TAD Lista Simplesmente Encadeada com Nó Sentinela

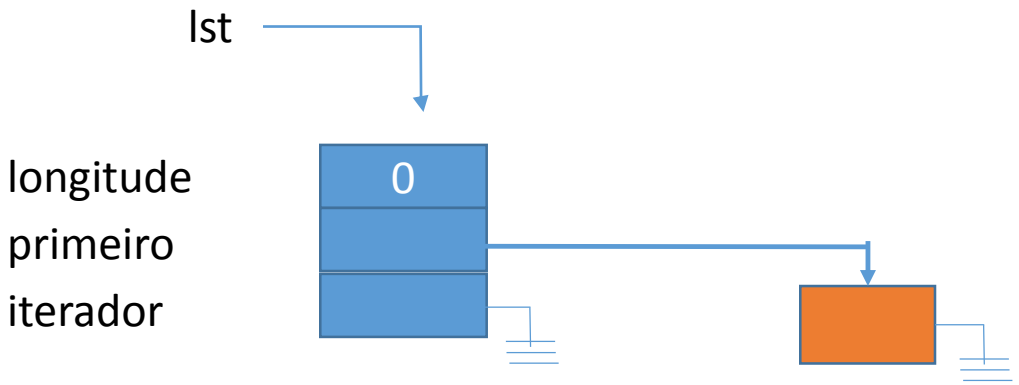
Prof. D.Sc. Saulo Ribeiro

Lista simplesmente encadeada com Nó Sentinela

- Com esta implementação tem-se uma economia considerável em relação a lista duplamente encadeada, pois, economiza-se ponteiros, já que não se precisa do duplo encadeamento.
- O sentinela é um nó que se adiciona no final da lista; ele não possui informação válida, mas permite realizar todas as operações modificadoras em O(1) sem a necessidade do encadeamento para trás.

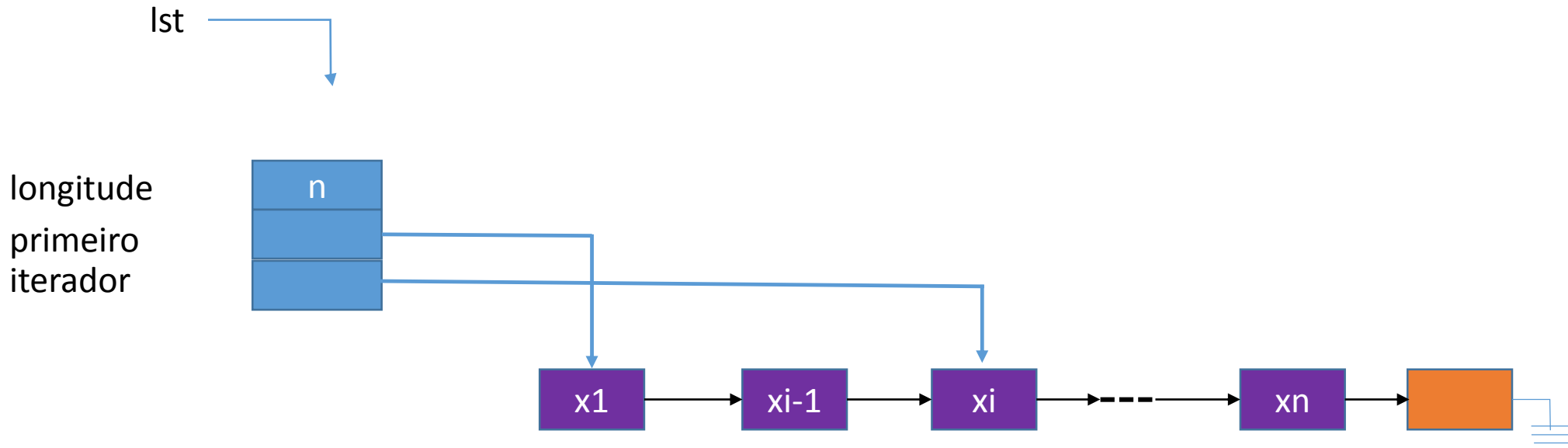
Lista simplesmente encadeada com Nó Sentinela

- **Lista vazia**: o **iterador** está indefinido: não aponta para ninguém (NULL). Primeiro aponta para o nó sentinela.
- lst = < > [ ]



