MÓDULOS

LISTAS

AGREGAR ADELANTE

```
Procedure agregarAdelante (Var L: Lista; n:integer);
var
   nue:lista;
Begin
  new (nuevo                    // creo nodo
  nue^.dato:=n;                 // le asigno el dato
  nue^.sig:= L;                 // nuevo nodo apunta a L
  L:=nue;                       // L apunta al nuevo nodo
End;
```

AGREGAR ATRÁS

```
Procedure agregarFinal (Var L, ult: Lista; n: integer);
var
   nue:lista;
Begin
  new (nue);                    // creo nodo
  nue^.dato:=n;                 // le asigno el dato
  nue^.sig:= nil;               // nuevo nodo no apunta a nada
  if (L= nil) then
       L:= nue;
  else
       ult^.sig:= nue;
  ult:= nue;
End;
```

INSERTAR ORDENADO

```
Procedure InsertarNodo ( var L: lista; num: integer);
 var
      ant, nue, act: lista;
 begin
    new (nue);                                  (1)
    nue^.dato := num;                           (2)
    nue^.sig := nil;                            (3)
    if (L= nil) then                            (4)
       L:= nue
    else begin                                  (5)
       act:= L;
       ant := L;
       while (act<>NIL) and (act^.dato < nue^.dato) do begin   (6)
            ant := act;
            act:= act^.sig;
       end;
       if (ant = act) then begin                (7)
            nue^.sig:=L;
            L := nue;
       end
       else begin                               (8)
           ant^.sig:=nue;
           nue^.sig := act ;
       end
end;
```

(1) Creo nuevo nodo.
(2) Inserto el dato en el nodo.
(3) El punter del nuevo nodo es nil (vacío).
(4) Si L está vacío, L va a apuntar el nuevo nodo.
(5) Si L no está vacío, act y ant van a apuntar a L (al primer nodo).
(6) Mientras la lista no termine y el dato del nodo sea menor al nuevo dato, recorro la lista.
(7) Si act = nil, el nuevo nodo se inserta al principio
(8) Si no, el nuevo nodo se ubica en el medio o al final

BORRAR ELEMENTO

```
Procedure eliminarValor (var L: Lista; n:integer);
Var
     act, ant: lista;
     ok: boolean;
Begin
 act:= L;                                       (1)
 ant:= L;
 ok:= false;
 while (act <> nil) and (not ok) do             (2)
    if (act^.dato = n) then                     (3)
          ok:= true
    else begin
          ant:=act;
          act:= act^.sig;
    end;
if (ok=true) then begin                         (4)
   if (act = L) then
      L:= act^.sig
   else begin
      ant^.sig:= act^.sig
   dispose (act);                               (5)
end;
End;
```

(1) act y ant apuntan al primer nodo.
(2) Mientras la lista no termine y no haya encontrado el dato a eliminar, proceso el nodo.
(3) Si encuentro el dato, entonces devuelvo true, sino, sigo recorriendo la lista.
(4) Sale del while cuando termina la lista o encuentra al elemento. Si lo encontró (ok=true), y si es el primer elemento, L va a apuntar al nodo que sigue. Si el nodo no es el primero, el ant apunta al nodo que le sigue a act, porque act lo voy a eliminar.
(5) Elimino act.

VECTORES

AGREGAR

```
Procedure agregar (var vec:numeros; var dimL:integer num:integer;
                   var  ok:boolean);
begin
   ok:= false;
   If ( (dimL + 1) <= tam*) then begin          (1)
      vec[dimL+1]:= num;                        (2)
      dimL:= dimL + 1;                          (3)
      ok:= true;                                (4)
   end;
end;
```

(1) Verifico si existe lugar
(2) Agrego el número
(3) Aumento dimensión lógica
(4) Devuelve true si pudo agregar el dato
*tam sería una CONST con la dimensión física

INSERTAR EN UNA POSICIÓN DETERMINADA

```
Procedure insertar  (var vec:numeros; var dimL:integer; num:integer,
                     pos:integer; var ok:boolean);
begin
   ok:=false;
          (1)                          (2)
   if ( (dimL + 1) <= tam) and (pos=>1) and (pos<=dimL)) then
begin
                  for i:= dimL downto pos do   (3)
                        vec[i+1]:= vec[i];
                  vec[pos]:= num;
                  dimL:= dimL + 1;             (4)
                  ok:= true;                   (5)
   end;
end;
```

(1) Verifico si existe lugar
(2) Verifico posición válida
(3) Hago el corrimiento de elementos
(4) Aumento dimensión lógica
(5) Devuelvo true si puso insertar

BORRAR UN ELEMENTO

```
Procedure borrar ( var vec :numeros;  var dimL :integer; var ok :boolean;
                   pos :integer);
begin
  ok:=false;
  if ( (pos => 1) and (pos<=dimL) ) then begin     (1)
     for i:= pos to (dimL-1) do                    (2)
        vec[i]:= vec[i+1];

     dimL:= dimL - 1;                              (3)
     ok:= true;                                    (4)
   end;
end;
```

(1) Verifico posición
(2) Hago el corrimiento de elementos.
(3) Disminuyo dimensión lógica.
(4) Devuelvo true si encontró el elemento y lo eliminó

BUSCAR EN VECTOR SIN ORDEN

```
Function buscar (vec: numeros;  dimL, num: integer): boolean;
Var
      encontre: boolean;
      pos:integer;
begin
  encontre:=false;
  pos:= 1;
  while ( (pos <=dimL) and (not encontre) ) do begin
        if (vec[pos] = num) then
              encontre:=true
        else
              pos:= pos + 1;
  end;
  buscar:= encontre;
end;
```

ORDENAR VECTOR

```
Procedure Ordenar ( var v: tVector; dimL: indice );
Var
      i, j, p: indice;
      item : tipoElem;
begin
    for i:=1 to dimLog-1 do begin
       p := i;
       for j := i+1 to dimLog do
              if v[j] < v[pos] then pos:=j;
       item := v[pos];
       v[pos] := v[i];
       v[i] := item;
    end;
end;
```

VECTOR CONTADOR

```
Procedure InicializarVcontador (var vec: vectorcont);
Var
   I:integer;
Begin
   For i:=1 to dimF do
      vec[i]:=0;
end:
```