

Dominando Estruturas de Dados 1

Listas Encadeadas Simples

Prof. dr. Samuel Martins (Samuka)
@xavecoding @hisamuka



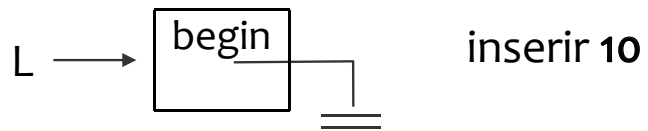
Lista Encadeada Simples

Diversos tipos de operações:

- **Inserção na cabeça (início) da lista;**
- Impressão dos Elementos da Lista
- Inserção na cauda (fim) da lista;
- Remover elementos da lista;
- Contar o número de elementos da Lista;
- Verificar se a lista está vazia e retornar verdadeiro/falso
- Retornar o primeiro elemento;
- Retornar o último elemento;
- Retornar um elemento na posição i

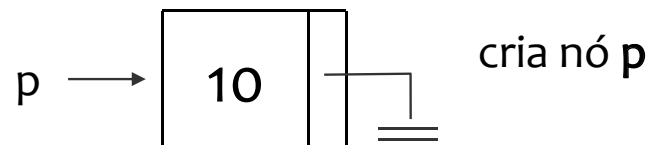
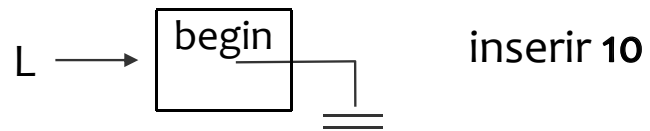
Inserção na cabeça (início) da lista

Caso 1: Lista está vazia



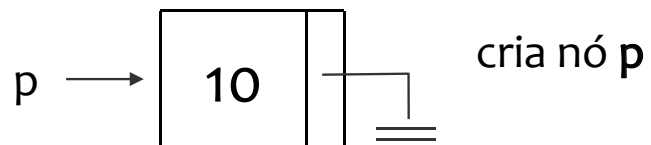
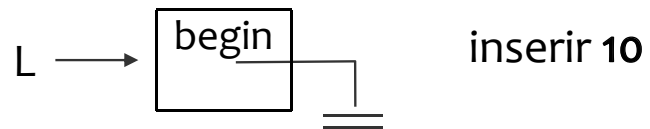
Inserção na cabeça (início) da lista

