Procedimientos de Listas SIMPLES:

```
procedure DesvincularListaSimple(var lista:PuntLista);
var
   aux:PuntLista;
begin
   aux:=lista;
   lista:=aux^.sig;
   aux^.sig:=nil;
end;
Procedure AgregarAlFinal(var lista: puntero; cursor, nodo: puntero);
begin
   if(lista = nil) then begin
      lista := nodo;
      nodo^.sig := lista;
   end else if(cursor^.sig = lista) then begin
      // Llegar al inicio de nuevo
      cursor^.sig := nodo;
      nodo^.sig := lista;
   end else
      AgregarAlFinal(lista, cursor^.sig, nodo);
end;
function BuscaAnterior(lista: p_nodo; valor: integer): p_nodo;
begin
   if (lista <> nil) and (lista^.valor = valor) then
         BuscaAnterior := lista
   else if (lista^.next <> nil) then begin
      if (lista^.next^.valor <> valor) then
         BuscaAnterior := BuscaAnterior(lista^.next, valor)
      else
         BuscaAnterior := lista;
   end else
      BuscaAnterior := nil;
end;
```

```
procedure insertarOrdenado(var clientesNivel: PuntClientes; clientes: PuntClientes);
begin
   if (clientesNivel = nil) then begin
      clientesNivel := clientes
   else if clientes^.monto_pares <= clientesNivel^.monto_pares then begin  //cuando lista es mayor es ascendente... y sino, descendente.
      clientes^.Punt_num := clientesNivel;
      clientesNivel := clientes;
   end else
      insertarOrdenado(clientesNivel^.Punt_num, clientes);
   end;
end;

Procedure imprimirlista(lista:puntlista);
Begin
   If (lista <> Nil) Then
      Begin
         writeln(lista^.codigo);
         imprimirlista(lista^.sig);
      End;
End;
```

Procedimientos de List Doblemente Vinculada

```
Procedure InsertarOrdDoblementeVinculada(var Lista:Plista; Nodo:Plista);
begin
   if (Lista = Nil) then
      Lista := Nodo
   else
      if (Nodo^.valor <= Lista^.valor) then
      begin
         Nodo^.sig := lista;
         Nodo^.ant := Lista^.ant;
         Lista^.ant := nodo;
         Lista := Nodo;
      end
      else
      InsertarOrdDoblementeVinculada(Lista^.sig,Nodo);
end;

procedure DesvincularLista (var Lista: PuntLista; desvinculador: PuntLista);
Begin
   If (desvinculador^.sig = nil) and (desvinculador^.ant = nil) then
   begin
      Lista := Nil
      desvinculador^.sig := nil;
      desvinculador^.ant := nil;
   end

   else
      if (desvinculador^.sig <> nil) and (desvinculador^.ant <> nil) then
      begin
         desvinculador^.ant^sig := desvinculador^.sig
         desvinculador^.sig^.ant := desvinculador^.ant
      end
```

```
        else if (desvinculador^.sig = nil) then
            Lista := Desvinculador^.sig

        else  (desvinculador^.ant <> nil)
            desvinculador^.ant^sig := desvinculador^.sig;
    desvinculador^.sig := Nil;
    desvinculador^.ant := Nil;
end;
```

