UTEC
UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA

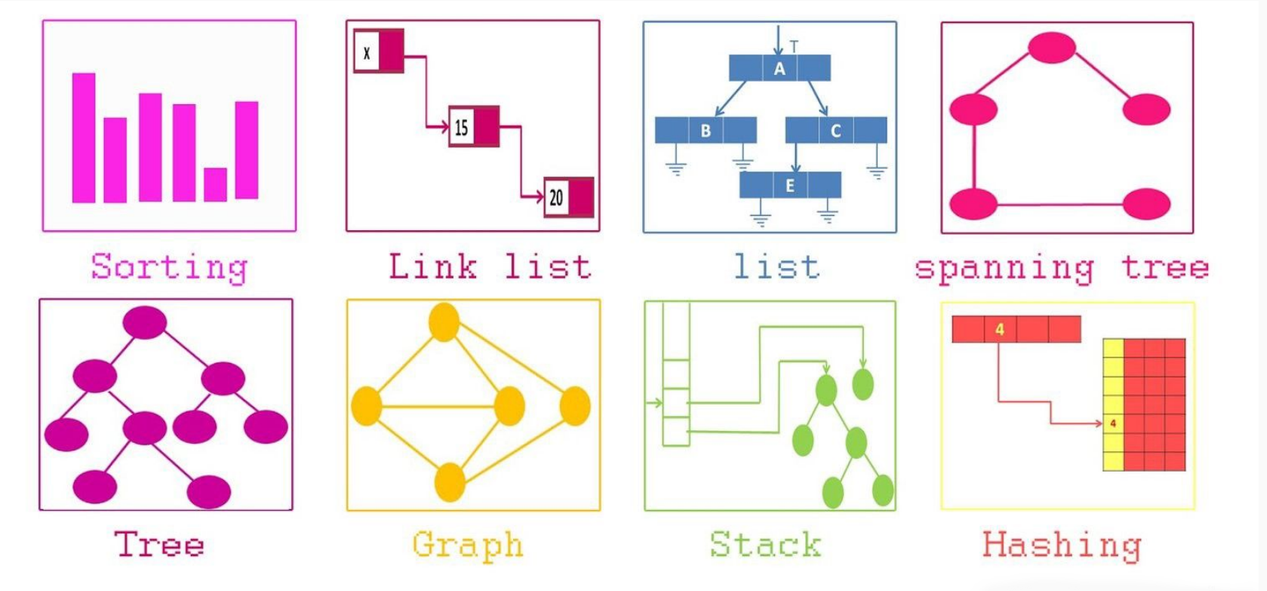
# Welcome to
# Algorithms and Data Structures!
# CS2100

# Qué es una estructura de datos?

Son colecciones que mantienen diferentes relaciones entre los datos que almacenan. Dichas estructuras permiten un eficiente acceso y modificación de dichos datos

Cada estructura de datos tiene un uso para diferentes problemas. Por ejemplo, una estructura de datos puede utilizarse para guardar información de personas en base a sus nombres

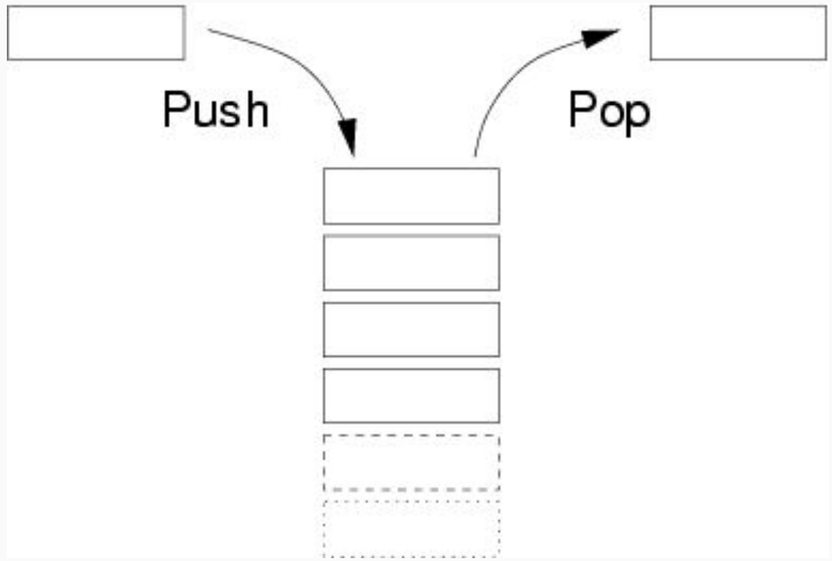
Son estructuras programadas para almacenar datos en memoria (RAM o disco) para que varias operaciones puedan realizarse de manera fácil



