

**Introducción a la Programación II**  
**Examen Recuperatorio 7/11/2009**

**Enunciado:**

Suponga que se tiene un árbol cuyos nodos poseen (**codalumno** (int), **codmateria** (int), **nota** (real)). Está ordenado por **codalumno** y la información corresponde a las notas de un turno de examen. Debe solicitar el ingreso por teclado de 2 códigos de alumnos (**codalumnomin** y **codalumnomax**); recorrer el árbol una sola vez pasando por todos los exámenes correspondientes a alumnos cuyos códigos están entre los dos solicitados. Debe además armar una lista simple donde cada nodo corresponde a una materia y posee además la cantidad de alumnos que rindieron y la nota más alta de dicha materia. La lista debe estar siempre ordenada de menor a mayor por cantidad de alumnos que rindieron la materia. Debe realizar el diagrama de estructura y modularizar convenientemente, no puede utilizar estructuras auxiliares.

## Resolución:

**Program** Recuperatorio09;

### **Type**

```
ptroArbol = ^Tarbol;  
Tarbol = record  
    codalumno:integer;  
    codmateria:integer;  
    nota:real;  
    izq:ptroArbol;  
    der:ptroArbol;  
end;
```

```
ptroLista=^Tlista;  
Tlista = record  
    codmateria:integer;  
    mayornota:real;  
    cantalumnos:integer;  
    sig:ptroLista;  
end;
```

**Procedure** insertarNodo(**var** arbol:ptroArbol; nuevoNodo:ptroArbol);  
*{Inserta el nodo que recibe en forma ordenada en el árbol recursivamente}*

**begin**

```
if (arbol = NIL) then  
    arbol := nuevoNodo  
else  
    if (arbol^.codalumno < nuevoNodo^.codalumno) then  
        insertarNodo(arbol^.der, nuevoNodo)  
else  
    if (arbol^.codalumno > nuevoNodo^.codalumno) then  
        insertarNodo(arbol^.izq, nuevoNodo);
```

**end**;

**Procedure** crearNodo(**var** nuevoNodo:ptroArbol; codalumno, codmateria:integer; nota:real);  
*{Dado el codigo de alumno, el codigo de materia y la nota crea el nuevo nodo}*

**begin**

```
new(nuevoNodo);  
nuevoNodo^.codalumno:=codalumno;  
nuevoNodo^.codmateria:=codmateria;  
nuevoNodo^.nota:=nota;  
nuevoNodo^.der:=NIL;  
nuevoNodo^.izq:=NIL;
```

