

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;
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;
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;
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;
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;
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;
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;
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;
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;
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;
  end;
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;
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;
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;
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;
    end;
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;
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;
        end;
    end;
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;
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;
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;
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;
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;
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;
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);
        end;
    end;
end;

```



```
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;
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;
    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;
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;
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;
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;
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;
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;
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;
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.
```