinas: t_vector_oficinas; oficinas: int16): real;
begin
  if (oficinas=1) then
    sumar_vector_oficinas:=vector_oficinas[oficinas].expensa
  else
    sumar_vector_oficinas:=sumar_vector_oficinas(vector_oficinas,oficinas-
1)+vector_oficinas[oficinas].expensa;
end;
var
  vector_oficinas: t_vector_oficinas;
  oficinas, codigo, pri, ult, pos: int16;
begin
  randomize;
  oficinas:=0;
  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
  cargar_vector_oficinas(vector_oficinas,oficinas);
  if (oficinas>0) then
  begin
    imprimir_vector_oficinas(vector_oficinas,oficinas);
    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();
    ordenar_vector_oficinas(vector_oficinas,oficinas);
    imprimir_vector_oficinas(vector_oficinas,oficinas);
    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();
    codigo:=1+random(high(int16));
    pri:=1; ult:=oficinas;
    pos:=buscar_vector_oficinas(vector_oficinas,codigo,pri,ult);
    if (pos<>0) then
    begin
      textcolor(green); write('El código de identificación de oficina '); textcolor(yellow);
write(codigo); textcolor(green); write(' se encontró en el vector, en la posición ');
textcolor(red); writeln(pos);
      textcolor(green); write('El DNI del propietario de la oficina con código de
identificación '); textcolor(yellow); write(codigo); textcolor(green); write(' es ');
textcolor(red); writeln(vector_oficinas[pos].dni);
    end
    else
    begin
      textcolor(green); write('El código de identificación de oficina '); textcolor(yellow);
write(codigo); textcolor(green); writeln(' no se encontró en el vector');
    end;
    writeln(); textcolor(red); writeln('INCISO (d):'); writeln();
    textcolor(green); write('El monto total de las expensas es $'); textcolor(red);
write(sumar_vector_oficinas(vector_oficinas,oficinas):0:2);
  end;
end.
```

Ejercicio 2.

Una agencia dedicada a la venta de autos ha organizado su stock y dispone, en papel, de la información de los autos en venta. Implementar un programa que:

(a) *Lea la información de los autos (patente, año de fabricación (2010 .. 2018), marca y modelo) y los almacene en dos estructuras de datos:*

> **(i)** *Una estructura eficiente para la búsqueda por patente.*
> **(ii)** *Una estructura eficiente para la búsqueda por marca. Para cada marca, se deben almacenar todos juntos los autos pertenecientes a ella.*

(b) *Invoque a un módulo que reciba la estructura generado en (a) (i) y una marca y retorne la cantidad de autos de dicha marca que posee la agencia.*

(c) *Invoque a un módulo que reciba la estructura generado en (a) (ii) y una marca y retorne la cantidad de autos de dicha marca que posee la agencia.*

(d) *Invoque a un módulo que reciba el árbol generado en (a) (i) y retorne una estructura con la información de los autos agrupados por año de fabricación.*

(e) *Invoque a un módulo que reciba el árbol generado en (a) (i) y una patente y devuelva el modelo del auto con dicha patente.*

(f) *Invoque a un módulo que reciba el árbol generado en (a) (ii) y una patente y devuelva el modelo del auto con dicha patente.*

```
program TP5_E2;
{$codepage UTF8}
uses crt;
const
  anio_ini=2010; anio_fin=2018;
  marca_salida='MMM';
type
  t_anio=anio_ini..anio_fin;
  t_registro_auto1=record
    patente: string;
    anio: t_anio;
    marca: string;
    modelo: string;
  end;
  t_abb_patentes=^t_nodo_abb_patentes;
  t_nodo_abb_patentes=record
    ele: t_registro_auto1;
    hi: t_abb_patentes;
    hd: t_abb_patentes;
  end;
  t_registro_auto2=record
    patente: string;
    anio: t_anio;
    modelo: string;
  end;
  t_lista_autos1=^t_nodo_autos1;
  t_nodo_autos1=record
    ele: t_registro_auto2;
    sig: t_lista_autos1;
```

```
  end;
  t_registro_marca=record
    marca: string;
    autos: t_lista_autos1;
  end;
  t_abb_marcas=^t_nodo_abb_marcas;
  t_nodo_abb_marcas=record
    ele: t_registro_marca;
    hi: t_abb_marcas;
    hd: t_abb_marcas;
  end;
  t_registro_auto3=record
    patente: string;
    marca: string;
    modelo: string;
  end;
  t_lista_autos2=^t_nodo_autos2;
  t_nodo_autos2=record
    ele: t_registro_auto3;
    sig: t_lista_autos2;
  end;
  t_vector_autos=array[t_anio] of t_lista_autos2;
procedure inicializar_vector_autos(var vector_autos: t_vector_autos);
var
  i: t_anio;
begin
  for i:= anio_ini to anio_fin do
    vector_autos[i]:=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_auto(var registro_auto1: t_registro_auto1);
var
  i: int16;
begin
  i:=random(100);
  if (i=0) then
    registro_auto1.marca:=marca_salida
  else
    registro_auto1.marca:='Marca '+random_string(1);
  if (registro_auto1.marca<>marca_salida) then
  begin
    registro_auto1.patente:=random_string(2);
    registro_auto1.anio:=anio_ini+random(anio_fin-anio_ini+1);
    registro_auto1.modelo:='Modelo '+random_string(2);
  end;
end;
procedure agregar_abb_patentes(var abb_patentes: t_abb_patentes; registro_auto1:
t_registro_auto1);
begin
  if (abb_patentes=nil) then
  begin
    new(abb_patentes);
    abb_patentes^.ele:=registro_auto1;
    abb_patentes^.hi:=nil;
    abb_patentes^.hd:=nil;
  end
  else
```

