Representación ligada.

Definición de un Nodo

```
clase Nodo
inicio
        Objeto objeto
        Nodo siguiente

        Nodo()
        inicio
        fin

        Nodo(Objeto objeto, Nodo siguiente)
        inicio
        siguiente  ← siguiente
        objeto ← objeto
        fin
fin
```

Definición de una Lista

```
clase Lista
inicio

        Nodo cabeza

        Lista()
        inicio
                cabeza ← null
        fin


fin
```

Determinar si una lista esta vacía

```
boolean vacia(Lista lista)
inicio
        boolean vacia ←falso
        si(listacabeza=null)
        inicio
                vacia ← verdadero
        fin
        regresa vacia
fin
```

Recorrer una lista

```
void imprimir(Lista lista)
inicio
        Nodo aux  ←lista.cabeza
        mientras(aux!=null)
        inicio
                imprime (lista.objeto)
                aux ←aux.siguiente
        fin
fin
```

Buscar un elemento en la lista

```
Nodo buscar(Lista lista, Objeto objeto)
inicio
        Nodo aux  ←lista.cabeza
        Nodo buscado ← null
        mientras(aux != null)
        inicio
                si(aux.objeto = objeto)
                        inicio
                                buscado ← aux
                                interrumpe
                        fin
        otro
                inicio
                        aux  ← aux.siguiente
                fin
        fin
        regresa buscado
fin
```

Insertar al inicio de la lista

```
Lista insertarAlInicio(Lista lista, Objeto objeto)
inicio
        si(vacia(lista))
        inicio
                Nodo nuevo ← nuevo Nodo(objeto,null)
                lista.cabeza ← nuevo
        fin
        otro
        incio
                Nodo nuevo ← nuevo Nodo(objeto,null)
                Nodo aux ← lista.cabeza
                nuevo.siguiente ← aux
                lista.cabeza ← nuevo
        fin
                regresa lista
fin
```

Insertar al final de la lista

```
Lista insertarAlFinal(Lista lista, Objeto objeto)
inicio
        si(vacia(lista))
        inicio
                Nodo nuevo ← nuevo Nodo(objeto,null)
                lista.cabeza ← nuevo
        fin
        otro
        incio
                Nodo nuevo ← nuevo Nodo(objeto,null)
                Nodo aux ← lista.cabeza
                mientras(aux.siguiente !=null)
                inicio
                        aux ← aux.siguiente
                fin
                aux.siguiente ← nuevo
        fin
                regresa lista
fin
```

Insertar elementos en orden

```
Lista insertarEnOrden(Lista lista, Objeto objeto)
inicio
	si(vacia(lista))
	inicio
		Nodo nuevo ← nuevo Nodo(objeto,null)
		lista.cabeza ← nuevo
	fin
	otro
	incio
		Nodo nuevo ← nuevo Nodo(objeto,null)
		Nodo anterior ← null
		Nodo aux ← lista.cabeza
		si(aux.objeto > objeto)
		inicio
			lista ← insertarAlInicio(lista, objeto)
		fin
		otro
		inicio
			mientras(aux.siguiente !=null)
			inicio
				si(aux.objeto<objeto)
				inicio
					anterior ←  aux
					aux ←  aux.siguiente
				fin
				otro
				inicio
					interrumpe
				fin
			fin
			si(aux.siguiente = null Y aux.objeto<objeto)
			inicio
				lista ← insertaralFinal(lista,objeto)
			fin
			otro
			inicio
			anterior.siguiente ←  nuevo
			nuevo.siguiente ←  aux
			fin
		fin
	fin
		regresa lista
fin
```

Borrar el primer elemento

```
Lista borrarAlInicio(Lista lista)
inicio
        si(vacia(lista))
        inicio
                IMPRIMIR La lista esta vacía
        fin
        otro
        incio
                Nodo aux ← lista.cabeza
                lista.cabeza ←lista.cabeza.siguiente
        fin
                regresa lista
fin
```

Borrar el último elemento

```
Lista borrarAlFinal(Lista lista)
inicio
        si(vacia(lista))
        inicio
                IMPRIMIR La lista esta vacía
        fin
        otro
        incio
                Nodo aux ← lista.cabeza
                Nodo anterior ← null
                mientras (aux.siguiente != null)
                inicio
                        anterior ← aux
                        aux ←aux.siguiente
                fin
                si ( anterior != null)
                inicio
                        anterior.siguiente ← null
                fin
                otro
                inicio
                        lista.cabeza← null
                fin
        fin
                regresa lista
fin
```

Borrar un elemento intermedio

```
Lista borrarElemento(Lista lista, Objeto objeto)
inicio
        si(vacia(lista))
        inicio
                IMPRIMIR La lista esta vacía
        fin
        otro
        incio
                si(buscar(lista,objeto)=null)
                inicio
                        IMPRIMIR El elemento no se encuentra
                fin
                otro
                inicio
                        Nodo aux ← lista.cabeza
                        Nodo anterior ← null
                        mientras (aux.objeto != objeto  Y aux.siguiente != null)
                        inicio
                                anterior ← aux
                                aux ←aux.siguiente
                        fin
                        si ( anterior = null)
                        inicio
                                lista  ← borrarAlInicio(lista)
                        fin
                        otro si(aux.siguiente = null)
                        inicio
                                lista ← borrarAlFinal(lista)
                        fin
                        otro
                        inicio
                                anterior.siguiente ←aux.siguiente
                        fin
                fin

        fin
                regresa lista
fin
```