

LISTAS LINEARES

LISTA LINEAR SIMPLESMENTE ENCADEADA - LLSE

Listas com Vetores: Desvantagens

- Tamanho máximo fixo;
- Mesmo vazias ocupam um grande espaço de memória:
 - mesmo que utilizemos um vetor de ponteiros, se quisermos prever uma lista de 10.000 elementos, teremos 40.000 bytes desperdiçados;
- Operações podem envolver muitos deslocamentos de dados:
 - inclusão em uma posição ou no início;
 - exclusão em uma posição ou no início.

LLS Encadeada

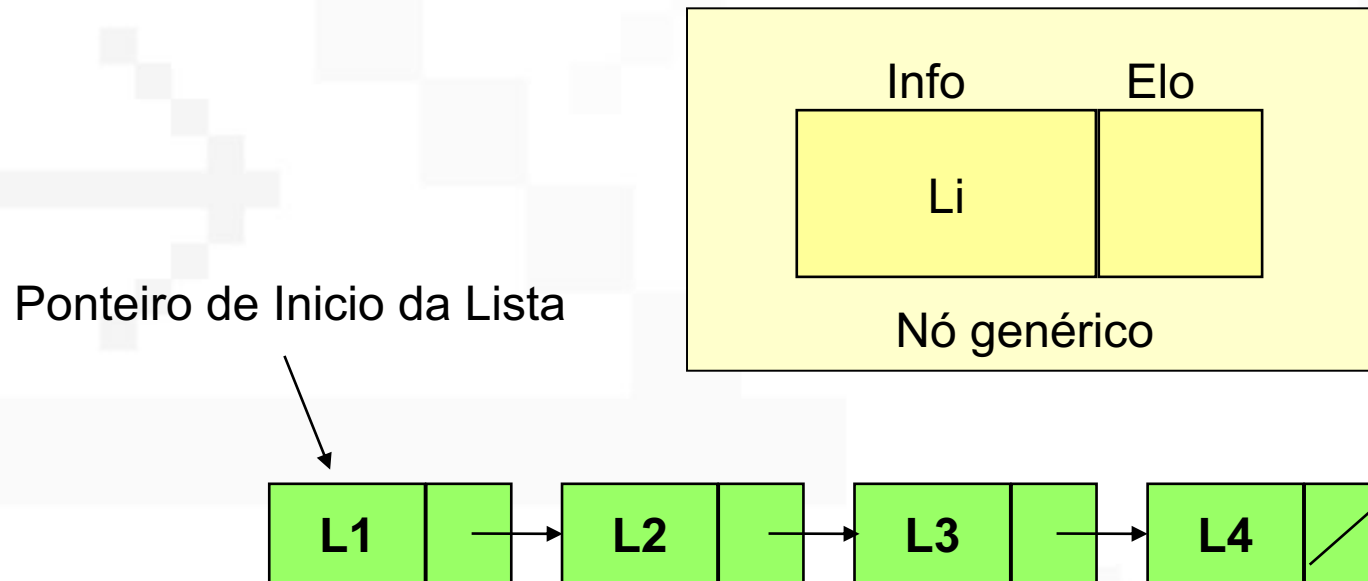
- Ordem dos nós da lista definida por informação contida em cada nó
- Ordem independe da posição física dos nós
- Contiguidade lógica dos nós na LLS Encadeada

Campo de elo

- contido em cada nó
- informa qual o próximo nó da lista

Requisitos para LLS Encadeada

- Ponteiro para o primeiro nó da lista
- Encadeamento entre os nós através do campo de elo
- Indicação de final de lista

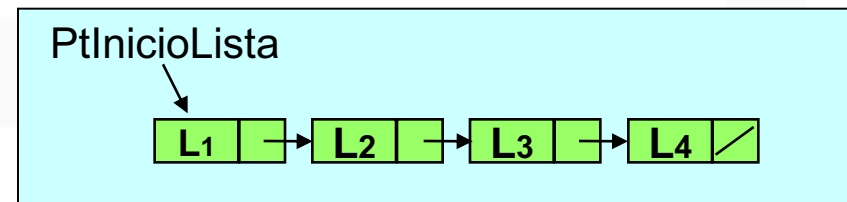
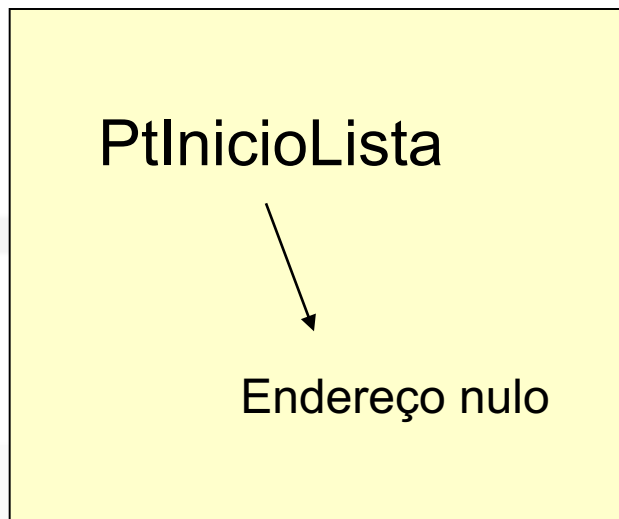


Operações básicas

- Criar e inicializar uma lista
- Inserir novo nó
- Remover um nó
- Acessar um nó
- Destruir lista

