**Trabajo Práctico N° 3:**

**Módulo Imperativo (Árboles 1).**

**Ejercicio 1.**

*Escribir un programa que:*

**(a)** *Implemente un módulo que lea información de socios de un club y las almacene en un árbol binario de búsqueda. De cada socio, se lee número de socio, nombre y edad. La lectura finaliza con el número de socio 0 y el árbol debe quedar ordenado por número de socio.*

**(b)** *Una vez generado el árbol, realice módulos independientes que reciban el árbol como*

*parámetro y que:*

**(i)** *Informe el número de socio más grande. Debe invocar a un módulo recursivo que retorne dicho valor.*

**(ii)** *Informe los datos del socio con el número de socio más chico. Debe invocar a un módulo recursivo que retorne dicho socio.*

**(iii)** *Informe el número de socio con mayor edad. Debe invocar a un módulo recursivo que retorne dicho valor.*

**(iv)** *Aumente en 1 la edad de todos los socios.*

**(v)** *Lea un valor entero e informe si existe o no existe un socio con ese valor. Debe invocar a un módulo recursivo que reciba el valor leído y retorne verdadero o falso.*

**(vi)** *Lea un nombre e informe si existe o no existe un socio con ese nombre. Debe invocar a un módulo recursivo que reciba el nombre leído y retorne verdadero o falso.*

**(vii)** *Informe la cantidad de socios. Debe invocar a un módulo recursivo que retorne dicha cantidad.*

**(viii)** *Informe el promedio de edad de los socios. Debe invocar al módulo recursivo del inciso (vii) e invocar a un módulo recursivo que retorne la suma de las edades de los socios.*

**(ix)** *Informe, a partir de dos valores que se leen, la cantidad de socios en el árbol cuyo número de socio se encuentra entre los dos valores ingresados. Debe invocar a un módulo recursivo que reciba los dos valores leídos y retorne dicha cantidad.*

**(x)** *Informe los números de socio en orden creciente.*

**(xi)** *Informe los números de socio pares en orden decreciente.*

program TP3\_E1;

*{$codepage UTF8}*

uses crt;

const

  numero\_salida=0;

type

  t\_registro\_socio=record

    numero: int16;

    nombre: string;

    edad: int8;

  end;

  t\_abb\_socios=^t\_nodo\_abb\_socios;

  t\_nodo\_abb\_socios=record

    ele: t\_registro\_socio;

    hi: t\_abb\_socios;

    hd: t\_abb\_socios;

  end;

function random\_string(length: int8): string;

var

  i: int8;

  string\_aux: string;

begin

  string\_aux:='';

  for i:= 1 to length do

    string\_aux:=string\_aux+chr(ord('A')+random(26));

  random\_string:=string\_aux;

end;

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

var

  i: int8;

begin

  i:=random(100);

  if (i=0) then

    registro\_socio.numero:=numero\_salida

  else

    registro\_socio.numero:=1+random(high(int16));

  if (registro\_socio.numero<>numero\_salida) then

  begin

    registro\_socio.nombre:=random\_string(5+random(6));

    registro\_socio.edad:=1+random(high(int8)-1);

  end;

end;

procedure agregar\_abb\_socios(var abb\_socios: t\_abb\_socios; registro\_socio: t\_registro\_socio);

begin

  if (abb\_socios=nil) then

  begin

    new(abb\_socios);

    abb\_socios^.ele:=registro\_socio;

    abb\_socios^.hi:=nil;

    abb\_socios^.hd:=nil;

  end

  else

    if (registro\_socio.numero<=abb\_socios^.ele.numero) then

      agregar\_abb\_socios(abb\_socios^.hi,registro\_socio)

    else

      agregar\_abb\_socios(abb\_socios^.hd,registro\_socio);

end;

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

var

  registro\_socio: t\_registro\_socio;

begin

  leer\_socio(registro\_socio);

  while (registro\_socio.numero<>numero\_salida) do

  begin

    agregar\_abb\_socios(abb\_socios,registro\_socio);

    leer\_socio(registro\_socio);

  end;

end;

