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.*

(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_E1ab;
{$codepage UTF8}
uses crt;
const
  anio_ini=2000; anio_fin=2023;
  materias_total=36;
  nota_ini=4; nota_fin=10;
  numero_salida=11111;
type
  t_materia=1..materias_total;
  t_nota=nota_ini..nota_fin;
  t_vector_notas=array[t_materia] of t_nota;
  t_registro_alumno1=record
    apellido: string;
    numero: int16;
    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_nodo_alumnos1=record
    ele: t_registro_alumno1;
    sig: t_lista_alumnos1;
  end;
  t_lista_alumnos2=^t_nodo_alumnos2;
  t_nodo_alumnos2=record
    ele: t_registro_alumno2;
    sig: t_lista_alumnos2;
  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_alumno(var registro_alumno1: t_registro_alumno1);
var
  i: int8;
begin
  registro_alumno1.apellido:=random_string(5+random(6));
  i:=random(100);
  if (i=0) then
    registro_alumno1.numero:=numero_salida
  else
    registro_alumno1.numero:=1+random(high(int16));
  registro_alumno1.anio_ingreso:=anio_ini+random(anio_fin-anio_ini+1);
  registro_alumno1.materias_aprobadas:=random(materias_total+1);
  for i:= 1 to registro_alumno1.materias_aprobadas do
    registro_alumno1.notas[i]:=nota_ini+random(nota_fin-nota_ini+1);
end;
procedure agregar_adelante_lista_alumnos1(var lista_alumnos1: t_lista_alumnos1;
registro_alumno1: t_registro_alumno1);
var
  nuevo: t_lista_alumnos1;
begin
  new(nuevo);
  nuevo^.ele:=registro_alumno1;
  nuevo^.sig:=lista_alumnos1;
  lista_alumnos1:=nuevo;
end;
procedure cargar_lista_alumnos1(var lista_alumnos1: t_lista_alumnos1);
var
  registro_alumno1: t_registro_alumno1;
begin
  repeat
    leer_alumno(registro_alumno1);
    agregar_adelante_lista_alumnos1(lista_alumnos1,registro_alumno1);
  until (registro_alumno1.numero=numero_salida);
end;
procedure imprimir_registro_alumno1(registro_alumno1: t_registro_alumno1; alumno: int16);
begin
  textcolor(green); write('El apellido del alumno '); textcolor(yellow); write(alumno);
textcolor(green); write(' es '); textcolor(red); writeln(registro_alumno1.apellido);
  textcolor(green); write('El número de alumno del alumno '); textcolor(yellow);
write(alumno); textcolor(green); write(' es '); textcolor(red);
writeln(registro_alumno1.numero);
  textcolor(green); write('El año de ingreso del alumno '); textcolor(yellow); write(alumno);
textcolor(green); write(' es '); textcolor(red); writeln(registro_alumno1.anio_ingreso);
  textcolor(green); write('La cantidad de materias aprobadas del alumno '); textcolor(yellow);
write(alumno); textcolor(green); write(' es '); textcolor(red);
writeln(registro_alumno1.materias_aprobadas);
end;
procedure imprimir_lista_alumnos1(lista_alumnos1: t_lista_alumnos1);
var
  i: int16;
begin
  i:=0;
  while (lista_alumnos1<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información del alumno '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_alumno1(lista_alumnos1^.ele,i);
    writeln();
    lista_alumnos1:=lista_alumnos1^.sig;
  end;
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<>0) 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
  nuevo: t_lista_alumnos2;
begin
  new(nuevo);
  nuevo^.ele:=registro_alumno2;
  nuevo^.sig:=lista_alumnos2;
  lista_alumnos2:=nuevo;
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;
  end;
end;
procedure imprimir_registro_alumno2(registro_alumno2: t_registro_alumno2; alumno: int16);
begin
  textcolor(green); write('El número de alumno del alumno '); textcolor(yellow);
write(alumno); textcolor(green); write(' es '); textcolor(red);
writeln(registro_alumno2.numero);
  textcolor(green); write('El promedio del alumno '); textcolor(yellow); write(alumno);
textcolor(green); write(' es '); textcolor(red); writeln(registro_alumno2.promedio:0:2);
end;
procedure imprimir_lista_alumnos2(lista_alumnos2: t_lista_alumnos2);
var
  i: int16;
begin
  i:=0;
  while (lista_alumnos2<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información del alumno '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_alumno2(lista_alumnos2^.ele,i);
    writeln();
    lista_alumnos2:=lista_alumnos2^.sig;
  end;
end;
var
  lista_alumnos1: t_lista_alumnos1;
  lista_alumnos2: t_lista_alumnos2;
begin
  randomize;
  lista_alumnos1:=nil;
```

