Módulos de Imperativo.

Vectores.

Cargar Vector:

```
procedure cargarVector( var num:vector; var diml:integer );
var
 dato:integer;
begin
  dimL := 0;
  read (dato);
  while (dato <> 0) and ( dimL < dimF ) do begin
       dimL := dimL + 1;
     num [dimL] := dato;
     read (dato)
  end;
End;
```

Crear un vector a partir de una lista ordenada:

```
procedure InsertarElemento (var pri: lista; per: producto);
var ant, nue, act: lista;
begin
 new (nue);
 nue^.dato := per;
 act := pri;
 ant := pri;
 while (act<>NIL) and (act^.dato.CodP < per.CodP) do begin
    ant := act;
    act := act^.sig ;
 end;
 if (ant = act) then
   pri := nue
 else
   ant^.sig  := nue;
 nue^.sig := act ;
end;
```

```
procedure CrearVector(var v:vector);
var
  p: producto;
  i: integer;
begin
  for i:= 1 to dimf do begin
    Leer(p);
    while(p.CodP<>-1)do begin
      InsertarElemento( v[i], p);
      Leer(p);
    end;
  end;
end;
```

Búsqueda Dicotómica recursiva:

```
procedure BusquedaD(v:vector; ini,fin,dato:integer; var pos:integer );
var
  medio:integer;
begin
  if (fin<ini) then
    writeln ('No hay elementos.');
  else begin
    medio:= (ini+fin) div 2;
    if(v[medio]=dato) then
      pos:=medio
    else begin
      if(v[medio]>dato) then begin
        fin:= medio-1;
        BusquedaD(v,ini,fin,dato,pos);
      end
      else begin
        ini:= medio+1;
        BusquedaD(v,ini,fin,dato,pos);
      end;
```

```
  end;
end;
```

Máximo Recursivo:

```
procedure maximo(v:vector;var max:integer;diml:integer; i:integer);
begin
     if(i<=diml) then begin
          if(v[i]>max) then
            max:= v[i];
          i:= i+1;
          maximo (v,max,diml,i);
   end;
end;
```

Ordenar Vector:

```
procedure ordenar (var v:vector; diml:integer);
var
   j, i, aux: integer;
begin
   for i:=2 to diml do begin
      aux:= v[i];
      j:= i-1;
      while(j>0) and (v[j]>aux) do begin
         v[j+1]:= v[j];
         j:= j-1;
      end;
      v[j+1]:= aux;
   end;
end;
```

Ordenar Vector con Condición:

```
procedure ordenar (var v: vector; diml: integer);
var
   j,i: integer;
   x: alumno;
begin
   for i:=2 to diml do begin
      x:=v[i];
      j:=i-1;
      while(j>0) and (v[j].legajo>x.legajo) do begin
         v[j+1]:=v[j];
         j:=j-1;
      end;
      v[j+1]:=x;
   end;
end;
```

Suma de sus elementos Recursivo:

```
procedure sumar (v:vector; diml:integer;i:integer; var suma:integer);
begin
  if (i<=diml) then begin
      suma:=suma+v[i];
      i:=i+1;
      sumar(v,diml,i,suma);
  end;
end;
```

Listas.

Agregar adelante:

```
Procedure AgregarAdelante (var L:lista; per:integer);
Var nue:Lista;
Begin
    New(nue);
    nue^.datos:=per;
    nue^.sig:=L;
    L:=nue;
End;
```

Agregar al final:

```
procedure AgregarAlFinal (var pri, ult: lista; per: integer);
var  nue : lista;
begin
   new (nue);
   nue^.datos:= per;
   nue^.sig := NIL;
   if pri <> Nil then
          ult^.sig := nue
   else
          pri := nue;
   ult := nue;
end;
```

Insertar Ordenado:

```
procedure InsertarElemento ( var pri: lista; per: integer);
var ant, nue, act: lista;
begin
     new (nue);
     nue^.datos := per;
     act := pri;
     ant := pri;
     while (act<>NIL) and (act^.datos < per) do begin
         ant := act;
         act := act^.sig ;
    end;
    if (ant = act)  then pri := nue
              else  ant^.sig  := nue;
    nue^.sig := act ;
end;
```

Buscar elemento Recursivo:

```
procedure Buscar (l:lista; var x:boolean; dato:integer);
begin
  if(l<>nil) and (x<>true) then begin
    if (l^.datos=dato) then
      x:=true;
    Buscar (l^.sig,x,dato);
  end;
end;
```

Maximo elemento recursivo:

```
procedure Maximo(l:lista; var max:integer);
begin
  if(l<>nil)then begin
     if(l^.datos>max)then
          max:=l^.datos;
      maximo(l^.sig,max);
 end;
end;
```

