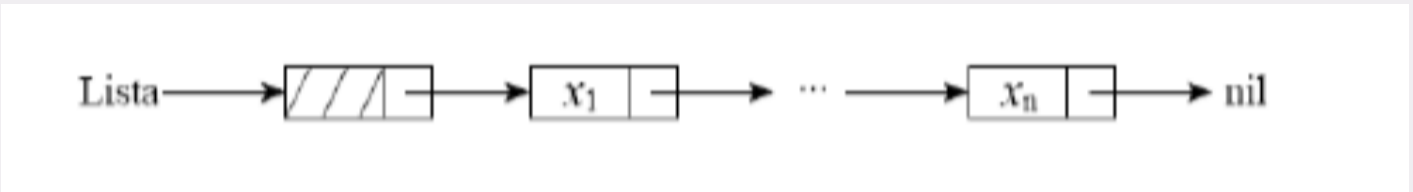
Alocação Encadeada

- Cada item é encadeado como seguinte mediante uma variável do tipo Apontador
- Permite utilizar posições não contiguas de memória
- É possível inserir e retirar elementos sem a necessidadede deslocar os itens seguintes da lista
- Há uma célula cabeça para simplificar as operações sobre a lista

Alocação Encadeada

- A lista é constituída de células ou nós
- Cada nó contém um item da lista e um apontador para a célula seguinte
- O registro TipoLista contém um apontador para o nó cabeça e um apontador para a última célula da lista

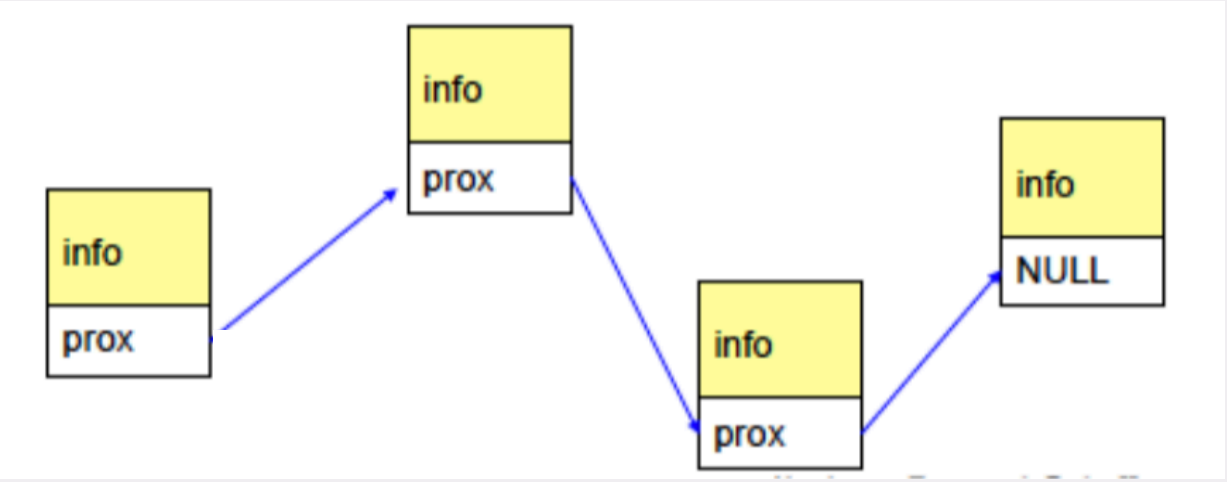


Alocação Encadeada

- Localização na memória
  - Posições não sequenciadas
- Visitas
  - Apenas na direção x_i para x_{i+1}
  - Permite apenas acesso não sequencial
- Inserção
  - Realizada em qualquer posição com custo constante
- Remoção
  - Custo constante em qualquer posição

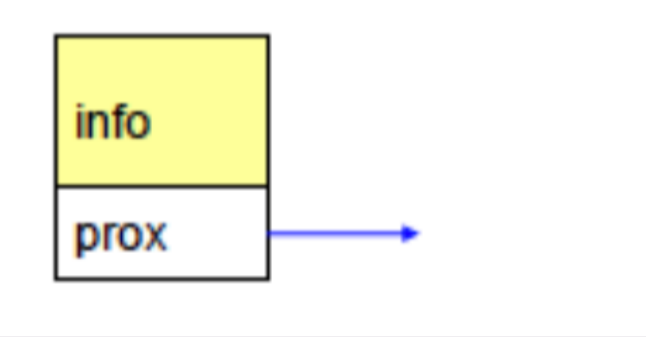
Alocação Encadeada

- Características:
  - Tamanho da lista não é pré-definido
  - Cada elemento guarda quem é o próximo
  - Elementos não estão contíguos na memória



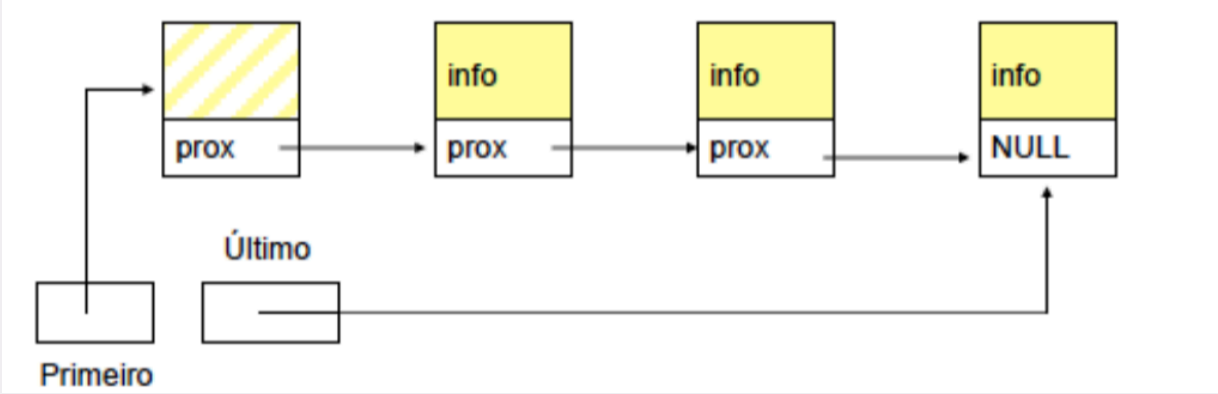
Sobre os Nós (Elementos) da Lista

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



Sobre a Lista

- Uma lista é definida como um apontador para a primeira célula
- Uma lista pode ter uma célula cabeça