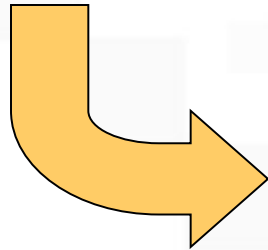
Criação de LLS Encadeada

- Inicializar apontador do início da lista em endereço nulo
- Lista inicialmente vazia



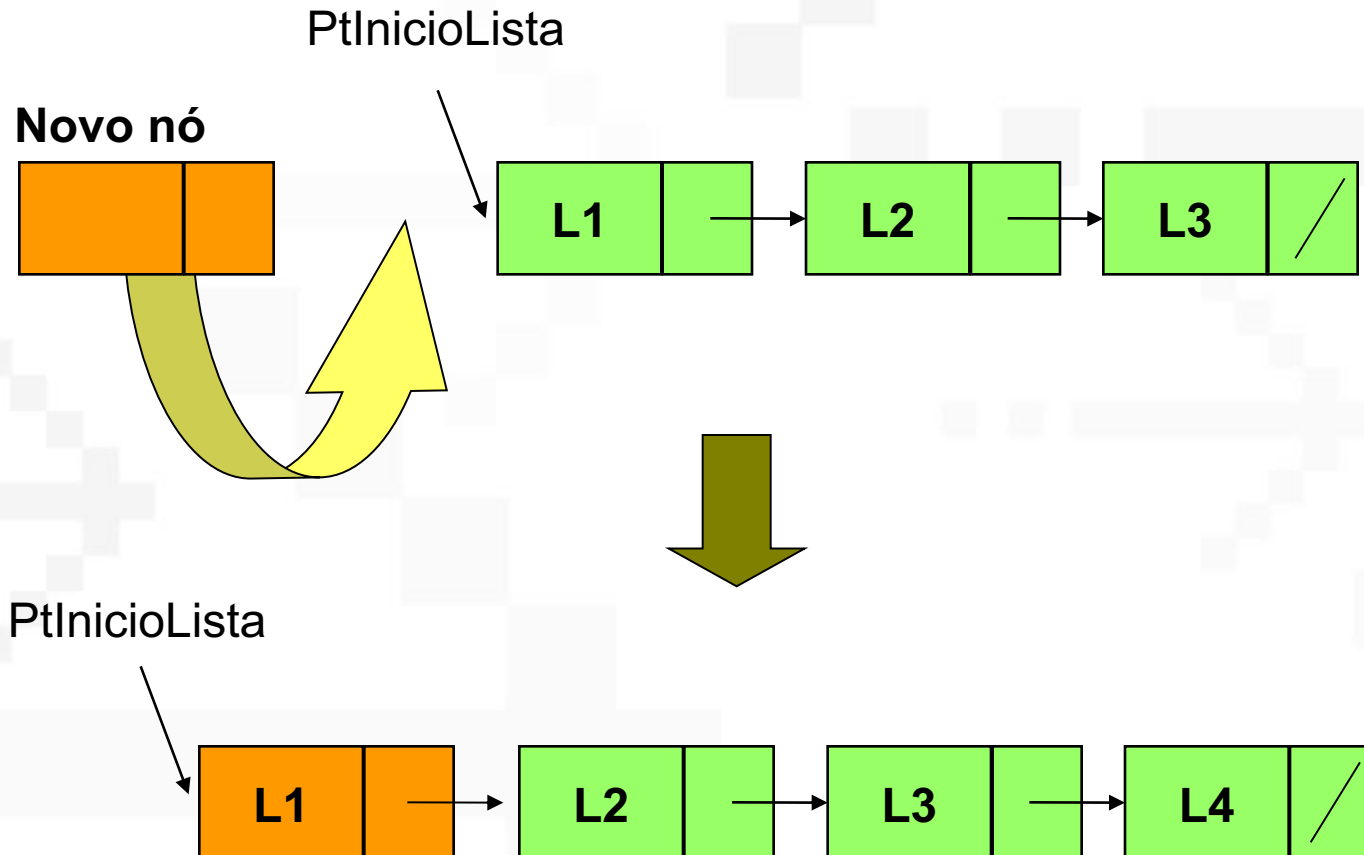
Inserção de um novo nó

- Alocar o novo nó
- Preencher com valor
- Encadear na posição solicitada

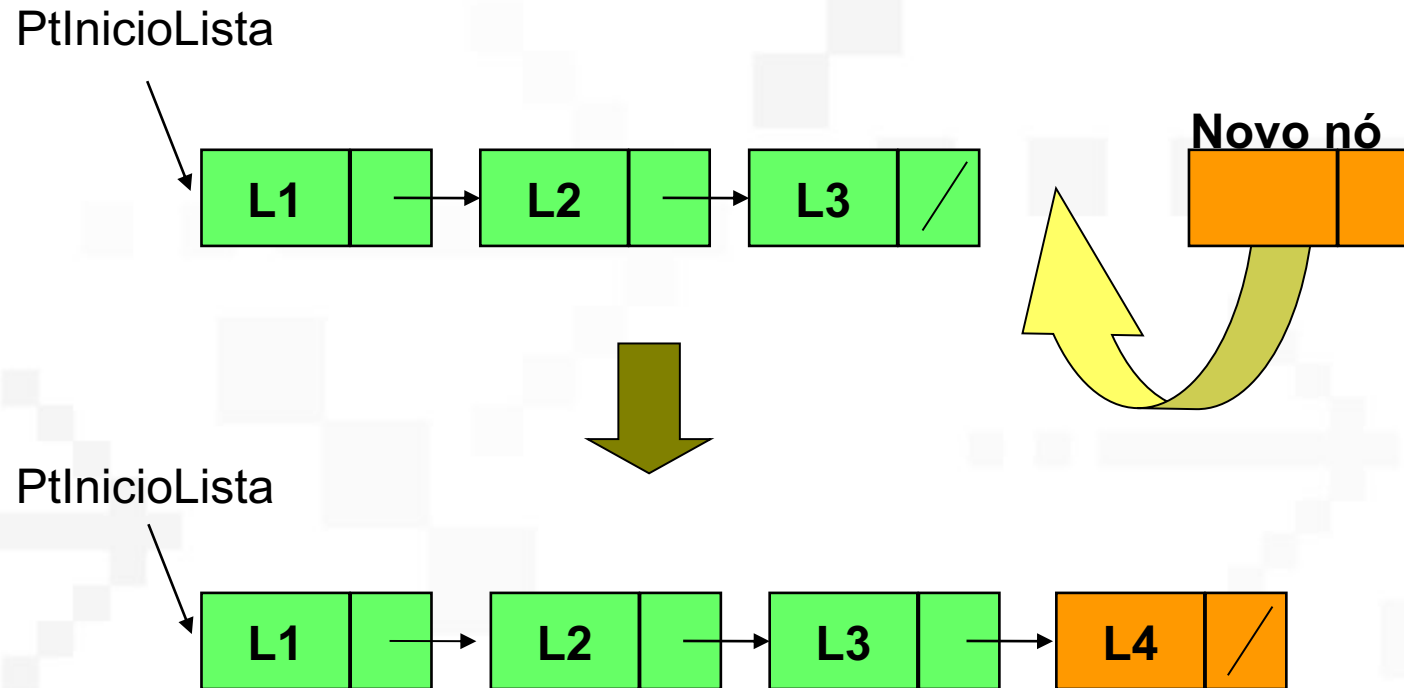


- No início da lista (primeiro nó)
- No final da lista (último nó)
- No meio da lista