Minimo elemento recursivo:

```
procedure Minimo(l:lista; var min:integer);
begin
  if(l<>nil)then begin
      if(l^.datos<min)then
           min:=l^.datos;
      minimo(l^.sig,min);
 end;
end;
```

Recorrida ida y vuelta:

```
procedure recorrido ( pri : lista);
Begin
   while(pri <> NIL) do begin
       writeln(pri^.datos);
    pri:=pri^.sig;
    end;
end;

procedure recorridoIda ( pri : lista);
Begin
    if (pri <> NIL) then begin
        writeln(pri^.datos);
    recorridoIda(pri^.sig);
```

```
    end;
end;

procedure recorridoVuelta ( pri : lista);
begin
   if (pri <> NIL) then begin
        recorridoVuelta(pri^.sig);
    writeln(pri^.datos);
 end;
end;
```

Merge:

```
procedure AgregarAtras (var L:lista; N:lista);
var nue,act:lista;
begin
    act:=L;
    new n
    nue:=N;
    nue^.sig:=nil;
    if(L<>nil) then begin
        act:=L;
        while(act^.sig<>nil)do
                act:=act^.sig;
        act^.sig:=nue;
    end
    else
       L:=nue;
end;

procedure Borrar (var l:lista);
begin
   l:= l^.sig;
end;
```

```
procedure buscarMin (var v:vector;var ISBN:lista);
var
   min,i,pos:integer;
begin
   min:=9999;
   for i:=1 to Dimf do begin
      if (v[i]<>NIl)then
         if(v[i]^.datos<min)then begin
            min:=v[i]^.datos;
            pos:=i;
            ISBN:=v[i];
         end;
   end;
   if(min<>9999)then
      Borrar(v[pos])
   else
      isbn:=nil;
end;

procedure Merge(var v:vector; var l:lista);
var
   ISBN:lista;
begin
   buscarmin(v,ISBN);
   while(ISBN<>nil) do begin
      agregaratras(l,Isbn);
      buscarmin(v,Isbn);
   end;
end;
```

Merge Acumulador:

```
procedure AgregarAtras (var L:lista; N:lista);
var nue,act:lista;
begin
    act:=L;
    nue:=N;
    nue^.sig:=nil;
    if(L<>nil) then begin
        act:=L;
        while(act^.sig<>nil)do
                act:=act^.sig;
        act^.sig:=nue;
    end
    else
       L:=nue;
end;

procedure Borrar(var l:lista);
begin
   l:=l^.sig;
end;

procedure buscarMin(var v:vector;var E:lista);
var
   min:CADENA;
   i,pos:integer;
begin
   min:= 'zzzzzzzz';
   for i:= 1 to Dimf do begin
      if (v[i]<>NIl) then
         if(v[i]^.dato.impu<min)then begin
               min:=v[i]^.dato.impu;
               pos:=i;
```

```
            E:=v[i];
        end;
    end;
    if(min<>'zzzzzzzz')then
        Borrar(v[pos])
    else
        E:=nil;
end;

procedure MergeAcumulador(var v:vector; var l:lista);
var
    E,aux:lista;
    GastoTot: integer;
begin
    buscarmin(v,E);
    while(E<>nil) do begin
        aux:=e;
        GastoTot:=0;
        while(E<>NIl) and(E^.dato.Impu=aux^.dato.impu) do begin
            gastotot:=gastotot+E^.dato.gastop;
            buscarmin(v,E);
        end;
        aux^.dato.gastop:=gastotot;
        aux^.sig:=nil;
        agregaratras(l,aux);
    end;
end;
```

Arboles.

Crear Arbol:

```
Program arboles;
Type
  arbol= ^nodoA;
  nodoA = Record
                dato: integer;
                HI: arbol;
                HD: arbol;
   end;
 listaNivel = ^nodoN;
 nodoN = record
   info: arbol;
   sig: listaNivel;
 end;
```

Insertar ABB:

```
procedure InsertarABB(var a:arbol; dato:integer);
var
    aux:arbol;
begin
   if(a=nil)then begin
      new(aux);
      aux^.dato:=dato;
      aux^.HI:=nil;
      aux^.HD:=nil;
      a:=aux;
  end
  else
      if(a^.dato>dato)then
            InsertarABB(a^.HI, dato)
      else
            InsertarABB(a^.HD, dato);
```

```
end;
```

Recorrido Acotado Contador:

```
procedure RecorridoAcotadoPre(a:arbol; var contador:integer);
begin
   if(a<>nil)then begin
      if(a^.dato.monto_total>100)then
          contador:=contador+1;
      RecorridoAcotadoPre(a^.HI,contador);
      RecorridoAcotadoPre(a^.HD,contador);
   end;
end;
```

