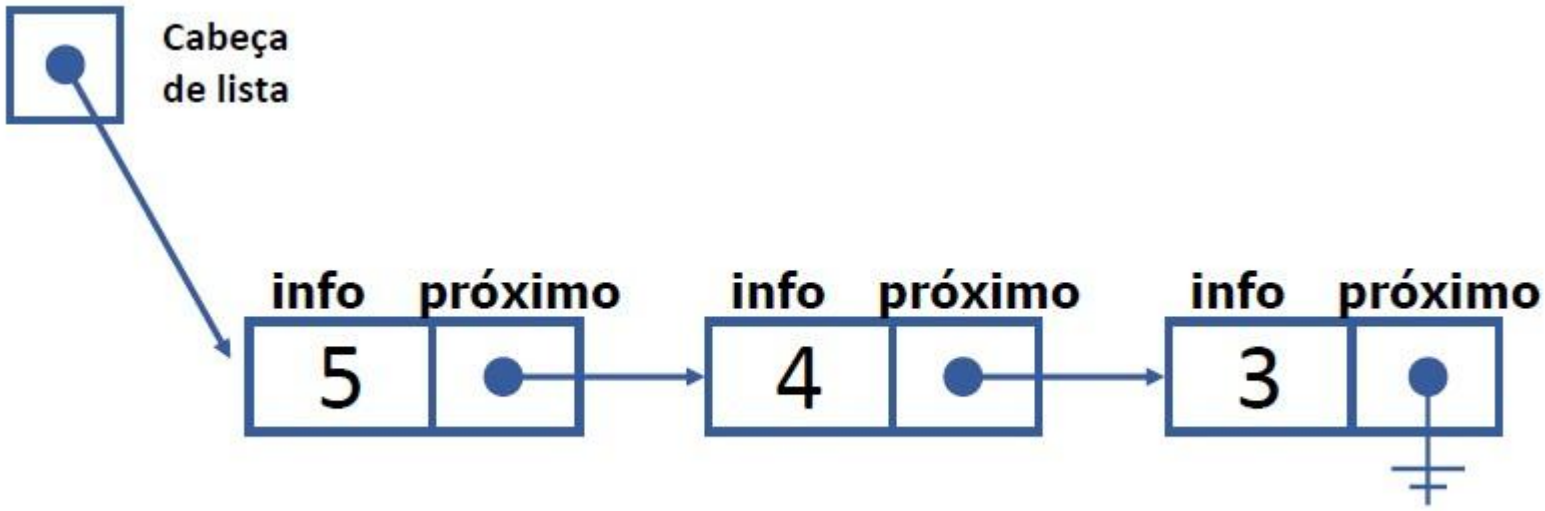
ESTRUTURA DE DADOS

Aula 8 – Lista Encadeada Dupla

Prof. Rodrigo Maciel

LISTA ENCADEADA SIMPLES - OBSERVAÇÕES

- Inserção e exclusão no início são operações muito rápidas;
- Localizar ou excluir próximo a um item específico requer buscar, em média, metade dos itens;
- Utiliza somente a memória que necessitar, podendo ser expandida de acordo com o aumento da lista;

