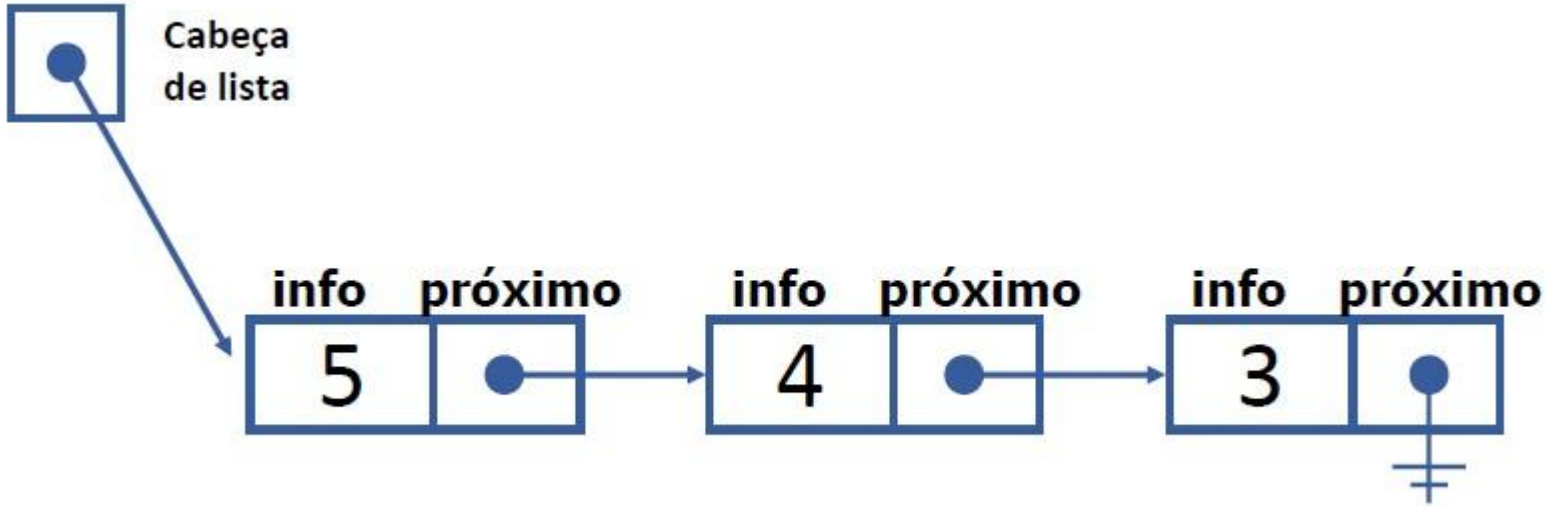
ESTRUTURA DE DADOS

Aula 7 – Lista Encadeada Simples

Prof. Thyerri

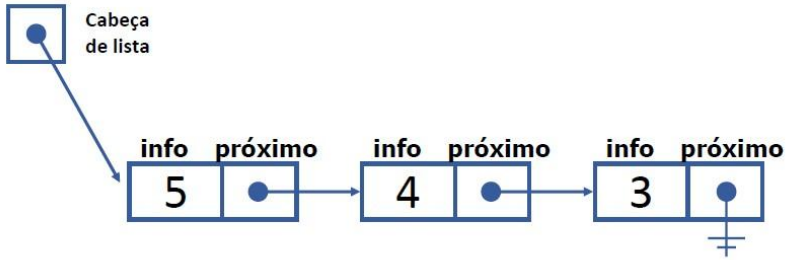
LISTA ENCADEADA SIMPLES - NÓS

- Cada item de dados é incorporado em um nó;
- Cada nó possui uma referência para o próximo nó da lista;
- Um campo da própria lista contém uma referência para o primeiro nó.