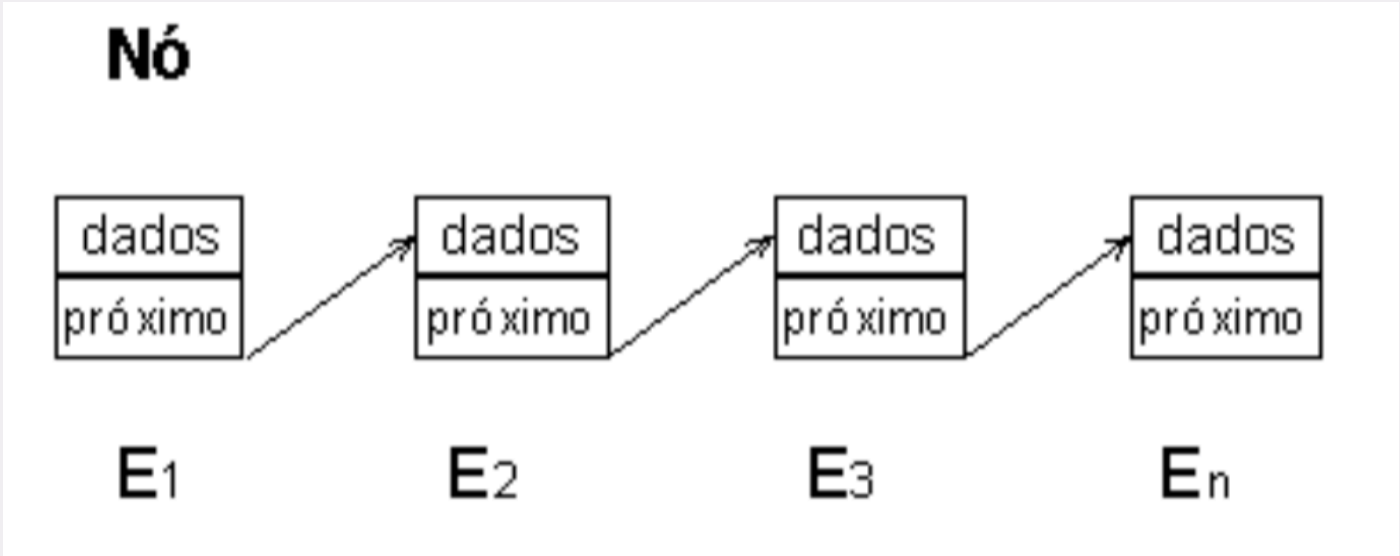
- Uma lista pode ter um apontador para o ultimo elemento

Tipos de Listas Encadeadas

- **Simplesmente encadeada**
- **Duplamente encadeada**
- Ordenadas
- Circulares

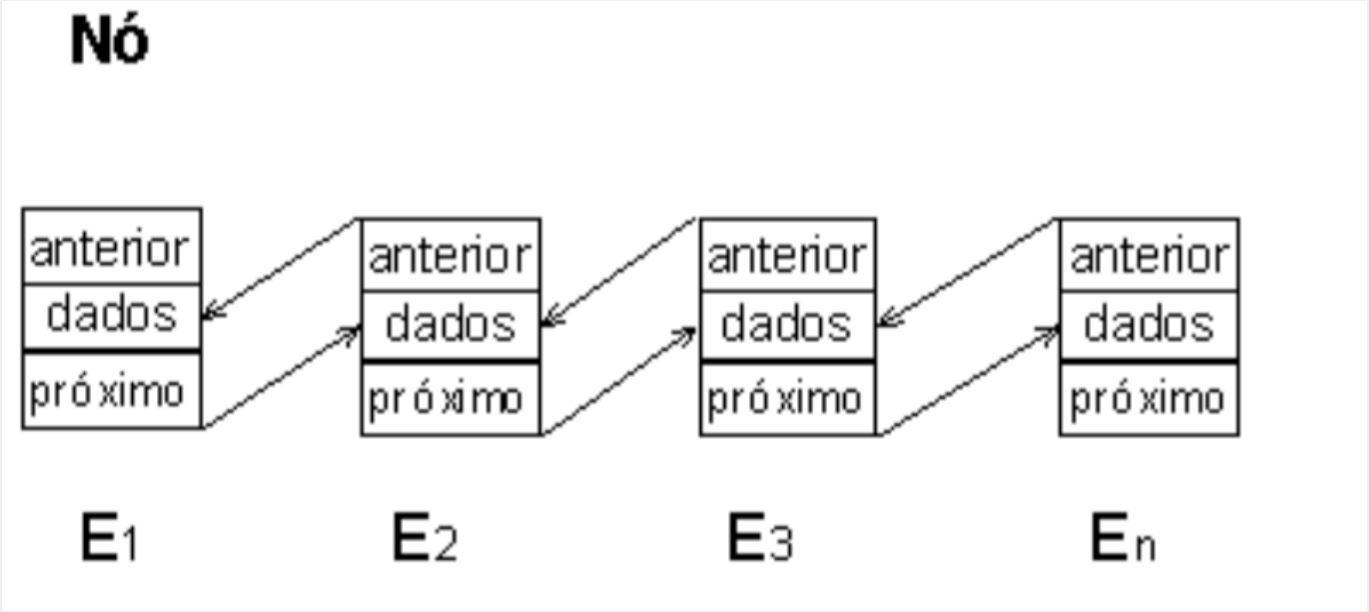
Listas Simplesmente Encadeadas

- Os elementos da lista possuem apenas um ponteiro que aponta para o elemento sucessor ou próximo.