Caso 1: Lista está vazia

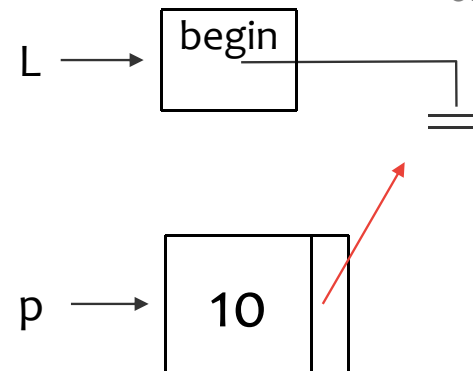


Inserção na cabeça (início) da lista

Caso 1: Lista está vazia

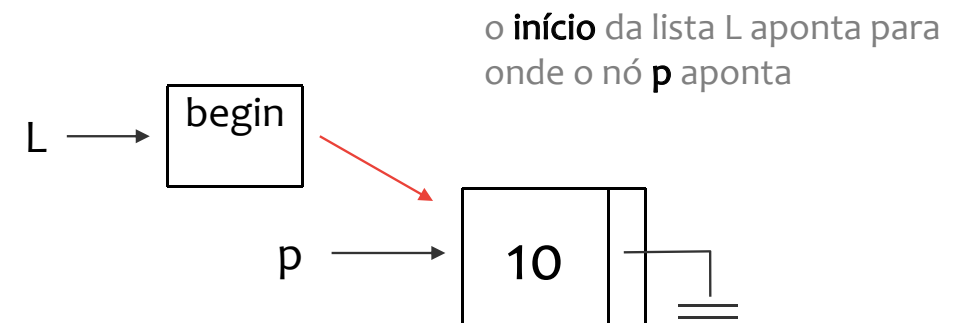
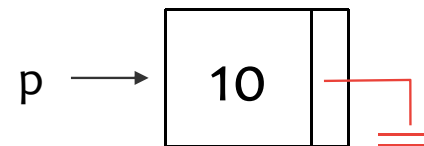
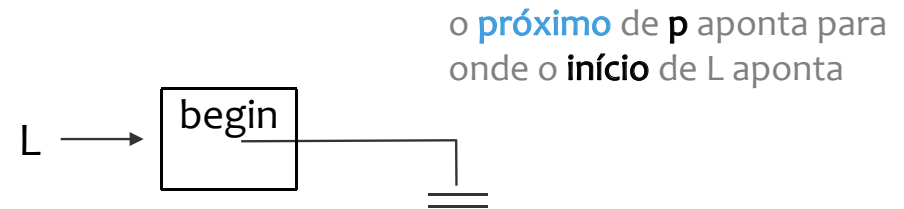
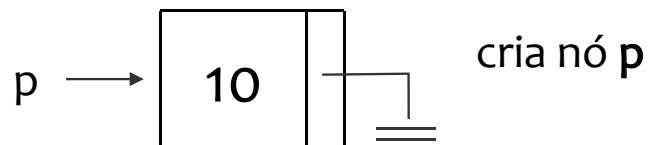
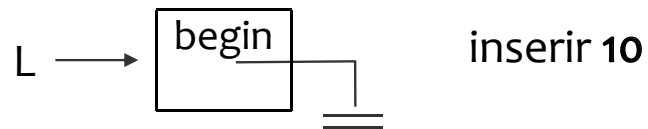


o próximo de p aponta para onde o início de L aponta



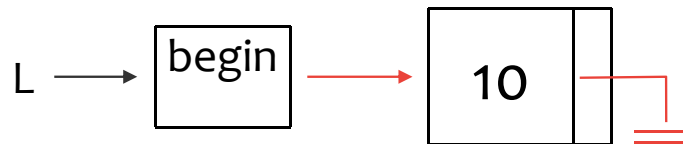
Inserção na cabeça (início) da lista

Caso 1: Lista está vazia



Inserção na cabeça (início) da lista

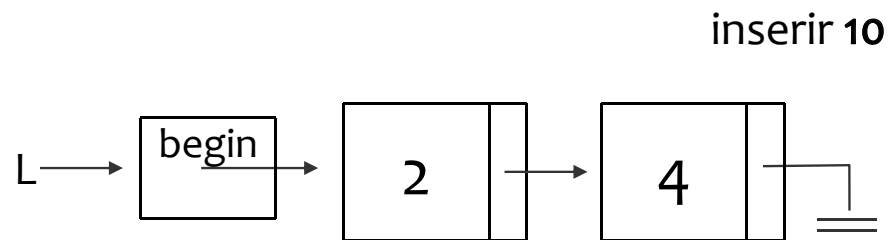
Caso 1: Lista está vazia



configuração final da lista
após a inserção

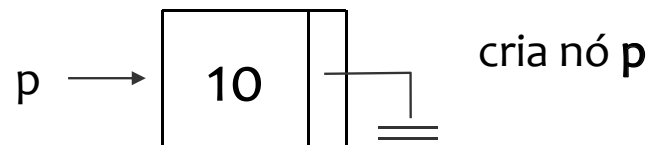
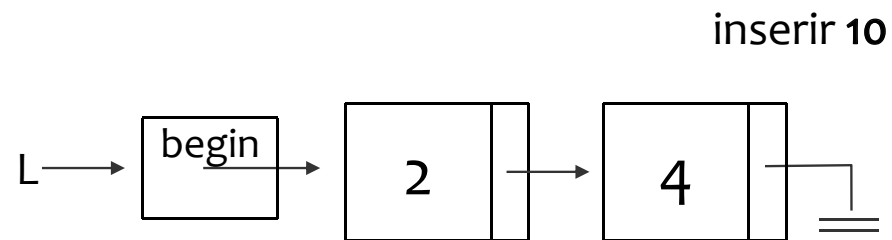
Inserção na cabeça (início) da lista

