Juan Menduiña

Trabajo Práctico N° 0:
Módulo Imperativo (Práctica Inicial).

Ejercicio 1.

Implementar un programa que procese la información de los alumnos de la Facultad de Informática.

(a) *Implementar un módulo que lea y retorne, en una estructura adecuada, la información de todos los alumnos. De cada alumno, se lee su apellido, número de alumno, año de ingreso, cantidad de materias aprobadas (a lo sumo, 36) y nota obtenida (sin contar los aplazos) en cada una de las materias aprobadas. La lectura finaliza cuando se ingresa el número de alumno 11111, el cual debe procesarse.*

```
program TP0_E1a;
{$codepage UTF8}
uses crt;
const
  materias_min=1; materias_max=36; nota_min=4; nota_max=10; numero_salida=11111;
materias_salida=0;
type
  t_str20=string[20];
  t_materias_totales=materias_min..materias_max;
  t_notas=nota_min..nota_max;
  t_vector_notas=array[t_materias_totales] of t_notas;
  t_registro_alumno1=record
    apellido: t_str20;
    numero: int32;
    anio_ingreso: int16;
    materias_aprobadas: int8;
    notas: t_vector_notas;
  end;
  t_lista_alumnos1=^t_nodo_alumnos1;
  t_nodo_alumnos1=record
    ele: t_registro_alumno1;
    sig: t_lista_alumnos1;
  end;
procedure cargar_registro_alumno1(var registro_alumno1: t_registro_alumno1);
var
  i: int8;
begin
  textcolor(green); write('Introducir apellido del alumno: ');
  textcolor(yellow); readln(registro_alumno1.apellido);
  textcolor(green); write('Introducir número del alumno: ');
  textcolor(yellow); readln(registro_alumno1.numero);
  textcolor(green); write('Introducir año de ingreso del alumno: ');
  textcolor(yellow); readln(registro_alumno1.anio_ingreso);
  textcolor(green); write('Introducir materias aprobadas del alumno: ');
  textcolor(yellow); readln(registro_alumno1.materias_aprobadas);
  if (registro_alumno1.materias_aprobadas<>materias_salida) then
  begin
    for i:= 1 to registro_alumno1.materias_aprobadas do
    begin
      textcolor(green); write('Introducir nota obtenida en la materia ', i, ': ');
      textcolor(yellow); readln(registro_alumno1.notas[i]);
    end;
  end;
end;
```

```
procedure agregar_adelante_lista_alumnos1(var lista_alumnos1: t_lista_alumnos1;
registro_alumno1: t_registro_alumno1);
var
  nueva: t_lista_alumnos1;
begin
  new(nueva);
  nueva^.ele:=registro_alumno1;
  nueva^.sig:=lista_alumnos1;
  lista_alumnos1:=nueva;
end;
procedure cargar_lista_alumnos1(var lista_alumnos1: t_lista_alumnos1);
var
  registro_alumno1: t_registro_alumno1;
begin
  repeat
    cargar_registro_alumno1(registro_alumno1);
    agregar_adelante_lista_alumnos1(lista_alumnos1,registro_alumno1);
  until (registro_alumno1.numero=numero_salida);
end;
var
  lista_alumnos1: t_lista_alumnos1;
begin
  lista_alumnos1:=nil;
  cargar_lista_alumnos1(lista_alumnos1);
end.
```