Listas Duplamente Encadeadas

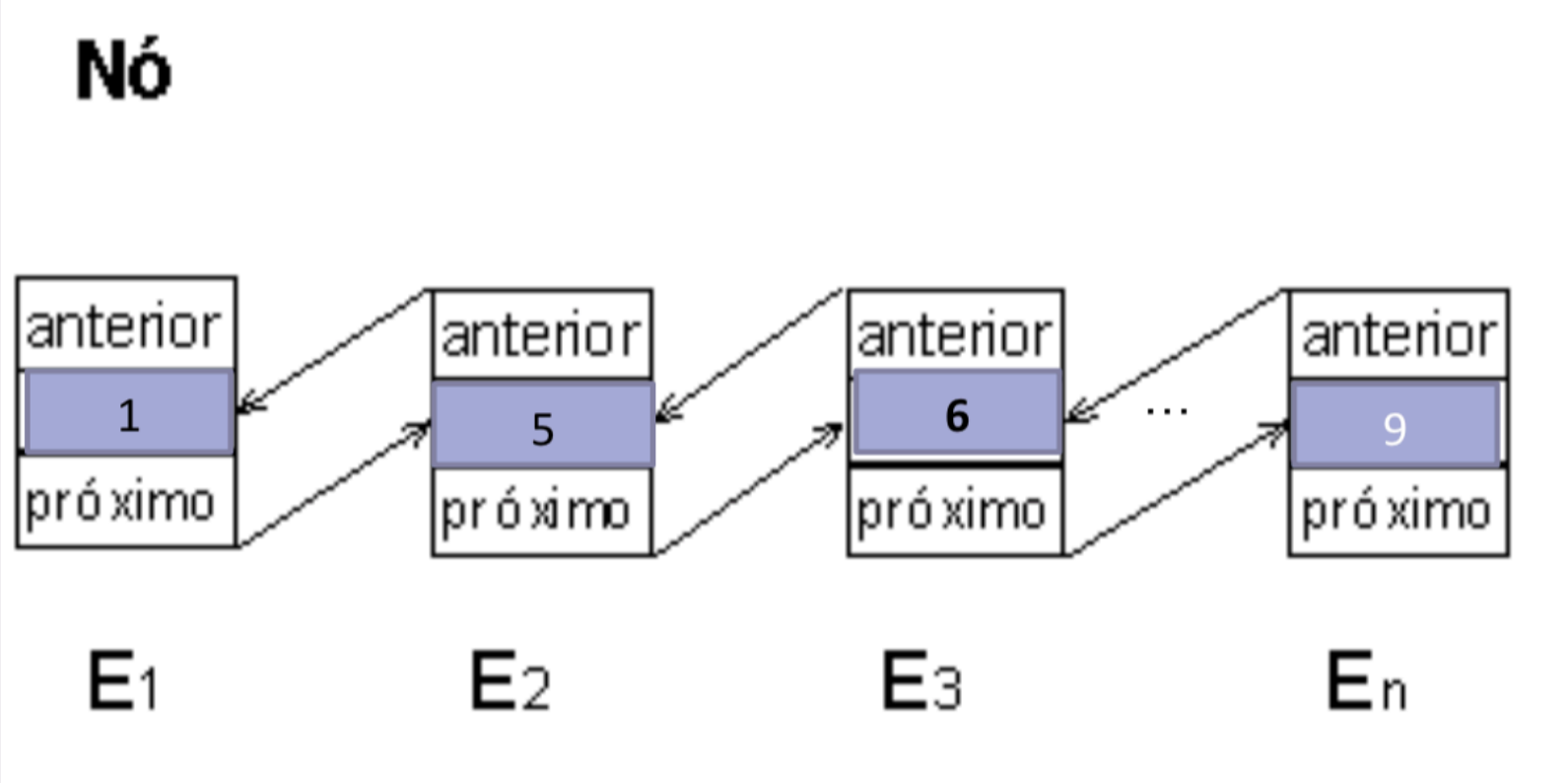
- Cada elemento possui um campo que aponta para o seu predecessor (anterior) e outro para o seu sucessor (próximo):



Listas Ordenadas

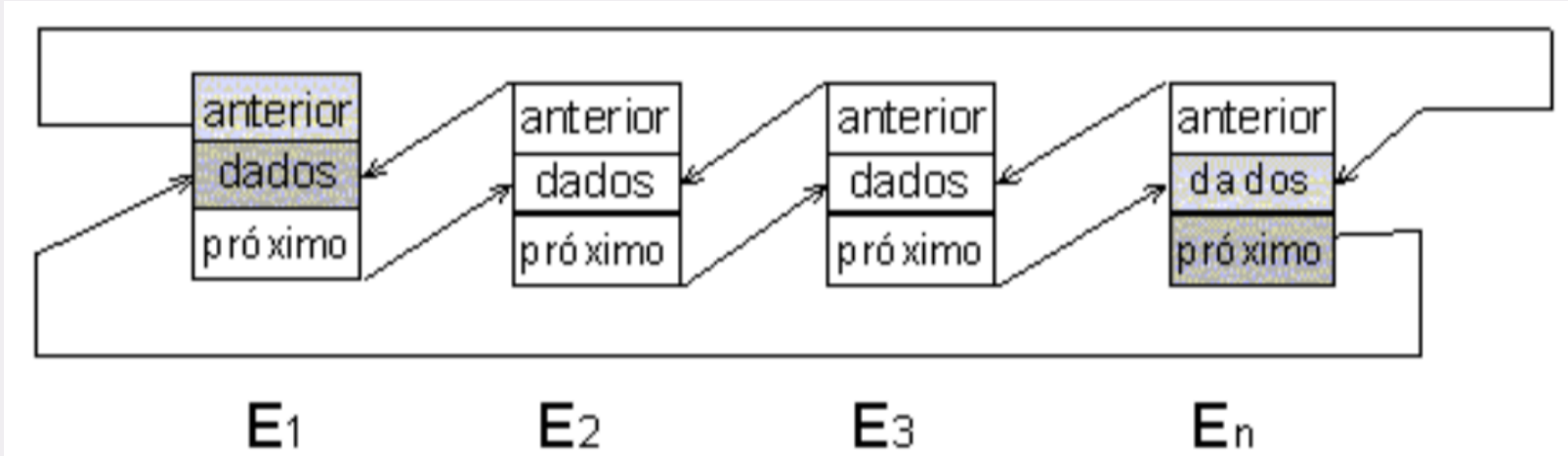
- A ordem linear da lista obedece a ordem linear dos elementos. Ou seja, ao inserir um novo elemento, o mesmo deve ser colocado em uma posição que garanta que a ordem da lista será mantida.
- [1, 5, 7, 9]->[1, 5, **6**, 7, 9]

