Listas Encadeadas

Luciano Nascimento Moreira

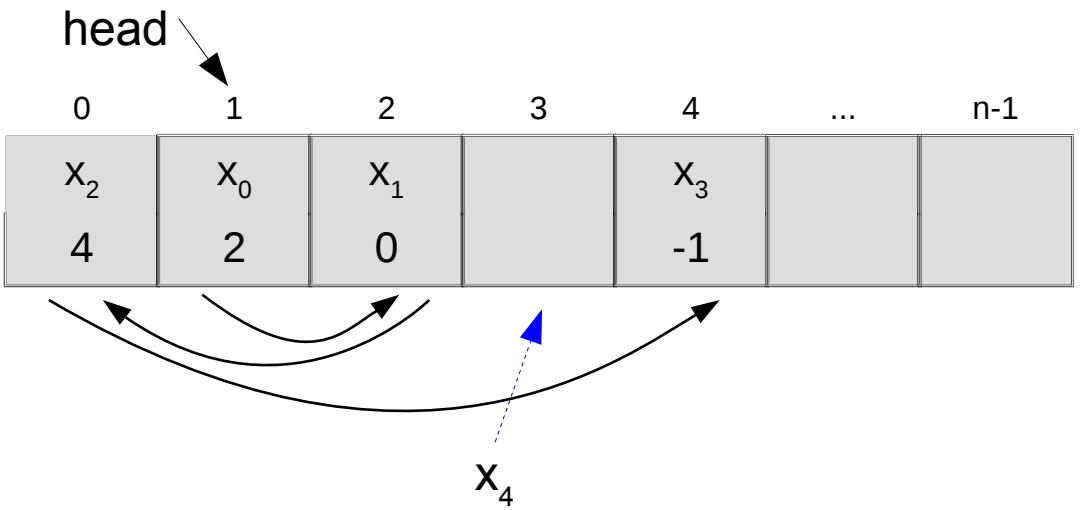
Listas Encadeadas

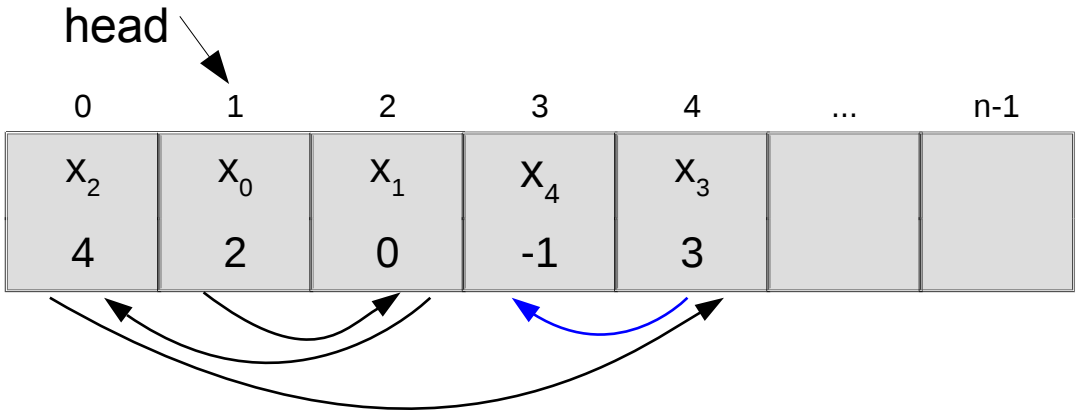
- Elementos consecutivos na lista não implica em elementos consecutivos na representação. A ordem é lógica.
- Cada item da lista contém a informação que é necessária para alcançar o próximo item
- Na implementação é necessário armazenar separadamente a informação do primeiro elemento da lista

Listas Encadeadas

- É possível inserir e retirar elementos sem necessidade de deslocar os itens seguintes da lista
- Existem duas formas de representar listas encadeadas, através de:
  - arrays, denominada lista encadeada estática
  - por referências (ponteiros), denominada lista encadeada dinâmica

