

# BCC202 - Estruturas de Dados I

## Aula 10: Listas (Parte I)

**Pedro Silva**

Universidade Federal de Ouro Preto, UFOP  
Departamento de Computação, DECOM  
Email: [silvap@ufop.edu.br](mailto:silvap@ufop.edu.br)



# Conteúdo

Introdução

TAD Lista

Implementação por ARRAY

Implementação por PONTEIRO

Conclusão

Exercícios

# Introdução

## Listas

- ▶ Tipo Abstrato de Dado (TAD) para representar um conjunto de dados.
- ▶ Pode ser implementada de duas formas principais:
  - a) Utilização de *vetor*.
  - b) Utilização ponteiros (**lista encadeada**).
- ▶ Dados podem ser **acessados**, **inseridos**, **localizados** ou **retirados**.

# Listas

## Outras Operações

- ▶ **Concatenação** de duas ou mais listas.
- ▶ **Partição** em duas ou mais listas.

## Listas: Definição

- ▶ **Sequência de zero ou mais itens.**
  - ▶  $x_0, x_1, x_2, \dots, x_{n-1}$ , na qual  $x_i$  é de um determinado tipo e  $n$  representa o tamanho da lista.
- ▶ **Sua principal propriedade estrutural envolve as posições relativas dos itens em uma dimensão.**
  - ▶ Assumindo  $n \geq 0$ ,  $x_0$  é o **primeiro** item da lista e  $x_{n-1}$  é o **último** item da lista.
    - ▶  $x_i$  **precede**  $x_{i+1}$  para  $i = 0, 1, 2, \dots, n-2$ .
    - ▶  $x_i$  **suced**e  $x_{i-1}$  para  $i = 1, 2, 3, \dots, n-1$ .
    - ▶ O elemento  $x_i$  é dito estar na  $i$ -ésima posição da lista.
    - ▶ Isto não implica que os dados estejam ordenados, estes conceitos remetem apenas à **posição** na lista.
  - ▶ A lista é **linear**.

# TAD Lista

## TAD Lista

O que o TAD Lista deveria conter?



## TAD Lista

### O que o TAD Lista deveria conter?

- ▶ Representação do **tipo** da lista.
- ▶ Conjunto de **operações** que atuam sobre a lista.

### Quais operações deveriam fazer parte da lista?

## TAD Lista

### O que o TAD Lista deveria conter?

- ▶ Representação do **tipo** da lista.
- ▶ Conjunto de **operações** que atuam sobre a lista.

### Quais operações deveriam fazer parte da lista?

- ▶ **Depende de cada aplicação.**
- ▶ Mas, um conjunto *padrão* pode ser definido.

## Conjunto Padrão de operações

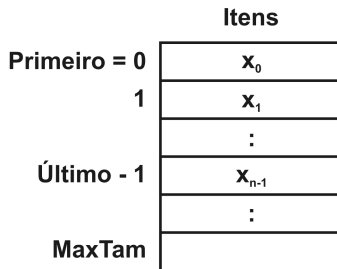
- ▶ Operações necessárias à grande maioria das aplicações:
  - ▶ **Criar** uma lista linear vazia.
  - ▶ **Inserir** um novo item.
  - ▶ **Retirar** um item.
  - ▶ **Localizar** um item para examinar ou alterar seu conteúdo.
  - ▶ **Combinar** duas ou mais listas em uma única lista.
  - ▶ **Dividir** uma lista em duas ou mais listas.
  - ▶ Fazer uma **cópia** da lista.
  - ▶ **Ordenar** os itens da lista de acordo com um de seus campos.
  - ▶ **Pesquisar** a ocorrência de um item com um valor específico.

## Implementação de uma Lista

- ▶ Várias estruturas de dados podem ser usadas para representar listas lineares, cada uma com vantagens e desvantagens particulares.
- ▶ As duas representações mais utilizadas são:
  - ▶ Implementação por **arrays**.
  - ▶ Implementação por **ponteiros**.

## Implementação por ARRAY

## TAD Lista: Implementação por ARRAY



- ▶ Os itens são armazenados em posições contíguas de memória.
- ▶ A lista pode ser percorrida em qualquer direção.

## TAD Lista: Implementação por ARRAY

Itens	
Primeiro = 0	$x_0$
1	$x_1$
	:
Último - 1	$x_{n-1}$
	:
MaxTam	

- ▶ A **inserção** de um novo item no **final** tem
- ▶ A **inserção** de um novo item no meio tem
- ▶ A **remoção** de um item no meio tem

## TAD Lista: Implementação por ARRAY

Itens	
Primeiro = 0	$x_0$
1	$x_1$
	:
Último - 1	$x_{n-1}$
	:
MaxTam	

- ▶ A **inserção** de um novo item no **final** tem **Custo**  $O(1)$ .
- ▶ A **inserção** de um novo item no meio tem
- ▶ A **remoção** de um item no meio tem



## TAD Lista: Implementação por ARRAY

Itens	
Primeiro = 0	$x_0$
1	$x_1$
	:
Último - 1	$x_{n-1}$
	:
MaxTam	

- ▶ A **inserção** de um novo item no **final** tem **Custo**  $O(1)$ .
- ▶ A **inserção** de um novo item no meio tem **Custo**  $O(n)$ .
- ▶ A **remoção** de um item no meio tem