Listas Ordenadas



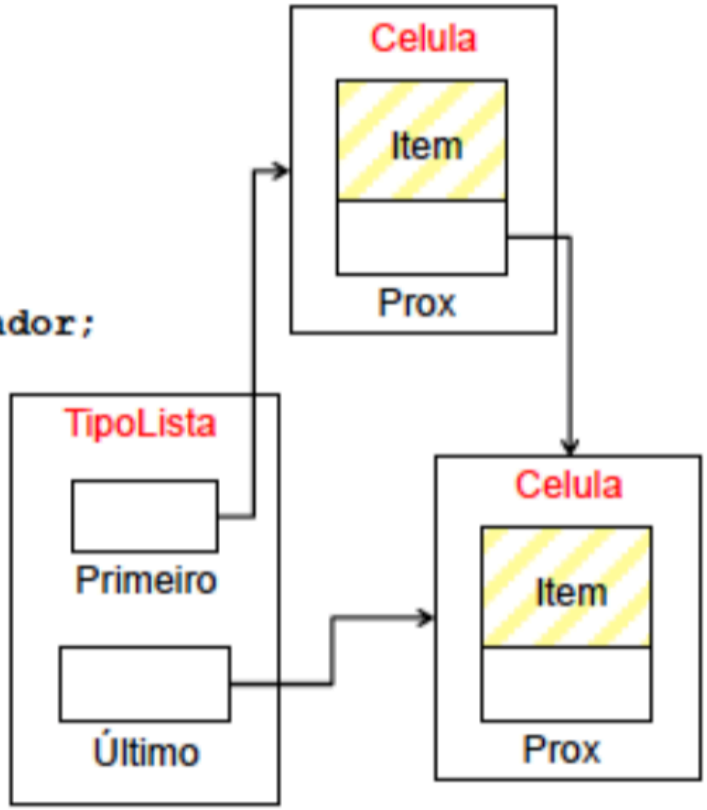
Listas Circulares

- O ponteiro próximo do último elemento aponta para o primeiro; e o ponteiro anterior do primeiro elemento aponta para o último.

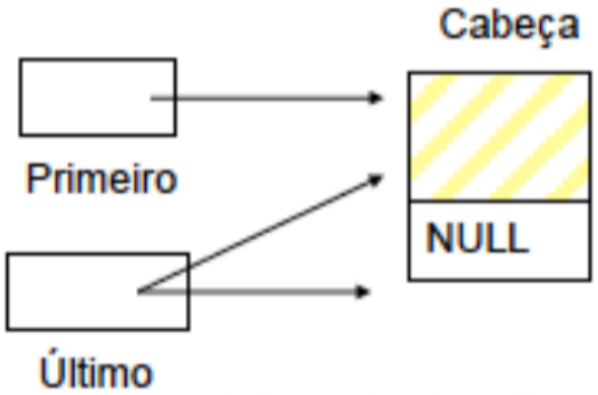


Implementação de Listas Simplesmente encadeadas

```
typedef int TipoChave;

typedef struct {
  TipoChave Chave;
  /* outros componentes */
} TipoItem;

typedef struct Celula_str *Apontador;

typedef struct Celula_str {
  TipoItem Item;
  Apontador Prox;
} Celula;

typedef struct {
  Apontador Primeiro, Ultimo;
} TipoLista;
```