```
    if (registro_auto1.patente<=abb_patentes^.ele.patente) then
      agregar_abb_patentes(abb_patentes^.hi,registro_auto1)
    else
      agregar_abb_patentes(abb_patentes^.hd,registro_auto1);
end;
procedure cargar_registro_auto2(var registro_auto2: t_registro_auto2; registro_auto1:
t_registro_auto1);
begin
  registro_auto2.patente:=registro_auto1.patente;
  registro_auto2.anio:=registro_auto1.anio;
  registro_auto2.modelo:=registro_auto1.modelo;
end;
procedure agregar_adelante_lista_autos1(var lista_autos1: t_lista_autos1; registro_auto1:
t_registro_auto1);
var
  nuevo: t_lista_autos1;
begin
  new(nuevo);
  cargar_registro_auto2(nuevo^.ele,registro_auto1);
  nuevo^.sig:=lista_autos1;
  lista_autos1:=nuevo;
end;
procedure cargar_registro_marca(var registro_marca: t_registro_marca; registro_auto1:
t_registro_auto1);
begin
  registro_marca.marca:=registro_auto1.marca;
  registro_marca.autos:=nil;
  agregar_adelante_lista_autos1(registro_marca.autos,registro_auto1);
end;
procedure agregar_abb_marcas(var abb_marcas: t_abb_marcas; registro_auto1: t_registro_auto1);
begin
  if (abb_marcas=nil) then
  begin
    new(abb_marcas);
    cargar_registro_marca(abb_marcas^.ele,registro_auto1);
    abb_marcas^.hi:=nil;
    abb_marcas^.hd:=nil;
  end
  else
    if (registro_auto1.marca=abb_marcas^.ele.marca) then
      agregar_adelante_lista_autos1(abb_marcas^.ele.autos,registro_auto1)
    else if (registro_auto1.marca<abb_marcas^.ele.marca) then
      agregar_abb_marcas(abb_marcas^.hi,registro_auto1)
    else
      agregar_abb_marcas(abb_marcas^.hd,registro_auto1);
end;
procedure cargar_abbs(var abb_patentes: t_abb_patentes; var abb_marcas: t_abb_marcas);
var
  registro_auto1: t_registro_auto1;
begin
  leer_auto(registro_auto1);
  while (registro_auto1.marca<>marca_salida) do
  begin
    agregar_abb_patentes(abb_patentes,registro_auto1);
    agregar_abb_marcas(abb_marcas,registro_auto1);
    leer_auto(registro_auto1);
  end;
end;
procedure imprimir_registro_auto1(registro_auto1: t_registro_auto1);
begin
  textcolor(green); write('La patente del auto es '); textcolor(red);
writeln(registro_auto1.patente);
  textcolor(green); write('El año de fabricación del auto es '); textcolor(red);
writeln(registro_auto1.anio);
  textcolor(green); write('La marca del auto es '); textcolor(red);
writeln(registro_auto1.marca);
```

```
  textcolor(green); write('El modelo del auto es '); textcolor(red);
writeln(registro_auto1.modelo);
  writeln();
end;
procedure imprimir_abb_patentes(abb_patentes: t_abb_patentes);
begin
  if (abb_patentes<>nil) then
  begin
    imprimir_abb_patentes(abb_patentes^.hi);
    imprimir_registro_auto1(abb_patentes^.ele);
    imprimir_abb_patentes(abb_patentes^.hd);
  end;
end;
procedure imprimir_registro_auto2(registro_auto2: t_registro_auto2; marca: string; auto:
int16);
begin
  textcolor(green); write('La patente del auto '); textcolor(yellow); write(auto);
textcolor(green); write(' de la marca '); textcolor(yellow); write(marca); textcolor(green);
write(' es '); textcolor(red); writeln(registro_auto2.patente);
  textcolor(green); write('El año de fabricación del auto '); textcolor(yellow); write(auto);
textcolor(green); write(' de la marca '); textcolor(yellow); write(marca); textcolor(green);
write(' es '); textcolor(red); writeln(registro_auto2.anio);
  textcolor(green); write('El modelo del auto '); textcolor(yellow); write(auto);
textcolor(green); write(' de la marca '); textcolor(yellow); write(marca); textcolor(green);
write(' es '); textcolor(red); writeln(registro_auto2.modelo);
end;
procedure imprimir_lista_autos1(lista_autos1: t_lista_autos1; marca: string);
var
  i: int16;
begin
  i:=0;
  while (lista_autos1<>nil) do
  begin
    i:=i+1;
    imprimir_registro_auto2(lista_autos1^.ele,marca,i);
    lista_autos1:=lista_autos1^.sig;
  end;
end;
procedure imprimir_registro_marca(registro_marca: t_registro_marca);
begin
  textcolor(green); write('La marca del auto es '); textcolor(red);
writeln(registro_marca.marca);
  imprimir_lista_autos1(registro_marca.autos,registro_marca.marca);
  writeln();
end;
procedure imprimir_abb_marcas(abb_marcas: t_abb_marcas);
begin
  if (abb_marcas<>nil) then
  begin
    imprimir_abb_marcas(abb_marcas^.hi);
    imprimir_registro_marca(abb_marcas^.ele);
    imprimir_abb_marcas(abb_marcas^.hd);
  end;
end;
function contar_abb_patentes(abb_patentes: t_abb_patentes; marca: string): int8;
begin
  if (abb_patentes=nil) then
    contar_abb_patentes:=0
  else
    if (marca=abb_patentes^.ele.marca) then
      contar_abb_patentes:=contar_abb_patentes(abb_patentes^.hi,marca)+contar_abb_patentes(abb
_patentes^.hd,marca)+1
    else
      contar_abb_patentes:=contar_abb_patentes(abb_patentes^.hi,marca)+contar_abb_patentes(abb
_patentes^.hd,marca);
end;
```

