**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\_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: string;

    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\_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: string;

    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\_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;

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\_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: string;

    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\_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;

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\_rendidos;

    registro\_alumno2.promedio\_sin\_aplazos:=suma\_sin\_aplazos/registro\_alumno1.materias\_aprobadas;

  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.