Estrutura de Dados

Aula 04 – Lista dinâmica

Prof. Dr. Daniel Vecchiato

Agenda

- Introdução
- Lista encadeada com cabeça
- Lista encadeada sem cabeça
- Exercícios

Introdução

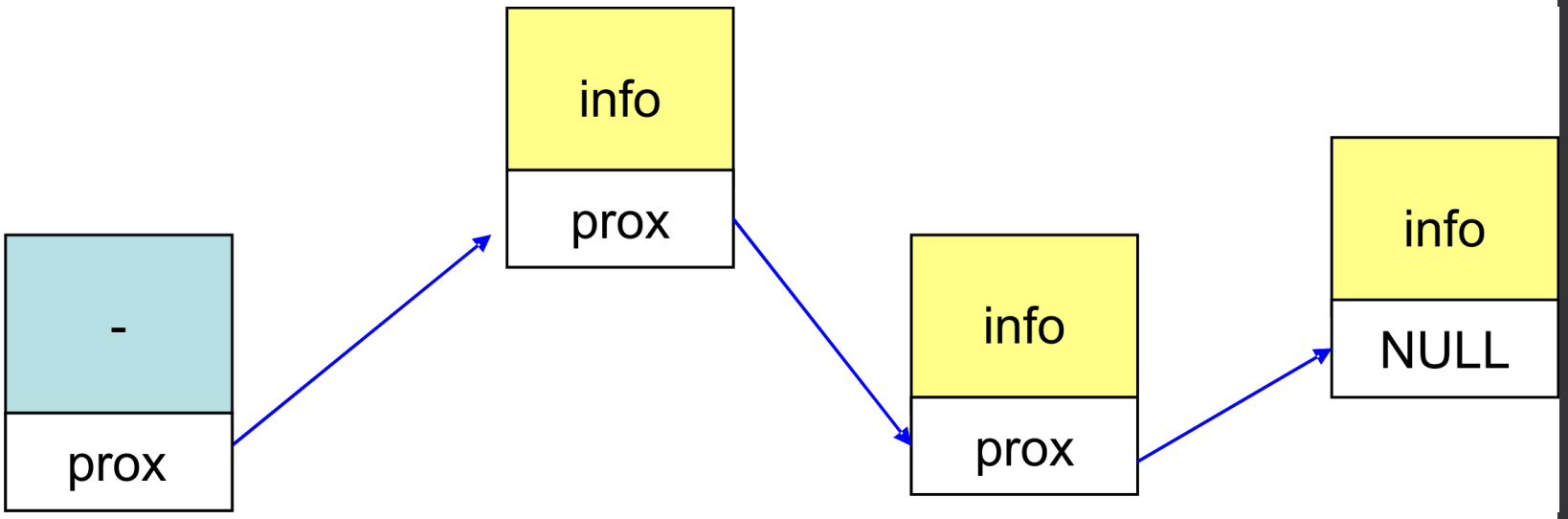
- Lista dinâmica
  - Implementação de lista utilizando ponteiros e alocação dinâmica de memória
  - Também chamada de lista encadeada ou lista simplesmente encadeada

- Tamanho da lista não é pré-definido
  - Nas listas estáticas, há o problema quando a lista diminui drasticamente de tamanho
    - Listas dinâmicas não possuem essa desvantagem

- Algumas variações
  - Lista encadeada com cabeça
  - Lista encadeada sem cabeça
  - E várias outras...

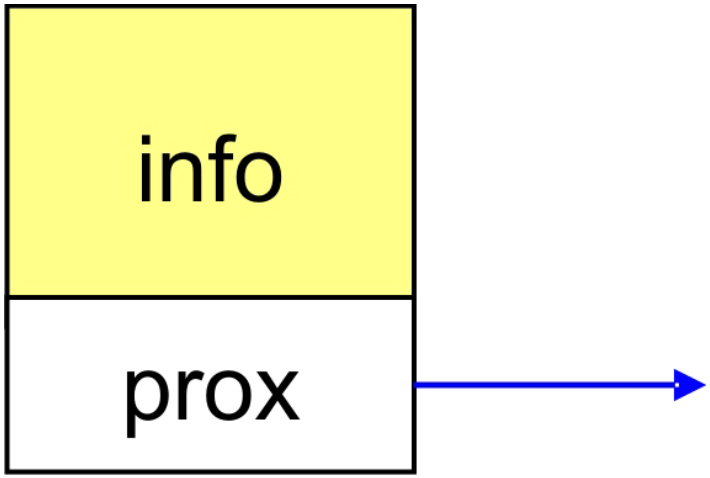
Introdução

- Cada elemento aponta para o próximo elemento
- Elementos não estão contíguos na memória
  - alocação dinâmica



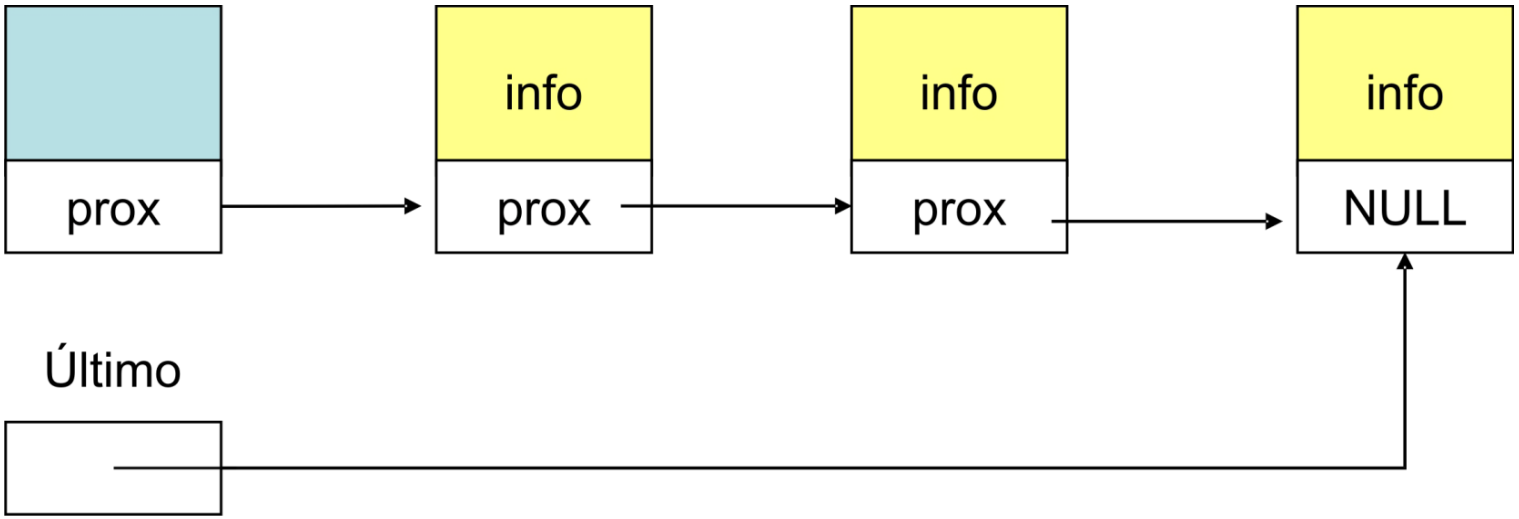
Introdução

- A estrutura de um elemento possui:
  - Campos representando as informações do elemento
  - Ponteiro para o próximo elemento



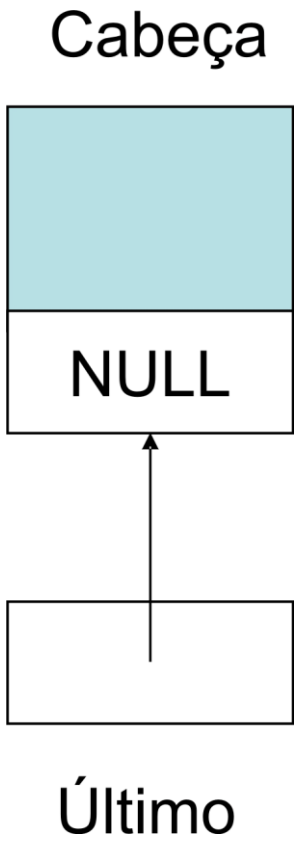
Lista encadeada com cabeça

- Possui uma cabeça, antecedendo o primeiro elemento
  - Aponta para o início da lista
- Possui uma apontador para o último elemento