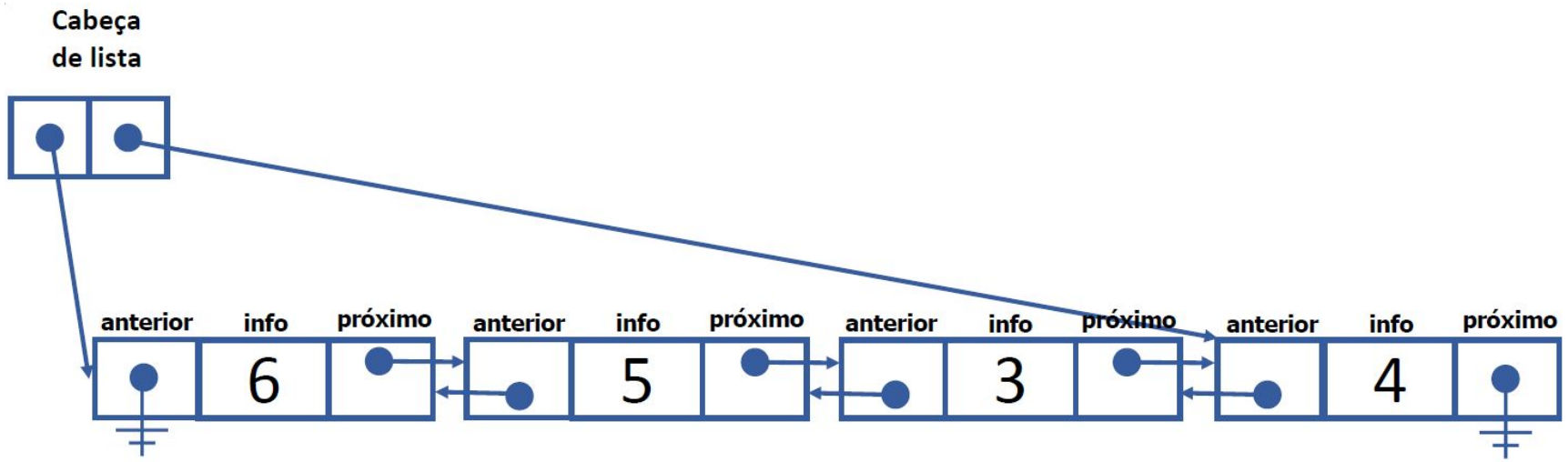


LISTA ENCADEADA SIMPLES - OBSERVAÇÕES

- Problema das listas encadeadas simples:
  - Um comando **atual = atual.proximo** vai para o próximo nó, mas não há maneira correspondente de ir para o nó anterior.
- Exemplo:
  - Em um editor de texto implementado com uma lista encadeada simples, não seria possível voltar para o caractere anterior.

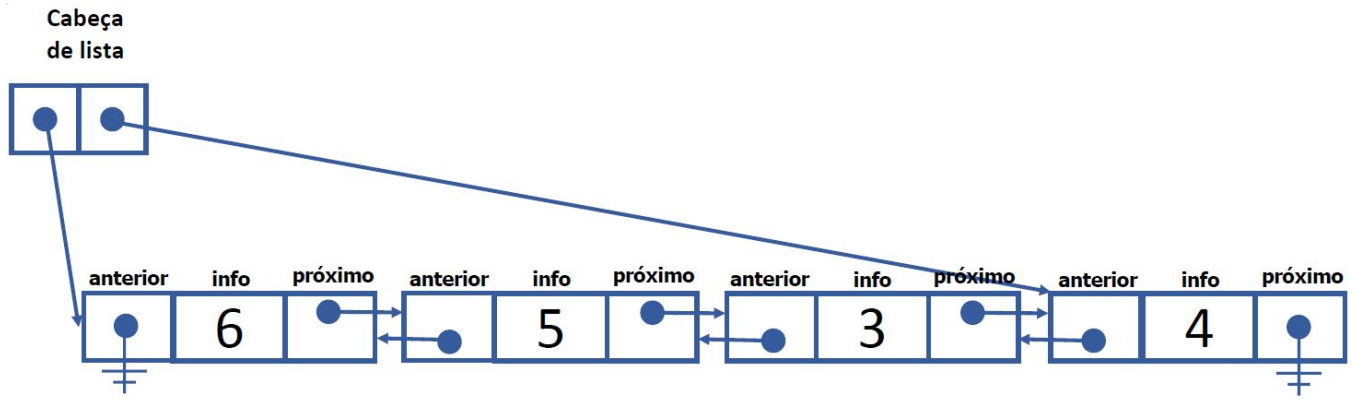
LISTA DUPLAMENTE ENCADEADA

- Permite percorrer a lista para trás, assim como para frente;
- Cada nó tem duas referências para outros nós, ao invés de uma;
- A primeira referência é para o próximo nó e a segunda é para o nó anterior.

