**Trabajo Práctico N° 4:**

**Módulo Imperativo (Árboles 2).**

**Ejercicio 1.**

*Implementar un programa modularizado para una librería que:*

**(a)** *Almacene los productos vendidos en una estructura eficiente para la búsqueda por código de producto. De cada producto, deben quedar almacenados la cantidad total de unidades vendidas y el monto total. De cada venta, se lee código de venta, código del producto vendido, cantidad de unidades vendidas y precio unitario. El ingreso de las ventas finaliza cuando se lee el código de venta -1.*

**(b)** *Imprima el contenido del árbol ordenado por código de producto.*

**(c)** *Contenga un módulo que reciba la estructura generada en el inciso (a) y retorne el código de producto con mayor cantidad de unidades vendidas.*

**(d)** *Contenga un módulo que reciba la estructura generada en el inciso (a) y un código de producto y retorne la cantidad de códigos menores que él que hay en la estructura.*

**(e)** *Contenga un módulo que reciba la estructura generada en el inciso (a) y dos códigos de producto y retorne el monto total entre todos los códigos de productos comprendidos entre los dos valores recibidos (sin incluir).*

program TP4\_E1;

*{$codepage UTF8}*

uses crt;

const

  codigo\_venta\_salida=-1;

type

  t\_registro\_venta=record

    codigo\_venta: int16;

    codigo\_producto: int16;

    cantidad: int8;

    precio: real;

  end;

  t\_registro\_producto=record

    codigo\_producto: int16;

    cantidad\_total: int16;

    monto\_total: real;

  end;

  t\_abb\_productos=^t\_nodo\_abb\_productos;

  t\_nodo\_abb\_productos=record

    ele: t\_registro\_producto;

    hi: t\_abb\_productos;

    hd: t\_abb\_productos;

  end;

procedure leer\_venta(var registro\_venta: t\_registro\_venta);

var

  i: int8;

begin

  i:=random(100);

  if (i=0) then

    registro\_venta.codigo\_venta:=codigo\_venta\_salida

  else

    registro\_venta.codigo\_venta:=random(high(int16));

  if (registro\_venta.codigo\_venta<>codigo\_venta\_salida) then

  begin

    registro\_venta.codigo\_producto:=1+random(high(int16));

    registro\_venta.cantidad:=1+random(high(int8));

    registro\_venta.precio:=1+random(100);

  end;

end;

procedure cargar\_registro\_producto(var registro\_producto: t\_registro\_producto; registro\_venta: t\_registro\_venta);

begin

  registro\_producto.codigo\_producto:=registro\_venta.codigo\_producto;

  registro\_producto.cantidad\_total:=registro\_venta.cantidad;

  registro\_producto.monto\_total:=registro\_venta.cantidad\*registro\_venta.precio;

end;

procedure agregar\_abb\_productos(var abb\_productos: t\_abb\_productos; registro\_venta: t\_registro\_venta);

begin

  if (abb\_productos=nil) then

  begin

    new(abb\_productos);

    cargar\_registro\_producto(abb\_productos^.ele,registro\_venta);

    abb\_productos^.hi:=nil;

    abb\_productos^.hd:=nil;

  end

  else

    if (registro\_venta.codigo\_producto=abb\_productos^.ele.codigo\_producto) then

    begin

      abb\_productos^.ele.cantidad\_total:=abb\_productos^.ele.cantidad\_total+registro\_venta.cantidad;

      abb\_productos^.ele.monto\_total:=abb\_productos^.ele.monto\_total+registro\_venta.cantidad\*registro\_venta.precio;

    end

    else

      if (registro\_venta.codigo\_producto<abb\_productos^.ele.codigo\_producto) then

        agregar\_abb\_productos(abb\_productos^.hi,registro\_venta)

      else

        agregar\_abb\_productos(abb\_productos^.hd,registro\_venta);

end;

procedure cargar\_abb\_productos(var abb\_productos: t\_abb\_productos);

var

  registro\_venta: t\_registro\_venta;

begin

  leer\_venta(registro\_venta);

  while (registro\_venta.codigo\_venta<>codigo\_venta\_salida) do

  begin

    agregar\_abb\_productos(abb\_productos,registro\_venta);

    leer\_venta(registro\_venta);

  end;

end;

procedure imprimir\_registro\_producto(registro\_producto: t\_registro\_producto);

begin

  textcolor(green); write('El código de producto del producto es '); textcolor(red); writeln(registro\_producto.codigo\_producto);

  textcolor(green); write('La cantidad total de unidades vendidas del producto es '); textcolor(red); writeln(registro\_producto.cantidad\_total);

  textcolor(green); write('El monto total del producto es $'); textcolor(red); writeln(registro\_producto.monto\_total:0:2);

  writeln();

end;

procedure imprimir\_abb\_productos(abb\_productos: t\_abb\_productos);

begin

  if (abb\_productos<>nil) then

  begin

    imprimir\_abb\_productos(abb\_productos^.hi);

    imprimir\_registro\_producto(abb\_productos^.ele);

    imprimir\_abb\_productos(abb\_productos^.hd);

  end;

end;

procedure buscar\_codigo\_mayor\_cantidad(abb\_productos: t\_abb\_productos; var cantidad\_max, codigo\_max: int16);

begin

  if (abb\_productos<>nil) then

  begin

    buscar\_codigo\_mayor\_cantidad(abb\_productos^.hi,cantidad\_max,codigo\_max);

    if (abb\_productos^.ele.cantidad\_total>cantidad\_max) then

    begin

      cantidad\_max:=abb\_productos^.ele.cantidad\_total;

      codigo\_max:=abb\_productos^.ele.codigo\_producto;

    end;

    buscar\_codigo\_mayor\_cantidad(abb\_productos^.hd,cantidad\_max,codigo\_max);

  end;

end;

function contar\_codigos(abb\_productos: t\_abb\_productos; codigo: int16): int16;

begin

  if (abb\_productos=nil) then

    contar\_codigos:=0

  else

    if (abb\_productos^.ele.codigo\_producto<codigo) then

      contar\_codigos:=contar\_codigos(abb\_productos^.hi,codigo)+contar\_codigos(abb\_productos^.hd,codigo)+1