```
function contar_autos(lista_autos1: t_lista_autos1): int8;
var
  autos: int8;
begin
  autos:=0;
  while (lista_autos1<>nil) do
  begin
    autos:=autos+1;
    lista_autos1:=lista_autos1^.sig;
  end;
  contar_autos:=autos;
end;
function contar_abb_marcas(abb_marcas: t_abb_marcas; marca: string): int8;
begin
  if (abb_marcas=nil) then
    contar_abb_marcas:=0
  else
    if (marca=abb_marcas^.ele.marca) then
      contar_abb_marcas:=contar_autos(abb_marcas^.ele.autos)
    else if (marca<abb_marcas^.ele.marca) then
      contar_abb_marcas:=contar_abb_marcas(abb_marcas^.hi,marca)
    else
      contar_abb_marcas:=contar_abb_marcas(abb_marcas^.hd,marca)
end;
procedure cargar_registro_auto3(var registro_auto3: t_registro_auto3; registro_auto1:
t_registro_auto1);
begin
  registro_auto3.patente:=registro_auto1.patente;
  registro_auto3.marca:=registro_auto1.marca;
  registro_auto3.modelo:=registro_auto1.modelo;
end;
procedure agregar_adelante_lista_autos2(var lista_autos2: t_lista_autos2; registro_auto1:
t_registro_auto1);
var
  nuevo: t_lista_autos2;
begin
  new(nuevo);
  cargar_registro_auto3(nuevo^.ele,registro_auto1);
  nuevo^.sig:=lista_autos2;
  lista_autos2:=nuevo;
end;
procedure cargar_vector_autos(var vector_autos: t_vector_autos; abb_patentes: t_abb_patentes);
begin
  if (abb_patentes<>nil) then
  begin
    cargar_vector_autos(vector_autos,abb_patentes^.hi);
    agregar_adelante_lista_autos2(vector_autos[abb_patentes^.ele.anio],abb_patentes^.ele);
    cargar_vector_autos(vector_autos,abb_patentes^.hd);
  end
end;
procedure imprimir_registro_auto3(registro_auto3: t_registro_auto3; anio: t_anio; auto:
int16);
begin
  textcolor(green); write('La patente del auto '); textcolor(yellow); write(auto);
textcolor(green); write(' del año de fabricación '); textcolor(yellow); write(anio);
textcolor(green); write(' es '); textcolor(red); writeln(registro_auto3.patente);
  textcolor(green); write('La marca el auto '); textcolor(yellow); write(auto);
textcolor(green); write(' del año de fabricación '); textcolor(yellow); write(anio);
textcolor(green); write(' es '); textcolor(red); writeln(registro_auto3.marca);
  textcolor(green); write('El modelo del auto '); textcolor(yellow); write(auto);
textcolor(green); write(' del año de fabricación '); textcolor(yellow); write(anio);
textcolor(green); write(' es '); textcolor(red); writeln(registro_auto3.modelo);
end;
procedure imprimir_lista_autos2(lista_autos2: t_lista_autos2; anio: t_anio);
var
  i: int16;
```

```
begin
  i:=0;
  while (lista_autos2<>nil) do
  begin
    i:=i+1;
    imprimir_registro_auto3(lista_autos2^.ele,anio,i);
    lista_autos2:=lista_autos2^.sig;
  end;
end;
procedure imprimir_vector_autos(vector_autos: t_vector_autos);
var
  i: t_anio;
begin
  for i:= anio_ini to anio_fin do
  begin
    textcolor(green); write('La información de los autos del año de fabricación ');
textcolor(yellow); write(i); textcolor(green); writeln(' es:');
    imprimir_lista_autos2(vector_autos[i],i);
    writeln();
  end;
end;
function buscar_abb_patentes(abb_patentes: t_abb_patentes; patente: string): string;
begin
  if (abb_patentes=nil) then
    buscar_abb_patentes:='No existe la patente'
  else
    if (patente=abb_patentes^.ele.patente) then
      buscar_abb_patentes:=abb_patentes^.ele.modelo
    else if (patente<abb_patentes^.ele.patente) then
      buscar_abb_patentes:=buscar_abb_patentes(abb_patentes^.hi,patente)
    else
      buscar_abb_patentes:=buscar_abb_patentes(abb_patentes^.hd,patente);
end;
function buscar_patente(lista_autos1: t_lista_autos1; patente: string): string;
begin
  while ((lista_autos1<>nil) and (lista_autos1^.ele.patente<>patente)) do
    lista_autos1:=lista_autos1^.sig;
  if (lista_autos1<>nil) then
    buscar_patente:=lista_autos1^.ele.modelo
  else
    buscar_patente:='No existe la patente';
end;
function buscar_abb_marcas(abb_marcas: t_abb_marcas; patente: string): string;
var
  modelo: string;
begin
  if (abb_marcas=nil) then
    buscar_abb_marcas:='No existe la patente'
  else
  begin
    modelo:=buscar_patente(abb_marcas^.ele.autos,patente);
    if (modelo='No existe la patente') then
      modelo:=buscar_abb_marcas(abb_marcas^.hi,patente);
    if (modelo='No existe la patente') then
      modelo:=buscar_abb_marcas(abb_marcas^.hd,patente);
    buscar_abb_marcas:=modelo;
  end;
end;
var
  vector_autos: t_vector_autos;
  abb_patentes: t_abb_patentes;
  abb_marcas: t_abb_marcas;
  marca, patente: string;
begin
  randomize;
  abb_patentes:=nil; abb_marcas:=nil;
```

