BUSQUEDA (DESORDENADA)

```
Function BusquedaDesordenada(L : Lista ; VALOR : Dato) :boolean;
Var
  OK:boolean;
Begin
  OK:=false;
  while (L <> NIL) and (ok= false) do begin
    if (L^.Dato = VALOR) then
      OK:=TRUE;
    else
      L:=L^.Sig;
  end;
  BusquedaDesordenada:=OK;
End;
```

BUSQUEDA (ORDENADA)

```
Function BusquedaOrdenada(L : Lista ; VALOR : Dato) :Boolean;
Var
  OK:boolean;
Begin
  OK:=false;
  while (L <> NIL) AND (L^.Dato ** VALOR) do begin
    L:=L^.Sig;
  end;
  if (L <> NIL) AND (L^.DATO = VALOR) then
    OK:=TRUE;
  BusquedaDesordenada:=OK;
End;
```

** Representa una operación determinada por el inciso

ELIMINAR (SIN REPETICIONES)

```
Procedure EliminarSinRepeticiones(var L : Lista ; VALOR : Dato);
Var
  Act , Ant : Lista;
Begin
  ant:=L;
  Act:=L;
  while (Act <> NIL) AND (Act^.Dato <> VALOR) do begin
    Ant:=Act;
    Act:=Act^.Sig;
  end;
  if (Act <> NIL) then begin
    if(Act = L) then
      L:=L^.Sig
    else
      Ant^.Sig:=Act^.Sig;
    dispose(Act);
  end;
End;
```

ELIMINAR (CON REPETICIONES)

```
Procedure EliminarConRepeticiones(var L : Lista ; VALOR : Dato);
Var
  Act , Ant : Lista;
Begin
  ant:=L; Act:=L;
  while (Act <> NIL) do begin
    if (Act^.Dato <> VALOR) then begin
      Ant:=Act;
      Act:=Act^.Sig;
    end;
    else begin
      if(act=L) then begin
        L:=act^.sig;
        ant:=L;
      end;
      else
        Ant^.Sig:=Act^.Sig;
      dispose(Act);
      Act:=Ant;
    end;
  end;
End;
```

CREAR

```
Program Crear;
Type

  Lista=^Nodo;

  Nodo=Record
    Dato:integer;
    Sig:Lista;
  End;
Var
  L:Lista;
Begin
  L:=NIL;
End.
```

AGREGAR ADELANTE

```
Procedure AgregarAdelante (var L:Lista; D:Dato);
Var
  Nue:Lista;
Begin
  new(Nue);
  Nue^.Dato:=D;
  Nue^.Sig:=L;
  L:=Nue;
End;
```

INSERTAR ORDENADO

```
Procedure InsertarOrdenado (var L : Lista ; D : Dato);
Var
  Act , Ant , Nue : Lista;
Begin
  new(Nue);
  Nue^.Dato:=D;
  act:=l;
  ant:=l;
  while (Act <> NIL) AND (Dato ** act^.Dato) do begin
    Ant:=Act;
    Act:=Act^.Sig;
  end;
  if ((act = ant) then
    L:=Nue
  else
    Ant^.Sig:=Nue;
  Nue^.Sig:=Act;
End;
```

** Representa una operación determinada por el inciso

AGREGAR ATRAS

```
Procedure AgregarAtras (var Pri, Ult : Lista ; D : Dato);
Var
  Nue:Lista;
Begin
  new(Nue);
  Nue^.Dato:=D;
  Nue^.Sig:=NIL;
  if (Pri = NIL) then begin
    Pri:=Nue;
  end
  else
    Ult^.sig:=Nue;
  Ult:=Nue;
End;
```

IMPRIMIR

```
Procedure ImprimirLista (L:Lista);
Begin
  while(L <> NIL) do begin
    writeln(L^.Dato);
    L:=L^Sig;
  end;
End;
```

RECORRER

```
Procedure RecorrerLista (L:Lista);
Begin
  while(L <> NIL) do begin
    L:=L^Sig;
  end;
End;
```