# Algoritmos e Estrutura de Dados

Aula 7 – Estrutura de Dados: Listas

Prof. Tiago A. E. Ferreira

# Introdução

- Um das formas mais usadas para se manter dados agrupados é a **lista**
  - ***Lista de compras, itens de estoque, notas de alunos, informações de funcionários, etc.***
- Lista Linear agrupa informações referentes a um conjunto de elementos que, de alguma forma, se relacionam entre si

# Definição de Uma Lista

- É uma coleção **L:[a1, a2, ..., an]**, **n≥0**, cuja propriedade estrutural baseia-se apenas na posição relativa dos elementos, que são dispostos linearmente.
  - Se n = 0, a lista L é **vazia**.
  - Caso contrário:
    - a1 é o primeiro elemento de L;
    - $a_n$ é o último elemento de L;
    - $a_k$, 1<k<n, é precedido pelo elemento $a_{k-1}$ e seguido por $a_{k+1}$ em L

# Operação Sobre uma Lista

- Operações comuns
  - Pesquisa, inserção, alteração e remoção de um determinado elemento da lista
- Outras operações:
  - Determinação do número total de elementos da lista;
  - Ordenamento da lista;
  - União de duas ou mais listas;
  - Particionamento da lista e sub-listas;
  - etc...

# Casos Especiais de Listas

- No caso de se considerar apenas as operações de acesso, inserção e remoção, restritas aos extremos da lista, tem-se casos especiais de listas:
  - Pilha
  - Fila
  - Fila dupla

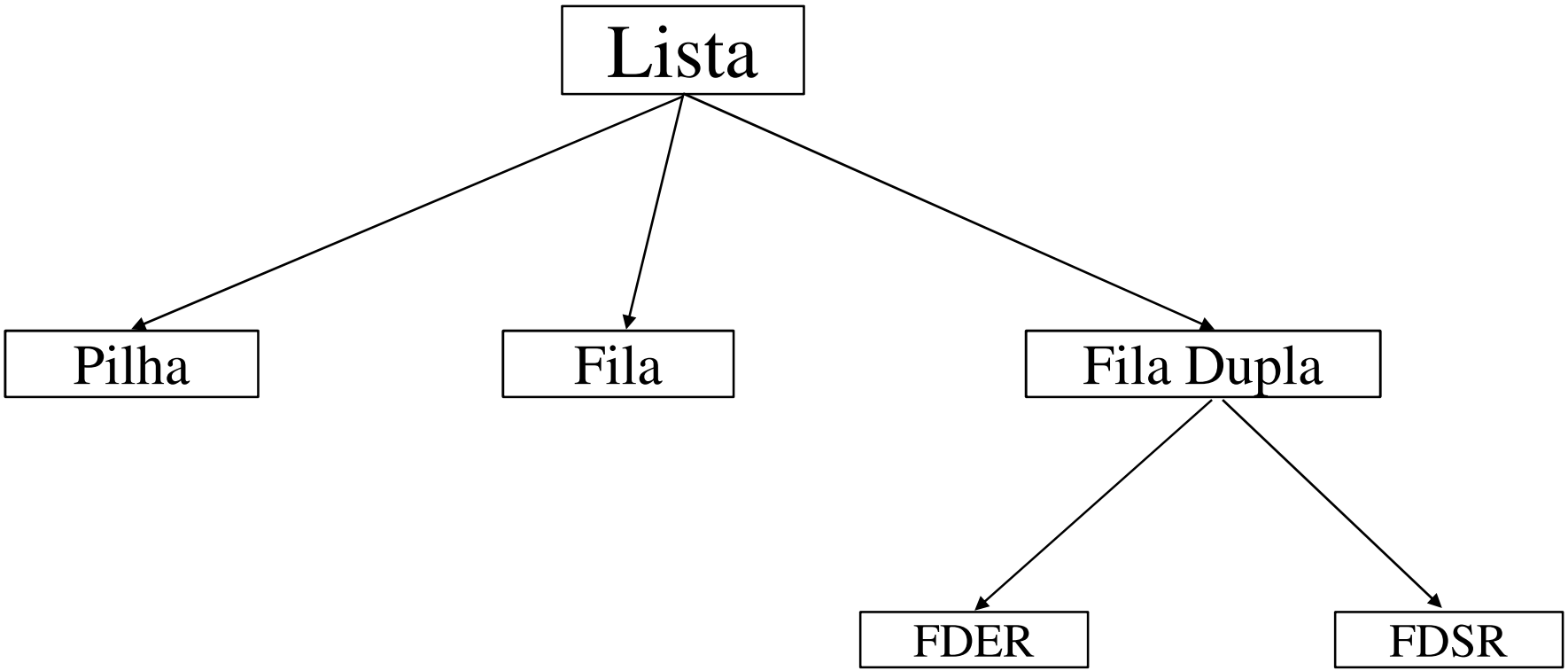
# Tipos Especiais de Listas

- **Pilha**: lista linear onde todas as inserções e remoções são realizadas em um único extremo da lista. Conhecidas também como listas LIFO (Last-In/First-Out)
- **Fila**: lista linear onde todas as inserções são realizadas num determinado extremo da lista e as remoções, no outro extremo. Conhecidas também como FIFO (First-In/First-Out)

# Tipos Especiais

- **Fila Dupla**: lista linear onde as inserções e remoções podem ser feitas em qualquer extremo.
  - Fila Dupla de Entrada Restrita (**FDER**): inserção restrita a um único extremo.
  - Fila Dupla de Saída Restrita (**FDSR**): remoção restrita a um único extremo.