(b) *Implementar un módulo que reciba la estructura generada en el inciso (a) y retorne número de alumno y promedio de cada alumno.*

```
program TP0_E1b;
{$codepage UTF8}
uses crt;
const
  materias_min=1; materias_max=36; nota_min=4; nota_max=10; numero_salida=11111;
materias_salida=0;
type
  t_str20=string[20];
  t_materias_totales=materias_min..materias_max;
  t_notas=nota_min..nota_max;
  t_vector_notas=array[t_materias_totales] of t_notas;
  t_registro_alumno1=record
    apellido: t_str20;
    numero: int32;
    anio_ingreso: int16;
    materias_aprobadas: int8;
    notas: t_vector_notas;
  end;
  t_registro_alumno2=record
    numero: int32;
    promedio: real;
  end;
  t_lista_alumnos1=^t_nodo_alumnos1;
  t_lista_alumnos2=^t_nodo_alumnos2;
  t_nodo_alumnos1=record
    ele: t_registro_alumno1;
    sig: t_lista_alumnos1;
  end;
  t_nodo_alumnos2=record
    ele: t_registro_alumno2;
    sig: t_lista_alumnos2;
  end;
procedure cargar_registro_alumno1(var registro_alumno1: t_registro_alumno1);
```

```
var
  i: int8;
begin
  textcolor(green); write('Introducir apellido del alumno: ');
  textcolor(yellow); readln(registro_alumno1.apellido);
  textcolor(green); write('Introducir número del alumno: ');
  textcolor(yellow); readln(registro_alumno1.numero);
  textcolor(green); write('Introducir año de ingreso del alumno: ');
  textcolor(yellow); readln(registro_alumno1.anio_ingreso);
  textcolor(green); write('Introducir materias aprobadas del alumno: ');
  textcolor(yellow); readln(registro_alumno1.materias_aprobadas);
  if (registro_alumno1.materias_aprobadas<>materias_salida) then
  begin
    for i:= 1 to registro_alumno1.materias_aprobadas do
    begin
      textcolor(green); write('Introducir nota obtenida en la materia ', i, ': ');
      textcolor(yellow); readln(registro_alumno1.notas[i]);
    end;
  end;
end;
procedure agregar_adelante_lista_alumnos1(var lista_alumnos1: t_lista_alumnos1;
registro_alumno1: t_registro_alumno1);
var
  nueva: t_lista_alumnos1;
begin
  new(nueva);
  nueva^.ele:=registro_alumno1;
  nueva^.sig:=lista_alumnos1;
  lista_alumnos1:=nueva;
end;
procedure cargar_registro_alumno2(var registro_alumno2: t_registro_alumno2;
registro_alumno1: t_registro_alumno1);
var
  i: int8;
  suma: int16;
begin
  suma:=0;
  registro_alumno2.numero:=registro_alumno1.numero;
  if (registro_alumno1.materias_aprobadas<>materias_salida) then
  begin
    for i:= 1 to registro_alumno1.materias_aprobadas do
      suma:=suma+registro_alumno1.notas[i];
  registro_alumno2.promedio:=suma/registro_alumno1.materias_aprobadas;
  end
  else
    registro_alumno2.promedio:=suma;
end;
procedure agregar_adelante_lista_alumnos2(var lista_alumnos2: t_lista_alumnos2;
registro_alumno2: t_registro_alumno2);
var
  nueva: t_lista_alumnos2;
begin
  new(nueva);
  nueva^.ele:=registro_alumno2;
  nueva^.sig:=lista_alumnos2;
  lista_alumnos2:=nueva;
end;
procedure cargar_lista_alumnos1(var lista_alumnos1: t_lista_alumnos1);
var
  registro_alumno1: t_registro_alumno1;
begin
  repeat
    cargar_registro_alumno1(registro_alumno1);
    agregar_adelante_lista_alumnos1(lista_alumnos1,registro_alumno1);
  until (registro_alumno1.numero=numero_salida);
end;
```

```
procedure cargar_lista_alumnos2(var lista_alumnos2: t_lista_alumnos2; lista_alumnos1:
t_lista_alumnos1);
var
  registro_alumno2: t_registro_alumno2;
begin
  while (lista_alumnos1<>nil) do
  begin
    cargar_registro_alumno2(registro_alumno2,lista_alumnos1^.ele);
    agregar_adelante_lista_alumnos2(lista_alumnos2,registro_alumno2);
    lista_alumnos1:=lista_alumnos1^.sig;
    textcolor(green); write('El promedio del alumno '); textcolor(red);
write(lista_alumnos2^.ele.numero); textcolor(green); write(' es '); textcolor(red);
writeln(lista_alumnos2^.ele.promedio:0:2);
  end;
end;
var
  lista_alumnos1: t_lista_alumnos1;
  lista_alumnos2: t_lista_alumnos2;
begin
  lista_alumnos1:=nil; lista_alumnos2:=nil;
  cargar_lista_alumnos1(lista_alumnos1);
  cargar_lista_alumnos2(lista_alumnos2,lista_alumnos1);
end.
```