**end**;

```
Procedure cargarArbol (var arbol:ptroArbol);  
{Carga el arbol con alguno valores para poder probar el programa}  
var nodoAInsertar:ptroArbol;  
begin  
    nodoAInsertar:=NIL;  
    CrearNodo (nodoAInsertar,6,3,9.5);  
    insertarNodo (arbol,nodoAInsertar);  
    nodoAInsertar:=NIL;  
    CrearNodo (nodoAInsertar,3,2,5);  
    insertarNodo (arbol,nodoAInsertar);  
    nodoAInsertar:=NIL;  
    CrearNodo (nodoAInsertar,11,1,7.8);  
    insertarNodo (arbol,nodoAInsertar);  
    nodoAInsertar:=NIL;  
    CrearNodo (nodoAInsertar,9,3,6.5);  
    insertarNodo (arbol,nodoAInsertar);  
    nodoAInsertar:=NIL;  
    CrearNodo (nodoAInsertar,12,2,4);  
    insertarNodo (arbol,nodoAInsertar);  
    nodoAInsertar:=NIL;  
    CrearNodo (nodoAInsertar,10,1,2);  
    insertarNodo (arbol,nodoAInsertar);  
    nodoAInsertar:=NIL;  
    CrearNodo (nodoAInsertar,7,3,2);  
    insertarNodo (arbol,nodoAInsertar);  
    nodoAInsertar:=NIL;  
    CrearNodo (nodoAInsertar,5,2,6.5);  
    insertarNodo (arbol,nodoAInsertar);  
    nodoAInsertar:=NIL;  
    CrearNodo (nodoAInsertar,4,1,3);  
    insertarNodo (arbol,nodoAInsertar);  
    nodoAInsertar:=NIL;  
    CrearNodo (nodoAInsertar,2,3,4.5);  
    insertarNodo (arbol,nodoAInsertar);  
    nodoAInsertar:=NIL;  
    CrearNodo (nodoAInsertar,1,1,7);  
    insertarNodo (arbol,nodoAInsertar);  
    nodoAInsertar:=NIL;  
    CrearNodo (nodoAInsertar,8,2,10);  
    insertarNodo (arbol,nodoAInsertar);  
{con estos datos, la cantidad de materias es 3,  
codmateria 1, cantidad de alumnos 4, mejor nota 7.8  
codmateria 2, cantidad de alumnos 4, mejor nota 10  
codmateria 3, cantidad de alumnos 4, mejor nota 9.5}  
end;
```

```
Procedure devolverNodo(var lista:ptroLista; codmateria:integer; var
ptroRecorredor:ptroLista);
{Buscar el codmateria en la lista y devuelve el puntero al nodo quitandolo
de la lista para que sea modificado e insertado nuevamente}
var ptroAnterior:ptroLista;
begin
  ptroRecorredor:=lista;
  if (lista <> NIL) then
    begin
      if (lista^.codmateria = codmateria) then{el codmateria esta en la primer a posicion}
        begin
          lista:=lista^.sig;
          ptroRecorredor^.sig:=NIL;
        end
      else
        begin
          ptroAnterior:=ptroRecorredor;{ptroAnterior siempre va una posicion atras de
ptroRecorredor}
          ptroRecorredor:=ptroRecorredor^.sig;
          while ((ptroRecorredor <> NIL) and (ptroRecorredor^.codmateria <> codmateria)) do
            begin
              ptroAnterior:=ptroRecorredor;
              ptroRecorredor:=ptroRecorredor^.sig;
            end;
          if(ptroRecorredor <> NIL) then{osea que ptroRecorredor está apuntando a la materia}
            ptroAnterior^.sig:=ptroRecorredor^.sig;{desvinculo el nodo correspondiente a
codmateria de la lista}
          end;
        end;
      end;
    end;
end;
```

```
Procedure insertarOrdenado(var lista:ptroLista; var nuevoNodo:ptroLista);
{Dada la lista y el nuevo nodo, lo inserta en forma ordenada recursivamente}
begin
  if(lista = NIL) then
    lista:=nuevoNodo
  else
    if(lista^.cantalumnos > nuevoNodo^.cantalumnos) then
      begin
        nuevoNodo^.sig:=lista;
        lista:=nuevoNodo;
      end
    else
      insertarOrdenado(lista^.sig,nuevoNodo);
    end;
```

```
Procedure crearNodoLista (var nuevoNodo:ptroLista; codmateria:integer; nota:real);  
{Dado el codigo de materia y la nota crea el nuevo nodo}  
begin  
  new(nuevoNodo);  
  nuevoNodo^.codmateria:=codmateria;  
  nuevoNodo^.mayornota:=nota;  
  nuevoNodo^.cantalumns:=1;  
  nuevoNodo^.sig:=NIL;  
end;
```

```
Procedure modificarNodo (var nodoAInsertar:ptroLista; nota:real);  
{Dado el nodo, y la nueva nota, modifico el nodo, sumando 1 al alumno y  
dejo la mayor nota entre la que ya tenia y la nueva}  
begin  
  if(nodoAInsertar^.mayornota < nota) then  
    nodoAInsertar^.mayornota:=nota;  
    nodoAInsertar^.cantalumns:=nodoAInsertar^.cantalumns + 1;  
    nodoAInsertar^.sig:=NIL;  
end;
```

```
Procedure modificarLista (var lista:ptroLista; codmateria:integer; nota:real);  
{Dado un codigo de materia y una nota, modifico la lista segun corresponda,  
agrego la nueva materia ordenada por nota, si es que no existe, o obtengo el nodo  
de la materia, lo modifico y lo vuelvo a insertar ordenado}  
var nodoAInsertar:ptroLista;  
begin  
  nodoAInsertar:=NIL;  
  devolverNodo (lista, codmateria, nodoAInsertar);  
  if (nodoAInsertar = NIL) then  
    CrearNodoLista (nodoAInsertar, codmateria, nota)  
  else  
    modificarNodo (nodoAInsertar, nota);  
    insertarOrdenado (lista, nodoAInsertar);  
end;
```

**Procedure** buscarPosicion(arbol:ptroArbol; var lista:ptroLista; codmin,codmax:integer);  
*{Recorre recursivamente el arbol burscando a los codigos de alumnos que se encuentran entre codmin y codmax, y agrega los datos a la lista. (Problema a resolver en el examen)}*

```
begin
  if (arbol <> NIL) then
    begin
      if (arbol^.codalumno >= codmin) and (arbol^.codalumno <= codmax) then
        begin
          modificarLista(lista,arbol^.codmateria,arbol^.nota);
          buscarPosicion(arbol^.izq,lista,codmin,codmax);
          buscarPosicion(arbol^.der,lista,codmin,codmax);
        end
      else
        if (arbol^.codalumno < codmin) then
          buscarPosicion(arbol^.der,lista,codmin,codmax)
        else
          if (arbol^.codalumno > codmax) then
            buscarPosicion(arbol^.izq,lista,codmin,codmax);
          end;
        end;
      end;
    end;
```

**procedure** mostrarLista(lista:ptroLista);  
*{Muestra la lista de materias recursivamente}*

```
begin
  if(lista <> NIL) then
    begin
      writeln('');
      writeln('Materia: ',lista^.codmateria);
      writeln('Mayor Nota: ',lista^.mayornota);
      writeln('Cantidad de Alumnos: ',lista^.cantalumnos);
      writeln('-----');
      mostrarLista(lista^.sig);
    end;
  end;
```

*{\*\*\*\*\* Programa Principal \*\*\*\*\*}*

```
VAR lista:ptroLista;
    arbol:ptroArbol;
    codmin,codmax:integer;
```

```
BEGIN
  arbol:=NIL;
  lista:=NIL;
  cargarArbol(arbol);
  codmin:=1;
  codmax:=12;
  buscarPosicion(arbol,lista,codmin,codmax);
  mostrarLista(lista);
END.
```