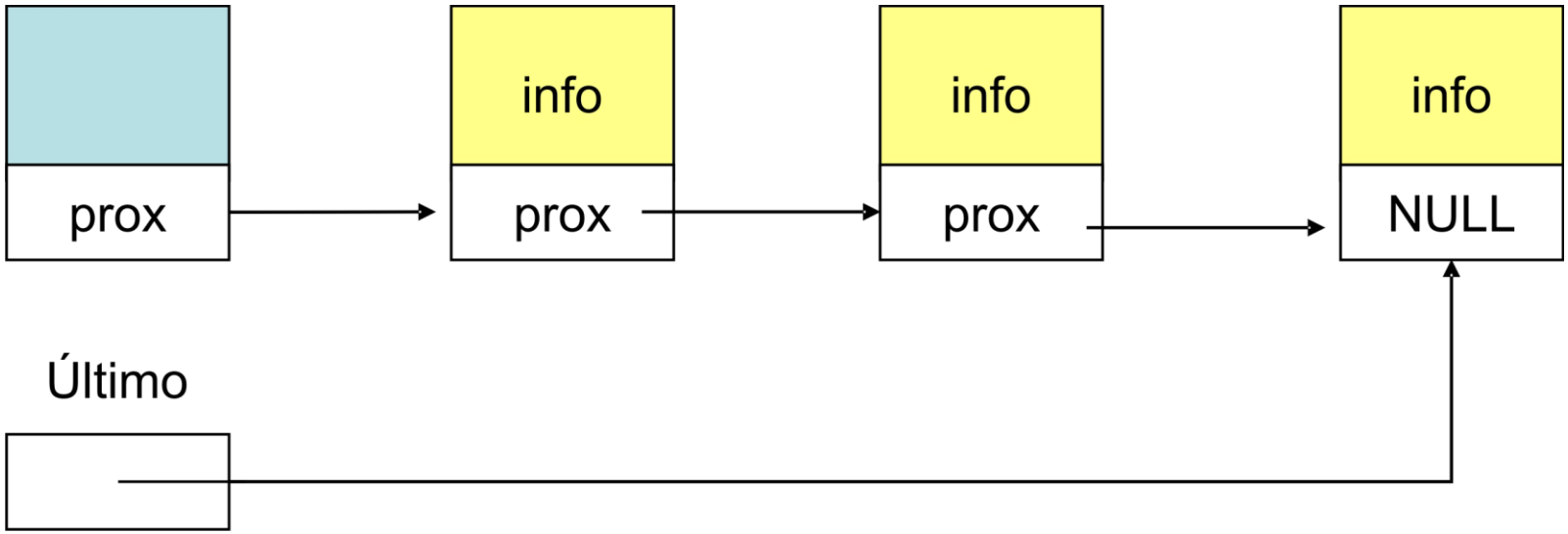
Lista encadeada com cabeça

- Lista vazia



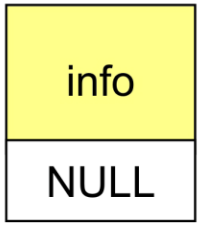
Lista encadeada com cabeça

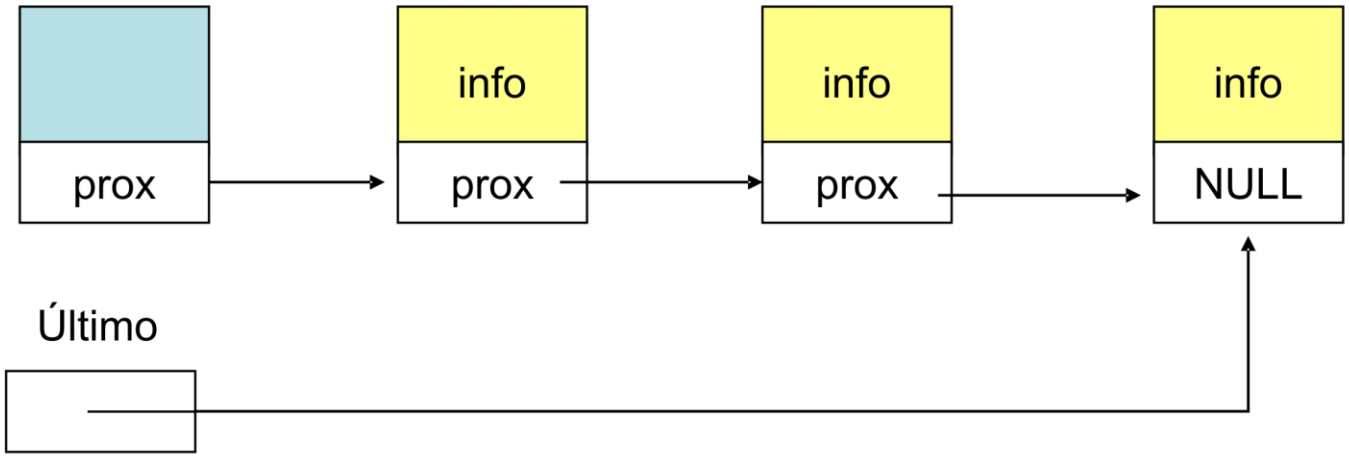
- Inserção de novos elementos
  - 3 possíveis casos
    - Na primeira posição
    - Na última posição
    - Após um elemento E