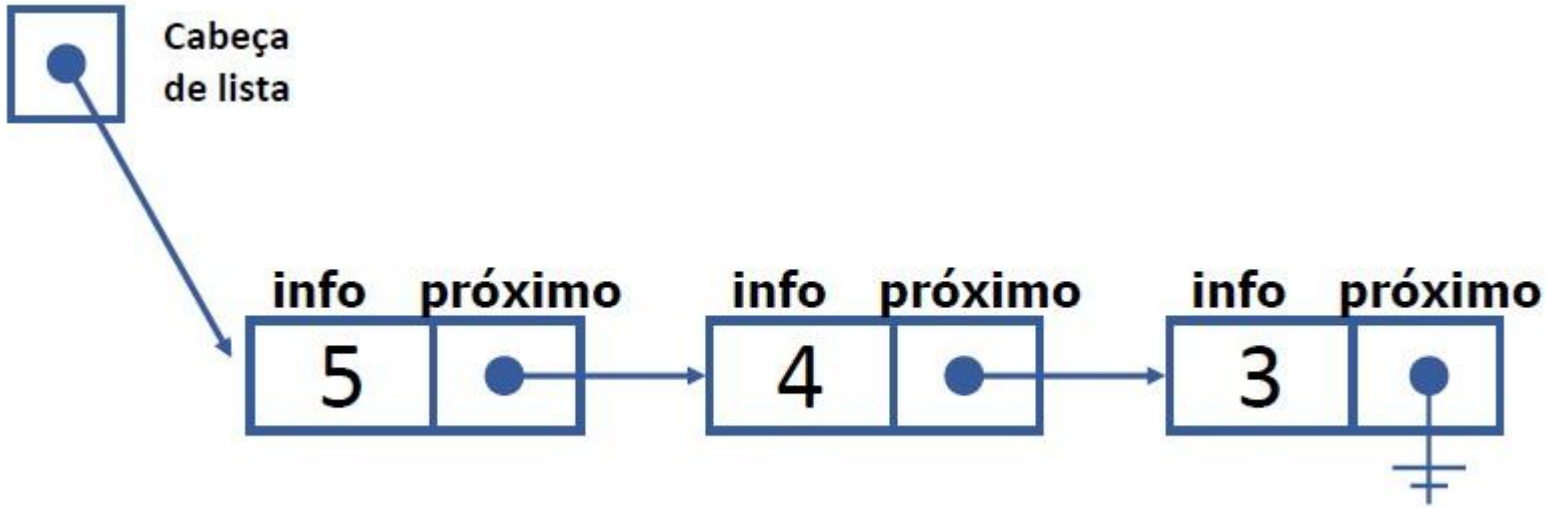
LISTAS ENCADEADAS - Posição X Relacionamento

- **Vetor (posição)**
  - Cada item ocupa uma certa posição;
  - Cada posição pode ser acessada usando um número de índice.
- **Lista (relacionamento)**
  - A única maneira de encontrar um elemento é seguir a sequência de elementos;
  - Um item de dados não pode ser acessado diretamente, ou seja, o relacionamento entre eles deve ser utilizado;
  - Inicia com o primeiro item, vai para o segundo, então o terceiro, até encontrar o item pesquisado.



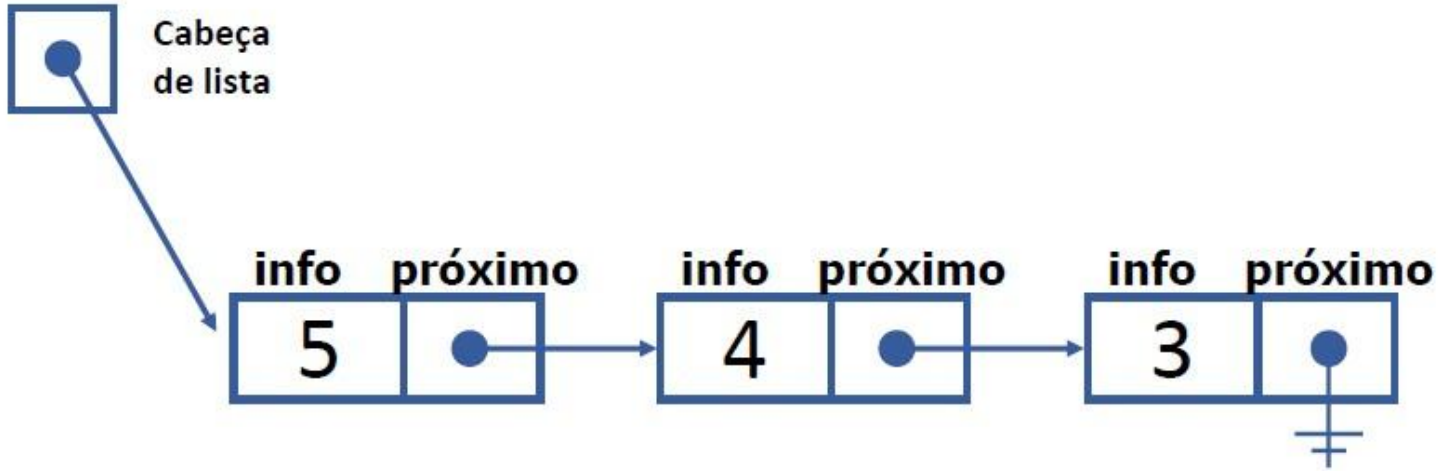
LISTAS ENCADEADAS

- Cada elemento da lista é armazenado em um objeto;
- Cada elemento da lista referencia o próximo e só é alocado dinamicamente quando é necessário;
- Para referenciar o primeiro elemento é utilizado um elemento chamado cabeça da lista.