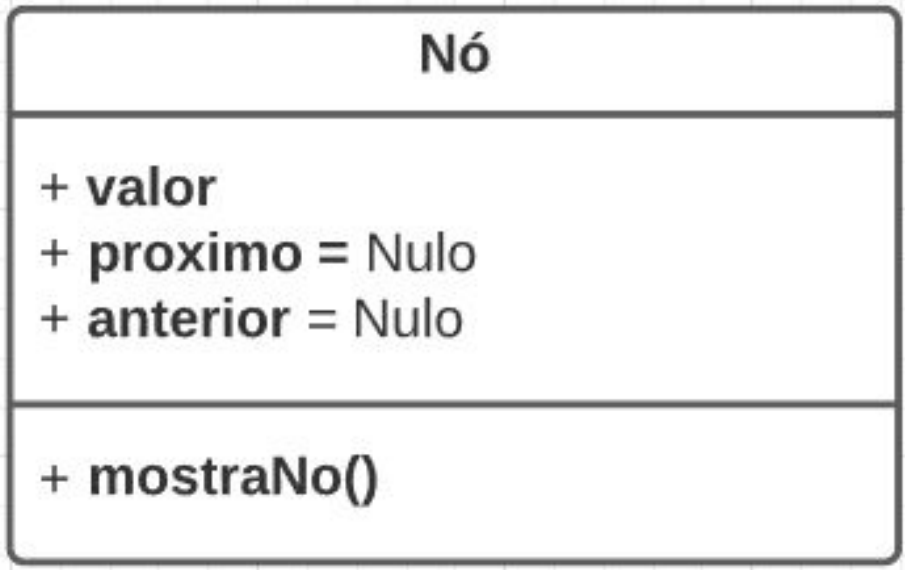


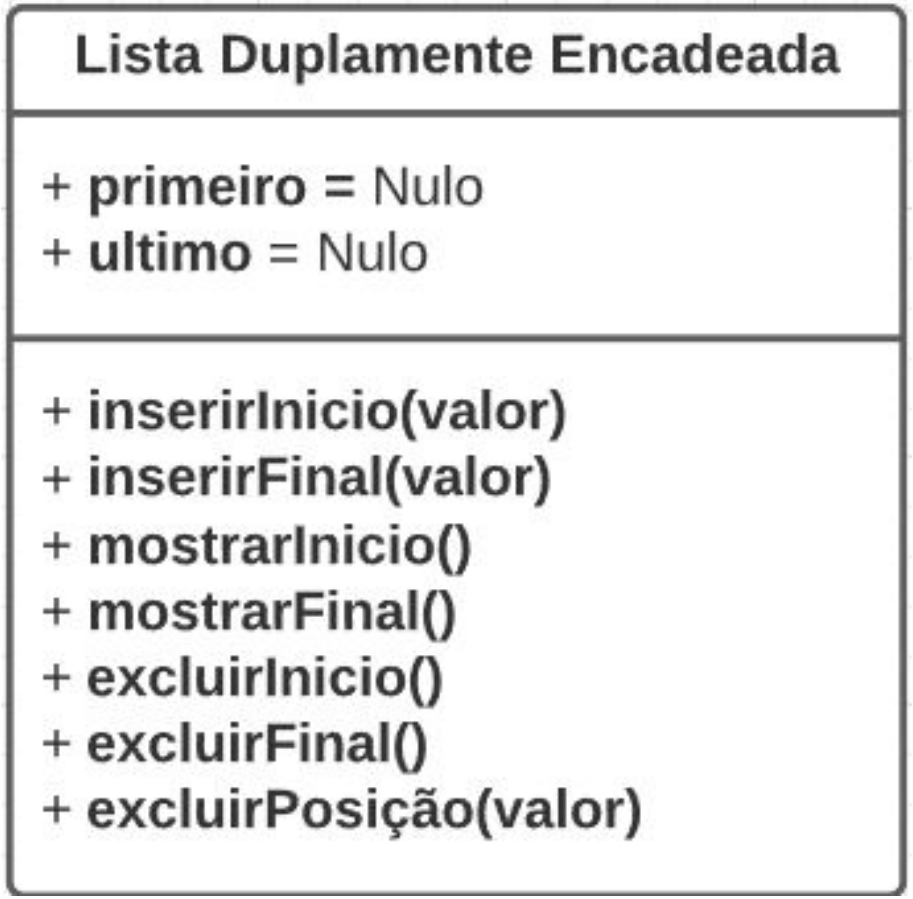
LISTA DUPLAMENTE ENCADEADA - OPERAÇÕES

- Inserir no início;
- Inserir no final;
- Mostrar pelo início;
- Mostrar pelo final;
- Pesquisar valor;
- Excluir no início;
- Excluir no final;
- Excluir de qualquer posição



