FACULTA DE INGENIERÍA Y ARQUITECTURA
CARRERA DE INGENIERÍA DE SISTEMAS

Listas Enlazadas

UNIVERSIDAD DE LIMA

UNIVERSIDAD DE LIMA

FUNDAMENTO TEÓRICO

Si en tiempo de diseño del algoritmo:

Se conoce la cantidad de elementos a listar.

- Podemos usar Organización Estática de Memoria: Arreglos.
  - Obliga en tiempo de compilación a reservar la memoria a usar.
  - Implementaciones más rápidas: pues ya se gestionó la memoria.

Si no se conoce la cantidad de elementos a listar

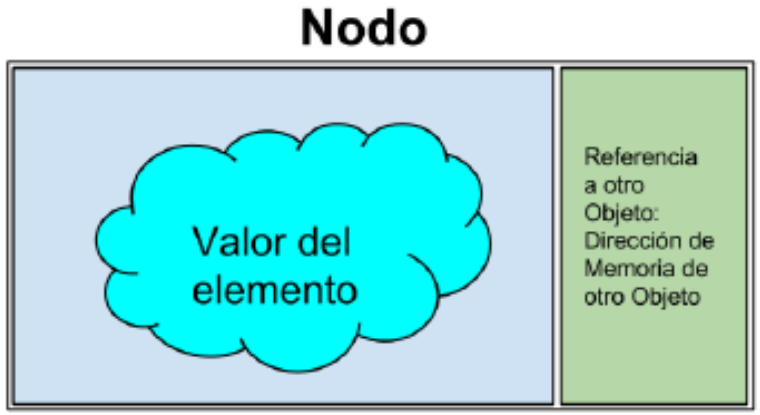
- Podemos usar Organización Dinámica: Listas Enlazadas.
  - Colección de nodos lógicamente uno detrás de otro.
  - Cada nodo se compone de al menos 2 campos:
    - La información propiamente dicha
    - Una referencia al siguiente elemento de la lista.

UNIVERSIDAD DE LIMA

Representación en Memoria

Una Lista Enlazada es una secuencia finita de elementos llamados nodos, donde cada nodo contiene datos y una referencia (puntero) al siguiente nodo de la lista.

Estructura de un Nodo

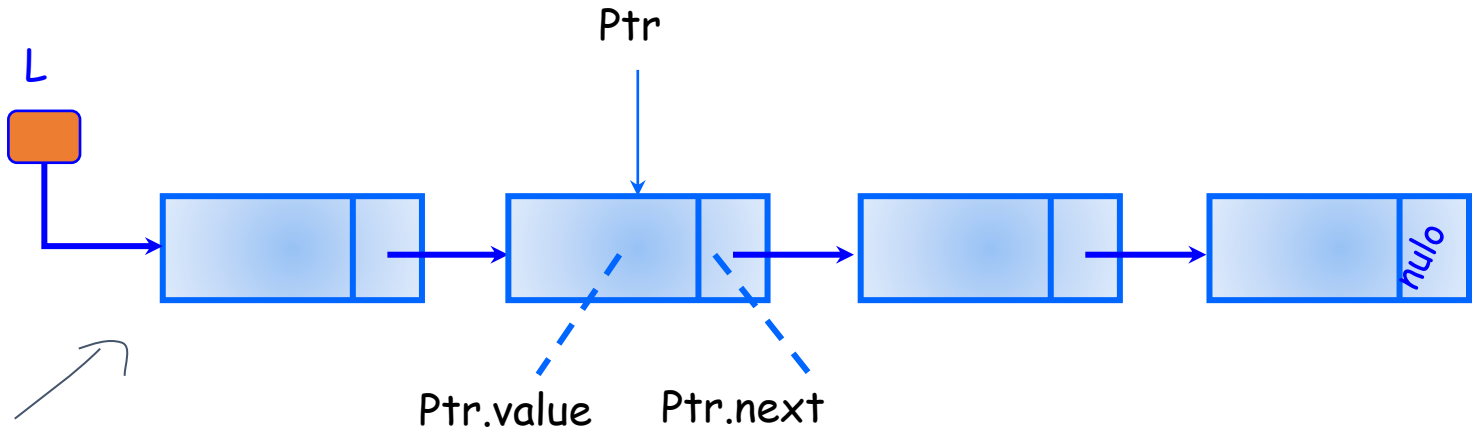


UNIVERSIDAD DE LIMA

Representación en Memoria

Notación:

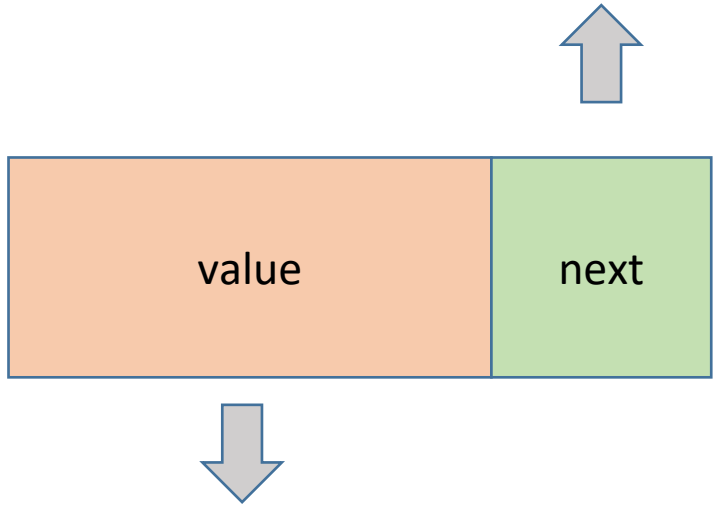
- L : Referencia al primer nodo de la lista.
- Ptr : Referencia a un nodo cualquiera de la lista (puntero)
- Ptr.value : Información del nodo referenciado por Ptr.
- Ptr.next : Referencia a la siguiente nodo de referenciado por Ptr.



DEFINICIÓN DE NODO

Referencia al siguiente nodo:

- En C / C++: punteros
- En Java: atributo de clase - tipo Nodo



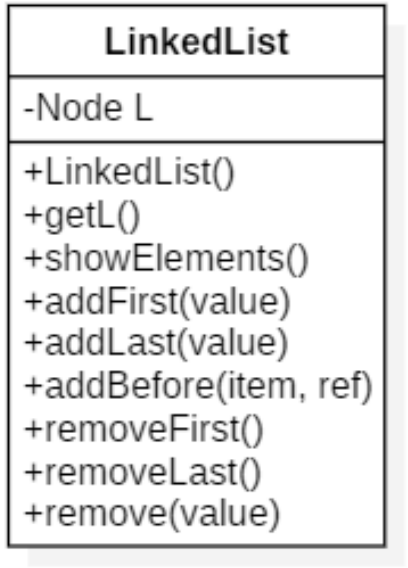
Dato que almacena el nodo:

- int
- float
- String
- Objeto (definido por el desarrollador)

```
public class Node {
    private int value;
    private Node next;

    public Node(int value){
        this.value = value;
    }
    public int value(){
        return value;
    }

    public Node next(){
        return next;
    }

    public void setNext(Node node){
        this.next = node;
    }
}
```

UNIVERSIDAD DE LIMA

DEFINICIÓN DE LISTA ENLAZADA




```
//Constructor
public LinkedList(){
        this.L = null;
}
```

Lista enlazada vacía: Valor de L es nulo

UNIVERSIDAD DE LIMA

OPERACIONES

1. Recorrido

Especificaciones:

- Objetivo : Procesa cada nodo de lista una vez.
- Entrada : Ninguna
- Precondiciones : Ninguna
- Salida : (Según proceso)
- Postcondiciones : (Según proceso)
- Pasos:
  - Obtener una referencia al primer nodo de la lista (L), almacenarlo en una variable auxiliar (ptr).
  - Mientras exista un nodo (es decir, ptr != nulo):
    - Procesar el nodo actual **ptr** (depende del proceso, puede ser imprimirlo en pantalla, incrementar un contador, agregarlo a una suma, etc).
    - Desplazarse al siguiente nodo: ptr = ptr.next()

UNIVERSIDAD DE LIMA

EJEMPLO: MOSTRAR ELEMENTOS

```
public void showElements(){
	//Obtener referencia al primer nodo (variable ptr)
	Node ptr = L;
	//Mientras existan mas nodos por procesar
	while(ptr != null){
		//Procesamos la informacion del nodo actual: ptr.value()
		System.out.print(ptr.value() + " --> ");
		ptr = ptr.next();
	}
	//El ultimo nodo tendra null como siguiente
	System.out.println("null");
}
```

UNIVERSIDAD DE LIMA

OPERACIONES

2. Insertar al Inicio

Especificaciones:

- Objetivo : Insertar un nuevo nodo al comienzo de la Lista
- Entrada : **valor** a insertar
- Precondiciones : Ninguna
- Salida : Ninguna
- Postcondiciones : Lista contiene al elemento **valor** en la primera posición.
- Pasos:
  - Crear un nuevo nodo con el valor a insertar: **newNode**.
  - Si la lista se encuentra vacía (L == null), la lista resultante tendrá un solo elemento. Por tanto, el sucesor de **newNode** debe ser null: newNode.setNext(null)
  - Si la lista enlazada no se encuentra vacía, **newNode** deberá aparecer primero. Por lo tanto, el sucesor de **newNode** debe ser el primer nodo de la lista original (L) newNode.setNext(L).
  - El nuevo nodo inicial (L) de la lista ahora será **newNode :** L = newNode.

UNIVERSIDAD DE LIMA

INSERTAR AL INICIO

```
public void addFirst(int value){
        //Creacion de nodo
        Node newNode = new Node(value);
        //Si la lista esta vacia
        if(L == null){
                //Unico elemento, no tiene sucesor
                newNode.setNext(null);
        //Si no esta vacia
        }else{
                //El sucesor de newNode debe ser el primero de la lista original
                newNode.setNext(L);
        }
        //El nodo inicial (L) de la lista enlazada ahora debe ser nuevoNodo
        L = newNode;
}
```

UNIVERSIDAD DE LIMA

OPERACIONES

3. Insertar al Final

Especificaciones:

- Objetivo : Insertar un nuevo nodo al final de la Lista
- Entrada : **valor** a insertar
- Precondiciones : Ninguna
- Salida : Ninguna
- Postcondiciones : Lista contiene al elemento **valor** en la última posición.
- Pasos:
  - Crear un nuevo nodo con el valor a insertar: **newNode**.
  - Si la lista se encuentra vacía, insertar el elemento al inicio: addFirst(newNode).
  - Si la lista no está vacía, recorrerla hasta llegar al último elemento. Es decir, detenernos cuando el siguiente del nodo actual (ptr) sea null.
    - Mientras ptr.next() no sea null
      - Desplazarse al siguiente nodo: ptr = ptr.next()
  - Una vez que la variable **ptr** apunte al último elemento, actualizamos su nodo siguiente. El sucesor debe ser el nuevo nodo creado.
    - ptr.setNext(newNode)

UNIVERSIDAD DE LIMA

INSERTAR AL FINAL

```
public void addLast(int value){
        //Crear nuevo nodo
        Node newNode = new Node(value);
        //Si la lista esta vacia, insertar al inicio
        if(L == null){
                addFirst(value);
        }else{
                //Si no esta vacia, recorremos hasta encontrar el ultimo nodo
                Node ptr = L;
                //Ultimo nodo: aquel que tiene como siguiente (next) el valor null
                while(ptr.next() != null){
                        ptr = ptr.next();
                }
                ptr.setNext(newNode);
                //El siguiente del nodo creado debe ser nulo (es el ultimo)
                newNode.setNext(null);
        }
}
```

UNIVERSIDAD DE LIMA

OPERACIONES

4. Insertar antes de un nodo

Especificaciones:

- Objetivo : Insertar **item** en la posición anterior a otro nodo con información **ref**
- Entrada : **ref, item**
- Precondiciones : Valor **ref** pertenece a la lista enlazada
- Salida : Ninguna
- Postcondiciones : Lista L contiene a **item**, en la posición previa al nodo **ref.**
- Pasos:
  - Crear un nuevo nodo con el valor a insertar: **newNode**.
  - Si la lista se encuentra vacía, insertar el elemento al inicio: addFirst(newNode).
  - Si la lista no está vacía, recorrerla hasta llegar al último elemento. Es decir, detenernos cuando el valor siguiente de nuestro nodo de referencia (**ptr**) sea null. Almacenamos una referencia al predecesor en la variable **prev.**
  - Si existe el elemento con valor **ref** (ptr no es null)**:**
    - Si el nodo no tiene predecesor (prev = null), insertar el valor al inicio.
    - En caso contrario, actualizar el nodo predecesor (**prev**) para que apunte a **newNode**. Finalmente, actualizar **newNode** para que apunte nodo de valor **ref (ptr)**.