```
  lista_alumnos2:=nil;
  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
  cargar_lista_alumnos1(lista_alumnos1);
  imprimir_lista_alumnos1(lista_alumnos1);
  writeln(); textcolor(red); writeln('INCISO (b):'); writeln();
  cargar_lista_alumnos2(lista_alumnos2,lista_alumnos1);
  imprimir_lista_alumnos2(lista_alumnos2);
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
  anio_ini=2000; anio_fin=2023;
  nota_ini=1; nota_fin=10;
  nota_corte=4; nota_salida=0;
  numero_salida=11111;
type
  t_anio=anio_ini..anio_fin;
  t_nota=nota_ini..nota_fin;
  t_lista_notas=^t_nodo_notas;
  t_nodo_notas=record
    ele: t_nota;
    sig: t_lista_notas;
  end;
  t_registro_alumno1=record
    apellido: string;
    numero: int32;
    anio_ingreso: t_anio;
    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_nodo_alumnos1=record
    ele: t_registro_alumno1;
    sig: t_lista_alumnos1;
  end;
  t_lista_alumnos2=^t_nodo_alumnos2;
  t_nodo_alumnos2=record
    ele: t_registro_alumno2;
    sig: t_lista_alumnos2;
  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 agregar_adelante_lista_notas(var lista_notas: t_lista_notas; nota: t_nota);
var
  nuevo: t_lista_notas;
begin
  new(nuevo);
  nuevo^.ele:=nota;
  nuevo^.sig:=lista_notas;
  lista_notas:=nuevo;
end;
procedure leer_nota(var nota: int8);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    nota:=nota_salida
  else
    nota:=nota_ini+random(nota_fin);
end;
procedure leer_alumno(var registro_alumno1: t_registro_alumno1);
var
  nota: int8;
  materias_aprobadas, i: int8;
  examenes_rendidos: int16;
begin
  registro_alumno1.notas:=nil;
  examenes_rendidos:=0; materias_aprobadas:=0;
  registro_alumno1.apellido:=random_string(5+random(6));
  i:=random(100);
  if (i=0) then
    registro_alumno1.numero:=numero_salida
  else
    registro_alumno1.numero:=1+random(high(int16));
  registro_alumno1.anio_ingreso:=anio_ini+random(anio_fin-anio_ini+1);
  leer_nota(nota);
  while (nota<>nota_salida) do
  begin
    agregar_adelante_lista_notas(registro_alumno1.notas,nota);
    examenes_rendidos:=examenes_rendidos+1;
    if (nota>=nota_corte) then
      materias_aprobadas:=materias_aprobadas+1;
    leer_nota(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
  nuevo: t_lista_alumnos1;
begin
  new(nuevo);
  nuevo^.ele:=registro_alumno1;
  nuevo^.sig:=lista_alumnos1;
  lista_alumnos1:=nuevo;
end;
procedure cargar_lista_alumnos1(var lista_alumnos1: t_lista_alumnos1);
var
  registro_alumno1: t_registro_alumno1;
begin
  repeat
    leer_alumno(registro_alumno1);
    agregar_adelante_lista_alumnos1(lista_alumnos1,registro_alumno1);
  until (registro_alumno1.numero=numero_salida);
end;
```