IMPLEMENTAÇÃO - CLASSES





IMPLEMENTAÇÃO - INSERE NO INÍCIO

insere_inicio(valor):

novo <- No(valor)

se lista_vazia() **então**

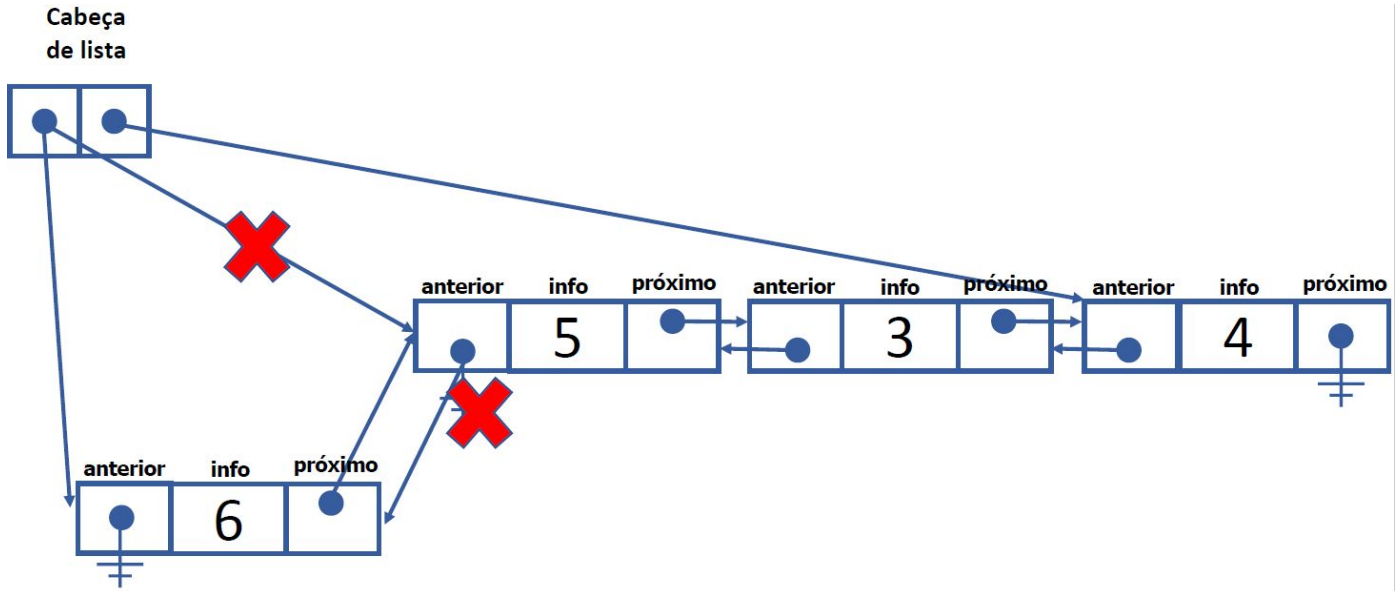
ultimo <- novo

senão

primeiro.anterior <- novo

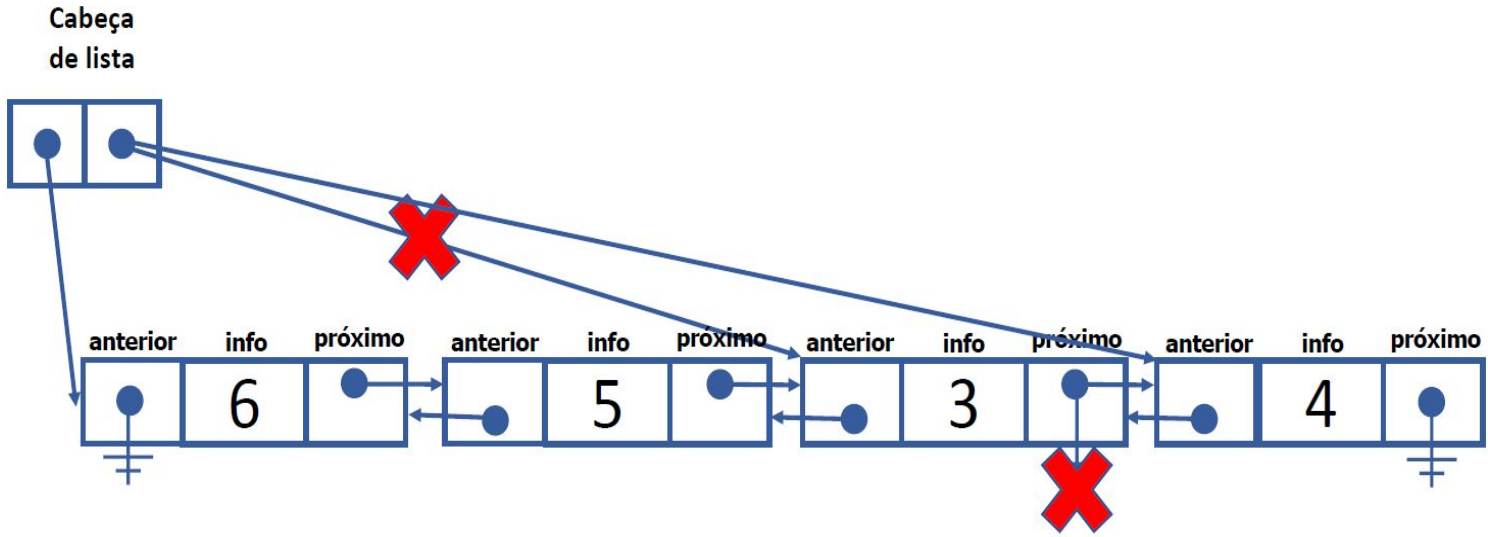
novo.proximo <- primeiro