```
  inicializar_vector_autos(vector_autos);
  writeln(); textcolor(red); writeln('INCISO (a)'); writeln();
  cargar_abbs(abb_patentes,abb_marcas);
  if ((abb_patentes<>nil) and (abb_marcas<>nil)) then
  begin
    writeln(); textcolor(red); writeln('ABB_PATENTES:'); writeln();
    imprimir_abb_patentes(abb_patentes);
    writeln(); textcolor(red); writeln('ABB_MARCAS:'); writeln();
    imprimir_abb_marcas(abb_marcas);
    writeln(); textcolor(red); writeln('INCISO (b)'); writeln();
    marca:='Marca '+random_string(1);
    textcolor(green); write('La cantidad de autos en el abb_patentes de la marca ');
textcolor(yellow); write(marca); textcolor(green); write(' es '); textcolor(red);
writeln(contar_abb_patentes(abb_patentes,marca));
    writeln(); textcolor(red); writeln('INCISO (c)'); writeln();
    textcolor(green); write('La cantidad de autos en el abb_marcas de la marca ');
textcolor(yellow); write(marca); textcolor(green); write(' es '); textcolor(red);
writeln(contar_abb_marcas(abb_marcas,marca));
    writeln(); textcolor(red); writeln('INCISO (d)'); writeln();
    cargar_vector_autos(vector_autos,abb_patentes);
    imprimir_vector_autos(vector_autos);
    writeln(); textcolor(red); writeln('INCISO (e)'); writeln();
    patente:=random_string(2);
    textcolor(green); write('El modelo del auto de la patente '); textcolor(yellow);
write(patente); textcolor(green); write(' es '); textcolor(red);
writeln(buscar_abb_patentes(abb_patentes,patente));
    writeln(); textcolor(red); writeln('INCISO (f)'); writeln();
    textcolor(green); write('El modelo del auto de la patente '); textcolor(yellow);
write(patente); textcolor(green); write(' es '); textcolor(red);
write(buscar_abb_marcas(abb_marcas,patente));
  end;
end.
```

Ejercicio 3.

Un supermercado requiere el procesamiento de sus productos. De cada producto, se conoce código, rubro (1..10), stock y precio unitario. Se pide:

(a) *Generar una estructura adecuada que permita agrupar los productos por rubro. A su vez, para cada rubro, se requiere que la búsqueda de un producto por código sea lo más eficiente posible. La lectura finaliza con el código de producto igual a -1.*

(b) *Implementar un módulo que reciba la estructura generada en (a), un rubro y un código de producto y retorne si dicho código existe o no para ese rubro.*

(c) *Implementar un módulo que reciba la estructura generada en (a) y retorne, para cada rubro, el código y stock del producto con mayor código.*

(d) *Implementar un módulo que reciba la estructura generada en (a), dos códigos y retorne, para cada rubro, la cantidad de productos con códigos entre los dos valores ingresados.*

```
program TP5_E3;
{$codepage UTF8}
uses crt;
const
  rubro_ini=1; rubro_fin=10;
  codigo_salida=-1;
type
  t_rubro=rubro_ini..rubro_fin;
  t_registro_producto1=record
    codigo: int16;
    rubro: t_rubro;
    stock: int16;
    precio: real;
  end;
  t_registro_producto2=record
    codigo: int16;
    stock: int16;
    precio: real;
  end;
  t_abb_productos=^t_nodo_abb_productos;
  t_nodo_abb_productos=record
    ele: t_registro_producto2;
    hi: t_abb_productos;
    hd: t_abb_productos;
  end;
  t_vector_abbs=array[t_rubro] of t_abb_productos;
  t_registro_producto3=record
    codigo: int16;
    stock: int16;
  end;
  t_vector_productos=array[t_rubro] of t_registro_producto3;
  t_vector_cantidades=array[t_rubro] of int16;
procedure inicializar_vector_abbs(var vector_abbs: t_vector_abbs);
var
  i: t_rubro;
begin
  for i:= rubro_ini to rubro_fin do
    vector_abbs[i]:=nil;
end;
procedure inicializar_vector_productos(var vector_productos: t_vector_productos);
```

```
var
  i: t_rubro;
begin
  for i:= rubro_ini to rubro_fin do
  begin
    vector_productos[i].codigo:=codigo_salida;
    vector_productos[i].stock:=0;
  end;
end;
procedure inicializar_vector_cantidades(var vector_cantidades: t_vector_cantidades);
var
  i: t_rubro;
begin
  for i:= rubro_ini to rubro_fin do
    vector_cantidades[i]:=0;
end;
procedure leer_producto(var registro_producto1: t_registro_producto1);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_producto1.codigo:=codigo_salida
  else
    registro_producto1.codigo:=1+random(high(int16));
  if (registro_producto1.codigo<>codigo_salida) then
  begin
    registro_producto1.rubro:=rubro_ini+random(rubro_fin);
    registro_producto1.stock:=1+random(high(int16));
    registro_producto1.precio:=1+random(100);
  end;
end;
procedure cargar_registro_producto2(var registro_producto2: t_registro_producto2;
registro_producto1: t_registro_producto1);
begin
  registro_producto2.codigo:=registro_producto1.codigo;
  registro_producto2.stock:=registro_producto1.stock;
  registro_producto2.precio:=registro_producto1.precio;
end;
procedure agregar_abb_productos(var abb_productos: t_abb_productos; registro_producto1:
t_registro_producto1);
begin
  if (abb_productos=nil) then
  begin
    new(abb_productos);
    cargar_registro_producto2(abb_productos^.ele,registro_producto1);
    abb_productos^.hi:=nil;
    abb_productos^.hd:=nil;
  end
  else
    if (registro_producto1.codigo<=abb_productos^.ele.codigo) then
      agregar_abb_productos(abb_productos^.hi,registro_producto1)
    else
      agregar_abb_productos(abb_productos^.hd,registro_producto1);
end;
procedure cargar_vector_abbs(var vector_abbs: t_vector_abbs);
var
  registro_producto1: t_registro_producto1;
begin
  leer_producto(registro_producto1);
  while (registro_producto1.codigo<>codigo_salida) do
  begin
    agregar_abb_productos(vector_abbs[registro_producto1.rubro],registro_producto1);
    leer_producto(registro_producto1);
  end;
end;
```