Listas Encadeadas Estáticas





Listas Encadeadas

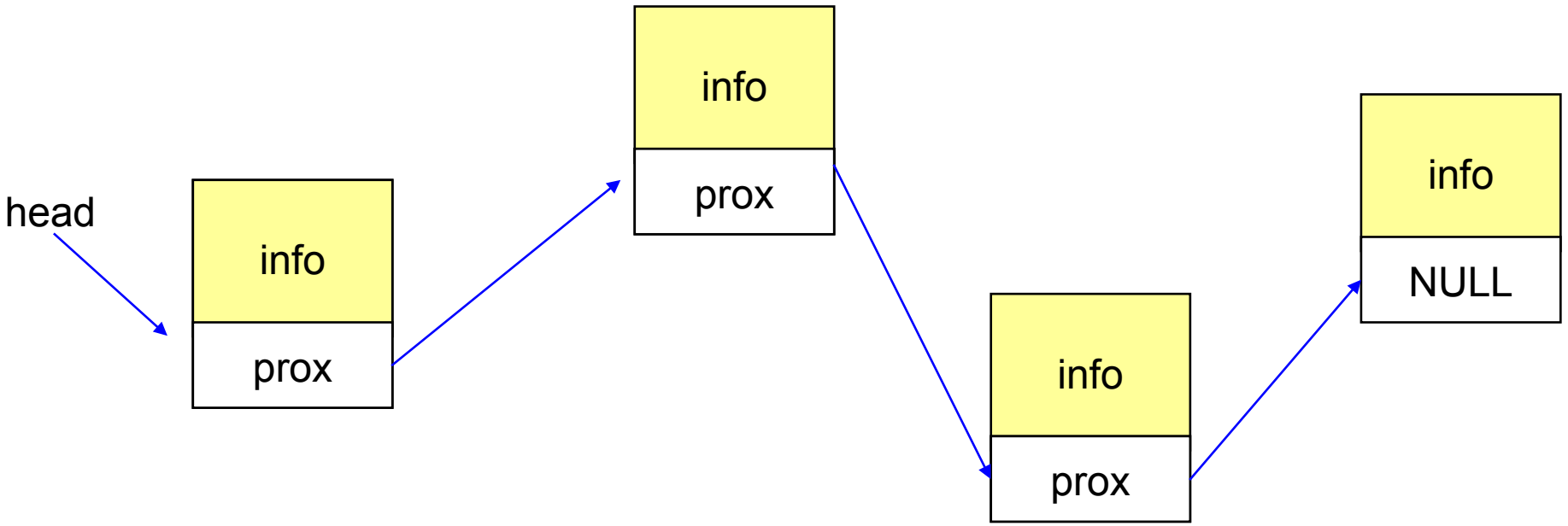
- Vantagens
  - Elimina o problema dos deslocamentos de células
  - Não é necessário saber previamente o número de elementos a serem armazenados (lista dinâmica)
- Desvantagens
  - Não se consegue acessar os elementos da lista em tempo constante
  - Mais operações para manter integridade dos dados

Listas Encadeadas Dinâmicas

- Cada item da lista é encadeado com o seguinte através de uma variável de referência (ponteiro)
- Esta implementação permite utilizar posições não contíguas de memória, sendo possível inserir e retirar elementos sem deslocar os itens da lista
- A lista é constituída de células, onde cada célula contém um item da lista e um apontador para a célula seguinte
- É conveniente utilizar listas encadeadas dinâmicas em aplicações em que não existe previsão sobre o crescimento da lista, por não ser necessário definir o tamanho da lista a priori

Listas Encadeadas Dinâmicas

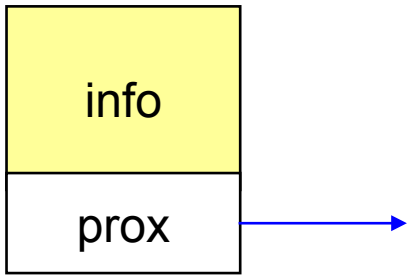
- Características:
  - Elementos não estão contíguos na memória