Inserção no início de LLS Encadeada

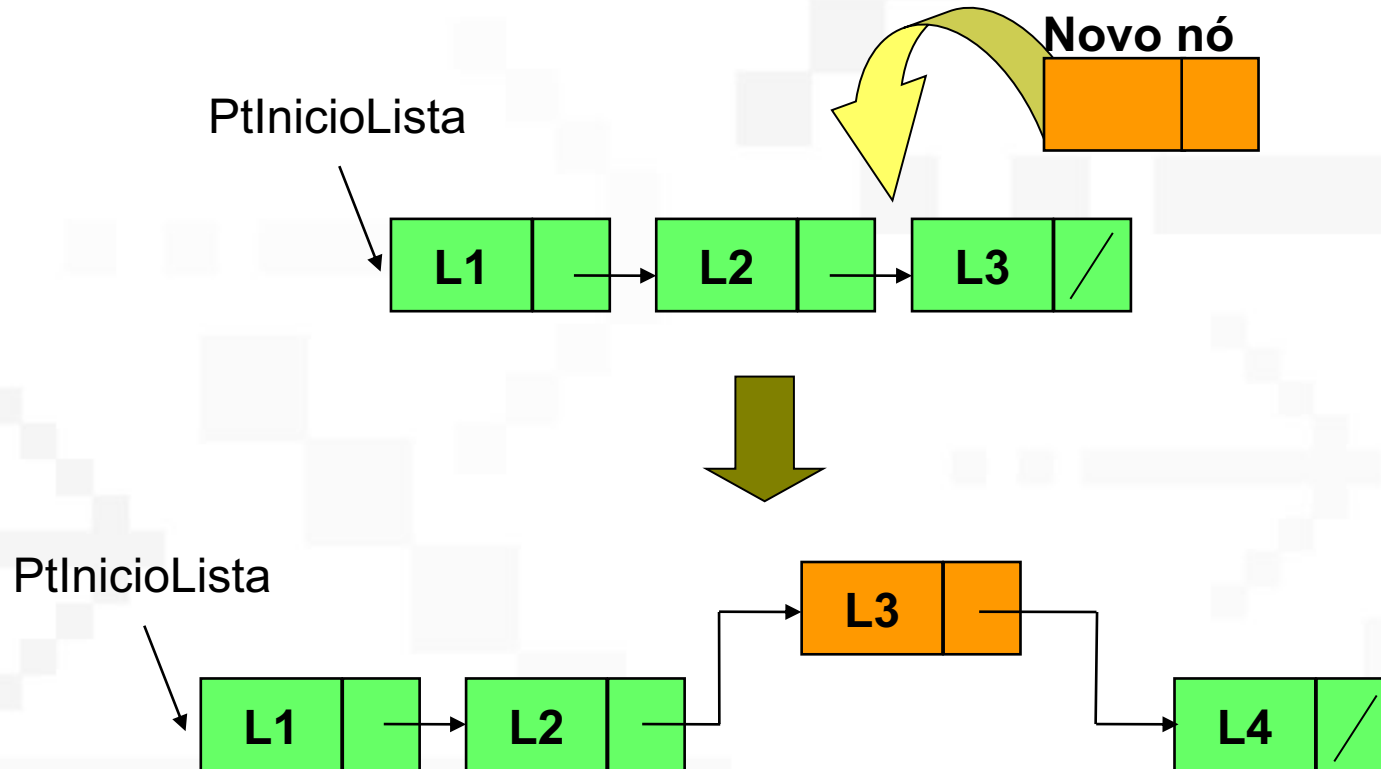


Inserção no final de LLS Encadeada



Percorrer a lista a partir do primeiro nó até o final

Inserção no meio de LLS Encadeada

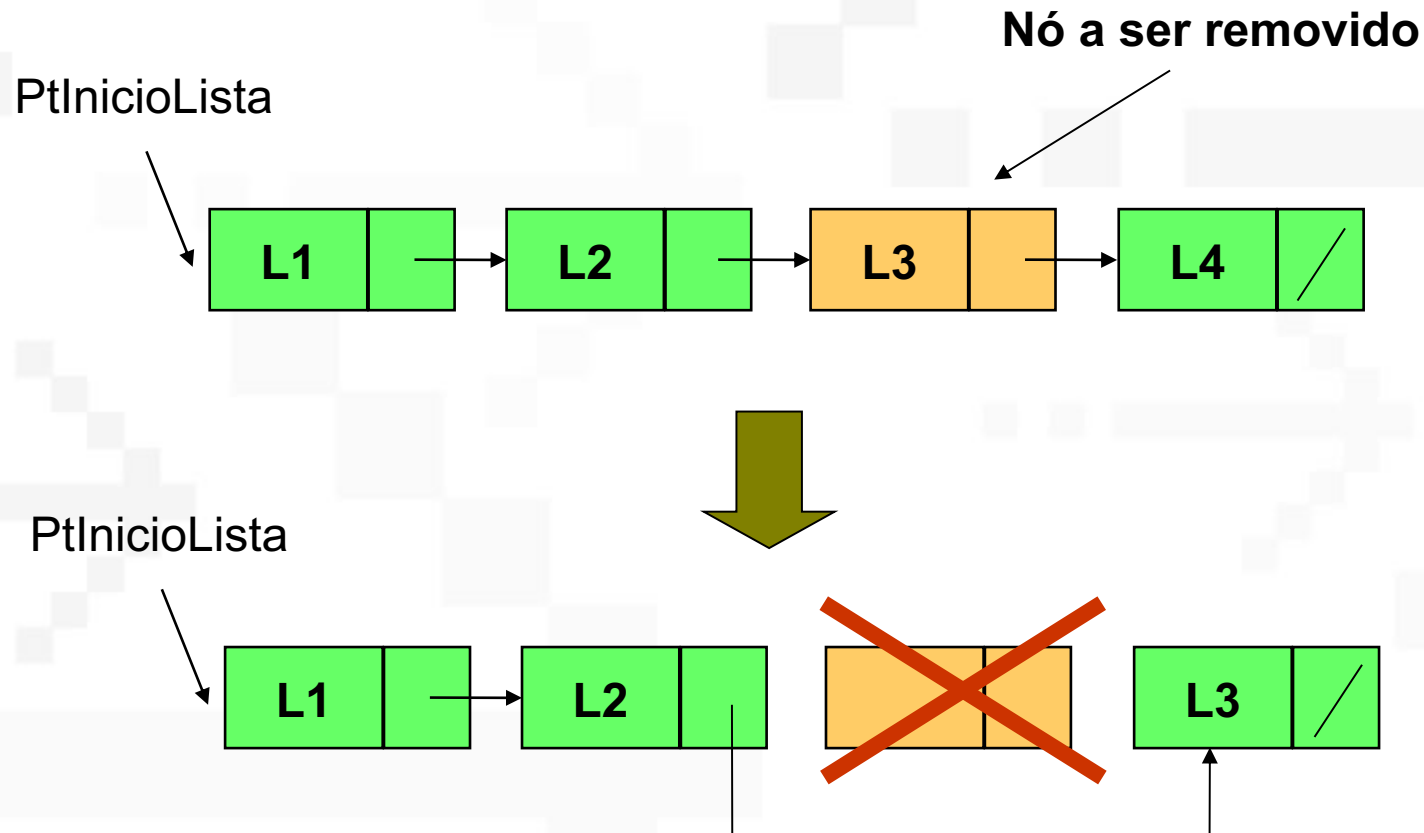


Percorrer a lista a partir do primeiro nó até a posição de inserção

Remoção de um nó

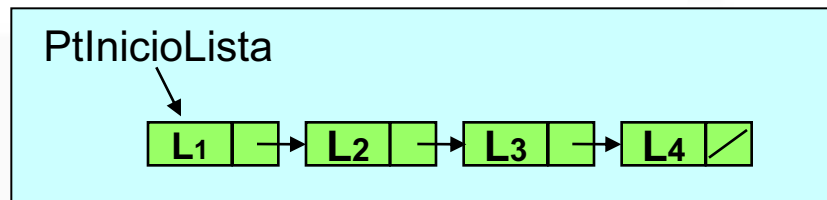
- Localizar o nó a ser removido
 - Atualizar encadeamento da lista
 - Liberar espaço ocupado pelo nó
-
- Se for o último nó da lista → lista fica vazia
 - Atualizar apontador da lista para endereço nulo