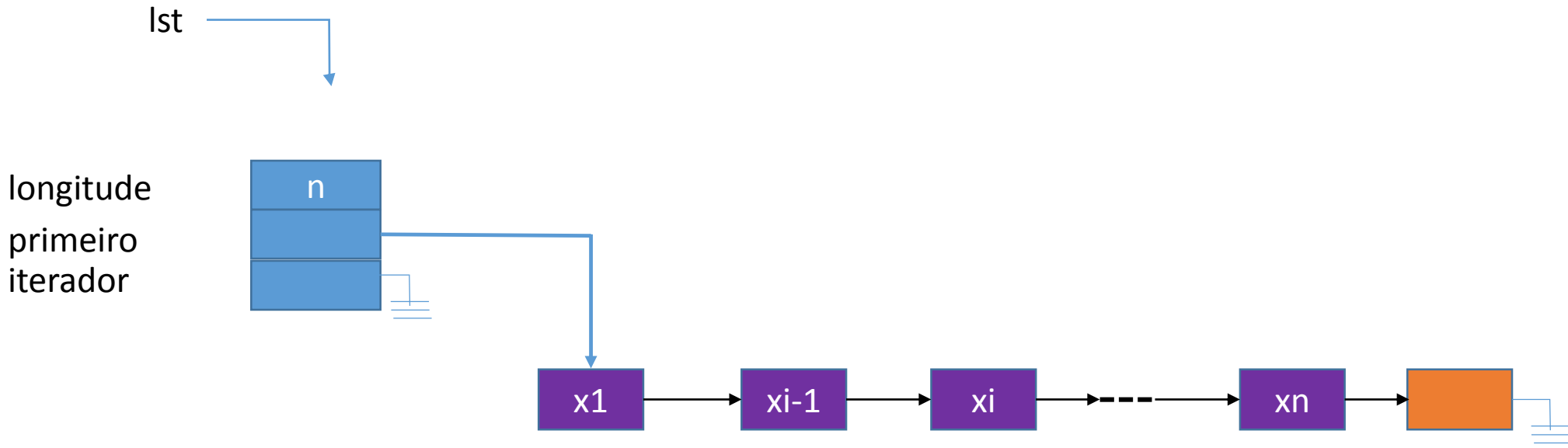
Lista simplesmente encadeada com Nó Sentinela

- **Lista cheia com o iterador definido** (iterador aponta para um elemento da lista). Primeiro aponta para o 1º nó.
- lst = < x1,...,[xi],...xn >



Lista simplesmente encadeada com Nó Sentinela

- **Lista cheia com o iterador indefinido** (iterador aponta NULL). Primeiro aponta para o 1º nó.
- lst = < x1,...,xi,...xn >[ ]



Lista simplesmente encadeada com Nó Sentinela

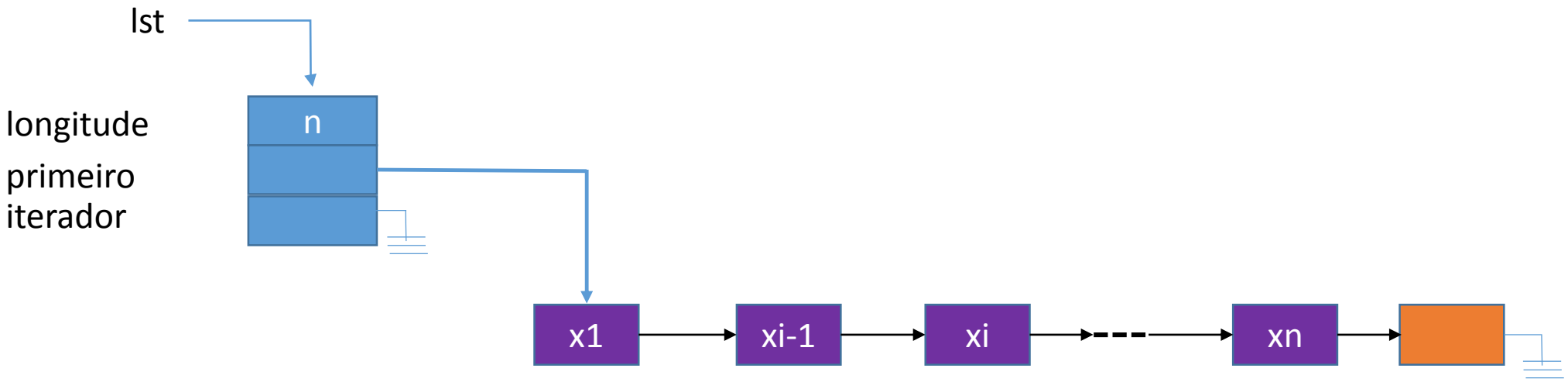

```
typedef int TipoL;

typedef struct ListaNo{
        TipoL info;
        struct ListaNo *prox;
} *pListaNo;

typedef struct{
        pListaNo primeiro, iterador;
        int longitude;
} Tlista, * Lista;
```

