TC1031: Programación de estructuras de datos y algoritmos fundamentales

Dr. Esteban Castillo Juarez
ITESM, Campus Santa Fe

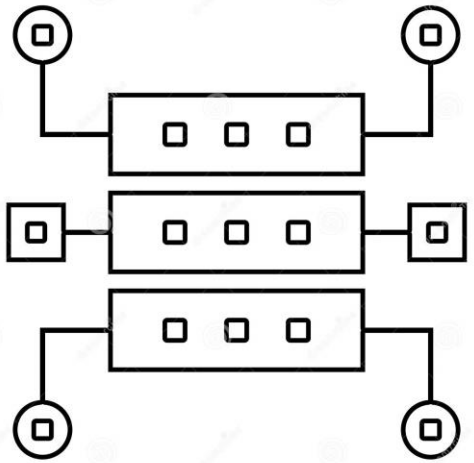
esteban.castillojz@tec.mx

Tecnológico de Monterrey

Estructuras de datos analizadas

1. Struct (nativo del lenguaje C) ✓
2. Arreglo unidimensional (vector) ✓
3. Arreglo bidimensional (matriz) ✓
4. Arreglo multidimensional ✓
5. Pila estática ✓
6. Cola estática ✓
7. Pila dinámica ✓
8. Cola dinámica ✓
9. Lista ligada ✓
10. Lista doblemente ligada ✗

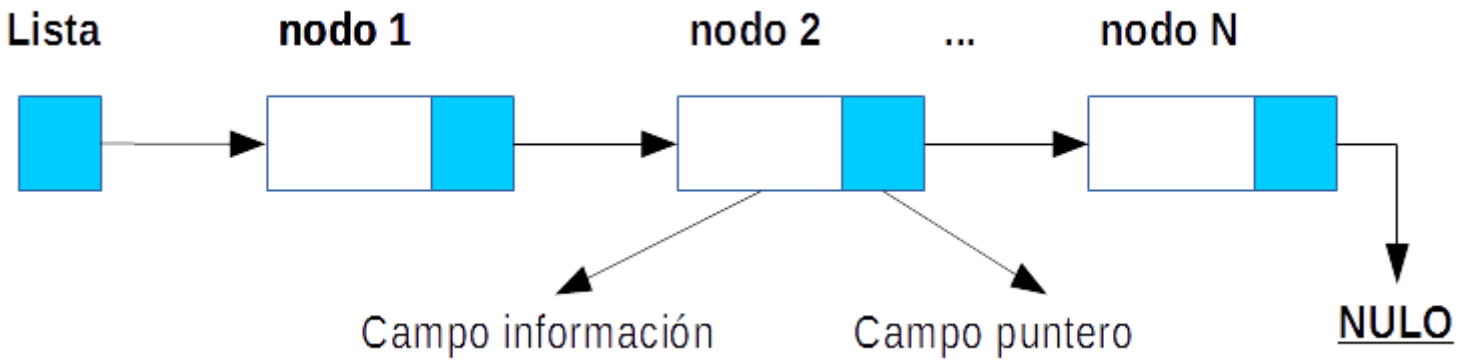
11. Tabla hash ✗
12. Arboles ✗
13. Arboles binarios ✗
14. Grafos ✗



Tecnológico de Monterrey

Lista doblemente ligada

- En un arreglo clásico de C se puede seleccionar una posición especifica de la estructura de datos sin mayor complicación. **En el caso de las listas ligadas** (y otras estructuras vistas en clase), se requiere iterar sobre cada elemento siguiendo una lógica especifica.



- Para resolver el problema anterior, aparece el concepto de **lista doblemente ligada**.

Tecnológico de Monterrey

Lista doblemente ligada

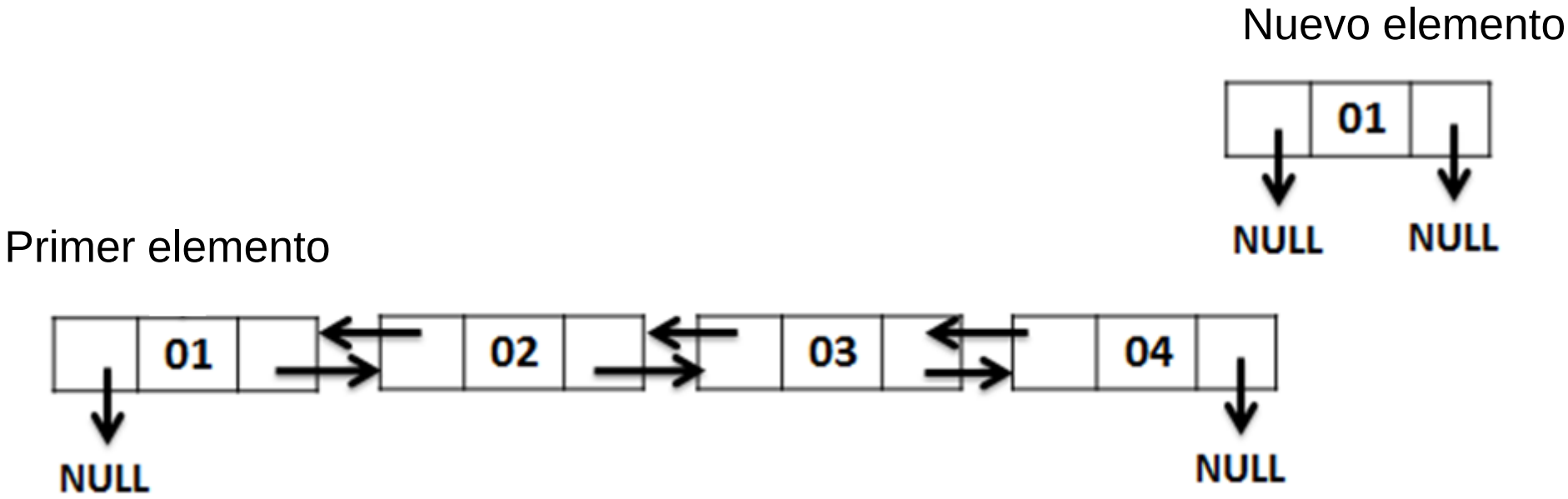
- Una lista doblemente ligada es una lista lineal en la que cada nodo tiene dos enlaces, uno al nodo siguiente y otro al anterior.

- Las listas doblemente ligadas no necesitan un nodo especial para acceder a ellas, pueden recorrerse en ambos sentidos a partir de cualquier nodo.

- A partir de cualquier nodo, siempre es posible alcanzar cualquier otro en lista doblemente ligada, hasta que se llega a uno de los extremos.