primeiro <- novo



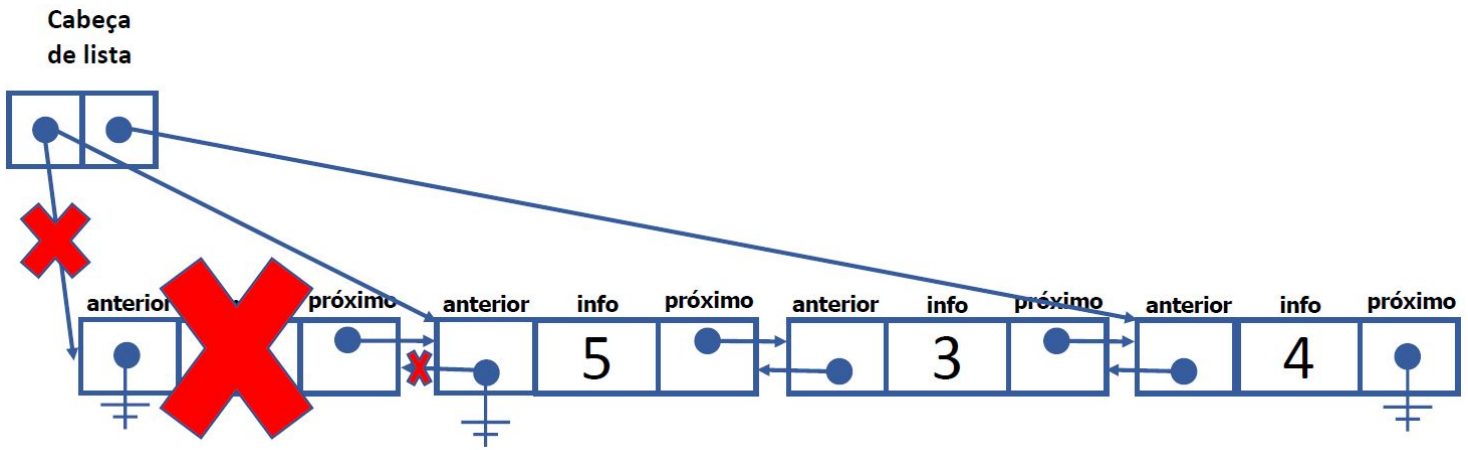
IMPLEMENTAÇÃO - INSERE NO FINAL

```
insere_final(valor):
    novo <- No(valor)
    se lista_vazia() então
        primeiro <- novo
    senão
        ultimo.proximo <- novo
        novo.anterior <- ultimo
    ultimo <- novo
```



