**Trabajo Práctico N° 1:**

**Módulo Imperativo (Ordenación).**

**Ejercicio 1.**

*Se desea procesar la información de las ventas de productos de un comercio (como máximo, 50). Implementar un programa que invoque los siguientes módulos:*

**(a)** *Un módulo que retorne la información de las ventas en un vector. De cada venta, se conoce el día de la venta, código del producto (entre 1 y 15) y cantidad vendida (como máximo, 99 unidades). El código debe generarse automáticamente (random) y la cantidad se debe leer. El ingreso de las ventas finaliza con el día de venta 0 (no se procesa).*

**(b)** *Un módulo que muestre el contenido del vector resultante del inciso (a).*

**(c)** *Un módulo que ordene el vector de ventas por código.*

**(d)** *Un módulo que muestre el contenido del vector resultante del inciso (c).*

**(e)** *Un módulo que elimine, del vector ordenado, las ventas con código de producto entre dos valores que se ingresan como parámetros.*

**(f)** *Un módulo que muestre el contenido del vector resultante del inciso (e).*

**(g)** *Un módulo que retorne la información (ordenada por código de producto de menor a mayor) de cada código par de producto junto a la cantidad total de productos vendidos.*

**(h)** *Un módulo que muestre la información obtenida en el inciso (g).*

program TP1\_E1;

*{$codepage UTF8}*

uses crt;

const

  ventas\_total=50;

  dia\_ini=1; dia\_fin=31;

  codigo\_ini=1; codigo\_fin=15;

  cantidad\_total=99;

  dia\_salida=0;

type

  t\_venta=1..ventas\_total;

  t\_codigo=codigo\_ini..codigo\_fin;

  t\_cantidad=1..cantidad\_total;

  t\_registro\_venta=record

    dia: int8;

    codigo: t\_codigo;

    cantidad: t\_cantidad;

  end;

  t\_vector\_ventas=array[t\_venta] of t\_registro\_venta;

  t\_vector\_cantidades=array[t\_codigo] of int16;

procedure leer\_venta(var registro\_venta: t\_registro\_venta);

var

  i: int8;

begin

  i:=random(100);

  if (i=0) then

    registro\_venta.dia:=dia\_salida

  else

    registro\_venta.dia:=dia\_ini+random(dia\_fin);

  if (registro\_venta.dia<>dia\_salida) then

  begin

    registro\_venta.codigo:=codigo\_ini+random(codigo\_fin);

    registro\_venta.cantidad:=1+random(cantidad\_total);

  end;

end;

procedure cargar\_vector\_ventas(var vector\_ventas: t\_vector\_ventas; var ventas: int8);

var

  registro\_venta: t\_registro\_venta;

begin

  leer\_venta(registro\_venta);

  while ((registro\_venta.dia<>dia\_salida) and (ventas<ventas\_total)) do

  begin

    ventas:=ventas+1;

    vector\_ventas[ventas]:=registro\_venta;

    leer\_venta(registro\_venta);

  end;

end;

procedure imprimir\_registro\_venta(registro\_venta: t\_registro\_venta; venta: t\_venta);

begin

  textcolor(green); write('El día de la venta '); textcolor(yellow); write(venta); textcolor(green); write(' es '); textcolor(red); writeln(registro\_venta.dia);

  textcolor(green); write('El código de producto de la venta'); textcolor(yellow); write(venta); textcolor(green); write(' es '); textcolor(red); writeln(registro\_venta.codigo);

  textcolor(green); write('La cantidad vendida del producto de la venta '); textcolor(yellow); write(venta); textcolor(green); write(' es '); textcolor(red); writeln(registro\_venta.cantidad);

end;

procedure imprimir\_vector\_ventas(vector\_ventas: t\_vector\_ventas; ventas: int8);

var

  i: t\_venta;

begin

  for i:= 1 to ventas do

  begin

    textcolor(green); write('La información de la venta '); textcolor(yellow); write(i); textcolor(green); writeln(' es:');