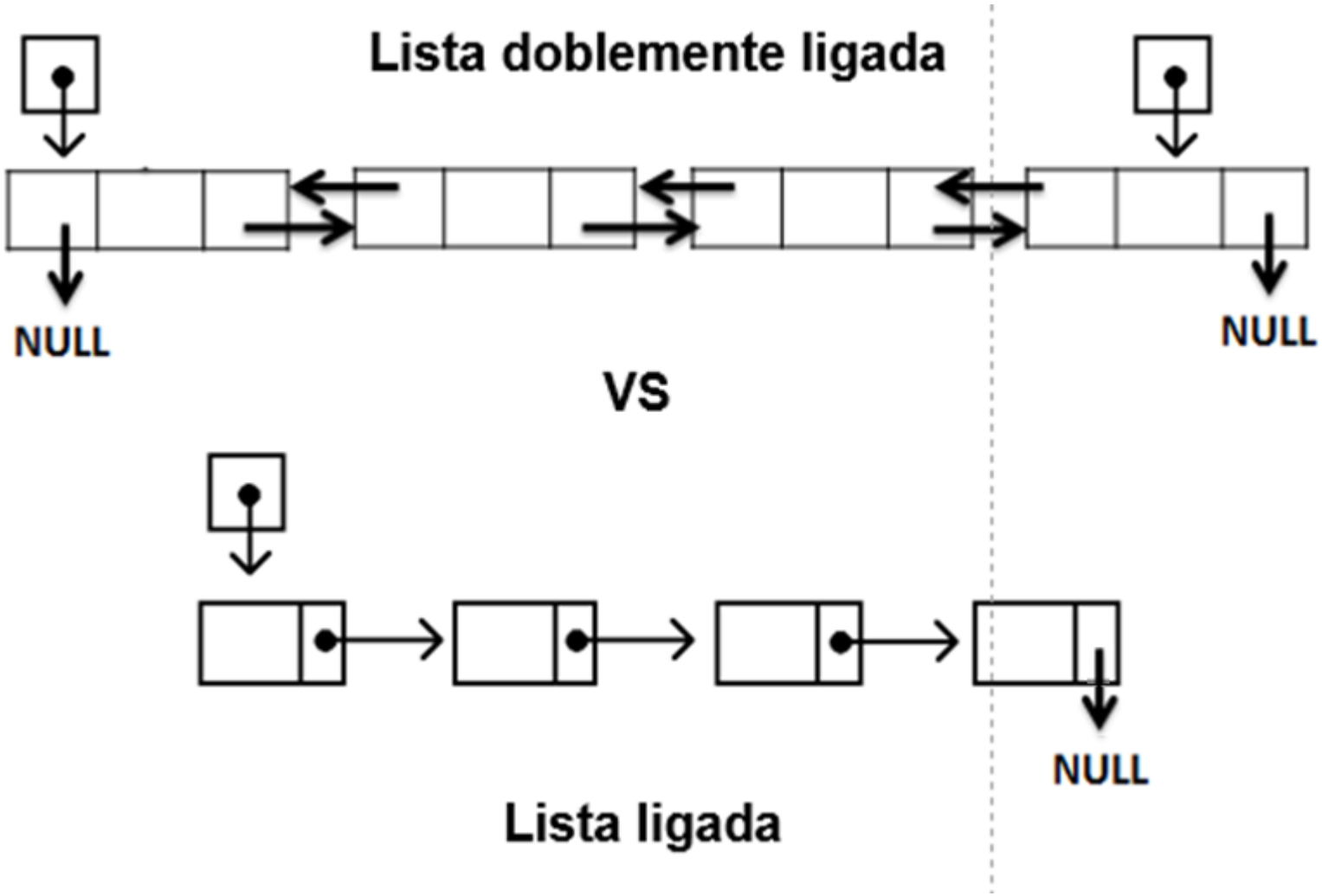
Lista doblemente ligada

Descripción grafica



Tecnológico de Monterrey

Lista doblemente ligada

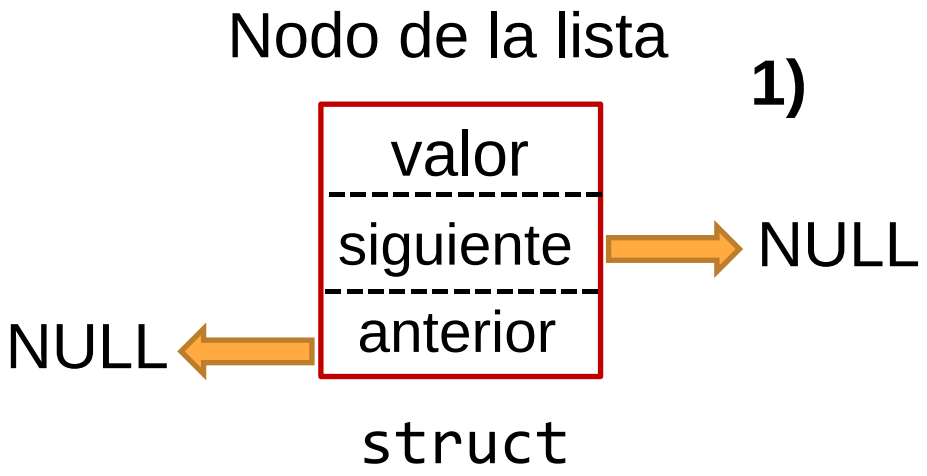


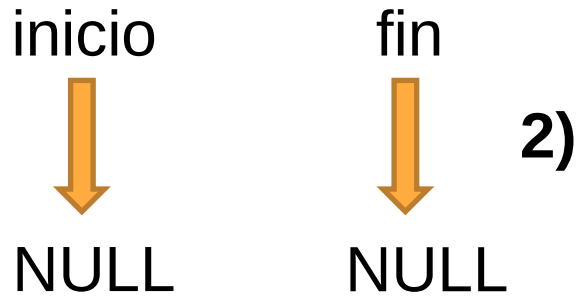
Tecnológico de Monterrey

Implementación de una lista doblemente ligada

Inicialización de la lista

```c
//Libreria de entrada y salida de datos
#include<stdio.h>
//Libreria para el manejo de memoria
#include <stdlib.h>

// Definicion de la estructura de un nodo en la lista
struct nodo {
   int valor;
   struct nodo *siguiente;
   struct nodo *anterior;
};

//Elemento que apunta a la parte inicial de la lista
struct nodo *inicio = NULL;
//Elemento que apunta a la parte final de la lista
struct nodo *fin = NULL;
```



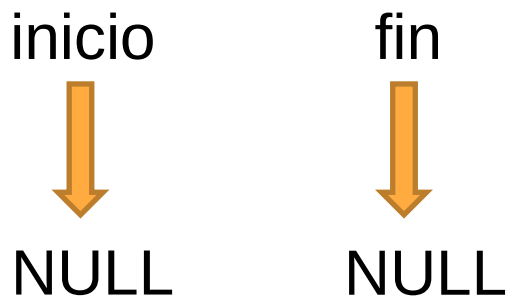


Tecnológico de Monterrey

Implementación de una lista doblemente ligada

Creación de funciones misceláneas (1)

```
//Verifica si la lista esta vacia
int listaVacia()
{
  if (inicio == NULL)
     return 1;
  else
    return 0;
}
```



Tecnológico de Monterrey

Implementación de una lista doblemente ligada

Creación de funciones misceláneas (2)