    else

      contar\_codigos:=contar\_codigos(abb\_productos^.hi,codigo);

end;

procedure verificar\_codigos(var codigo1, codigo2: int16);

var

  aux: int16;

begin

  if (codigo1>codigo2) then

  begin

    aux:=codigo1;

    codigo1:=codigo2;

    codigo2:=aux;

  end;

end;

function contar\_monto\_total(abb\_productos: t\_abb\_productos; codigo1, codigo2: int16): real;

begin

  if (abb\_productos=nil) then

    contar\_monto\_total:=0

  else

    if (codigo1>=abb\_productos^.ele.codigo\_producto) then

      contar\_monto\_total:=contar\_monto\_total(abb\_productos^.hd,codigo1,codigo2)

    else if (codigo2<=abb\_productos^.ele.codigo\_producto) then

      contar\_monto\_total:=contar\_monto\_total(abb\_productos^.hi,codigo1,codigo2)

    else

      contar\_monto\_total:=contar\_monto\_total(abb\_productos^.hi,codigo1,codigo2)+contar\_monto\_total(abb\_productos^.hd,codigo1,codigo2)+abb\_productos^.ele.monto\_total;

end;

var

  abb\_productos: t\_abb\_productos;

  cantidad\_max, codigo\_max, codigo, codigo1, codigo2: int16;

begin

  randomize;

  abb\_productos:=nil;

  cantidad\_max:=low(int16); codigo\_max:=0;

  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();

  cargar\_abb\_productos(abb\_productos);

  if (abb\_productos<>nil) then

  begin

    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();

    imprimir\_abb\_productos(abb\_productos);

    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();

    buscar\_codigo\_mayor\_cantidad(abb\_productos,cantidad\_max,codigo\_max);

    textcolor(green); write('El código de producto con mayor cantidad de unidades vendidas es '); textcolor(red); writeln(codigo\_max);

    writeln(); textcolor(red); writeln('INCISO (d):'); writeln();

    codigo:=1+random(high(int16));

    textcolor(green); write('La cantidad de códigos menores que el código de producto '); textcolor(yellow); write(codigo); textcolor(green); write(' es '); textcolor(red); writeln(contar\_codigos(abb\_productos,codigo));

    writeln(); textcolor(red); writeln('INCISO (e):'); writeln();

    codigo1:=1+random(high(int16)); codigo2:=1+random(high(int16));

    verificar\_codigos(codigo1,codigo2);

    textcolor(green); write('El monto total en el abb cuyo código de producto se encuentra entre '); textcolor(yellow); write(codigo1); textcolor(green); write(' y '); textcolor(yellow); write(codigo2); textcolor(green); write(' es $'); textcolor(red); write(contar\_monto\_total(abb\_productos,codigo1,codigo2):0:2);

  end;

end.

**Ejercicio 2.**

*Una biblioteca nos ha encargado procesar la información de los préstamos realizados durante el año 2021. De cada préstamo, se conoce el ISBN del libro, el número de socio, día y mes del préstamo y cantidad de días prestados. Implementar un programa con:*

**(a)** *Un módulo que lea préstamos y retorne 2 estructuras de datos con la información de los préstamos. La lectura de los préstamos finaliza con ISBN -1. Las estructuras deben ser eficientes para buscar por ISBN.*

**(i)** *En una estructura, cada préstamo debe estar en un nodo.*

**(ii)** *En otra estructura, cada nodo debe contener todos los préstamos realizados al ISBN (prestar atención sobre los datos que se almacenan).*

**(b)** *Un módulo recursivo que reciba la estructura generada en (i) y retorne el ISBN más grande.*

**(c)** *Un módulo recursivo que reciba la estructura generada en (ii) y retorne el ISBN más pequeño.*

**(d)** *Un módulo recursivo que reciba la estructura generada en (i) y un número de socio. El módulo debe retornar la cantidad de préstamos realizados a dicho socio.*

**(e)** *Un módulo recursivo que reciba la estructura generada en (ii) y un número de socio. El módulo debe retornar la cantidad de préstamos realizados a dicho socio.*

**(f)** *Un módulo que reciba la estructura generada en (i) y retorne una nueva estructura ordenada ISBN, donde cada ISBN aparezca una vez junto a la cantidad total de veces que se prestó.*

**(g)** *Un módulo que reciba la estructura generada en (ii) y retorne una nueva estructura ordenada ISBN, donde cada ISBN aparezca una vez junto a la cantidad total de veces que se prestó.*

**(h)** *Un módulo recursivo que reciba la estructura generada en (g) y muestre su contenido.*

**(i)** *Un módulo recursivo que reciba la estructura generada en (i) y dos valores de ISBN. El módulo debe retornar la cantidad total de préstamos realizados a los ISBN comprendidos entre los dos valores recibidos (incluidos).*

**(j)** *Un módulo recursivo que reciba la estructura generada en (ii) y dos valores de ISBN. El módulo debe retornar la cantidad total de préstamos realizados a los ISBN comprendidos entre los dos valores recibidos (incluidos).*

program TP4\_E2;

*{$codepage UTF8}*

uses crt;

const

  dia\_ini=1; dia\_fin=31;

  mes\_ini=1; mes\_fin=12;

  isbn\_salida=-1;

type

  t\_dia=dia\_ini..dia\_fin;

  t\_mes=mes\_ini..mes\_fin;

  t\_registro\_prestamo1=record

    isbn: int8;

    socio: int8;

    dia: t\_dia;

    mes: t\_mes;

    dias\_prestados: int8;

  end;

  t\_abb\_prestamos=^t\_nodo\_abb\_prestamos;

  t\_nodo\_abb\_prestamos=record

    ele: t\_registro\_prestamo1;

    hi: t\_abb\_prestamos;

    hd: t\_abb\_prestamos;

  end;

  t\_registro\_prestamo2=record

    socio: int8;

    dia: t\_dia;

    mes: t\_mes;

    dias\_prestados: int8;

  end;

  t\_lista\_prestamos=^t\_nodo\_prestamos;