Lista encadeada com cabeça

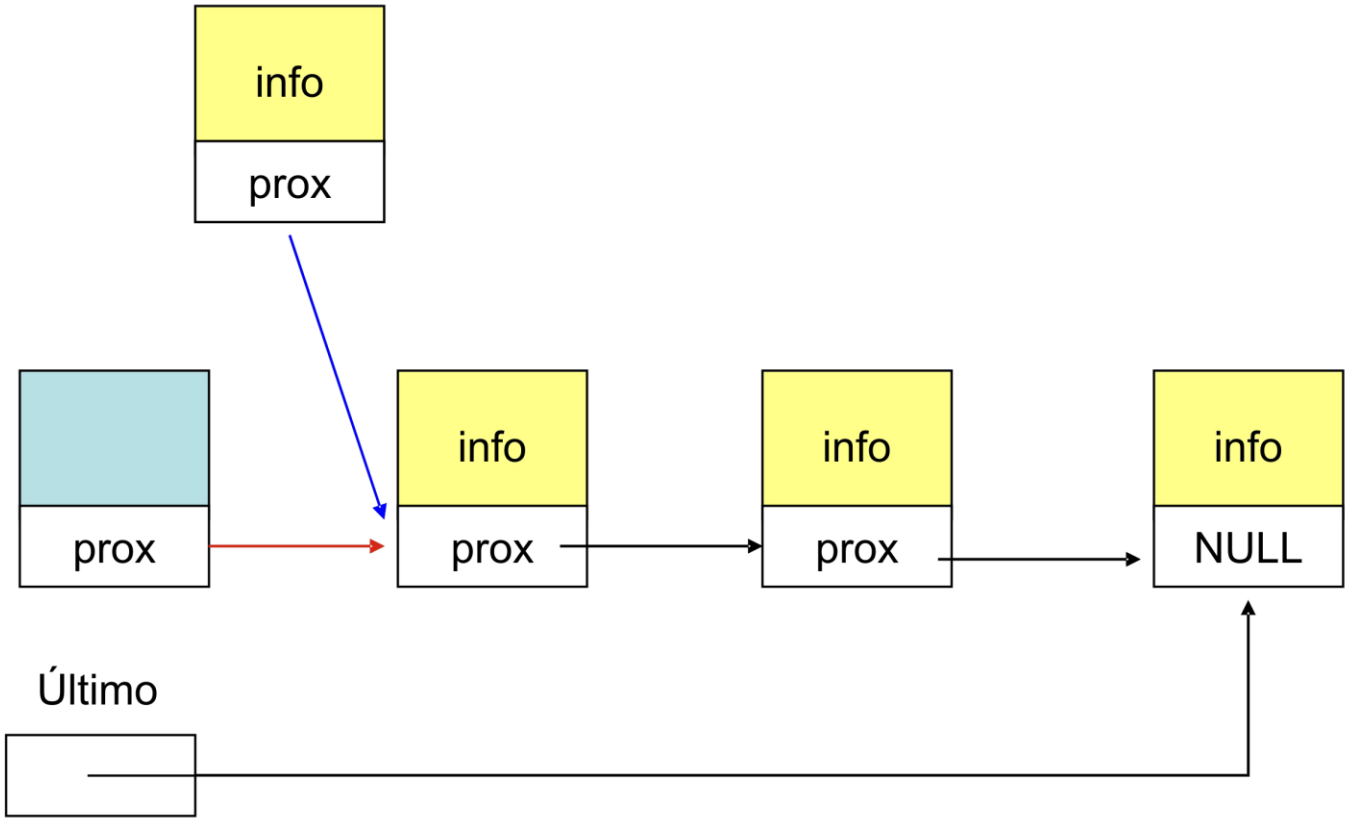
- Inserção na primeira posição





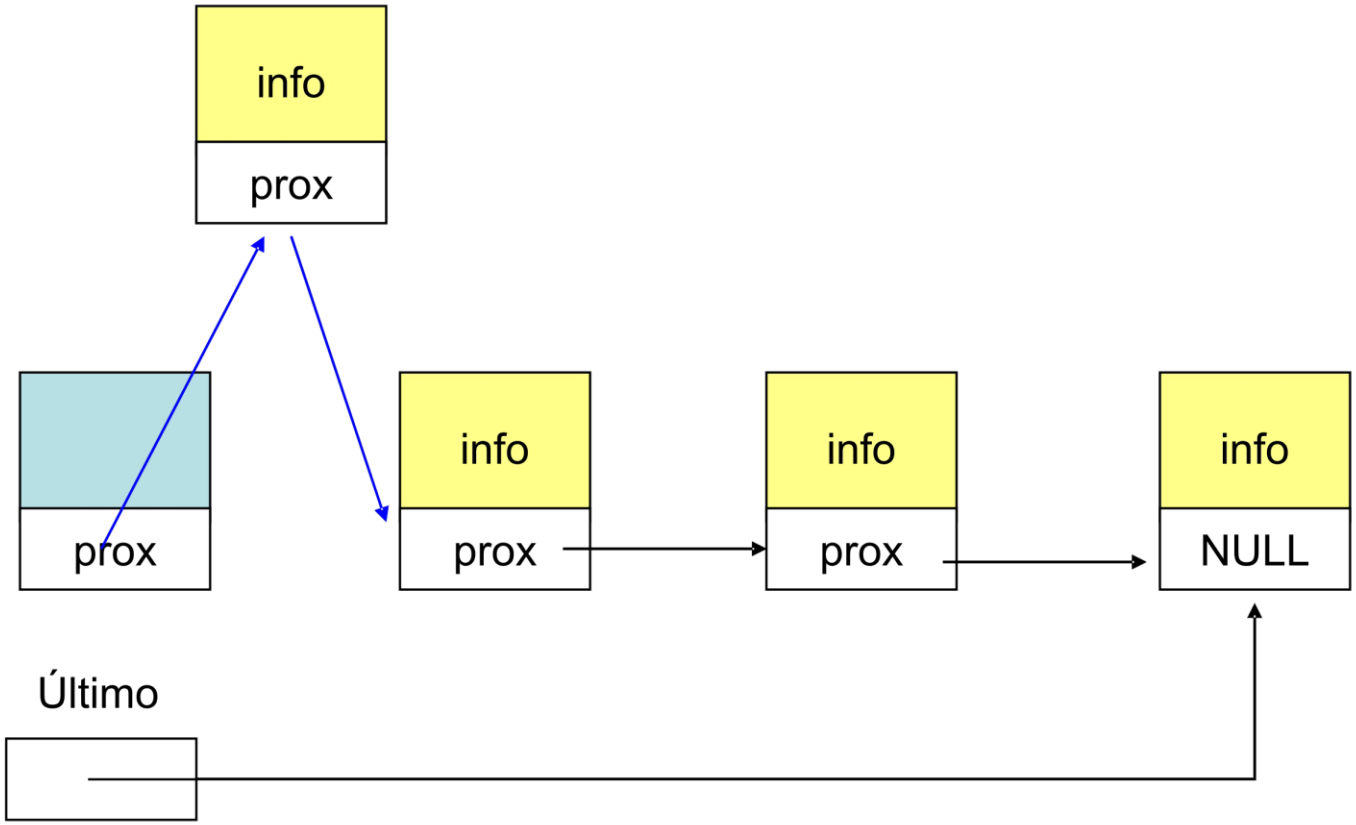
Lista encadeada com cabeça

- Inserção na primeira posição



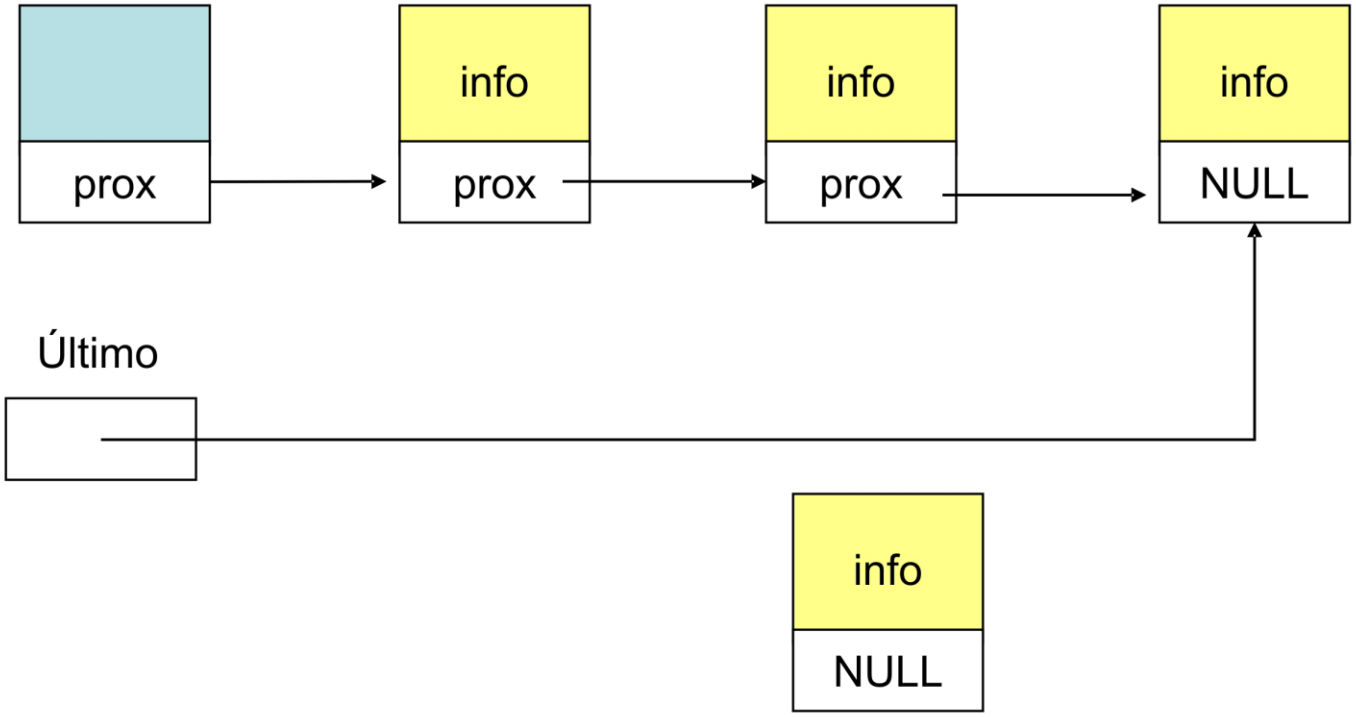
Lista encadeada com cabeça

- Inserção na primeira posição



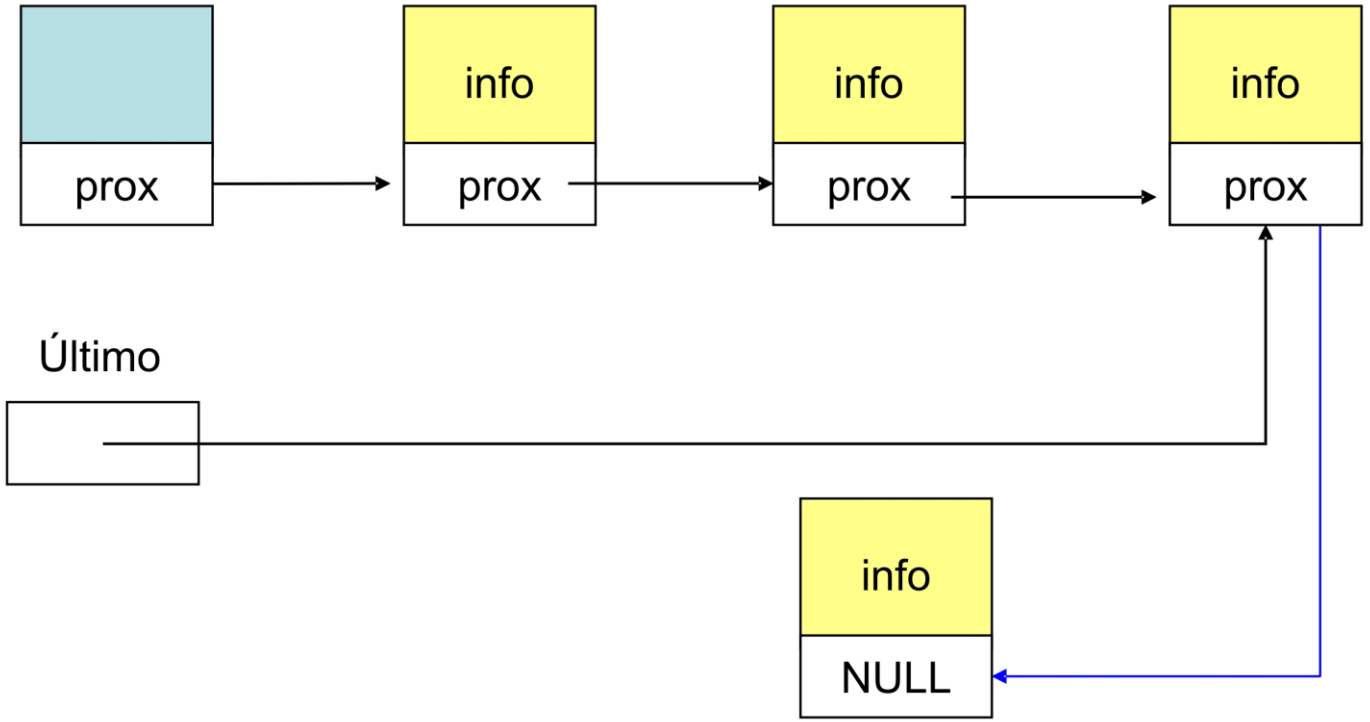