# Stack

Es una estructura de datos básica que se puede representar como una pila.

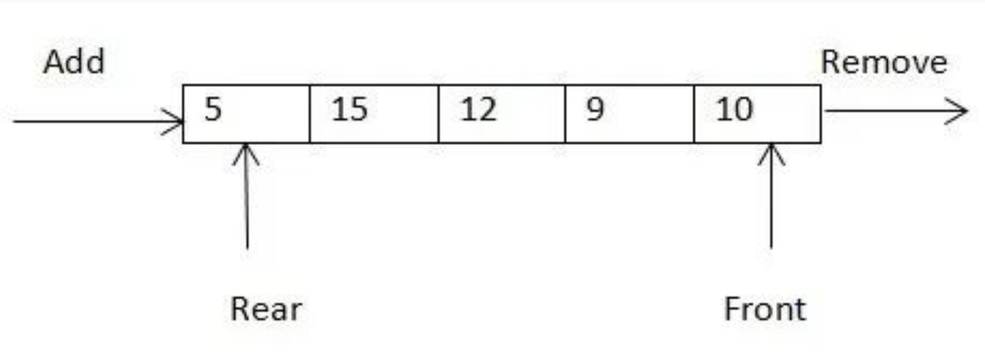
Su implementación se llama también LIFO (Last In First Out)



# Queue

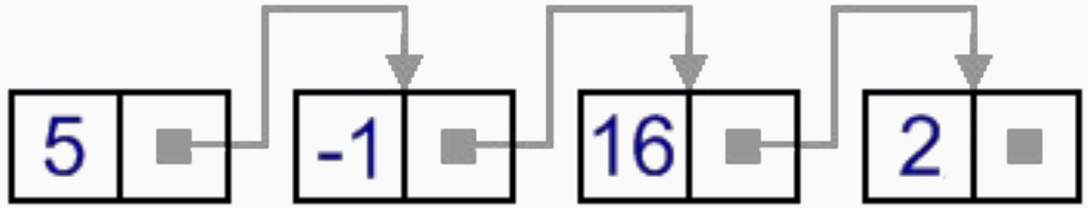
Es una estructura de datos básica muy similar a stack.

Su implementación se llama también FIFO (First In First Out)



# Forward List (Simple Linked List)

Es un contenedor secuencial, el cual permite tener datos en espacios de almacenamiento no relacionados (memoria)