Lista simplesmente encadeada com Nó Sentinela

- **Inserir um elemento na Lista:**
- void insLista (Lista lst, TipoL elem): adiciona um elemento antes do iterador
- Diante da dificuldade de se alterar o campo de encadeamento do antecessor(prox) do iterador, a fim de se adicionar um novo Nó, então se cria o Nó a ser adicionado e para ele se copia a informação (info) do iterador, e no campo info do iterador se coloca o elemento a ser adicionado(elem). Depois basta acertar os ponteiros.
- Custo da operação modificadora: O(1)
- Assim, não foi preciso de um ponteiro auxiliar para percorrer a lista e parar um posição antes do iterador. Se isto fosse feito o custo da operação passaria a ser O(n).

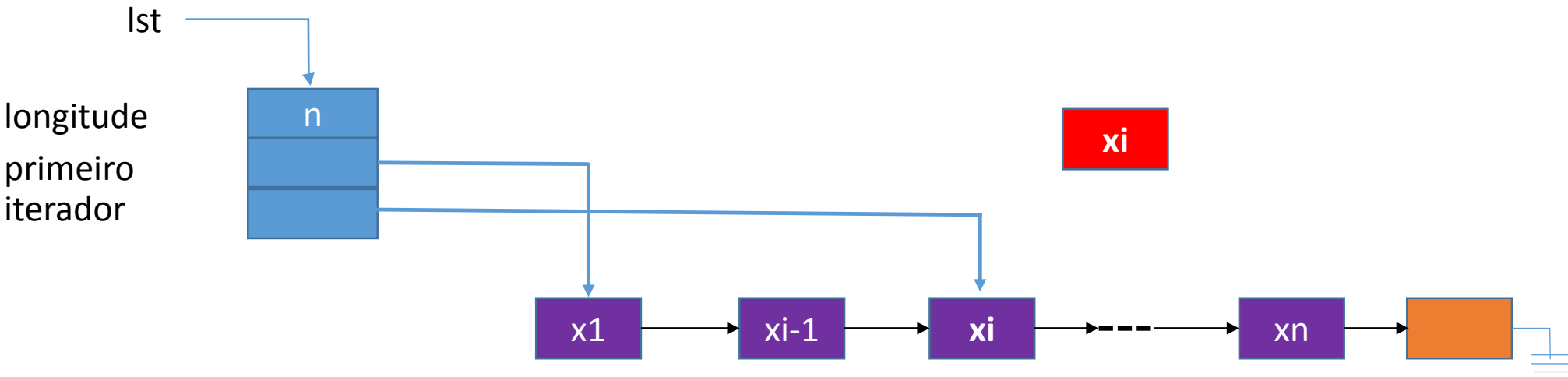
Lista simplesmente encadeada com Nó Sentinela