Lista encadeada com cabeça

- Inserção na última posição



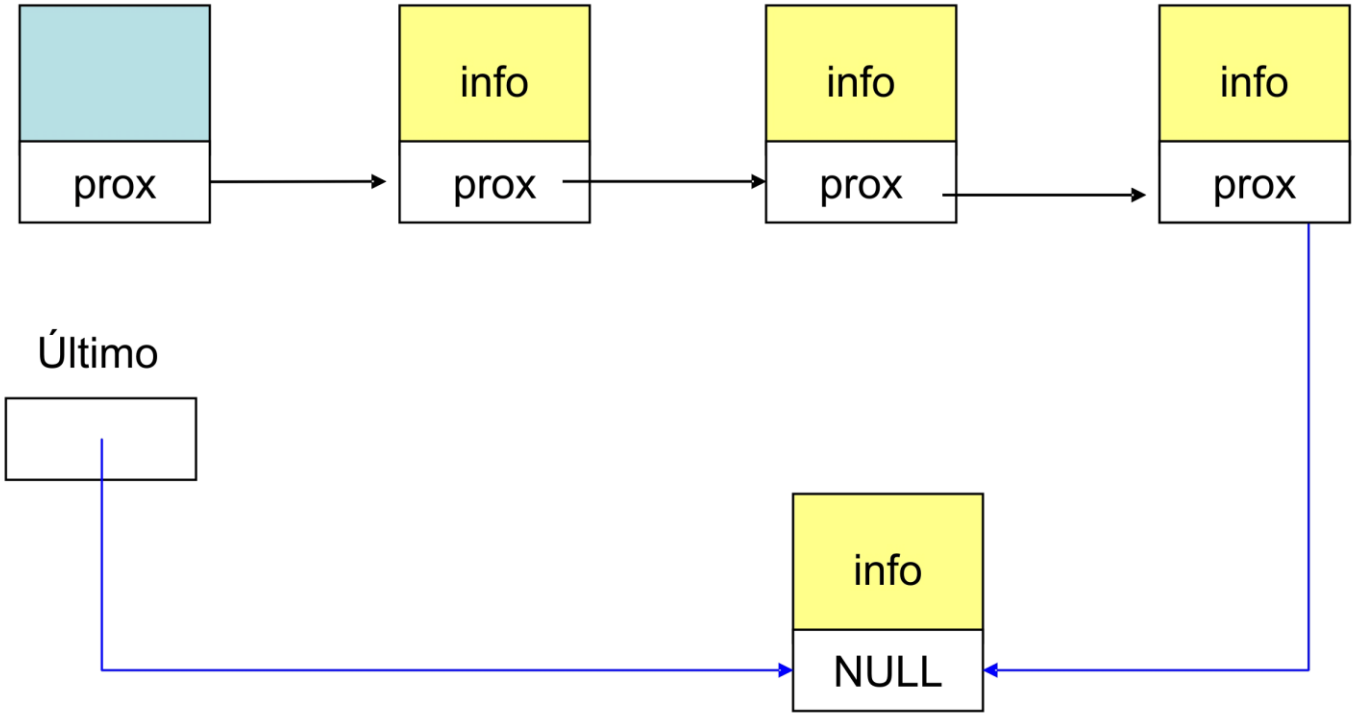
Lista encadeada com cabeça

- Inserção na última posição



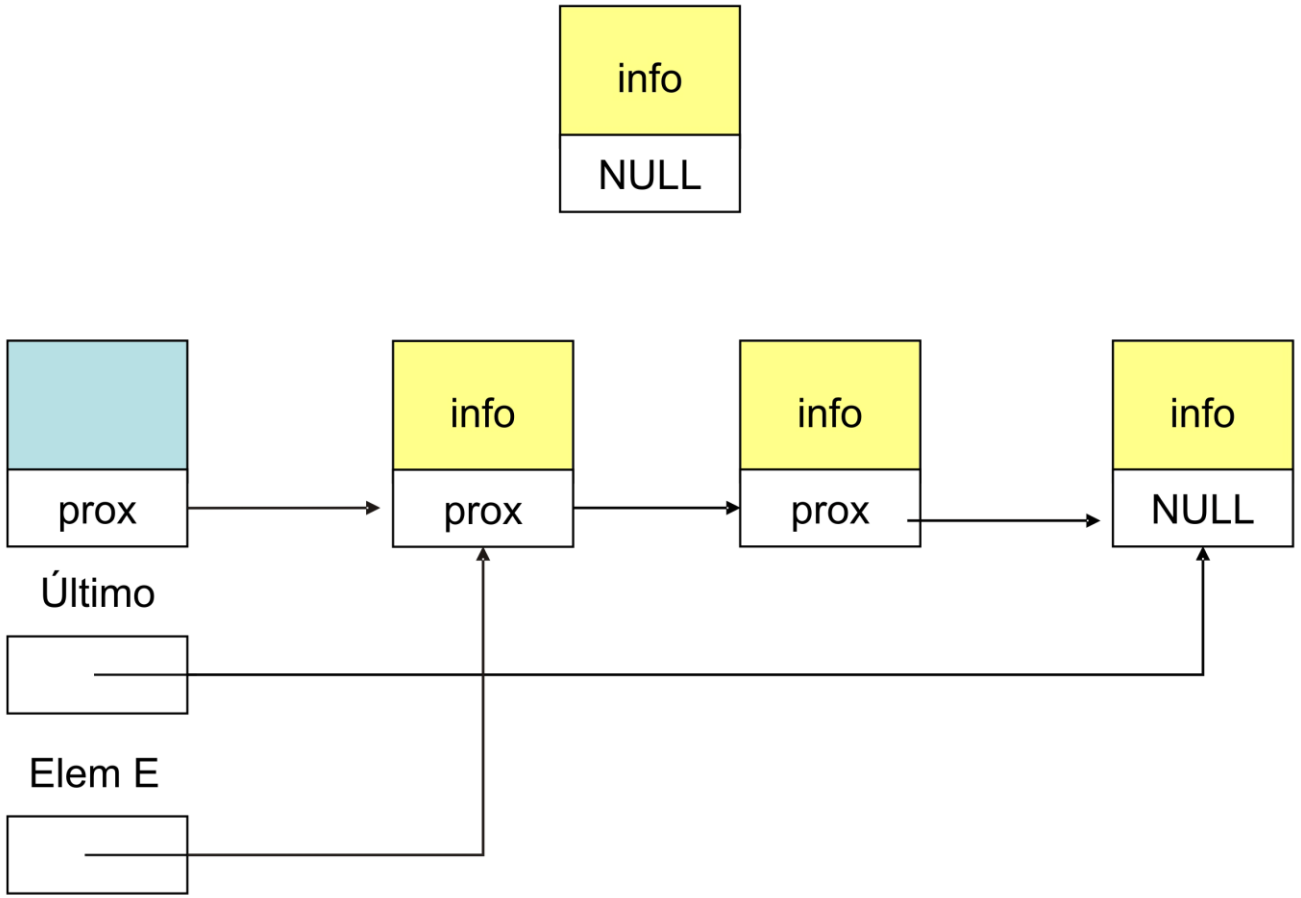
Lista encadeada com cabeça

- Inserção na última posição



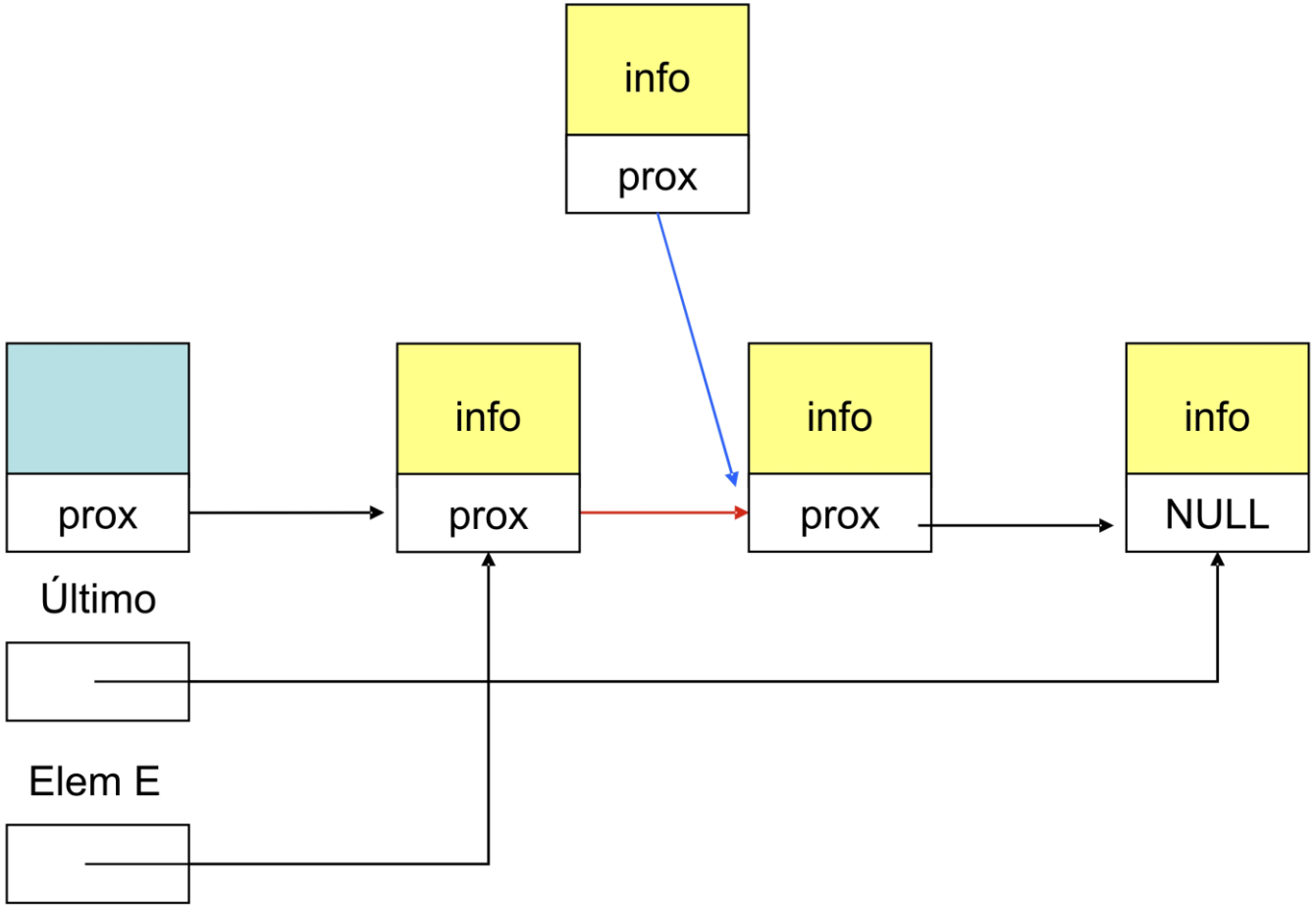
Lista encadeada com cabeça

- Inserção após um elemento E