# Forward List (Node)

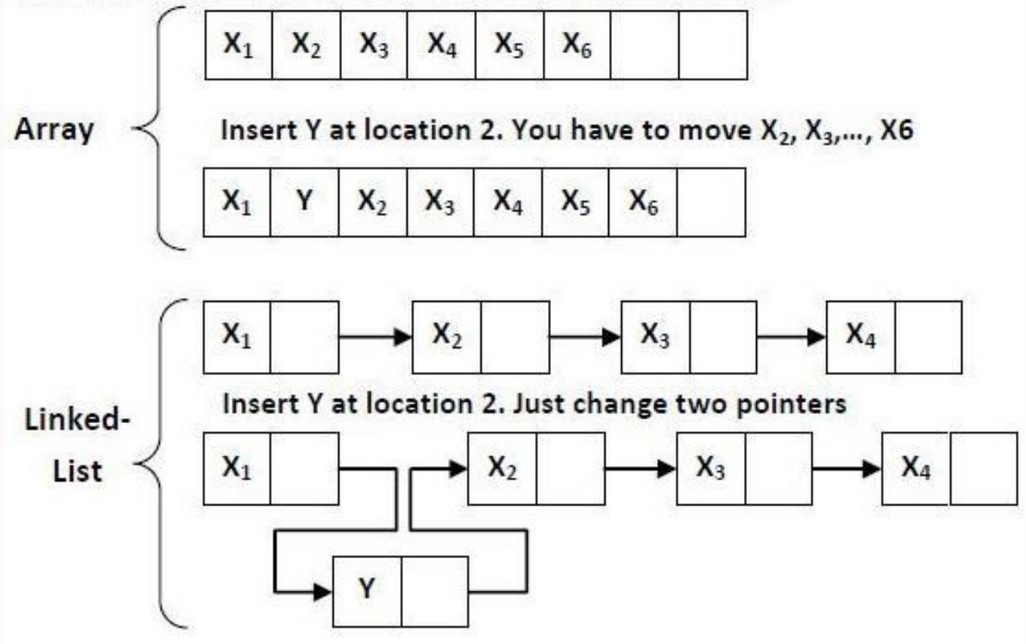
```
struct Node {
        int data;
        Node* next;
};
```

```
class List {
    private:
            Node* head;
…
};
```

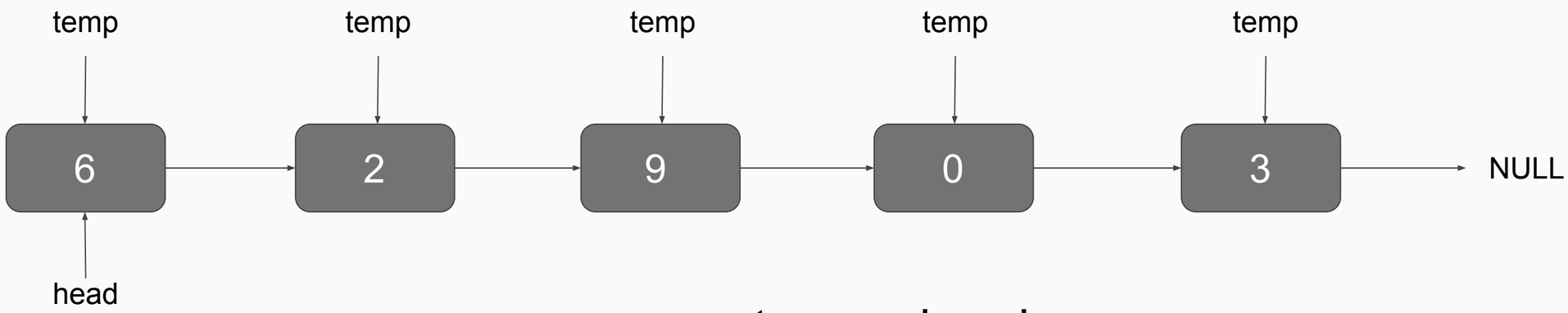
# Forward List VS Array

¿Cuál es la diferencia con arreglo?