  t\_nodo\_prestamos=record

    ele: t\_registro\_prestamo2;

    sig: t\_lista\_prestamos;

  end;

  t\_registro\_isbn1=record

    isbn: int8;

    prestamos: t\_lista\_prestamos;

  end;

  t\_abb\_isbns=^t\_nodo\_abb\_isbns;

  t\_nodo\_abb\_isbns=record

    ele: t\_registro\_isbn1;

    hi: t\_abb\_isbns;

    hd: t\_abb\_isbns;

  end;

  t\_registro\_isbn2=record

    isbn: int8;

    prestamos: int16;

  end;

  t\_lista\_isbns=^t\_nodo\_isbns;

  t\_nodo\_isbns=record

    ele: t\_registro\_isbn2;

    sig: t\_lista\_isbns;

  end;

procedure leer\_prestamo(var registro\_prestamo1: t\_registro\_prestamo1);

var

  i: int8;

begin

  i:=random(100);

  if (i=0) then

    registro\_prestamo1.isbn:=isbn\_salida

  else

    registro\_prestamo1.isbn:=1+random(high(int8));

  if (registro\_prestamo1.isbn<>isbn\_salida) then

  begin

    registro\_prestamo1.socio:=1+random(high(int8));

    registro\_prestamo1.dia:=dia\_ini+random(dia\_fin);

    registro\_prestamo1.mes:=mes\_ini+random(mes\_fin);

    registro\_prestamo1.dias\_prestados:=1+random(high(int8));

  end;

end;

procedure agregar\_abb\_prestamos(var abb\_prestamos: t\_abb\_prestamos; registro\_prestamo1: t\_registro\_prestamo1);

begin

  if (abb\_prestamos=nil) then

  begin

    new(abb\_prestamos);

    abb\_prestamos^.ele:=registro\_prestamo1;

    abb\_prestamos^.hi:=nil;

    abb\_prestamos^.hd:=nil;

  end

  else

    if (registro\_prestamo1.isbn<=abb\_prestamos^.ele.isbn) then

      agregar\_abb\_prestamos(abb\_prestamos^.hi,registro\_prestamo1)

    else

      agregar\_abb\_prestamos(abb\_prestamos^.hd,registro\_prestamo1);

end;

procedure cargar\_registro\_prestamo2(var registro\_prestamo2: t\_registro\_prestamo2; registro\_prestamo1: t\_registro\_prestamo1);

begin

  registro\_prestamo2.socio:=registro\_prestamo1.socio;

  registro\_prestamo2.dia:=registro\_prestamo1.dia;

  registro\_prestamo2.mes:=registro\_prestamo1.mes;

  registro\_prestamo2.dias\_prestados:=registro\_prestamo1.dias\_prestados;

end;

procedure agregar\_adelante\_lista\_prestamos(var lista\_prestamos: t\_lista\_prestamos; registro\_prestamo1: t\_registro\_prestamo1);

var

  nuevo: t\_lista\_prestamos;

begin

  new(nuevo);

  cargar\_registro\_prestamo2(nuevo^.ele,registro\_prestamo1);

  nuevo^.sig:=lista\_prestamos;

  lista\_prestamos:=nuevo;

end;

procedure cargar\_registro\_isbn1(var registro\_isbn1: t\_registro\_isbn1; registro\_prestamo1: t\_registro\_prestamo1);

begin

  registro\_isbn1.isbn:=registro\_prestamo1.isbn;

  registro\_isbn1.prestamos:=nil;

  agregar\_adelante\_lista\_prestamos(registro\_isbn1.prestamos,registro\_prestamo1);

end;

procedure agregar\_abb\_isbns(var abb\_isbns: t\_abb\_isbns; registro\_prestamo1: t\_registro\_prestamo1);

begin

  if (abb\_isbns=nil) then

  begin

    new(abb\_isbns);

    cargar\_registro\_isbn1(abb\_isbns^.ele,registro\_prestamo1);

    abb\_isbns^.hi:=nil;

    abb\_isbns^.hd:=nil;

  end

  else

    if (registro\_prestamo1.isbn=abb\_isbns^.ele.isbn) then

      agregar\_adelante\_lista\_prestamos(abb\_isbns^.ele.prestamos,registro\_prestamo1)

    else if (registro\_prestamo1.isbn<abb\_isbns^.ele.isbn) then

      agregar\_abb\_isbns(abb\_isbns^.hi,registro\_prestamo1)

    else

      agregar\_abb\_isbns(abb\_isbns^.hd,registro\_prestamo1);

end;

procedure cargar\_abbs(var abb\_prestamos: t\_abb\_prestamos; var abb\_isbns: t\_abb\_isbns);

var

  registro\_prestamo1: t\_registro\_prestamo1;

begin

  leer\_prestamo(registro\_prestamo1);

  while (registro\_prestamo1.isbn<>isbn\_salida) do

  begin

    agregar\_abb\_prestamos(abb\_prestamos,registro\_prestamo1);

    agregar\_abb\_isbns(abb\_isbns,registro\_prestamo1);

    leer\_prestamo(registro\_prestamo1);

  end;

end;

procedure imprimir\_registro\_prestamo1(registro\_prestamo1: t\_registro\_prestamo1);

begin

  textcolor(green); write('El ISBN del préstamo es '); textcolor(red); writeln(registro\_prestamo1.isbn);

  textcolor(green); write('El número de socio del préstamo es '); textcolor(red); writeln(registro\_prestamo1.socio);

  textcolor(green); write('El día del préstamo es '); textcolor(red); writeln(registro\_prestamo1.dia);

  textcolor(green); write('El mes del préstamo es '); textcolor(red); writeln(registro\_prestamo1.mes);

  textcolor(green); write('La cantidad de días prestados del préstamo es '); textcolor(red); writeln(registro\_prestamo1.dias\_prestados);

  writeln();

end;

procedure imprimir\_abb\_prestamos(abb\_prestamos: t\_abb\_prestamos);

begin

  if (abb\_prestamos<>nil) then

  begin

    imprimir\_abb\_prestamos(abb\_prestamos^.hi);

    imprimir\_registro\_prestamo1(abb\_prestamos^.ele);