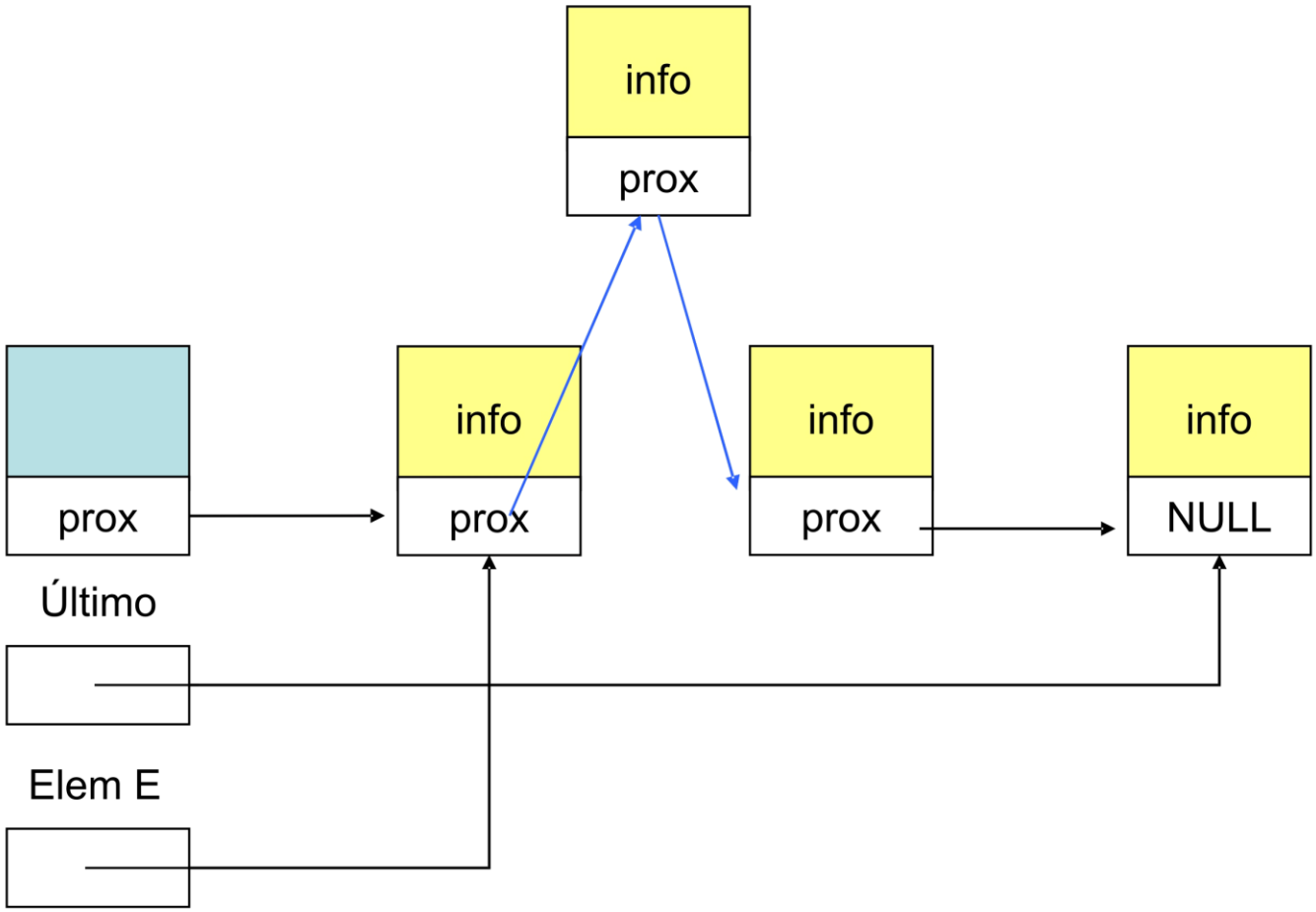
Lista encadeada com cabeça

- Inserção após um elemento E



Lista encadeada com cabeça

- Inserção após um elemento E

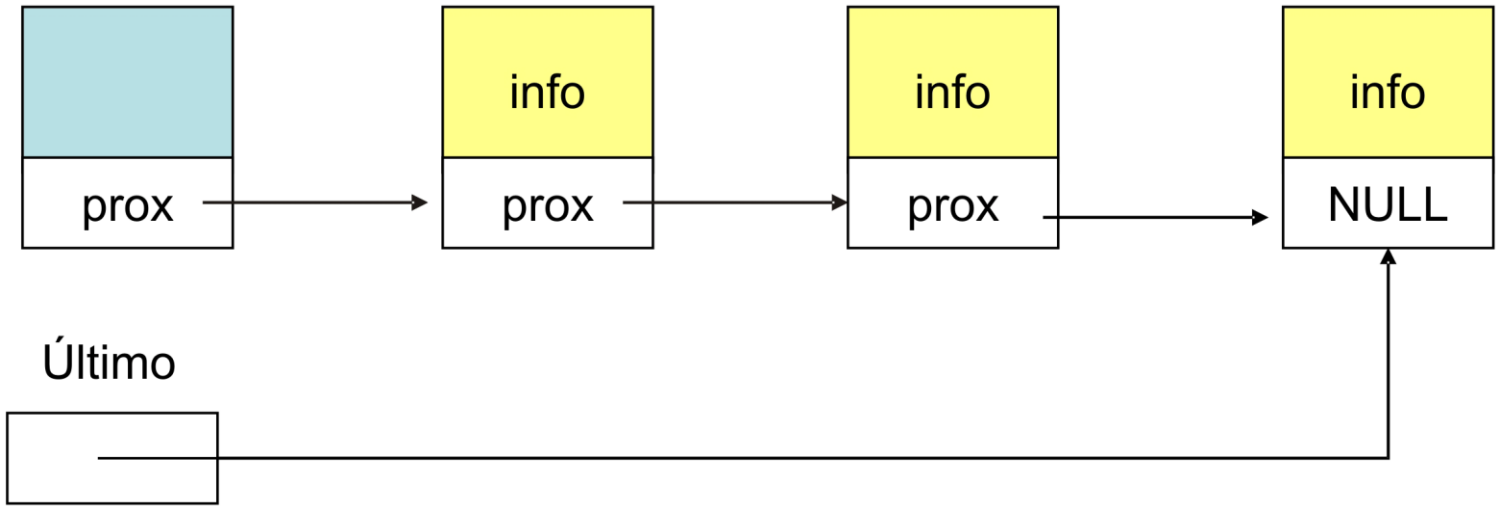


Lista encadeada com cabeça

- Remoção de elementos
  - Os mesmos 3 possíveis casos
    - Na primeira posição
    - Na última posição
    - Um elemento E