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

begin

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

  textcolor(green); write('El nombre del socio es '); textcolor(red); writeln(registro\_socio.nombre);

  textcolor(green); write('La edad del socio es '); textcolor(red); writeln(registro\_socio.edad);

  writeln();

end;

procedure imprimir1\_abb\_socios(abb\_socios: t\_abb\_socios);

begin

  if (abb\_socios<>nil) then

  begin

    imprimir1\_abb\_socios(abb\_socios^.hi);

    imprimir\_registro\_socio(abb\_socios^.ele);

    imprimir1\_abb\_socios(abb\_socios^.hd);

  end;

end;

function buscar\_mayor\_numero(abb\_socios: t\_abb\_socios): int16;

begin

  if (abb\_socios^.hd=nil) then

    buscar\_mayor\_numero:=abb\_socios^.ele.numero

  else

    buscar\_mayor\_numero:=buscar\_mayor\_numero(abb\_socios^.hd);

end;

function buscar\_menor\_numero(abb\_socios: t\_abb\_socios): int16;

begin

  if (abb\_socios^.hi=nil) then

    buscar\_menor\_numero:=abb\_socios^.ele.numero

  else

    buscar\_menor\_numero:=buscar\_menor\_numero(abb\_socios^.hi);

end;

procedure buscar\_numero\_mayor\_edad(abb\_socios: t\_abb\_socios; var edad\_max: int8; var numero\_max: int16);

begin

  if (abb\_socios<>nil) then

  begin

    buscar\_numero\_mayor\_edad(abb\_socios^.hi,edad\_max,numero\_max);

    if (abb\_socios^.ele.edad>edad\_max) then

    begin

      edad\_max:=abb\_socios^.ele.edad;

      numero\_max:=abb\_socios^.ele.numero;

    end;

    buscar\_numero\_mayor\_edad(abb\_socios^.hd,edad\_max,numero\_max);

  end;

end;

procedure aumentar\_edad(var abb\_socios: t\_abb\_socios);

begin

  if (abb\_socios<>nil) then

  begin

    aumentar\_edad(abb\_socios^.hi);

    abb\_socios^.ele.edad:=abb\_socios^.ele.edad+1;

    aumentar\_edad(abb\_socios^.hd);

  end;

end;

function buscar\_numero(abb\_socios: t\_abb\_socios; numero: int16): boolean;

begin

  if (abb\_socios=nil) then

    buscar\_numero:=false

  else

    if (numero=abb\_socios^.ele.numero) then

      buscar\_numero:=true

    else if (numero<abb\_socios^.ele.numero) then

      buscar\_numero:=buscar\_numero(abb\_socios^.hi,numero)

    else

      buscar\_numero:=buscar\_numero(abb\_socios^.hd,numero);

end;

function buscar\_nombre(abb\_socios: t\_abb\_socios; nombre: string): boolean;

begin

  if (abb\_socios=nil) then

    buscar\_nombre:=false

  else

    if (nombre=abb\_socios^.ele.nombre) then

      buscar\_nombre:=true

    else

      buscar\_nombre:=buscar\_nombre(abb\_socios^.hi,nombre) or buscar\_nombre(abb\_socios^.hd,nombre);

end;

function contar\_socios1(abb\_socios: t\_abb\_socios): int16;

begin

  if (abb\_socios=nil) then

    contar\_socios1:=0

  else

    contar\_socios1:=contar\_socios1(abb\_socios^.hi)+contar\_socios1(abb\_socios^.hd)+1;

end;

function contar\_edades(abb\_socios: t\_abb\_socios): int16;

begin

  if (abb\_socios=nil) then

    contar\_edades:=0

  else

    contar\_edades:=contar\_edades(abb\_socios^.hi)+contar\_edades(abb\_socios^.hd)+abb\_socios^.ele.edad;

end;

function calcular\_edad\_promedio(abb\_socios: t\_abb\_socios): real;