# Hierarquias



# Implementações das Listas

- Quanto a alocação de memória, a implementação de listas lineares pode ser:

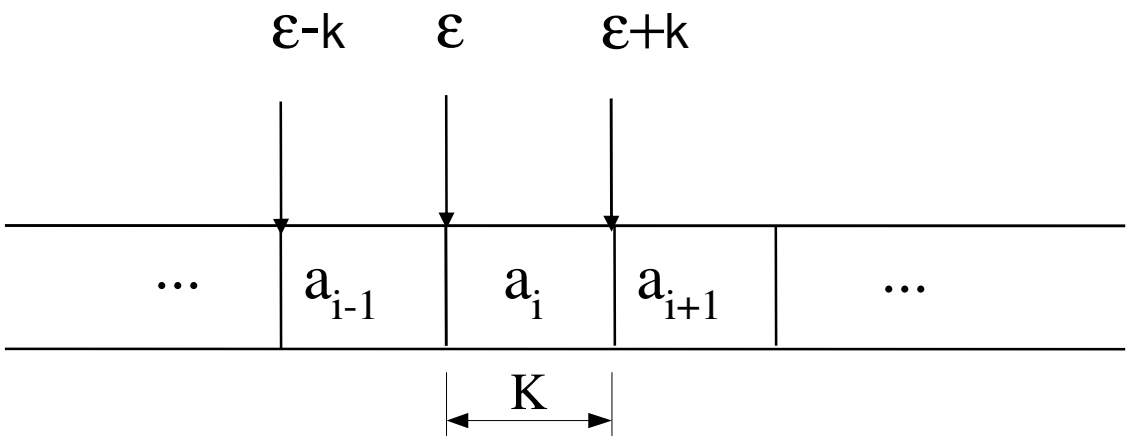
| | Seqüencial | Encadeada |
|---|---|---|
| Estática | Estática Seqüencial | Estática Encadeada |
| Dinâmica | Dinâmica Seqüencial | Dinâmica Encadeada |

# Alocação Estática e Dinâmica

- Estática: quantidade total de memória utilizada pelos dados de um programa é previamente conhecida e definida de modo imutável. Durante toda a execução a quantidade de memória utilizada não varia
- Dinâmica: durante a execução, a quantidade de memória utilizada pelos dados do programa é variável

# Alocação Seqüencial

- Seqüencial: elementos da lista são colocados em posições de memória consecutivas
  - Se cada célula de memória ocupa k bytes, então:

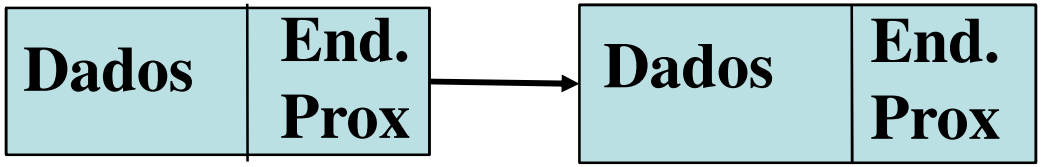


# Alocação Seqüencial

- Pontos Fortes:
  - Fácil Endereçamento
  - Aritmética simples (endereços)
  - Fácil inserção e supressão de elementos no final da lista
- Pontos Fracos:
  - Difícil inserção e supressão de elementos no meio da lista
  - Difícil movimentação de elementos na lista

# Alocação Encadeada

- Encadeada: elementos podem ocupar quaisquer posições de memória
  - Para manter a ordem linear, juntamente com cada elemento é armazenado o endereço do próximo elemento da lista
  - Elementos ocupam blocos de memória chamados nós
    - Cada nó possui dois campos: dados e endereço do nó seguinte da lista



# Alocação Encadeada

- A grande vantagem da alocação encadeada vem com as operações de inserção e supressão de elementos no meio da lista
  - Como os elementos não mais estão ordenados na memória (endereço), fica bem mais simples a gerência da lista com as operações de inserção e remoção de elementos

# Listas Seqüenciais

- Estática Seqüencial
  - É implementada usando um vetor
  - Deve-se determinar qual a quantidade máxima de elementos que a lista poderá armazenar.
  - A memória para armazenamento dos dados é alocada em tempo de compilação

# Lista Estática Seqüencial

- Uma lista é apenas um conjunto de **nós**
  - **Um vetor de nós é uma idéia imediata para sua implementação**
  - **Entretanto, os nós podem, ou não, ser ordenados pelos índices do vetor**
    - **Cada um dos nós pode conter em si próprio um ponteiro para o próximo elemento, ex.:**

**Em C:**

```
#define NUMNODES 500
struct nodetype{
  int info, next;
};
struct nodetype node[NUMNODES];
```

**Em Pyton:**

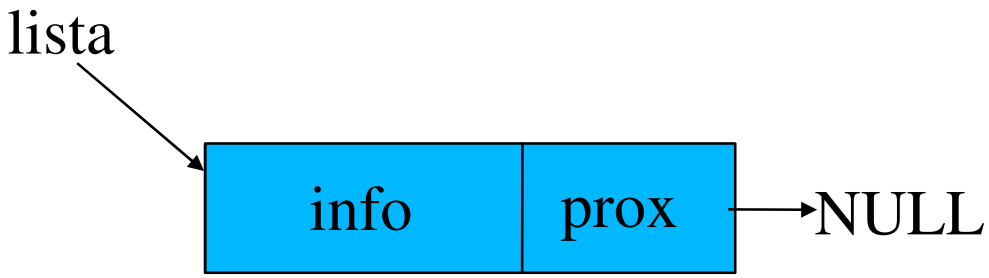
```
Max = 500
Nodo = (info, next)
Lista = []
for i in range(Max):
        Lista += [Nodo]
```

# Listas Lineares

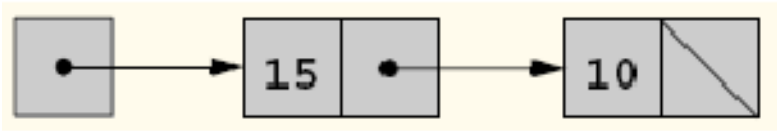
- Dinâmica Encadeada
  - É implementada usando objetos (ou ponteiros)
  - Em tempo de execução, a memória é dinamicamente alocada conforme novos elementos são colocados na lista e também dinamicamente liberada quando elementos são retirados