    imprimir\_registro\_venta(vector\_ventas[i],i);

    writeln();

  end;

end;

procedure ordenar\_vector\_ventas(var vector\_ventas: t\_vector\_ventas; ventas: int8);

var

  item: t\_registro\_venta;

  i, j, k: t\_venta;

begin

  for i:= 1 to (ventas-1) do

  begin

    k:=i;

    for j:= (i+1) to ventas do

      if (vector\_ventas[j].codigo<vector\_ventas[k].codigo) then

        k:=j;

    item:=vector\_ventas[k];

    vector\_ventas[k]:=vector\_ventas[i];

    vector\_ventas[i]:=item;

  end;

end;

procedure verificar\_codigos(var codigo1, codigo2: t\_codigo);

var

  aux: t\_codigo;

begin

  if (codigo1>codigo2) then

  begin

    aux:=codigo1;

    codigo1:=codigo2;

    codigo2:=aux;

  end;

end;

procedure eliminar\_vector\_ventas(var vector\_ventas: t\_vector\_ventas; var ventas: int8; codigo1, codigo2: t\_codigo);

var

  i, i\_izq, i\_der, salto: t\_codigo;

begin

  i:=1;

  while ((i<ventas) and (vector\_ventas[i].codigo<=codigo1)) do

    i:=i+1;

  i\_izq:=i;

  while ((i<ventas) and (vector\_ventas[i].codigo<codigo2)) do

    i:=i+1;

  i\_der:=i;

  salto:=i\_der-i\_izq;

  while (i\_izq+salto<=ventas) do

  begin

    vector\_ventas[i\_izq]:=vector\_ventas[i\_izq+salto];

    i\_izq:=i\_izq+1;

  end;

  ventas:=ventas-salto;

end;

procedure inicializar\_vector\_cantidades(var vector\_cantidades: t\_vector\_cantidades);

var

  i: t\_codigo;

begin

  for i:= codigo\_ini to codigo\_fin do

  begin

    vector\_cantidades[i]:=0;

  end;

end;

procedure cargar\_vector\_cantidades(var vector\_cantidades: t\_vector\_cantidades; vector\_ventas: t\_vector\_ventas; ventas: int8);

var

  i: t\_venta;

  codigo: t\_codigo;

begin

  for i:= 1 to ventas do

  begin

    codigo:=vector\_ventas[i].codigo;

    if (codigo mod 2=0) then

      vector\_cantidades[codigo]:=vector\_cantidades[codigo]+vector\_ventas[i].cantidad;

  end;

end;

procedure imprimir\_vector\_cantidades(vector\_cantidades: t\_vector\_cantidades);

var

  i: t\_codigo;

begin

  for i:= codigo\_ini to codigo\_fin do

  begin

    textcolor(green); write('La cantidad total de productos vendidos del código de producto '); textcolor(yellow); write(i); textcolor(green); write(' es '); textcolor(red); writeln(vector\_cantidades[i]);

  end;

end;

var

  vector\_ventas: t\_vector\_ventas;

  vector\_cantidades: t\_vector\_cantidades;

  codigo1, codigo2: t\_codigo;

  ventas: int8;

begin

  randomize;

  ventas:=0;

  inicializar\_vector\_cantidades(vector\_cantidades);

  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();

  cargar\_vector\_ventas(vector\_ventas,ventas);

  if (ventas<>0) then

  begin

    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();

    imprimir\_vector\_ventas(vector\_ventas,ventas);

    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();

    ordenar\_vector\_ventas(vector\_ventas,ventas);

    writeln(); textcolor(red); writeln('INCISO (d):'); writeln();

    imprimir\_vector\_ventas(vector\_ventas,ventas);

    writeln(); textcolor(red); writeln('INCISO (e):'); writeln();

    codigo1:=codigo\_ini+random(codigo\_fin); codigo2:=codigo\_ini+random(codigo\_fin);

    verificar\_codigos(codigo1,codigo2);

    eliminar\_vector\_ventas(vector\_ventas,ventas,codigo1,codigo2);

    if (ventas<>0) then

    begin

      writeln(); textcolor(red); writeln('INCISO (f):'); writeln();

      imprimir\_vector\_ventas(vector\_ventas,ventas);

      writeln(); textcolor(red); writeln('INCISO (g):'); writeln();

      cargar\_vector\_cantidades(vector\_cantidades,vector\_ventas,ventas);

      writeln(); textcolor(red); writeln('INCISO (h):'); writeln();

      imprimir\_vector\_cantidades(vector\_cantidades);

    end;

  end;

end.