```
procedure imprimir_registro_alumno1(registro_alumno1: t_registro_alumno1; alumno: int16);
begin
  textcolor(green); write('El apellido del alumno '); textcolor(yellow); write(alumno);
textcolor(green); write(' es '); textcolor(red); writeln(registro_alumno1.apellido);
  textcolor(green); write('El número de alumno del alumno '); textcolor(yellow);
write(alumno); textcolor(green); write(' es '); textcolor(red);
writeln(registro_alumno1.numero);
  textcolor(green); write('El año de ingreso del alumno '); textcolor(yellow); write(alumno);
textcolor(green); write(' es '); textcolor(red); writeln(registro_alumno1.anio_ingreso);
  textcolor(green); write('La cantidad de exámenes rendidos del alumno '); textcolor(yellow);
write(alumno); textcolor(green); write(' es '); textcolor(red);
writeln(registro_alumno1.examenes_rendidos);
  textcolor(green); write('La cantidad de materias aprobadas del alumno '); textcolor(yellow);
write(alumno); textcolor(green); write(' es '); textcolor(red);
writeln(registro_alumno1.materias_aprobadas);
end;
procedure imprimir_lista_alumnos1(lista_alumnos1: t_lista_alumnos1);
var
  i: int16;
begin
  i:=0;
  while (lista_alumnos1<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información del alumno '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_alumno1(lista_alumnos1^.ele,i);
    writeln();
    lista_alumnos1:=lista_alumnos1^.sig;
  end;
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<>0) 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_corte) 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_rendidos
;
    if (registro_alumno1.materias_aprobadas<>0) then
      registro_alumno2.promedio_sin_aplazos:=suma_sin_aplazos/registro_alumno1.materias_aproba
das
    else
      registro_alumno2.promedio_sin_aplazos:=suma_sin_aplazos;
  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
  nuevo: t_lista_alumnos2;
begin
```

```
  new(nuevo);
  nuevo^.ele:=registro_alumno2;
  nuevo^.sig:=lista_alumnos2;
  lista_alumnos2:=nuevo;
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;
  end;
end;
procedure imprimir_registro_alumno2(registro_alumno2: t_registro_alumno2; alumno: int16);
begin
  textcolor(green); write('El número de alumno del alumno '); textcolor(yellow);
write(alumno); textcolor(green); write(' es '); textcolor(red);
writeln(registro_alumno2.numero);
  textcolor(green); write('El promedio CON aplazos del alumno '); textcolor(yellow);
write(alumno); textcolor(green); write(' es '); textcolor(red);
writeln(registro_alumno2.promedio_con_aplazos:0:2);
  textcolor(green); write('El promedio SIN aplazos del alumno '); textcolor(yellow);
write(alumno); textcolor(green); write(' es '); textcolor(red);
writeln(registro_alumno2.promedio_sin_aplazos:0:2);
end;
procedure imprimir_lista_alumnos2(lista_alumnos2: t_lista_alumnos2);
var
  i: int16;
begin
  i:=0;
  while (lista_alumnos2<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información del alumno '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_alumno2(lista_alumnos2^.ele,i);
    writeln();
    lista_alumnos2:=lista_alumnos2^.sig;
  end;
end;
var
  lista_alumnos1: t_lista_alumnos1;
  lista_alumnos2: t_lista_alumnos2;
begin
  randomize;
  lista_alumnos1:=nil;
  lista_alumnos2:=nil;
  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
  cargar_lista_alumnos1(lista_alumnos1);
  imprimir_lista_alumnos1(lista_alumnos1);
  writeln(); textcolor(red); writeln('INCISO (b):'); writeln();
  cargar_lista_alumnos2(lista_alumnos2,lista_alumnos1);
  imprimir_lista_alumnos2(lista_alumnos2);
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.*

(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_E2;
{$codepage UTF8}
uses crt;
const
  zona_ini=1; zona_fin=5;
  tipo_ini=1; tipo_fin=3;
  preciom2_salida=-1.0;
type
  t_zona=zona_ini..zona_fin;
  t_tipo=tipo_ini..tipo_fin;
  t_registro_propiedad1=record
    zona: t_zona;
    codigo: int16;
    tipo: t_tipo;
    m2: real;
    preciom2: real;
  end;
  t_registro_propiedad2=record
    codigo: int16;
    tipo: t_tipo;
    precio_total: real;
  end;
  t_lista_propiedades1=^t_nodo_propiedades1;
  t_nodo_propiedades1=record
    ele: t_registro_propiedad2;
    sig: t_lista_propiedades1;
  end;
  t_lista_propiedades2=^t_nodo_propiedades2;
  t_nodo_propiedades2=record
    ele: int16;
    sig: t_lista_propiedades2;
  end;
  t_vector_propiedades=array[t_zona] of t_lista_propiedades1;
procedure inicializar_vector_propiedades(var vector_propiedades: t_vector_propiedades);
var
  i: t_zona;
begin
  for i:= zona_ini to zona_fin do
    vector_propiedades[i]:=nil;
end;
procedure leer_propiedad(var registro_propiedad1: t_registro_propiedad1);
var
  i: int8;
begin
```