    imprimir\_abb\_prestamos(abb\_prestamos^.hd);

  end;

end;

procedure imprimir\_registro\_prestamo2(registro\_prestamo2: t\_registro\_prestamo2; isbn: int8; prestamo: int16);

begin

  textcolor(green); write('El número de socio del préstamo '); textcolor(yellow); write(prestamo); textcolor(green); write(' del ISBN '); textcolor(yellow); write(isbn); textcolor(green); write(' es '); textcolor(red); writeln(registro\_prestamo2.socio);

  textcolor(green); write('El día del préstamo '); textcolor(yellow); write(prestamo); textcolor(green); write(' del ISBN '); textcolor(yellow); write(isbn); textcolor(green); write(' es '); textcolor(red); writeln(registro\_prestamo2.dia);

  textcolor(green); write('El mes del préstamo '); textcolor(yellow); write(prestamo); textcolor(green); write(' del ISBN '); textcolor(yellow); write(isbn); textcolor(green); write(' es '); textcolor(red); writeln(registro\_prestamo2.mes);

  textcolor(green); write('La cantidad de días prestados del préstamo '); textcolor(yellow); write(prestamo); textcolor(green); write(' del ISBN '); textcolor(yellow); write(isbn); textcolor(green); write(' es '); textcolor(red); writeln(registro\_prestamo2.dias\_prestados);

end;

procedure imprimir\_lista\_prestamos(lista\_prestamos: t\_lista\_prestamos; isbn: int8);

var

  i: int16;

begin

  i:=0;

  while (lista\_prestamos<>nil) do

  begin

    i:=i+1;

    imprimir\_registro\_prestamo2(lista\_prestamos^.ele,isbn,i);

    lista\_prestamos:=lista\_prestamos^.sig;

  end;

end;

procedure imprimir\_registro\_isbn1(registro\_isbn1: t\_registro\_isbn1);

begin

  textcolor(green); write('El ISBN del préstamo es '); textcolor(red); writeln(registro\_isbn1.isbn);

  imprimir\_lista\_prestamos(registro\_isbn1.prestamos,registro\_isbn1.isbn);

  writeln();

end;

procedure imprimir\_abb\_isbns(abb\_isbns: t\_abb\_isbns);

begin

  if (abb\_isbns<>nil) then

  begin

    imprimir\_abb\_isbns(abb\_isbns^.hi);

    imprimir\_registro\_isbn1(abb\_isbns^.ele);

    imprimir\_abb\_isbns(abb\_isbns^.hd);

  end;

end;

function buscar\_mayor\_isbn(abb\_prestamos: t\_abb\_prestamos): int8;

begin

  if (abb\_prestamos^.hd=nil) then

    buscar\_mayor\_isbn:=abb\_prestamos^.ele.isbn

  else

    buscar\_mayor\_isbn:=buscar\_mayor\_isbn(abb\_prestamos^.hd);

end;

function buscar\_menor\_isbn(abb\_isbns: t\_abb\_isbns): int8;

begin

  if (abb\_isbns^.hi=nil) then

    buscar\_menor\_isbn:=abb\_isbns^.ele.isbn

  else

    buscar\_menor\_isbn:=buscar\_menor\_isbn(abb\_isbns^.hi);

end;

function contar\_abb\_prestamos(abb\_prestamos: t\_abb\_prestamos; socio: int8): int16;

begin

  if (abb\_prestamos=nil) then

    contar\_abb\_prestamos:=0

  else

    if (socio=abb\_prestamos^.ele.socio) then

      contar\_abb\_prestamos:=contar\_abb\_prestamos(abb\_prestamos^.hi,socio)+contar\_abb\_prestamos(abb\_prestamos^.hd,socio)+1

    else

      contar\_abb\_prestamos:=contar\_abb\_prestamos(abb\_prestamos^.hi,socio)+contar\_abb\_prestamos(abb\_prestamos^.hd,socio);

end;

function contar\_socios(lista\_prestamos: t\_lista\_prestamos; socio: int8): int16;

var

  socios: int16;

begin

  socios:=0;

  while (lista\_prestamos<>nil) do

  begin

    if (socio=lista\_prestamos^.ele.socio) then

      socios:=socios+1;

    lista\_prestamos:=lista\_prestamos^.sig;

  end;

  contar\_socios:=socios;

end;

function contar\_abb\_isbns(abb\_isbns: t\_abb\_isbns; socio: int8): int16;

begin

  if (abb\_isbns=nil) then

    contar\_abb\_isbns:=0

  else

    contar\_abb\_isbns:=contar\_abb\_isbns(abb\_isbns^.hi,socio)+contar\_abb\_isbns(abb\_isbns^.hd,socio)+contar\_socios(abb\_isbns^.ele.prestamos,socio);

end;

procedure cargar1\_registro\_isbn2(var registro\_isbn2: t\_registro\_isbn2; isbn: int8);

begin

  registro\_isbn2.isbn:=isbn;

  registro\_isbn2.prestamos:=1;

end;

procedure agregar\_adelante\_lista\_isbns1(var lista\_isbns1: t\_lista\_isbns; isbn: int8);

var

  nuevo: t\_lista\_isbns;

begin

  new(nuevo);

  cargar1\_registro\_isbn2(nuevo^.ele,isbn);

  nuevo^.sig:=lista\_isbns1;

  lista\_isbns1:=nuevo;

end;

procedure cargar\_lista\_isbns1(var lista\_isbns1: t\_lista\_isbns; abb\_prestamos: t\_abb\_prestamos);

begin

  if (abb\_prestamos<>nil) then

  begin

    cargar\_lista\_isbns1(lista\_isbns1,abb\_prestamos^.hd);

    if ((lista\_isbns1<>nil) and (lista\_isbns1^.ele.isbn=abb\_prestamos^.ele.isbn)) then

      lista\_isbns1^.ele.prestamos:=lista\_isbns1^.ele.prestamos+1

    else

      agregar\_adelante\_lista\_isbns1(lista\_isbns1,abb\_prestamos^.ele.isbn);