**Ejercicio 2.**

*El administrador de un edificio de oficinas cuenta, en papel, con la información del pago de las expensas de dichas oficinas. Implementar un programa que invoque a módulos para cada uno de los siguientes puntos:*

**(a)** *Generar un vector, sin orden, con, a lo sumo, las 300 oficinas que administra. De cada oficina, se ingresa el código de identificación, DNI del propietario y valor de la expensa. La lectura finaliza cuando se ingresa el código de identificación -1, el cual no se procesa.*

**(b)** *Ordenar el vector, aplicando el método de inserción, por código de identificación de la oficina.*

**(c)** *Ordenar el vector aplicando el método de selección, por código de identificación de la oficina.*

program TP1\_E2;

*{$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:=1+random(high(int16));

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

  begin

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

    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 ordenacion\_insercion\_vector\_oficinas(var vector\_oficinas: t\_vector\_oficinas; oficinas: int16);

var

  actual: t\_registro\_oficina;

  i, j: t\_oficina;

begin

  for i:= 2 to oficinas do

  begin

    actual:=vector\_oficinas[i];

    j:=i-1;

    while ((j>0) and (vector\_oficinas[j].codigo>actual.codigo)) do

    begin

      vector\_oficinas[j+1]:=vector\_oficinas[j];

      j:=j-1;

    end;

    vector\_oficinas[j+1]:=actual;

  end;

end;

procedure ordenacion\_seleccion\_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;

var

  vector\_oficinas: t\_vector\_oficinas;

  oficinas: 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();

    ordenacion\_insercion\_vector\_oficinas(vector\_oficinas,oficinas);

    imprimir\_vector\_oficinas(vector\_oficinas,oficinas);

    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();

    ordenacion\_seleccion\_vector\_oficinas(vector\_oficinas,oficinas);

    imprimir\_vector\_oficinas(vector\_oficinas,oficinas);

  end;

end.

**Ejercicio 3.**

*Netflix ha publicado la lista de películas que estarán disponibles durante el mes de diciembre de 2022. De cada película, se conoce: código de película, código de género (1: acción, 2: aventura, 3: drama, 4: suspenso, 5: comedia, 6: bélico, 7: documental y 8: terror) y puntaje promedio otorgado por las críticas. Implementar un programa que invoque a módulos para cada uno de los siguientes puntos:*

**(a)** *Leer los datos de películas, almacenarlos por orden de llegada y agrupados por código de género y retorne en una estructura de datos adecuada. La lectura finaliza cuando se lee el código de la película -1.*

**(b)** *Generar y retornar, en un vector, para cada género, el código de película con mayor puntaje obtenido entre todas las críticas, a partir de la estructura generada en (a).*

**(c)** *Ordenar los elementos del vector generado en (b) por puntaje, utilizando alguno de los dos métodos vistos en la teoría.*

**(d)** *Mostrar el código de película con mayor puntaje y el código de película con menor puntaje, del vector obtenido en el inciso (c).*

program TP1\_E3;