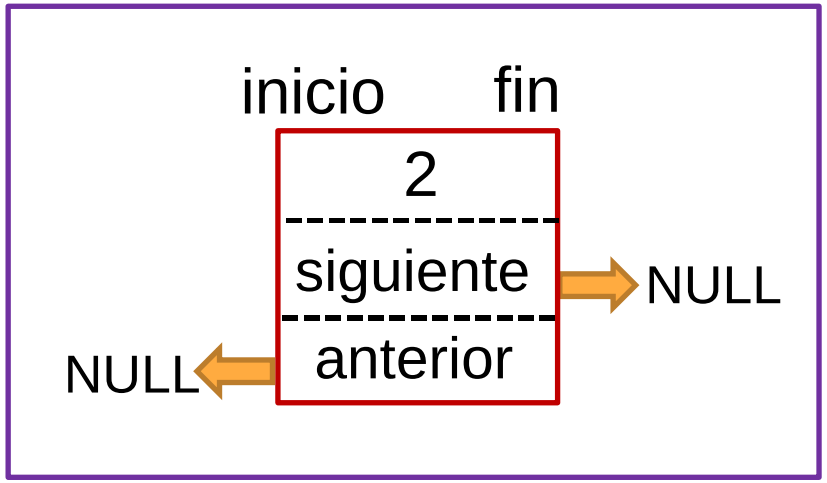
```
//Devuelve la longitud de una lista
int longitudLista() {
    struct nodo *temporal;
    int longitud;
    temporal=inicio;
    while(temporal!=NULL) {
        longitud++;
        temporal=temporal->siguiente;
    }
    return longitud;
}
```

Tecnológico de Monterrey

Implementación de una lista doblemente ligada

Añadir datos al final de la lista (1)

```
//Inserta un elemento al final de una lista
void insertaFinalLista(int numero) {
  printf("Nuevo elemento en el final de la lista: %d \n",numero);
  struct nodo *elemento = malloc(sizeof(struct nodo));
  if (elemento == NULL) {
    printf("No se puede crear un elemento en la lista");
    return;
  }
  if(listaVacia()==1){
    elemento->valor=numero;
    elemento->anterior=NULL;
    elemento->siguiente=NULL;
    inicio=elemento;
    fin=elemento;
  }
```

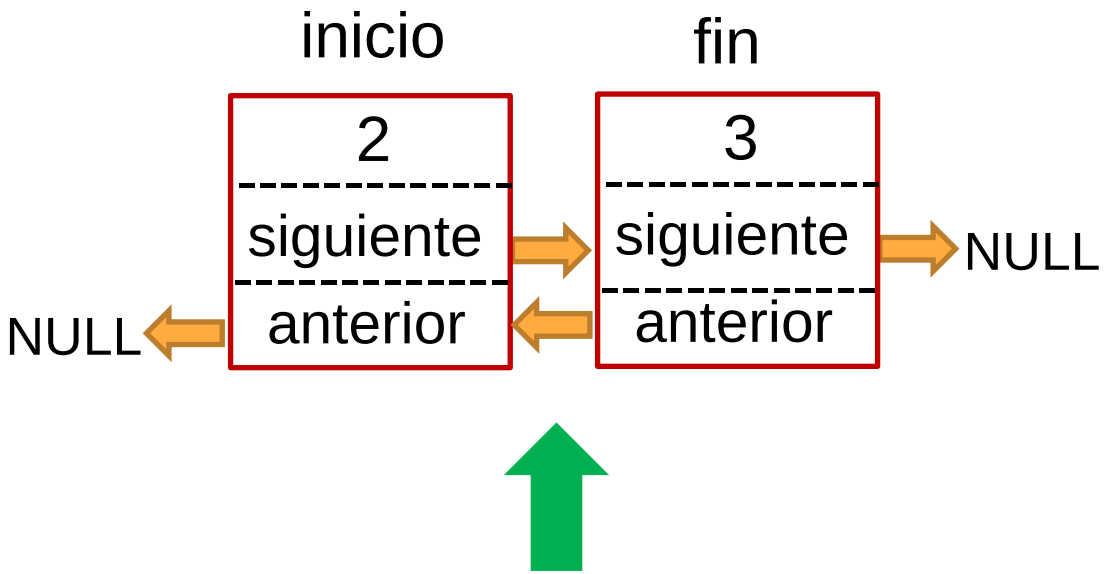


Tecnológico de Monterrey

Implementación de una lista doblemente ligada

Añadir datos al final de la lista (2)

```
else{
    struct nodo *indiceActual=inicio;
    struct nodo *indiceAnterior=NULL;
    while(indiceActual!=NULL){
        indiceAnterior=indiceActual;
        indiceActual=indiceActual->siguiente;
    }
    elemento->valor=numero;
    elemento->siguiente=NULL;
    elemento->anterior=indiceAnterior;
    indiceAnterior->siguiente=elemento;
    fin=elemento;
}

}
```



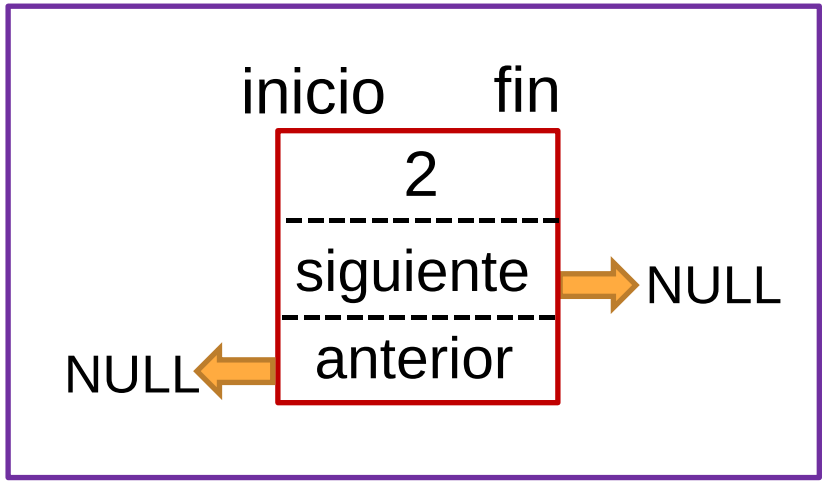
Estilo cola

Tecnológico de Monterrey

Implementación de una lista doblemente ligada

Añadir datos al inicio de la lista (1)

```
//Inserta un elemento al inicio de una lista
void insertaInicioLista(int numero){
  printf("Nuevo elemento en el inicio de la lista: %d \n",numero);
  struct nodo *elemento = malloc(sizeof(struct nodo));
  if (elemento == NULL) {
    printf("No se puede crear un elemento en la lista");
    return;
  }
  if(listaVacia()==1){
    elemento->valor=numero;
    elemento->anterior=NULL;
    elemento->siguiente=NULL;
    inicio=elemento;
    fin=elemento;
  }
```