    cargar\_lista\_isbns1(lista\_isbns1,abb\_prestamos^.hi);

  end;

end;

function contar\_prestamos(lista\_prestamos: t\_lista\_prestamos): int16;

var

  prestamos: int16;

begin

  prestamos:=0;

  while (lista\_prestamos<>nil) do

  begin

    prestamos:=prestamos+1;

    lista\_prestamos:=lista\_prestamos^.sig;

  end;

  contar\_prestamos:=prestamos;

end;

procedure cargar2\_registro\_isbn2(var registro\_isbn2: t\_registro\_isbn2; registro\_isbn1: t\_registro\_isbn1);

begin

  registro\_isbn2.isbn:=registro\_isbn1.isbn;

  registro\_isbn2.prestamos:=contar\_prestamos(registro\_isbn1.prestamos);

end;

procedure agregar\_adelante\_lista\_isbns2(var lista\_isbns2: t\_lista\_isbns; registro\_isbn1: t\_registro\_isbn1);

var

  nuevo: t\_lista\_isbns;

begin

  new(nuevo);

  cargar2\_registro\_isbn2(nuevo^.ele,registro\_isbn1);

  nuevo^.sig:=lista\_isbns2;

  lista\_isbns2:=nuevo;

end;

procedure cargar\_lista\_isbns2(var lista\_isbns2: t\_lista\_isbns; abb\_isbns: t\_abb\_isbns);

begin

  if (abb\_isbns<>nil) then

  begin

    cargar\_lista\_isbns2(lista\_isbns2,abb\_isbns^.hd);

    agregar\_adelante\_lista\_isbns2(lista\_isbns2,abb\_isbns^.ele);

    cargar\_lista\_isbns2(lista\_isbns2,abb\_isbns^.hi);

  end;

end;

procedure imprimir\_registro\_isbn2(registro\_isbn2: t\_registro\_isbn2);

begin

  textcolor(green); write('El ISBN es '); textcolor(red); writeln(registro\_isbn2.isbn);

  textcolor(green); write('La cantidad total de veces que se prestó es '); textcolor(red); writeln(registro\_isbn2.prestamos);

end;

procedure imprimir1\_lista\_isbns(lista\_isbns: t\_lista\_isbns);

begin

  while (lista\_isbns<>nil) do

  begin

    imprimir\_registro\_isbn2(lista\_isbns^.ele);

    writeln();

    lista\_isbns:=lista\_isbns^.sig;

  end;

end;

procedure imprimir2\_lista\_isbns(lista\_isbns: t\_lista\_isbns);

begin

  if (lista\_isbns<>nil) then

  begin

    imprimir\_registro\_isbn2(lista\_isbns^.ele);

    imprimir2\_lista\_isbns(lista\_isbns^.sig);

  end;

end;

procedure verificar\_isbns(var isbn1, isbn2: int8);

var

  aux: int8;

begin

  if (isbn1>isbn2) then

  begin

    aux:=isbn1;

    isbn1:=isbn2;

    isbn2:=aux;

  end;

end;

function contar\_isbns1(abb\_prestamos: t\_abb\_prestamos; isbn1, isbn2: int8): int16;

begin

  if (abb\_prestamos=nil) then

    contar\_isbns1:=0

  else

    if (isbn1>abb\_prestamos^.ele.isbn) then

      contar\_isbns1:=contar\_isbns1(abb\_prestamos^.hd,isbn1,isbn2)

    else if (isbn2<abb\_prestamos^.ele.isbn) then

      contar\_isbns1:=contar\_isbns1(abb\_prestamos^.hi,isbn1,isbn2)

    else

      contar\_isbns1:=contar\_isbns1(abb\_prestamos^.hi,isbn1,isbn2)+contar\_isbns1(abb\_prestamos^.hd,isbn1,isbn2)+1;

end;

function contar\_isbns2(abb\_isbns: t\_abb\_isbns; isbn1, isbn2: int8): int16;

begin

  if (abb\_isbns=nil) then

    contar\_isbns2:=0

  else

    if (isbn1>abb\_isbns^.ele.isbn) then

      contar\_isbns2:=contar\_isbns2(abb\_isbns^.hd,isbn1,isbn2)

    else if (isbn2<abb\_isbns^.ele.isbn) then

      contar\_isbns2:=contar\_isbns2(abb\_isbns^.hi,isbn1,isbn2)

    else

      contar\_isbns2:=contar\_isbns2(abb\_isbns^.hi,isbn1,isbn2)+contar\_isbns2(abb\_isbns^.hd,isbn1,isbn2)+contar\_prestamos(abb\_isbns^.ele.prestamos);

end;

var

  lista\_isbns1, lista\_isbns2: t\_lista\_isbns;

  abb\_prestamos: t\_abb\_prestamos;

  abb\_isbns: t\_abb\_isbns;

  socio, isbn1, isbn2: int8;

begin

  randomize;

  abb\_prestamos:=nil; abb\_isbns:=nil;

  lista\_isbns1:=nil; lista\_isbns2:=nil;

  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();

  cargar\_abbs(abb\_prestamos,abb\_isbns);

  if ((abb\_prestamos<>nil) and (abb\_isbns<>nil)) then

  begin

    writeln(); textcolor(red); writeln('ABB\_PRESTAMOS:'); writeln();

    imprimir\_abb\_prestamos(abb\_prestamos);

    writeln(); textcolor(red); writeln('ABB\_ISBNS:'); writeln();

    imprimir\_abb\_isbns(abb\_isbns);

    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();

    textcolor(green); write('El ISBN más grande es '); textcolor(red); writeln(buscar\_mayor\_isbn(abb\_prestamos));