Desvantagem: utilização de memória extra para as referências

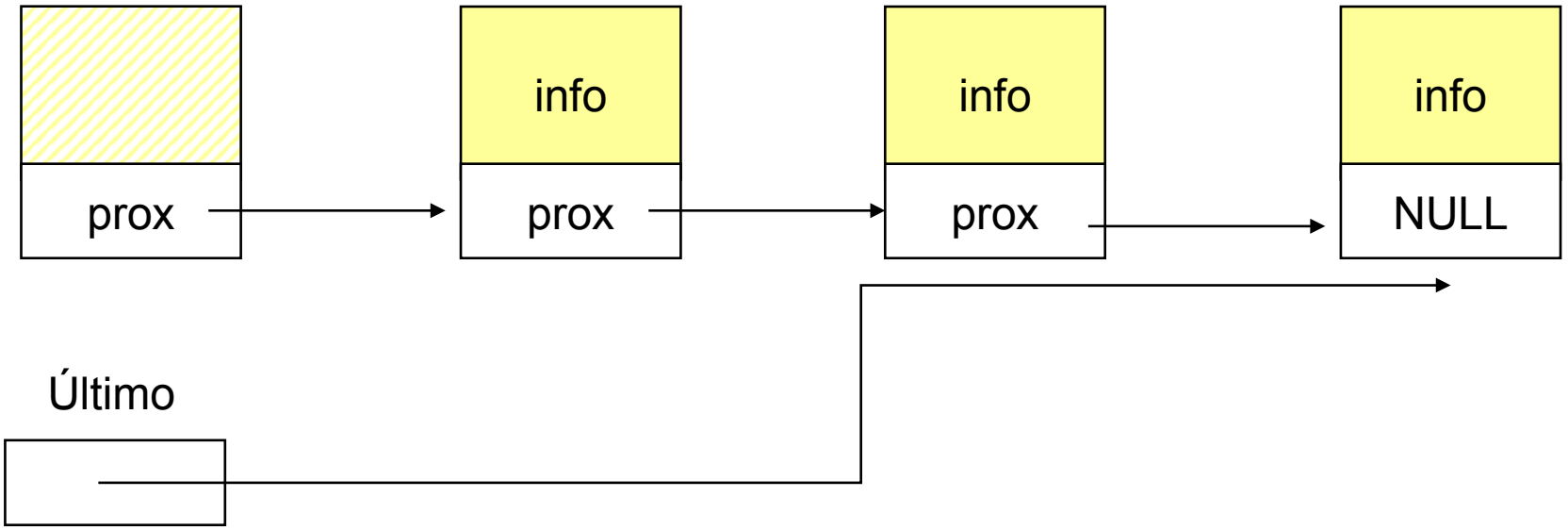
Sobre os Elementos da Lista

- Elemento: guarda as informações sobre cada elemento.
- Para isso define-se cada elemento como uma estrutura que possui:
  - Campos de informações
  - Ponteiro (referência) para o próximo elemento