```
procedure imprimir_registro_producto2(registro_producto2: t_registro_producto2; rubro:
t_rubro);
begin
  textcolor(green); write('El código de producto del producto del rubro '); textcolor(yellow);
write(rubro); textcolor(green); write(' es '); textcolor(red);
writeln(registro_producto2.codigo);
  textcolor(green); write('El stock del producto del rubro '); textcolor(yellow);
write(rubro); textcolor(green); write(' es '); textcolor(red);
writeln(registro_producto2.stock);
  textcolor(green); write('El precio del producto del rubro '); textcolor(yellow);
write(rubro); textcolor(green); write(' es '); textcolor(red);
writeln(registro_producto2.precio:0:2);
end;
procedure imprimir_abb_productos(abb_productos: t_abb_productos; rubro: t_rubro);
begin
  if (abb_productos<>nil) then
  begin
    imprimir_abb_productos(abb_productos^.hi,rubro);
    imprimir_registro_producto2(abb_productos^.ele,rubro);
    imprimir_abb_productos(abb_productos^.hd,rubro);
  end;
end;
procedure imprimir_vector_abbs(vector_abbs: t_vector_abbs);
var
  i: t_rubro;
begin
  for i:= rubro_ini to rubro_fin do
  begin
    textcolor(green); write('La información de los productos del rubro '); textcolor(yellow);
write(i); textcolor(green); writeln(' es:');
    imprimir_abb_productos(vector_abbs[i],i);
    writeln();
  end;
end;
function buscar_abb_productos(abb_productos: t_abb_productos; codigo: int16): boolean;
begin
  if (abb_productos=nil) then
    buscar_abb_productos:=false
  else
    if (codigo=abb_productos^.ele.codigo) then
      buscar_abb_productos:=true
    else if (codigo<abb_productos^.ele.codigo) then
      buscar_abb_productos:=buscar_abb_productos(abb_productos^.hi,codigo)
    else
      buscar_abb_productos:=buscar_abb_productos(abb_productos^.hd,codigo);
end;
procedure cargar_registro_producto3(var registro_producto3: t_registro_producto3;
abb_productos: t_abb_productos);
begin
  if (abb_productos^.hd=nil) then
  begin
    registro_producto3.codigo:=abb_productos^.ele.codigo;
    registro_producto3.stock:=abb_productos^.ele.stock;
  end
  else
    cargar_registro_producto3(registro_producto3,abb_productos^.hd);
end;
procedure cargar_vector_productos(var vector_productos: t_vector_productos; vector_abbs:
t_vector_abbs);
var
  i: t_rubro;
begin
  for i:= rubro_ini to rubro_fin do
    if (vector_abbs[i]<>nil) then
      cargar_registro_producto3(vector_productos[i],vector_abbs[i]);
end;
```

```
procedure imprimir_registro_producto3(registro_producto3: t_registro_producto3; rubro:
t_rubro);
begin
  textcolor(green); write('El mayor código de producto del rubro '); textcolor(yellow);
write(rubro); textcolor(green); write(' es '); textcolor(red);
writeln(registro_producto3.codigo);
  textcolor(green); write('El stock del mayor código de producto del rubro ');
textcolor(yellow); write(rubro); textcolor(green); write(' es '); textcolor(red);
writeln(registro_producto3.stock);
end;
procedure imprimir_vector_productos(vector_productos: t_vector_productos);
var
  i: t_rubro;
begin
  for i:= rubro_ini to rubro_fin do
  begin
    imprimir_registro_producto3(vector_productos[i],i);
    writeln();
  end;
end;
procedure verificar_codigos(var codigo1, codigo2: int16);
var
  aux: int16;
begin
  if (codigo1>codigo2) then
  begin
    aux:=codigo1;
    codigo1:=codigo2;
    codigo2:=aux;
  end;
end;
function contar_productos(abb_productos: t_abb_productos; codigo1, codigo2: int16): int16;
begin
  if (abb_productos=nil) then
    contar_productos:=0
  else
    if (codigo1>=abb_productos^.ele.codigo) then
      contar_productos:=contar_productos(abb_productos^.hd,codigo1,codigo2)
    else if (codigo2<=abb_productos^.ele.codigo) then
      contar_productos:=contar_productos(abb_productos^.hi,codigo1,codigo2)
    else
      contar_productos:=contar_productos(abb_productos^.hi,codigo1,codigo2)+contar_productos(a
bb_productos^.hd,codigo1,codigo2)+1;
end;
procedure cargar_vector_cantidades(var vector_cantidades: t_vector_cantidades; vector_abbs:
t_vector_abbs; codigo1, codigo2: int16);
var
  i: t_rubro;
begin
  for i:= rubro_ini to rubro_fin do
    vector_cantidades[i]:=contar_productos(vector_abbs[i],codigo1,codigo2);
end;
procedure imprimir_vector_cantidades(vector_cantidades: t_vector_cantidades; codigo1, codigo2:
int16);
var
  i: t_rubro;
begin
  for i:= rubro_ini to rubro_fin do
  begin
    textcolor(green); write('La cantidad de productos del rubro '); textcolor(yellow);
write(i); textcolor(green); write(' (cuyo código de producto se encuentra entre ');
textcolor(yellow); write(codigo1); textcolor(green); write(' y '); textcolor(yellow);
write(codigo2); textcolor(green); write(') es '); textcolor(red);
writeln(vector_cantidades[i]);
  end;
end;
```

```
var
  vector_abbs: t_vector_abbs;
  vector_productos: t_vector_productos;
  vector_cantidades: t_vector_cantidades;
  rubro: t_rubro;
  codigo, codigo1, codigo2: int16;
begin
  randomize;
  inicializar_vector_abbs(vector_abbs);
  inicializar_vector_productos(vector_productos);
  inicializar_vector_cantidades(vector_cantidades);
  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
  cargar_vector_abbs(vector_abbs);
  imprimir_vector_abbs(vector_abbs);
  writeln(); textcolor(red); writeln('INCISO (b):'); writeln();
  rubro:=rubro_ini+random(rubro_fin); codigo:=1+random(high(int16));
  textcolor(green); write('¿El código '); textcolor(yellow); write(codigo); textcolor(green);
write(' se encuentra en el abb del rubro '); textcolor(yellow); write(rubro);
textcolor(green); write('?: '); textcolor(red);
writeln(buscar_abb_productos(vector_abbs[rubro],codigo));
  writeln(); textcolor(red); writeln('INCISO (c):'); writeln();
  cargar_vector_productos(vector_productos,vector_abbs);
  imprimir_vector_productos(vector_productos);
  writeln(); textcolor(red); writeln('INCISO (d):'); writeln();
  codigo1:=1+random(high(int16)); codigo2:=1+random(high(int16));
  verificar_codigos(codigo1,codigo2);
  cargar_vector_cantidades(vector_cantidades,vector_abbs,codigo1,codigo2);
  imprimir_vector_cantidades(vector_cantidades,codigo1,codigo2);
end.
```

Ejercicio 4.

Una oficina requiere el procesamiento de los reclamos de las personas. De cada reclamo, se lee código, DNI de la persona, año y tipo de reclamo. La lectura finaliza con el código de igual a -1. Se pide:

(a) *Un módulo que retorne estructura adecuada para la búsqueda por DNI. Para cada DNI, se deben tener almacenados cada reclamo y la cantidad total de reclamos que realizó.*

(b) *Un módulo que reciba la estructura generada en (a) y un DNI y retorne la cantidad de reclamos efectuados por ese DNI.*

(c) *Un módulo que reciba la estructura generada en (a) y dos DNI y retorne la cantidad de reclamos efectuados por todos los DNI comprendidos entre los dos DNI recibidos.*

(d) *Un módulo que reciba la estructura generada en (a) y un año y retorne los códigos de los reclamos realizados en el año recibido.*