- Ubicación en la memória
- Tiempos de acceso
- Tamaño
- Dimensiones

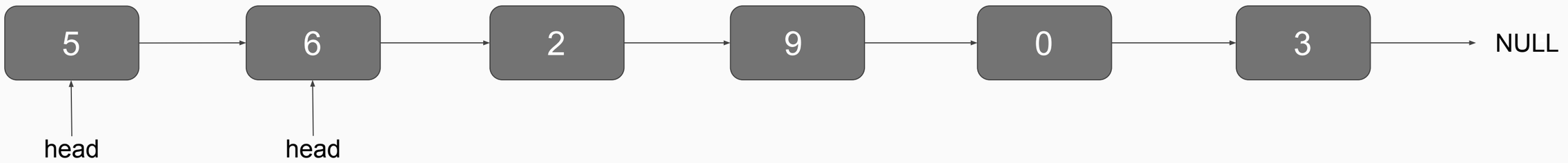


# Forward List

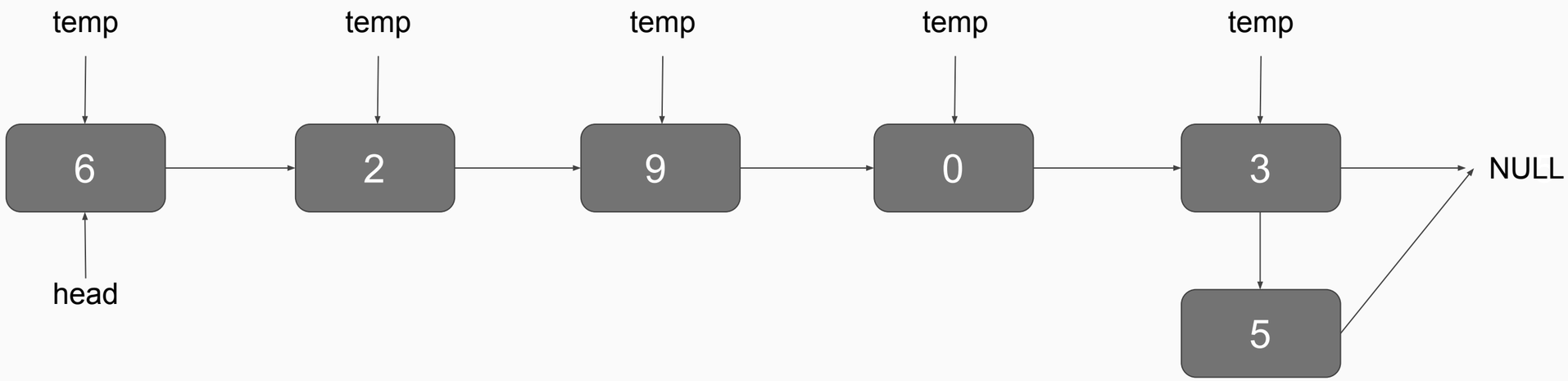



```
temp = head;
while (temp != null) {
    print temp->data;
    temp = temp->next;
}
```

# Forward List (push front 5)

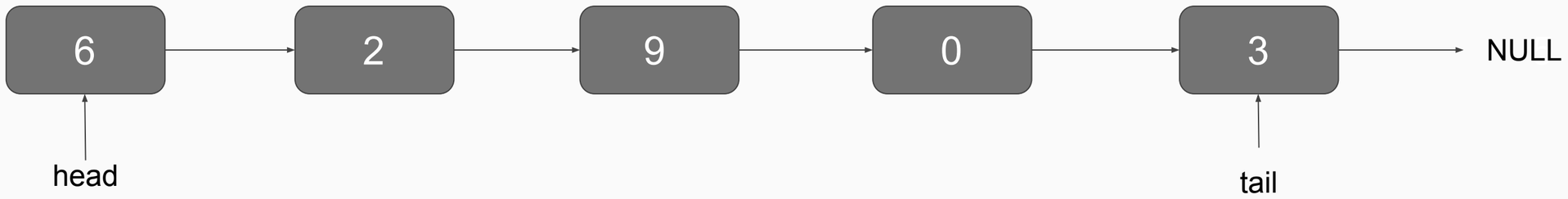


# Forward List (push back 5)



# Forward List (small improve)

Cuánto demoraría concatenar dos listas?




Cómo imprimiríamos la lista al revés?
Cómo borraríamos el último elemento?