## TAD Lista: Implementação por ARRAY

Itens	
Primeiro = 0	$x_0$
1	$x_1$
	:
Último - 1	$x_{n-1}$
	:
MaxTam	

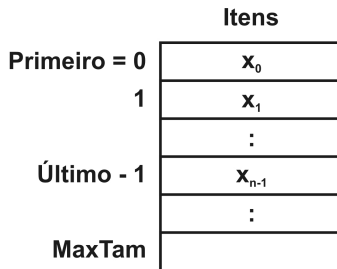
- ▶ A **inserção** de um novo item no **final** tem **Custo**  $O(1)$ .
- ▶ A **inserção** de um novo item no meio tem **Custo**  $O(n)$ .
- ▶ A **remoção** de um item no meio tem **Custo**  $O(n)$ .

## TAD Lista: Implementação por ARRAY

Itens	
Primeiro = 0	$x_0$
1	$x_1$
	:
Último - 1	$x_{n-1}$
	:
MaxTam	

- ▶ A **inserção** de um novo item no **final** tem **Custo**  $O(1)$ .
- ▶ A **inserção** de um novo item no meio tem **Custo**  $O(n)$ .
- ▶ A **remoção** de um item no meio tem **Custo**  $O(n)$ .
- ▶ Os dois últimos requerem deslocamentos.

## TAD Lista: Implementação por ARRAY



- ▶ Os itens são armazenados em um array de tamanho suficiente para comportar todos os elementos da lista.
- ▶ O campo **Ultimo** aponta para a **posição seguinte** à do último elemento da lista.

## TAD Lista: Implementação por ARRAY

Itens	
Primeiro = 0	$x_0$
1	$x_1$
	:
Último - 1	$x_{n-1}$
	:
MaxTam	

- ▶ O  $i$ -ésimo item da lista está armazenado na  $(i-1)$ -ésima posição do array,  $0 \leq i < \text{Último}$ .
- ▶ A constante MaxTam define o tamanho máximo permitido para a lista.

## TAD Lista: Implementação por ARRAY: Código Exemplo (.h)

```
1 |
2 | #define INICIO 0
3 | #define MAXTAM 1000
4 |
5 | typedef struct{
6 |     int chave; /* outros campos */
7 | } Item;
8 |
9 | typedef struct{
10 |     Item item[MAXTAM];
11 |     int primeiro, ultimo;
12 | } Lista;
13 | ...
```

## TAD Lista: Implementação por ARRAY: Código Exemplo (.h)

```
14  ...
15  /* procedimentos e funcoes do TAD */
16  void ListaInicia(Lista *pLista);
17
18  int ListaEhVazia(Lista *pLista);
19
20  int ListaInsere(Lista *pLista, Item x);
21
22  int ListaRetira(Lista *pLista, int p, Item *pX);
23
24  void ListaImprime(Lista *pLista);
25
26  int ListaGet(Lista *pLista, int p, Item *pX);
27
28  int ListaTamanho(Lista *pLista);
```

## TAD Lista: Implementação por ARRAY: Código Exemplo (.c)

```
1 #include "lista_vetor.h"
2
3 void ListaInicia(Lista *pLista) {
4     pLista->primeiro = INICIO;
5     pLista->ultimo = pLista->primeiro;
6 }
7
8 int ListaEhVazia(Lista *pLista) {
9     return (pLista->ultimo == pLista->primeiro);
10 }
11
12 int ListaInsere(Lista *pLista, Item x) {
13     if (pLista->ultimo == MAXTAM)
14         return 0; /* lista cheia */
15     pLista->item[pLista->ultimo++] = x;
16     return 1;
17 }
18
19 ...
```



## TAD Lista: Implementação por ARRAY: Código Exemplo (.c)

```
21 ...
22
23 int ListaRetira(Lista *pLista, int p, Item *pX) {
24     if (ListaEhVazia(pLista) || p >= pLista->ultimo)
25         return 0;
26
27     int cont;
28     *pX = pLista->Item[p];
29     pLista->ultimo--;
30     for (cont = p+1; cont <= pLista->ultimo; cont++)
31         pLista->item[cont - 1] = pLista->item[cont];
32     return 1;
33 }
34
35 void ListaImprime(Lista *pLista) {
36     int i;
37     for (i = pLista->primeiro; i < pLista->ultimo; i++)
38         printf("%d\n", pLista->item[i].chave);
39 }
```

## TAD Lista: Implementação por ARRAY

### Vantagens

- ▶ Economia de memória (apontadores implícitos).
- ▶ Acesso a qualquer elemento da lista é feito em tempo  $O(1)$ .

### Desvantagens

- ▶ Custo para inserir itens da lista pode ser  $O(n)$ .
- ▶ Custo para retirar itens da lista pode ser  $O(n)$ .
- ▶ Quando não existe previsão sobre o crescimento da lista, o array que define a lista deve ser alocado de forma dinâmica.
- ▶ Pode ser necessária a realocação do array.

