

# Resumen

# Algoritmos

## Algoritmos de carga

A lo sumo: no excede X limite, pero puede ser menor o igual q el. Para ser true

```
Procedure agregarAtras(var L,Ult: lista; n: integer);
var
  nue: lista;
Begin
  new(nue);
  nue^.dato:= n;
  nue^.sig:= nil;
  if(L = nil)then
    L:= nue
  else
    Ult^.sig:= nue;
  Ult:= nue;
End;
```

```
Procedure agregarAdelante(var L: lista; n: integer);
Var
  nue: lista;
Begin
  new(nue);
  nue^.dato:= n;
  nue^.sig:= L;
  L:= nue;
End;
```

```
Procedure insertarOrdenado(var L: lista; n: integer);
Var
  nue: lista;
  ant,act: lista;
Begin
  new(nue);
  nue^.dato:= n;
  ant:= L;
  act:= L;
  While(act <> nil) and (n > act^.dato) do //ascendente
    begin
      ant:= act;
      act:= act^.sig;
    end
  if(act = ant) then //principio o lista vacia
    L:= nue
  else // al medio o al final
    ant^.sig:= nue;
  nue^.sig:= act;
End;
```



# AgregarAtras

```
procedure agregarAtras(var L,Ult: lista; d: cosa);
var
    nue: lista;
begin
    new(nue);
    nue^.dato:= d;
    nue^.sig:= nil;
    if(L = nil)do //si es el primer nodo
        L:= nue;
    else // si no es el primer nodo
        Ult^.sig:= nue;
    Ult:= nue;
end;
```

# AgregarAdelante

```
procedure agregarAdelante(var L: lista; d: cosa);
var
    nue: lista;
begin
    new(nue);
    nue^.dato:= d;
    nue^.sig:= L;
    L:= nue;
end;
```

# InsertarOrdenado

```
procedure insertarOrdenado(var L: lista; d: cliente);
var
  nue: lista;
  ant, act: lista;
begin
  new(nue);
  nue^.dato:= d;
  ant:= L;
  act:= L;
  While(act <> nil)and(d.dni > act^.dato.dni); // > ascendente | < descendente
  begin
    ant:= act;
    act:= act^.sig;
  end;
  if(act = ant)then //al principio o vacio
    L:= nue
  else
    ant^.sig:= nue;
    nue^.sig:= act;
  end;
end;
```

# Eliminar En Listas

Hay 4 variantes del eliminar en listas

## Desordenado

Eliminar la Primera  
Ocurrencia en una Lista  
Desordenada

Se tiene 1 5 2 10 3 1  
Se busca borrar el 2  
Queda 1 5 10 3 1

Eliminar todas las Ocurrencias  
en una Lista Desordenada

Se tiene 1 5 2 10 3 1  
Se busca borrar el 1  
Queda 5 2 10 3

## Ordenado

Eliminar la Primera  
Ocurrencia en una Lista  
Ordenada

Se tiene 1 5 9 11 32  
Se busca borrar el 6  
Recorre solo hasta el 9.  $[6 < 9]$

Eliminar todas las Ocurrencias  
en una Lista Ordenada

Se tiene 1 5 9 11 32 32  
Se busca borrar el 32  
Queda 1 5 9 11. También cheque q el element actual no sea  
mas grande q el buscado

# Eliminar En Listas

Desordenado

Primera Ocurrencia

```
procedure eliminarLaPrimeraOcurrenciaEnUnaListaDesordenada(var L: lista; dni: integer; ok: boolean);
var
  ant, act: lista;
begin
  ant:= L;
  act:= L;
  ok:= false;
  While(act <> nil)and(dni <> act^.dato.dni)do; // mientras no se encuentre el dni y no terminemos la lista
  begin
    ant:= act;
    act:= act^.sig;
  end;
  if(act <> nil)then //no es vacia la lista y encuentre el nodo a eliminar
  begin
    ok:= true;
    if(act = L)then //el elemento a eliminar es el primero
      L:= act^.sig;
    else //el elemento es algun otro, pero no el primero
      ant^.sig:= act^.sig;
    dispose(act); //se elima el actual
  end;
end;
```

TodasLasOcurrencias

```
procedure elimarTodasLasOcurrenciasDesordenado(var L: lista; valor: integer);
var
  ant, act: lista;
begin
  ant:= L;
  act:= L;
  While(act <> nil)do // mientras no terminemos la lista
  begin
    if(valor <> act^.dato)then
    begin
      ant:= act;
      act:= act^.sig;
    end
    else //no es vacia la lista y encuentre un nodo a eliminar
    begin
      if(act = L)then //el elemento a eliminar es el primero
      begin
        L:= act^.sig;
        ant:= L;
      end
      else //el elemento es algun otro, pero no el primero
        ant^.sig:= act^.sig;
      dispose(act); //se elima el actual
      act:= ant;
    end;
  end;
end;
```