```
  i:=random(100);
  if (i=0) then
    registro_propiedad1.preciom2:=preciom2_salida
  else
    registro_propiedad1.preciom2:=1+random(100);
  if (registro_propiedad1.preciom2<>preciom2_salida) then
  begin
    registro_propiedad1.zona:=zona_ini+random(zona_fin);
    registro_propiedad1.codigo:=1+random(high(int16));
    registro_propiedad1.tipo:=tipo_ini+random(tipo_fin);
    registro_propiedad1.m2:=1+random(100);
  end;
end;
procedure cargar_registro_propiedad2(var registro_propiedad2: t_registro_propiedad2;
registro_propiedad1: t_registro_propiedad1);
begin
  registro_propiedad2.codigo:=registro_propiedad1.codigo;
  registro_propiedad2.tipo:=registro_propiedad1.tipo;
  registro_propiedad2.precio_total:=registro_propiedad1.m2*registro_propiedad1.preciom2;
end;
procedure agregar_ordenado_lista_propiedades1(var lista_propiedades1: t_lista_propiedades1;
registro_propiedad1: t_registro_propiedad1);
var
  anterior, actual, nuevo: t_lista_propiedades1;
begin
  new(nuevo);
  cargar_registro_propiedad2(nuevo^.ele,registro_propiedad1);
  actual:=lista_propiedades1;
  while ((actual<>nil) and (actual^.ele.tipo<nuevo^.ele.tipo)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=lista_propiedades1) then
    lista_propiedades1:=nuevo
  else
    anterior^.sig:=nuevo;
  nuevo^.sig:=actual;
end;
procedure cargar_vector_propiedades(var vector_propiedades: t_vector_propiedades);
var
  registro_propiedad1: t_registro_propiedad1;
begin
  leer_propiedad(registro_propiedad1);
  while (registro_propiedad1.preciom2<>preciom2_salida) do
  begin
    agregar_ordenado_lista_propiedades1(vector_propiedades[registro_propiedad1.zona],registro_
propiedad1);
    leer_propiedad(registro_propiedad1);
  end;
end;
procedure imprimir_registro_propiedad2(registro_propiedad2: t_registro_propiedad2; zona:
t_zona; propiedad: int16);
begin
  textcolor(green); write('El código de la propiedad '); textcolor(yellow); write(propiedad);
textcolor(green); write(' de la zona '); textcolor(yellow); write(zona); textcolor(green);
write(' es '); textcolor(red); writeln(registro_propiedad2.codigo);
  textcolor(green); write('El tipo de la propiedad '); textcolor(yellow); write(propiedad);
textcolor(green); write(' de la zona '); textcolor(yellow); write(zona); textcolor(green);
write(' es '); textcolor(red); writeln(registro_propiedad2.tipo);
  textcolor(green); write('El precio total de la propiedad '); textcolor(yellow);
write(propiedad); textcolor(green); write(' de la zona '); textcolor(yellow); write(zona);
textcolor(green); write(' es '); textcolor(red);
writeln(registro_propiedad2.precio_total:0:2);
end;
procedure imprimir_lista_propiedades1(lista_propiedades1: t_lista_propiedades1; zona: t_zona);
```

```
var
  i: int16;
begin
  i:=0;
  while (lista_propiedades1<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información de la propiedad '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_propiedad2(lista_propiedades1^.ele,zona,i);
    writeln();
    lista_propiedades1:=lista_propiedades1^.sig;
  end;
end;
procedure imprimir_vector_propiedades(vector_propiedades: t_vector_propiedades);
var
  i: t_zona;
begin
  for i:= zona_ini to zona_fin do
    imprimir_lista_propiedades1(vector_propiedades[i],i);
end;
procedure agregar_adelante_lista_propiedades2(var lista_propiedades2: t_lista_propiedades2;
codigo: int16);
var
  nuevo: t_lista_propiedades2;
begin
  new(nuevo);
  nuevo^.ele:=codigo;
  nuevo^.sig:=lista_propiedades2;
  lista_propiedades2:=nuevo;
end;
procedure cargar_lista_propiedades2(var lista_propiedades2: t_lista_propiedades2;
vector_propiedades: t_vector_propiedades; zona: t_zona; tipo: t_tipo);
begin
  while ((vector_propiedades[zona]<>nil) and (vector_propiedades[zona]^.ele.tipo<=tipo)) do
  begin
    if (vector_propiedades[zona]^.ele.tipo=tipo) then
      agregar_adelante_lista_propiedades2(lista_propiedades2,vector_propiedades[zona]^.ele.cod
igo);
    vector_propiedades[zona]:=vector_propiedades[zona]^.sig;
  end;
end;
procedure imprimir_lista_propiedades2(lista_propiedades2: t_lista_propiedades2);
var
  i: int16;
begin
  i:=0;
  while (lista_propiedades2<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('El código de la propiedad '); textcolor(yellow); write(i);
textcolor(green); write(' es '); textcolor(red); writeln(lista_propiedades2^.ele);
    lista_propiedades2:=lista_propiedades2^.sig;
  end;
end;
var
  vector_propiedades: t_vector_propiedades;
  lista_propiedades2: t_lista_propiedades2;
  zona: t_zona;
  tipo: t_tipo;
begin
  randomize;
  inicializar_vector_propiedades(vector_propiedades);
  lista_propiedades2:=nil;
  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
  cargar_vector_propiedades(vector_propiedades);
```