Remoção de um nó



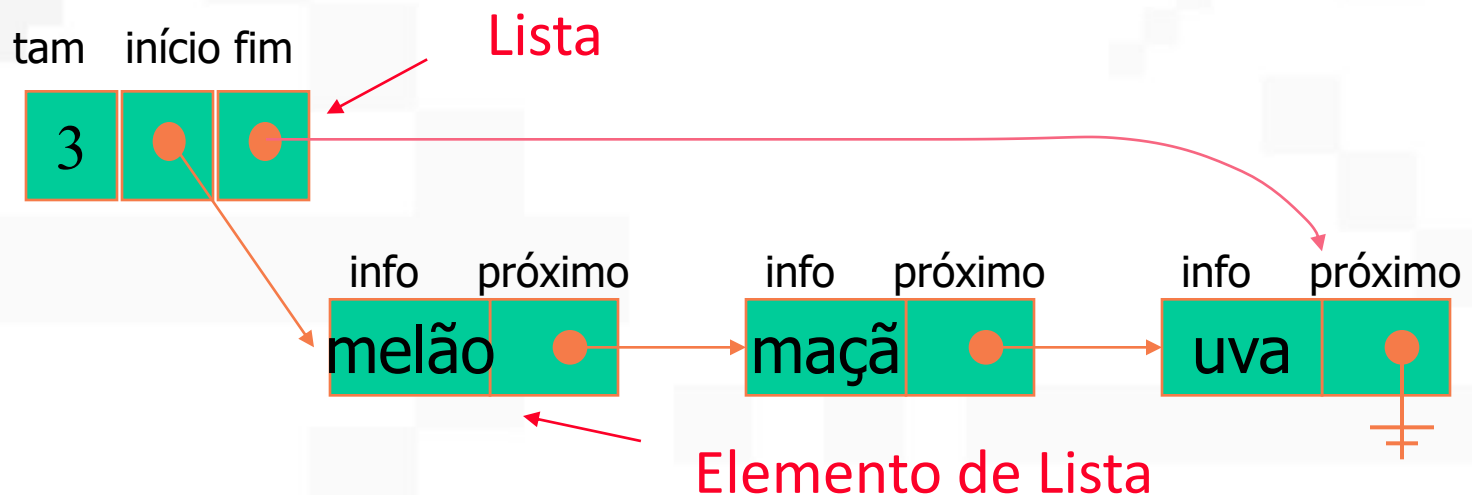
Acesso a um nó

- Os nós não podem ser acessados diretamente
- Percorrer a lista até encontrar o nó buscado
- Nó identificado por
 - sua posição na lista
 - conteúdo