# Eliminar En Listas

## Ordenado

### Primera Ocurrencia

```

procedure eliminarLaPrimeraOcurrenciaListaOrdenada(var L: lista; dni: integer);
var
  ant, act: lista;
  contIteraciones: integer;
begin
  <!-- Writeln('Flag 0'); -->
  ant := L;
  act := L;
  <!-- contIteraciones := 0; -->
  While(act <> nil) and (dni > act^.dato) do //ascendente // mientras no se encuentre el dni y el dni actual no sea mayor al buscar
  begin
    ant := act;
    act := act^.sig;
    contIteraciones := contIteraciones + 1;
    <!-- Writeln('Flag 1'); -->
  end;
  <!-- Writeln('Flag 2'); -->
  if (act <> nil) and (dni = act^.dato) then //no es vacia la lista y encontro el nodo a eliminar
  begin
    if (act = L) then //el elemento a eliminar es el primero
      L := act^.sig;
    else //el elemento es algun otro, pero no el primero
      ant^.sig := act^.sig;
    dispose(act); //se elimina el actual
    //contIteraciones := contIteraciones + 1;
  end;
  <!-- Writeln('Cant Iteraciones ', contIteraciones); -->
end;

```

### Todas Las Ocurrencias

```

procedure buscarOrdenado(var act, ant: lista; d: integer);
begin
  while (act <> nil) and (act^.dato < d) do begin
    ant := act;
    act := act^.sig;
  end;
end;

procedure eliminarNodo (var pri, act, ant: lista);
var
  aux: lista;
begin
  aux := act^.sig;
  if (act = pri) then
    pri := pri^.sig;
  else
    ant^.sig := act^.sig;
  dispose(act);
  act := aux;
end;

procedure eliminarTodasLasOcurrenciasOrdenado (var L: lista; d: integer);
var
  act, ant: lista;
begin
  act := L;
  ant := L;
  buscarOrdenado(act, ant, d);

  while (act <> nil) and (act^.dato = d) do
    eliminarNodo(L, act, ant);
end;

```

# Vectores

## EliminarPosVector

```

Procedure eliminarPosVector(var v: vector; var ok: boolean; pos: integer; var dimL: integer);
var i: integer;
Begin
  ok:= ((pos > 0) and (pos <= dimL)); //verifico q la pos sea valida
  if(ok)then
    begin
      for i:= pos to (dimL-1) do //hasta menos 1 porq se "elima una posicion"
        begin
          v[pos]:= v[i+1]; //basicamente haces desplazamientos, 1:= [i+1=2]; 2:= [i+1=3] etc etc..
        end;
        dimL:= dimL-1; //decremento la dimL, ya q "borre un elemento del vector"
      end;
    end;
end;

```

## InsertarEnVector

```

Procedure insertar(var v: vector; var ok: boolean; pos: integer; var dimL: integer; numOcosa: integer);
var i: integer;
Begin
  ok:= ((pos >= 1) and (pos <= dimL) and (dimL+1 <= dimF)); //verifico q la pos sea valida
  if(ok)then
    begin
      for i:= dimL downto pos do //Arranco en la posDimL y voy hasta pos
        begin
          v[i+1]:= v[i]; //En la posActual+1, me cargo lo que hay en la posActual, son desplazamientos
        end;
        v[pos] := numOcosa; // Asigno el nuevo valor en la posición indicada
        dimL:= dimL+1; //incremento la dimL, ya q inserte un nuevo valor en el vector
      end;
    end;
end;

```

## InsertarOrdenado

```
function buscarPosicion(v: vEmpleado; dimL: integer; cod: integer): integer;
var
  pos: integer;
begin
  pos:= 1;
  //lo va dejar ordenado tipo 1 2 3 4 5
  While(pos <= dimL) and (cod > v[pos].codPais)do
    begin
      pos:= pos+1;
    end;
  buscarPosicion:= pos;
end;

procedure insertarPosicion(var v: vEmpleado; var dimL: integer; pos: integer; e: empleado);
var
  i: integer;
begin
  if((pos >= 1) and (pos <= dimL) and (dimL+1 <= dimF2k)) then
    begin
      for i:= dimL downto pos do
        v[i+1]:= v[i];
      v[pos]:= e;
      dimL:= dimL+1;
    end;
end;

procedure insertarOrdenado(var v: vEmpleado; var dimL: integer; e: empleado);
var pos: integer;
begin
  pos:= buscarPosicion(v,dimL,e.codPais);
  insertarPosicion(v,dimL,pos,e);
end;

procedure cargarVector(var v: vEmpleado; var dimL: integer);
var
  e: empleado;
begin
  dimL:= 0;
  While(dimL < dimF2k)do
    begin
      leerEmpleados(e);
      insertarOrdenado(v,dimL,e);
    end;
end;
```

## OrdenarVector

```
{
  este metodo busca en todo el array el minimo y lo va posicionando al principiando uno a uno,
  busco [i] si es mas chico q algun elemento del array me lo guardo en la iteracion que este i
}

procedure ordenarVector(var v: vOrdenar; dimL: integer);
var a,b,i,min: integer;
begin
  for i:= 1 to (dimL-1) do
    begin
      a:= i; //me paro en la [x posicion]
      for b:= i+1 to dimL do
        begin
          if(v[b] < v[a])then {pregunta si i+1 es mayor al primer campo, si es asi, cambia de lugar y asi se
            begin
              a:= b; //en A tengo guardado la posicion del minimo de todo el vector
            end;
          {aca hacen el swap, intercambia los valores en sus posiciones correspondientes
          Minmo lo guarda en la iteracion i, y lo que habia en i posicion lo intercambia en el lugar de la po
          min:= v[a]; //salvo el valor del minimo
          v[a]:= v[i]; //swap de valores
          v[i]:= min; //guardo en la pos i el valor minimo de todo el vector
        end;
      end;
    end;
```

Ordena de menor a mayor < '  
Ordena de mayor a menor > '