## Implementação por PONTEIRO

## TAD Lista: Implementação por PONTEIRO

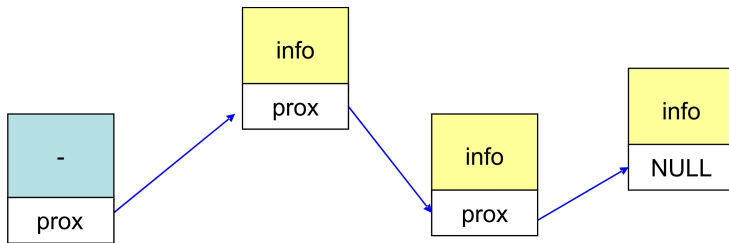
- ▶ Qual o principal problema de utilizar array para implementar listas?
- ▶ E se a lista aumentar e depois diminuir drasticamente de tamanho?
- ▶ **Solução:**

## TAD Lista: Implementação por PONTEIRO

- ▶ Qual o principal problema de utilizar array para implementar listas?
- ▶ E se a lista aumentar e depois diminuir drasticamente de tamanho?
- ▶ **Solução: Listas Encadeadas!!!.**
- ▶ O que é?
  - ▶ Implementação de uma lista utilizando apenas **ponteiros**.

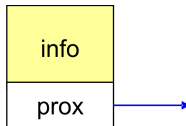
## TAD Lista: Lista Encadeada

- ▶ Características:
  - ▶ Tamanho da lista não é pré-definido.
  - ▶ Cada elemento aponta para o próximo.
  - ▶ Elementos não estão contíguos na memória (a alocação é dinâmica).



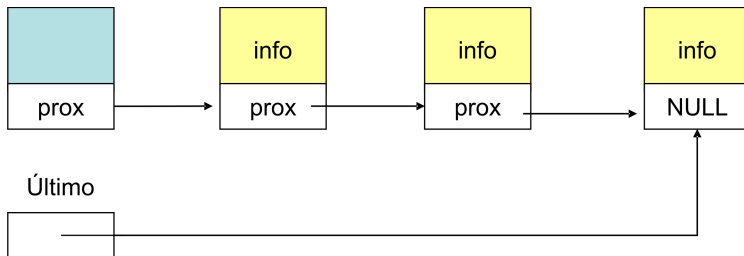
## TAD Lista: Lista Encadeada: Elementos

- ▶ **Elemento da lista:**
  - ▶ Armazena as informações sobre cada elemento.
  - ▶ Aponta para o próximo elemento.
- ▶ Assim, é definido como uma estrutura que possui:
  - ▶ Campos representando as informações do elemento.
  - ▶ Ponteiro para o próximo elemento.



## TAD Lista: Lista Encadeada: Elementos

- ▶ Uma lista pode ter uma célula **cabeça**, antecedendo o primeiro elemento.
- ▶ Pode possuir também um apontador para o **último** elemento.

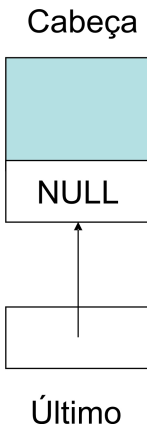




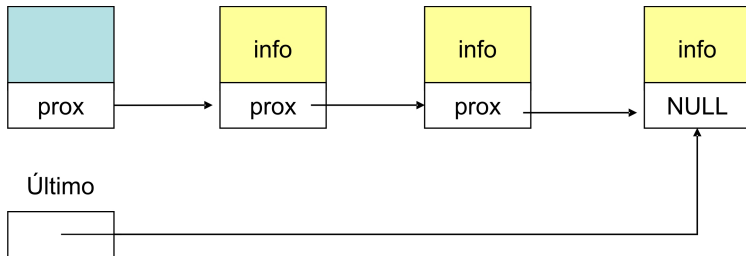
## Lista Encadeada COM cabeça: Estruturas básicas

```
1 typedef struct {  
2     /* Componentes de um item: "info" */  
3 } Item;  
4  
5 typedef struct cel {  
6     struct cel *pProx;  
7     Item item;  
8 } Celula;  
9  
10 typedef struct {  
11     Celula *pPrimeiro, *pUltimo;  
12 } Lista;
```

## TAD Lista: Lista Encadeada: Criar Lista Vazia

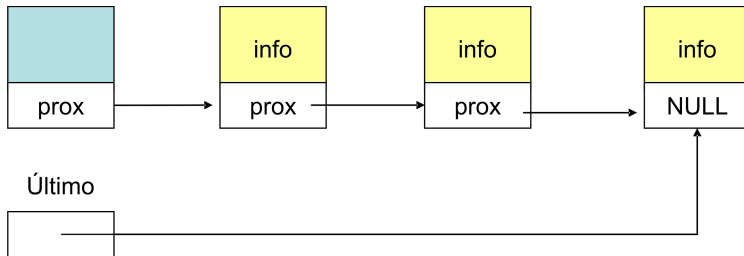


## TAD Lista: Lista Encadeada: INSERÇÃO de Novos Elementos



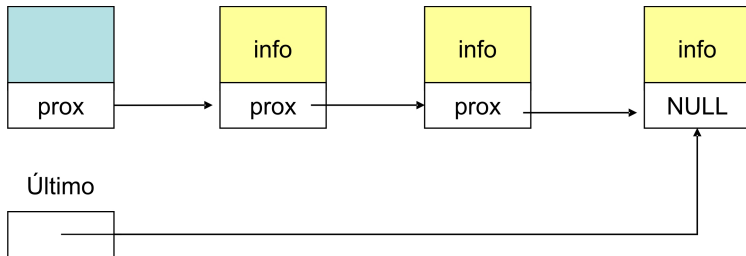
- 3 opções de posição onde se pode **inserir**:

## TAD Lista: Lista Encadeada: INSERÇÃO de Novos Elementos



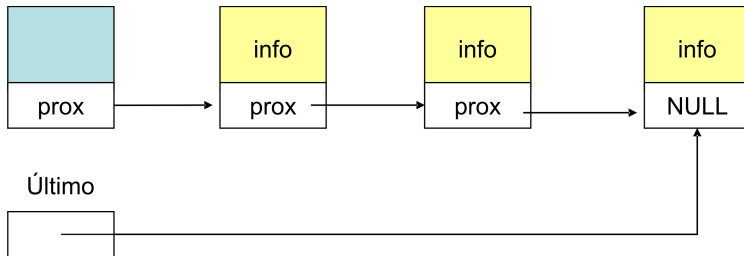
- ▶ 3 opções de posição onde se pode **inserir**:
  - ▶ Primeira posição.

## TAD Lista: Lista Encadeada: INSERÇÃO de Novos Elementos



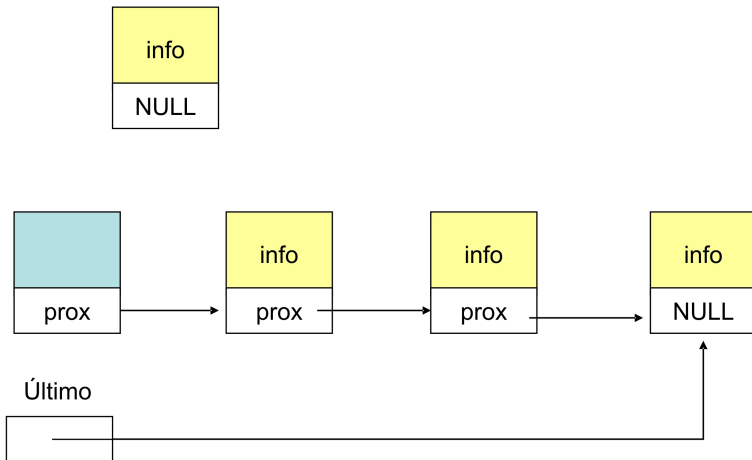
- ▶ 3 opções de posição onde se pode **inserir**:
  - ▶ Primeira posição.
  - ▶ Última posição.

## TAD Lista: Lista Encadeada: INSERÇÃO de Novos Elementos

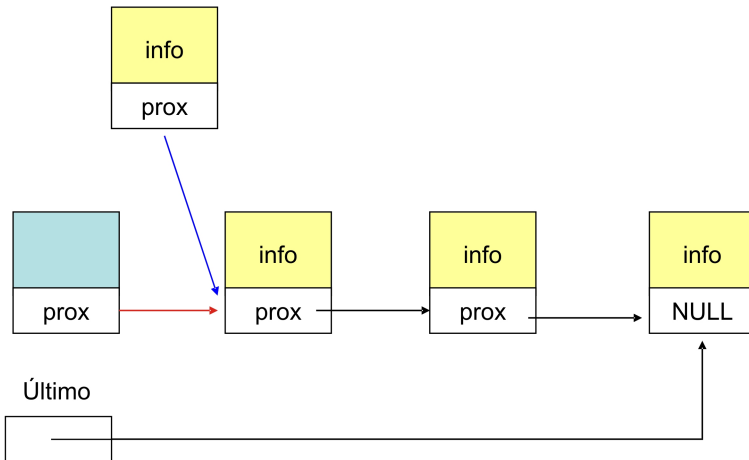


- ▶ 3 opções de posição onde se pode **inserir**:
  - ▶ Primeira posição.
  - ▶ Última posição.
  - ▶ Após um elemento E.