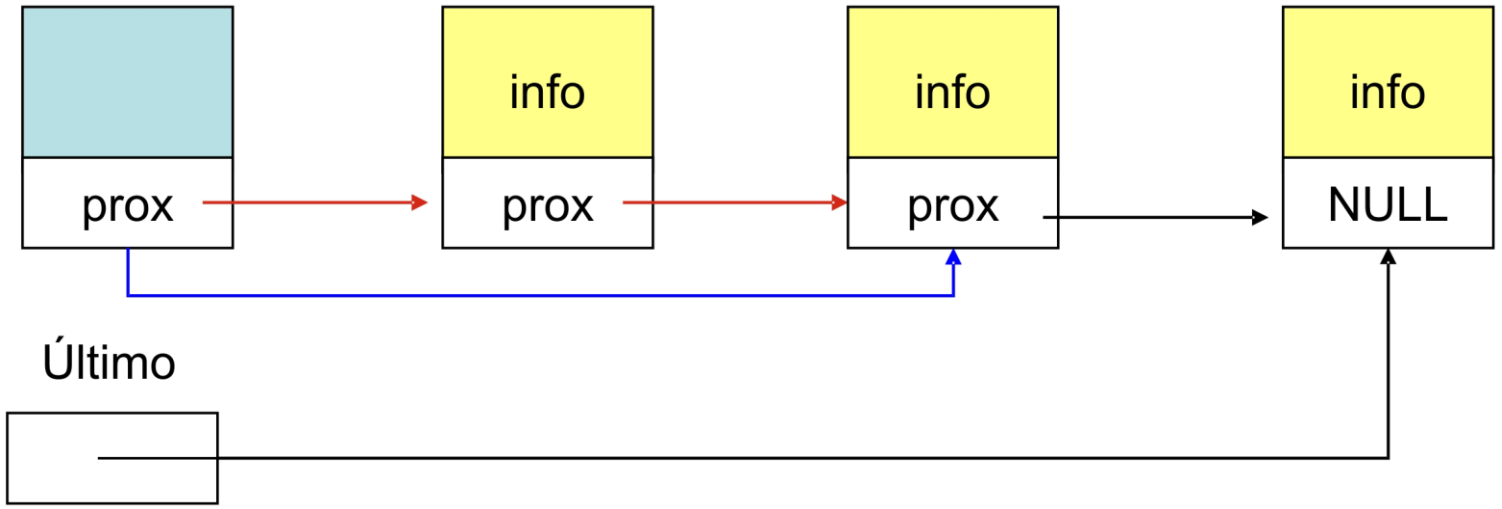
Lista encadeada com cabeça

- Remoção na primeira posição



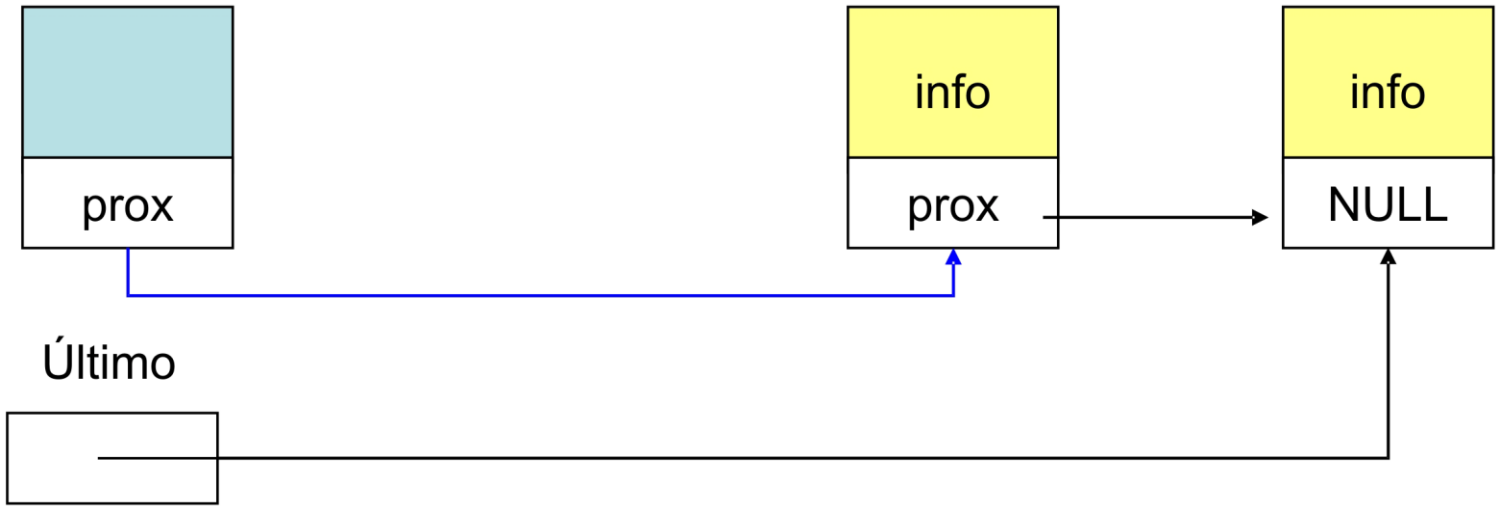
Lista encadeada com cabeça

- Remoção na primeira posição



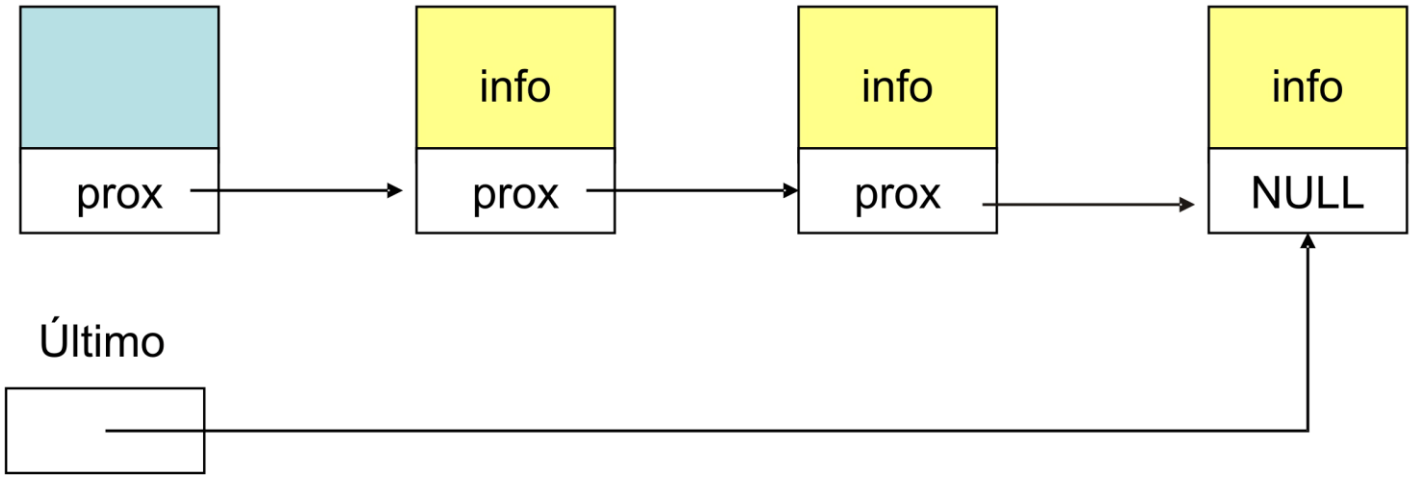
Lista encadeada com cabeça

- Remoção na primeira posição



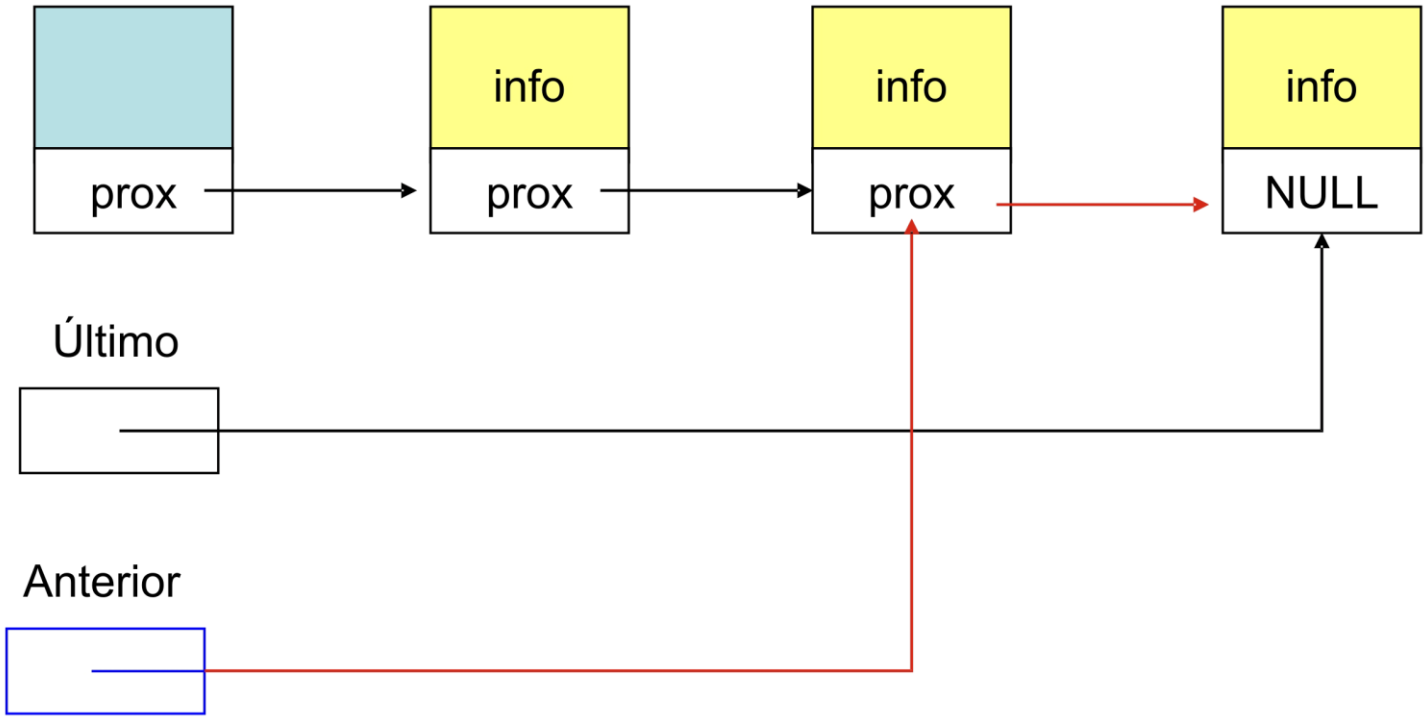
Lista encadeada com cabeça

- Remoção na última posição



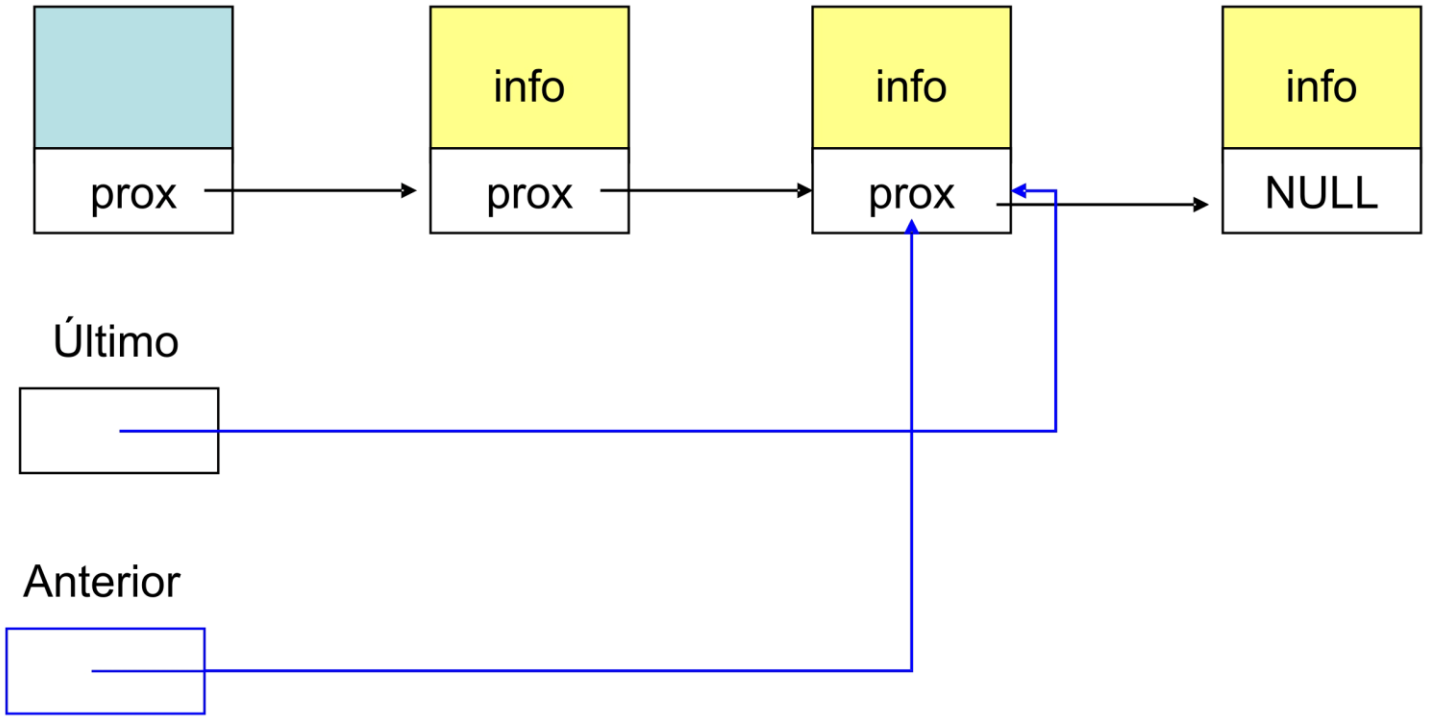
Lista encadeada com cabeça

- Remoção na última posição



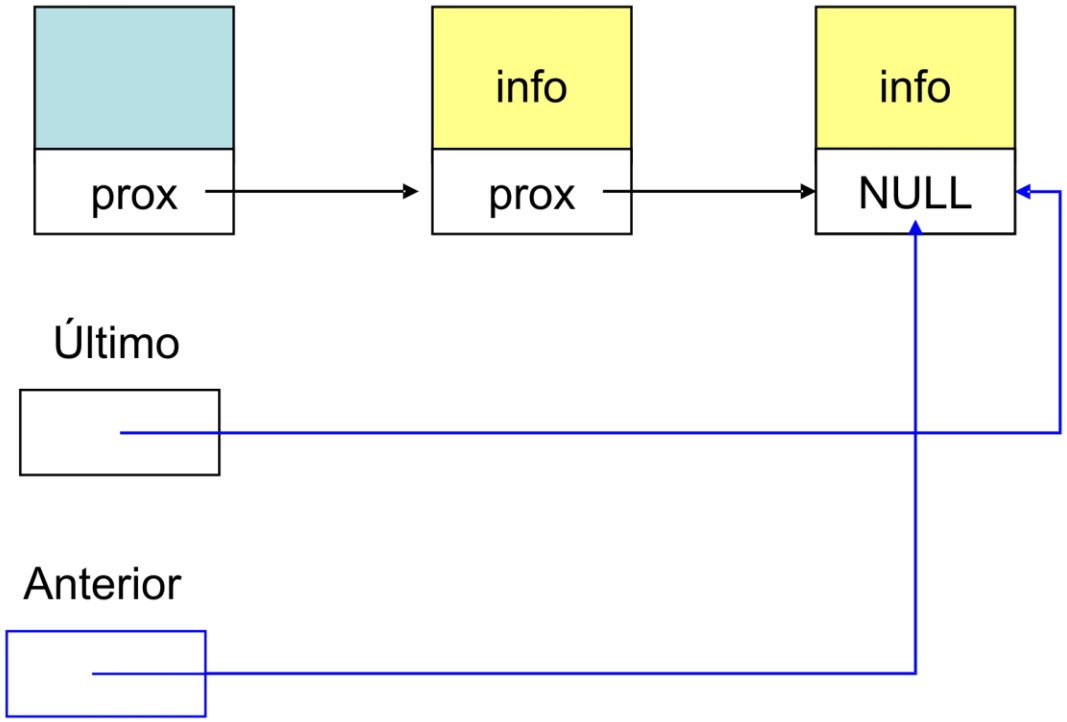
Lista encadeada com cabeça

- Remoção na última posição



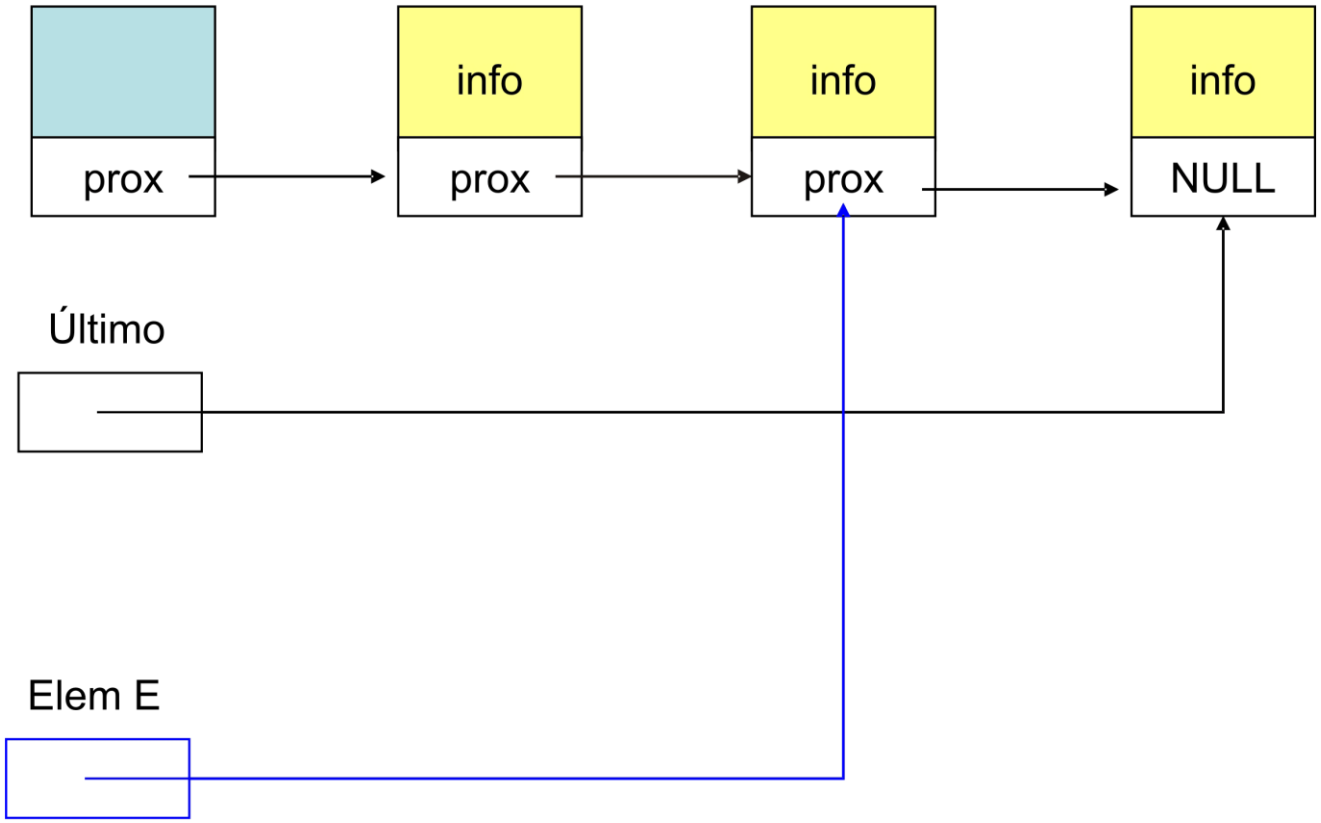
Lista encadeada com cabeça

- Remoção na última posição



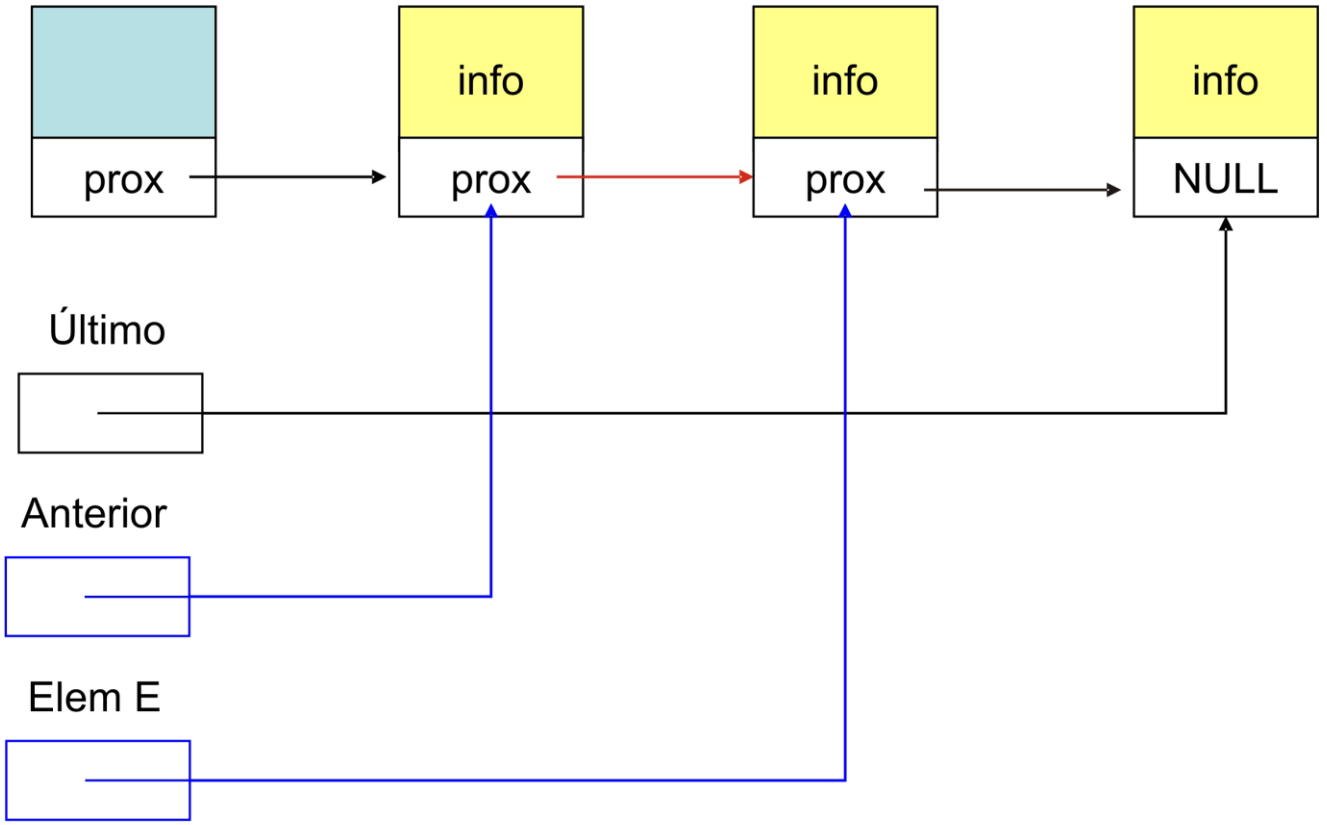
Lista encadeada com cabeça

- Remoção na posição de um elemento E



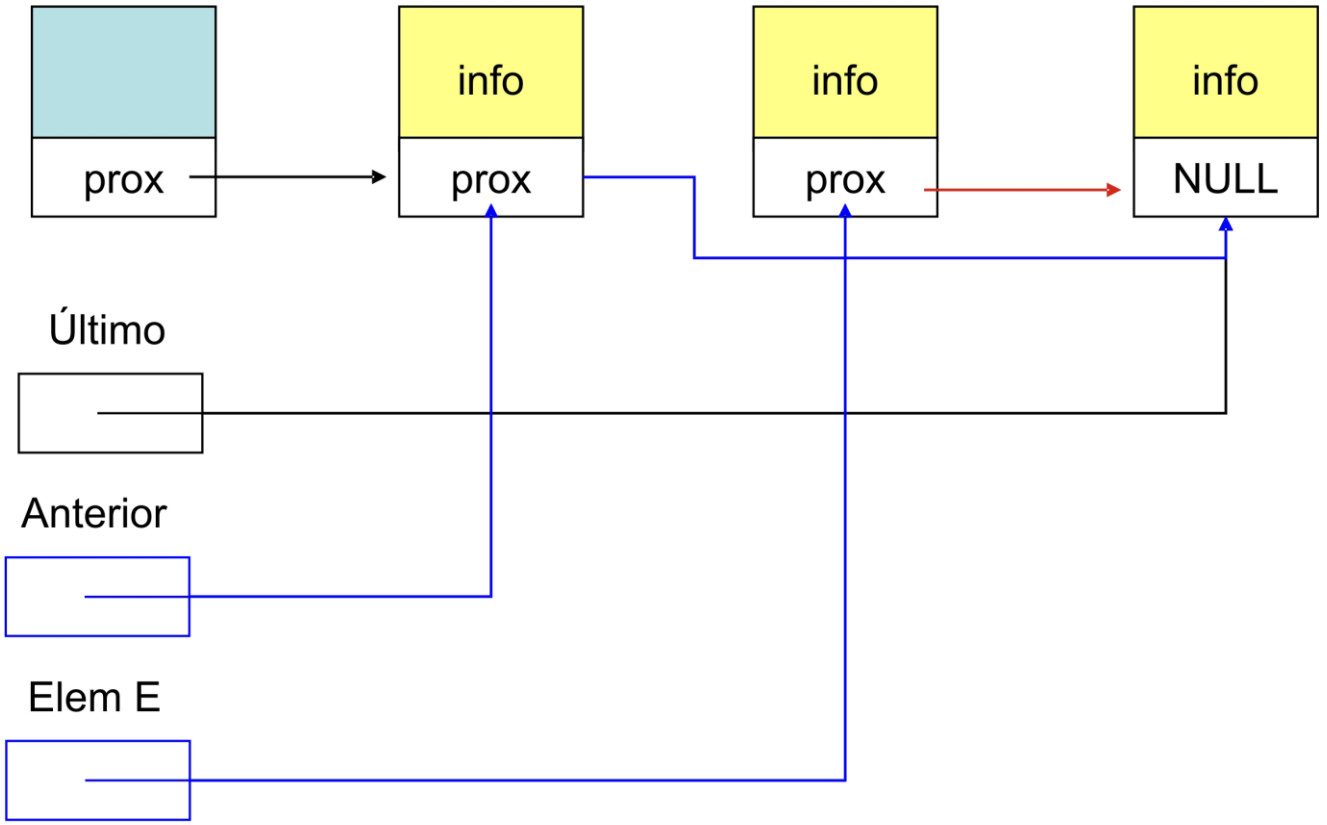
Lista encadeada com cabeça

- Remoção na posição de um elemento E



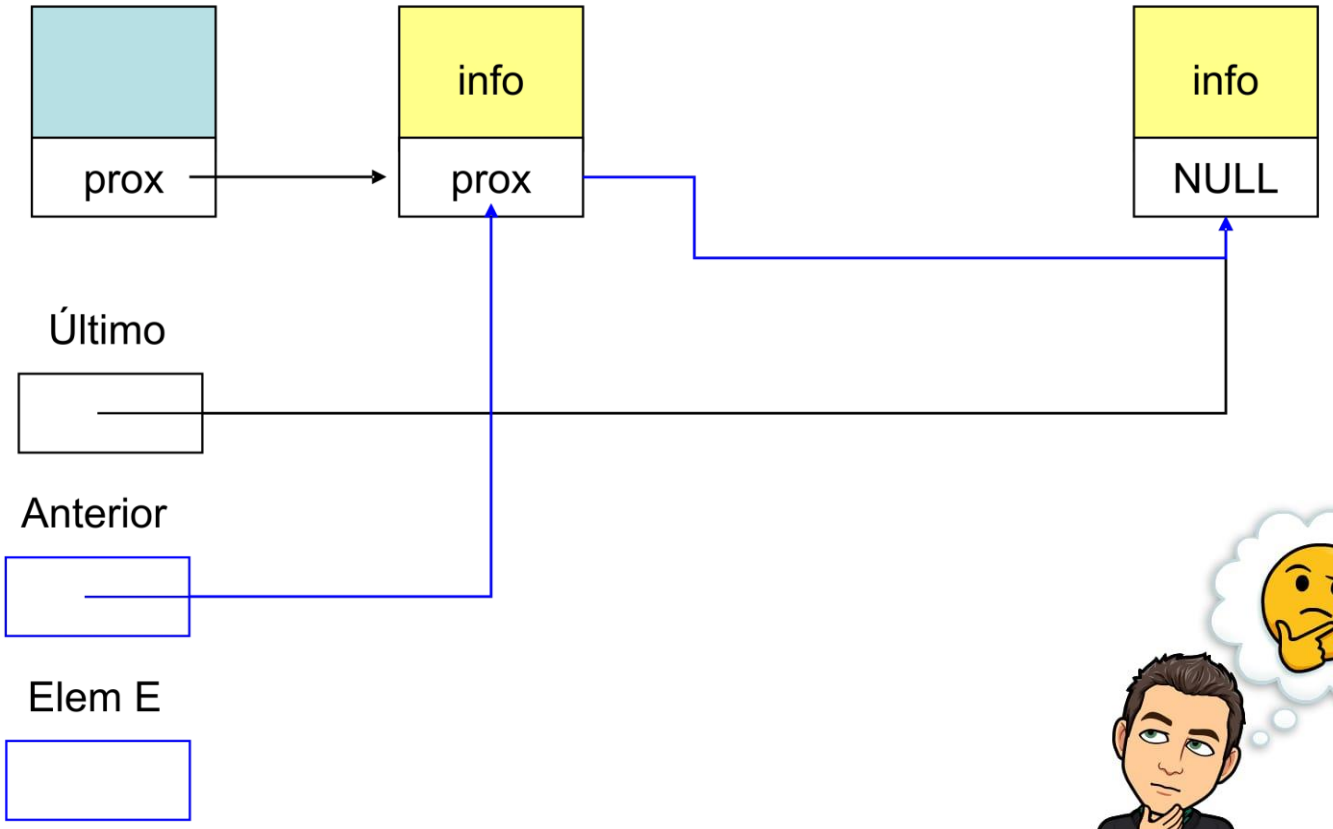
Lista encadeada com cabeça

- Remoção na posição de um elemento E



Lista encadeada com cabeça

- Remoção na posição de um elemento E



Implementação

Lista encadeada com cabeça

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int chave;
} TItem;

typedef struct celula {
    struct celula *pProx;
    TItem item;
} TCelula;

typedef struct {
    TCelula *pPrimeiro, *pUltimo;
} TLista;
```

Lista encadeada com cabeça

```
void iniciarLista (TLista *pLista);
int isVazia (TLista *pLista);
int inserir (TLista *pLista, TItem x);
int removerPrimeiro (TLista *pLista, TItem *pX);
void imprimir (TLista *pLista);
