# Nodo

## Estructura

| *typedef struct* Node {  *int* element;  *struct* Node \*prev;  *struct* Node \*next;  } Node; |
| --- |

## Métodos

| Node\* createNode(*int* element); | |
| --- | --- |
| *Crea un nuevo nodo que contiene el elemento especificado.*  *@param* ***element*** *El elemento*  ***@return*** *El nodo creado* | |

| *void* deleteNode(Node\* node); | |
| --- | --- |
| *Elimina el nodo especificado, liberando el bloque de memoria previamente asignado al mismo.*  ***@param******node*** *El nodo* | |

# Lista

## Estructura

| *typedef struct* List {  *int* size;  Node \*first;  Node \*last;  } List; |
| --- |

## Métodos

| *List \*CreateList();* | |
| --- | --- |
| *Crea una nueva lista vacía.*  ***@return*** *La lista* | |
| **Lista antes de llamar al método** | **Lista después de llamar al método** |
| **NULL** | [] |

| *void addFirst(List \*list, int element);* | | |
| --- | --- | --- |
| *Añade el elemento como el primer elemento en lista.*  ***@param******list*** *La lista*  ***@param******element*** *El elemento a añadir* | | |
| **Elemento** | **Lista antes de llamar al método** | **Lista después de llamar al método** |
| 0 | [2, 4, 6, 8, 10] | [0, 2, 4, 6, 8, 10] |

| *void addLast(List \*list, int element);* | | |
| --- | --- | --- |
| *Añade el elemento como el último elemento en lista.*  ***@param******list*** *La lista*  ***@param******element*** *El elemento a añadir* | | |
| **Elemento** | **Lista antes de llamar al método** | **Lista después de llamar al método** |
| 12 | [2, 4, 6, 8, 10] | [2, 4, 6, 8, 10, 12] |

| *void add(List \*list, int element, int index);* | | | |
| --- | --- | --- | --- |
| *Inserta el elemento en la posición especificada en lista, desplazando el elemento que está ocupando actualmente esa posición (si lo hay) y cualquier elemento posterior hacia la derecha.*  ***@param******list*** *La lista*  ***@param******element*** *El elemento a añadir*  ***@param******index*** *La posición en la que será añadido el elemento* | | | |
| **Elemento** | **Índice** | **Lista antes de llamar al método** | **Lista después de llamar al método** |
| 7 | 3 | [2, 4, 6, 8, 10] | [2, 4, 6, 7, 8, 10] |
| 0 | 0 | [0, 2, 4, 6, 8, 10] |
| 12 | 5 | [2, 4, 6, 8, 10, 12] |

**Nota**: Según lo evidenciado a través de las últimas dos llamadas, se puede decir que las siguientes llamadas son equivalentes:

| *add(list, element, 0) = addFirst(l, e)*  *add(list, element, n) = addLast(l, e)* |
| --- |

Donde ‘n’ es el tamaño de la lista. Entonces, ambos casos son **casos especiales** de inserción en lista.

| *void addAll(List \*this, List \*other);* | | |
| --- | --- | --- |
| *Añade todos los elementos de la segunda lista al final de la primera*  *lista.*  ***@param******this*** *La lista a la que se le añadirán los elementos*  ***@param******other*** *La lista que contiene los elementos a añadir* | | |
| **Lista 1** | **Lista 2** | **Lista 1 después de llamar al método** |
| [2, 4, 6, 8, 10] | [1, 3, 5, 7, 9] | [2, 4, 6, 8, 10, 1, 3, 5, 7, 9] |

| *int deleteFirst(List \*list);* | | |
| --- | --- | --- |
| *Elimina el primer elemento en lista, y lo devuelve.*  ***@param******list*** *La lista*  ***@return*** *El primer elemento en lista* | | |
| **Lista antes de llamar al método** | **Lista después de llamar al método** | **Retorna** |
| [2, 4, 6, 8, 10] | [4, 6, 8, 10] | 2 |

| *int deleteLast(List \*list);* | | |
| --- | --- | --- |
| *Elimina el último elemento en lista, y lo devuelve.*  ***@param******list*** *La lista*  ***@return*** *El último elemento en lista* | | |
| **Lista antes de llamar al método** | **Lista después de llamar al método** | **Retorna** |
| [2, 4, 6, 8, 10] | [2, 4, 6, 8] | 10 |

| *int delete(List \*list, int index);* | | | |
| --- | --- | --- | --- |
| *Elimina el elemento en la posición especificada en lista, desplazando cualquier elemento posterior hacia la izquierda. Luego, devuelve el elemento eliminado de la lista.*  ***@param******list*** *La lista*  ***@param******index*** *El índice del elemento a eliminar* | | | |
| **Índice** | **Lista antes de llamar al método** | **Lista después de llamar al método** | **Retorna** |
| 3 | [2, 4, 6, 8, 10] | [2, 4, 6, 10] | 8 |
| 0 | [4, 6, 8, 10] | 2 |
| 4 | [2, 4, 6, 8] | 10 |

**Nota**: Según lo evidenciado a través de las últimas dos llamadas, se puede decir que las siguientes llamadas son equivalentes:

| *delete(list, 0) = deleteFirst(l)*  *delete(list, n - 1) = deleteLast(l)* |
| --- |

Donde ‘n’ es el tamaño de la lista. Entonces, ambos casos son **casos especiales** de inserción en lista.

| *int indexOf(List \*list, int element);* | | |
| --- | --- | --- |
| *Devuelve el índice de la primera vez que aparece el elemento en la lista, o -1 si esta no contiene el elemento.*  ***@param******list*** *La lista*  ***@param******element*** *El elemento a buscar*  ***@return*** *El índice de la primera vez que aparece el elemento en la lista, o -1* | | |
| **Elemento** | **Lista** | **Retorna** |
| 6 | [2, 4, 6, 8, 10] | 2 |
| 12 | -1 |

| ***bool*** *contains(List \*list, int element);* | | |
| --- | --- | --- |
| *Determina si la lista contiene el elemento especificado.*  ***@param******list*** *La lista*  ***@param******element*** *El elemento a buscar*  ***@return*** *true si la lista contiene el elemento; de lo contrario, false.* | | |
| **Elemento** | **Lista** | **Retorna** |
| 8 | [2, 4, 6, 8, 10] | true |
| 0 | false |

| *void* clearList(List \*list); | |
| --- | --- |
| *Elimina todos los elementos en lista, liberando los bloques de memoria previamente asignados a cada uno de ellos.*  *La lista quedará vacía después de que este método haya sido llamado.*  ***@param******list*** *La lista* | |
| **Lista antes de llamar al método** | **Lista después de llamar al método** |
| [2, 4, 6, 8, 10] | [] |

| *void* deleteList(List \*list); | |
| --- | --- |
| *Elimina la lista, liberando los bloques de memorias previamente asignados tanto a la lista misma como a los elementos presentes en ella. La lista será una referencia nula después de que este método haya sido llamado.*  *@param* ***list*** *La lista* | |
| **Lista antes de llamar al método** | **Lista después de llamar al método** |
| [2, 4, 6, 8, 10] | **NULL** |