LISTAS ENCADEADAS - OPERAÇÕES

- Inserir no início
- Excluir do início
- Mostrar lista
- Pesquisar
- Excluir de qualquer posição



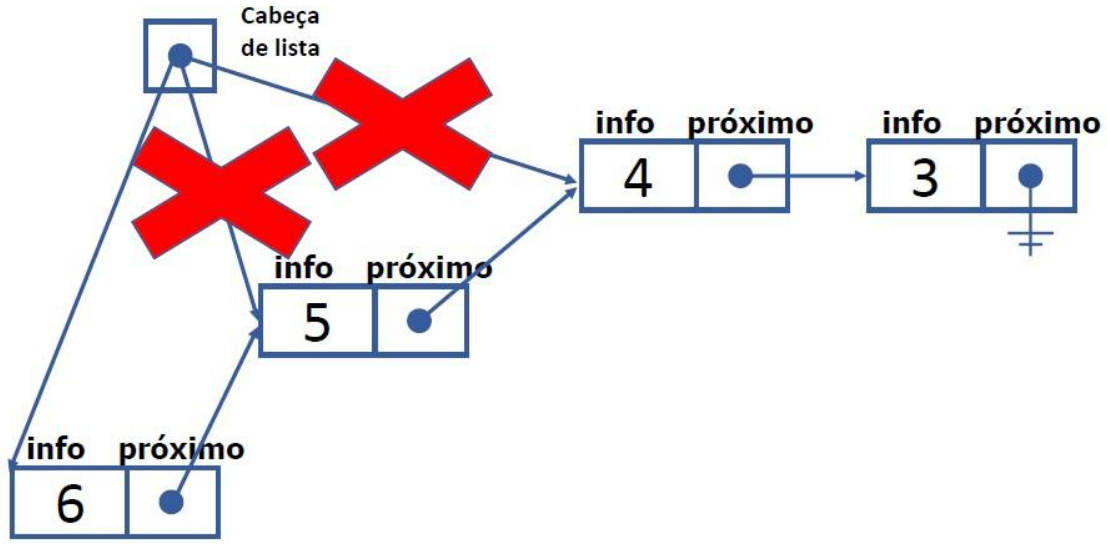
LISTAS ENCADEADAS - INSERE NO INÍCIO

- Insere um novo nó no início da lista;
- É o local mais fácil para inserir um nó;
- O próximo deste novo elemento deve apontar para quem era o primeiro da lista;
- A cabeça da lista aponta para o novo elemento.

LISTAS ENCADEADAS - INSERE NO INÍCIO

Procedimentos:

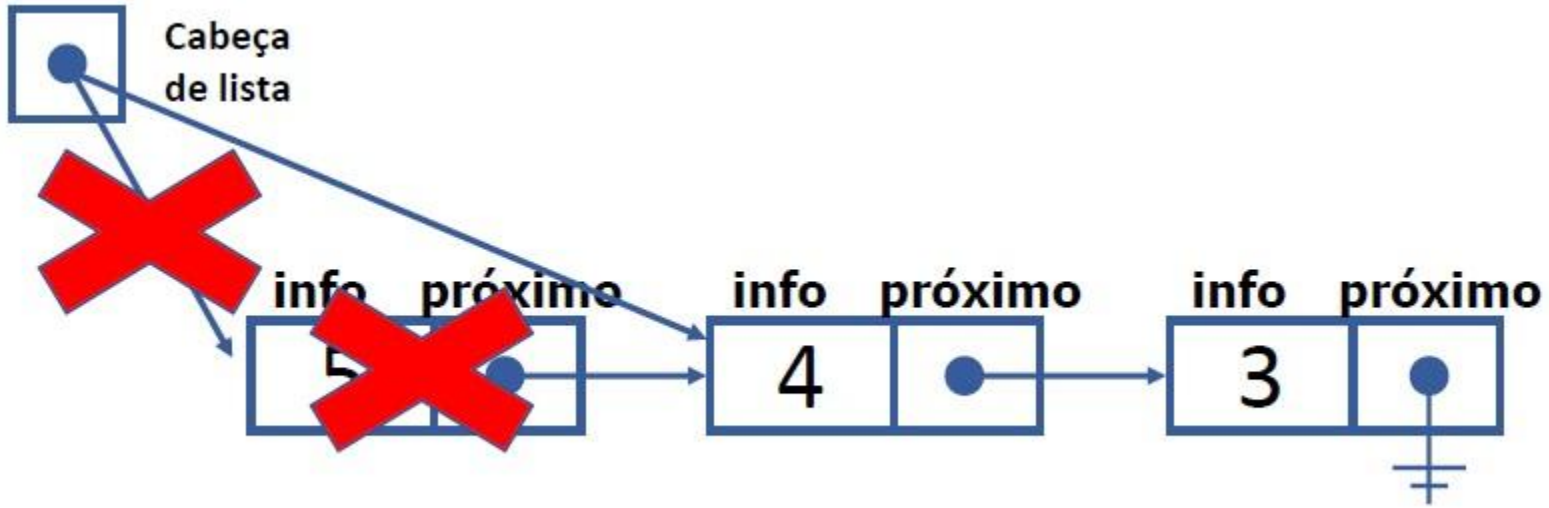
1. Criar um novo elemento (nó);
2. O próximo do novo nó criado vai apontar para o primeiro nó pré-existente da lista;
3. Destrói a conexão da cabeça da lista que aponta para o primeiro nó;
4. Faz o um novo apontamento da cabeça da lista para o nó recém criado.



LISTAS ENCADEADAS - EXCLUIR DO INÍCIO

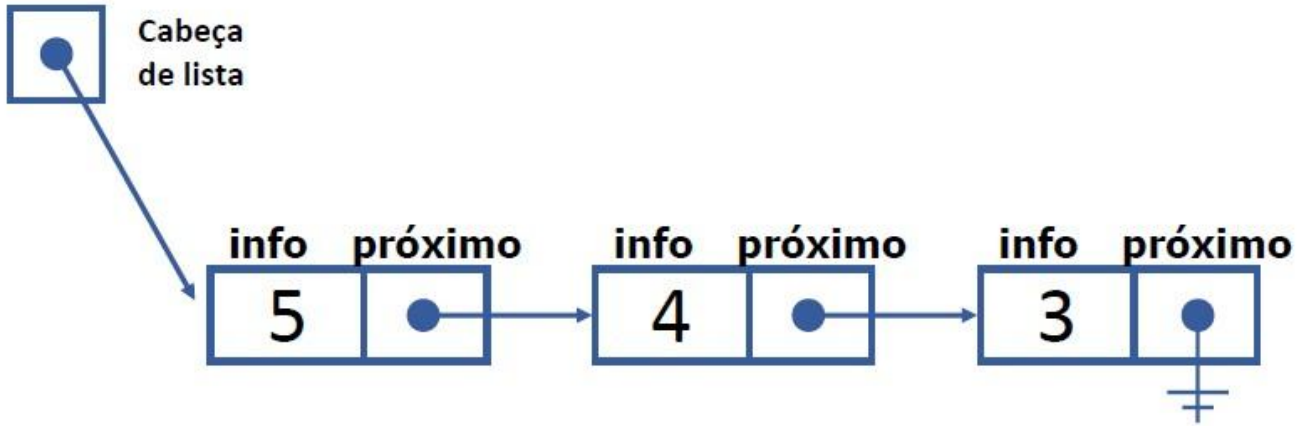
Procedimentos:

1. É o inverso do insere no início;
2. Desconecta o primeiro nó roteando o primeiro para apontar para o segundo nó;
3. O segundo nó é encontrado por meio do campo próximo no primeiro nó.