*{$codepage UTF8}*

uses crt;

const

  genero\_ini=1; genero\_fin=8;

  codigo\_salida=-1;

type

  t\_genero=genero\_ini..genero\_fin;

  t\_registro\_pelicula1=record

    codigo: int16;

    genero: t\_genero;

    puntaje: real;

  end;

  t\_registro\_pelicula2=record

    codigo: int16;

    puntaje: real;

  end;

  t\_lista\_peliculas=^t\_nodo\_peliculas;

  t\_nodo\_peliculas=record

    ele: t\_registro\_pelicula2;

    sig: t\_lista\_peliculas;

  end;

  t\_vector\_peliculas1=array[t\_genero] of t\_lista\_peliculas;

  t\_vector\_peliculas2=array[t\_genero] of t\_registro\_pelicula2;

procedure inicializar\_vector\_peliculas1(var vector\_peliculas1: t\_vector\_peliculas1);

var

  i: t\_genero;

begin

  for i:= genero\_ini to genero\_fin do

    vector\_peliculas1[i]:=nil;

end;

procedure leer\_pelicula(var registro\_pelicula1: t\_registro\_pelicula1);

var

  i: int8;

begin

  i:=random(100);

  if (i=0) then

    registro\_pelicula1.codigo:=codigo\_salida

  else

    registro\_pelicula1.codigo:=1+random(high(int16));

  if (registro\_pelicula1.codigo<>codigo\_salida) then

  begin

    registro\_pelicula1.genero:=genero\_ini+random(genero\_fin);

    registro\_pelicula1.puntaje:=1+random(10);

  end;

end;

procedure cargar\_registro\_pelicula2(var registro\_pelicula2: t\_registro\_pelicula2; registro\_pelicula1: t\_registro\_pelicula1);

begin

  registro\_pelicula2.codigo:=registro\_pelicula1.codigo;

  registro\_pelicula2.puntaje:=registro\_pelicula1.puntaje;

end;

procedure agregar\_atras\_lista\_peliculas(var lista\_peliculas: t\_lista\_peliculas; registro\_pelicula1: t\_registro\_pelicula1);

var

  aux, ult: t\_lista\_peliculas;

begin

  new(aux);

  cargar\_registro\_pelicula2(aux^.ele,registro\_pelicula1);

  aux^.sig:=nil;

  if (lista\_peliculas=nil) then

    lista\_peliculas:=aux

  else

  begin

    ult:=lista\_peliculas;

    while (ult^.sig<>nil) do

      ult:=ult^.sig;

    ult^.sig:=aux;

  end;

end;

procedure cargar\_vector\_peliculas1(var vector\_peliculas1: t\_vector\_peliculas1);

var

  registro\_pelicula1: t\_registro\_pelicula1;

begin

  leer\_pelicula(registro\_pelicula1);

  while (registro\_pelicula1.codigo<>codigo\_salida) do

  begin

    agregar\_atras\_lista\_peliculas(vector\_peliculas1[registro\_pelicula1.genero],registro\_pelicula1);

    leer\_pelicula(registro\_pelicula1);

  end;

end;

procedure imprimir\_registro\_pelicula2(registro\_pelicula2: t\_registro\_pelicula2; genero: t\_genero; pelicula: int16);

begin

  textcolor(green); write('El código de película de la película '); textcolor(yellow); write(pelicula); textcolor(green); write(' del género '); textcolor(yellow); write(genero); textcolor(green); write(' es '); textcolor(red); writeln(registro\_pelicula2.codigo);

  textcolor(green); write('El puntaje de la película '); textcolor(yellow); write(pelicula); textcolor(green); write(' del género '); textcolor(yellow); write(genero); textcolor(green); write(' es '); textcolor(red); writeln(registro\_pelicula2.puntaje:0:2);

end;

procedure imprimir\_lista\_peliculas(lista\_peliculas: t\_lista\_peliculas; genero: t\_genero);

var

  i: int16;

begin

  i:=0;

  while (lista\_peliculas<>nil) do

  begin

    i:=i+1;

    imprimir\_registro\_pelicula2(lista\_peliculas^.ele,genero,i);

    lista\_peliculas:=lista\_peliculas^.sig;

  end;

end;

procedure imprimir\_vector\_peliculas1(vector\_peliculas1: t\_vector\_peliculas1);

var

  i: t\_genero;