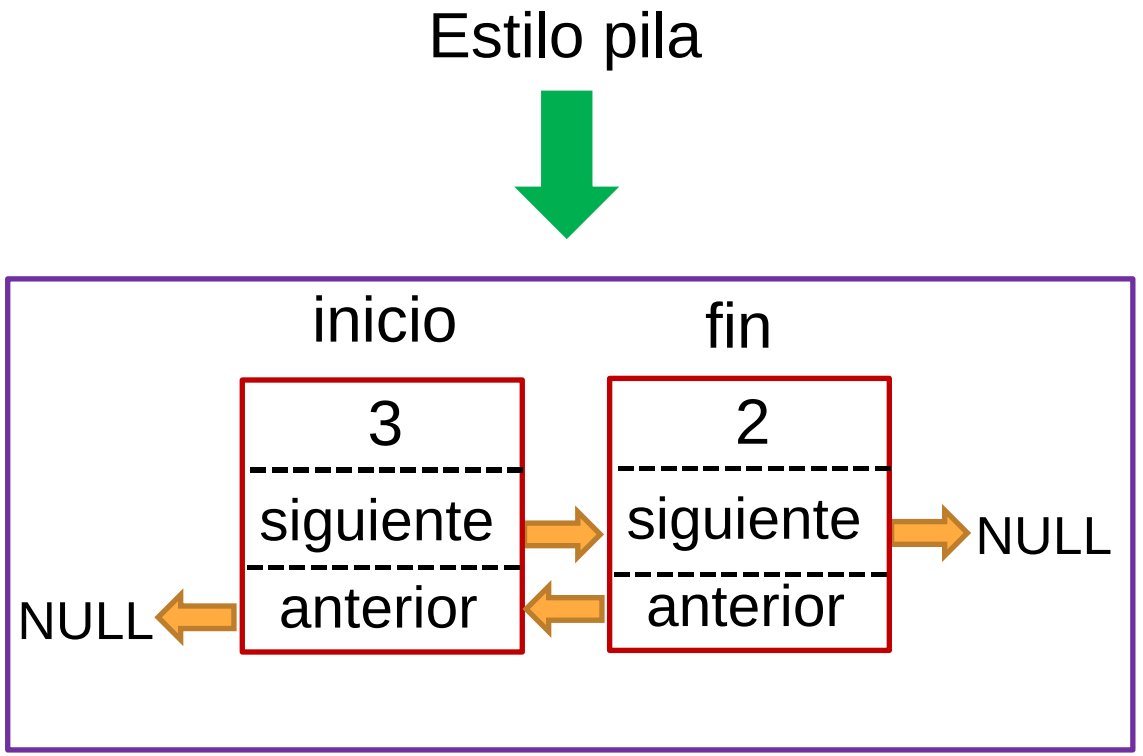


Tecnológico de Monterrey

Implementación de una lista doblemente ligada

Añadir datos al inicio de la lista (1)

```
    else{
        inicio->anterior=elemento;
        elemento->valor=numero;
        elemento->siguiente=inicio;
        elemento->anterior=NULL;
        inicio = elemento;
    }
}
```



Tecnológico de Monterrey

Implementación de una lista doblemente ligada

Eliminar datos de lista (1)

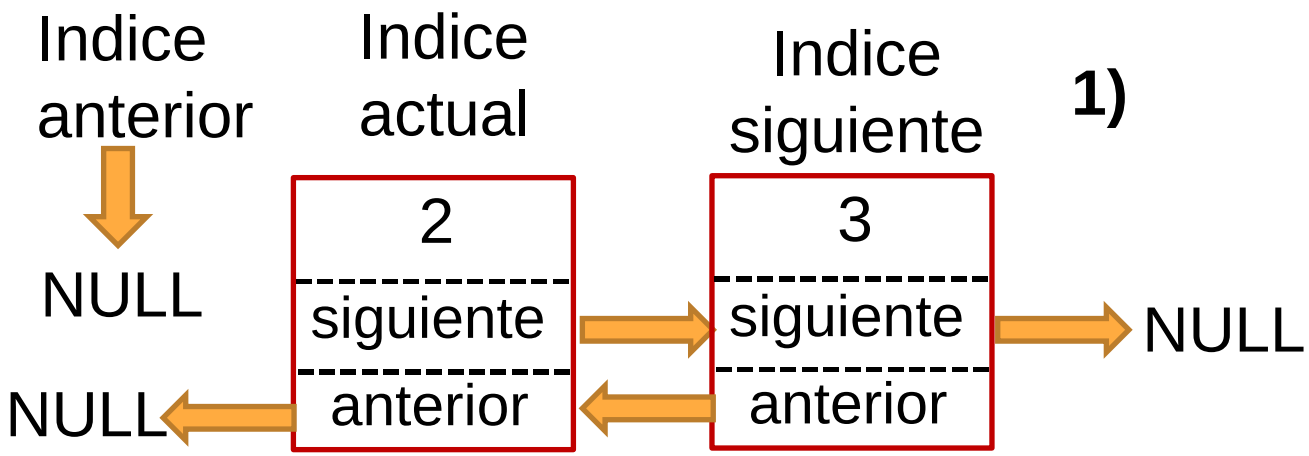
```
//Eliminar un elemento de la lista
void eliminaLista(int numero){
    if (listaVacia()==1){
        printf("La lista esta vacia\n");
        return;
    }
    struct nodo *indiceAnterior=NULL;
    struct nodo *indiceActual=inicio;
    struct nodo *indiceSiguiente=inicio->siguiente;
    struct nodo *temporal;
```

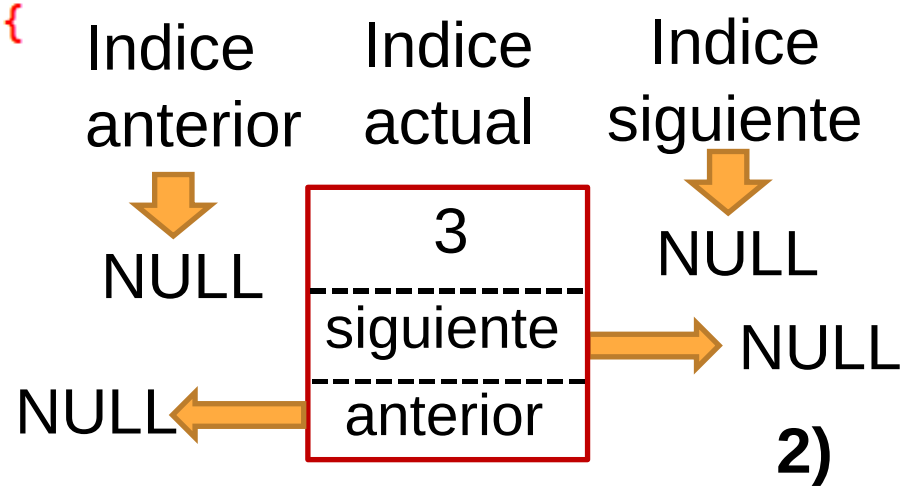
Tecnológico de Monterrey

Implementación de una lista doblemente ligada

Eliminar datos de lista (2)

```
while(indiceActual!=NULL)
{
    if(indiceActual->valor==numero){
        // primer elemento en la lista
        if(indiceAnterior==NULL && indiceSiguiente==NULL){
            temporal = indiceActual;
            inicio = indiceActual->siguiente;
            inicio->anterior=NULL;
            printf("Eliminando: %d\n",temporal->valor);
            free(temporal);
            return;
        }
```