    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();

    textcolor(green); write('El ISBN más chico es '); textcolor(red); writeln(buscar\_menor\_isbn(abb\_isbns));

    writeln(); textcolor(red); writeln('INCISO (d):'); writeln();

    socio:=1+random(high(int8));

    textcolor(green); write('La cantidad de préstamos en el abb\_prestamos realizados al número de socio '); textcolor(yellow); write(socio); textcolor(green); write(' es '); textcolor(red); writeln(contar\_abb\_prestamos(abb\_prestamos,socio));

    writeln(); textcolor(red); writeln('INCISO (e):'); writeln();

    socio:=1+random(high(int8));

    textcolor(green); write('La cantidad de préstamos en el abb\_isbns realizados al número de socio '); textcolor(yellow); write(socio); textcolor(green); write(' es '); textcolor(red); writeln(contar\_abb\_isbns(abb\_isbns,socio));

    writeln(); textcolor(red); writeln('INCISO (f):'); writeln();

    cargar\_lista\_isbns1(lista\_isbns1,abb\_prestamos);

    imprimir1\_lista\_isbns(lista\_isbns1);

    writeln(); textcolor(red); writeln('INCISO (g):'); writeln();

    cargar\_lista\_isbns2(lista\_isbns2,abb\_isbns);

    imprimir1\_lista\_isbns(lista\_isbns2);

    writeln(); textcolor(red); writeln('INCISO (h):'); writeln();

    imprimir2\_lista\_isbns(lista\_isbns1);

    writeln();

    imprimir2\_lista\_isbns(lista\_isbns2);

    writeln(); textcolor(red); writeln('INCISO (i):'); writeln();

    isbn1:=1+random(high(int8)); isbn2:=1+random(high(int8));

    verificar\_isbns(isbn1,isbn2);

    textcolor(green); write('La cantidad total de préstamos en el abb\_prestamos cuyo ISBN se encuentra entre '); textcolor(yellow); write(isbn1); textcolor(green); write(' y '); textcolor(yellow); write(isbn2); textcolor(green); write(' (incluídos) es '); textcolor(red); writeln(contar\_isbns1(abb\_prestamos,isbn1,isbn2));

    writeln(); textcolor(red); writeln('INCISO (j):'); writeln();

    textcolor(green); write('La cantidad total de préstamos en el abb\_isbns cuyo ISBN se encuentra entre '); textcolor(yellow); write(isbn1); textcolor(green); write(' y '); textcolor(yellow); write(isbn2); textcolor(green); write(' (incluídos) es '); textcolor(red); write(contar\_isbns2(abb\_isbns,isbn1,isbn2));

  end;

end.

**Ejercicio 3.**

*Una facultad nos ha encargado procesar la información de sus alumnos de la carrera XXX. Esta carrera tiene 30 materias. Implementar un programa con:*

**(a)** *Un módulo que lea la información de los finales rendidos por los alumnos y los almacene en dos estructuras de datos.*

**(i)** *Una estructura que, para cada alumno, se almacenen sólo código y nota de las materias aprobadas (4 a 10). De cada final rendido, se lee el código del alumno, el código de materia y la nota (valor entre 1 y 10). La lectura de los finales finaliza con nota -1. La estructura debe ser eficiente para buscar por código de alumno.*

**(ii)** *Otra estructura que almacene para cada materia, su código y todos los finales rendidos en esa materia (código de alumno y nota).*

**(b)** *Un módulo que reciba la estructura generada en (i) y un código de alumno y retorne los códigos y promedios de los alumnos cuyos códigos sean mayor al ingresado.*

**(c)** *Un módulo que reciba la estructura generada en (i), dos códigos de alumnos y un valor entero y retorne la cantidad de alumnos con cantidad de finales aprobados igual al valor ingresado para aquellos alumnos cuyos códigos están comprendidos entre los dos códigos de alumnos ingresados.*

program TP4\_E3;

*{$codepage UTF8}*

uses crt;

const

  materias\_total=30;

  nota\_corte=4;

  nota\_ini=1; nota\_fin=10;

  nota\_salida=-1;

type

  t\_materia=1..materias\_total;

  t\_nota=nota\_salida..nota\_fin;

  t\_registro\_final1=record

    codigo\_alumno: int8;

    codigo\_materia: t\_materia;

    nota: t\_nota;

  end;

  t\_vector\_notas=array[t\_materia] of t\_nota;

  t\_registro\_alumno1=record

    codigo\_alumno: int8;

    notas: t\_vector\_notas;

  end;

  t\_abb\_alumnos1=^t\_nodo\_abb\_alumnos1;

  t\_nodo\_abb\_alumnos1=record

    ele: t\_registro\_alumno1;

    hi: t\_abb\_alumnos1;

    hd: t\_abb\_alumnos1;

  end;

  t\_registro\_final2=record

    codigo\_alumno: int8;

    nota: t\_nota;

  end;

  t\_lista\_finales=^t\_nodo\_finales;

  t\_nodo\_finales=record

    ele: t\_registro\_final2;

    sig: t\_lista\_finales;

  end;

  t\_vector\_finales=array[t\_materia] of t\_lista\_finales;

  t\_registro\_alumno2=record

    codigo\_alumno: int8;

    promedio: real;

  end;

  t\_abb\_alumnos2=^t\_nodo\_abb\_alumnos2;

  t\_nodo\_abb\_alumnos2=record

    ele: t\_registro\_alumno2;

    hi: t\_abb\_alumnos2;

    hd: t\_abb\_alumnos2;

  end;

procedure inicializar\_vector\_finales(var vector\_finales: t\_vector\_finales);

var

  i: t\_materia;

begin

  for i:= 1 to materias\_total do

    vector\_finales[i]:=nil;

end;

procedure leer\_final(var registro\_final1: t\_registro\_final1);

var

  i: int8;

begin

  i:=random(100);

  if (i=0) then

    registro\_final1.nota:=nota\_salida

  else

    registro\_final1.nota:=nota\_ini+random(nota\_fin);

  if (registro\_final1.nota<>nota\_salida) then

  begin

    registro\_final1.codigo\_alumno:=1+random(high(int8));

    registro\_final1.codigo\_materia:=1+random(materias\_total);

  end;

end;

procedure inicializar\_vector\_notas(var vector\_notas: t\_vector\_notas);

var

  i: t\_materia;

begin

  for i:= 1 to materias\_total do

    vector\_notas[i]:=0;

end;

procedure cargar\_registro\_alumno1(var registro\_alumno1: t\_registro\_alumno1; registro\_final1: t\_registro\_final1);

begin

  registro\_alumno1.codigo\_alumno:=registro\_final1.codigo\_alumno;