## TAD Lista: Lista Encadeada: INSERÇÃO na Primeira Posição

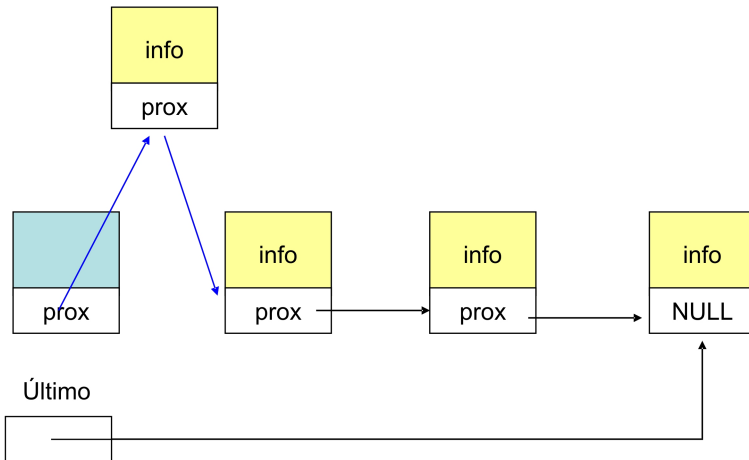


## TAD Lista: Lista Encadeada: INSERÇÃO na Primeira Posição

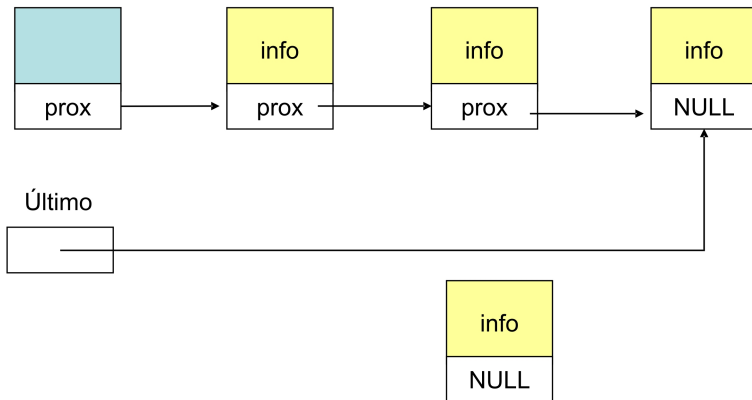




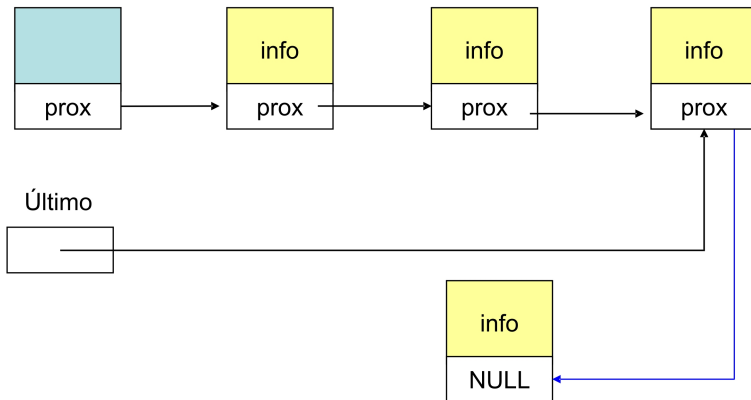
## TAD Lista: Lista Encadeada: INSERÇÃO na Primeira Posição



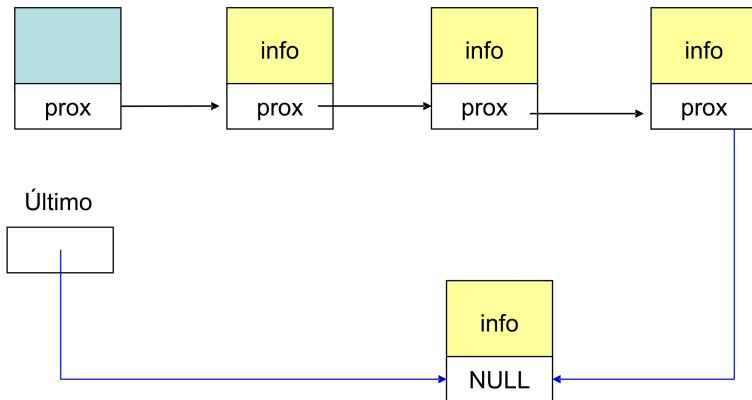
## TAD Lista: Lista Encadeada: INSERÇÃO na Última Posição



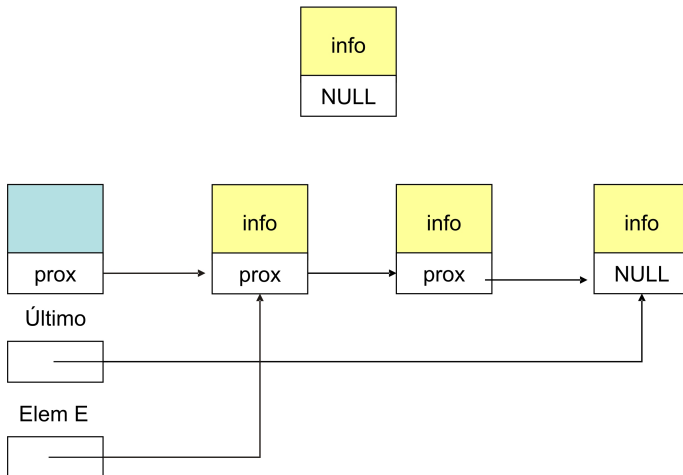
## TAD Lista: Lista Encadeada: INSERÇÃO na Última Posição



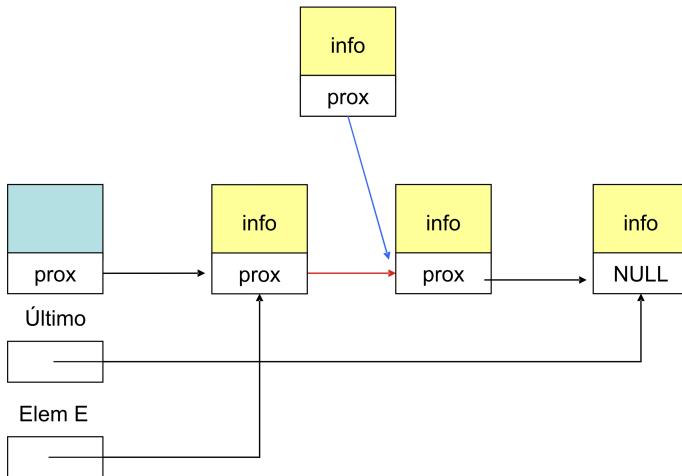
## TAD Lista: Lista Encadeada: INSERÇÃO na Última Posição