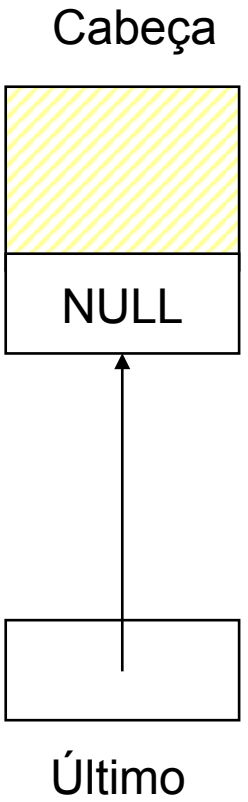
Sobre a Lista

- Uma lista **pode** ter uma célula cabeça

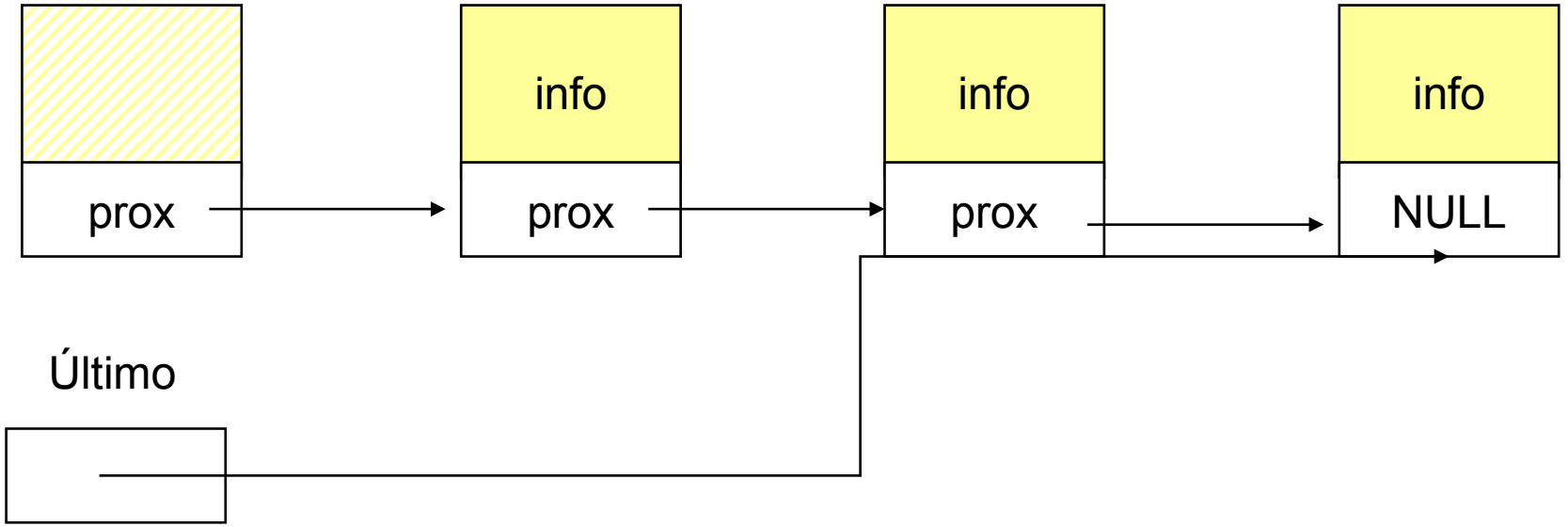


- Uma lista **pode** ter um apontador para o último elemento

Cria Lista Vazia