begin

  for i:= genero\_ini to genero\_fin do

  begin

    textcolor(green); write('La información de las películas del género '); textcolor(yellow); write(i); textcolor(green); writeln(' es:');

    imprimir\_lista\_peliculas(vector\_peliculas1[i],i);

    writeln();

  end;

end;

procedure cargar\_vector\_peliculas2(var vector\_peliculas2: t\_vector\_peliculas2; vector\_peliculas1: t\_vector\_peliculas1);

var

  i: t\_genero;

  codigo\_max: int16;

  puntaje\_max: real;

begin

  for i:= genero\_ini to genero\_fin do

  begin

    puntaje\_max:=-9999999; codigo\_max:=-1;

    while (vector\_peliculas1[i]<>nil) do

    begin

      if (vector\_peliculas1[i]^.ele.puntaje>puntaje\_max) then

      begin

        puntaje\_max:=vector\_peliculas1[i]^.ele.puntaje;

        codigo\_max:=vector\_peliculas1[i]^.ele.codigo;

      end;

      vector\_peliculas1[i]:=vector\_peliculas1[i]^.sig;

    end;

    vector\_peliculas2[i].codigo:=codigo\_max;

    vector\_peliculas2[i].puntaje:=puntaje\_max;

  end;

end;

procedure imprimir\_vector\_peliculas2(vector\_peliculas2: t\_vector\_peliculas2);

var

  i: t\_genero;

begin

  for i:= genero\_ini to genero\_fin do

  begin

    imprimir\_registro\_pelicula2(vector\_peliculas2[i],i,1);

    writeln();

  end;

end;

procedure ordenar\_vector\_peliculas2(var vector\_peliculas2: t\_vector\_peliculas2);

var

  item: t\_registro\_pelicula2;

  i, j, k: t\_genero;

begin

  for i:= genero\_ini to (genero\_fin-1) do

  begin

    k:=i;

    for j:= (i+1) to genero\_fin do

      if (vector\_peliculas2[j].puntaje<vector\_peliculas2[k].puntaje) then

        k:=j;

    item:=vector\_peliculas2[k];

    vector\_peliculas2[k]:=vector\_peliculas2[i];

    vector\_peliculas2[i]:=item;

  end;

end;

var

  vector\_peliculas1: t\_vector\_peliculas1;

  vector\_peliculas2: t\_vector\_peliculas2;

begin

  randomize;

  inicializar\_vector\_peliculas1(vector\_peliculas1);

  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();

  cargar\_vector\_peliculas1(vector\_peliculas1);

  imprimir\_vector\_peliculas1(vector\_peliculas1);

  writeln(); textcolor(red); writeln('INCISO (b):'); writeln();

  cargar\_vector\_peliculas2(vector\_peliculas2,vector\_peliculas1);

  imprimir\_vector\_peliculas2(vector\_peliculas2);

  writeln(); textcolor(red); writeln('INCISO (c):'); writeln();

  ordenar\_vector\_peliculas2(vector\_peliculas2);

  imprimir\_vector\_peliculas2(vector\_peliculas2);

  writeln(); textcolor(red); writeln('INCISO (d):'); writeln();

  textcolor(green); write('El código de película con mayor y menor puntaje son '); textcolor(red); write(vector\_peliculas2[genero\_fin].codigo); textcolor(green); write(' y '); textcolor(red); write(vector\_peliculas2[genero\_ini].codigo); textcolor(green); write(', respectivamente');

end.

**Ejercicio 4.**

*Una librería requiere el procesamiento de la información de sus productos. De cada producto, se conoce el código del producto, código de rubro (del 1 al 8) y precio. Implementar un programa que invoque a módulos para cada uno de los siguientes puntos:*

**(a)** *Leer los datos de los productos y almacenarlos ordenados por código de producto y agrupados por rubro, en una estructura de datos adecuada. El ingreso de los productos finaliza cuando se lee el precio 0.*

