Cola:

* Características de la estructura:
* Cantidad de elementos: Variable (puede crecer o decrecer).
* Homogeneidad: Homogénea, todos los elementos son del mismo tipo.
* Forma de acceso: FIFO (First In, First Out).
* Inserción y eliminación: Inserción al final, eliminación al inicio.
* Representación: Arreglos o listas enlazadas.
* Ejemplo: Personas en fila de espera, el primero en llegar es el primero en irse después de ser atendido.
* Operaciones que “tienen sentido” para la estructura:
* *enqueue*: Agregar un elemento al final de la cola.
* *dequeue*: Eliminar el elemento del inicio.
* *peek/front*: Ver el elemento en el inicio.
* *isEmpty*: Verificar si la cola está vacía.
* *size*: Retornar la cantidad de elementos.
* Operaciones que “no tengan sentido” para la estructura:
* *insertAt(index)*: No se insertan elementos en posiciones arbitrarias.
* *removeAt(index)*: Tampoco se eliminan elementos por posición.
* *getLast*: El acceso directo al final no respeta la idea FIFO.
* Operaciones “con sentido” pero no indispensables:
* *clear()*: Vaciar la cola por completo.
* *contains(element)*: Verificar si un elemento está en la cola (requiere recorrerla).
* *toArray()*: Convertir la cola en un arreglo para facilitar otras operaciones.
* *reverse()*: Dar vuelta la cola

Lista enlazada:

* Características de la estructura:
* Cantidad de elementos: Variable (puede crecer o decrecer linealmente).
* Homogeneidad: Homogénea, todos los nodos contienen el mismo tipo de dato.
* Forma de acceso: Secuencial, se accede nodo por nodo desde el inicio.
* Conectividad: Cada nodo contiene un valor y un puntero al siguiente nodo.
* Representación: Cada nodo está conectado en una cadena lineal.
* Uso común: Manejo eficiente de inserciones y eliminaciones sin necesidad de mover elementos.
* Operaciones que “tienen sentido” para la estructura:
* *insertFirst()*: Insertar un nodo al principio de la lista.
* *insertLast()*: Insertar un nodo al final de la lista.
* *removeFirst()*: Eliminar el primer nodo.
* *traverse()*: Recorrer la lista desde el primer nodo hasta el final.
* *search(value)*: Buscar un valor recorriendo la lista.
* *size()*: Devolver la cantidad de nodos.
* Operaciones que “no tengan sentido” para la estructura:
* *randomAccess(index)*: Las listas enlazadas no permiten acceso por índice como los arreglos.
* *sortInPlace()*: Requiere recorrer y reorganizar enlaces; no es natural como en un arreglo.
* Operaciones “con sentido” pero no indispensables:
* insertAs(position): Se puede hacer, pero requiere recorrer hasta la posición deseada.
* removeAt(position): También se puede, pero no es directa como en un arreglo.
* reverse(): Se puede invertir la lista modificando los punteros.
* toArray(): Convertir la lista a un arreglo para operar con índice.

Lista doblemente enlazada:

* Características de la estructura:
* Cantidad de elementos: Variable, se ajusta dinámicamente.
* Homogeneidad: Homogénea, todos los nodos contienen el mismo tipo de dato.
* Forma de acceso: Secuencial, pero en ambos sentidos (hacia adelante y hacia atrás).
* Conectividad: Cada nodo tiene un puntero al siguiente y al anterior.
* Ventaja: Más eficiente para operaciones que requieren retroceso o eliminación intermedia.
* Representación: Cabeza ↔ Nodo1 ↔ Nodo2 ↔ Nodo3 ↔ Cola
* Operaciones que “tienen sentido” para la estructura:
* *insertFirst()*: Inserta un nodo al principio.
* *insertLast()*: Inserta un nodo al final.
* *removeFirst()*: Elimina el nodo al comienzo.
* *removeLast()*: Elimina el nodo al final.
* *traverseForward()*: Recorrer desde el principio.
* *traverseBackward()*: Recorrer desde el final (gracias al puntero al nodo anterior).
* *search(value)*: Buscar un valor.
* *insertAfter(node)*: Insertar un nodo luego de uno específico.
* *delete(node)*: Eliminar un nodo intermedio (sin recorrer toda la lista).
* Operaciones que “no tengan sentido” para la estructura:
* *randomAccess(index)*: Aunque se puede recorrer, no se accede por índice como un arreglo.
* *sortInPlace()*: Ordenar requiere recorrer y modificar muchos punteros, no es natural.
* Operaciones “con sentido” pero no indispensables:
* *insertAt(position)* : Se puede, pero implica recorrer desde el comienzo o final.
* *reverse()*: Invertir la lista es directo, solo intercambiando punteros.
* *toArray()*: Convertirla en un arreglo para operar más fácilmente con índices.

Cola doblemente terminada:

* Características de la estructura:
* Cantidad de elementos: Variable, se puede agregar o sacar elementos dinámicamente.
* Homogeneidad: Homogénea, todos los elementos son del mismo tipo.
* Forma de acceso: Acceso por ambos extremos, se puede insertar y eliminar tanto al inicio como al final.
* Flexibilidad: Combina comportamientos de pilas y colas.
* Implementación: Arreglos dinámicos o listas doblemente enlazadas.
* Operaciones que “tienen sentido” para la estructura:
* *addFirst()*: Insertar un elemento al inicio de la cola.
* *addLast()*: Insertar un elemento al final de la cola.
* *removeFirst()*: Eliminar el elemento del inicio.
* *removeLast()*: Eliminar el elemento del final.
* *peekFirst()*: Ver el primer elemento sin eliminarlo.
* *peekLast()*: Ver el último elemento sin eliminarlo.
* *isEmpty()*: Verificar si la cola está vacía.
* *size()*: Retornar la cantidad de elementos.
* Operaciones que “no tengan sentido” para la estructura:
* *insertAt(position)*: Acceder a una posición arbitraria rompe el diseño por extremos.
* *removeAt(index)*: Eliminar por índice no es coherente con su uso como estructura de extremos.
* Operaciones “con sentido” pero no indispensables:
* *clear()*: Vaciar completamente la estructura
* *contains(element)*: Ver si contiene cierto elemento (requiere recorrer)
* *toArray()*: Convertir en un arreglo si se necesita manipular con índices