- **Inserir um elemento na Lista:**
- **Passo 1: cria-se o novo Nó**



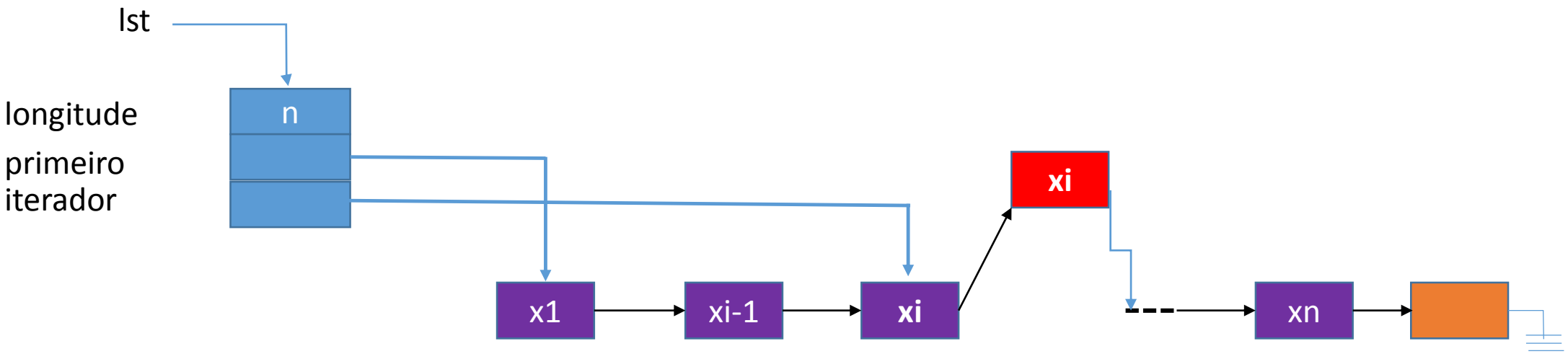
Lista simplesmente encadeada com Nó Sentinela

- **Inserir um elemento na Lista:**
- **Passo 2: copia-se para o novo Nó a informação(info) do iterador**



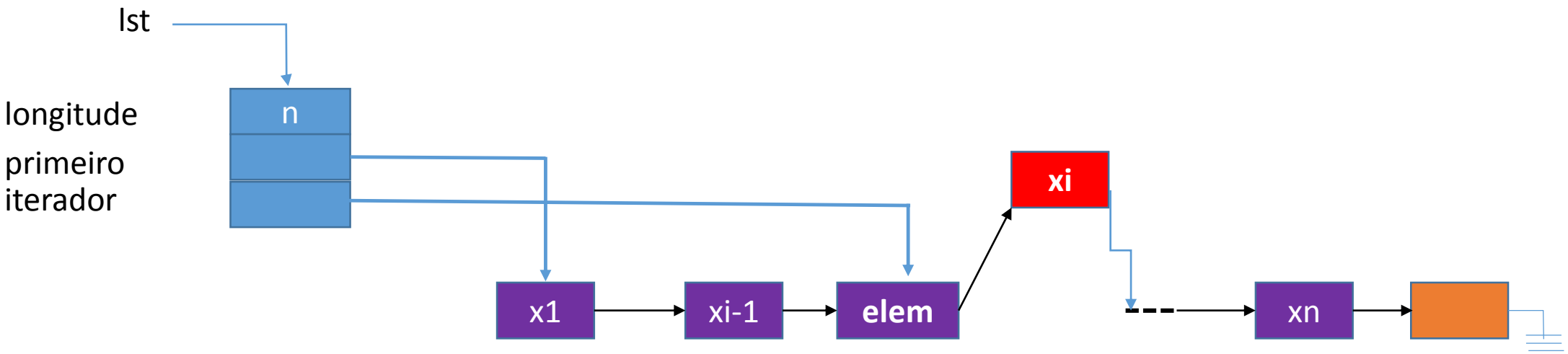
Lista simplesmente encadeada com Nó Sentinela

- **Inserir um elemento na Lista:**
- **Passo 3: se adiciona o novo Nó depois do iterador e se acerta os apontadores**



Lista simplesmente encadeada com Nó Sentinela

- **Inserir um elemento na Lista:**
- **Passo 4: se copia o novo elemento(elem) para o campo info do iterador.**