# Forward List

1. **Cuánto tiempo demora encontrar un elemento al comienzo? Al final? En cualquier posición?**

   O(1), O(1) y O(n)

2. **Y para insertar un elemento después del primer nodo? Después del último nodo? Después de cualquier nodo?**

   O(1), O(1) y O(n)

3. **Cómo sería el caso 2 pero antes del nodo?**

   O(1), O(n) y O(n)

# Forward List

1. **Cuánto tiempo demora borrar un elemento al comienzo? Al final? En cualquier posición?**

   O(1), O(n) y O(n)

2. **Y para obtener el siguiente elemento de un nodo al comienzo? En cualquier posición?**

   O(1) y O(n)

3. **Cómo sería para obtener el nodo anterior al final? En cualquier posición?**

   O(n) y O(n)

# Forward List

1. **Se puede acelerar la operación de inserción en un nodo previo?**

   Reemplazando el contenido del nodo actual por el nuevo, y re-insertando el nodo después con el contenido antiguo.

2. **Se puede acelerar la operación de borrado de un elemento que no sea el último?**

   Copiando el contenido del siguiente nodo en el actual y borrando el siguiente.

# Lists

**T front();** *// Retorna el elemento al comienzo*

**T back();** *// Retorna el elemento al final*

**void push_front(T);** *// Agrega un elemento al comienzo*

**void push_back(T);** *// Agrega un elemento al final*

**void pop_front();** *// Remueve el elemento al comienzo*

**void pop_back();** *// Remueve el elemento al final*

**T operator[](int);** *// Retorna el elemento en la posición indicada*

**bool empty();** *// Retorna si la lista está vacía o no*

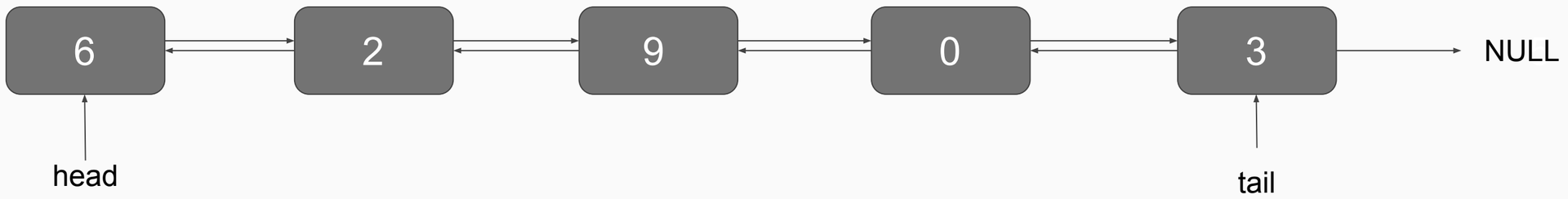
**int size();** *// Retorna el tamaño de la lista*