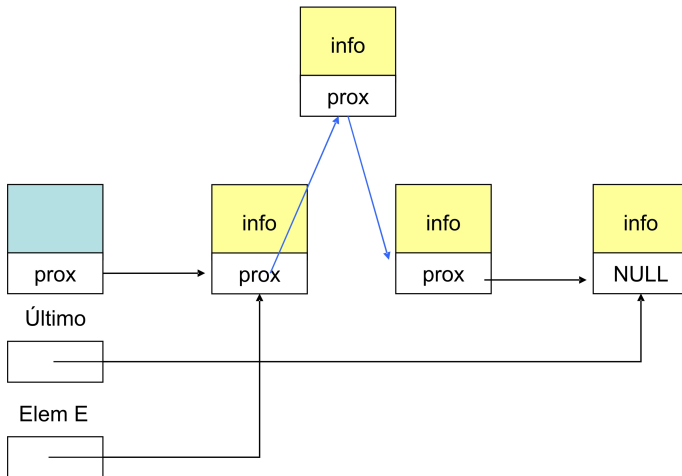
## TAD Lista: Lista Encadeada: INSERÇÃO Após o Elemento E



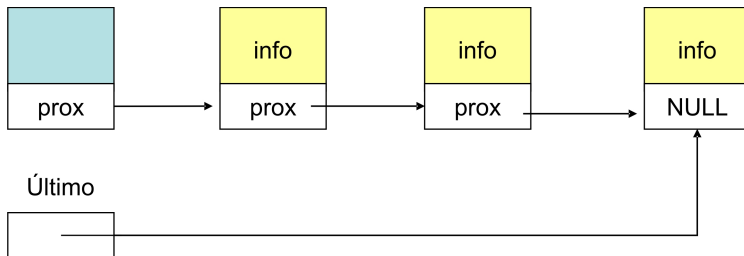
## TAD Lista: Lista Encadeada: INSERÇÃO Após o Elemento E



## TAD Lista: Lista Encadeada: INSERÇÃO Após o Elemento E



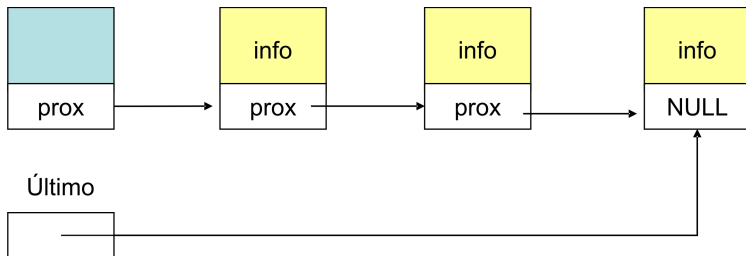
## TAD Lista: Lista Encadeada: RETIRADA de Elementos



- ▶ 3 opções de posição onde se pode **retirar**:

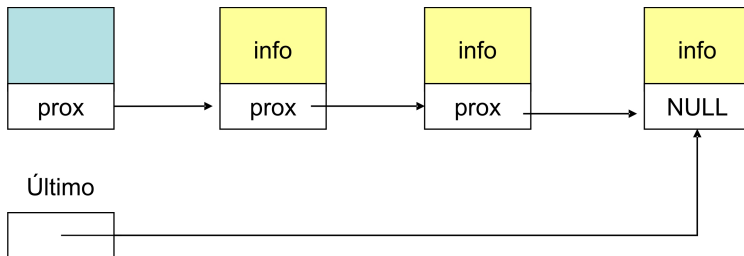


## TAD Lista: Lista Encadeada: RETIRADA de Elementos



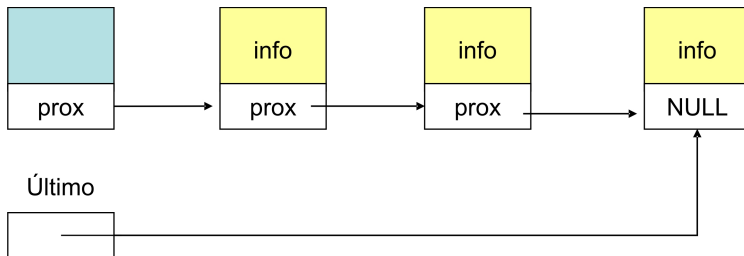
- ▶ 3 opções de posição onde se pode **retirar**:
  - ▶ Primeira posição.

## TAD Lista: Lista Encadeada: RETIRADA de Elementos



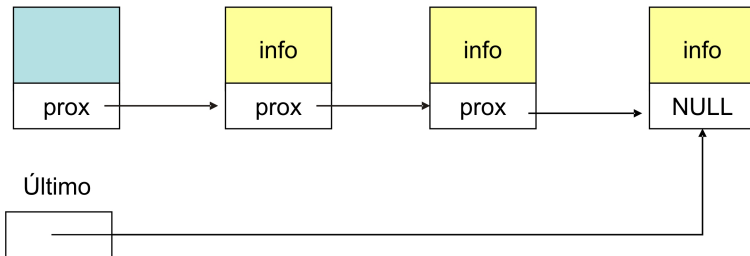
- ▶ 3 opções de posição onde se pode **retirar**:
  - ▶ Primeira posição.
  - ▶ Última posição.