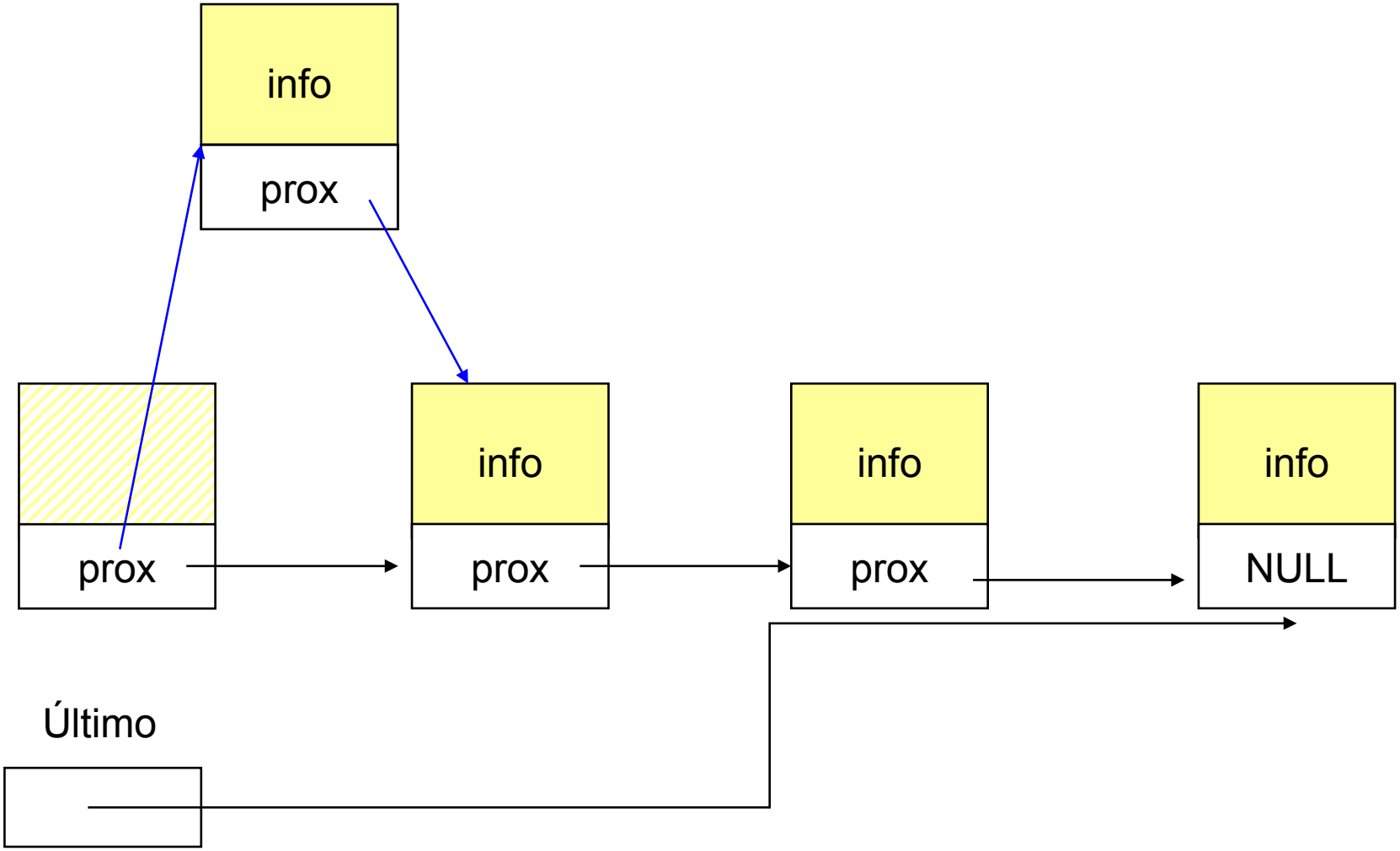


Inserção de Elementos na Lista

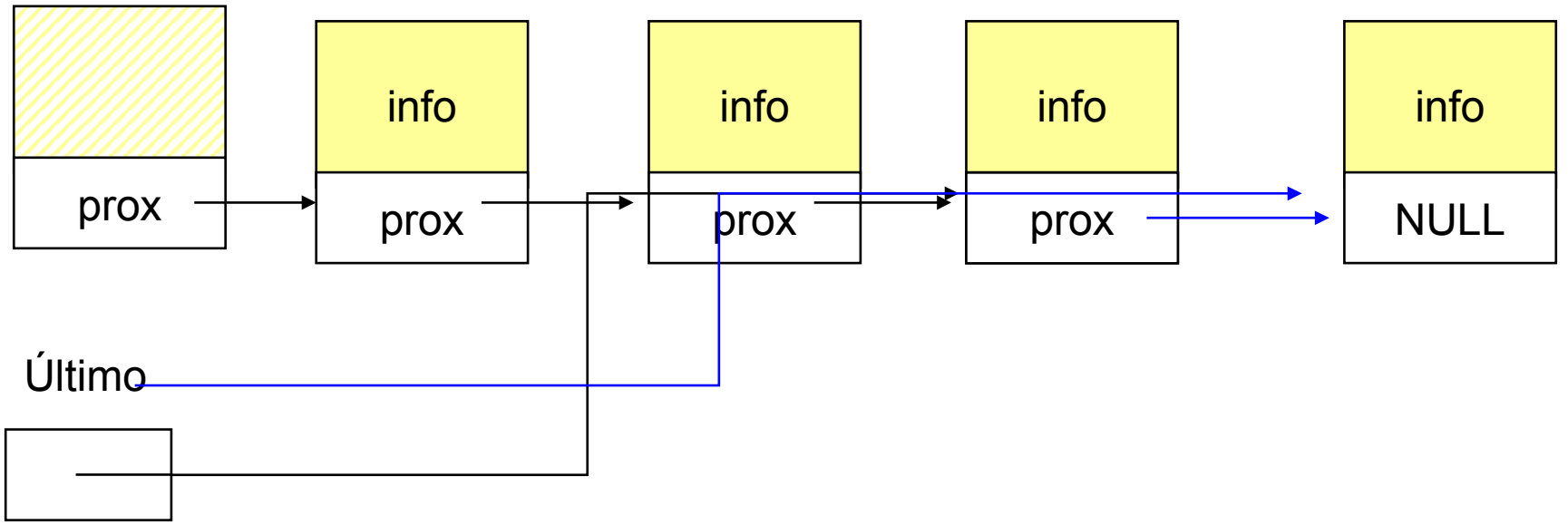


- 3 opções de posição onde pode inserir:
  - 1ª. posição
  - última posição
  - Após um elemento qualquer E