```
program TP5_E4;
{$codepage UTF8}
uses crt;
const
  anio_ini=2000; anio_fin=2023;
  codigo_salida=-1;
type
  t_anio=anio_ini..anio_fin;
  t_registro_reclamo1=record
    codigo: int16;
    dni: int8;
    anio: t_anio;
    tipo: string;
  end;
  t_registro_reclamo2=record
    codigo: int16;
    anio: t_anio;
    tipo: string;
  end;
  t_lista_reclamos=^t_nodo_reclamos;
  t_nodo_reclamos=record
    ele: t_registro_reclamo2;
    sig: t_lista_reclamos;
  end;
  t_registro_dni=record
    dni: int8;
    reclamos: t_lista_reclamos;
    cantidad: int16;
  end;
  t_abb_dnis=^t_nodo_abb_dnis;
  t_nodo_abb_dnis=record
    ele: t_registro_dni;
    hi: t_abb_dnis;
    hd: t_abb_dnis;
  end;
  t_lista_codigos=^t_nodo_codigos2;
  t_nodo_codigos2=record
    ele: int16;
    sig: t_lista_codigos;
  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_reclamo(var registro_reclamo1: t_registro_reclamo1);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_reclamo1.codigo:=codigo_salida
  else
    registro_reclamo1.codigo:=1+random(high(int16));
  if (registro_reclamo1.codigo<>codigo_salida) then
  begin
    registro_reclamo1.dni:=1+random(high(int8));
    registro_reclamo1.anio:=anio_ini+random(anio_fin-anio_ini+1);
    registro_reclamo1.tipo:=random_string(5+random(6));
  end;
end;
procedure cargar_registro_reclamo2(var registro_reclamo2: t_registro_reclamo2;
registro_reclamo1: t_registro_reclamo1);
begin
  registro_reclamo2.codigo:=registro_reclamo1.codigo;
  registro_reclamo2.anio:=registro_reclamo1.anio;
  registro_reclamo2.tipo:=registro_reclamo1.tipo;
end;
procedure agregar_adelante_lista_reclamos(var lista_reclamos: t_lista_reclamos;
registro_reclamo1: t_registro_reclamo1);
var
  nuevo: t_lista_reclamos;
begin
  new(nuevo);
  cargar_registro_reclamo2(nuevo^.ele,registro_reclamo1);
  nuevo^.sig:=lista_reclamos;
  lista_reclamos:=nuevo;
end;
procedure cargar_registro_dni(var registro_dni: t_registro_dni; registro_reclamo1:
t_registro_reclamo1);
begin
  registro_dni.dni:=registro_reclamo1.dni;
  registro_dni.reclamos:=nil;
  agregar_adelante_lista_reclamos(registro_dni.reclamos,registro_reclamo1);
  registro_dni.cantidad:=1;
end;
procedure agregar_abb_dnis(var abb_dnis: t_abb_dnis; registro_reclamo1: t_registro_reclamo1);
begin
  if (abb_dnis=nil) then
  begin
    new(abb_dnis);
    cargar_registro_dni(abb_dnis^.ele,registro_reclamo1);
    abb_dnis^.hi:=nil;
    abb_dnis^.hd:=nil;
  end
  else
    if (registro_reclamo1.dni=abb_dnis^.ele.dni) then
    begin
      agregar_adelante_lista_reclamos(abb_dnis^.ele.reclamos,registro_reclamo1);
      abb_dnis^.ele.cantidad:=abb_dnis^.ele.cantidad+1;
    end
```

```
    else if (registro_reclamo1.dni<abb_dnis^.ele.dni) then
      agregar_abb_dnis(abb_dnis^.hi,registro_reclamo1)
    else
      agregar_abb_dnis(abb_dnis^.hd,registro_reclamo1);
end;
procedure cargar_abb_dnis(var abb_dnis: t_abb_dnis);
var
  registro_reclamo1: t_registro_reclamo1;
begin
  leer_reclamo(registro_reclamo1);
  while (registro_reclamo1.codigo<>codigo_salida) do
  begin
    agregar_abb_dnis(abb_dnis,registro_reclamo1);
    leer_reclamo(registro_reclamo1);
  end;
end;
procedure imprimir_registro_reclamo2(registro_reclamo2: t_registro_reclamo2; dni: int8;
reclamo: int16);
begin
  textcolor(green); write('El código de reclamo del reclamo '); textcolor(yellow);
write(reclamo); textcolor(green); write(' del DNI '); textcolor(yellow); write(dni);
textcolor(green); write(' es '); textcolor(red); writeln(registro_reclamo2.codigo);
  textcolor(green); write('El año del reclamo '); textcolor(yellow); write(reclamo);
textcolor(green); write(' del DNI '); textcolor(yellow); write(dni); textcolor(green); write('
es '); textcolor(red); writeln(registro_reclamo2.anio);
  textcolor(green); write('El tipo de reclamo del reclamo '); textcolor(yellow);
write(reclamo); textcolor(green); write(' del DNI '); textcolor(yellow); write(dni);
textcolor(green); write(' es '); textcolor(red); writeln(registro_reclamo2.tipo);
end;
procedure imprimir_lista_reclamos(lista_reclamos: t_lista_reclamos; dni: int8);
var
  i: int16;
begin
  i:=0;
  while (lista_reclamos<>nil) do
  begin
    i:=i+1;
    imprimir_registro_reclamo2(lista_reclamos^.ele,dni,i);
    lista_reclamos:=lista_reclamos^.sig;
  end;
end;
procedure imprimir_registro_dni(registro_dni: t_registro_dni);
begin
  textcolor(green); write('El DNI de la persona es '); textcolor(red);
writeln(registro_dni.dni);
  textcolor(green); write('La cantidad total de reclamos que realizó la persona es ');
textcolor(red); writeln(registro_dni.cantidad);
  imprimir_lista_reclamos(registro_dni.reclamos,registro_dni.dni);
  writeln();
end;
procedure imprimir_abb_dnis(abb_dnis: t_abb_dnis);
begin
  if (abb_dnis<>nil) then
  begin
    imprimir_abb_dnis(abb_dnis^.hi);
    imprimir_registro_dni(abb_dnis^.ele);
    imprimir_abb_dnis(abb_dnis^.hd);
  end;
end;
function contar_reclamos1(abb_dnis: t_abb_dnis; dni: int8): int16;
begin
  if (abb_dnis=nil) then
    contar_reclamos1:=0
  else
    if (dni=abb_dnis^.ele.dni) then
      contar_reclamos1:=abb_dnis^.ele.cantidad
```