Caso 2: Lista possui elementos



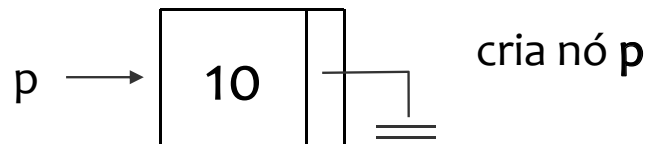
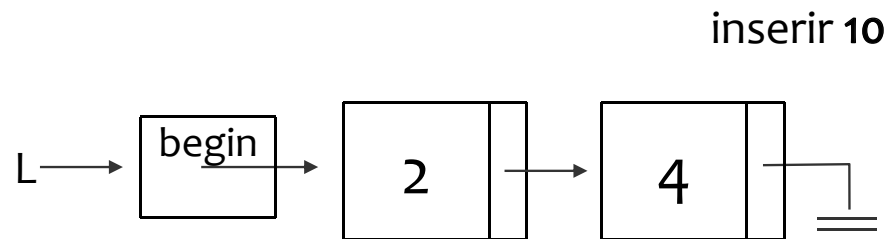
Inserção na cabeça (início) da lista

Caso 2: Lista possui elementos

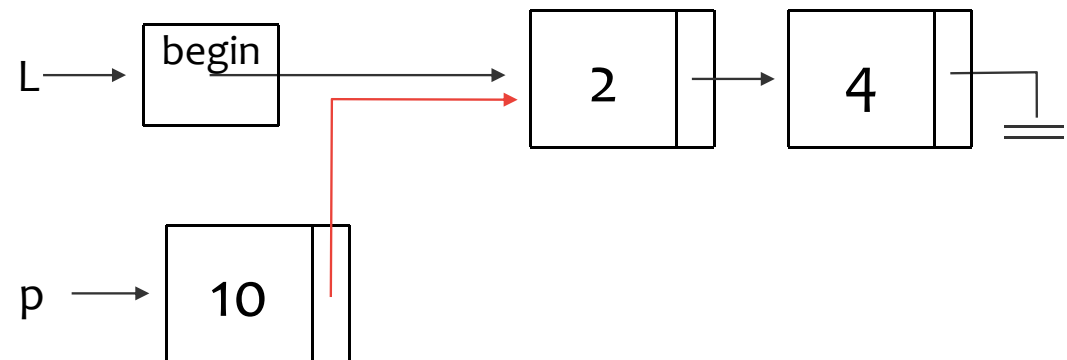


Inserção na cabeça (início) da lista

Caso 2: Lista possui elementos

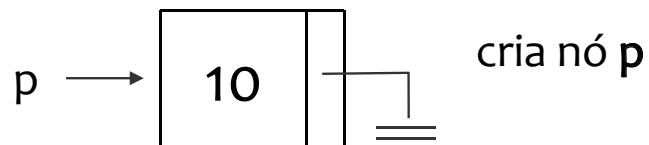
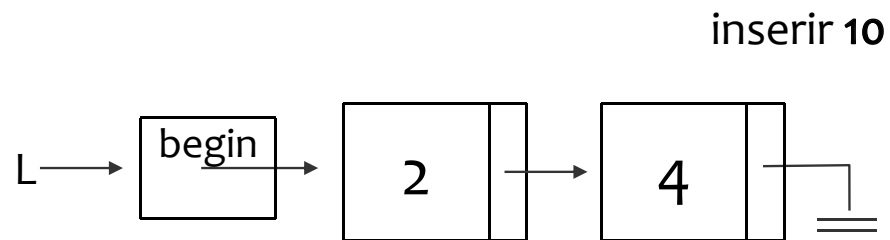