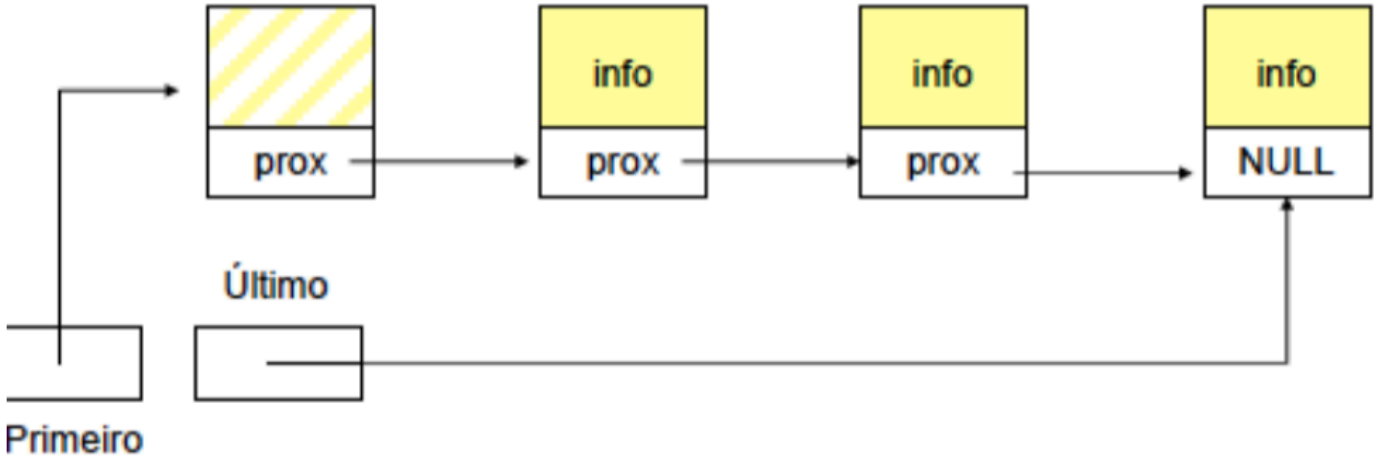


Cria Lista Vazia



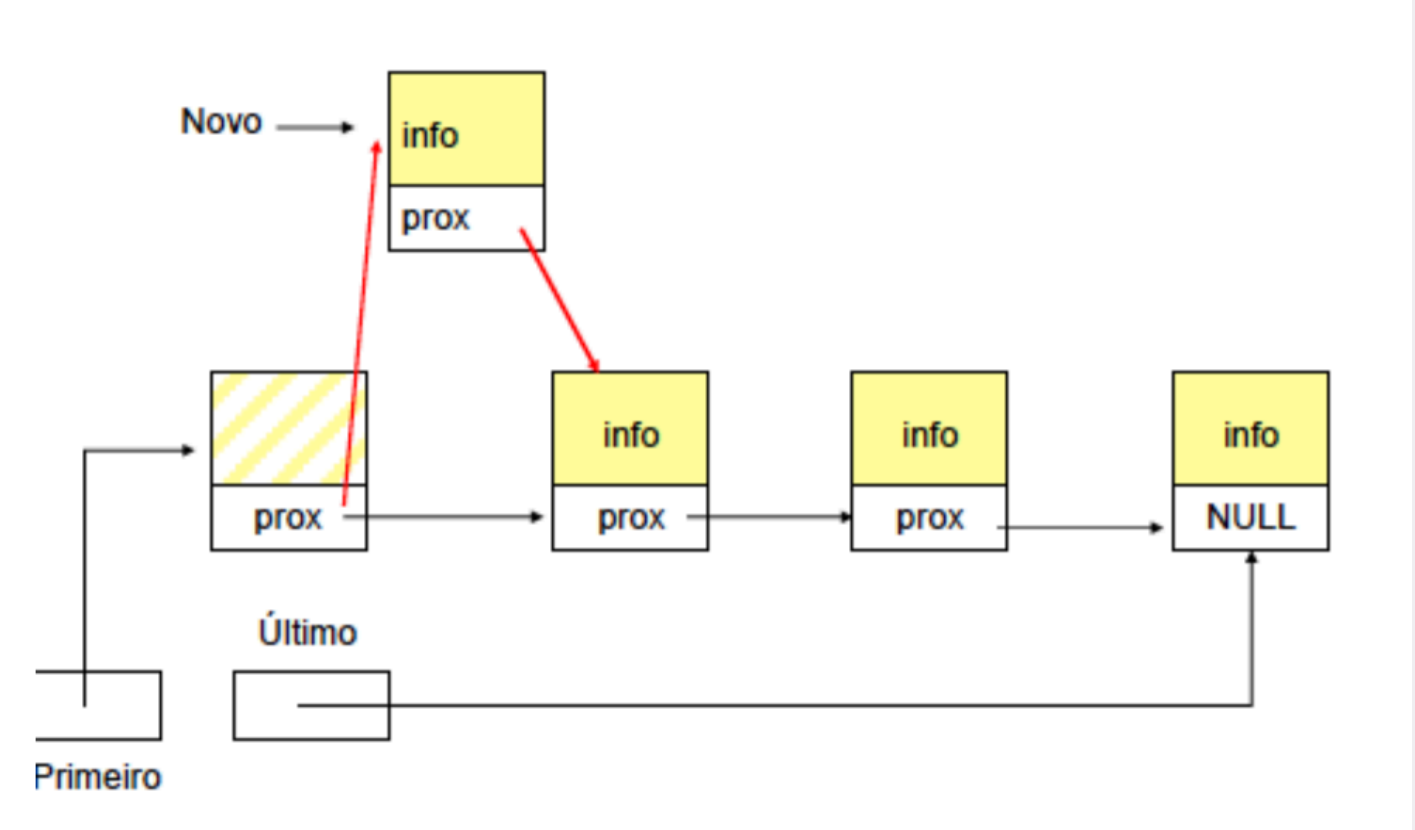

```
void FLVazia(TipoLista *Lista)
{
  Lista->Primeiro = (Apontador) malloc(sizeof(Celula));
  Lista->Ultimo = Lista->Primeiro;
  Lista->Primeiro->Prox = NULL;
}

int Vazia(TipoLista Lista)
{
  return (Lista.Primeiro == Lista.Ultimo);
}
```

Inserção de Elementos na Lista

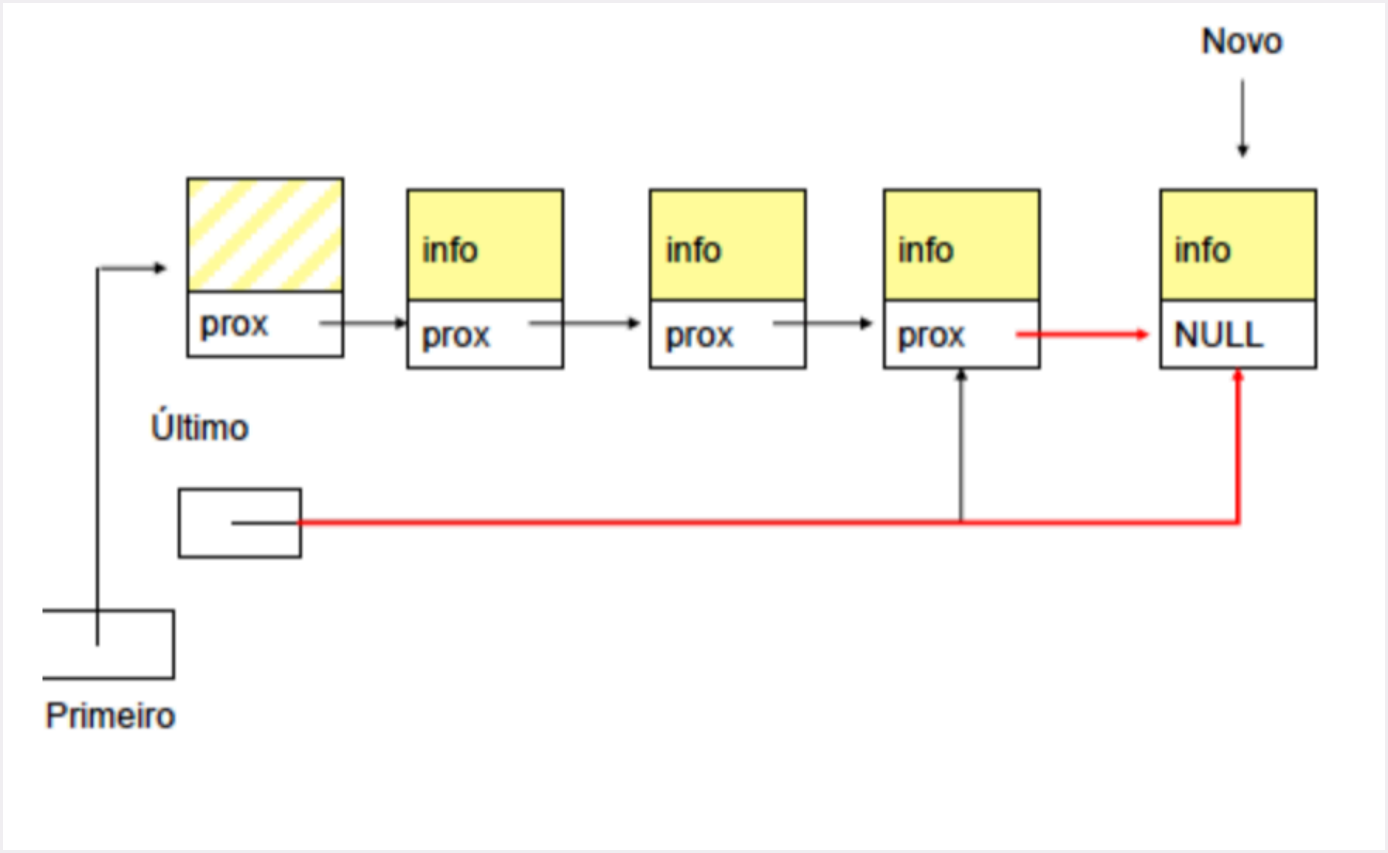


- 3 opções de posição onde pode inserir:
  - 1ª. posição
  - última posição
  - Após um elemento qualquer E

