Juan Menduiña

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.cant
idad;
      abb_productos^.ele.monto_total:=abb_productos^.ele.monto_total+registro_venta.cantidad*r
egistro_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^.h
d,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_t
otal(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,soc
io)+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,isbn
1,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_final
1);
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.
```