o próximo de p aponta para onde o início de L aponta

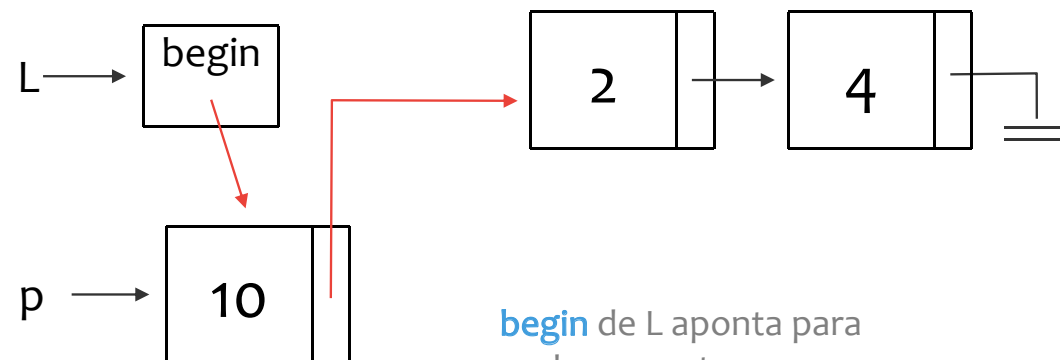
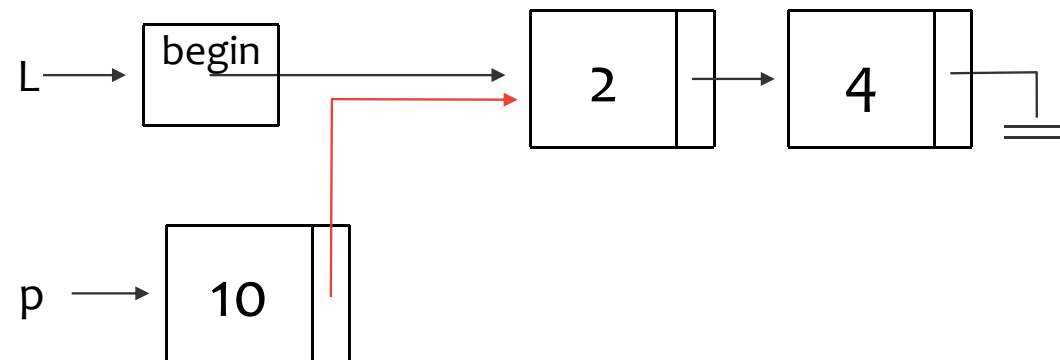


Inserção na cabeça (início) da lista

Caso 2: Lista possui elementos



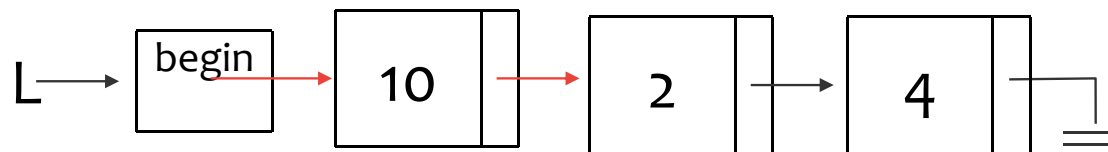
o próximo de p aponta para onde o início de L aponta



begin de L aponta para onde p aponta

Inserção na cabeça (início) da lista

Caso 2: Lista possui elementos



configuração final da
lista
após a inserção

Lista Encadeada Simples

Diversos tipos de operações:

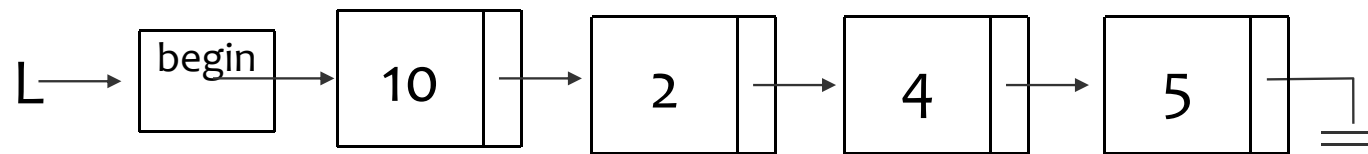
- **Inserção na cabeça (início) da lista;**
- Impressão dos Elementos da Lista
- Inserção na cauda (fim) da lista;
- Remover elementos da lista;
- Contar o número de elementos da Lista;
- Verificar se a lista está vazia e retornar verdadeiro/falso
- Retornar o primeiro elemento;
- Retornar o último elemento;
- Retornar um elemento na posição i