Implementación de una lista doblemente ligada

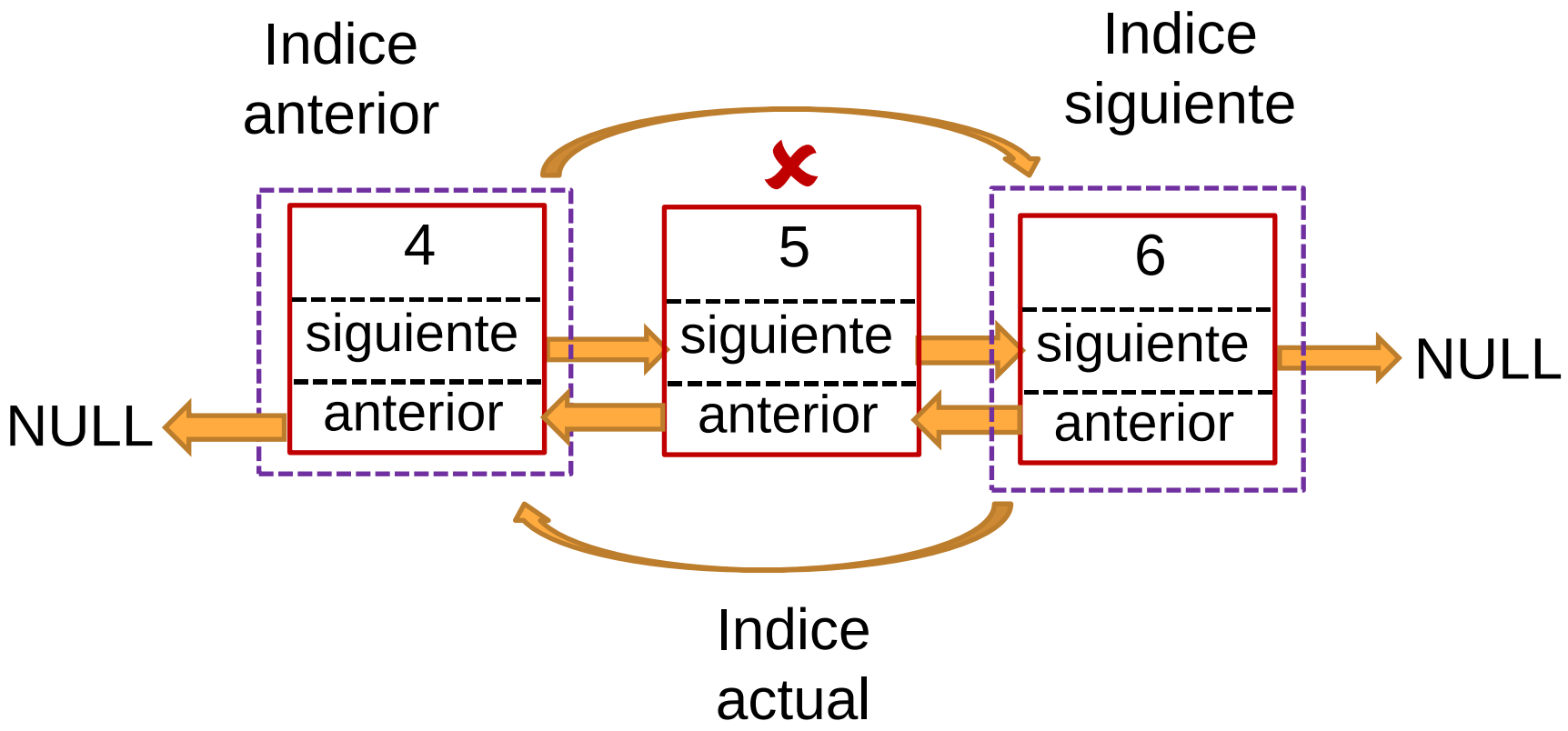
Eliminar datos de lista (3)

```
            //elemento a eliminar va en medio o al final
            else{
                temporal = indiceActual;
                printf("Eliminando: %d\n",temporal->valor);
                if (indiceActual==fin){
                    indiceAnterior->siguiente=NULL;
                    fin=indiceAnterior;
                }
                else{
                    indiceAnterior->siguiente=indiceSiguiente;
                    indiceSiguiente->anterior=indiceAnterior;
                }

                free(temporal);
                return;
            }
        }
        indiceAnterior=indiceActual;
        indiceActual=indiceActual->siguiente;
        indiceSiguiente=indiceSiguiente->siguiente;
    }
    printf("Elemento: %d no se puede eliminar\n", numero);
}
```

Tecnológico de Monterrey

Implementación de una lista doblemente ligada

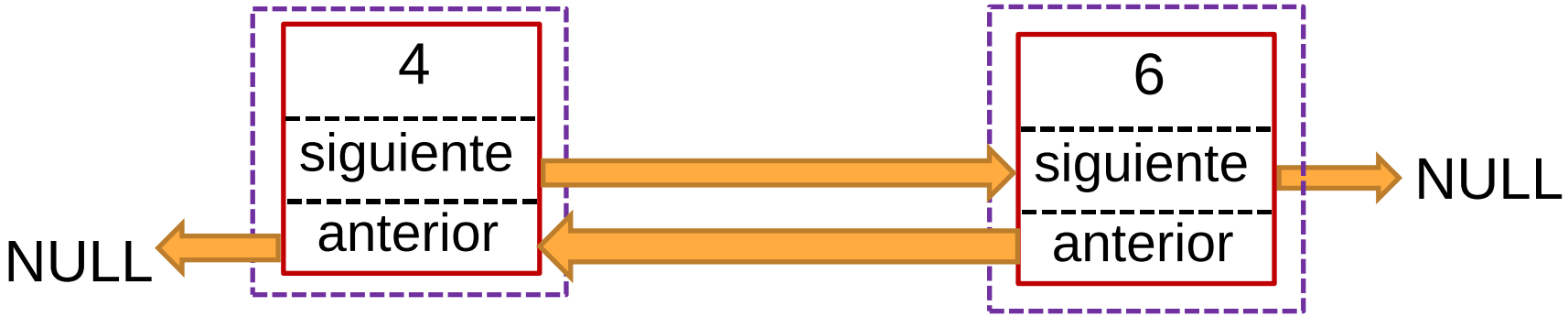
Eliminar datos de lista (4)



Tecnológico de Monterrey

Implementación de una lista doblemente ligada

Eliminar datos de lista (4)



Tecnológico de Monterrey

Implementación de una lista doblemente ligada

Mostrar elementos de la lista desde el inicio

```
//Imprime todos los elementos de una lista desde el inicio
void mostrarElementosListaInicio()
{
    int i=0;
    struct nodo *temporal;
    temporal = inicio;
    printf("Elementos en la lista:\n");
    while (temporal != NULL)
    {
      printf("Elemento[%d]=%d\n",i,temporal->valor);
      temporal = temporal->siguiente;
      i++;
    }
}
```

Tecnológico de Monterrey

Implementación de una lista doblemente ligada

Mostrar elementos de la lista desde el fin

```
//Imprime todos los elementos de una lista desde el fin
void mostrarElementosListaFin()
{
    int i=longitudLista();
    struct nodo *temporal;
    temporal = fin;
    printf("Elementos en la lista:\n");
    while (temporal != NULL)
    {
      printf("Elemento[%d]=%d\n",i,temporal->valor);
      temporal = temporal->anterior;
      i--;
    }
}
```