```
  imprimir_vector_propiedades(vector_propiedades);
  writeln(); textcolor(red); writeln('INCISO (b):'); writeln();
  zona:=zona_ini+random(zona_fin);
  tipo:=tipo_ini+random(tipo_fin);
  cargar_lista_propiedades2(lista_propiedades2,vector_propiedades,zona,tipo);
  if (lista_propiedades2<>nil) then
    imprimir_lista_propiedades2(lista_propiedades2);
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 1000 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.*

(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_E3;
{$codepage UTF8}
uses crt;
const
  productos_total=1000;
  codigo_venta_salida=-1;
  cantidad_salida=0;
type
  t_producto=1..productos_total;
  t_registro_producto=record
    codigo_producto: int16;
    cantidad: int8;
    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_producto] of t_registro_producto;
procedure cargar_vector_productos(var vector_productos: t_vector_productos);
var
  i: t_producto;
begin
  for i:= 1 to productos_total do
  begin
    vector_productos[i].codigo_producto:=i;
    vector_productos[i].cantidad:=1+random(high(int8));
    vector_productos[i].precio:=1+random(100);
```

```
  end;
end;
function buscar_vector_productos(vector_productos: t_vector_productos; codigo_producto:
int16): t_producto;
var
  pos: t_producto;
begin
  pos:=1;
  while (vector_productos[pos].codigo_producto<>codigo_producto) do
    pos:=pos+1;
  buscar_vector_productos:=pos;
end;
procedure actualizar_vector_productos(var vector_productos: t_vector_productos; var
registro_producto: t_registro_producto; pos: t_producto);
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 leer_producto(var registro_producto: t_registro_producto; var vector_productos:
t_vector_productos; var monto_total: real);
var
  pos: t_producto;
  i: int8;
begin
  i:=random(10);
  if (i=0) then
    registro_producto.cantidad:=cantidad_salida
  else
    registro_producto.cantidad:=1+random(high(int8));
  if (registro_producto.cantidad<>cantidad_salida) then
  begin
    registro_producto.codigo_producto:=1+random(productos_total);
    pos:=buscar_vector_productos(vector_productos,registro_producto.codigo_producto);
    actualizar_vector_productos(vector_productos,registro_producto,pos);
    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
  nuevo: t_lista_productos;
begin
  new(nuevo);
  nuevo^.ele:=registro_producto;
  nuevo^.sig:=lista_productos;
  lista_productos:=nuevo;
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
  leer_producto(registro_producto,vector_productos,monto_total);
  while (registro_producto.cantidad<>cantidad_salida) do
  begin
    agregar_adelante_lista_productos(lista_productos,registro_producto);
    leer_producto(registro_producto,vector_productos,monto_total);
  end;
end;
```