begin

  calcular\_edad\_promedio:=contar\_edades(abb\_socios)/contar\_socios1(abb\_socios);

end;

procedure verificar\_numeros(var numero1, numero2: int16);

var

  aux: int8;

begin

  if (numero1>numero2) then

  begin

    aux:=numero1;

    numero1:=numero2;

    numero2:=aux;

  end;

end;

function contar\_socios2(abb\_socios: t\_abb\_socios; numero1, numero2: int16): int16;

begin

  if (abb\_socios=nil) then

    contar\_socios2:=0

  else

    if (numero1>=abb\_socios^.ele.numero) then

      contar\_socios2:=contar\_socios2(abb\_socios^.hd,numero1,numero2)

    else if (numero2<=abb\_socios^.ele.numero) then

      contar\_socios2:=contar\_socios2(abb\_socios^.hi,numero1,numero2)

    else

      contar\_socios2:=contar\_socios2(abb\_socios^.hi,numero1,numero2)+contar\_socios2(abb\_socios^.hd,numero1,numero2)+1;

end;

procedure imprimir2\_abb\_socios(abb\_socios: t\_abb\_socios);

begin

  if (abb\_socios<>nil) then

  begin

    imprimir2\_abb\_socios(abb\_socios^.hi);

    textcolor(green); write('Número de socio: '); textcolor(red); writeln(abb\_socios^.ele.numero);

    imprimir2\_abb\_socios(abb\_socios^.hd);

  end;

end;

procedure imprimir3\_abb\_socios(abb\_socios: t\_abb\_socios);

begin

  if (abb\_socios<>nil) then

  begin

    imprimir3\_abb\_socios(abb\_socios^.hd);

    if (abb\_socios^.ele.numero mod 2=0) then

    begin

      textcolor(green); write('Número de socio: '); textcolor(red); writeln(abb\_socios^.ele.numero);

    end;

    imprimir3\_abb\_socios(abb\_socios^.hi);

  end;

end;

var

  abb\_socios: t\_abb\_socios;

  edad\_max: int8;

  numero\_max, numero, numero1, numero2: int16;

  nombre: string;

begin

  randomize;

  abb\_socios:=nil;

  edad\_max:=low(int8); numero\_max:=numero\_salida;

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

  cargar\_abb\_socios(abb\_socios);

  if (abb\_socios<>nil) then

  begin

    imprimir1\_abb\_socios(abb\_socios);

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

    textcolor(green); write('El número de socio más grande es '); textcolor(red); writeln(buscar\_mayor\_numero(abb\_socios));

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

    textcolor(green); write('El número de socio más chico es '); textcolor(red); writeln(buscar\_menor\_numero(abb\_socios));

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

    buscar\_numero\_mayor\_edad(abb\_socios,edad\_max,numero\_max);

    textcolor(green); write('El número de socio con mayor edad es '); textcolor(red); writeln(numero\_max);

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

    aumentar\_edad(abb\_socios);

    imprimir1\_abb\_socios(abb\_socios);

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

    numero:=1+random(high(int16));

    textcolor(green); write('¿El número de socio '); textcolor(yellow); write(numero); textcolor(green); write(' se encuentra en el abb?: '); textcolor(red); writeln(buscar\_numero(abb\_socios,numero));

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

    nombre:=random\_string(5+random(6));

    textcolor(green); write('¿El nombre de socio '); textcolor(yellow); write(nombre); textcolor(green); write(' se encuentra en el abb?: '); textcolor(red); writeln(buscar\_nombre(abb\_socios,nombre));

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

    textcolor(green); write('La cantidad de socios es '); textcolor(red); writeln(contar\_socios1(abb\_socios));

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

    textcolor(green); write('El promedio de edad de los socios es '); textcolor(red); writeln(calcular\_edad\_promedio(abb\_socios):0:2);

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

    numero1:=1+random(high(int16)); numero2:=1+random(high(int16));

    verificar\_numeros(numero1,numero2);

    textcolor(green); write('La cantidad de socios en el abb cuyo número de socio se encuentra entre '); textcolor(yellow); write(numero1); textcolor(green); write(' y '); textcolor(yellow); write(numero2); textcolor(green); write(' es '); textcolor(red); writeln(contar\_socios2(abb\_socios,numero1,numero2));

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

    imprimir2\_abb\_socios(abb\_socios);

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

    imprimir3\_abb\_socios(abb\_socios);

  end;

end.