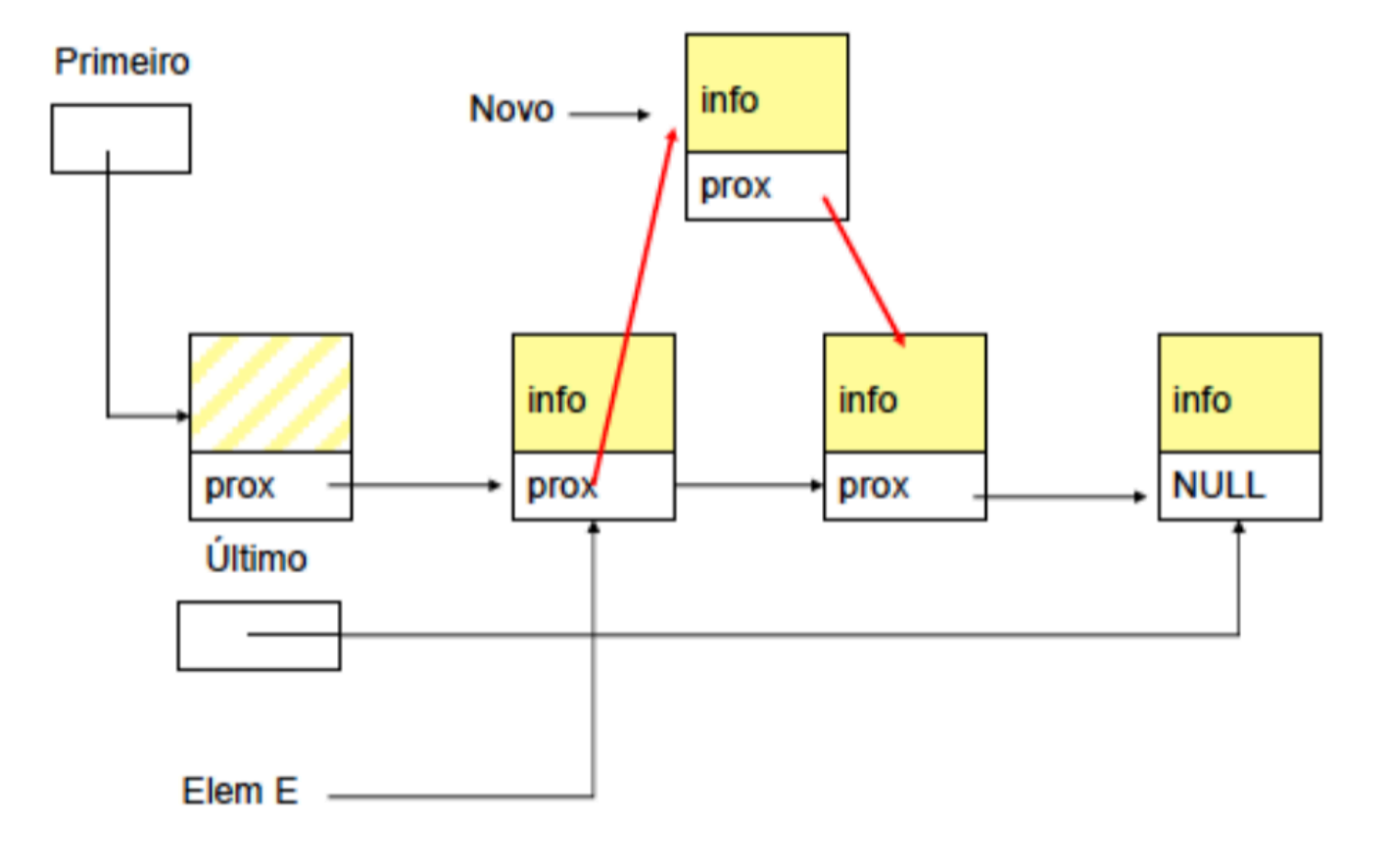
Inserir na Primeira Posição



Inserir na última posição



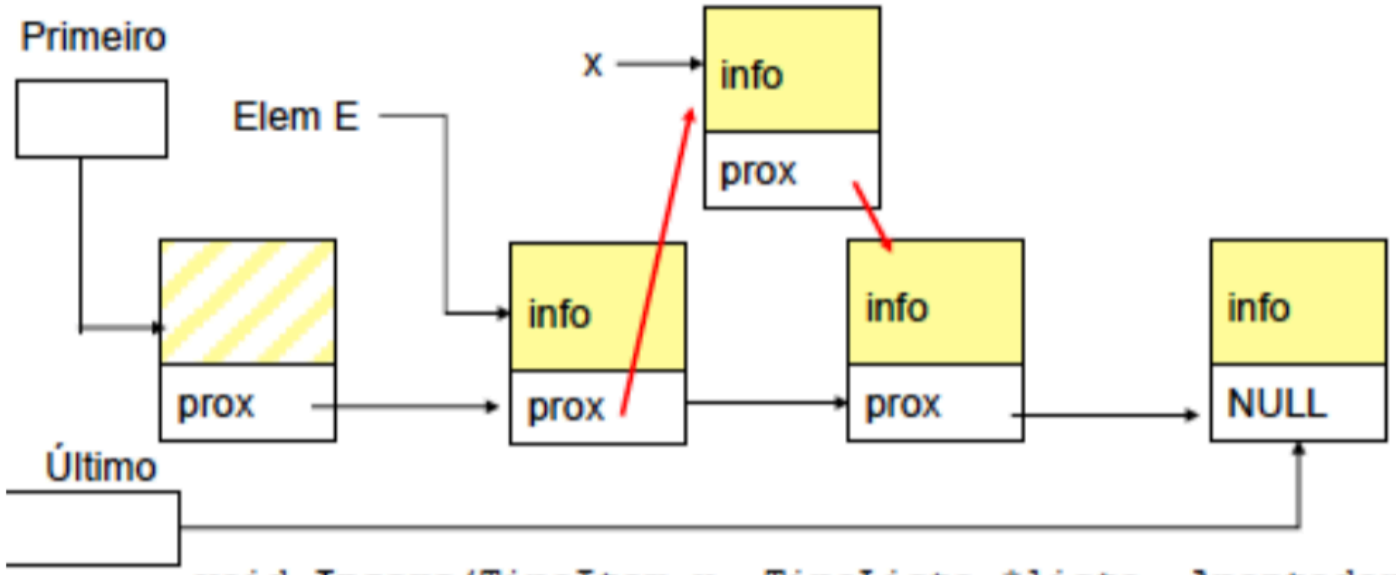
Inseir Após o Elemento E



Inserção de Elementos

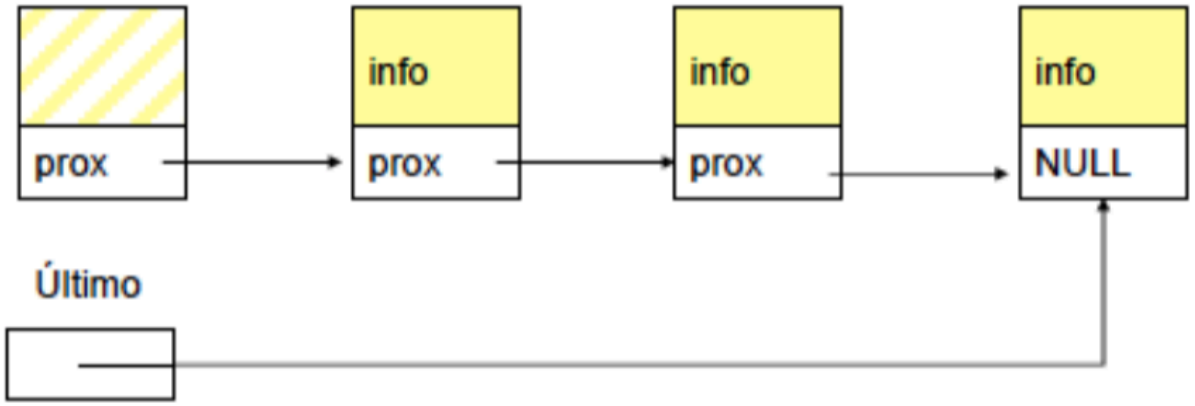
- Na verdade, as 3 operações de inserção são equivalentes a inserir após uma célula apontada por p
  - 1° posição (p é a célula cabeça)