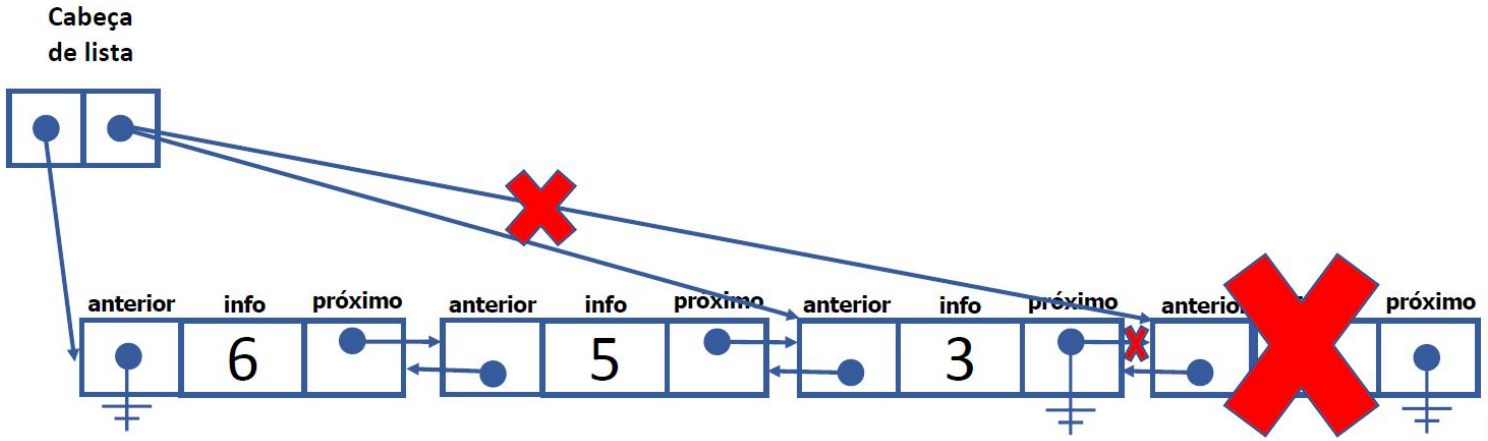
IMPLEMENTAÇÃO - EXCLUIR NO INÍCIO

```
excluir_inicio():
    temp <- primeiro
    se primeiro.proximo = Nulo então
        ultimo <- Nulo
    senão
        primeiro.proximo.anterior = Nulo
    primeiro <- primeiro.proximo
    return temp
```



