# Universidad Nacional Autónoma de México

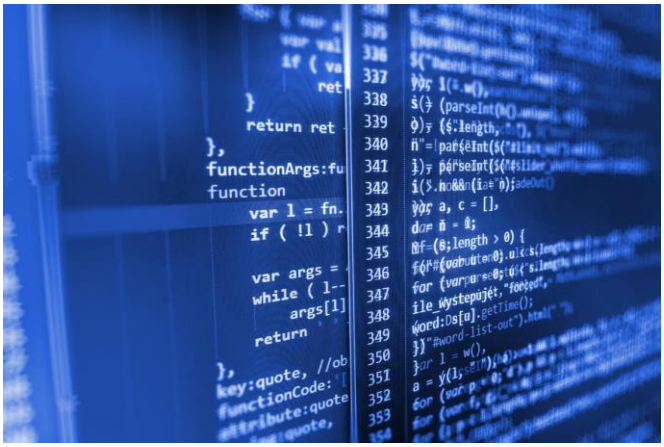
## Facultad de Estudios Superiores Aragón

**Ingeniería en Computación**

**Estructura de Datos**

**Jesús Hernández Cabrera**

**Axel Yahir Moreno Rodríguez**



**Turno Vespertino**

**Grupo 1360**

Main.java | DoubleLinkedList.java | NodoDoble.java

Estructura de Datos > Tareas > Tarea 5 > tarea5 > src > main > java > listas > unam > Main.java > Main > main(String[])

```java
package listas.unam;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        DoubleLinkedList<Integer> numeros= new DoubleLinkedList<>();

        numeros.agregarAlInicio(valor:90);
        numeros.agregarAlInicio(valor:80);
        numeros.agregarAlInicio(valor:70);
        numeros.agregarAlInicio(valor:65);
        numeros.agregarAlInicio(valor:60);
        numeros.agregarAlInicio(valor:50);
        numeros.transversal(direccion:"derecha");

        numeros.eliminar(pos:2);
        numeros.transversal(direccion:"derecha");

        numeros.actualizar(aBuscar:4, valor:88);
        numeros.transversal(direccion:"derecha");

        numeros.buscar(valor:80);

    }
}
```

Main.java | DoubleLinkedList.java | NodoDoble.java

Estructura de Datos > Tareas > Tarea 5 > tarea5 > src > main > java > listas > unam > DoubleLinkedList.java > DoubleLinkedList<T> > estaVacia()

```java
public class DoubleLinkedList<T> {

    public void agregarAlInicio(T valor) {
        NodoDoble<T> nuevo = new NodoDoble<>(valor);
        if (this.estaVacia()) {
            this.head = nuevo;
            this.tail = nuevo;
        } else {
            this.head.setAnterior(nuevo);
            nuevo.setSiguiente(this.head);
            this.head = nuevo;
        }
        this.tamanio++;
    }

    public void agregarAlFinal(T valor){
        NodoDoble<T> nuevo = new NodoDoble<>(valor);
        if (this.estaVacia()) {
            this.head = nuevo;
            this.tail = nuevo;
        } else {
            this.tail.setSiguiente(nuevo);
            nuevo.setAnterior(tail);
            this.tail = nuevo;
        }
        this.tamanio++;
    }

    public void agregarDespuesDe(T referencia, T valor) {
```