(c) *Analizar: ¿qué cambios requieren los incisos (a) y (b), si no se sabe de antemano la cantidad de materias aprobadas de cada alumno y si, además, se desean registrar los aplazos? ¿cómo puede diseñarse una solución modularizada que requiera la menor cantidad de cambios?*

```
program TP0_E1c;
{$codepage UTF8}
uses crt;
const
  nota_min=1; nota_max=10; nota_aprobado=4; numero_salida=11111; examenes_salida=0;
nota_salida=0;
type
  t_str20=string[20];
  t_notas=nota_min..nota_max;
  t_lista_notas=^t_nodo_notas;
  t_nodo_notas=record
    ele: t_notas;
    sig: t_lista_notas;
  end;
  t_registro_alumno1=record
    apellido: t_str20;
    numero: int32;
    anio_ingreso: int16;
    notas: t_lista_notas;
    examenes_rendidos: int16;
    materias_aprobadas: int8;
  end;
  t_registro_alumno2=record
    numero: int32;
    promedio_con_aplazos: real;
    promedio_sin_aplazos: real;
  end;
  t_lista_alumnos1=^t_nodo_alumnos1;
  t_lista_alumnos2=^t_nodo_alumnos2;
  t_nodo_alumnos1=record
    ele: t_registro_alumno1;
    sig: t_lista_alumnos1;
  end;
```

```
  t_nodo_alumnos2=record
    ele: t_registro_alumno2;
    sig: t_lista_alumnos2;
  end;
procedure agregar_adelante_lista_notas(var lista_notas: t_lista_notas; nota: t_notas);
var
  nueva: t_lista_notas;
begin
  new(nueva);
  nueva^.ele:=nota;
  nueva^.sig:=lista_notas;
  lista_notas:=nueva;
end;
procedure cargar_registro_alumno1(var registro_alumno1: t_registro_alumno1);
var
  nota: int8;
  materias_aprobadas: int8;
  examenes_rendidos: int16;
begin
  registro_alumno1.notas:=nil; examenes_rendidos:=0; materias_aprobadas:=0;
  textcolor(green); write('Introducir apellido del alumno: ');
  textcolor(yellow); readln(registro_alumno1.apellido);
  textcolor(green); write('Introducir número del alumno: ');
  textcolor(yellow); readln(registro_alumno1.numero);
  textcolor(green); write('Introducir año de ingreso del alumno: ');
  textcolor(yellow); readln(registro_alumno1.anio_ingreso);
  textcolor(green); write('Introducir nota obtenida en el examen ', examenes_rendidos+1, '
(nota de salida igual a 0): ');
  textcolor(yellow); readln(nota);
  while (nota<>nota_salida) do
  begin
    agregar_adelante_lista_notas(registro_alumno1.notas,nota);
    examenes_rendidos:=examenes_rendidos+1;
    if (nota>=nota_aprobado) then
      materias_aprobadas:=materias_aprobadas+1;
    textcolor(green); write('Introducir nota obtenida en el examen ', examenes_rendidos+1, '
(nota de salida igual a 0): ');
    textcolor(yellow); readln(nota);
  end;
  registro_alumno1.examenes_rendidos:=examenes_rendidos;
  registro_alumno1.materias_aprobadas:=materias_aprobadas;
end;
procedure agregar_adelante_lista_alumnos1(var lista_alumnos1: t_lista_alumnos1;
registro_alumno1: t_registro_alumno1);
var
  nueva: t_lista_alumnos1;
