# Atividade - Lista Duplamente Encadeada

Tecnólogo em Análise e Desenvolvimento de Sistemas
Estrutura de Dados II
Jonathas Jivago de Almeida Cruz

José Nilton Silva Lima

## Introdução

A estrutura de dados *Lista Duplamente Encadeada* consiste em uma coleção linear de nós onde cada elemento mantém referências tanto para o próximo quanto para o anterior na sequência. Essa característica a diferencia da lista simplesmente encadeada, permitindo navegação bidirecional e operações mais eficientes em determinados cenários.

As principais operações associadas à lista duplamente encadeada são:

- `inserirNoInicio(valor)`: adiciona elemento no início da lista
- `inserirNoFim(valor)`: adiciona elemento no final da lista
- `inserirNaPosicao(valor, posicao)`: insere elemento em posição específica
- `removerNoInicio()`: remove e retorna o primeiro elemento
- `removerNoFim()`: remove e retorna o último elemento
- `removerNaPosicao(posicao)`: remove elemento em posição específica
- `exibirOrdemNormal()`: exibe elementos do início ao fim
- `exibirOrdemInversa()`: exibe elementos do fim ao início
- `estaVazia()`: verifica se a lista está vazia
- `esvaziar()`: remove todos os elementos
- `obterTamanho()`: retorna quantidade de elementos

## Implementação em TypeScript

```
1 class No<T> {
2     public valor: T;
3     public proximo: No<T> | null;
```

```
4       public anterior: No<T> | null;
5
6       constructor(valor: T) {
7           this.valor = valor;
8           this.proximo = null;
9           this.anterior = null;
10      }
11  }
12
13  class ListaDuplamenteEncadeada<T> {
14      private cabeca: No<T> | null;
15      private cauda: No<T> | null;
16      private tamanho: number;
17
18      constructor() {
19          this.cabeca = null;
20          this.cauda = null;
21          this.tamanho = 0;
22      }
23
24      public inserirNoInicio(valor: T): void {
25          const novoNo = new No(valor);
26
27          if (this.estaVazia()) {
28              this.cabeca = novoNo;
29              this.cauda = novoNo;
30          } else {
31              novoNo.proximo = this.cabeca;
32              this.cabeca!.anterior = novoNo;
33              this.cabeca = novoNo;
34          }
35          this.tamanho++;
36      }
37
38      public inserirNoFim(valor: T): void {
39          const novoNo = new No(valor);
40
41          if (this.estaVazia()) {
42              this.cabeca = novoNo;
43              this.cauda = novoNo;
44          } else {
45              novoNo.anterior = this.cauda;
46              this.cauda!.proximo = novoNo;
47              this.cauda = novoNo;
48          }
49          this.tamanho++;
50      }
51
52      public inserirNaPosicao(valor: T, posicao: number): void {
53          if (posicao < 0 || posicao > this.tamanho) {
54              throw new Error("Posição inválida");
```

```
        }

        if (posicao === 0) return this.inserirNoInicio(valor);
        if (posicao === this.tamanho) return this.inserirNoFim(valor);

        const novoNo = new No(valor);
        let atual = this.cabeca;

        for (let i = 0; i < posicao - 1; i++) {
            atual = atual!.proximo;
        }

        novoNo.proximo = atual!.proximo;
        novoNo.anterior = atual;
        atual!.proximo!.anterior = novoNo;
        atual!.proximo = novoNo;

        this.tamanho++;
    }

    public removerNoInicio(): T | null {
        if (this.estaVazia()) return null;

        const valorRemovido = this.cabeca!.valor;

        if (this.tamanho === 1) {
            this.cabeca = null;
            this.cauda = null;
        } else {
            this.cabeca = this.cabeca!.proximo;
            this.cabeca!.anterior = null;
        }

        this.tamanho--;
        return valorRemovido;
    }

    public removerNoFim(): T | null {
        if (this.estaVazia()) return null;

        const valorRemovido = this.cauda!.valor;

        if (this.tamanho === 1) {
            this.cabeca = null;
            this.cauda = null;
        } else {
            this.cauda = this.cauda!.anterior;
            this.cauda!.proximo = null;
        }

        this.tamanho--;
```

```
106         return valorRemovido;
107     }
108 
109     public removerNaPosicao(posicao: number): T | null {
110         if (posicao < 0 || posicao >= this.tamanho || this.estaVazia()) {
111             return null;
112         }
113 
114         if (posicao === 0) return this.removerNoInicio();
115         if (posicao === this.tamanho - 1) return this.removerNoFim();
116 
117         let atual = this.cabeca;
118         for (let i = 0; i < posicao; i++) {
119             atual = atual!.proximo;
120         }
121 
122         atual!.anterior!.proximo = atual!.proximo;
123         atual!.proximo!.anterior = atual!.anterior;
124 
125         this.tamanho--;
126         return atual!.valor;
127     }
128 
129     public exibirOrdemNormal(): void {
130         let atual = this.cabeca;
131         let resultado = "";
132 
133         while (atual !== null) {
134             resultado += atual.valor + " <-> ";
135             atual = atual.proximo;
136         }
137 
138         console.log(resultado + "null");
139     }
140 
141     public exibirOrdemInversa(): void {
142         let atual = this.cauda;
143         let resultado = "";
144 
145         while (atual !== null) {
146             resultado += atual.valor + " <-> ";
147             atual = atual.anterior;
148         }
149 
150         console.log("null <-> " + resultado);
151     }
152 
153     public estaVazia(): boolean {
154         return this.tamanho === 0;
155     }
156 
```

```
    public esvaziar(): void {
        this.cabeca = null;
        this.cauda = null;
        this.tamanho = 0;
    }

    public obterTamanho(): number {
        return this.tamanho;
    }
}
```

# Exemplo de Uso

```
const lista = new ListaDuplamenteEncadeada<number>();

// Inserções
lista.inserirNoInicio(10);
lista.inserirNoFim(20);
lista.inserirNoInicio(5);
lista.inserirNaPosicao(15, 2);

// Exibição
lista.exibirOrdemNormal();
lista.exibirOrdemInversa();

// Remoções
lista.removerNoInicio();
lista.removerNoFim();
lista.removerNaPosicao(1);

// Verificações
console.log("Tamanho:", lista.obterTamanho());
console.log("Vazia?", lista.estaVazia());
```