**Ejercicio 2.**

*Escribir un programa que:*

**(a)** *Implemente un módulo que lea información de ventas de un comercio. De cada venta, se lee código de producto, fecha y cantidad de unidades vendidas. La lectura finaliza con el código de producto 0. Un producto puede estar en más de una venta. Se pide:*

**(i)** *Generar y retornar un árbol binario de búsqueda de ventas ordenado por código de producto.*

**(ii)** *Generar y retornar otro árbol binario de búsqueda de productos vendidos ordenado por código de producto. Cada nodo del árbol debe contener el código de producto y la cantidad total de unidades vendidas.*

*Nota: El módulo debe retornar los dos árboles.*

**(b)** *Implemente un módulo que reciba el árbol generado en (i) y un código de producto y retorne la cantidad total de unidades vendidas de ese producto.*

**(c)** *Implemente un módulo que reciba el árbol generado en (ii) y un código de producto y retorne la cantidad total de unidades vendidas de ese producto.*

program TP3\_E2;

*{$codepage UTF8}*

uses crt;

const

  codigo\_salida=0;

type

  t\_registro\_venta=record

    codigo: int8;

    fecha: int8;

    cantidad: int8;

  end;

  t\_registro\_producto=record

    codigo: int8;

    cantidad: int16;

  end;

  t\_abb\_ventas=^t\_nodo\_abb\_ventas;

  t\_nodo\_abb\_ventas=record

    ele: t\_registro\_venta;

    hi: t\_abb\_ventas;

    hd: t\_abb\_ventas;

  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:=codigo\_salida

  else

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

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

  begin

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

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

  end;

end;

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

begin

  if (abb\_ventas=nil) then

  begin

    new(abb\_ventas);

    abb\_ventas^.ele:=registro\_venta;

    abb\_ventas^.hi:=nil;

    abb\_ventas^.hd:=nil;

  end

  else

    if (registro\_venta.codigo<=abb\_ventas^.ele.codigo) then

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

    else

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

end;

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

begin

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

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

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=abb\_productos^.ele.codigo) then

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

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

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

    else

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

end;

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

var

  registro\_venta: t\_registro\_venta;

begin

  leer\_venta(registro\_venta);

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

  begin

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

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

    leer\_venta(registro\_venta);

  end;

end;

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

begin

  textcolor(green); write('El código de producto de la venta es '); textcolor(red); writeln(registro\_venta.codigo);

  textcolor(green); write('La fecha de la venta es '); textcolor(red); writeln(registro\_venta.fecha);

  textcolor(green); write('La cantidad de unidades vendidas de la venta es '); textcolor(red); writeln(registro\_venta.cantidad);

  writeln();

end;

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

begin

  if (abb\_ventas<>nil) then

  begin

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

    imprimir\_registro\_venta(abb\_ventas^.ele);

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

  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);

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

  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;

function contar\_abb\_ventas(abb\_ventas: t\_abb\_ventas; codigo: int8): int16;

begin

  if (abb\_ventas=nil) then

    contar\_abb\_ventas:=0

  else

    if (codigo=abb\_ventas^.ele.codigo) then

      contar\_abb\_ventas:=contar\_abb\_ventas(abb\_ventas^.hi,codigo)+abb\_ventas^.ele.cantidad

    else if (codigo<abb\_ventas^.ele.codigo) then

      contar\_abb\_ventas:=contar\_abb\_ventas(abb\_ventas^.hi,codigo)

    else

      contar\_abb\_ventas:=contar\_abb\_ventas(abb\_ventas^.hd,codigo);