# Listas Dinâmicas Encadeadas

- É possível encarar uma lista com um conjunto dinâmico de **nós**
  - **A idéia básica é a determinação de uma classe (ou estrutura) que defina um nó**




  - **A parir desta estrutura é possível criar uma lista**



# Listas Ordenadas e Desordenadas

- Na lista desordenada os elementos são colocados na primeira posição vazia da lista (início ou final)
- Na lista ordenada, é escolhido um elemento da estrutura que será o campo de ordenação da lista
  - Quando se deseja inserir um novo elemento, deve ser verificado em que local o mesmo deve ser colocado para que seja mantida a ordem da lista

# Tipos de Listas

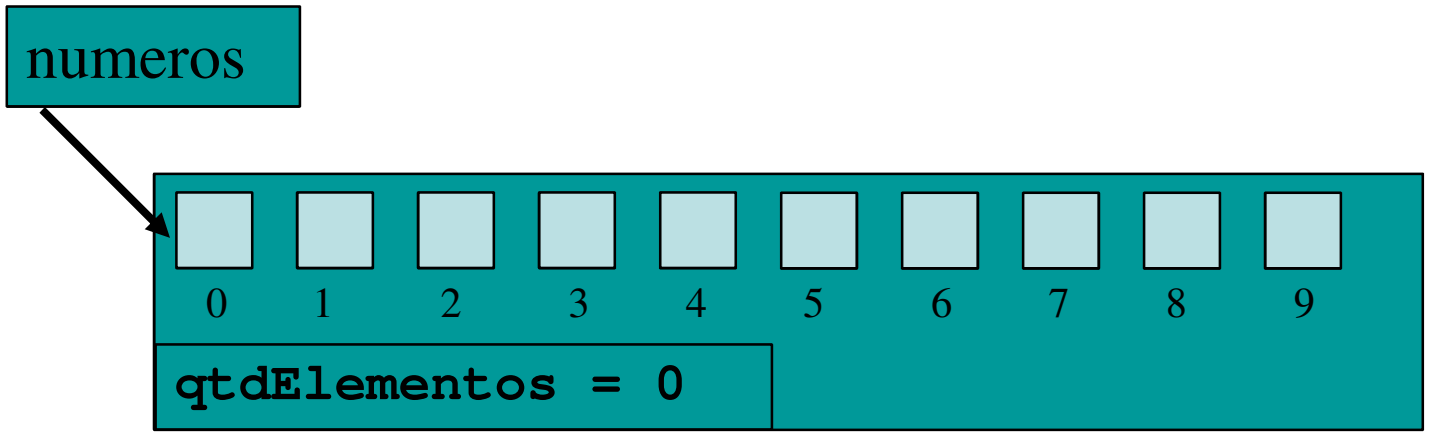
- Lista Estática Desordenada
- Lista Estática Ordenada
- Lista Dinâmica Desordenada
- Lista Dinâmica Ordenada

# Operações Básicas

- Inserir elemento
- Remover elemento
- Consultar elemento (ou Pesquisar)
- Alterar elemento
- Listar os elementos

# Lista Estática Desordenada

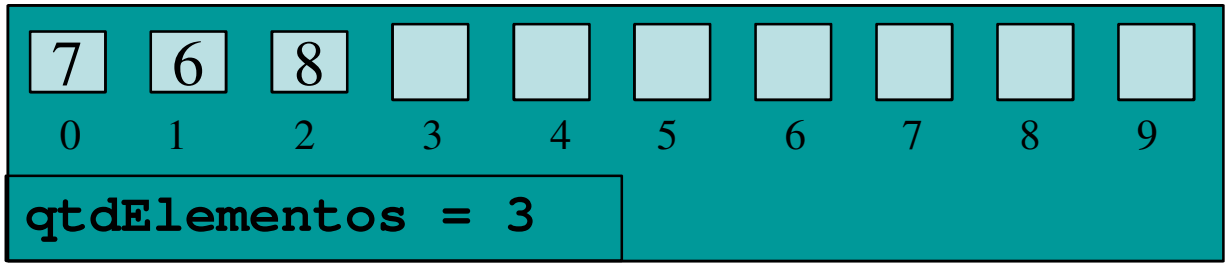
- Exemplo: Lista de Números
  - Executar operações com uma lista implementada usando um vetor de 10 elementos que armazena apenas números.
- Representação Gráfica



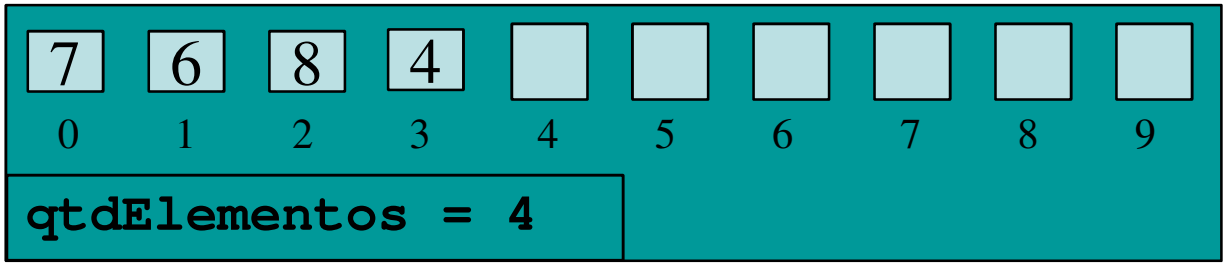
# Lista Estática Desordenada

- Inserindo Elementos:
  - Verificar se a lista ainda tem posição disponível
    - Caso afirmativo, o elemento é colocado na primeira posição vazia (e.g. no final da lista), que é indicada pela variável qtdElementos.
    - Posteriormente, incrementar a quantidade de elementos da lista

# Lista Estática Desordenada




Se desejarmos inserir o número 4, ele será colocado na posição 3 depois a variável qtdElementos é incrementada:

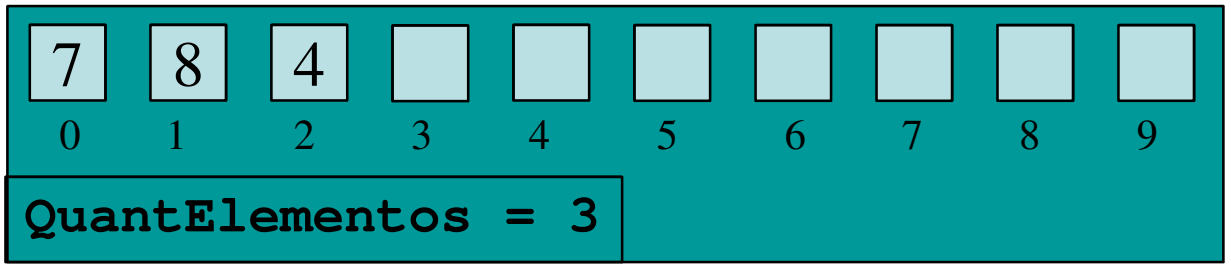


# Lista Estática Desordenada

- Removendo Elementos
  - Verificar se o elemento encontra-se na lista
    - Caso afirmativo, o elemento é removido trazendo todos os elementos que se encontram após o mesmo (se existirem), uma posição a frente, fechando o espaço deixado pelo elemento removido.
    - A quantidade de elementos deve ser decrementada

# Lista Estática Desordenada

- Consultando Elementos



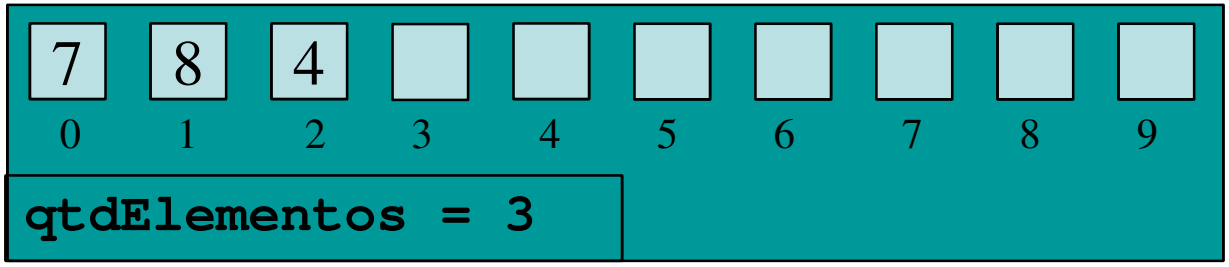

- A consulta do número 8 retorna o segundo elemento do vetor (índice 1).

- A consulta do número 1000 retorna uma mensagem de erro (Elemento não encontrado!).

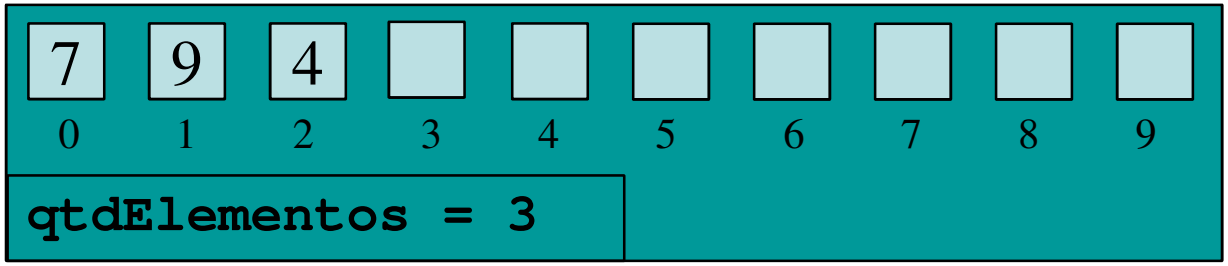
# Lista Estática Desordenada

- Alterando Elementos
  - Para alterar o valor de um elemento primeiramente faz-se uma consulta para verificar se o mesmo se encontra no vetor.
  - Em caso de encontrá-lo no vetor, é informado o novo valor para o elemento.
  - Se ele não está no vetor é apresentada uma mensagem de erro.

# Lista Estática Desordenada




Para alterar o número 8 do vetor para 9:

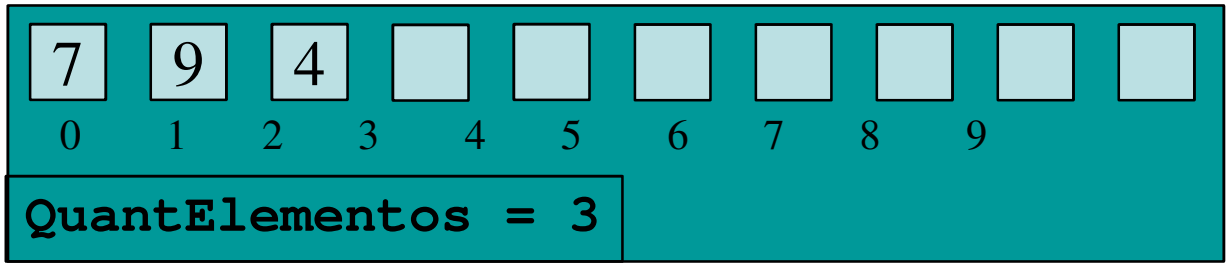


# Lista Estática Desordenada

- Listando os Elementos
  - Para listar os elementos da lista é feito um laço da primeira até a ultima posição ocupada da lista e todos os elementos são impressos

# Lista Estática Desordenada

- Listando os Elementos




Listagem:

```
Elemento 0: 7
Elemento 1: 9
Elemento 2: 4
```

# Listas Dinâmicas

- Comumente chamadas de **Listas Ligadas ou Listas Encadeadas**
- Estrutura de tamanho variável que utiliza apenas a quantidade de memória que precisa.
- São representadas como seqüências de dados definidas pelo encadeamento dos elementos.
- **Cada elemento** é chamado de **nó** da lista e contém os dados e um ponteiro (ou link) para o próximo nó da lista

