Juan Menduiña

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_pelicu
la1);
    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);
  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_prod
ucto1);
    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.
```