## TAD Lista: Lista Encadeada: RETIRADA de Elementos

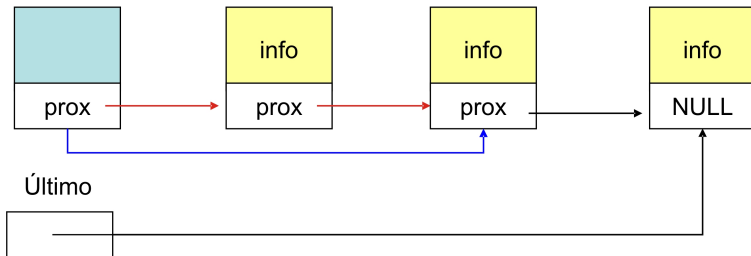


- ▶ 3 opções de posição onde se pode **retirar**:
  - ▶ Primeira posição.
  - ▶ Última posição.
  - ▶ Um elemento E.

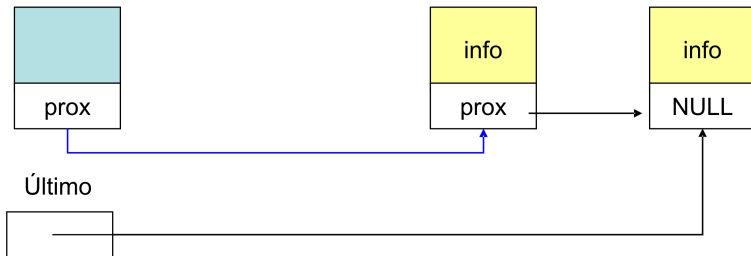
## TAD Lista: Lista Encadeada: RETIRADA na Primeira Posição



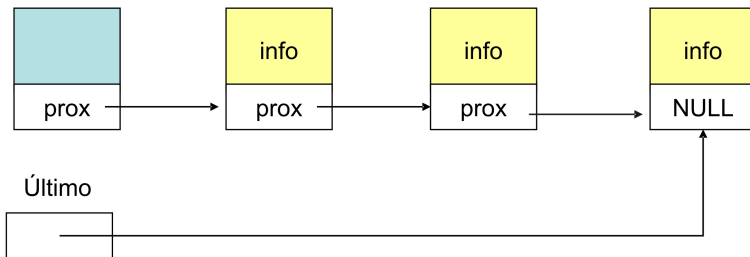
## TAD Lista: Lista Encadeada: RETIRADA na Primeira Posição



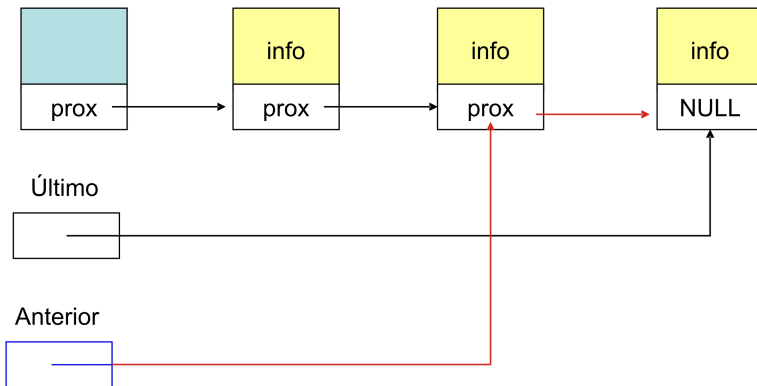
## TAD Lista: Lista Encadeada: RETIRADA na Primeira Posição