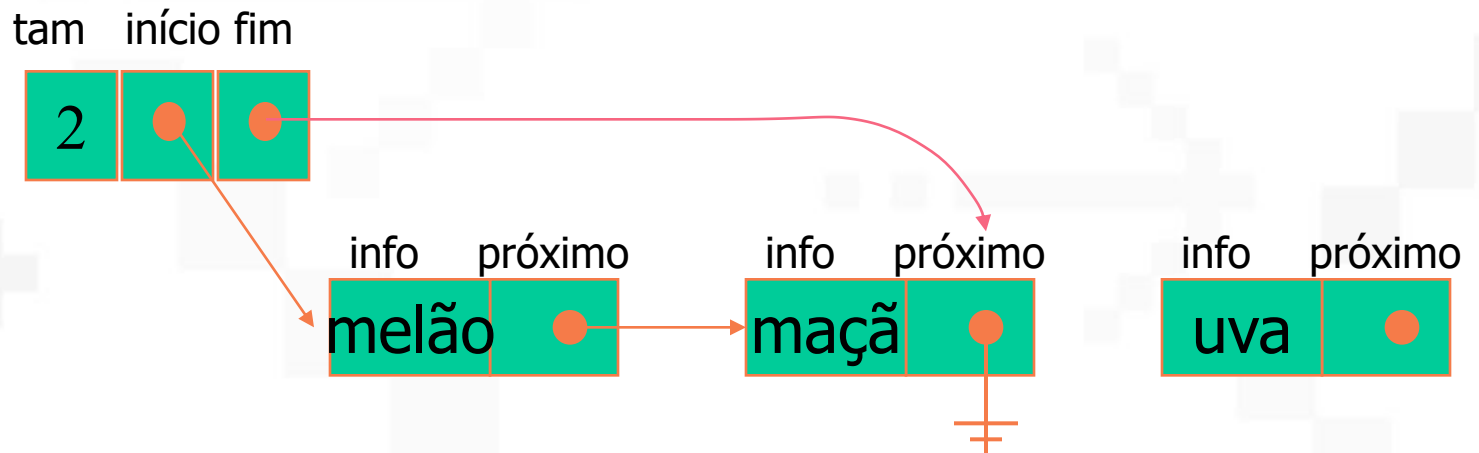


Listas: Outra representação

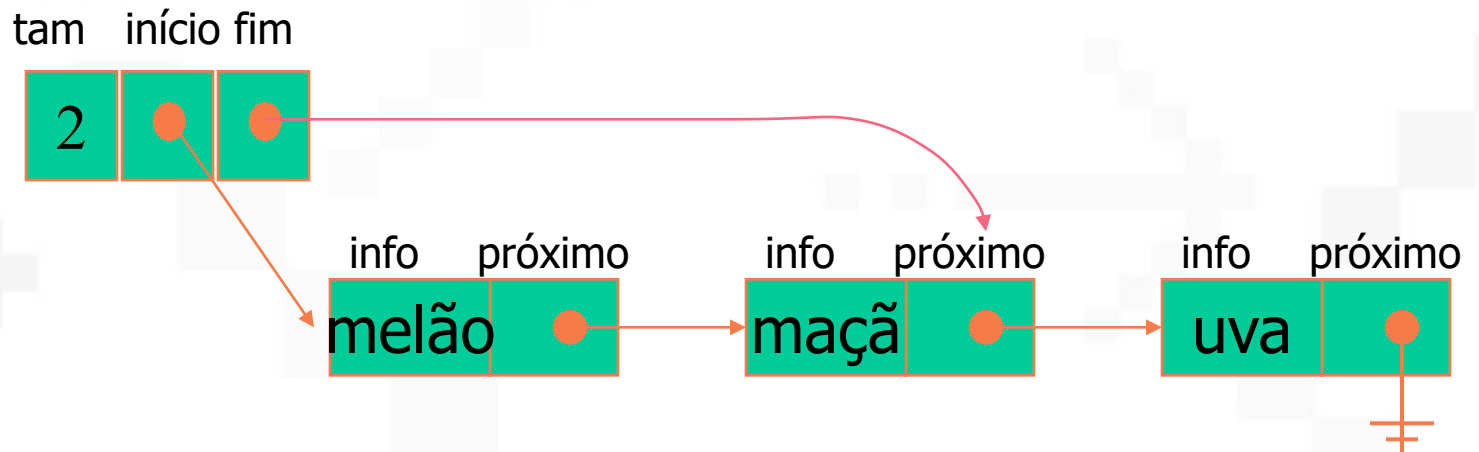
- referenciamos o último elemento também;
- adicionamos no **fim**;
- excluímos do **início**.



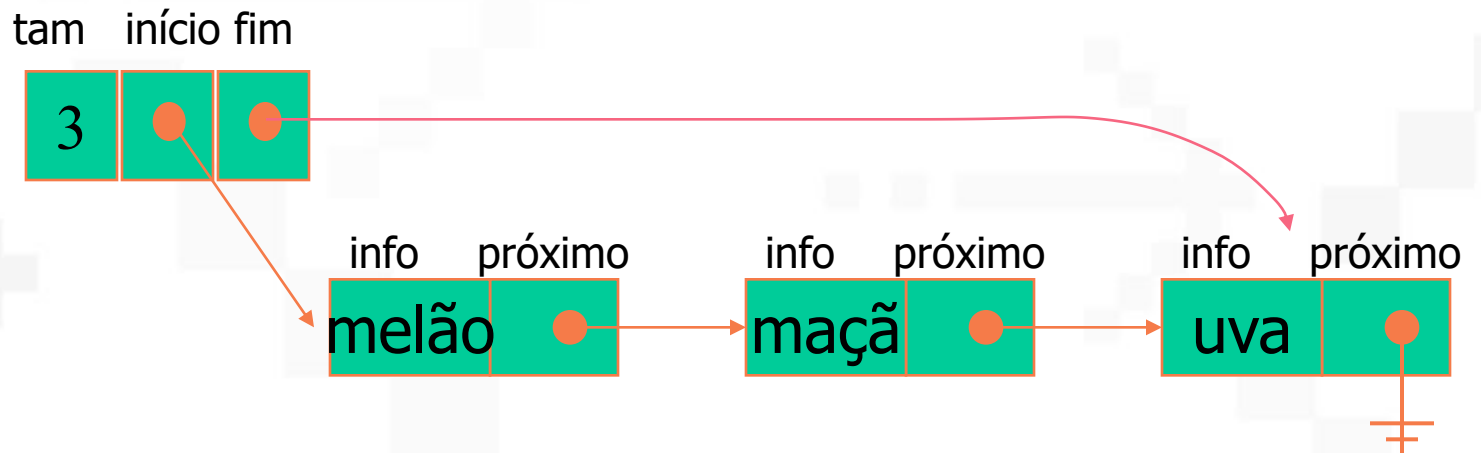
Adiciona no Fim



Adiciona no Fim

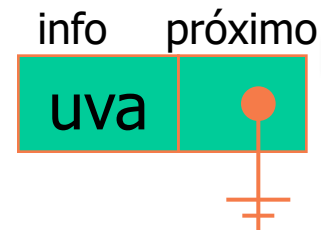
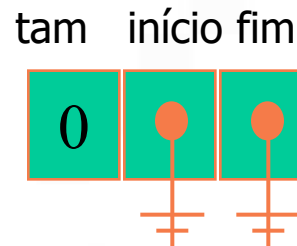