**void clear();** *// Elimina todos los elementos de la lista*

**void sort();** *// Ordena la lista*

**void reverse();** *// Revierte la lista*

# Doubly Linked List



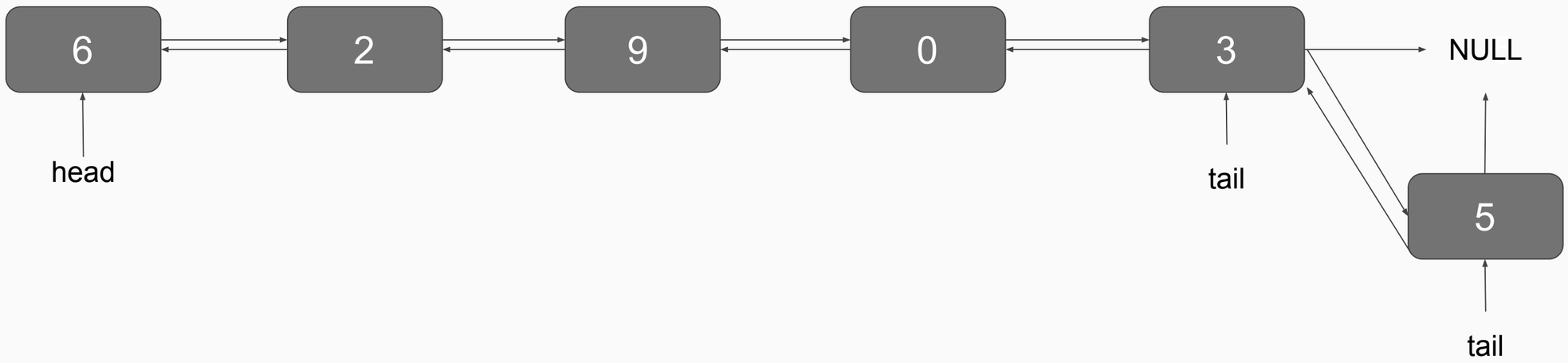

Cuáles son las ventajas?

# Doubly Linked List (node)

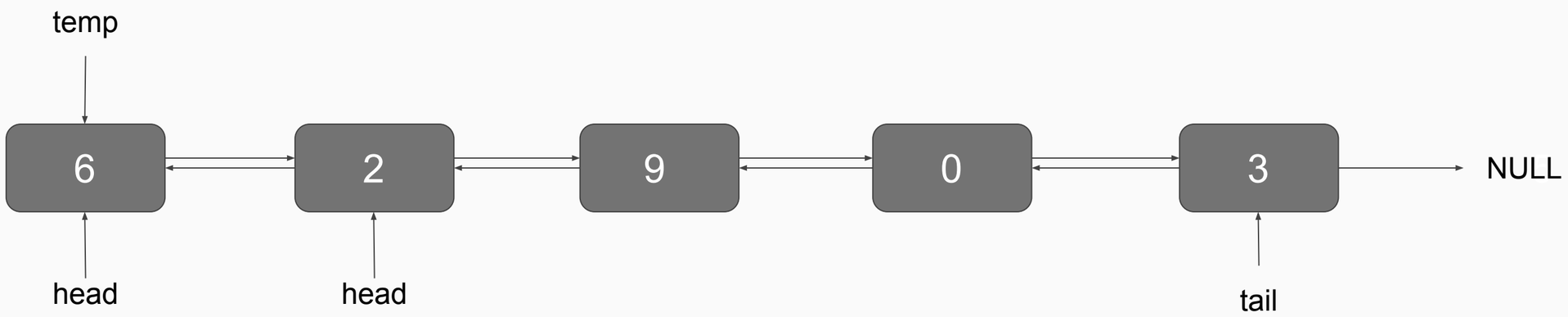
```
struct Node {
        int data;
        Node* next;
        Node* prev;
};
```

```
class List {
    private:
            Node* head;
            Node* tail;
};
```

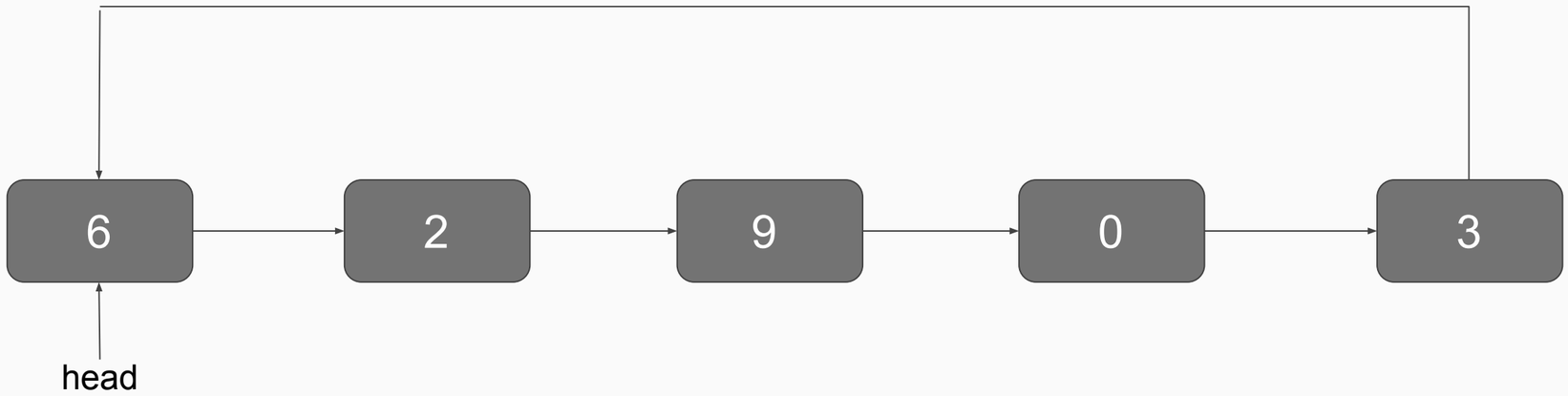
# Doubly Linked List (push back 5)