Tecnológico de Monterrey

Implementación de una lista doblemente ligada

```
void main() {
    printf("Uso de una lista doblemente ligada\n\n");
    printf("Operaciones \"insercion al final\"\n");
    insertaFinalLista(4);
    insertaFinalLista(5);
    insertaFinalLista(6);
    printf("\n");
    printf("Operaciones \"insercion al inicio\"\n");
    insertaInicioLista(1);
    insertaInicioLista(2);
    insertaInicioLista(3);
    printf("\n");
    printf("Operaciones \"eliminar\"\n");
    eliminaLista(2);
    eliminaLista(4);
    eliminaLista(5);
    printf("\n");
    printf("Operacion \"mostrar desde inicio\"\n");
    mostrarElementosListaInicio();
    printf("\n");
    printf("Operacion \"mostrar desde final\"\n");
    mostrarElementosListaFin();
    printf("\n");
}
```

Función principal

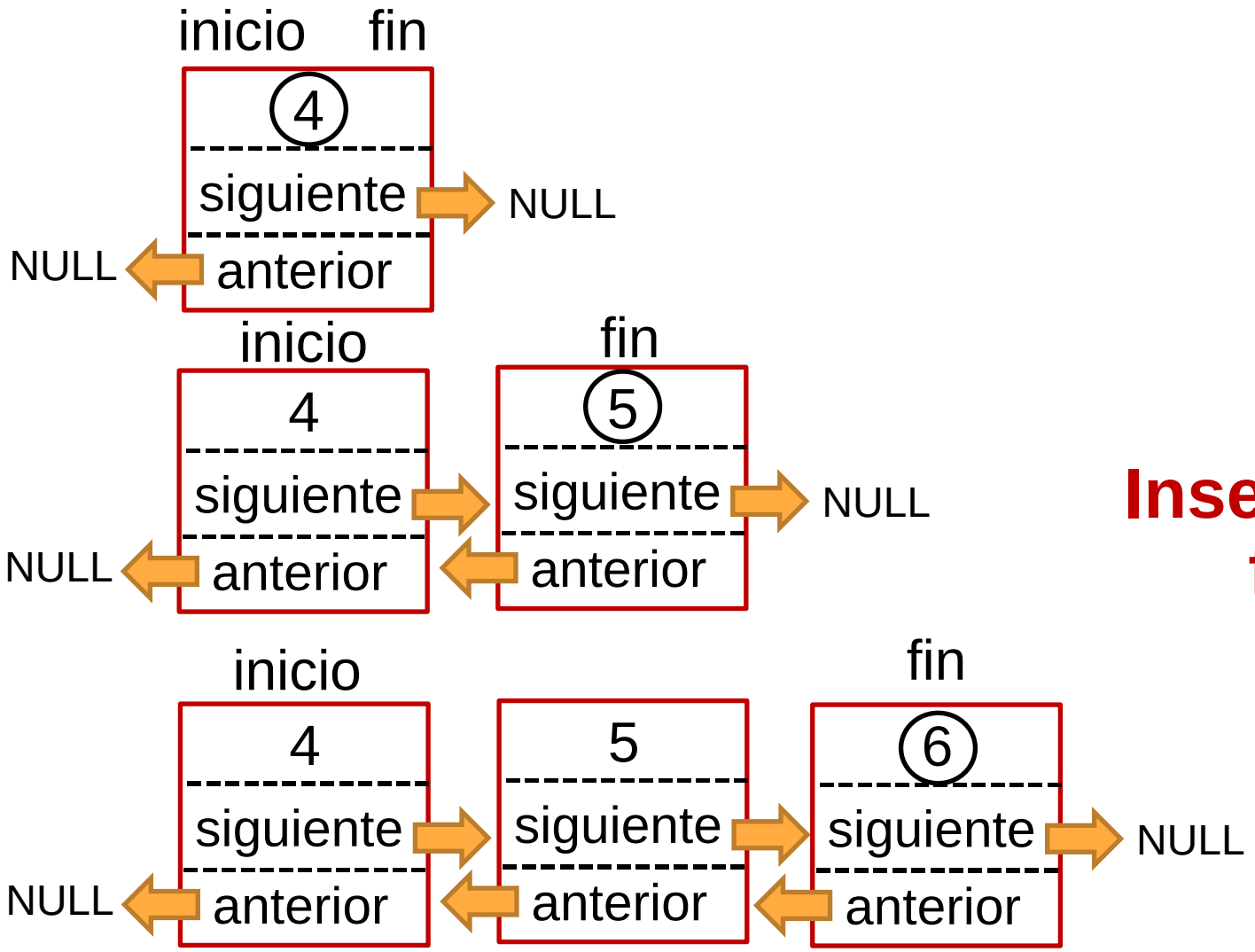
Tecnológico de Monterrey

Implementación de una lista doblemente ligada

```
Uso de una lista doblemente ligada

Operaciones "insercion al final"
Nuevo elemento en el final de la lista: 4
Nuevo elemento en el final de la lista: 5
Nuevo elemento en el final de la lista: 6

Operaciones "insercion al inicio"
Nuevo elemento en el inicio de la lista: 1
Nuevo elemento en el inicio de la lista: 2
Nuevo elemento en el inicio de la lista: 3

Operaciones "eliminar"
Eliminando: 2
Eliminando: 4
Eliminando: 5

Operacion "mostrar desde inicio"
Elementos en la lista:
Elemento[0]=3
Elemento[1]=1
Elemento[2]=6

Operacion "mostrar desde final"
Elementos en la lista:
Elemento[3]=6
Elemento[2]=1
Elemento[1]=3
```