Adiciona no Fim



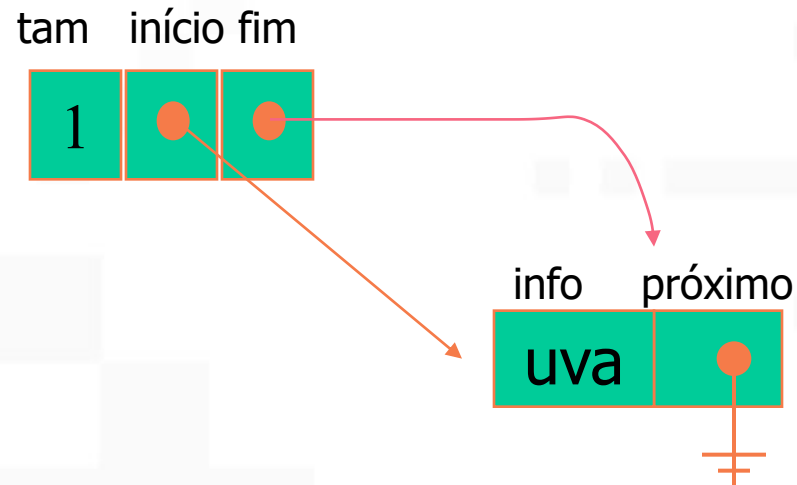
Adiciona no Fim

Caso especial: Lista vazia

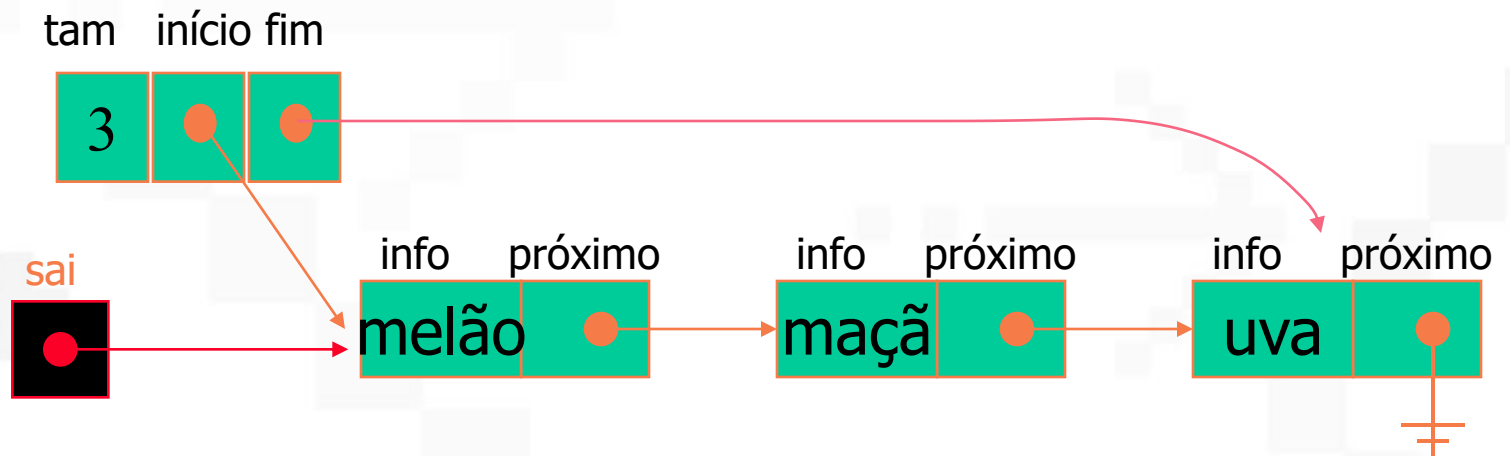


Adiciona no Fim

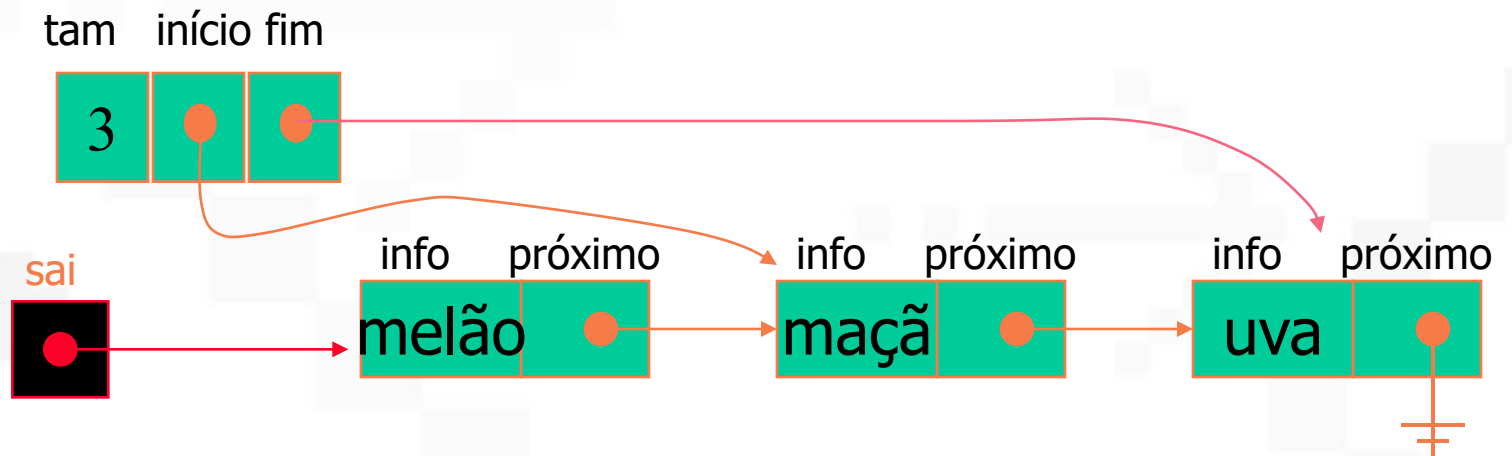
Caso especial: Lista Vazia



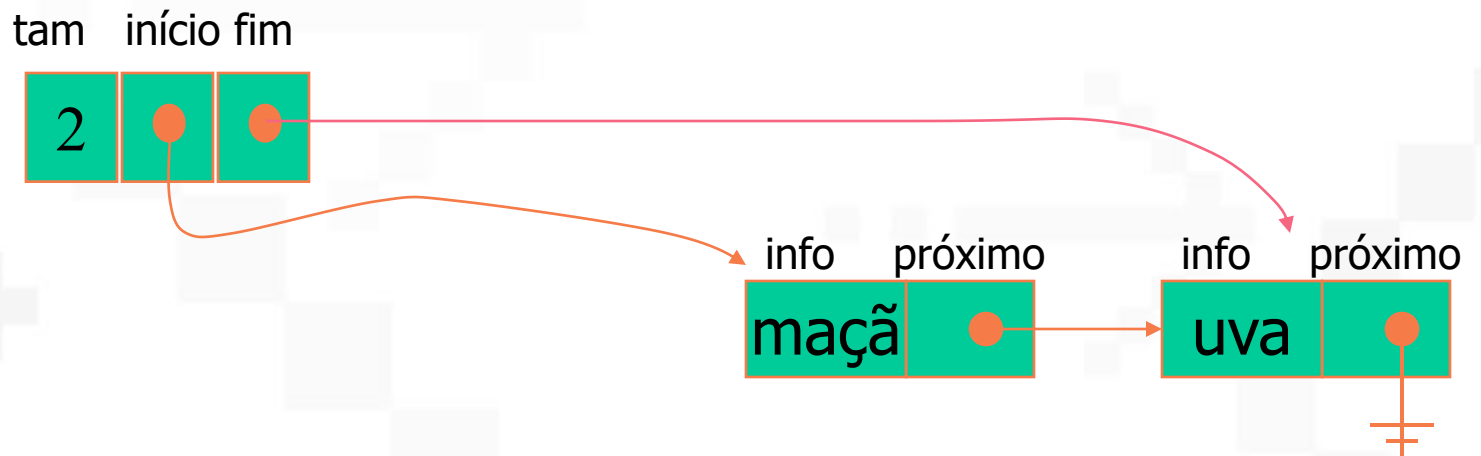
Retira no Início



Retira no Início

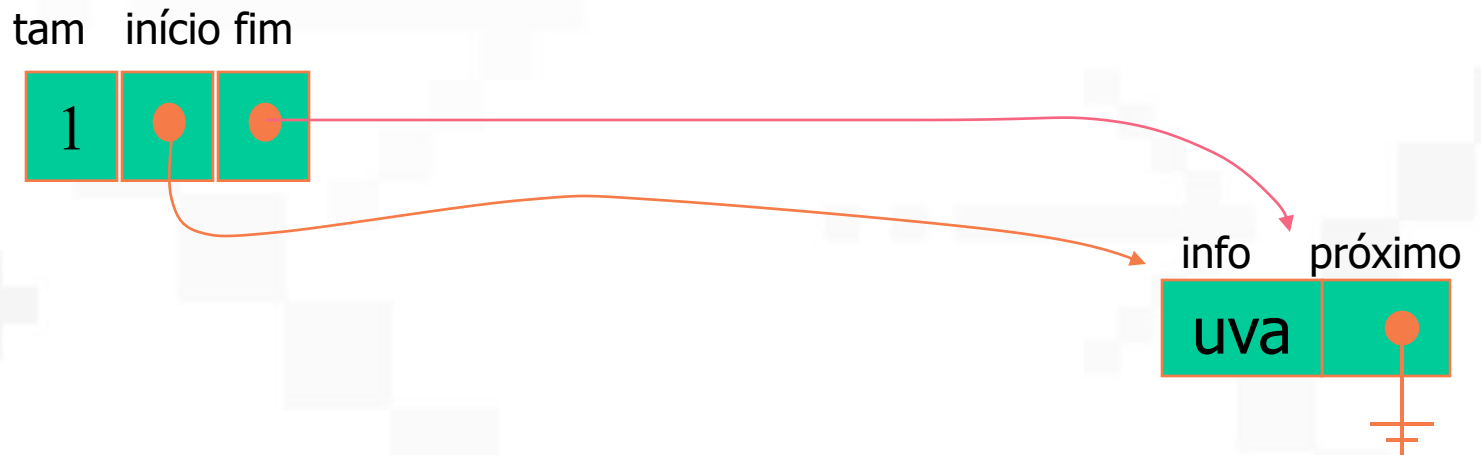


Retira no Início



Retira

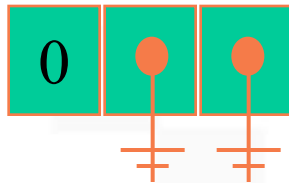
Caso especial: Lista Unitária



Retira

Caso especial: Lista Unitária

tam início fim



Exercício – Implementação de métodos da Lista

Aluno
<ul style="list-style-type: none">- nome : String- matricula : int
<ul style="list-style-type: none">+ Aluno()+ Aluno(in nome : String, in matricula : int)+ getNome() : String+ getMatricula() : int+ setNome(in nome : String) : void+ setMatricula(in matricula : int) : void

No
<ul style="list-style-type: none">- informacao : Aluno- proximo : *No
<ul style="list-style-type: none">+ No()+ No(in elemento : Aluno, in proximo : *No)+ getInformacao() : Aluno+ getProximo() : *No+ setInformacao(in informacao : Aluno) : void+ setProximo(in proximo : *No) : void

Lista
<ul style="list-style-type: none">- quantidadeDeElementos : int- inicio : *No- fim : *No
<ul style="list-style-type: none">+ LLSE()+ inserirInicio(in informacao : Aluno) : void+ inserirFim(in informacao : Aluno) : void+ inserirPosicao(in informacao : Aluno, in posicao : int) : void+ retirarInicio() : Aluno+ retirarFim() : Aluno+ retirarPosicao(in posicao : int) : Aluno+ retirarInformacao(in informacao : int) : Aluno+ acessarInformacao(in chave : int) : Aluno+ acessarPosicao(in posicao : int) : Aluno+ getQuantidadeDeElementos() : int+ estaVazia() : boolean