# Listas Lineares

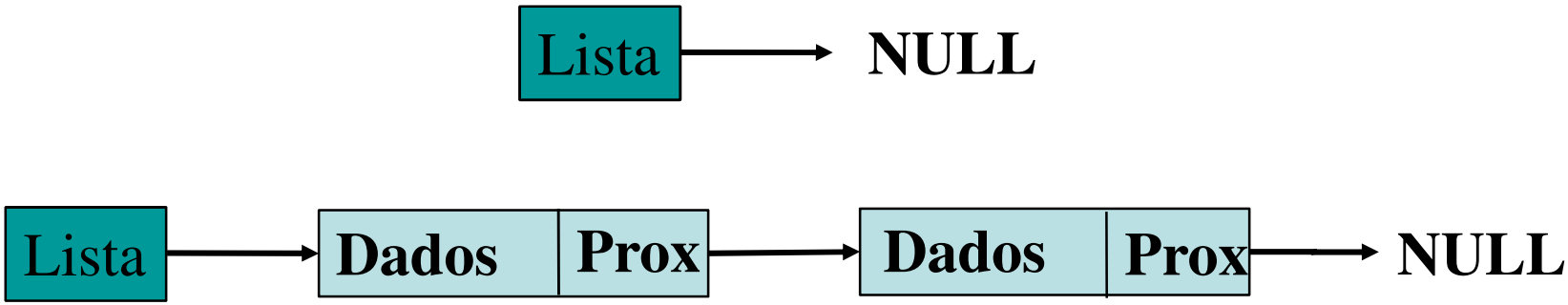
- Vantagens
  - A memória é alocada e liberada quando necessário.
  - A alocação dinâmica nos oferece a necessária flexibilidade para mantermos, sem um grande número de movimentações de nós na lista, a estrutura devidamente ordenada a cada inserção e/ou retirada de elemento
- **Desvantagem**
  - Maior grau de complexidade de implementação

# Listas Dinâmicas

- Aplicações
  - São adequadas para aplicações onde não é possível prever a demanda por memória, permitindo a manipulação de quantidades imprevisíveis de dados, de qualquer formato

# Listas Dinâmicas

- Início da Lista
  - O início da lista é estabelecido por um ponteiro para o 1º nó da lista.
  - Caso a lista esteja vazia, inicialmente o ponteiro aponta para NULL ou None



# Operações Básicas

- Criar a Lista
- Inserir elemento
- Remover elemento
- Consultar elemento
- Alterar elemento
- Listar os elementos

# Lista Encadeada - Criação

## Criando a Lista

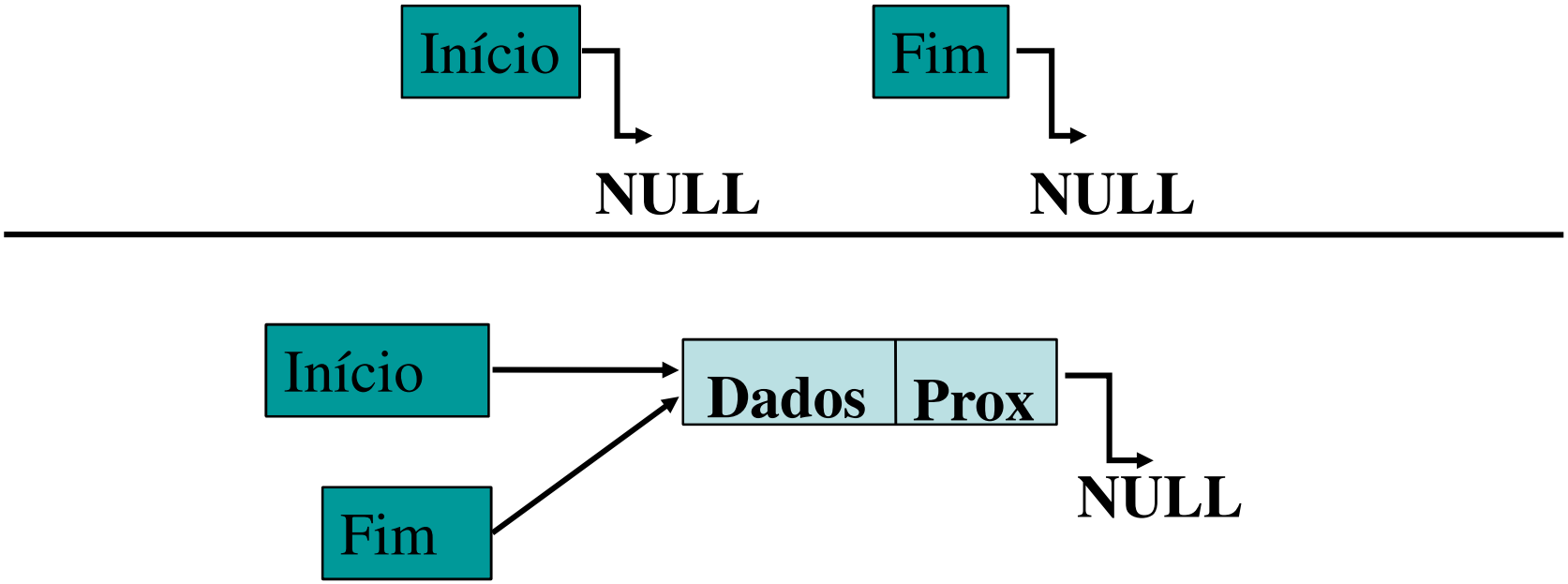
- Inicialmente, declara-se dois ponteiros
  - Um para o início e outro para o fim da lista.
- O ponteiro para o fim da lista permite realizar inserções sem que seja necessário percorrer toda a lista
- Inicialmente, como a lista está vazia, ambos apontam para NULL

# Lista Dinâmica - Inserção

- Inserindo Elementos
  - **Caso 1**: Lista Vazia
    - Cria-se o novo nó e os dois ponteiros apontam para o novo nó inserido na lista, que por sua vez aponta para NULL.
  - **Caso 2**: Lista Não Vazia – Inserção no final da lista
    - Cria-se o novo nó que aponta para NULL; o último nó da lista aponta para o novo nó; e, ponteiro de fim aponta para o novo nó