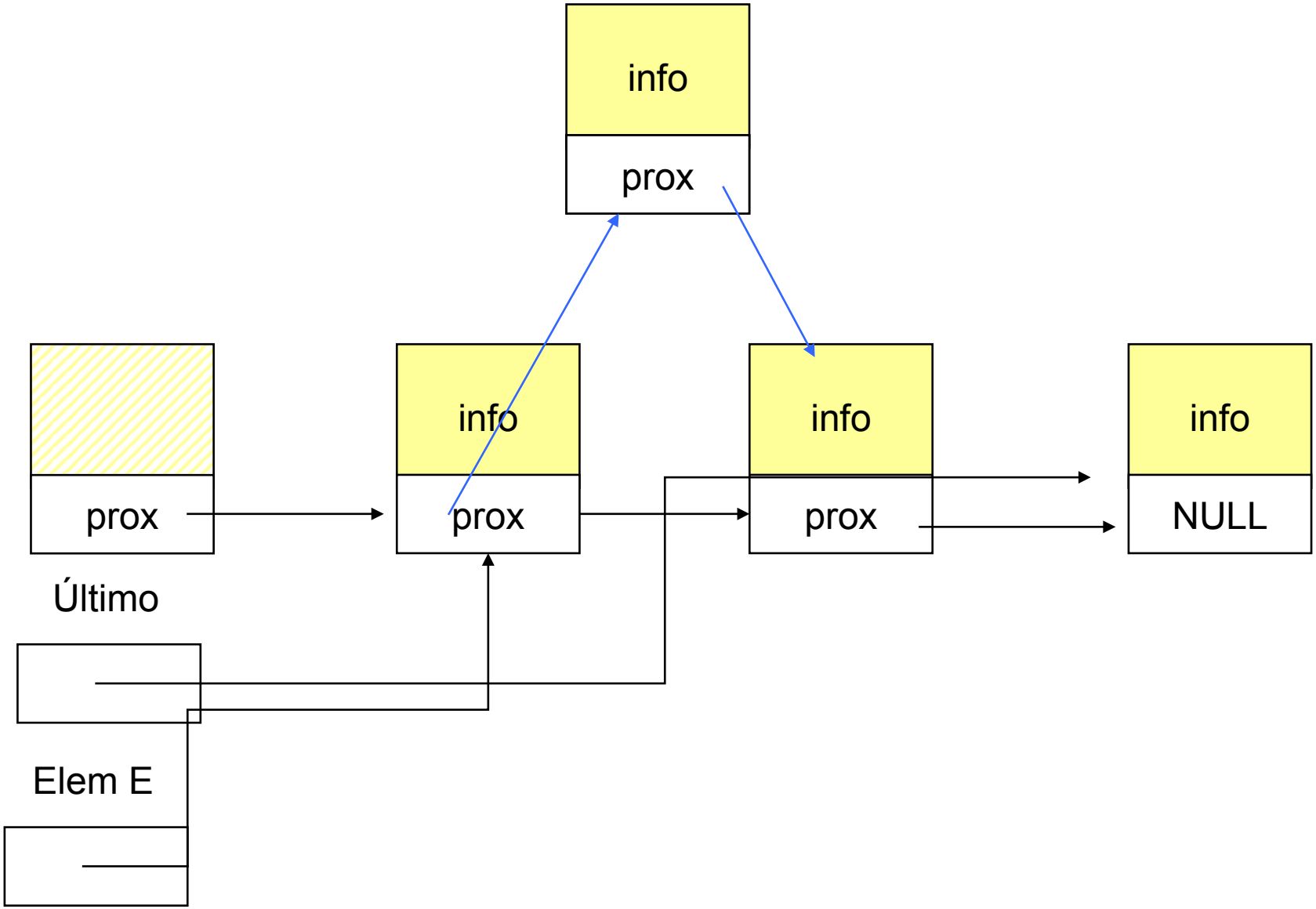
Inserção na Primeira Posição



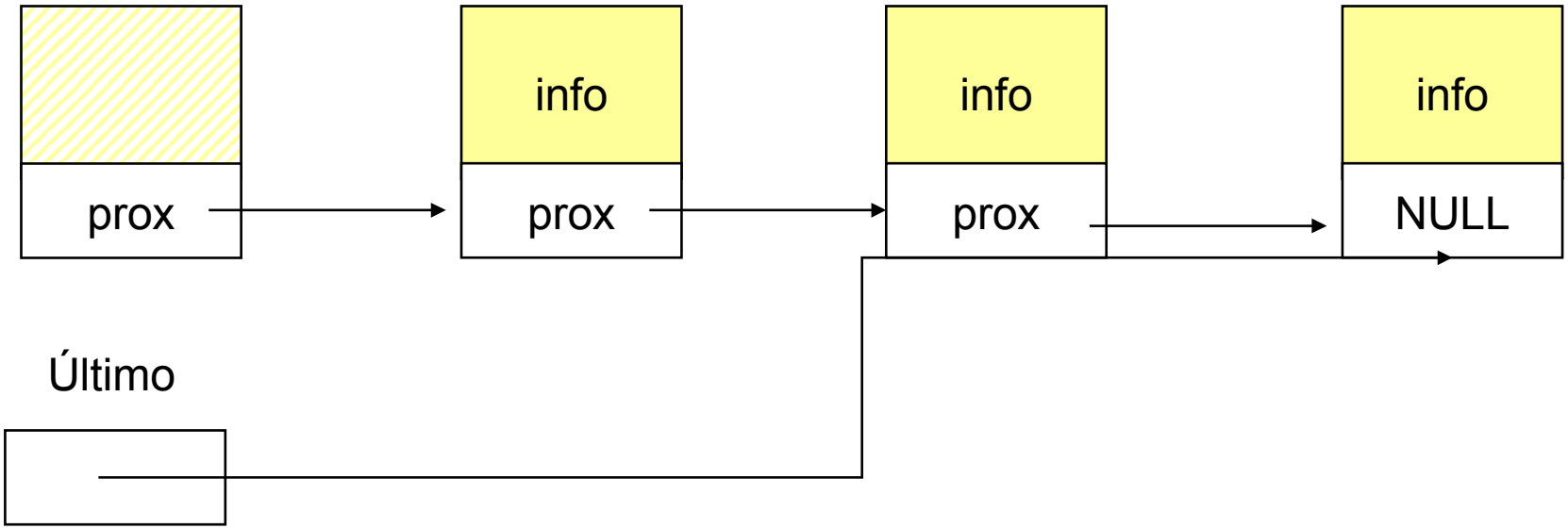
Inserção na Última Posição



Inserção Após o Elemento E

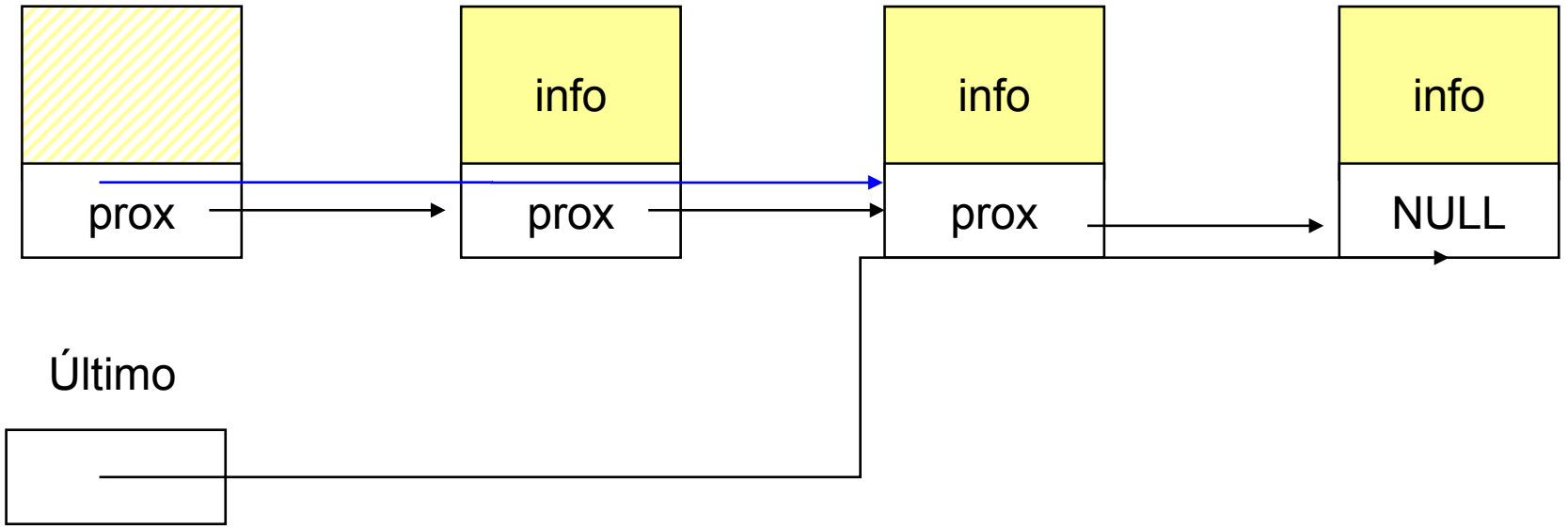