# Doubly Linked List (pop front)



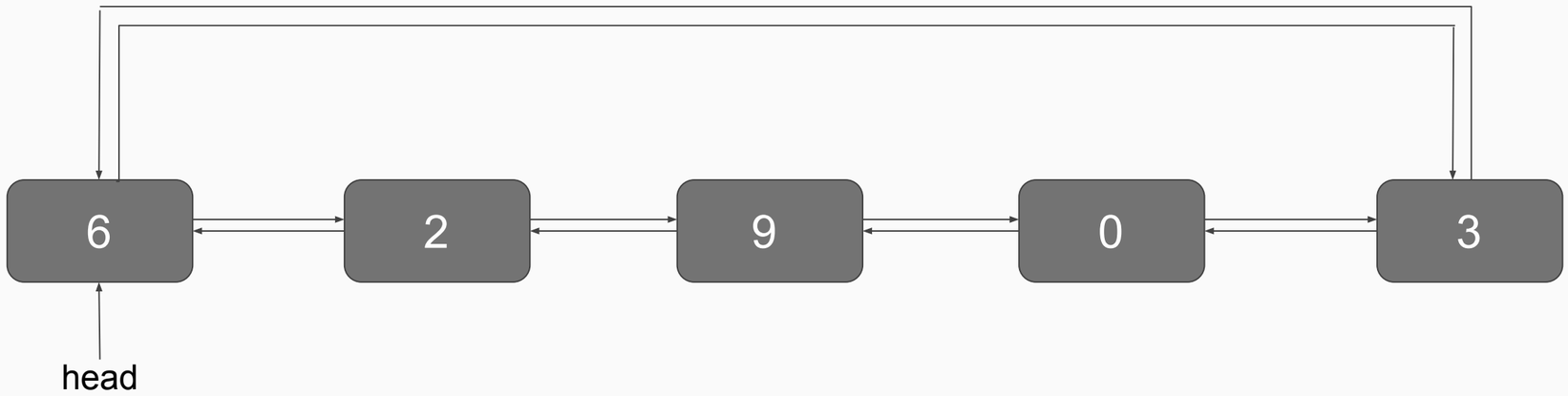
# Circular Linked List






Qué usos le darían a una lista circular?

# Circular Doubly Linked List




Siempre habrá un next y un prev

# Lists with templates

```
template <typename T>
struct Node {
    T data;
    Node<T>* next;
    Node<T>* prev;
};
```

```
typename <typename T>
class List {
  private:
    Node<T>* head;
    Node<T>* tail;
};
```

```
List<int>* test = new List<int>()
List<char>* test = new List<char>()
List<float>* test = new List<float>()
```

Cómo implementarían un array dinámico?

# Recuerden...

El curso se enfoca en que entiendan cada estructura de datos, sus algoritmos y usos comunes.

No hay que reinventar la rueda, en el mercado lo más probable es que utilicen otras implementaciones, por ejemplo de la Standard Template Library (STL).

De todas formas, habrá situaciones en las que probablemente implementarán sus propias estructuras.

UTEC
UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA

# Welcome to
# Algorithms and Data Structures!
# CS2100