Ejecución del programa

Implementación de una lista doblemente ligada

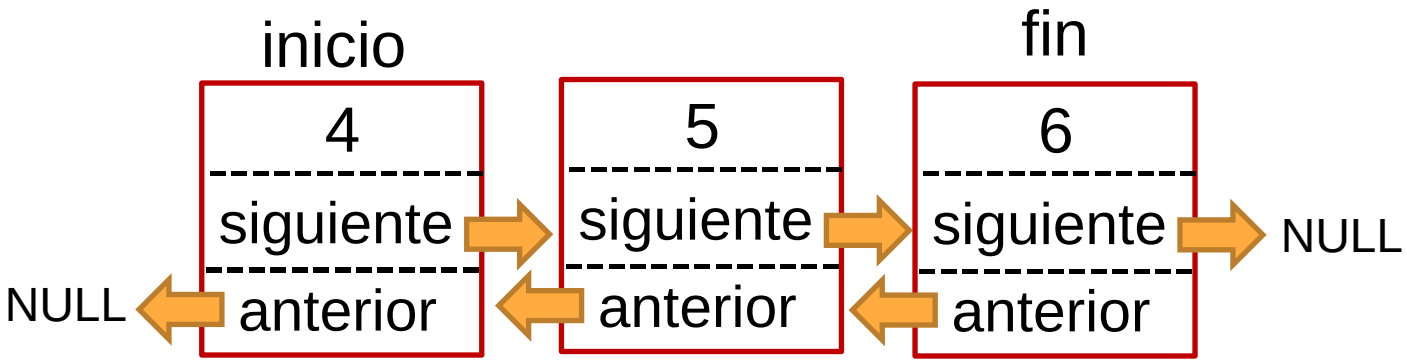


Ejecución del programa

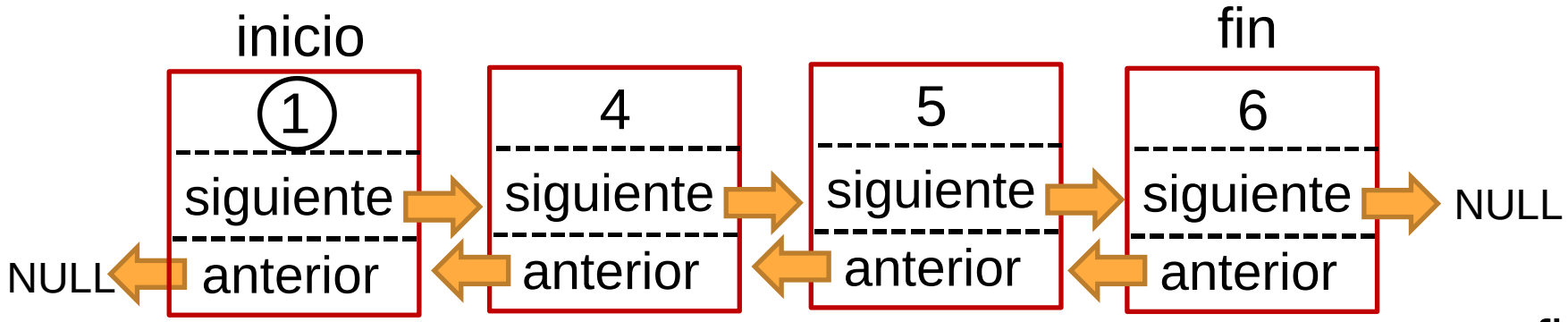
Inserción al final

Tecnológico de Monterrey

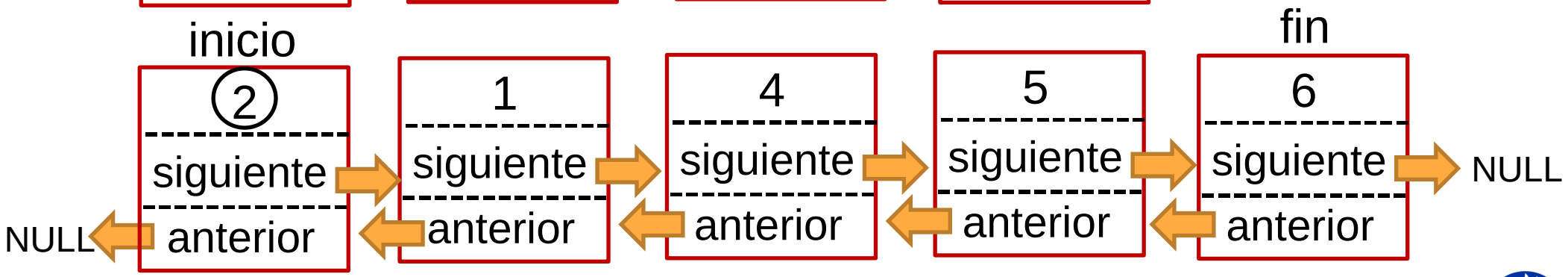
Implementación de una lista doblemente ligada