Retirada de Elementos na Lista

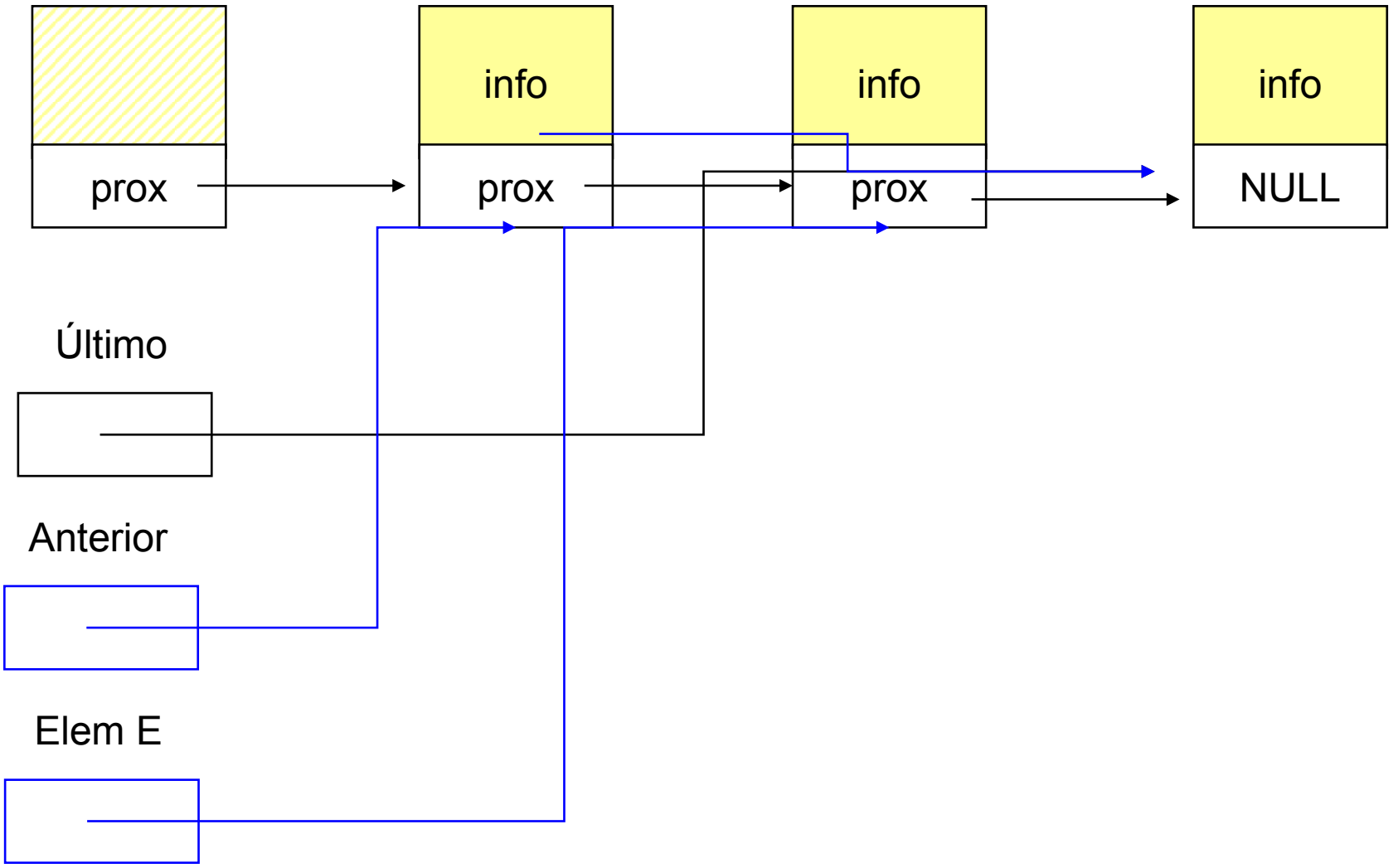


- 3 opções de posição de onde pode retirar:
  - 1ª. posição
  - última posição
  - Um elemento qualquer E