```
    else if (dni<abb_dnis^.ele.dni) then
      contar_reclamos1:=contar_reclamos1(abb_dnis^.hi,dni)
    else
      contar_reclamos1:=contar_reclamos1(abb_dnis^.hd,dni);
end;
procedure verificar_dnis(var dni1, dni2: int8);
var
  aux: int8;
begin
  if (dni1>dni2) then
  begin
    aux:=dni1;
    dni1:=dni2;
    dni2:=aux;
  end;
end;
function contar_reclamos2(abb_dnis: t_abb_dnis; dni1, dni2: int8): int16;
begin
  if (abb_dnis=nil) then
    contar_reclamos2:=0
  else
    if (dni1>=abb_dnis^.ele.dni) then
      contar_reclamos2:=contar_reclamos2(abb_dnis^.hd,dni1,dni2)
    else if (dni2<=abb_dnis^.ele.dni) then
      contar_reclamos2:=contar_reclamos2(abb_dnis^.hi,dni1,dni2)
    else
      contar_reclamos2:=contar_reclamos2(abb_dnis^.hi,dni1,dni2)+contar_reclamos2(abb_dnis^.hd
,dni1,dni2)+1;
end;
procedure agregar_adelante_lista_codigos(var lista_codigos: t_lista_codigos; codigo: int16);
var
  nuevo: t_lista_codigos;
begin
  new(nuevo);
  nuevo^.ele:=codigo;
  nuevo^.sig:=lista_codigos;
  lista_codigos:=nuevo;
end;
procedure recorrer_lista_reclamos(var lista_codigos: t_lista_codigos; lista_reclamos:
t_lista_reclamos; anio: t_anio);
begin
  while (lista_reclamos<>nil) do
  begin
    if (anio=lista_reclamos^.ele.anio) then
      agregar_adelante_lista_codigos(lista_codigos,lista_reclamos^.ele.codigo);
    lista_reclamos:=lista_reclamos^.sig;
  end;
end;
procedure cargar_lista_codigos(var lista_codigos: t_lista_codigos; abb_dnis: t_abb_dnis; anio:
t_anio);
begin
  if (abb_dnis<>nil) then
  begin
    cargar_lista_codigos(lista_codigos,abb_dnis^.hd,anio);
    recorrer_lista_reclamos(lista_codigos,abb_dnis^.ele.reclamos,anio);
    cargar_lista_codigos(lista_codigos,abb_dnis^.hi,anio);
  end;
end;
procedure imprimir_lista_codigos(lista_codigos: t_lista_codigos; anio: t_anio);
var
  i: int16;
begin
  i:=0;
  while (lista_codigos<>nil) do
  begin
    i:=i+1;
```

```
    textcolor(green); write('Código de reclamo '); textcolor(yellow); write(i);
textcolor(green); write(' del año '); textcolor(yellow); write(anio); textcolor(green);
write(': '); textcolor(red); writeln(lista_codigos^.ele);
    lista_codigos:=lista_codigos^.sig;
  end;
end;
var
  lista_codigos: t_lista_codigos;
  abb_dnis: t_abb_dnis;
  anio: t_anio;
  dni, dni1, dni2: int8;
begin
  randomize;
  abb_dnis:=nil;
  lista_codigos:=nil;
  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
  cargar_abb_dnis(abb_dnis);
  if (abb_dnis<>nil) then
  begin
    imprimir_abb_dnis(abb_dnis);
    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();
    dni:=1+random(high(int8));
    textcolor(green); write('La cantidad de reclamos del DNI '); textcolor(yellow);
write(dni); textcolor(green); write(' es '); textcolor(red);
writeln(contar_reclamos1(abb_dnis,dni));
    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();
    dni1:=1+random(high(int8)); dni2:=1+random(high(int8));
    verificar_dnis(dni1,dni2);
    textcolor(green); write('La cantidad de reclamos en el abb cuyo DNI se encuentra entre ');
textcolor(yellow); write(dni1); textcolor(green); write(' y '); textcolor(yellow);
write(dni2); textcolor(green); write(' es '); textcolor(red);
writeln(contar_reclamos2(abb_dnis,dni1,dni2));
    writeln(); textcolor(red); writeln('INCISO (d):'); writeln();
    anio:=anio_ini+random(anio_fin-anio_ini+1);
    cargar_lista_codigos(lista_codigos,abb_dnis,anio);
    if (lista_codigos<>nil) then
      imprimir_lista_codigos(lista_codigos,anio);
  end;
end.
```

Ejercicio 5.

Realizar el inciso (a) del ejercicio anterior, pero sabiendo que todos los reclamos de un mismo DNI se leen de forma consecutiva (no significa que vengan ordenados los DNI).