end;

function contar\_abb\_productos(abb\_productos: t\_abb\_productos; codigo: int8): int16;

begin

  if (abb\_productos=nil) then

    contar\_abb\_productos:=0

  else

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

      contar\_abb\_productos:=abb\_productos^.ele.cantidad

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

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

    else

      contar\_abb\_productos:=contar\_abb\_productos(abb\_productos^.hd,codigo);

end;

var

  abb\_ventas: t\_abb\_ventas;

  abb\_productos: t\_abb\_productos;

  codigo: int8;

begin

  randomize;

  abb\_ventas:=nil; abb\_productos:=nil;

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

  cargar\_abbs(abb\_ventas,abb\_productos);

  if ((abb\_ventas<>nil) and (abb\_productos<>nil)) then

  begin

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

    imprimir\_abb\_ventas(abb\_ventas);

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

    imprimir\_abb\_productos(abb\_productos);

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

    codigo:=1+random(high(int8));

    textcolor(green); write('La cantidad total de unidades vendidas en el abb\_ventas del código de producto '); textcolor(yellow); write(codigo); textcolor(green); write(' es '); textcolor(red); writeln(contar\_abb\_ventas(abb\_ventas,codigo));

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

    textcolor(green); write('La cantidad total de unidades vendidas en el abb\_productos del código de producto '); textcolor(yellow); write(codigo); textcolor(green); write(' es '); textcolor(red); write(contar\_abb\_productos(abb\_productos,codigo));

  end;

end.

**Ejercicio 3.**

*Implementar un programa que contenga:*

**(a)** *Un módulo que lea información de alumnos de Taller de Programación y los almacene en una estructura de datos. De cada alumno, se lee legajo, DNI, año de ingreso y los códigos y notas de los finales rendidos. La estructura generada debe ser eficiente para la búsqueda por número de legajo. La lectura de los alumnos finaliza con legajo 0 y, para cada alumno, el ingreso de las materias finaliza con el código de materia -1.*

**(b)** *Un módulo que reciba la estructura generada en (a) y retorne los DNI y año de ingreso de aquellos alumnos cuyo legajo sea inferior a un valor ingresado como parámetro.*

**(c)** *Un módulo que reciba la estructura generada en (a) y retorne el legajo más grande.*

**(d)** *Un módulo que reciba la estructura generada en (a) y retorne el DNI más grande.*

**(e)** *Un módulo que reciba la estructura generada en (a) y retorne la cantidad de alumnos con legajo impar.*

**(f)** *Un módulo que reciba la estructura generada en (a) y retorne el legajo y el promedio del alumno con mayor promedio.*

**(g)** *Un módulo que reciba la estructura generada en (a) y un valor entero. Este módulo debe retornar los legajos y promedios de los alumnos cuyo promedio supera el valor ingresado.*

program TP3\_E3;

*{$codepage UTF8}*

uses crt;

const

  nota\_ini=1; nota\_fin=10;

  legajo\_salida=0; codigo\_salida=-1;

type

  t\_nota=nota\_ini..nota\_fin;

  t\_registro\_final=record

    codigo: int8;

    nota: t\_nota;

  end;

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

  t\_nodo\_finales=record

    ele: t\_registro\_final;

    sig: t\_lista\_finales;

  end;

  t\_registro\_alumno1=record

    legajo: int16;

    dni: int32;

    anio\_ingreso: int16;

    finales: t\_lista\_finales;

  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\_alumno2=record

    dni: int32;

    anio\_ingreso: int16;

  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;

  t\_registro\_alumno3=record

    legajo: int16;

    promedio: real;

  end;

  t\_abb\_alumnos3=^t\_nodo\_abb\_alumnos3;

  t\_nodo\_abb\_alumnos3=record

    ele: t\_registro\_alumno3;

    hi: t\_abb\_alumnos3;

    hd: t\_abb\_alumnos3;

  end;

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

var

  i: int8;

begin

  i:=random(10);

  if (i=0) then

    registro\_final.codigo:=codigo\_salida

  else

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

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

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

end;