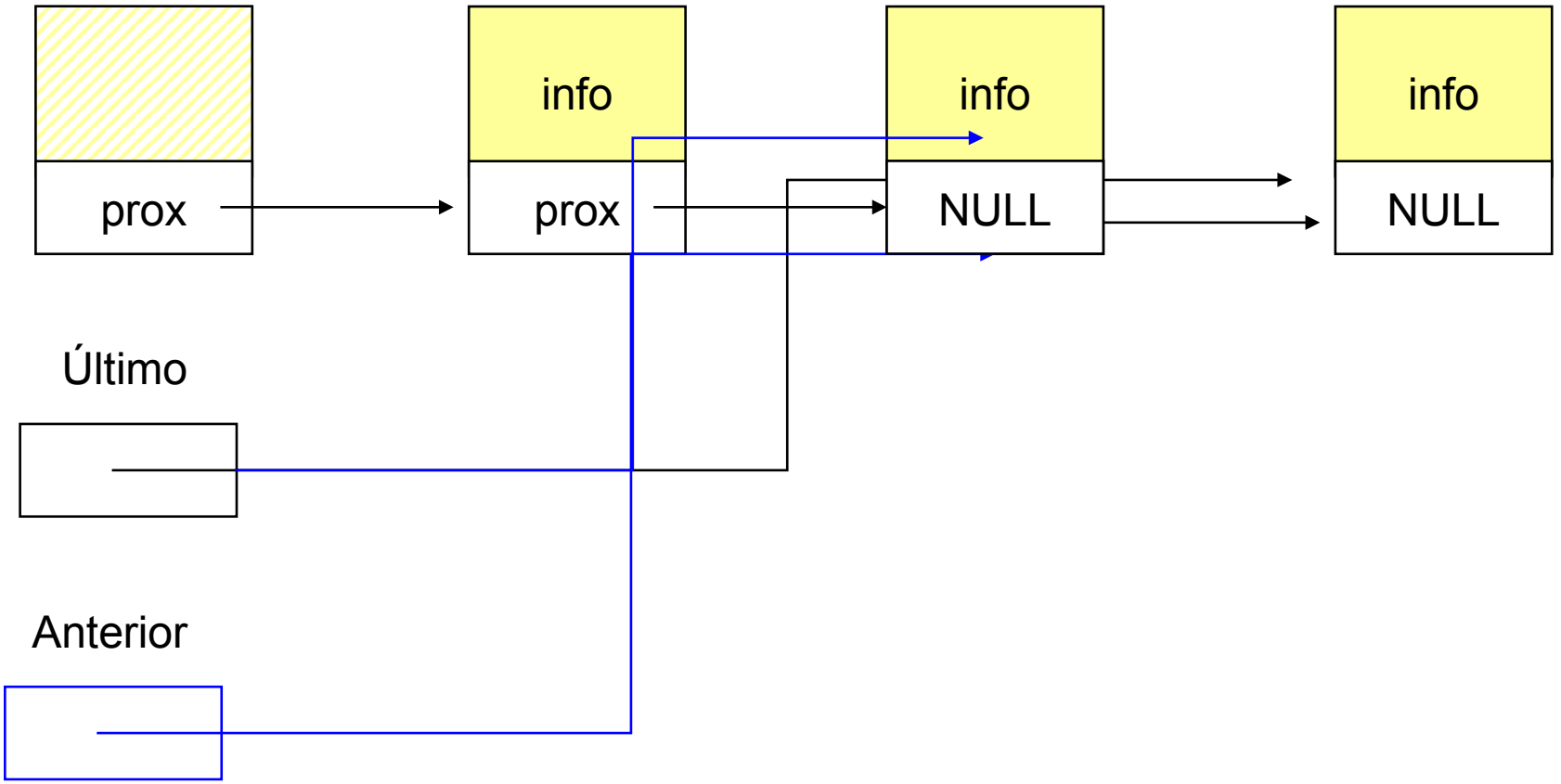
Retirada do Elemento na Primeira Posição da Lista



Retirada do Elemento E da Lista



Retirada do Último Elemento da Lista



Lista Encadeada Dinâmica

```
public class Lista {
    private static class Celula {
        Object item; Celula prox;
    }
    private Celula primeiro, ultimo, pos;

    public Lista () { // Cria uma Lista vazia
        primeiro = new Celula();
        pos = primeiro;
        ultimo = primeiro;
        primeiro.prox = null;
    }
```

Lista Encadeada Dinâmica

```
public void insere (Object x) {
    ultimo.prox = new Celula();
    ultimo = ultimo.prox;
    ultimo.item = x;
    ultimo.prox = null;
}
public boolean vazia () {
    return (primeiro == ultimo);
}
public void imprime () {
    Celula aux = primeiro.prox;
    while (aux ! = null) {
        System.out.println (aux.item.toString());
        aux = aux.prox ;
    }
}
```

Lista Encadeada Dinâmica

```
    public Object retira(Object chave) throws Exception {
        if (vazia() || (chave == null))
            throw new Exception
                        ("Erro: Lista vazia ou chave invalida");
        Celula aux = primeiro;
        while (aux.prox != null && !aux.prox.item.equals(chave))
            aux = aux.prox;
        if (aux.prox == null)
            return null; // não encontrada
        Celula q = aux.prox;
        Object item = q.item;
        aux.prox = q.prox ;
        if (aux.prox == null)
            ultimo = aux;
    }
}
```

Operações sobre Lista usando Referências

- Vantagens:
  - Permite inserir ou retirar itens do meio da lista a um custo constante (importante quando a lista tem de ser mantida em ordem).
  - Bom para aplicações em que não existe previsão sobre o crescimento da lista (o tamanho máximo da lista não precisa ser definido a priori).
- Desvantagem:
  - Utilização de memória extra para armazenar os apontadores.

Exercícios

- O que precisa ser feito para criar um novo elemento para a lista?
- Escreva uma função que receba uma lista como parâmetro e retira o seu primeiro elemento, apagando-o.
- Escreva uma função que receba uma lista e um ponteiro para uma célula como parâmetros e insira a célula na primeira posição da lista.
- Imagine uma lista duplamente encadeada