begin
  new(nueva);
  nueva^.ele:=registro_alumno1;
  nueva^.sig:=lista_alumnos1;
  lista_alumnos1:=nueva;
end;
procedure cargar_registro_alumno2(var registro_alumno2: t_registro_alumno2;
registro_alumno1: t_registro_alumno1);
var
  suma_con_aplazos, suma_sin_aplazos: int16;
begin
  suma_con_aplazos:=0; suma_sin_aplazos:=0;
  registro_alumno2.numero:=registro_alumno1.numero;
  if (registro_alumno1.examenes_rendidos<>examenes_salida) then
  begin
    while (registro_alumno1.notas<>nil) do
    begin
      suma_con_aplazos:=suma_con_aplazos+registro_alumno1.notas^.ele;
      if (registro_alumno1.notas^.ele>=nota_aprobado) then
        suma_sin_aplazos:=suma_sin_aplazos+registro_alumno1.notas^.ele;
```

```
      registro_alumno1.notas:=registro_alumno1.notas^.sig;
    end;
    registro_alumno2.promedio_con_aplazos:=suma_con_aplazos/registro_alumno1.examenes_rendid
os;
    registro_alumno2.promedio_sin_aplazos:=suma_sin_aplazos/registro_alumno1.materias_aproba
das;
  end
  else
  begin
    registro_alumno2.promedio_con_aplazos:=suma_con_aplazos;
    registro_alumno2.promedio_sin_aplazos:=suma_sin_aplazos;
  end;
end;
procedure agregar_adelante_lista_alumnos2(var lista_alumnos2: t_lista_alumnos2;
registro_alumno2: t_registro_alumno2);
var
  nueva: t_lista_alumnos2;
begin
  new(nueva);
  nueva^.ele:=registro_alumno2;
  nueva^.sig:=lista_alumnos2;
  lista_alumnos2:=nueva;
end;
procedure cargar_lista_alumnos1(var lista_alumnos1: t_lista_alumnos1);
var
  registro_alumno1: t_registro_alumno1;
begin
  repeat
    cargar_registro_alumno1(registro_alumno1);
    agregar_adelante_lista_alumnos1(lista_alumnos1,registro_alumno1);
  until (registro_alumno1.numero=numero_salida);
end;
procedure cargar_lista_alumnos2(var lista_alumnos2: t_lista_alumnos2; lista_alumnos1:
t_lista_alumnos1);
var
  registro_alumno2: t_registro_alumno2;
begin
  while (lista_alumnos1<>nil) do
  begin
    cargar_registro_alumno2(registro_alumno2,lista_alumnos1^.ele);
    agregar_adelante_lista_alumnos2(lista_alumnos2,registro_alumno2);
    lista_alumnos1:=lista_alumnos1^.sig;
    textcolor(green); write('El promedio CON aplazos del alumno '); textcolor(red);
write(lista_alumnos2^.ele.numero); textcolor(green); write(' es '); textcolor(red);
writeln(lista_alumnos2^.ele.promedio_con_aplazos:0:2);
    textcolor(green); write('El promedio SIN aplazos del alumno '); textcolor(red);
write(lista_alumnos2^.ele.numero); textcolor(green); write(' es '); textcolor(red);
writeln(lista_alumnos2^.ele.promedio_sin_aplazos:0:2);
  end;
end;
var
  lista_alumnos1: t_lista_alumnos1;
  lista_alumnos2: t_lista_alumnos2;
begin
  lista_alumnos1:=nil; lista_alumnos2:=nil;
  cargar_lista_alumnos1(lista_alumnos1);
  cargar_lista_alumnos2(lista_alumnos2,lista_alumnos1);
end.
```