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

var

  nuevo: t\_lista\_finales;

begin

  new(nuevo);

  nuevo^.ele:=registro\_final;

  nuevo^.sig:=lista\_finales;

  lista\_finales:=nuevo;

end;

procedure leer\_finales(var lista\_finales: t\_lista\_finales);

var

  registro\_final: t\_registro\_final;

begin

  leer\_final(registro\_final);

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

  begin

    agregar\_adelante\_lista\_finales(lista\_finales,registro\_final);

    leer\_final(registro\_final);

  end;

end;

procedure leer\_alumno(var registro\_alumno1: t\_registro\_alumno1);

var

  i: int8;

begin

  i:=random(100);

  if (i=0) then

    registro\_alumno1.legajo:=legajo\_salida

  else

    registro\_alumno1.legajo:=1+random(high(int16));

  if (registro\_alumno1.legajo<>legajo\_salida) then

  begin

    registro\_alumno1.dni:=10000000+random(40000001);

    registro\_alumno1.anio\_ingreso:=2000+random(25);

    registro\_alumno1.finales:=nil;

    leer\_finales(registro\_alumno1.finales);

  end;

end;

procedure agregar\_abb\_alumnos1(var abb\_alumnos1: t\_abb\_alumnos1; registro\_alumno1: t\_registro\_alumno1);

begin

  if (abb\_alumnos1=nil) then

  begin

    new(abb\_alumnos1);

    abb\_alumnos1^.ele:=registro\_alumno1;

    abb\_alumnos1^.hi:=nil;

    abb\_alumnos1^.hd:=nil;

  end

  else

    if (registro\_alumno1.legajo<=abb\_alumnos1^.ele.legajo) then

      agregar\_abb\_alumnos1(abb\_alumnos1^.hi,registro\_alumno1)

    else

      agregar\_abb\_alumnos1(abb\_alumnos1^.hd,registro\_alumno1);

end;

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

var

  registro\_alumno1: t\_registro\_alumno1;

begin

  leer\_alumno(registro\_alumno1);

  while (registro\_alumno1.legajo<>legajo\_salida) do

  begin

    agregar\_abb\_alumnos1(abb\_alumnos1,registro\_alumno1);

    leer\_alumno(registro\_alumno1);

  end;

end;

procedure imprimir\_registro\_final(registro\_final: t\_registro\_final; legajo, final: int16);

begin

  textcolor(green); write('El código del final '); textcolor(yellow); write(final); textcolor(green); write(' del legajo '); textcolor(yellow); write(legajo); textcolor(green); write(' es '); textcolor(red); writeln(registro\_final.codigo);

  textcolor(green); write('La nota del final '); textcolor(yellow); write(final); textcolor(green); write(' del legajo '); textcolor(yellow); write(legajo); textcolor(green); write(' es '); textcolor(red); writeln(registro\_final.nota);

end;

procedure imprimir\_lista\_finales(lista\_finales: t\_lista\_finales; legajo: int16);

var

  i: int16;

begin

  i:=0;

  while (lista\_finales<>nil) do

  begin

    i:=i+1;

    imprimir\_registro\_final(lista\_finales^.ele,legajo,i);

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

  end;

end;

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

begin

  textcolor(green); write('El legajo del alumno es '); textcolor(red); writeln(registro\_alumno1.legajo);

  textcolor(green); write('El DNI del alumno es '); textcolor(red); writeln(registro\_alumno1.dni);

  textcolor(green); write('El año de ingreso del alumno es '); textcolor(red); writeln(registro\_alumno1.anio\_ingreso);

  imprimir\_lista\_finales(registro\_alumno1.finales,registro\_alumno1.legajo);

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

begin

  registro\_alumno2.dni:=registro\_alumno1.dni;

  registro\_alumno2.anio\_ingreso:=registro\_alumno1.anio\_ingreso;