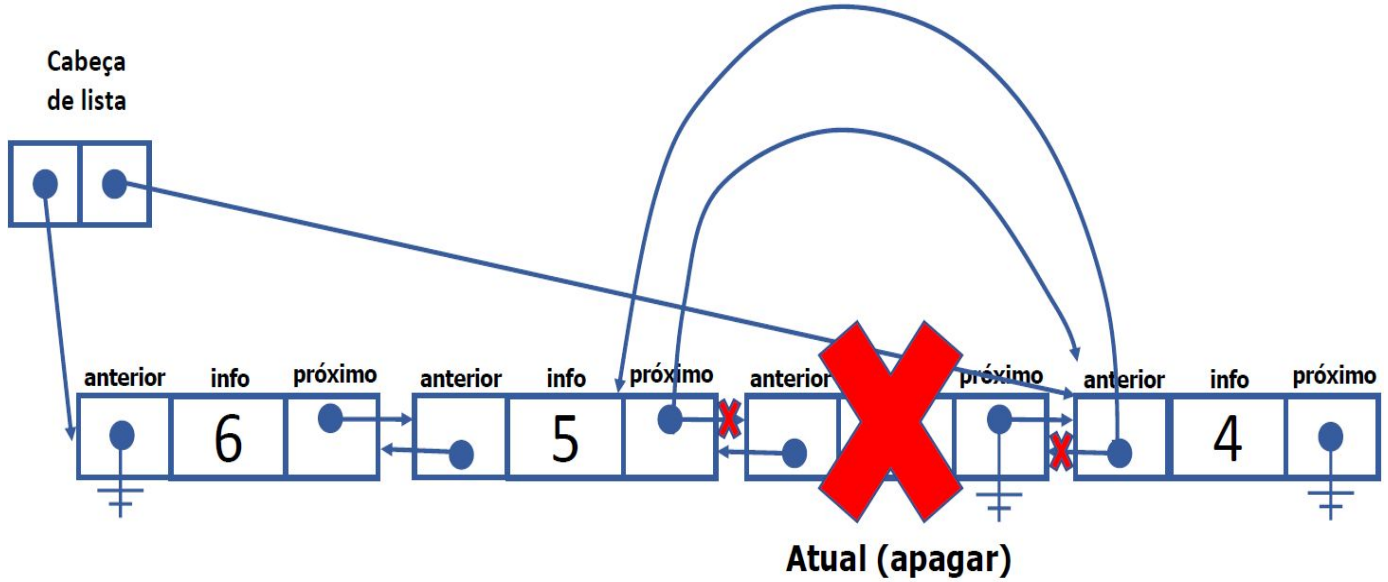
IMPLEMENTAÇÃO - EXCLUIR NO FINAL

```
excluir_final():
    temp <- ultimo
    se primeiro.proximo = Nulo então
        primeiro <- Nulo
    senão
        ultimo.anterior.proximo = Nulo
    ultimo <- ultimo.anterior
    return temp
```



IMPLEMENTAÇÃO - EXCLUIR QUALQUER VALOR

```
excluir_qualquer(valor):
    atual <- primeiro
    enquanto atual.valor ≠ valor então
        atual <- atual.proximo
        se atual = Nulo então
            return Nulo
    se atual = primeiro então
        primeiro <- atual.proximo
    senão
        atual.anterior.proximo <- atual.proximo
    se atual = ultimo então
        ultimo <- atual.anterior
    senão
        atual.proximo.anterior <- atual.anterior
    return atual
```



IMPLEMENTAÇÃO - MOSTRAR DO INÍCIO

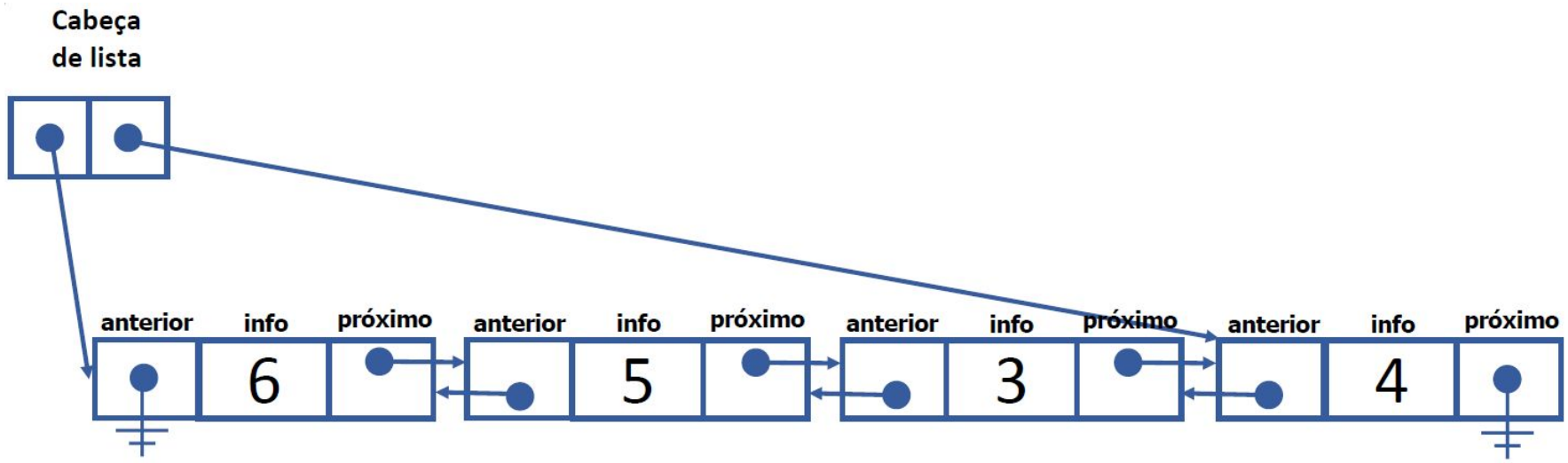
mostar_inicio():