Ejercicio 2.

Implementar un programa que procese información de propiedades que están a la venta en una inmobiliaria.

(a) *Implementar un módulo para almacenar, en una estructura adecuada, las propiedades agrupadas por zona. Las propiedades de una misma zona deben quedar almacenadas ordenadas por tipo de propiedad. Para cada propiedad, debe almacenarse el código, el tipo de propiedad y el precio total. De cada propiedad, se lee: zona (1 a 5), código de propiedad, tipo de propiedad, cantidad de metros cuadrados y precio del metro cuadrado. La lectura finaliza cuando se ingresa el precio del metro cuadrado -1.*

```
program TP0_E2a;
{$codepage UTF8}
uses crt;
const
  zona_min=1; zona_max=5; tipo_min=1; tipo_max=3; preciom2_salida=-1;
type
  t_zonas=zona_min..zona_max;
  t_tipos=tipo_min..tipo_max;
  t_registro_propiedad=record
    codigo: int16;
    tipo: t_tipos;
    precio_total: real;
  end;
  t_lista_propiedades=^t_nodo_propiedades;
  t_nodo_propiedades=record
    ele: t_registro_propiedad;
    sig: t_lista_propiedades;
  end;
  t_vector_propiedades=array[t_zonas] of t_lista_propiedades;
procedure inicializar_vector_propiedades(var vector_propiedades: t_vector_propiedades);
var
  i: t_zonas;
begin
  for i:= zona_min to zona_max do
    vector_propiedades[i]:=nil;
end;
procedure cargar_registro_propiedad(var registro_propiedad: t_registro_propiedad; preciom2:
real);
var
  m2: real;
begin
  textcolor(green); write('Introducir código de la propiedad: ');
  textcolor(yellow); readln(registro_propiedad.codigo);
  textcolor(green); write('Introducir tipo de propiedad (1: Casa, 2: Departamento, 3: Lote):
');
  textcolor(yellow); readln(registro_propiedad.tipo);
  textcolor(green); write('Introducir cantidad de metros cuadrados de la propiedad: ');
  textcolor(yellow); readln(m2);
  registro_propiedad.precio_total:=m2*preciom2;
end;
procedure agregar_ordenado_lista_propiedades(var lista_propiedades: t_lista_propiedades;
registro_propiedad: t_registro_propiedad);
var
  anterior, actual, nueva: t_lista_propiedades;
begin
  new(nueva); nueva^.ele:=registro_propiedad;
  anterior:=lista_propiedades; actual:=lista_propiedades;
  while ((actual<>nil) and (actual^.ele.tipo<nueva^.ele.tipo)) do
  begin
```

```
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=lista_propiedades) then
    lista_propiedades:=nueva
  else
    anterior^.sig:=nueva;
  nueva^.sig:=actual;
end;
procedure cargar_vector_propiedades(var vector_propiedades: t_vector_propiedades);
var
  registro_propiedad: t_registro_propiedad;
  zona: t_zonas;
  preciom2: real;
begin
  textcolor(green); write('Introducir precio del metro cuadrado de la propiedad: ');
  textcolor(yellow); readln(preciom2);
  while (preciom2<>preciom2_salida) do
  begin
    textcolor(green); write('Introducir zona de la propiedad (1 a 5): ');
    textcolor(yellow); readln(zona);
    cargar_registro_propiedad(registro_propiedad,preciom2);
    agregar_ordenado_lista_propiedades(vector_propiedades[zona],registro_propiedad);
    textcolor(green); write('Introducir precio del metro cuadrado de la propiedad: ');
    textcolor(yellow); readln(preciom2);
  end;
end;
var
  vector_propiedades: t_vector_propiedades;
begin
  inicializar_vector_propiedades(vector_propiedades);
  cargar_vector_propiedades(vector_propiedades);
end.
```