**(b)** *Una vez almacenados, mostrar los códigos de los productos pertenecientes a cada rubro.*

**(c)** *Generar un vector (de, a lo sumo, 30 elementos) con los productos del rubro 3. Considerar que puede haber más o menos de 30 productos del rubro 3. Si la cantidad de productos del rubro 3 es mayor a 30, almacenar los primeros 30 que están en la lista e ignorar el resto.*

**(d)** *Ordenar, por precio, los elementos del vector generado en (c) utilizando alguno de los dos métodos vistos en la teoría.*

**(e)** *Mostrar los precios del vector resultante del inciso (d).*

**(f)** *Calcular el promedio de los precios del vector resultante del inciso (d).*

program TP1\_E4;

*{$codepage UTF8}*

uses crt;

const

  rubro\_ini=1; rubro\_fin=8;

  precio\_salida=0.0;

  productos\_rubro3\_total=30;

type

  t\_rubro=rubro\_ini..rubro\_fin;

  t\_registro\_producto1=record

    codigo: int16;

    rubro: t\_rubro;

    precio: real;

  end;

  t\_registro\_producto2=record

    codigo: int16;

    precio: real;

  end;

  t\_lista\_productos=^t\_nodo\_productos;

  t\_nodo\_productos=record

    ele: t\_registro\_producto2;

    sig: t\_lista\_productos;

  end;

  t\_vector\_productos1=array[t\_rubro] of t\_lista\_productos;

  t\_vector\_productos2=array[1..productos\_rubro3\_total] of t\_registro\_producto2;

procedure inicializar\_vector\_productos1(var vector\_productos1: t\_vector\_productos1);

var

  i: t\_rubro;

begin

  for i:= rubro\_ini to rubro\_fin do

    vector\_productos1[i]:=nil;

end;

procedure leer\_producto(var registro\_producto1: t\_registro\_producto1);

var

  i: int8;

begin

  i:=random(100);

  if (i=0) then

    registro\_producto1.precio:=precio\_salida

  else

    registro\_producto1.precio:=1+random(100);

  if (registro\_producto1.precio<>precio\_salida) then

  begin

    registro\_producto1.codigo:=1+random(high(int16));

    registro\_producto1.rubro:=rubro\_ini+random(rubro\_fin);

  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.precio:=registro\_producto1.precio;

end;

procedure agregar\_ordenado\_lista\_productos(var lista\_productos: t\_lista\_productos; registro\_producto1: t\_registro\_producto1);

var

  anterior, actual, nuevo: t\_lista\_productos;

begin

  new(nuevo);

  cargar\_registro\_producto2(nuevo^.ele,registro\_producto1);

  anterior:=lista\_productos; actual:=lista\_productos;

  while ((actual<>nil) and (actual^.ele.codigo<nuevo^.ele.codigo)) do

  begin

    anterior:=actual;

    actual:=actual^.sig;

  end;

  if (actual=lista\_productos) then

    lista\_productos:=nuevo

  else

    anterior^.sig:=nuevo;

  nuevo^.sig:=actual;

end;

procedure cargar\_vector\_productos1(var vector\_productos1: t\_vector\_productos1);

var

  registro\_producto1: t\_registro\_producto1;

begin

  leer\_producto(registro\_producto1);

  while (registro\_producto1.precio<>precio\_salida) do

  begin

    agregar\_ordenado\_lista\_productos(vector\_productos1[registro\_producto1.rubro],registro\_producto1);

    leer\_producto(registro\_producto1);

  end;

end;

procedure imprimir\_registro\_producto2(registro\_producto2: t\_registro\_producto2; rubro: t\_rubro; producto: int16);

begin

  textcolor(green); write('El código de producto del producto '); textcolor(yellow); write(producto); textcolor(green); write(' del código de rubro '); textcolor(yellow); write(rubro); textcolor(green); write(' es '); textcolor(red); writeln(registro\_producto2.codigo);

  textcolor(green); write('El precio del producto '); textcolor(yellow); write(producto); textcolor(green); write(' del código de rubro '); textcolor(yellow); write(rubro); textcolor(green); write(' es '); textcolor(red); writeln(registro\_producto2.precio:0:2);

end;

procedure imprimir\_lista\_productos(lista\_productos: t\_lista\_productos; rubro: t\_rubro);

var

  i: int16;

begin

  i:=0;

  while (lista\_productos<>nil) do

  begin

    i:=i+1;

    imprimir\_registro\_producto2(lista\_productos^.ele,rubro,i);

    lista\_productos:=lista\_productos^.sig;

  end;

end;

procedure imprimir\_vector\_productos1(vector\_productos1: t\_vector\_productos1);