## TAD Lista: Lista Encadeada: RETIRADA na Última Posição

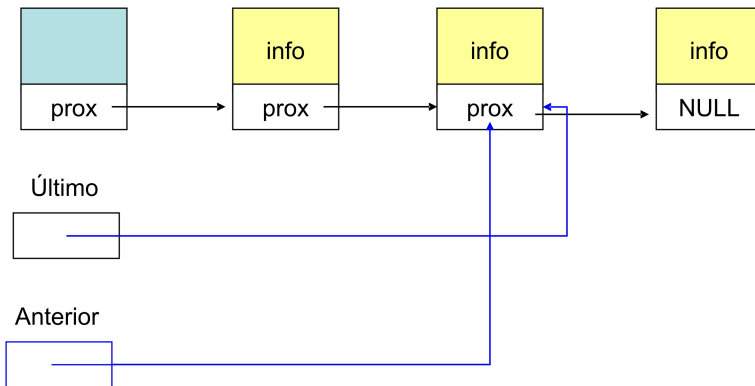


## TAD Lista: Lista Encadeada: RETIRADA na Última Posição

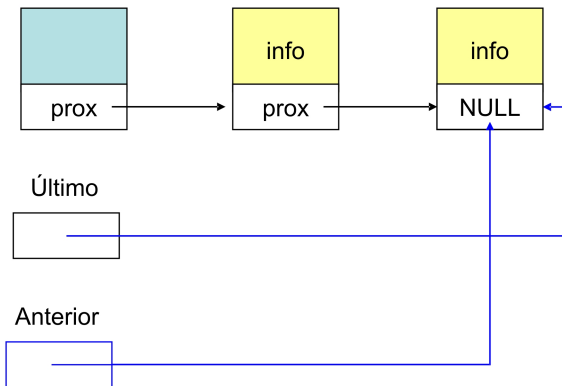




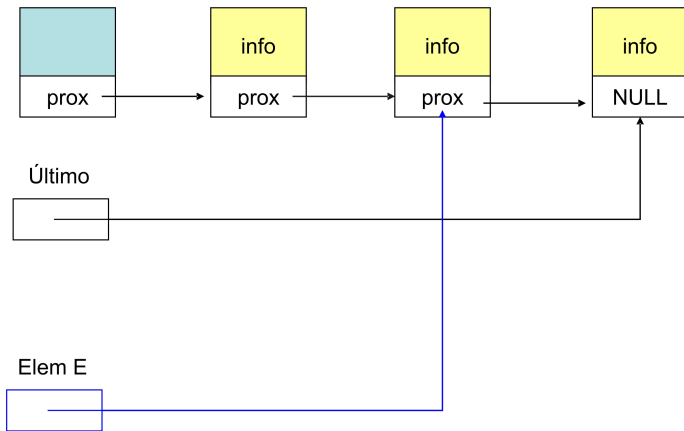
## TAD Lista: Lista Encadeada: RETIRADA na Última Posição



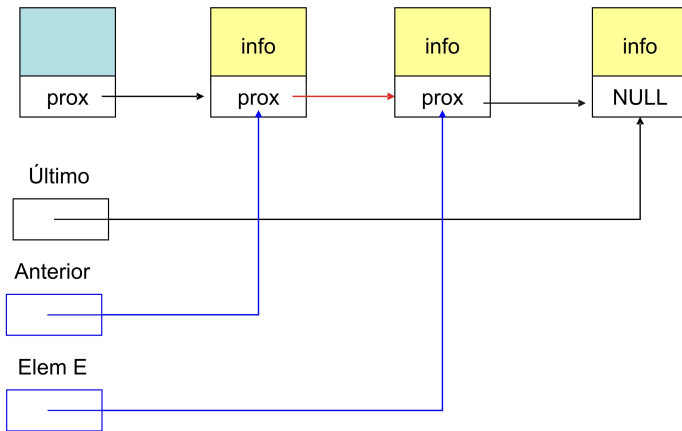
## TAD Lista: Lista Encadeada: RETIRADA na Última Posição



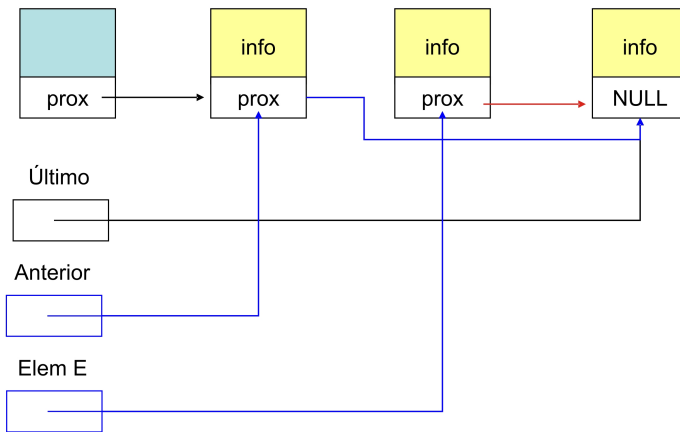
## TAD Lista: Lista Encadeada: RETIRADA do Elemento E



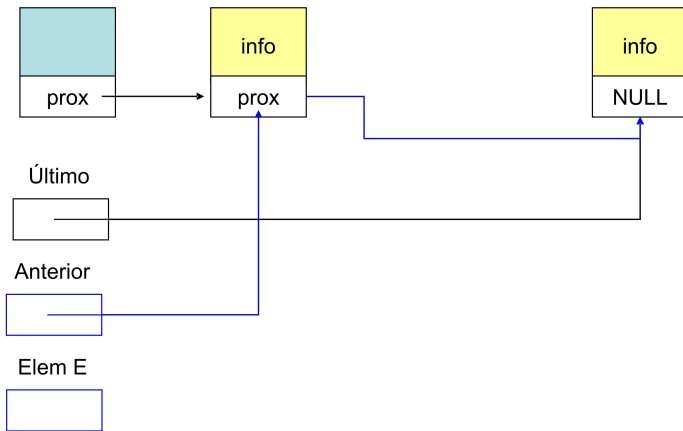
## TAD Lista: Lista Encadeada: RETIRADA do Elemento E



## TAD Lista: Lista Encadeada: RETIRADA do Elemento E



## TAD Lista: Lista Encadeada: RETIRADA do Elemento E



## Conclusão

## Conclusão

- ▶ Nesta aula tivemos o primeiro contato com a estrutura de dados **Lista**.
- ▶ Este tópico é muito importante para o entendimento das estruturas de dados que virão a seguir.

### Listas: Vetor vs Ponteiros

- ▶ Vantagens.
- ▶ Desvantagens.

### Próxima aula

Lista (Parte 2) – Implementação de lista encadeada (por ponteiro).



## Exercícios

## Exercício 01

Implemente um TAD Lista utilizando Array, acrescentando as seguintes operações ao conjunto de operações vistos nesta aula:

- ▶ Concatenação de duas listas.
- ▶ Divisão de uma lista em duas em uma posição  $i$ .
- ▶ Cópia de uma lista.
- ▶ Pesquisa por um elemento na lista.