(b) *Implementar un módulo que reciba la estructura generada en (a), un número de zona y un tipo de propiedad y retorne los códigos de las propiedades de la zona recibida y del tipo recibido.*

```
program TP0_E2b;
{$codepage UTF8}
uses crt;
const
  zona_min=1; zona_max=5; tipo_min=1; tipo_max=3; preciom2_salida=-1;
type
  t_zonas=zona_min..zona_max;
  t_tipos=tipo_min..tipo_max;
  t_registro_propiedad=record
    codigo: int16;
    tipo: t_tipos;
    precio_total: real;
  end;
  t_lista_propiedades=^t_nodo_propiedades;
  t_nodo_propiedades=record
    ele: t_registro_propiedad;
    sig: t_lista_propiedades;
  end;
  t_vector_propiedades=array[t_zonas] of t_lista_propiedades;
procedure inicializar_vector_propiedades(var vector_propiedades: t_vector_propiedades);
var
  i: t_zonas;
begin
  for i:= zona_min to zona_max do
```

```
    vector_propiedades[i]:=nil;
end;
procedure cargar_registro_propiedad(var registro_propiedad: t_registro_propiedad; preciom2:
real);
var
  m2: real;
begin
  textcolor(green); write('Introducir código de la propiedad: ');
  textcolor(yellow); readln(registro_propiedad.codigo);
  textcolor(green); write('Introducir tipo de propiedad (1: Casa, 2: Departamento, 3: Lote):
');
  textcolor(yellow); readln(registro_propiedad.tipo);
  textcolor(green); write('Introducir cantidad de metros cuadrados de la propiedad: ');
  textcolor(yellow); readln(m2);
  registro_propiedad.precio_total:=m2*preciom2;
end;
procedure agregar_ordenado_lista_propiedades(var lista_propiedades: t_lista_propiedades;
registro_propiedad: t_registro_propiedad);
var
  anterior, actual, nueva: t_lista_propiedades;
begin
  new(nueva); nueva^.ele:=registro_propiedad;
  anterior:=lista_propiedades; actual:=lista_propiedades;
  while ((actual<>nil) and (actual^.ele.tipo<nueva^.ele.tipo)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=lista_propiedades) then
    lista_propiedades:=nueva
  else
    anterior^.sig:=nueva;
  nueva^.sig:=actual;
end;
procedure cargar_vector_propiedades(var vector_propiedades: t_vector_propiedades);
var
  registro_propiedad: t_registro_propiedad;
  zona: t_zonas;
  preciom2: real;
begin
  textcolor(green); write('Introducir precio del metro cuadrado de la propiedad: ');
  textcolor(yellow); readln(preciom2);
  while (preciom2<>preciom2_salida) do
  begin
    textcolor(green); write('Introducir zona de la propiedad (1 a 5): ');
    textcolor(yellow); readln(zona);
    cargar_registro_propiedad(registro_propiedad,preciom2);
    agregar_ordenado_lista_propiedades(vector_propiedades[zona],registro_propiedad);
    textcolor(green); write('Introducir precio del metro cuadrado de la propiedad: ');
    textcolor(yellow); readln(preciom2);
  end;
end;
procedure buscar_codigos_propiedades(vector_propiedades: t_vector_propiedades; zona:
t_zonas; tipo: t_tipos);
var
  i: int16;
begin
  i:=1;
  while ((vector_propiedades[zona]<>nil) and (vector_propiedades[zona]^.ele.tipo=tipo)) do
  begin
    textcolor(green); write('El código de la propiedad '); textcolor(red); write(i);
textcolor(green); write(' es '); textcolor(red);
writeln(vector_propiedades[zona]^.ele.codigo);
    vector_propiedades[zona]:=vector_propiedades[zona]^.sig;
    i:=i+1;
  end;
```

```
end;
var
  zona: t_zonas;
  tipo: t_tipos;
  vector_propiedades: t_vector_propiedades;
begin
  inicializar_vector_propiedades(vector_propiedades);
  cargar_vector_propiedades(vector_propiedades);
  textcolor(green); write('Introducir zona de la propiedad que se desea buscar (1 a 5): ');
  textcolor(yellow); readln(zona);
  textcolor(green); write('Introducir tipo de propiedad que se desea buscar (1: Casa, 2:
Departamento, 3: Lote): ');
  textcolor(yellow); readln(tipo);
  buscar_codigos_propiedades(vector_propiedades,zona,tipo);
end.
```