PROBLEMS 7 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS — Run: Main

```
El dato esta en la posicion:3
PS C:\Users\axelx\Documents\FES\3er semestre\Estructura de Datos> ^C
PS C:\Users\axelx\Documents\FES\3er semestre\Estructura de Datos>
PS C:\Users\axelx\Documents\FES\3er semestre\Estructura de Datos>  c:; cd 'c:\Users\axelx\Documents\FES\3er semestre\Estructura de Datos'; &
 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\axelx\Documents\FES\3er semestre\Estr
uctura de Datos\Tareas\Tarea 5\tarea5\target\classes' 'listas.unam.Main'
<--| 50 |--><--| 60 |--><--| 65 |--><--| 70 |--><--| 80 |--><--| 90 |-->
<--| 50 |--><--| 60 |--><--| 70 |--><--| 80 |--><--| 90 |-->
<--| 50 |--><--| 60 |--><--| 70 |--><--| 80 |--><--| 88 |-->
El dato esta en la posicion:3
PS C:\Users\axelx\Documents\FES\3er semestre\Estructura de Datos>
```

```java
package listas.unam;

public class Main {
    public static void main(String[] args) {
        DoubleLinkedList<Integer> numeros= new DoubleLinkedList<>();

        numeros.agregarAlInicio(90);
        numeros.agregarAlInicio(80);
        numeros.agregarAlInicio(70);
        numeros.agregarAlInicio(65);
        numeros.agregarAlInicio(60);
        numeros.agregarAlInicio(50);
        numeros.transversal("derecha");

        numeros.eliminar(2);
        numeros.transversal("derecha");

        numeros.actualizar(4, 88);
        numeros.transversal("derecha");

        numeros.buscar(80);

    }
}
```

```java
package listas.unam;
public class DoubleLinkedList<T> {
    private NodoDoble<T> head;
    private NodoDoble<T> tail;
    private int tamanio;

    public DoubleLinkedList() {
    }

    public boolean estaVacia() {
        boolean res = false;
        if (this.head == null && this.tail == null) {
            res = true;
        }
        return res;
    }

    public int getTamanio() {
        return tamanio;
    }
```

```java
public void agregarAlInicio(T valor) {
    NodoDoble<T> nuevo = new NodoDoble<>(valor);
    if (this.estaVacia()) {
        this.head = nuevo;
        this.tail = nuevo;
    } else {
        this.head.setAnterior(nuevo);
        nuevo.setSiguiente(this.head);
        this.head = nuevo;
    }
    this.tamanio++;
}

public void agregarAlFinal(T valor){
    NodoDoble<T> nuevo = new NodoDoble<>(valor);
    if (this.estaVacia()) {
        this.head = nuevo;
        this.tail = nuevo;
    } else {
        this.tail.setSiguiente(nuevo);
        nuevo.setAnterior(tail);
        this.tail = nuevo;
    }
    this.tamanio++;
}

public void agregarDespuesDe(T referencia, T valor) {
    NodoDoble<T> aux = this.head;
    while (aux.getData() != referencia){
        aux = aux.getSiguiente();
    }
    if(aux!= null){
        NodoDoble<T> nuevo = new NodoDoble<>(valor);
        nuevo.setSiguiente(aux.getSiguiente());
        nuevo.setAnterior(aux);

        if (aux.getSiguiente() != null) {
            aux.getSiguiente().setAnterior(nuevo);
        } else {
            tail = nuevo;
        }
        aux.setSiguiente(nuevo);
        tamanio++;
    }
```

```
        }

    public T obtener(int posicion) {
        if (posicion >= 0 && posicion < tamanio) {
            NodoDoble<T> aux = head;
            for (int i = 0; i < posicion; i++) {
                aux = aux.getSiguiente();
            }
            return aux.getData();
        }
        throw new IndexOutOfBoundsException("Posición inválida");
    }

    public void eliminarPrimero(){
            if (head == tail) {
                head = tail = null;
            } else {
                head = head.getSiguiente();
                if (head != null) {
                    head.setAnterior(null);
                }
            }
            tamanio--;
    }

    public void eliminarFinal(){
        if (!estaVacia()) {
            if (head == tail) {
                head = tail = null;
            } else {
                tail = tail.getAnterior();
                if (tail != null) {
                    tail.setSiguiente(null);
                }
            }
            tamanio--;
        }
    }

    public void eliminar(int pos){
        if (pos == 0) {
            eliminarPrimero();
        }
        else if (pos == tamanio - 1) {
            eliminarFinal();
```

```
        }else{
        NodoDoble<T> aux = this.head;
        for (int i = 0; i < pos; i++){
            aux = aux.getSiguiente();
        }
        aux.getSiguiente().setAnterior(aux.getAnterior());
        aux.getAnterior().setSiguiente(aux.getSiguiente());
        tamanio--;
        }
    }

    public int buscar(T valor) {
        NodoDoble<T> aux = head;  // Asegúrate de que 'cabeza' sea el nombre
de tu nodo inicial.
        int pos = 0;
        while (aux != null) {
            if (aux.getData().equals(valor)) {
                System.out.println("El dato esta en la posicion:" +pos);
                return pos;  // Devuelve la posición si se encuentra el
valor.
            }
            aux = aux.getSiguiente();  // Avanza al siguiente nodo.
            pos++;
        }
        return -1;  // Devuelve -1 si no se encuentra el valor.
    }


    public void actualizar(int aBuscar, T valor) {
        if (aBuscar >= 0 && aBuscar < tamanio) {
            NodoDoble<T> aux;
                if (aBuscar < tamanio / 2) {

                 aux = head;
                for (int i = 0; i < aBuscar; i++) {
                aux = aux.getSiguiente();
                }
                } else {

                    aux = tail;
                    for (int i = tamanio - 1; i > aBuscar; i--) {
                     aux = aux.getAnterior();
                    }
                    }
                    aux.setData(valor);
```

```
                } else {
                throw new IndexOutOfBoundsException("Posición inválida");
                }

        }


    public void transversal(String direccion) {
        if (direccion.equalsIgnoreCase("derecha")) {
            NodoDoble<T> aux = head;
            while (aux != null) {
                System.out.print(aux);
                aux = aux.getSiguiente();
            }
        } else if (direccion.equalsIgnoreCase("izquierda")) {
            NodoDoble<T> aux = tail;
            while (aux != null) {
                System.out.print(aux);
                aux = aux.getAnterior();
            }
        }
        System.out.println("");
    }
}


package listas.unam;
public class NodoDoble<T> {
    private T data;
    private NodoDoble<T> siguiente;
    private NodoDoble<T> anterior;

    public NodoDoble() {
    }

    public NodoDoble(T data) {
        this.data = data;
    }

    public NodoDoble(T data, NodoDoble<T> siguiente, NodoDoble<T> anterior)
{
        this.data = data;
        this.siguiente = siguiente;
        this.anterior = anterior;
    }
```

```java
    public T getData() {
        return data;
    }

    public void setData(T data) {
        this.data = data;
    }

    public NodoDoble<T> getSiguiente() {
        return siguiente;
    }

    public void setSiguiente(NodoDoble<T> siguiente) {
        this.siguiente = siguiente;
    }

    public NodoDoble<T> getAnterior() {
        return anterior;
    }

    public void setAnterior(NodoDoble<T> anterior) {
        this.anterior = anterior;
    }

    @Override
    public String toString() {
        return "<--| "+ this.data +" |-->";
    }
}
```