```
program TP5_E5;
{$codepage UTF8}
uses crt;
const
  anio_ini=2000; anio_fin=2023;
  codigo_salida=-1;
type
  t_anio=anio_ini..anio_fin;
  t_registro_reclamo1=record
    codigo: int16;
    dni: int32;
    anio: t_anio;
    tipo: string;
  end;
  t_registro_reclamo2=record
    codigo: int16;
    anio: t_anio;
    tipo: string;
  end;
  t_lista_reclamos=^t_nodo_reclamos;
  t_nodo_reclamos=record
    ele: t_registro_reclamo2;
    sig: t_lista_reclamos;
  end;
  t_registro_dni=record
    dni: int32;
    reclamos: t_lista_reclamos;
    cantidad: int16;
  end;
  t_abb_dnis=^t_nodo_abb_dnis;
  t_nodo_abb_dnis=record
    ele: t_registro_dni;
    hi: t_abb_dnis;
    hd: t_abb_dnis;
  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_reclamo(var registro_reclamo1: t_registro_reclamo1; dni: int32);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    registro_reclamo1.codigo:=codigo_salida
  else
    registro_reclamo1.codigo:=1+random(high(int16));
  if (registro_reclamo1.codigo<>codigo_salida) then
  begin
    i:=random(2);
    if (i=0) then
```

```
      registro_reclamo1.dni:=dni
    else
      registro_reclamo1.dni:=10000000+random(40000001);
    registro_reclamo1.anio:=anio_ini+random(anio_fin-anio_ini+1);
    registro_reclamo1.tipo:=random_string(5+random(6));
  end;
end;
procedure cargar_registro_reclamo2(var registro_reclamo2: t_registro_reclamo2;
registro_reclamo1: t_registro_reclamo1);
begin
  registro_reclamo2.codigo:=registro_reclamo1.codigo;
  registro_reclamo2.anio:=registro_reclamo1.anio;
  registro_reclamo2.tipo:=registro_reclamo1.tipo;
end;
procedure agregar_adelante_lista_reclamos(var lista_reclamos: t_lista_reclamos;
registro_reclamo1: t_registro_reclamo1);
var
  nuevo: t_lista_reclamos;
begin
  new(nuevo);
  cargar_registro_reclamo2(nuevo^.ele,registro_reclamo1);
  nuevo^.sig:=lista_reclamos;
  lista_reclamos:=nuevo;
end;
procedure agregar_abb_dnis(var abb_dnis: t_abb_dnis; registro_dni: t_registro_dni);
begin
  if (abb_dnis=nil) then
  begin
    new(abb_dnis);
    abb_dnis^.ele:=registro_dni;
    abb_dnis^.hi:=nil;
    abb_dnis^.hd:=nil;
  end
  else
    if (registro_dni.dni<abb_dnis^.ele.dni) then
      agregar_abb_dnis(abb_dnis^.hi,registro_dni)
    else if (registro_dni.dni>abb_dnis^.ele.dni) then
      agregar_abb_dnis(abb_dnis^.hd,registro_dni);
end;
procedure cargar_abb_dnis(var abb_dnis: t_abb_dnis);
var
  registro_reclamo1: t_registro_reclamo1;
  registro_dni: t_registro_dni;
begin
  leer_reclamo(registro_reclamo1,10000000+random(40000001));
  while (registro_reclamo1.codigo<>codigo_salida) do
  begin
    registro_dni.dni:=registro_reclamo1.dni;
    registro_dni.reclamos:=nil;
    registro_dni.cantidad:=0;
    while ((registro_reclamo1.codigo<>codigo_salida) and
(registro_reclamo1.dni=registro_dni.dni)) do
    begin
      agregar_adelante_lista_reclamos(registro_dni.reclamos,registro_reclamo1);
      registro_dni.cantidad:=registro_dni.cantidad+1;
      leer_reclamo(registro_reclamo1,registro_dni.dni);
    end;
    agregar_abb_dnis(abb_dnis,registro_dni);
  end;
end;
procedure imprimir_registro_reclamo2(registro_reclamo2: t_registro_reclamo2; dni: int32;
reclamo: int16);
begin
  textcolor(green); write('El código de reclamo del reclamo '); textcolor(yellow);
write(reclamo); textcolor(green); write(' del DNI '); textcolor(yellow); write(dni);
textcolor(green); write(' es '); textcolor(red); writeln(registro_reclamo2.codigo);
```

```
  textcolor(green); write('El año del reclamo '); textcolor(yellow); write(reclamo);
textcolor(green); write(' del DNI '); textcolor(yellow); write(dni); textcolor(green); write('
es '); textcolor(red); writeln(registro_reclamo2.anio);
  textcolor(green); write('El tipo de reclamo del reclamo '); textcolor(yellow);
write(reclamo); textcolor(green); write(' del DNI '); textcolor(yellow); write(dni);
textcolor(green); write(' es '); textcolor(red); writeln(registro_reclamo2.tipo);
end;
procedure imprimir_lista_reclamos(lista_reclamos: t_lista_reclamos; dni: int32);
var
  i: int16;
begin
  i:=0;
  while (lista_reclamos<>nil) do
  begin
    i:=i+1;
    imprimir_registro_reclamo2(lista_reclamos^.ele,dni,i);
    lista_reclamos:=lista_reclamos^.sig;
  end;
end;
procedure imprimir_registro_dni(registro_dni: t_registro_dni);
begin
  textcolor(green); write('El DNI de la persona es '); textcolor(red);
writeln(registro_dni.dni);
  textcolor(green); write('La cantidad total de reclamos que realizó la persona es ');
textcolor(red); writeln(registro_dni.cantidad);
  imprimir_lista_reclamos(registro_dni.reclamos,registro_dni.dni);
  writeln();
end;
procedure imprimir_abb_dnis(abb_dnis: t_abb_dnis);
begin
  if (abb_dnis<>nil) then
  begin
    imprimir_abb_dnis(abb_dnis^.hi);
    imprimir_registro_dni(abb_dnis^.ele);
    imprimir_abb_dnis(abb_dnis^.hd);
  end;
end;
var
  abb_dnis: t_abb_dnis;
begin
  randomize;
  abb_dnis:=nil;
  cargar_abb_dnis(abb_dnis);
  imprimir_abb_dnis(abb_dnis);
end.
```