```

Lista encadeada com cabeça

```
void iniciarLista (TLista *pLista) {
    pLista->pPrimeiro = (TCelula *) malloc (sizeof (TCelula));
    pLista->pUltimo = pLista->pPrimeiro;
    pLista->pPrimeiro->pProx = NULL;
}

int isVazia (TLista *pLista) {
    return pLista->pPrimeiro == pLista->pUltimo;
}
```

Lista encadeada com cabeça

```
int inserir (TLista *pLista, TItem x) {
    pLista->pUltimo->pProx = (TCelula *) malloc (sizeof (TCelula));
    pLista->pUltimo = pLista->pUltimo->pProx;
    pLista->pUltimo->item = x;
    pLista->pUltimo->pProx = NULL;
    return 1;
}
```

Lista encadeada com cabeça

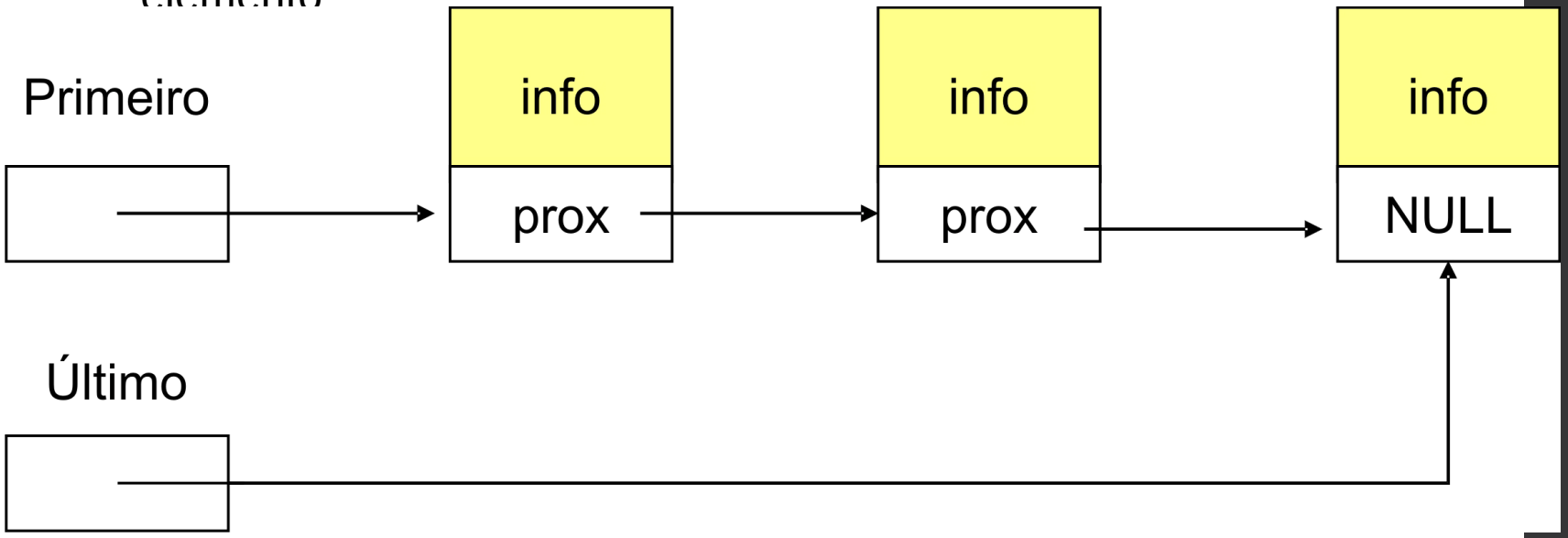
```
int removerPrimeiro (TLista *pLista, TItem *pX) {
    if (isVazia (pLista))
        return 0;
    TCelula *pAux;
    pAux = pLista->pPrimeiro->pProx;
    *pX = pAux->item;
    pLista->pPrimeiro->pProx = pAux->pProx;
    free (pAux);
    return 1;
}
```

Lista encadeada com cabeça

```
int main() {
    TLista lista;
    iniciarLista (&lista);

    printf("Vazia: %s\n", isVazia(&lista) == 1 ? "SIM":"NAO");

    TItem item1, item2;
    item1.chave = 10;
    inserir (&lista, item1);
    item2.chave = -5;
    inserir (&lista, item2);

    printf("Vazia: %s\n", isVazia(&lista) == 1 ? "SIM":"NAO");

    TItem itemRemovido;
    removerPrimeiro (&lista, &itemRemovido);
    printf("Item removido: %d\n", itemRemovido.chave);
}
```

Lista encadeada sem cabeça

- Semelhante a lista encadeada com cabeça
- Não possui uma célula representando a cabeça
- A cabeça é substituída por um apontador para o 1º elemento



Lista encadeada sem cabeça