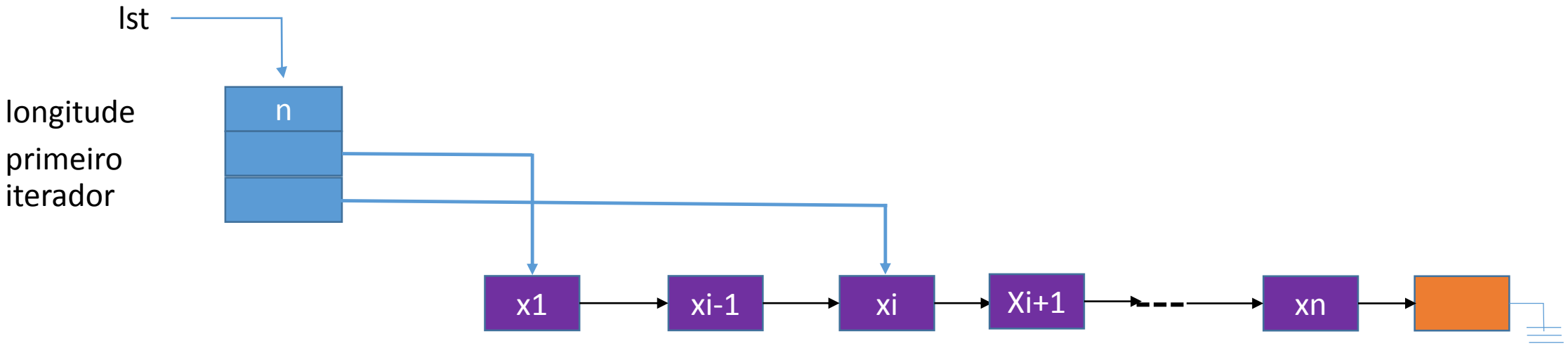


Lista simplesmente encadeada com Nó Sentinela

- **ELIMINAR um elemento na Lista:**
- Para esta rotina se apresenta o mesmo problema do caso da inserção.
- A solução é colocar a informação(info) do sucessor(xi+1) do iterador, no campo info do iterador. Depois é suficiente eliminar o sucessor, um processo simples de acertar os apontadores.
- Para se fazer a operação com custo O(1), é que se coloca o sentinela como parte da estrutura de dados, assim se evita um caso especial com o último elemento.

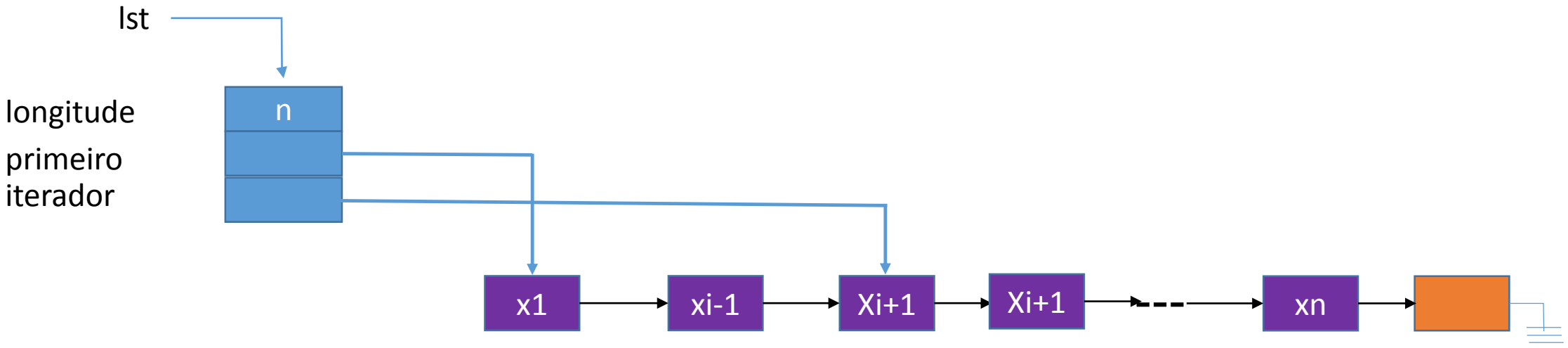
Lista simplesmente encadeada com Nó Sentinela

- **ELIMINAR um elemento na Lista:**
- **Situação Inicial:**



Lista simplesmente encadeada com Nó Sentinela

- **ELIMINAR um elemento na Lista:**
- **Passo 1: Copiar toda informação(info e prox) do sucessor(xi+1) do iterador para o Nó que ser quer eliminar(nó apontado pelo iterador)**



Lista simplesmente encadeada com Nó Sentinela

- **ELIMINAR um elemento na Lista:**
- **Passo 2: Eliminar o sucessor do iterador e acertar os apontadores**
- **Atenção: se o iterador estiver sobre o último elemento(xn), como foi copiado para ele(info e prox) do próximo nó(sentinela), é preciso verificar se o campo prox do iterador é null(e é de fato), assim basta depois fazer o iterador apontar para null.**