**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\_aprobadas

    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);

  anterior:=lista\_propiedades1; 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.codigo);

    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.