- Estrutura será as mesmas da lista encadeada com cabeça.

```
void iniciarLista (TLista *pLista) {
    pLista->pPrimeiro = NULL;
    pLista->pUltimo = NULL;
}

int isVazia (TLista *pLista) {
    return pLista->pPrimeiro == NULL;
}
```

Lista encadeada sem cabeça

```
int inserir (TLista *pLista, TItem x) {
    TCelula *novo = (TCelula *) malloc (sizeof (TCelula));
    novo->item = x;
    novo->pProx = NULL;

    if (isVazia (pLista)) {
        pLista->pPrimeiro = novo;
    } else {
        pLista->pUltimo->pProx = novo;
    }
    pLista->pUltimo = novo;

    return 1;
}
```

Lista encadeada sem cabeça

```
int removerPrimeiro (TLista *pLista, TItem *pX) {
    if (isVazia (pLista))
        return 0;
    TCelula *pAux;
    pAux = pLista->pPrimeiro;
    *pX = pAux->item;
    pLista->pPrimeiro = pAux->pProx;
    free (pAux);
    return 1;
}
```

Exercícios

Exercícios

- Implemente a função de imprimir todos os elementos da lista
- Na aula foi comentando sobre 3 casos de inserção e remoção de elementos de uma lista encadeada, mas foi mostrado a implementação de apenas 1 tipo de cada (inserção no final e remoção no início)
  - Implemente pelo menos mais 1 tipo de inserção
  - Implemente pelo menos mais 1 tipo de remoção

Estrutura de Dados

Material elaborado por:
Thiago Meirelles Ventura

Baseado em:

- Ascencio, A. F. G; Araújo, G. S. Estruturas de Dados. Pearson, 2011.
- Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. Algoritmos: teoria e prática. Elsevier, 2002.
- Aulas do Prof. Reinaldo Silva Fortes (http://www.decom.ufop.br/reinaldo/)