Lista Encadeada Simples

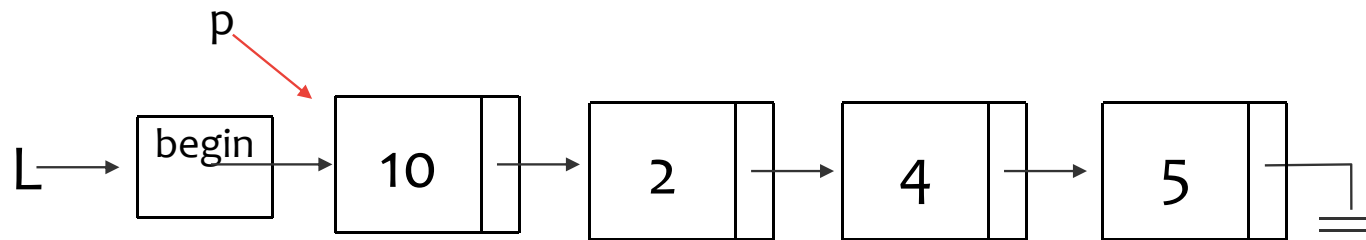
Diversos tipos de operações:

- Inserção na cabeça (início) da lista;
- **Impressão dos Elementos da Lista**
- Inserção na cauda (fim) da lista;
- Remover elementos da lista;
- Contar o número de elementos da Lista;
- Verificar se a lista está vazia e retornar verdadeiro/falso
- Retornar o primeiro elemento;
- Retornar o último elemento;
- Retornar um elemento na posição i

Impressão dos Elementos da Lista

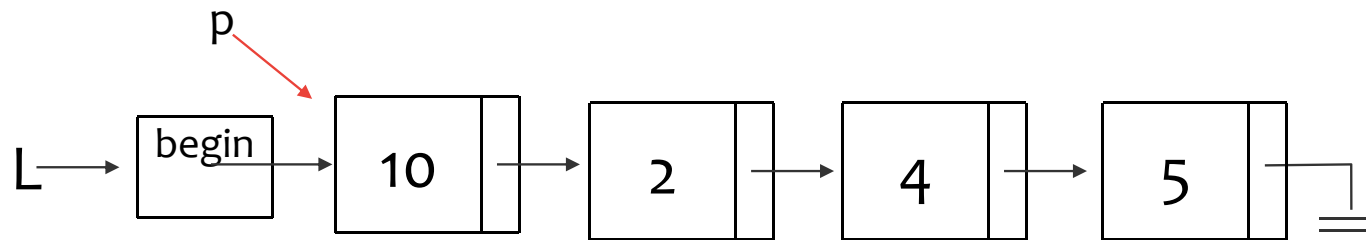


Impressão dos Elementos da Lista



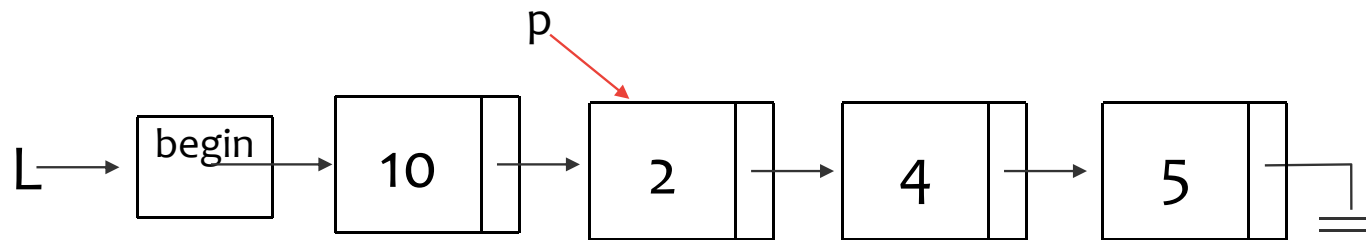
crie um ponteiro **p** e faça-o
apontar para o **início da lista**