```
procedure leer_venta(var registro_venta: t_registro_venta; var vector_productos:
t_vector_productos);
var
  i: int8;
  monto_total: real;
begin
  i:=random(100);
  if (i=0) then
    registro_venta.codigo_venta:=codigo_venta_salida
  else
    registro_venta.codigo_venta:=1+random(high(int16));
  if (registro_venta.codigo_venta<>codigo_venta_salida) then
  begin
    registro_venta.productos:=nil; monto_total:=0;
    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
  nuevo: t_lista_ventas;
begin
  new(nuevo);
  nuevo^.ele:=registro_venta;
  nuevo^.sig:=lista_ventas;
  lista_ventas:=nuevo;
end;
procedure cargar_lista_ventas(var lista_ventas: t_lista_ventas; vector_productos:
t_vector_productos);
var
  registro_venta: t_registro_venta;
begin
  leer_venta(registro_venta,vector_productos);
  while (registro_venta.codigo_venta<>codigo_venta_salida) do
  begin
    agregar_adelante_lista_ventas(lista_ventas,registro_venta);
    leer_venta(registro_venta,vector_productos);
  end;
end;
procedure imprimir_registro_producto(registro_producto: t_registro_producto; venta: int16;
codigo: int16);
begin
  textcolor(green); write('El código de producto del producto '); textcolor(yellow);
write(codigo); textcolor(green); write(' de la venta '); textcolor(yellow); write(venta);
textcolor(green); write(' es '); textcolor(red); writeln(registro_producto.codigo_producto);
  textcolor(green); write('La cantidad del producto '); textcolor(yellow); write(codigo);
textcolor(green); write(' de la venta '); textcolor(yellow); write(venta); textcolor(green);
write(' es '); textcolor(red); writeln(registro_producto.cantidad);
  textcolor(green); write('El precio unitario del producto '); textcolor(yellow);
write(codigo); textcolor(green); write(' de la venta '); textcolor(yellow); write(venta);
textcolor(green); write(' es '); textcolor(red); writeln(registro_producto.precio:0:2);
end;
procedure imprimir_lista_productos(lista_productos: t_lista_productos; venta: int16);
var
  i: int16;
begin
  i:=0;
  while (lista_productos<>nil) do
  begin
    i:=i+1;
    imprimir_registro_producto(lista_productos^.ele,venta,i);
    lista_productos:=lista_productos^.sig;
  end;
end;
procedure imprimir_registro_venta(registro_venta: t_registro_venta; venta: int16);
```

```
begin
  textcolor(green); write('El código de venta de la venta '); textcolor(yellow); write(venta);
textcolor(green); write(' es '); textcolor(red); writeln(registro_venta.codigo_venta);
  textcolor(green); write('El monto total de la venta '); textcolor(yellow); write(venta);
textcolor(green); write(' es '); textcolor(red); writeln(registro_venta.monto_total:0:2);
  imprimir_lista_productos(registro_venta.productos,venta);
end;
procedure imprimir_lista_ventas(lista_ventas: t_lista_ventas);
var
  i: int16;
begin
  i:=0;
  while (lista_ventas<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('La información de la venta '); textcolor(yellow); write(i);
textcolor(green); writeln(' es:');
    imprimir_registro_venta(lista_ventas^.ele,i);
    writeln();
    lista_ventas:=lista_ventas^.sig;
  end;
end;
procedure buscar_lista_productos(lista_productos: t_lista_productos; codigo_producto: int16;
var ventas: int32);
begin
  while (lista_productos<>nil) do
  begin
    if (lista_productos^.ele.codigo_producto=codigo_producto) then
      ventas:=ventas+lista_productos^.ele.cantidad;
    lista_productos:=lista_productos^.sig;
  end;
end;
function buscar_lista_ventas(lista_ventas: t_lista_ventas; codigo_producto: int16): int32;
var
  ventas: int32;
begin
  ventas:=0;
  while (lista_ventas<>nil) do
  begin
    buscar_lista_productos(lista_ventas^.ele.productos,codigo_producto,ventas);
    lista_ventas:=lista_ventas^.sig;
  end;
  buscar_lista_ventas:=ventas;
end;
var
  vector_productos: t_vector_productos;
  lista_ventas: t_lista_ventas;
  codigo_producto: int16;
begin
  randomize;
  lista_ventas:=nil;
  cargar_vector_productos(vector_productos);
  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();
  cargar_lista_ventas(lista_ventas,vector_productos);
  if (lista_ventas<>nil) then
  begin
    imprimir_lista_ventas(lista_ventas);
    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();
    codigo_producto:=1+random(productos_total);
    textcolor(green); write('La cantidad de unidades vendidas en la lista del código de
producto '); textcolor(yellow); write(codigo_producto); textcolor(green); write(' es ');
textcolor(red); write(buscar_lista_ventas(lista_ventas,codigo_producto));
  end;
end.
```