Representación secuencial

```
clase Lista
inicio
        entero ultimo
        Objeto [ ] lista
        Lista()
        inicio
                ultimo ← 0
                lista ← nuevo Objeto[MAX]
        fin
fin
```

Creación de la lista

```
Lista()
        inicio
                ultimo ← 0
                lista ← nuevo Objeto[MAX]
        fin
```

Determinar si una lista esta vacía

```
boolean vacia(Lista lista)
inicio
        boolean vacia ←falso
        si(lista.ultimo=0)
        inicio
                vacia ← verdadero
        fin
        regresa vacia
fin
```

Insertar un elemento al final

```
Lista insertar (Lista lista, Objeto objeto)
inicio
        si(lista.ultimo = MAX)
        inicio
                IMPRIMIR “La lista está llena”
        fin
        otro
        inicio
                lista.lista[lista.ultimo]←objeto
                lista.ultimo+=1;
        fin
        regresa lista
fin
```

Insertar en una determinada posición

```
Lista insertarPorPosicion (Lista lista, Objeto objeto, int pos)
inicio
        si (lista.ultimo = MAX)
        inicio
                IMPRIMIR “La lista está llena”
        fin
        otro si (p>lista.ultimo)
        inicio
                lista = insertar (lista, objeto)
        fin
        otro
        inicio
                lista ← moverAbajo(lista, p)
                lista.lista[p]←objeto
        fin
        regresa lista
fin
```

Insertar los elementos en orden

```
Lista insertarOrdenado(Lista lista, Objeto objeto)
inicio
        entero p ← 0
        si (lista.ultimo = MAX)
        inicio
                IMPRIMIR “La lista está llena”
        fin
        otro si (vacia(lista)
        inicio
                lista = insertar(lista,objeto)
        fin
        otro
        inicio
                p ← buscarPosicionInsertar(lista,objeto)
                lista ← insertarPorPosicion(lista,objeto,p)
        fin
        regresa lista
fin
```

Borrar un elemento

```
Lista borrarElemento(Lista lista, Objeto objeto)
inicio
        entero p ← 0
        p ← encontrar (objeto)
        si (p<0)
        inicio
                IMPRIMIR “No existe el elemento a borrar”
        fin
        otro
        inicio
                lista ← moverArriba(lista,p)
        fin
        regresa lista
fin
```

Recorrer toda la lista

```
void recorrer(Lista lista)
inicio
        para (i←0 hasta i ← lista.ultimo i+=1)
        inicio
                Objeto aux ← lista.lista[i]
                IMPRIMIR (objeto.atributo)
        fin
fin
```

Buscar un elemento

```
entero encontrar(Lista lista, Objeto objeto)
inicio
        entero pos ← -1
        entero i ← 0
        mientras(i<lista.ultimo)
        inicio
                Objeto aux = lista.lista[i]
                si(aux.atributo = objeto.atributo)
                inicio
                        posicion ← i
                        interrumpe
                fin
                otro
                inicio
                        i ← i+1
                fin
        regresa pos
        fin
fin
```

Corrimiento hacia arriba

```
Lista moverArriba(Lista lista, entero p)
inicio
        entero aux
        Objeto [ ] listaTmp
        aux ← p
        listaTmp ← lista.lista
        mientras(aux < lista.ultimo)
        inicio
                listaTmp[aux]←listaTmp[aux+1]
                aux ← aux +1
        fin
        lista.lista ← listaTmp
        lista.ultimo ← lista.ultimo-1
        regresa lista
fin
```

Corrimiento hacia abajo

```
Lista moverAbajo(Lista lista, entero p)
inicio
        entero aux
        Objeto [ ] listaTmp
        lista.ultimo ← lista.ultimo+1
        aux ← lista.ultimo
        listaTmp ← lista.lista
        mientras(aux >p)
        inicio
                listaTmp[aux] ← listaTmp[aux-1]
                aux←aux-1
        fin
        lista.lista ← listaTmp
        regresa lista
fin
```

Buscar posición para insertar

```
entero buscarPosicionInsertar(Lista lista, Objeto objeto)
inicio
        entero pos ← 0
        Objeto aux ← lista.lista[0]
        mientras (pos < lista.ultimo)
        inicio
                aux ← lista.lista[pos]
                si(objeto.atributo < aux.atributo)
                inicio
                        interrumpe
                fin
                otro
                inicio
                        pos ← pos +1
                fin
         fin
                regresa pos
fin
```