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\_lista\_productos(vector\_productos1[i],i);

    writeln();

  end;

end;

procedure cargar\_vector\_productos2(var vector\_productos2: t\_vector\_productos2; var productos\_rubro3: int8; lista\_productos: t\_lista\_productos);

begin

  while ((lista\_productos<>nil) and (productos\_rubro3<productos\_rubro3\_total)) do

  begin

    productos\_rubro3:=productos\_rubro3+1;

    vector\_productos2[productos\_rubro3]:=lista\_productos^.ele;

    lista\_productos:=lista\_productos^.sig;

  end;

end;

procedure imprimir\_vector\_productos2(vector\_productos2: t\_vector\_productos2; productos\_rubro3: int8);

var

  i: int8;

begin

  for i:= 1 to productos\_rubro3 do

  begin

    textcolor(green); write('La información del producto '); textcolor(yellow); write(i); textcolor(green); writeln(' del rubro 3 son:');

    imprimir\_registro\_producto2(vector\_productos2[i],3,i);

    writeln();

  end;

end;

procedure ordenar\_vector\_productos2(var vector\_productos2: t\_vector\_productos2; productos\_rubro3: int8);

var

  item: t\_registro\_producto2;

  i, j, k: int8;

begin

  for i:= 1 to (productos\_rubro3-1) do

  begin

    k:=i;

    for j:= (i+1) to productos\_rubro3 do

      if (vector\_productos2[j].precio<vector\_productos2[k].precio) then

        k:=j;

    item:=vector\_productos2[k];

    vector\_productos2[k]:=vector\_productos2[i];

    vector\_productos2[i]:=item;

  end;

end;

function calcular\_promedio\_vector\_productos2(vector\_productos2: t\_vector\_productos2; productos\_rubro3: int8): real;

var

  i: int8;

  precio\_total: real;

begin

  precio\_total:=0;

  for i:= 1 to productos\_rubro3 do

    precio\_total:=precio\_total+vector\_productos2[i].precio;

  calcular\_promedio\_vector\_productos2:=precio\_total/productos\_rubro3;

end;

var

  vector\_productos1: t\_vector\_productos1;

  vector\_productos2: t\_vector\_productos2;

  productos\_rubro3: int8;

begin

  randomize;

  productos\_rubro3:=0;

  inicializar\_vector\_productos1(vector\_productos1);

  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();

  cargar\_vector\_productos1(vector\_productos1);

  writeln(); textcolor(red); writeln('INCISO (b):'); writeln();

  imprimir\_vector\_productos1(vector\_productos1);

  writeln(); textcolor(red); writeln('INCISO (c):'); writeln();

  cargar\_vector\_productos2(vector\_productos2,productos\_rubro3,vector\_productos1[3]);

  if (productos\_rubro3>0) then

  begin

    imprimir\_vector\_productos2(vector\_productos2,productos\_rubro3);

    writeln(); textcolor(red); writeln('INCISO (d):'); writeln();

    ordenar\_vector\_productos2(vector\_productos2,productos\_rubro3);

    writeln(); textcolor(red); writeln('INCISO (e):'); writeln();

    imprimir\_vector\_productos2(vector\_productos2,productos\_rubro3);

    writeln(); textcolor(red); writeln('INCISO (f):'); writeln();

    textcolor(green); write('El promedio de los precios del vector\_productos2 es '); textcolor(red); write(calcular\_promedio\_vector\_productos2(vector\_productos2,productos\_rubro3):0:2);

  end;

end.