  - Última posição (p é o último)
  - Após um elemento qualquer E (p aponta para E)

Inserção após o Elemento E



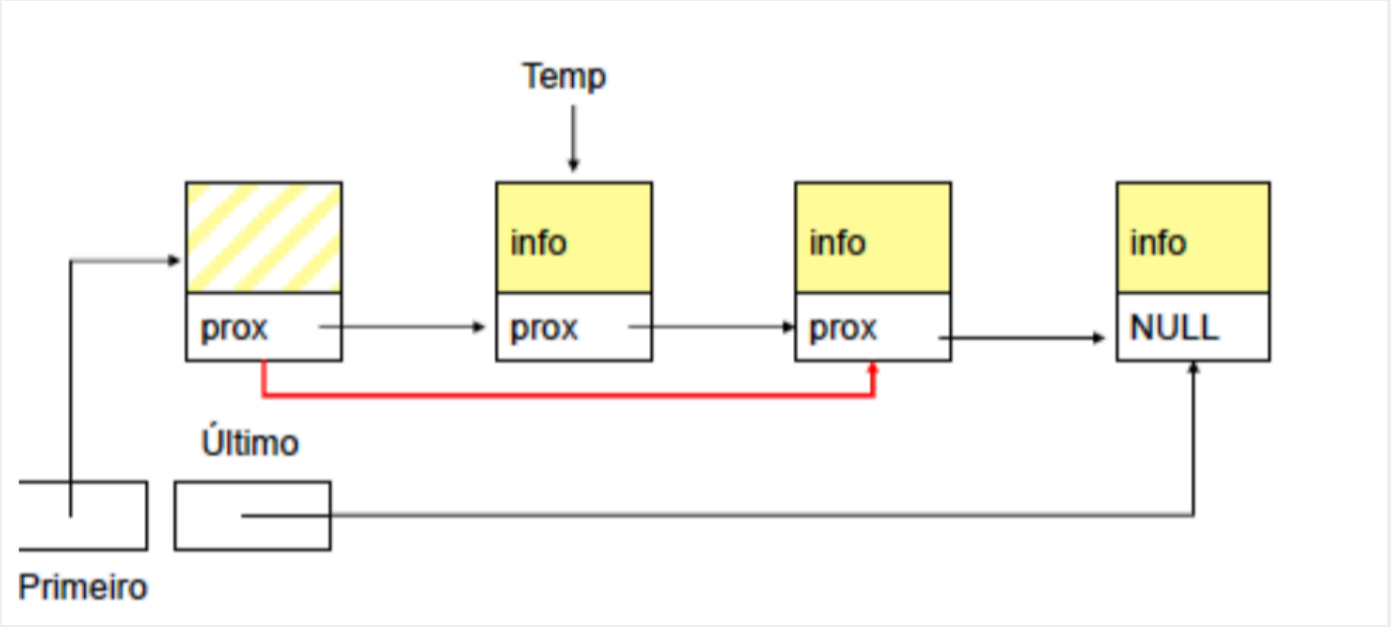

```
void Insere(TipoItem x, TipoLista *lista, Apontador E){
Apontador novo;

   novo = (Apontador) malloc(sizeof(Celula));
   novo->Item = x;
   novo->prox = E->prox;
   E->prox = novo;
}
```