UNIVERSIDAD DE LIMA

INSERTAR ANTES

```
public void addBefore(int reg, int ref){
        Node newNode = new Node(item);
        //ptr: nodo referencia - prev: nodo anterior a "ptr"
        Node ptr = L, prev = null;
        while(ptr != null && ptr.value() != ref){//Se detiene al encontrar el elemento "ref"
                prev = ptr; //Antes de "avanzar" dejo una referencia al elemento anterior
                ptr = ptr.next();
        } //Luego del bucle, "ptr" apunta al nodo de valor "ref" (si existe)

        if(ptr != null){//Solo si existe el elemento con valor "ref"
                if(prev == null){//Si el elemento encontrado es el primero
                        addFirst(item); //Agregar elemento al inicio
                }else{
                        //El nodo anterior (prev) tendra como sucesor al nodo que hemos creado
                        prev.setNext(newNode);
                        //El nodo siguiente de nuevoNodo debe ser el de valor "ref" (ptr)
                        newNode.setNext(ptr);
                }
        }
}
```

UNIVERSIDAD DE LIMA

OPERACIONES

5. Eliminar Nodo Inicial

Especificaciones:

- Objetivo : Eliminar el primer nodo de la lista
- Entrada : Ninguna
- Precondiciones : Lista L no vacía.
- Salida : Ninguna
- Postcondiciones : Lista L con un elementos menos (elemento inicial eliminado)
- Pasos:
  - Si la lista no está vacía (L != null)
    - El nuevo primer elemento (L) debe ser el segundo elemento de la lista original (L.next). Es decir: L = L.next().

```
public void removeFirst(){
        if(L != null){//Si la lista no esta vacia
                L = L.next(); // L.next es el nuevo nodo inicial
        }
}
```

UNIVERSIDAD DE LIMA

OPERACIONES

6. Eliminar Nodo Final

Especificaciones:

- Objetivo : Eliminar el ultimo nodo de la lista
- Entrada : Ninguna
- Precondiciones : Lista L no vacía.
- Salida : Ninguna
- Postcondiciones : Lista L con un elementos menos (elemento final eliminado)
- Pasos:
  - Recorrer la lista hasta encontrar el penúltimo elemento. Para ello procedemos de manera similar a la operación insertarAntes, con 2 referencias. Luego del bucle de búsqueda, **ptr** referenciará al último elemento y **prev** referenciará al penúltimo elemento.
  - Si **prev** es nulo: se elimina el nodo inicial.
  - En caso contrario, asignar null como el nodo siguiente de **prev.**

UNIVERSIDAD DE LIMA

ELIMINAR NODO FINAL

```
public void removeLast(){
        Node ptr = L;
        Node prev = null;
        if(ptr != null){ //Lista no vacia
                while(ptr.next() != null){ //para encontrar el ultimo elemento
                        prev = ptr;
                        ptr = ptr.next();
                }// Al salir del bucle: ptr - ultimo nodo, prev - penultimo nodo
                if(prev == null){//Si la lista tiene solo un elemento
                        removeFirst();
                }else{
                        //Penultimo nodo ahora sera el ultimo
                        prev.setNext(null);
                }
        }
}
```

UNIVERSIDAD DE LIMA

OPERACIONES

7. Eliminar Nodo

Especificaciones:

- Objetivo : Eliminar el nodo con un valor específico
- Entrada : **ref**, dato a eliminar
- Precondiciones : Lista L no vacía.
- Salida : Ninguna
- Postcondiciones : Lista L con un elementos menos (elemento con valor **ref** eliminado).
- Pasos:
  - Si el valor ref, se encuentra en el primer nodo (L): puede llamarse a la operación correspondiente.
  - En caso contrario, buscar el valor **ref** en la lista enlazada. Almacenar el nodo asociado en **ptr** y su predecesor en **prev.**
  - Asignar como sucesor de **prev** al nodo siguiente de **ptr.**

UNIVERSIDAD DE LIMA

ELIMINAR NODO

```
public void remove(int ref){
        if(L != null){ //Lista no vacia
                if(L.value() == ref){ //Si es el primer nodo
                        removeFirst();
                }else{
                        Node ptr = L.next();
                        Node prev = L;
                        //Buscar el nodo con valor ref (ptr) y su predecesor (anterior)
                        while(ptr != null && ptr.value() != ref){
                                prev = ptr;
                                ptr = ptr.next();
                        }
                        if(ptr != null){ //Si se encontro el elemento
                                prev.setNext(ptr.next());
                        }
                }
        }
}
```