atual <- primeiro

enquanto atual ≠ Nulo **então**

atual.mostra_no()

atual <- atual.proximo



IMPLEMENTAÇÃO - MOSTRAR DO FINAL

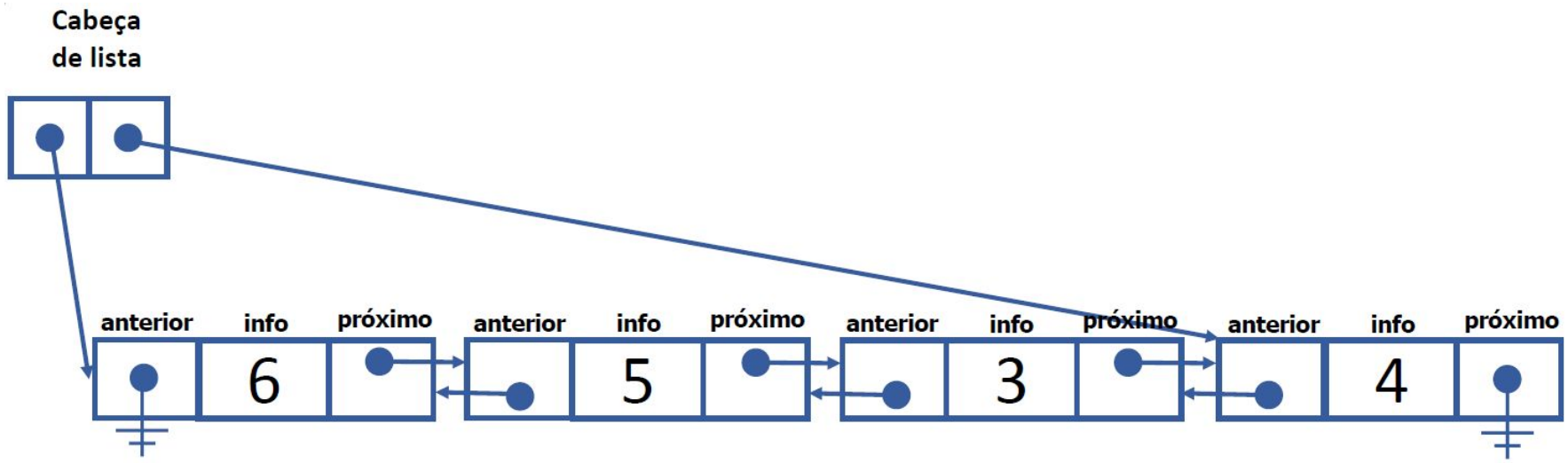
mostar_final():

atual <- ultimo

enquanto atual ≠ Nulo **então**

atual.mostra_no()

atual <- atual.anterior



IMPLEMENTAÇÃO - PESQUISAR VALOR

```
pesquisar(valor):
    se primeiro = Nulo então
        Erro de lista vazia
        return Nulo
    atual <- primeiro

    enquanto atual.valor ≠ valor então
        se atual.proximo = Nulo então
            return Nulo
        senão
            atual <- atual.proximo
    return atual
```