Remover Elementos

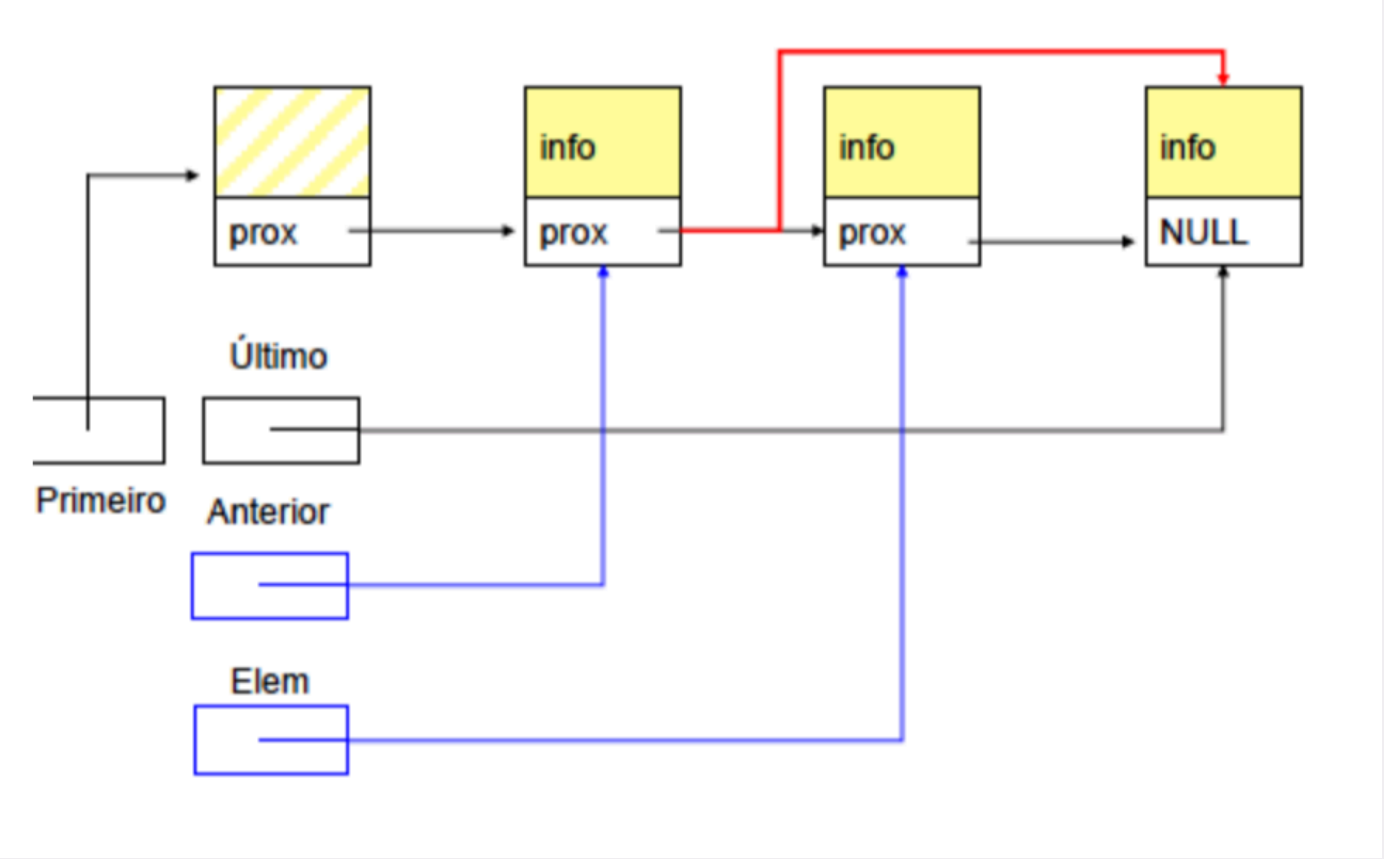


- 3 opções de posição de onde pode retirar:
  - 1ª. posição
  - última posição
  - Um elemento qualquer E

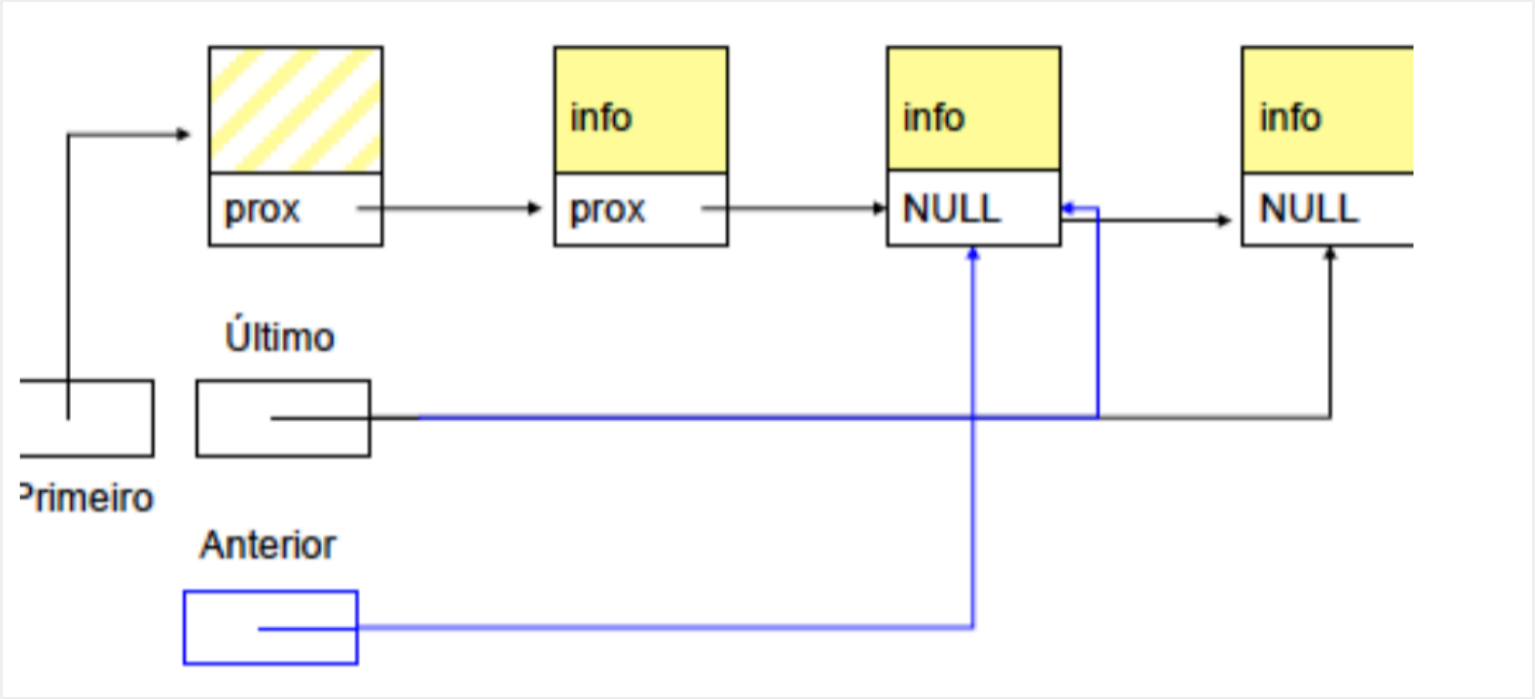
Remover o Primeiro Elemento



Remover um elemento qualquer



Remover o ultimo Elemento



Listas Usando Encadeamento – Vantagens e Desvantagens

- 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.

Exercício de Fixação

- Implementa uma lista simplesmente encadeada
- Implemente um método para remover o primeiro elemento de uma lista simplesmente encadeada
- Implemente um método para remover o último elemento de uma lista simplesmente encadeada
- Implemente um método para inserir no inicio de uma lista simplesmente encadeada
- Implemente um método para impressão dos elementos da lista
- Implemente um Lista duplamente encadeada

Exercício de Fixação

- Dada uma lista L1 ordenada encadeada simplesmente, escreva as operações:
  - Verifica se L1 está ordenada ou não (a ordem pode ser crescente ou decrescente)
  - Faça uma cópia da lista L1 em uma outra lista L2
  - Faça uma cópia da Lista L1 em L2, eliminando elementos repetidos
  - inverta L1 colocando o resultado em L2