### *Cargar una lista (caso 1)*

```
procedure Cargar(var L:lista);
var
 v: valor;
begin
 leerValor(v);
 while (v.condicion <> condicion) do begin
    AgregarAdelante(L,v);
    leerValor(v);
 end;
end;
```

### *Cargar una lista (caso 2)*

```
procedure Cargar(var L:lista);
var
 v: valor;
begin
 repeat
     leerValor(v);
     AgregarAdelante(L,v);
 until (v.condicion = condicion);
end;
```

### *Agregar Adelante*

```
procedure AgregarAdelante(var L:lista; v:valor);
var
  nue:lista;
begin
 new(nue);
 nue^.dato:= v;
 nue^.sig:= L;
 L:= nue;
end;
```

### *Agregar Atrás*

```
procedure AgregarAtras(var L,ult:lista; v:valor);
var
```

```
  nue: lista;
begin
  new(nue);
  nue^.dato:= v;
  nue^.sig:= nil;
  if (L = nil) then
     L:= nue
 else
     ult^.sig:= nue;
  ult:= nue;
end;
```

### *Insertar Ordenado*

```
procedure InsertarOrdenado(var L: lista; v:valor);
var
 nue,ant,act: lista;
begin
 new(nue);
 nue^.dato:= v;
 act:= 0;
 ant:= 0
 while (act <> nil) and (v.condicion > act^.dato.condicion) do
begin
     ant:= act;
     act:= act^.sig;
 end;
 if (act = ant) then
     L:= nue
 else
     ant^.sig:= nue;
  nue^.sig:= act;
end;
```

### *Borrar un elemento*

```
procedure eliminar (var L:lista; v: valor);
var
```

```
 ant,act: lista;
begin
  act:= L;
  ant:= L;
  while(act <> nil) and (act^.dato.valor <> valor) do begin
     ant:= act;
     act:= act^.sig;
  end;
  if (act <> nil) then begin
     if (ant = act) then
         L:= L^.sig
      else
         ant^.sig:= act^.sig;
     dispose(act);
  end;
end;
```

### *Cargar un vector de registros*

```
procedure CargarVec(var v: vector);
var
  i: integer;
  va: valor;
begin
   for i:= 1 to 5 do begin
      leer(va);
      v[i]:= va;
   end;
end;
```

### *Procesar vector de registros*

```
procedure procesar(vp:vecProductos; var cant: integer);
var
  i: rango;
Begin
  cant:= 0;
  for i:= 1 to 100 do begin
     if (vp[i].precio > 50) then
     cant:= cant +1;
```

```
  end;
end;
```

## *Corte de control*

```
while (L <> nil) do begin
     while (L <> nil) and (nomAct = L^.dato.nom) do begin
end;
```

## *Usar campos de una estructura como parámetros*

### *Para crearlo*

```
Procedure Des_dig(x: integer...);
```

### *Para invocarlo*

```
Des_dig(l^.dato.CAMPONECESARIO...)
```

## *Acceder a un campo de un registro almacenado en un vector.*

```
(...) v[i].datodelregistro (...)
```

## *Usar el campo de una lista como índice.*

### *Para acceder*

```
v[lista^.dato.camponecesario]:= (...)
```

Tener en cuenta que “lista^.dato.camponecesario” es el índice del vector y “v[lista^.dato.camponecesario]” contiene la info dentro del vector en la posición que indica ese campo de la lista.

## *Sumar info que tengo en una lista con la de un vector usando de indice otro campo del registro de la lista.*

```
(...) lista^.dato.campo1 + v[lista^.dato.camponecesario];
```

## *Acceder a la info de un vector que está guardado en un registro dentro de una lista*

```
(...) lista^.dato.registrotipovec[i]
```