Ejecución del programa

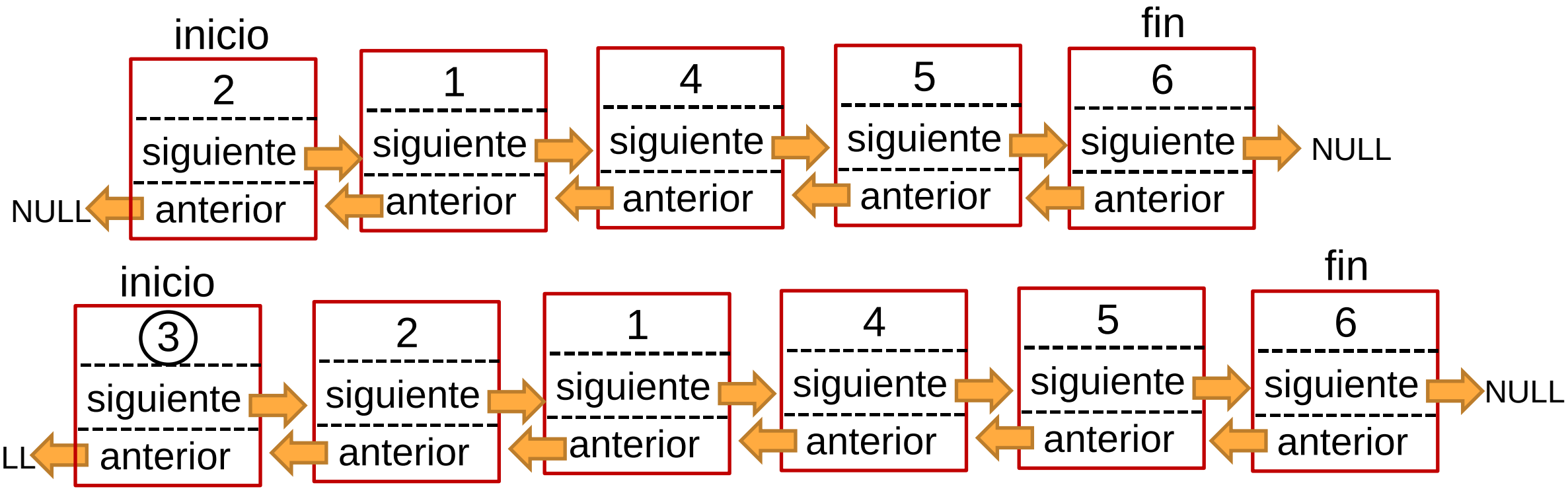


Inserción al inicio



Tecnológico de Monterrey

Implementación de una lista doblemente ligada

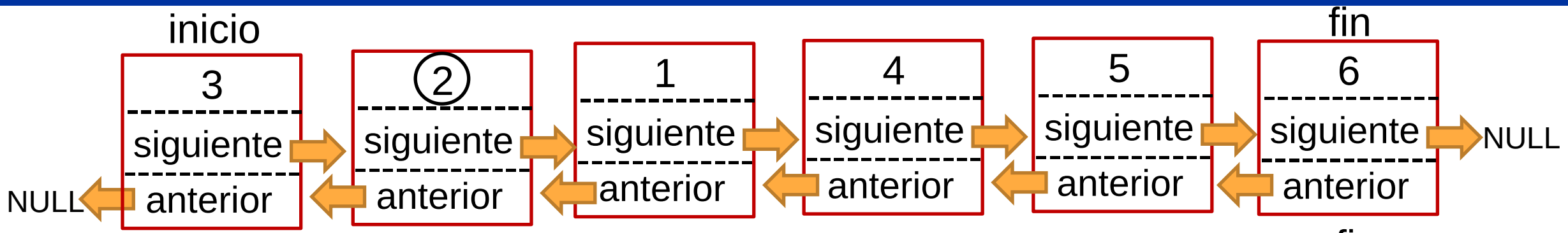


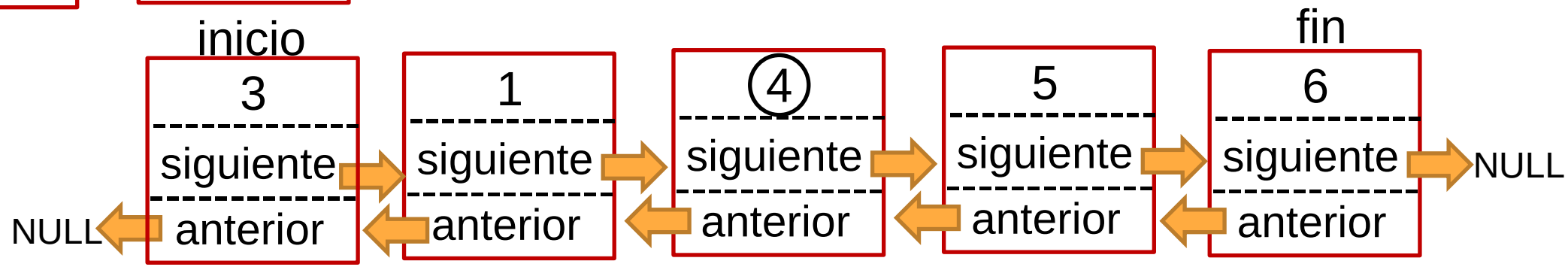
Inserción al inicio

Ejecución del programa

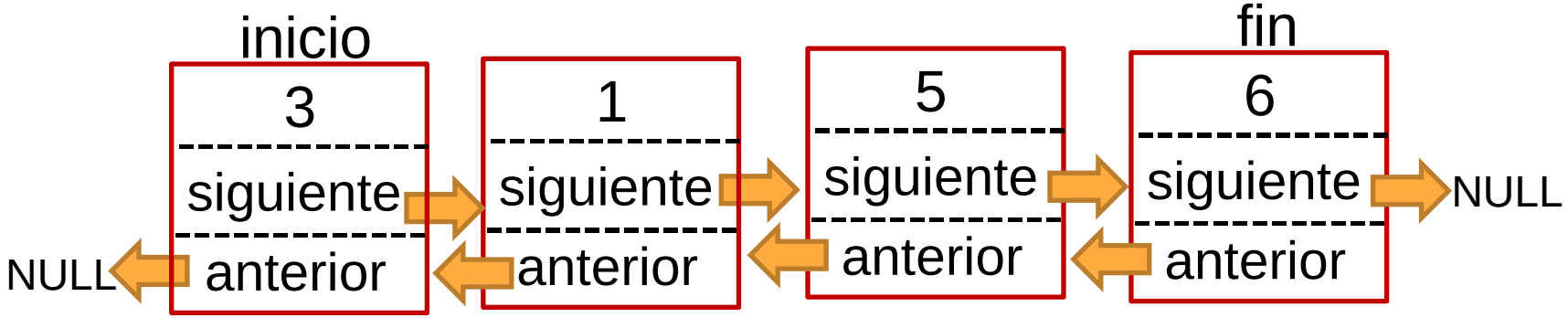
Tecnológico de Monterrey

Implementación de una lista doblemente ligada



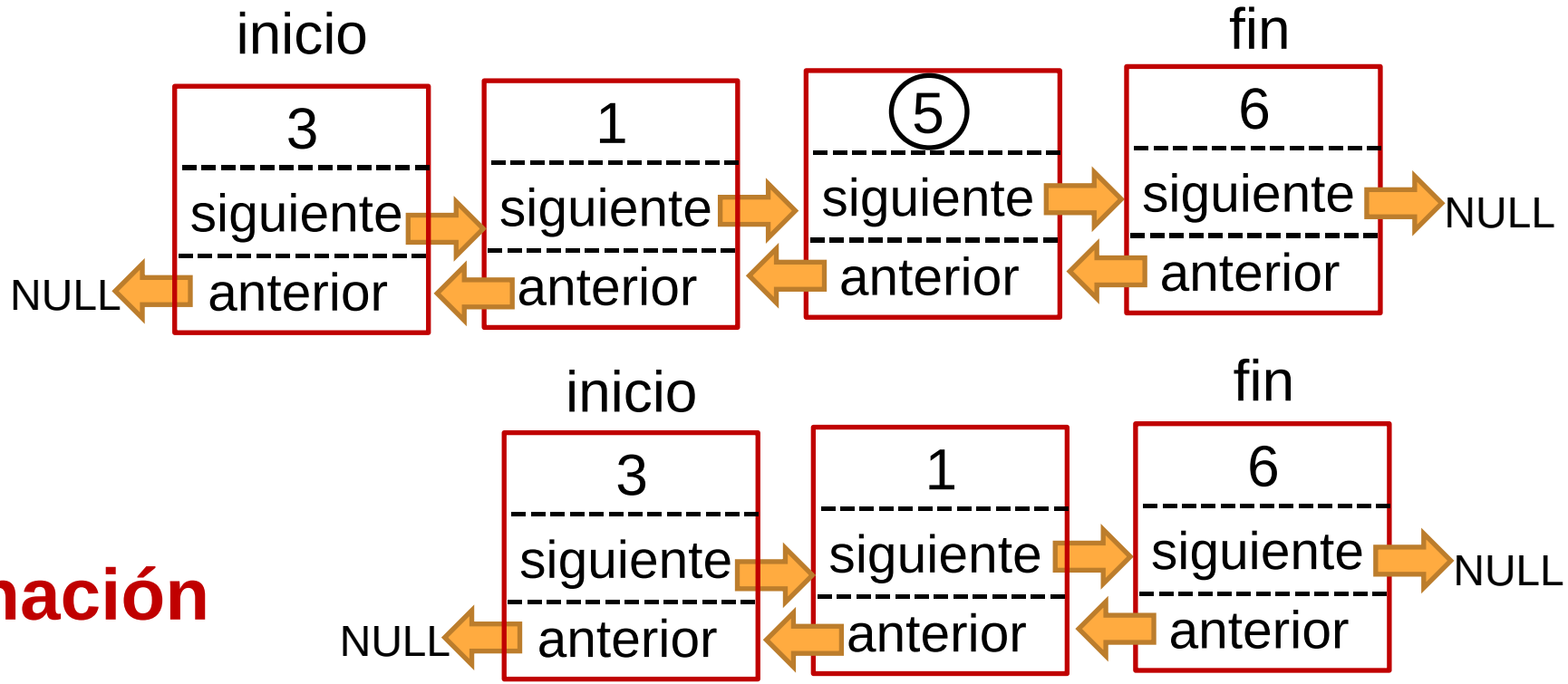


Eliminación



Ejecución del programa

Implementación de una lista doblemente ligada



Eliminación

Ejecución del programa

Tecnológico de Monterrey

Gracias!

Preguntas...

Tecnológico de Monterrey