Ejercicio 3.

Implementar un programa que procese las ventas de un supermercado. El supermercado dispone de una tabla con los precios y stocks de los 1.000 productos que tiene a la venta.

(a) *Implementar un módulo que retorne, en una estructura de datos adecuada, los tickets de las ventas. De cada venta, se lee código de venta y los productos vendidos. Las ventas finalizan con el código de venta -1. De cada producto, se lee código y cantidad de unidades solicitadas. Para cada venta, la lectura de los productos a vender finaliza con cantidad de unidades vendidas igual a 0. El ticket debe contener:*

- *Código de venta.*
- *Detalle (código de producto, cantidad y precio unitario) de los productos que se pudieron vender. En caso de no haber stock suficiente, se venderá la máxima cantidad posible.*
- *Monto total de la venta.*

```
program TP0_E3a;
{$codepage UTF8}
uses crt;
const
  productos_min=1; productos_max=1000; codigo_venta_salida=-1; ventas_salida=0;
type
  t_productos=productos_min..productos_max;
  t_registro_producto=record
    codigo_producto: int16;
    cantidad: int16;
    precio: real;
  end;
  t_lista_productos=^t_nodo_productos;
  t_nodo_productos=record
    ele: t_registro_producto;
    sig: t_lista_productos;
  end;
  t_registro_venta=record
    codigo_venta: int16;
    productos: t_lista_productos;
    monto_total: real;
  end;
  t_lista_ventas=^t_nodo_ventas;
  t_nodo_ventas=record
    ele: t_registro_venta;
    sig: t_lista_ventas;
  end;
  t_vector_productos=array[t_productos] of t_registro_producto;
procedure cargar_vector_productos(var vector_productos: t_vector_productos);
var
  i: int16;
begin
  for i:= productos_min to productos_max do
  begin
    vector_productos[i].codigo_producto:=i;
    vector_productos[i].cantidad:=random(10000);
    vector_productos[i].precio:=random(100000);
  end;
end;
function buscar_vector_productos(vector_productos: t_vector_productos; codigo_producto:
int16): t_productos;
var
```

```
  pos: t_productos;
begin
  pos:=1;
  while (vector_productos[pos].codigo_producto<>codigo_producto) do
    pos:=pos+1;
  buscar_vector_productos:=pos;
end;
procedure chequear_stock_vector_productos(var vector_productos: t_vector_productos; var
registro_producto: t_registro_producto; pos: t_productos);
begin
  if (registro_producto.cantidad<vector_productos[pos].cantidad) then
    vector_productos[pos].cantidad:=vector_productos[pos].cantidad-
registro_producto.cantidad
  else
  begin
    registro_producto.cantidad:=vector_productos[pos].cantidad;
    vector_productos[pos].cantidad:=0;
  end;
end;
procedure cargar_registro_producto(var registro_producto: t_registro_producto; var
vector_productos: t_vector_productos; var monto_total: real);
var
  pos: t_productos;
begin
  textcolor(green); write('Introducir código del produto: ');
  textcolor(yellow); readln(registro_producto.codigo_producto);
  textcolor(green); write('Introducir cantidad solicitada del producto: ');
  textcolor(yellow); readln(registro_producto.cantidad);
  pos:=buscar_vector_productos(vector_productos,registro_producto.codigo_producto);
  chequear_stock_vector_productos(vector_productos,registro_producto,pos);
  if (registro_producto.cantidad<>ventas_salida) then
  begin
    registro_producto.precio:=vector_productos[pos].precio;
    monto_total:=monto_total+registro_producto.precio*registro_producto.cantidad;
  end;
end;
procedure agregar_adelante_lista_productos(var lista_productos: t_lista_productos;
registro_producto: t_registro_producto);
var
  nueva: t_lista_productos;
begin
  new(nueva);
  nueva^.ele:=registro_producto;
  nueva^.sig:=lista_productos;
  lista_productos:=nueva;
end;
procedure cargar_lista_productos(var lista_productos: t_lista_productos; var
vector_productos: t_vector_productos; var monto_total: real);
var
  registro_producto: t_registro_producto;
begin
  cargar_registro_producto(registro_producto,vector_productos,monto_total);
  while (registro_producto.cantidad<>ventas_salida) do
  begin
    agregar_adelante_lista_productos(lista_productos,registro_producto);
    cargar_registro_producto(registro_producto,vector_productos,monto_total);
  end;
end;
procedure cargar_registro_venta(var registro_venta: t_registro_venta; var vector_productos:
t_vector_productos);
var
  monto_total: real;
begin
  registro_venta.productos:=nil; monto_total:=0;
  textcolor(green); write('Introducir código de la venta: ');
  textcolor(yellow); readln(registro_venta.codigo_venta);
```

```
  if (registro_venta.codigo_venta<>codigo_venta_salida) then
  begin
    cargar_lista_productos(registro_venta.productos,vector_productos,monto_total);
    registro_venta.monto_total:=monto_total;
  end;
end;
procedure agregar_adelante_lista_ventas(var lista_ventas: t_lista_ventas; registro_venta:
t_registro_venta);
var
  nueva: t_lista_ventas;
begin
  new(nueva);
  nueva^.ele:=registro_venta;
  nueva^.sig:=lista_ventas;
  lista_ventas:=nueva;
end;
procedure cargar_lista_ventas(var lista_ventas: t_lista_ventas; vector_productos:
t_vector_productos);
var
  registro_venta: t_registro_venta;
begin
  cargar_registro_venta(registro_venta,vector_productos);
  while (registro_venta.codigo_venta<>codigo_venta_salida) do
  begin
    agregar_adelante_lista_ventas(lista_ventas,registro_venta);
    cargar_registro_venta(registro_venta,vector_productos);
  end;
end;
var
  vector_productos: t_vector_productos;
  lista_ventas: t_lista_ventas;
begin
  randomize;
  cargar_vector_productos(vector_productos); lista_ventas:=nil;
  cargar_lista_ventas(lista_ventas,vector_productos);
end.
```