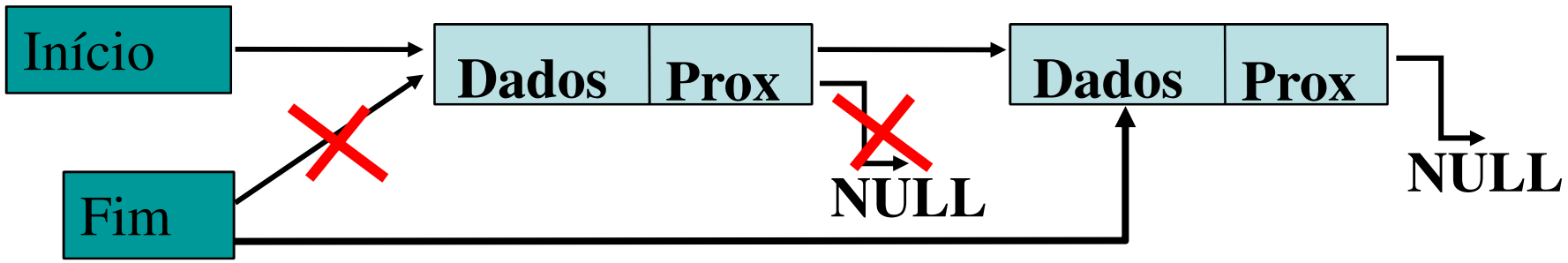
# Lista Dinâmica - Inserção

- Inserindo Elementos
  - Caso 1: Lista Vazia:

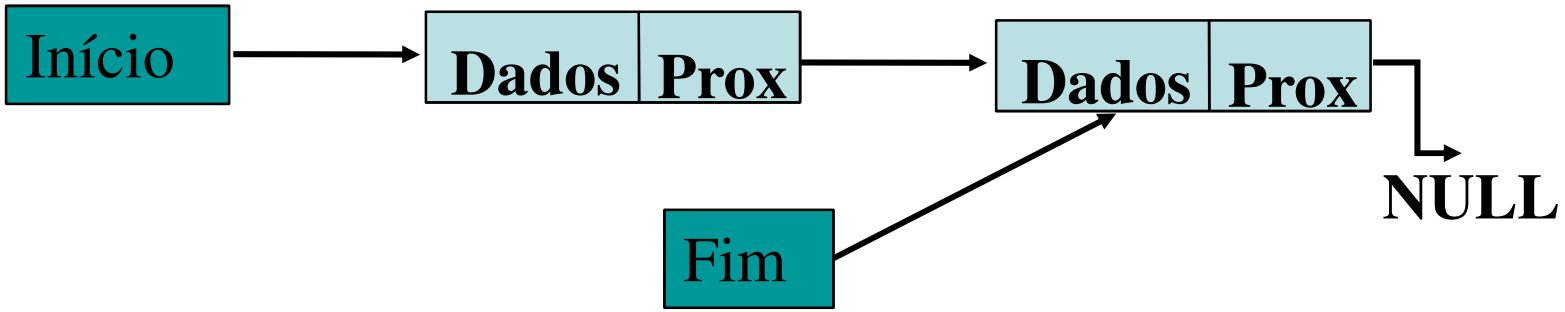


# Lista Dinâmica - Inserção

- Inserindo Elementos
  - Caso 2: Lista Não Vazia – Inserção no final da Lista

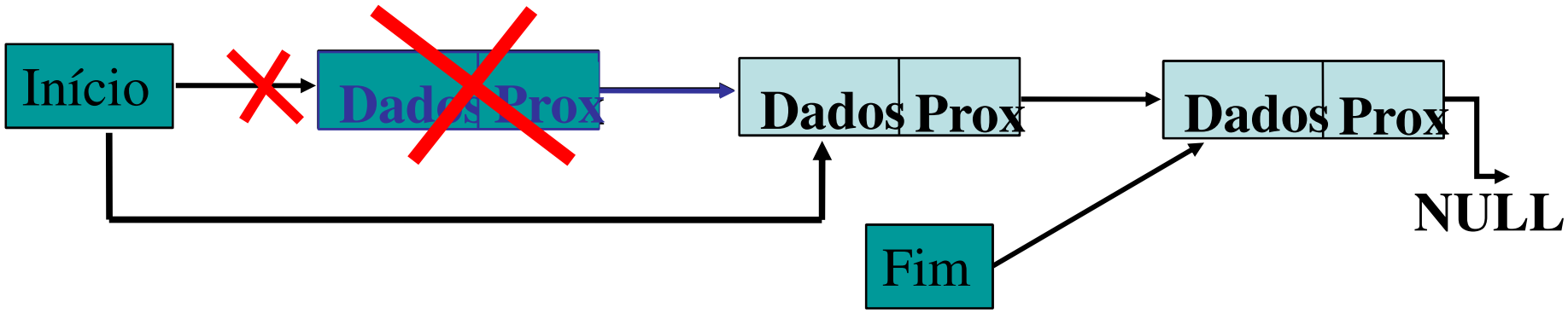



  - Resultado Final:



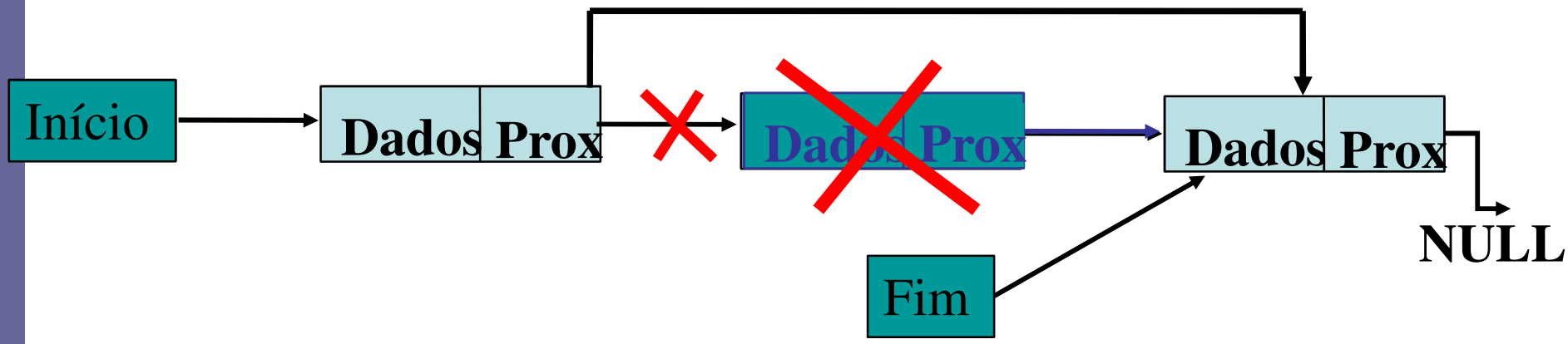
# Lista Encadeada - Remoção

- Removendo Elementos
  - Caso 1: Remover primeiro elemento da lista
    - O elemento a ser removido é marcado.
    - O ponteiro do início aponta para o próximo elemento.
    - A memória é liberada



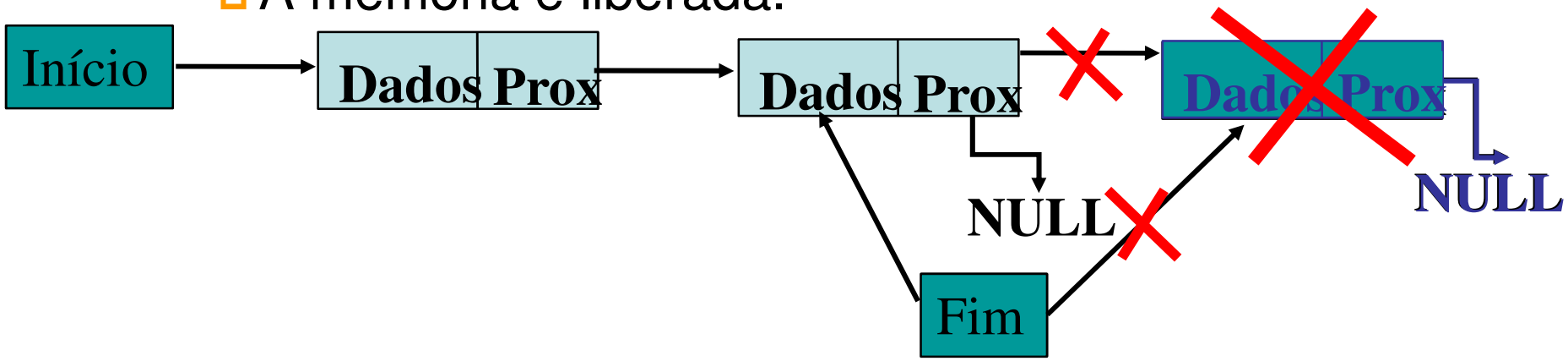
# Lista Encadeada - Remoção

- Removendo Elementos
  - Caso 2: Remover elemento do meio da lista
    - O elemento a ser removido é marcado.
    - O elemento anterior ao removido aponta para onde o removido apontava.
    - A memória é liberada.



# Lista Encadeada - Remoção

- Removendo Elementos
  - Caso 3: Remover elemento do final da lista
    - O elemento a ser removido é marcado.
    - O elemento anterior ao removido aponta para NULL.
    - O ponteiro para fim aponta para o anterior.
    - A memória é liberada.