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.dni<=abb\_alumnos2^.ele.dni) 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; legajo: int16);

begin

  if (abb\_alumnos1<>nil) then

  begin

    if (abb\_alumnos1^.ele.legajo<legajo) then

    begin

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

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

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

    end

    else

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

  end;

end;

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

begin

  textcolor(green); write('El DNI del alumno es '); textcolor(red); writeln(registro\_alumno2.dni);

  textcolor(green); write('El año de ingreso del alumno es '); textcolor(red); writeln(registro\_alumno2.anio\_ingreso);

  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;

function buscar\_mayor\_legajo(abb\_alumnos1: t\_abb\_alumnos1): int16;

begin

  if (abb\_alumnos1^.hd=nil) then

    buscar\_mayor\_legajo:=abb\_alumnos1^.ele.legajo

  else

    buscar\_mayor\_legajo:=buscar\_mayor\_legajo(abb\_alumnos1^.hd);

end;

procedure buscar\_mayor\_dni(abb\_alumnos1: t\_abb\_alumnos1; var dni\_max: int32);

begin

  if (abb\_alumnos1<>nil) then

  begin

    buscar\_mayor\_dni(abb\_alumnos1^.hi,dni\_max);

    if (abb\_alumnos1^.ele.dni>dni\_max) then

      dni\_max:=abb\_alumnos1^.ele.dni;

    buscar\_mayor\_dni(abb\_alumnos1^.hd,dni\_max);

  end;

end;

procedure contar\_legajos\_impar(abb\_alumnos1: t\_abb\_alumnos1; var legajos\_impar: int16);

begin

  if (abb\_alumnos1<>nil) then

  begin

    contar\_legajos\_impar(abb\_alumnos1^.hi,legajos\_impar);

    if (abb\_alumnos1^.ele.legajo mod 2<>0) then

      legajos\_impar:=legajos\_impar+1;

    contar\_legajos\_impar(abb\_alumnos1^.hd,legajos\_impar);

  end;

end;

function calcular\_promedio(lista\_finales: t\_lista\_finales): real;

var

  notas\_total, notas: int16;

begin

  notas\_total:=0; notas:=0;

  while (lista\_finales<>nil) do

  begin

    notas\_total:=notas\_total+lista\_finales^.ele.nota;

    notas:=notas+1;

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

  end;

  if (notas>0) then

    calcular\_promedio:=notas\_total/notas

  else

    calcular\_promedio:=notas\_total;

end;

procedure buscar\_legajo\_mayor\_promedio(abb\_alumnos1: t\_abb\_alumnos1; var promedio\_max: real; var legajo\_max: int16);

var

  promedio: real;

begin

  if (abb\_alumnos1<>nil) then

  begin

    buscar\_legajo\_mayor\_promedio(abb\_alumnos1^.hi,promedio\_max,legajo\_max);

    promedio:=calcular\_promedio(abb\_alumnos1^.ele.finales);

    if (promedio>promedio\_max) then

    begin

      promedio\_max:=promedio;

      legajo\_max:=abb\_alumnos1^.ele.legajo;

    end;

    buscar\_legajo\_mayor\_promedio(abb\_alumnos1^.hd,promedio\_max,legajo\_max);

  end;

end;

procedure cargar\_registro\_alumno3(var registro\_alumno3: t\_registro\_alumno3; legajo: int16; promedio\_alumno: real);

begin

  registro\_alumno3.legajo:=legajo;

  registro\_alumno3.promedio:=promedio\_alumno;

end;

procedure agregar\_abb\_alumnos3(var abb\_alumnos3: t\_abb\_alumnos3; legajo: int16; promedio\_alumno: real);

begin

  if (abb\_alumnos3=nil) then

  begin

    new(abb\_alumnos3);

    cargar\_registro\_alumno3(abb\_alumnos3^.ele,legajo,promedio\_alumno);

    abb\_alumnos3^.hi:=nil;

    abb\_alumnos3^.hd:=nil;

  end

  else

    if (legajo<=abb\_alumnos3^.ele.legajo) then

      agregar\_abb\_alumnos3(abb\_alumnos3^.hi,legajo,promedio\_alumno)