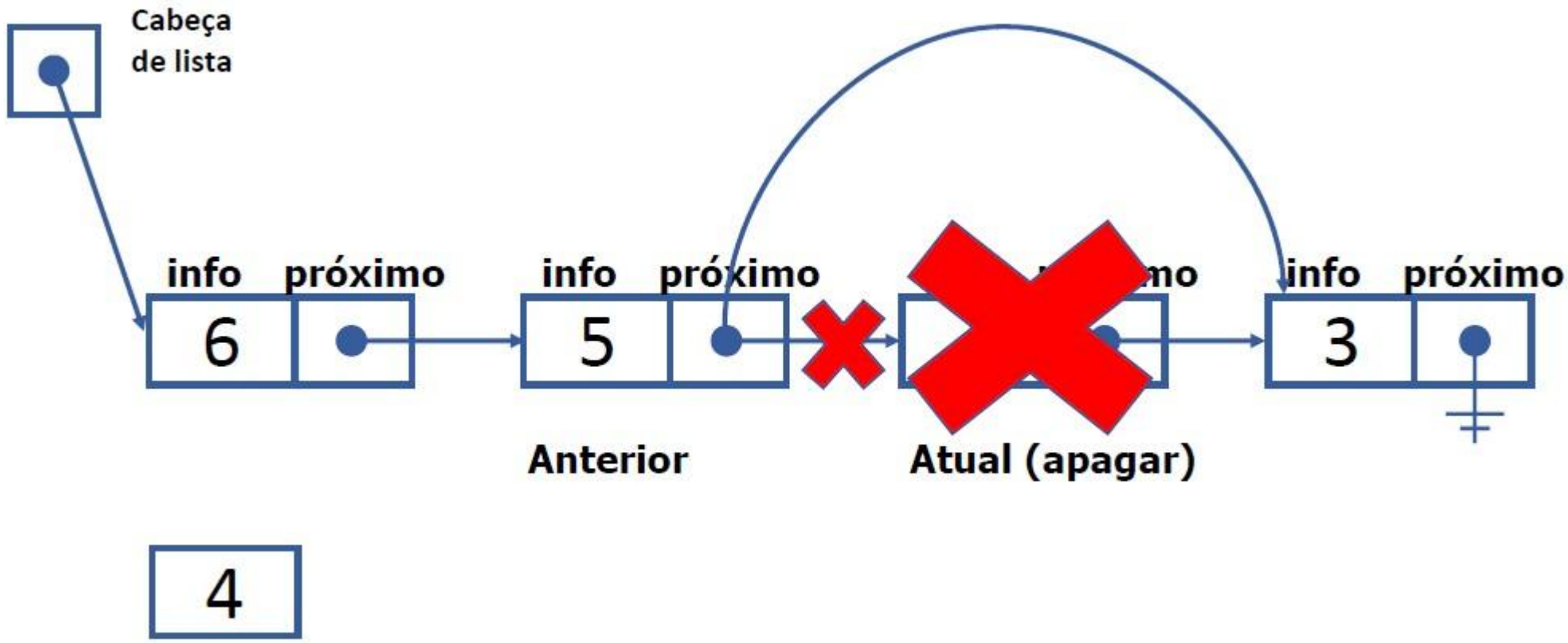
LISTAS ENCADEADAS - MOSTRAR LISTA E PESQUISAR

- Para exibir a lista, deverá ser iniciado no primeiro elemento, seguindo a sequência de referências de nó em nó;
- O final da lista é indicado pelo campo próximo no último nó apontando para null/none ao invés de outro nó;
- Os nós são percorridos e é verificado se o valor do elemento é aquele que está sendo procurado;
- Se atingir o final da lista sem encontrar o nó desejado, finaliza sem encontrar o elemento.