  inicializar\_vector\_notas(registro\_alumno1.notas);

  if (registro\_final1.nota>=nota\_corte) then

    registro\_alumno1.notas[registro\_final1.codigo\_materia]:=registro\_final1.nota;

end;

procedure agregar\_abb\_alumnos1(var abb\_alumnos1: t\_abb\_alumnos1; registro\_final1: t\_registro\_final1);

begin

  if (abb\_alumnos1=nil) then

  begin

    new(abb\_alumnos1);

    cargar\_registro\_alumno1(abb\_alumnos1^.ele,registro\_final1);

    abb\_alumnos1^.hi:=nil;

    abb\_alumnos1^.hd:=nil;

  end

  else

    if (registro\_final1.codigo\_alumno=abb\_alumnos1^.ele.codigo\_alumno) then

    begin

      if (registro\_final1.nota>=nota\_corte) then

        abb\_alumnos1^.ele.notas[registro\_final1.codigo\_materia]:=registro\_final1.nota;

    end

    else if (registro\_final1.codigo\_alumno<abb\_alumnos1^.ele.codigo\_alumno) then

      agregar\_abb\_alumnos1(abb\_alumnos1^.hi,registro\_final1)

    else

      agregar\_abb\_alumnos1(abb\_alumnos1^.hd,registro\_final1);

end;

procedure cargar\_registro\_final2(var registro\_final2: t\_registro\_final2; registro\_final1: t\_registro\_final1);

begin

  registro\_final2.codigo\_alumno:=registro\_final1.codigo\_alumno;

  registro\_final2.nota:=registro\_final1.nota;

end;

procedure agregar\_adelante\_lista\_finales(var lista\_finales: t\_lista\_finales; registro\_final1: t\_registro\_final1);

var

  nuevo: t\_lista\_finales;

begin

  new(nuevo);

  cargar\_registro\_final2(nuevo^.ele,registro\_final1);

  nuevo^.sig:=lista\_finales;

  lista\_finales:=nuevo;

end;

procedure cargar\_vector\_finales(var vector\_finales: t\_vector\_finales; registro\_final1: t\_registro\_final1);

begin

  agregar\_adelante\_lista\_finales(vector\_finales[registro\_final1.codigo\_materia],registro\_final1);

end;

procedure cargar\_estructuras(var abb\_alumnos1: t\_abb\_alumnos1; var vector\_finales: t\_vector\_finales);

var

  registro\_final1: t\_registro\_final1;

begin

  leer\_final(registro\_final1);

  while (registro\_final1.nota<>nota\_salida) do

  begin

    agregar\_abb\_alumnos1(abb\_alumnos1,registro\_final1);

    cargar\_vector\_finales(vector\_finales,registro\_final1);

    leer\_final(registro\_final1);

  end;

end;

procedure imprimir\_vector\_notas(vector\_notas: t\_vector\_notas; codigo\_alumno: int8);

var

  i: t\_materia;

begin

  for i:= 1 to materias\_total do

  begin

    if (vector\_notas[i]>0) then

    begin

      textcolor(green); write('La nota de la materia '); textcolor(yellow); write(i); textcolor(green); write(' del código de alumno '); textcolor(yellow); write(codigo\_alumno); textcolor(green); write(' es '); textcolor(red); writeln(vector\_notas[i]);

    end;

  end;

end;

procedure imprimir\_registro\_alumno1(registro\_alumno1: t\_registro\_alumno1);

begin

  textcolor(green); write('El código de alumno del alumno es '); textcolor(red); writeln(registro\_alumno1.codigo\_alumno);

  imprimir\_vector\_notas(registro\_alumno1.notas,registro\_alumno1.codigo\_alumno);

  writeln();

end;

procedure imprimir\_abb\_alumnos1(abb\_alumnos1: t\_abb\_alumnos1);

begin

  if (abb\_alumnos1<>nil) then

  begin

    imprimir\_abb\_alumnos1(abb\_alumnos1^.hi);

    imprimir\_registro\_alumno1(abb\_alumnos1^.ele);

    imprimir\_abb\_alumnos1(abb\_alumnos1^.hd);

  end;

end;

procedure imprimir\_registro\_final2(registro\_final2: t\_registro\_final2; materia: t\_materia; final: int16);

begin

  textcolor(green); write('El código de alumno del final '); textcolor(yellow); write(final); textcolor(green); write(' de la materia '); textcolor(yellow); write(materia); textcolor(green); write(' es '); textcolor(red); writeln(registro\_final2.codigo\_alumno);

  textcolor(green); write('La nota del final '); textcolor(yellow); write(final); textcolor(green); write(' de la materia '); textcolor(yellow); write(materia); textcolor(green); write(' es '); textcolor(red); writeln(registro\_final2.nota);

end;

procedure imprimir\_lista\_finales(lista\_finales: t\_lista\_finales; materia: t\_materia);

var

  i: int16;

begin

  i:=0;

  while (lista\_finales<>nil) do

  begin

    i:=i+1;

    imprimir\_registro\_final2(lista\_finales^.ele,materia,i);

    lista\_finales:=lista\_finales^.sig;

  end;

end;

procedure imprimir\_vector\_finales(vector\_finales: t\_vector\_finales);

var

  i: t\_materia;

begin

  for i:= 1 to materias\_total do

  begin

    textcolor(green); write('Los finales rendidos de la materia '); textcolor(yellow); write(i); textcolor(green); writeln(' son:');

    imprimir\_lista\_finales(vector\_finales[i],i);

    writeln();

  end;

end;

function calcular\_promedio(vector\_notas: t\_vector\_notas): real;

var

  i: t\_materia;

  notas\_total, notas: int16;

begin

  notas\_total:=0; notas:=0;

  for i:= 1 to materias\_total do

    if (vector\_notas[i]>=nota\_corte) then

    begin

      notas\_total:=notas\_total+vector\_notas[i];

      notas:=notas+1;

    end;

  if (notas>0) then

    calcular\_promedio:=notas\_total/notas

  else

    calcular\_promedio:=notas\_total;

end;