    else

      agregar\_abb\_alumnos3(abb\_alumnos3^.hd,legajo,promedio\_alumno);

end;

procedure cargar\_abb\_alumnos3(var abb\_alumnos3: t\_abb\_alumnos3; abb\_alumnos1: t\_abb\_alumnos1; promedio: real);

var

  promedio\_alumno: real;

begin

  if (abb\_alumnos1<>nil) then

  begin

    cargar\_abb\_alumnos3(abb\_alumnos3,abb\_alumnos1^.hi,promedio);

    promedio\_alumno:=calcular\_promedio(abb\_alumnos1^.ele.finales);

    if (promedio\_alumno>promedio) then

      agregar\_abb\_alumnos3(abb\_alumnos3,abb\_alumnos1^.ele.legajo,promedio\_alumno);

    cargar\_abb\_alumnos3(abb\_alumnos3,abb\_alumnos1^.hd,promedio);

  end;

end;

procedure imprimir\_registro\_alumno3(registro\_alumno3: t\_registro\_alumno3);

begin

  textcolor(green); write('El legajo del alumno es '); textcolor(red); writeln(registro\_alumno3.legajo);

  textcolor(green); write('El promedio del alumno es '); textcolor(red); writeln(registro\_alumno3.promedio:0:2);

  writeln();

end;

procedure imprimir\_abb\_alumnos3(abb\_alumnos3: t\_abb\_alumnos3);

begin

  if (abb\_alumnos3<>nil) then

  begin

    imprimir\_abb\_alumnos3(abb\_alumnos3^.hi);

    imprimir\_registro\_alumno3(abb\_alumnos3^.ele);

    imprimir\_abb\_alumnos3(abb\_alumnos3^.hd);

  end;

end;

var

  abb\_alumnos1: t\_abb\_alumnos1;

  abb\_alumnos2: t\_abb\_alumnos2;

  abb\_alumnos3: t\_abb\_alumnos3;

  legajo, legajos\_impar, legajo\_max: int16;

  dni\_max: int32;

  promedio\_max, promedio: real;

begin

  randomize;

  abb\_alumnos1:=nil;

  abb\_alumnos2:=nil;

  dni\_max:=low(int32);

  legajos\_impar:=0;

  promedio\_max:=-9999999; legajo\_max:=0;

  abb\_alumnos3:=nil;

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

  cargar\_abb\_alumnos1(abb\_alumnos1);

  if (abb\_alumnos1<>nil) then

  begin

    imprimir\_abb\_alumnos1(abb\_alumnos1);

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

    legajo:=1+random(high(int16));

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

    if (abb\_alumnos2<>nil) then

      imprimir\_abb\_alumnos2(abb\_alumnos2);

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

    textcolor(green); write('El legajo más grande es '); textcolor(red); writeln(buscar\_mayor\_legajo(abb\_alumnos1));

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

    buscar\_mayor\_dni(abb\_alumnos1,dni\_max);

    textcolor(green); write('El DNI más grande es '); textcolor(red); writeln(dni\_max);

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

    contar\_legajos\_impar(abb\_alumnos1,legajos\_impar);

    textcolor(green); write('La cantidad de alumnos con legajo impar es '); textcolor(red); writeln(legajos\_impar);

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

    buscar\_legajo\_mayor\_promedio(abb\_alumnos1,promedio\_max,legajo\_max);

    textcolor(green); write('El legajo y el promedio del alumno con mayor promedio son '); textcolor(red); write(legajo\_max); textcolor(green); write(' y '); textcolor(red); write(promedio\_max:0:2); textcolor(green); writeln(', respectivamente');

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

    promedio:=1+random(91)/10;

    cargar\_abb\_alumnos3(abb\_alumnos3,abb\_alumnos1,promedio);

    if (abb\_alumnos3<>nil) then

      imprimir\_abb\_alumnos3(abb\_alumnos3);

  end;

end.