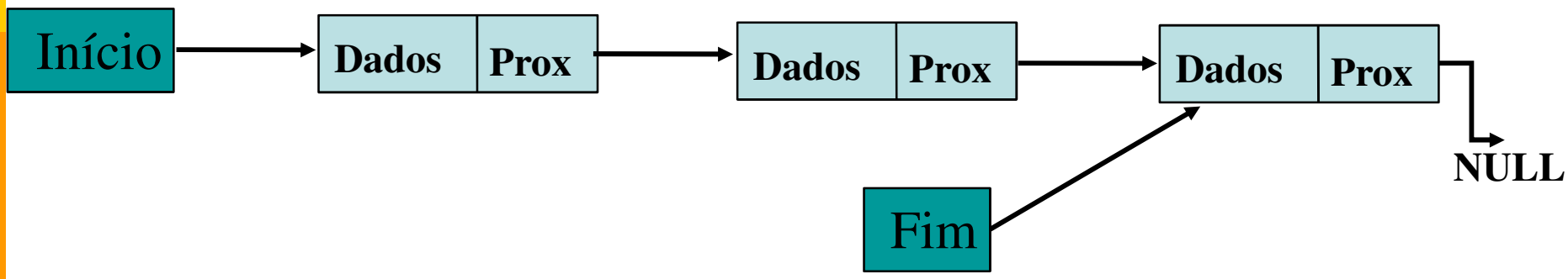


# Lista Encadeada Circular

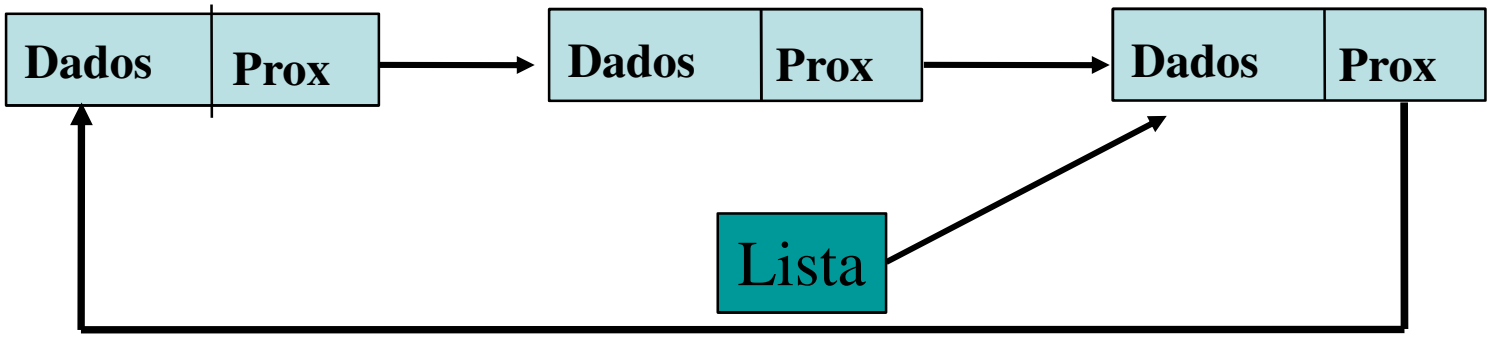
- Listas Encadeadas
  - Cada Nodo (ou Nó) tem dois campos:
    - Dado
    - Prox
  - Para o último Nodo, o campo prox aponta para NULL
- Listas Encadeadas Circular
  - Para estas listas, o campo prox do último Nodo aponta para o primeiro Nodo da lista, formando um “circulo” de encadeamento

# Lista Encadeada Circular

Lista simplesmente encadeada




Lista com encadeamento circular



# Encadeamento Circular

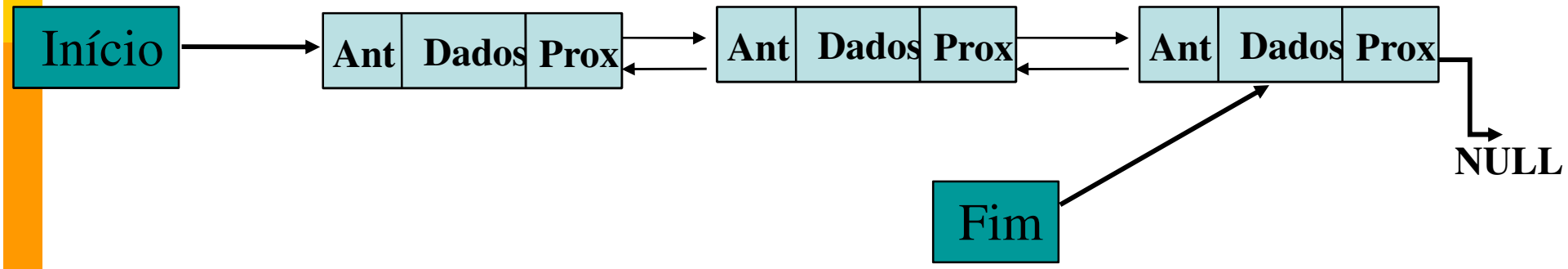
- A grande vantagem nesta lista é que só é necessário o uso de um ponteiro para o final da lista
  - Se a estrutura não for ordenada, o elemento deverá ser inserido sempre no primeiro lugar.
  - Ao remover um elemento, deve-se está atento para o caso de a fila ter somente um único elemento, pois a estrutura torna-se-á vazia

# Encadeamento duplo

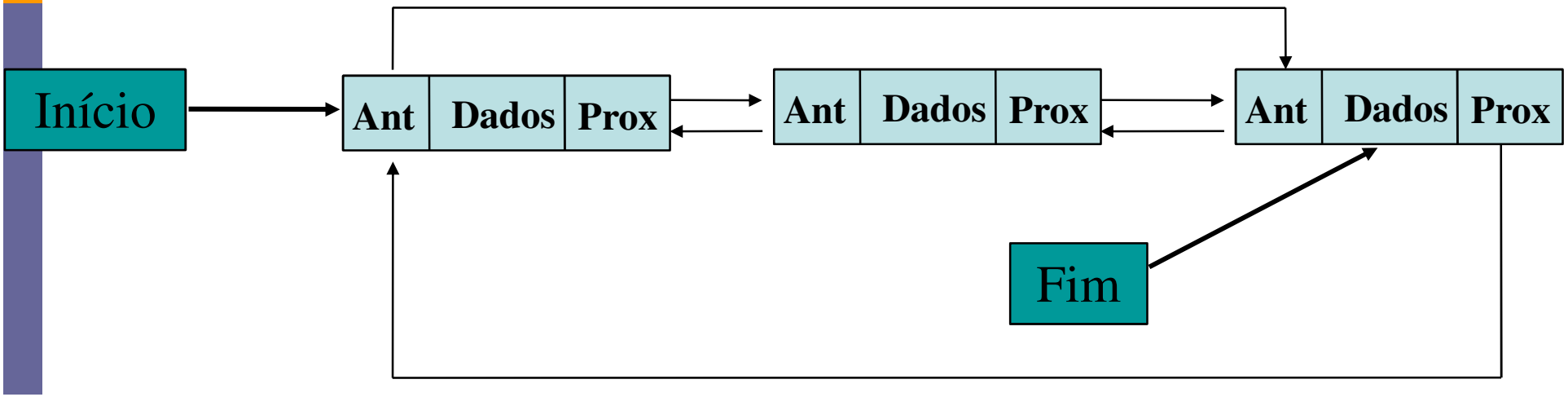
- Para uma lista com encadeamento duplo, cada Nodo tem três campos:
  - Dados
  - Prox. (Referência ao próximo Nodo)
  - Ant. (Referência ao Nodo Anterior)

# Encadeamentos Duplos

Lista duplamente encadeada




Lista com encadeamento duplo circular



# Exercício:

- Implemente em python:
  - Lista estática com 10 elementos
    - Com as funções Básicas sobre os elementos:
      - Inserir
      - Remover
      - Pesquisar
      - Alterar
      - Listar
  - Com as mesmas funções básicas:
    - Lista Encadeada simples
    - Lista Duplamente Encadeada
    - Lista Circularmente Encadeada
    - Lista Circularmente e Duplamente Encadeada