procedure cargar\_registro\_alumno2(var registro\_alumno2: t\_registro\_alumno2; registro\_alumno1: t\_registro\_alumno1);

begin

  registro\_alumno2.codigo\_alumno:=registro\_alumno1.codigo\_alumno;

  registro\_alumno2.promedio:=calcular\_promedio(registro\_alumno1.notas);

end;

procedure agregar\_abb\_alumnos2(var abb\_alumnos2: t\_abb\_alumnos2; registro\_alumno1: t\_registro\_alumno1);

begin

  if (abb\_alumnos2=nil) then

  begin

    new(abb\_alumnos2);

    cargar\_registro\_alumno2(abb\_alumnos2^.ele,registro\_alumno1);

    abb\_alumnos2^.hi:=nil;

    abb\_alumnos2^.hd:=nil;

  end

  else

    if (registro\_alumno1.codigo\_alumno<=abb\_alumnos2^.ele.codigo\_alumno) then

      agregar\_abb\_alumnos2(abb\_alumnos2^.hi,registro\_alumno1)

    else

      agregar\_abb\_alumnos2(abb\_alumnos2^.hd,registro\_alumno1);

end;

procedure cargar\_abb\_alumnos2(var abb\_alumnos2: t\_abb\_alumnos2; abb\_alumnos1: t\_abb\_alumnos1; codigo: int8);

begin

  if (abb\_alumnos1<>nil) then

  begin

    if (abb\_alumnos1^.ele.codigo\_alumno>codigo) then

    begin

      cargar\_abb\_alumnos2(abb\_alumnos2,abb\_alumnos1^.hi,codigo);

      agregar\_abb\_alumnos2(abb\_alumnos2,abb\_alumnos1^.ele);

      cargar\_abb\_alumnos2(abb\_alumnos2,abb\_alumnos1^.hd,codigo);

    end

    else

      cargar\_abb\_alumnos2(abb\_alumnos2,abb\_alumnos1^.hd,codigo);;

  end;

end;

procedure imprimir\_registro\_alumno2(registro\_alumno2: t\_registro\_alumno2);

begin

  textcolor(green); write('El código de alumno del alumno es '); textcolor(red); writeln(registro\_alumno2.codigo\_alumno);

  textcolor(green); write('El promedio del alumno es '); textcolor(red); writeln(registro\_alumno2.promedio:0:2);

  writeln();

end;

procedure imprimir\_abb\_alumnos2(abb\_alumnos2: t\_abb\_alumnos2);

begin

  if (abb\_alumnos2<>nil) then

  begin

    imprimir\_abb\_alumnos2(abb\_alumnos2^.hi);

    imprimir\_registro\_alumno2(abb\_alumnos2^.ele);

    imprimir\_abb\_alumnos2(abb\_alumnos2^.hd);

  end;

end;

procedure verificar\_codigos(var codigo1, codigo2: int8);

var

  aux: int8;

begin

  if (codigo1>codigo2) then

  begin

    aux:=codigo1;

    codigo1:=codigo2;

    codigo2:=aux;

  end;

end;

function contar\_notas(vector\_notas: t\_vector\_notas; finales: t\_materia): int8;

var

  i: t\_materia;

  notas: int8;

begin

  notas:=0;

  for i:= 1 to materias\_total do

    if (vector\_notas[i]>=nota\_corte) then

      notas:=notas+1;

  if (notas=finales) then

    contar\_notas:=1

  else

    contar\_notas:=0;

end;

function contar\_alumnos(abb\_alumnos1: t\_abb\_alumnos1; codigo1, codigo2: int16; finales: t\_materia): int16;

begin

  if (abb\_alumnos1=nil) then

    contar\_alumnos:=0

  else

    if (codigo1>=abb\_alumnos1^.ele.codigo\_alumno) then

      contar\_alumnos:=contar\_alumnos(abb\_alumnos1^.hd,codigo1,codigo2,finales)

    else if (codigo2<=abb\_alumnos1^.ele.codigo\_alumno) then

      contar\_alumnos:=contar\_alumnos(abb\_alumnos1^.hi,codigo1,codigo2,finales)

    else

      contar\_alumnos:=contar\_alumnos(abb\_alumnos1^.hi,codigo1,codigo2,finales)+contar\_alumnos(abb\_alumnos1^.hd,codigo1,codigo2,finales)+contar\_notas(abb\_alumnos1^.ele.notas,finales);

end;

var

  vector\_finales: t\_vector\_finales;

  abb\_alumnos1: t\_abb\_alumnos1;

  abb\_alumnos2: t\_abb\_alumnos2;

  finales: t\_materia;

  codigo, codigo1, codigo2: int8;

begin

  randomize;

  abb\_alumnos1:=nil; inicializar\_vector\_finales(vector\_finales);

  abb\_alumnos2:=nil;

  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();

  cargar\_estructuras(abb\_alumnos1,vector\_finales);

  if (abb\_alumnos1<>nil) then

  begin

    writeln(); textcolor(red); writeln('ABB\_ALUMNOS1:'); writeln();

    imprimir\_abb\_alumnos1(abb\_alumnos1);

    writeln(); textcolor(red); writeln('VECTOR\_FINALES:'); writeln();

    imprimir\_vector\_finales(vector\_finales);

    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();

    codigo:=1+random(high(int8));

    cargar\_abb\_alumnos2(abb\_alumnos2,abb\_alumnos1,codigo);

    if (abb\_alumnos2<>nil) then

      imprimir\_abb\_alumnos2(abb\_alumnos2);

    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();

    codigo1:=1+random(high(int8)); codigo2:=1+random(high(int8)); finales:=2;

    verificar\_codigos(codigo1,codigo2);

    textcolor(green); write('La cantidad de alumnos en el abb cuyo código de alumno se encuentra entre '); textcolor(yellow); write(codigo1); textcolor(green); write(' y '); textcolor(yellow); write(codigo2); textcolor(green); write(' y tienen '); textcolor(yellow); write(finales); textcolor(green); write(' finales aprobados es '); textcolor(red); write(contar\_alumnos(abb\_alumnos1,codigo1,codigo2,finales));

  end;

end.