**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(abb\_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: int8;

    dni: int32;

    anio: t\_anio;

    tipo: string;

  end;

  t\_registro\_reclamo2=record

    codigo: int8;

    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;

  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);

begin

  registro\_reclamo1.codigo:=0;

  while (registro\_reclamo1.codigo=0) do

    registro\_reclamo1.codigo:=codigo\_salida+random(high(int8));

  if (registro\_reclamo1.codigo<>codigo\_salida) then

  begin

    registro\_reclamo1.dni:=1+random(high(int32));

    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: 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;

function contar\_reclamos1(abb\_dnis: t\_abb\_dnis; dni: int32): 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: int32);

var

  aux: int32;

begin

  if (dni1>dni2) then

  begin

    aux:=dni1;

    dni1:=dni2;

    dni2:=aux;

  end;

end;

function contar\_reclamos2(abb\_dnis: t\_abb\_dnis; dni1, dni2: int32): 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: int8);

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);

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(': '); 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: int32;

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();

    textcolor(green); write('Introducir DNI para buscar en el abb la cantidad de reclamos de ese DNI: ');

    textcolor(yellow); readln(dni);

    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();

    textcolor(green); writeln('Introducir dos DNI para buscar en el abb la cantidad de reclamos cuyo DNI se encuentra entre esos dos DNI: ');

    textcolor(yellow); readln(dni1); readln(dni2);

    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();

    textcolor(green); write('Introducir año para buscar en el abb los códigos de reclamo de ese año: ');

    textcolor(yellow); readln(anio);

    cargar\_lista\_codigos(lista\_codigos,abb\_dnis,anio);

    if (lista\_codigos<>nil) then

      imprimir\_lista\_codigos(lista\_codigos);

  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: int8;

    dni: int32;

    anio: t\_anio;

    tipo: string;

  end;

  t\_registro\_reclamo2=record

    codigo: int8;

    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);

begin

  registro\_reclamo1.codigo:=0;

  while (registro\_reclamo1.codigo=0) do

    registro\_reclamo1.codigo:=codigo\_salida+random(high(int8));

  if (registro\_reclamo1.codigo<>codigo\_salida) then

  begin

    registro\_reclamo1.dni:=1+random(high(int32));

    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);

  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);

    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.