Procedimientos de Arbol:

```
function menorEnArbol(arbol: puntero): integer;
begin
   if(arbol <> nil) then begin
      if(arbol^.menor <> nil) then
         menorEnArbol := menorEnArbol(arbol^.menor)
      else
         menorEnArbol := arbol^.valor;
   end else
      menorEnArbol := 0;
end;
Procedure InsertarEnArbol(var Arbol:TipoArbol; Nodo:TipoArbol);
begin
   if (Arbol = Nil) then
      Arbol := Nodo
   else
   begin
      if (Nodo^.valor < Arbol^.valor) then
         InsertarEnArbol(Arbol^.menor,Nodo)
      else
         InsertarEnArbol(Arbol^.mayor,Nodo);
   end;
end;
```

```
Procedure CargarArbol(var Arbol:TipoArbol);
var
   i:integer;
   Nodo:TipoArbol;
begin
   i:= 0;            //PARA ENTRAR EN EL BUCLE
   while (i <> -1) do
   begin
      CrearNodo(Nodo);
      InsertarEnArbol(Arbol,Nodo);
      writeln('ingrese un valor distinto a -1 para seguir agregando nodos');
      readln(i);
   end;
end;

function longitud(nodo: TipoArbol): integer;
var longMayores, longMenores, raiz: integer;
begin
   if(nodo <> nil) then begin
      longMayores := longitud(nodo^.mayor);
      longMenores := longitud(nodo^.menor);
      raiz:= 1;

      longitud:= longMayores + longMenores + raiz;
   end else
      longitud := 0;
end;
Function BuscaEnArbol (Arbol: PuntArbol; codigo_buscado:integer):   boolean;
Begin
   If (Arbol <> Nil) Then
```

```
      Begin
        If (codigo_buscado = arbol^.codigo) Then
          BuscaEnArbol := true
        Else
          If (codigo_buscado < arbol^.codigo) Then
            BuscaEnArbol := BuscaEnArbol(Arbol^.menor,codigo_buscado)
        Else
          BuscaEnArbol := BuscaEnArbol(Arbol^.mayor,codigo_buscado);
      End
   Else
      BuscaEnArbol := false;
End;

procedure ImprimirArbolAscendente(arbol:PuntArbol);
begin
   if (arbol <> nil) then
   begin
      ImprimirArbolAscendente(arbol^.menor);
      writeln('nro_facturas: ',arbol^.nro_factura);
      writeln('facturas_impagas: ',arbol^.facturas_impagas);
      writeln('------------------------------');
      ImprimirArbolAscendente(arbol^.mayor);
   end;
end;

procedure imprimirDescendente (arbol:PuntArbol);
begin
   if (arbol <> nil) then
   begin
      writeln (arbol^.valor);
      imprimirAscendente(arbol^.mayores);
```

```
      imprimirAscendente(arbol^.menores);
   end;
end;

procedure imprimirPostOrder (arbol:PuntArbol);
//no se imprime un padre si no se han impreso todos sus hijos
begin
   if (arbol <> nil) then
   begin
      imprimirAscendente(arbol^.mayores);
      imprimirAscendente(arbol^.menores);
      writeln (arbol^.valor); //padre
   end;
end;

procedure imprimirPreOrder (arbol:PuntArbol);
//no se imprime un nodo si no se ha impreso su padre
begin
   if (arbol <> nil) then
   begin
      writeln (arbol^.valor);
      imprimirAscendente(arbol^.menores);
      imprimirAscendente(arbol^.mayores);
   end;
end;

function SumaArbol (nodo: PuntArbol): integer; //suma todos los valores de cada nodo
begin
   if (nodo <> nil) then
      SumaArbol:= nodo^.'VALOR QUE TENGA EL TYPE' + SumaArbol (nodo^.menor) +
SumaArbol (nodo^.mayor) //sumas mientras sea dist a nil
```

```
   else
      SumaArbol:= 0; //si es nil, suma 0
end;

Procedure CrearNodo(var Nodo:TipoArbol);
begin
   New(Nodo);
   writeln('inserte valor al nodo:');
   readln(Nodo^.valor);
   Nodo^.mayor := nil;
   Nodo^.menor := nil;
end;
```

Procedimientos Lista Circular:

```
Procedure CrearListaCircular(var lista: puntero);
var input: integer;
   nuevoNodo: puntero;
begin
   writeln('Meteme un valorcito');
   readln(input);

   if(input <> -1) then begin
      CrearNodo(nuevoNodo, input);
      AgregarAlFinal(lista, lista, nuevoNodo);

      CrearListaCircular(lista);
   end else
      writeln('Fin de la carga');
end;
Procedure ImprimirListaEnOrden(lista, cursor: puntero);
begin
```

```
  if(lista <> nil) then begin
     writeln(cursor^.valor);
     if cursor^.sig <> lista then
        ImprimirListaEnOrden(lista, cursor^.sig);
  end else
     writeln('La lista circular está vacía...');
end;
```

Procedimientos Archivos:

```
Function BuscaEnArchivo (Var Codigo: Tarchivo; Codigo_Buscado: integer):   boolean;
Var
   Dato:   integer;
Begin
   seek(Codigo,0);
   // EL ARCHIVO YA ESTA ABIERTO, CORRESPONDE SEEK(CODIGO,0)
   read (Codigo, Dato);

   While Not (eof(Codigo)) And (Codigo_Buscado > Dato)   Do
   begin
      read (Codigo, Dato);
      writeln(dato);
   end;
   If (Dato = Codigo_Buscado) And Not (eof(Codigo)) Then
      BuscaEnArchivo := true
   Else
      BuscaEnArchivo := false;
End;
```

```
assign (codigo,'/work/grupo_anonimo.dat');
```