Recorrido acotado:

```
procedure RecorridoAcotado(a:arbol; inf,sup:integer);
begin
   if(a<>nil)then begin
     if(a^.dato>=inf)then
        if( a^.dato<=sup)then begin
             writeln('Numero: ', a^.dato);
             RecorridoAcotado(a^.HI,inf,sup);
             RecorridoAcotado(a^.HD,inf,sup);
       end
       else
            RecorridoAcotado(a^.HI,inf,sup)
   else
        RecorridoAcotado(a^.HD,inf,sup);
 end;
end;
```

Formas de Imprimir:

```
procedure preOrden( a: arbol );
begin
 if (a <> nil ) then begin
```

```
    writeln (a^.dato, ' ');
    preOrden (a^.HI);
    preOrden (a^.HD)
 end;
end;
```

```
procedure PostOrden( a: arbol );
begin
 if (a <> nil ) then begin
   PostOrden (a^.HI);
   PostOrden (a^.HD);
   writeln (a^.dato, ' ');
 end;
end;
```

```
procedure EnOrden(a:arbol);
begin
 if(a<>nil) then begin
   EnOrden(a^.HI);
   writeln(a^.dato,' ');
   EnOrden(a^.HD);
 end;
end;
```

Arbol ordenado con lista sin repetición:

```
procedure Buscar(l:lista; var x:boolean; dato:integer);
begin
  if(l<>nil)and(x<>true)then begin
     if(l^.dato=dato)then
       x:=true;
    Buscar(l^.sig,x,dato);
  end;
end;
```

```
procedure AgregarAdelante (var L:lista; per:integer);
Var nue:Lista;
Begin
 New(nue);
 nue^.dato:=per;
 nue^.sig:=L;
 L:=nue;
End;

procedure InsertarABB(var a:arbol; dato:pizzas; CodP:integer);
var
 aux:arbol;
 ok:boolean;
begin
 if(a=nil)then begin
    new(aux);
    aux^.dato.CodC:=dato.CodC;
    aux^.dato.Total:= dato.Total;
    aux^.HI:=nil;
    aux^.HD:=nil;
    aux^.CodP:=nil;
    ok:=false;
    Buscar(aux^.CodP,ok,CodP);
    if(ok=false)then
        AgregarAdelante(aux^.CodP,Codp);
    a:=aux;
 end
 else
   if(a^.dato.CodC>dato.CodC)then
       InsertarABB(a^.HI, dato, CodP)
   else
      if(a^.dato.CodC<dato.CodC)then
           InsertarABB(a^.HD, dato, CodP)
      else begin
           a^.dato.total:=a^.dato.total + dato.total;
```

```
            Buscar(a^.CodP,ok,CodP);
            if(ok=false)then
              AgregarAdelante(a^.CodP,Codp);
        end;
end;
```

Maximo y minimo:

```
procedure Max(a:arbol; var x:integer);
begin
  if(a=nil)then
       x:=-1
  else
       if(a^.HD=nil)then
            x:=a^.dato
     else
          Max(a^.HD,x);
end;

procedure Min(a:arbol; var x:integer);
begin
    if(a=nil)then
        x:=-1
    else
        if(a^.HI=nil)then
            x:=a^.dato
        else
            Min(a^.HI,x);
end;
```

Buscar elemento:

```
procedure Buscar (a:arbol; var p:arbol;dato: integer);
begin
    if(a=nil) then
            p:=nil
```

```
    else
        if(dato=a^.dato)then
            p:=a
        else
            if(dato<a^.dato)then
                Buscar(a^.HI,p,dato)
            else
                Buscar(a^.HD,p,dato);
end;
```

Borrar elemento:

```
procedure Min(a:arbol; var x:integer);
begin
 if(a=nil)then
   x:=-1
 else
   if(a^.HI=nil)then
     x:=a^.dato
   else
      Min(a^.HI,x);
end;

procedure BorrarElemento(var a:arbol; dato:integer; var ok:boolean);
var
 x:integer;
 aux:arbol;
begin
   if(a=nil) then
      writeln('el dato no se encontro')
   else
      if(a^.dato>dato) then
           BorrarElemento(a^.HI, dato, ok)
      else
         if(a^.dato<dato) then
```

```
            BorrarElemento(a^.HD, dato, ok)
        else begin
            ok:=true;
            if(a^.HD=nil) and (a^.HI=nil)then begin
                aux:=a;
                dispose(aux);
                a:=nil;
            end
            else
                if(a^.HD= nil) and (a^.HI <>nil) then begin
                        aux:=a;
                        a:= a^.HI;
                        dispose(aux);
                    end
                else
                 if(a^.HI= nil) and (a^.HD <>nil) then begin
                    aux:=a;
                    a:= a^.HD;
                    dispose(aux);
                    end
                  else
                    if(a^.HD<>nil) and (a^.HI<>nil)then begin
                        Min(a^.HD,x);
                        a^.dato:=x;
                        BorrarElemento(a^.HD,x,ok);
                    end;
        end;
end;
```