(b) *Implementar un módulo que reciba la estructura generada en el inciso (a) y un código de producto y retorne la cantidad de unidades vendidas de ese código de producto.*

```
program TP0_E3b;
{$codepage UTF8}
uses crt;
const
  productos_min=1; productos_max=1000; codigo_venta_salida=-1; ventas_salida=0;
type
  t_productos=productos_min..productos_max;
  t_registro_producto=record
    codigo_producto: int16;
    cantidad: int16;
    precio: real;
  end;
  t_lista_productos=^t_nodo_productos;
  t_nodo_productos=record
    ele: t_registro_producto;
    sig: t_lista_productos;
  end;
  t_registro_venta=record
    codigo_venta: int16;
    productos: t_lista_productos;
    monto_total: real;
  end;
```

```
  t_lista_ventas=^t_nodo_ventas;
  t_nodo_ventas=record
    ele: t_registro_venta;
    sig: t_lista_ventas;
  end;
  t_vector_productos=array[t_productos] of t_registro_producto;
procedure cargar_vector_productos(var vector_productos: t_vector_productos);
var
  i: int16;
begin
  for i:= productos_min to productos_max do
  begin
    vector_productos[i].codigo_producto:=i;
    vector_productos[i].cantidad:=random(10000);
    vector_productos[i].precio:=random(100000);
  end;
end;
function buscar_vector_productos(vector_productos: t_vector_productos; codigo_producto:
int16): t_productos;
var
  pos: t_productos;
begin
  pos:=1;
  while (vector_productos[pos].codigo_producto<>codigo_producto) do
    pos:=pos+1;
  buscar_vector_productos:=pos;
end;
procedure chequear_stock_vector_productos(var vector_productos: t_vector_productos; var
registro_producto: t_registro_producto; pos: t_productos);
begin
  if (registro_producto.cantidad<vector_productos[pos].cantidad) then
    vector_productos[pos].cantidad:=vector_productos[pos].cantidad-
registro_producto.cantidad
  else
  begin
    registro_producto.cantidad:=vector_productos[pos].cantidad;
    vector_productos[pos].cantidad:=0;
  end;
end;
procedure cargar_registro_producto(var registro_producto: t_registro_producto; var
vector_productos: t_vector_productos; var monto_total: real);
var
  pos: t_productos;
begin
  textcolor(green); write('Introducir código del produto: ');
  textcolor(yellow); readln(registro_producto.codigo_producto);
  textcolor(green); write('Introducir cantidad solicitada del producto: ');
  textcolor(yellow); readln(registro_producto.cantidad);
  pos:=buscar_vector_productos(vector_productos,registro_producto.codigo_producto);
  chequear_stock_vector_productos(vector_productos,registro_producto,pos);
  if (registro_producto.cantidad<>ventas_salida) then
  begin
    registro_producto.precio:=vector_productos[pos].precio;
    monto_total:=monto_total+registro_producto.precio*registro_producto.cantidad;
  end;
end;
procedure agregar_adelante_lista_productos(var lista_productos: t_lista_productos;
registro_producto: t_registro_producto);
var
  nueva: t_lista_productos;
begin
  new(nueva);
  nueva^.ele:=registro_producto;
  nueva^.sig:=lista_productos;
  lista_productos:=nueva;
end;
```

```
procedure cargar_lista_productos(var lista_productos: t_lista_productos; var
vector_productos: t_vector_productos; var monto_total: real);
var
  registro_producto: t_registro_producto;
begin
  cargar_registro_producto(registro_producto,vector_productos,monto_total);
  while (registro_producto.cantidad<>ventas_salida) do
  begin
    agregar_adelante_lista_productos(lista_productos,registro_producto);
    cargar_registro_producto(registro_producto,vector_productos,monto_total);
  end;
end;
procedure cargar_registro_venta(var registro_venta: t_registro_venta; var vector_productos:
t_vector_productos);
var
  monto_total: real;
begin
  registro_venta.productos:=nil; monto_total:=0;
  textcolor(green); write('Introducir código de la venta: ');
  textcolor(yellow); readln(registro_venta.codigo_venta);
  if (registro_venta.codigo_venta<>codigo_venta_salida) then
  begin
    cargar_lista_productos(registro_venta.productos,vector_productos,monto_total);
    registro_venta.monto_total:=monto_total;
  end;
end;
procedure agregar_adelante_lista_ventas(var lista_ventas: t_lista_ventas; registro_venta:
t_registro_venta);
var
  nueva: t_lista_ventas;
begin
  new(nueva);
  nueva^.ele:=registro_venta;
  nueva^.sig:=lista_ventas;
  lista_ventas:=nueva;
end;
procedure cargar_lista_ventas(var lista_ventas: t_lista_ventas; vector_productos:
t_vector_productos);
var
  registro_venta: t_registro_venta;
begin
  cargar_registro_venta(registro_venta,vector_productos);
  while (registro_venta.codigo_venta<>codigo_venta_salida) do
  begin
    agregar_adelante_lista_ventas(lista_ventas,registro_venta);
    cargar_registro_venta(registro_venta,vector_productos);
  end;
end;
function buscar_ventas_producto(lista_ventas: t_lista_ventas; codigo_producto: int16):
int16;
var
  ventas: int16;
begin
  ventas:=0;
  while (lista_ventas<>nil) do
  begin
    while (lista_ventas^.ele.productos<>nil) do
    begin
      if (lista_ventas^.ele.productos^.ele.codigo_producto=codigo_producto) then
        ventas:=ventas+lista_ventas^.ele.productos^.ele.cantidad;
      lista_ventas^.ele.productos:=lista_ventas^.ele.productos^.sig;
    end;
    lista_ventas:=lista_ventas^.sig;
  end;
  buscar_ventas_producto:=ventas;
end;
```

```
var
  vector_productos: t_vector_productos;
  lista_ventas: t_lista_ventas;
  codigo_producto, ventas: int16;
begin
  randomize;
  cargar_vector_productos(vector_productos); lista_ventas:=nil; ventas:=0;
  cargar_lista_ventas(lista_ventas,vector_productos);
  textcolor(green); write('Introducir código del producto que se desea buscar: ');
  textcolor(yellow); readln(codigo_producto);
  ventas:=buscar_ventas_producto(lista_ventas,codigo_producto);
  textcolor(green); write('La cantidad de unidades vendidas del producto '); textcolor(red);
write(codigo_producto); textcolor(green); write(' es '); textcolor(red); write(ventas);
end.
```