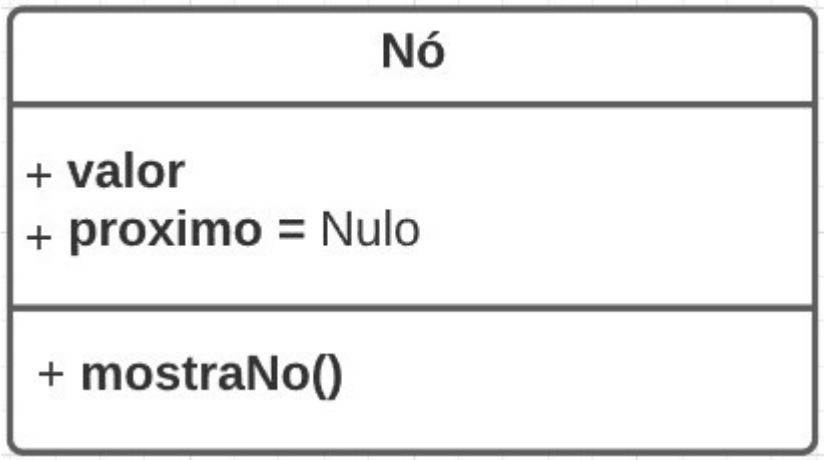
LISTAS ENCADEADAS - EXCLUIR DE POSIÇÃO QUALQUER

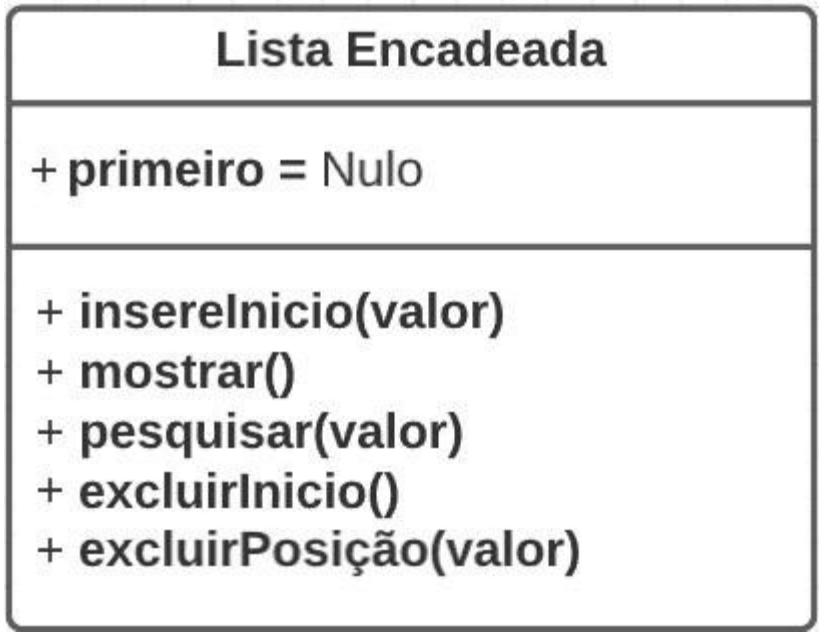
- O nó a ser eliminado deve ser localizado (pesquisa);
- Quando elimina o nó atual, terá que conectar o nó anterior ao nó seguinte.



LISTAS ENCADEADA SIMPLES - IMPLEMENTAÇÃO

Classes:





LISTAS ENCADEADA SIMPLES - IMPLEMENTAÇÃO

Algoritmo 1: Mostra Nó

mostraNo():

```
    imprime(valor)
```

Algoritmo 2: Inserir no início

inserirInicio(valor):

```
    novo <- Nó(valor)
    novo.próximo <- primeiro
    primeiro <- novo
```

LISTAS ENCADEADA SIMPLES - IMPLEMENTAÇÃO

Algoritmo 3: Mostrar Lista

```
mostar():
    se primeiro = Nulo então:
        imprime(Erro de lista vazia)
        return Nulo
    atual <- primeiro
    enquanto atual ≠ Nulo então:
        atual.mostraNo()
        atual <- atual.próximo
```

LISTAS ENCADEADA SIMPLES - IMPLEMENTAÇÃO

Algoritmo 4: Excluir no Início da Lista

```
excluirInicio():
    se primeiro = Nulo então:
        imprime(Erro de lista vazia)
        return Nulo
    temp <- primeiro
    primeiro <- primeiro.próximo
    return temp
```

LISTAS ENCADEADA SIMPLES - IMPLEMENTAÇÃO

Algoritmo 5: Pesquisar na Lista

```
pesquisar(valor):
    se primeiro = Nulo então:
        imprime(Erro de lista vazia)
        return Nulo
    atual <- primeiro
    enquanto atual.valor ≠ valor então:
        se atual.próximo = Nulo então:
            return Nulo
        senão:
                atual <- atual.proximo
    return atual
```

LISTAS ENCADEADA SIMPLES - IMPLEMENTAÇÃO

Algoritmo 6: Excluir em qualquer posição da Lista

```
excluirQualquer(valor):
    se primeiro = Nulo então:
        imprime(Erro de lista vazia)
        return Nulo
    atual <- primeiro
    anterior <- primeiro
    enquanto atual.valor ≠ valor então:
        se atual.próximo = Nulo então:
                return Nulo
        senão:
                anterior <- atual
                atual <- atual.próximo
    se atual = primeiro então:
        primeiro <- primeiro.próximo
    senão:
        anterior.próximo <- atual.próximo
    return atual
```