Impressão dos Elementos da Lista



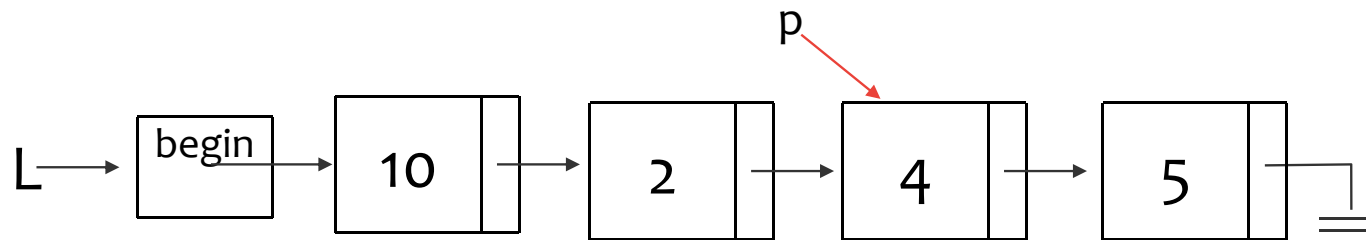
enquanto a lista **não chegou ao fim** (NULL):
imprima os dados apontado por **p** e
mova **p** para o **próximo elemento** da lista: **p = p->prox;**

Impressão dos Elementos da Lista



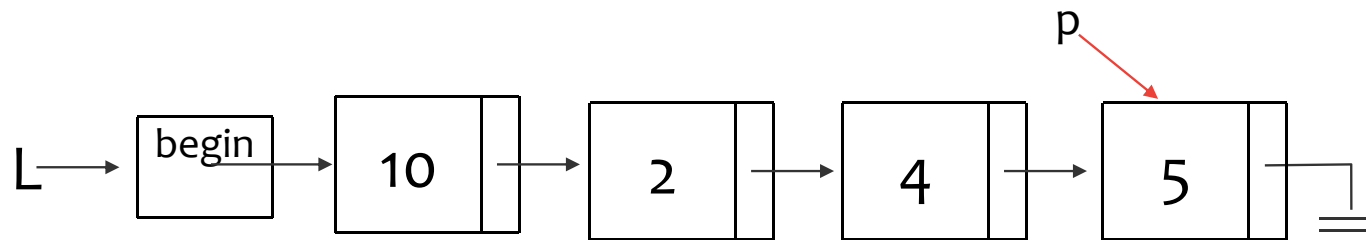
enquanto a lista **não chegou ao fim** (NULL):
imprima os dados apontado por **p** e
mova **p** para o **próximo elemento** da lista: **p = p->prox;**

Impressão dos Elementos da Lista



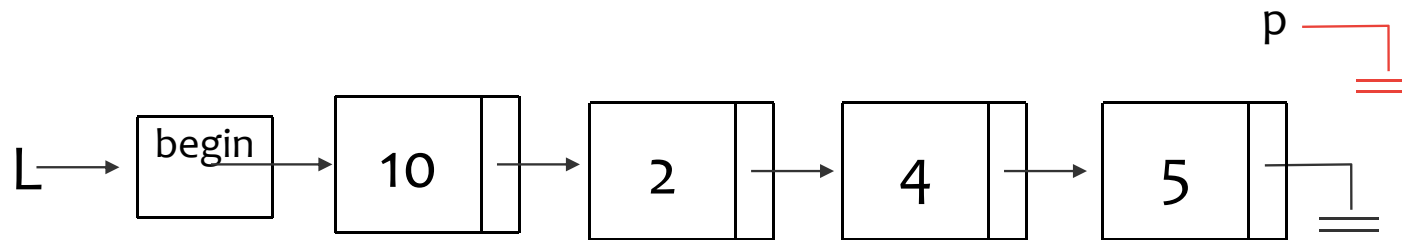
enquanto a lista **não chegou ao fim** (NULL):
 imprima os dados apontado por **p** e
 mova **p** para o **próximo elemento** da lista: **p = p->prox;**

Impressão dos Elementos da Lista



enquanto a lista **não chegou ao fim** (NULL):
imprima os dados apontado por **p** e
mova **p** para o **próximo elemento** da lista: **p = p->prox;**

Impressão dos Elementos da Lista



enquanto a lista **não chegou ao fim** (NULL):
imprima os dados apontado por **p** e
mova **p** para o **próximo elemento** da lista: **p = p->prox;**

Lista Encadeada Simples

Diversos tipos de operações:

- Inserção na cabeça (início) da lista;
- **Impressão dos Elementos da Lista**
- Inserção na cauda (fim) da lista;
- Remover elementos da lista;
- Contar o número de elementos da Lista;
- Verificar se a lista está vazia e retornar verdadeiro/falso
- Retornar o primeiro elemento;
- Retornar o último elemento;
- Retornar um elemento na posição i

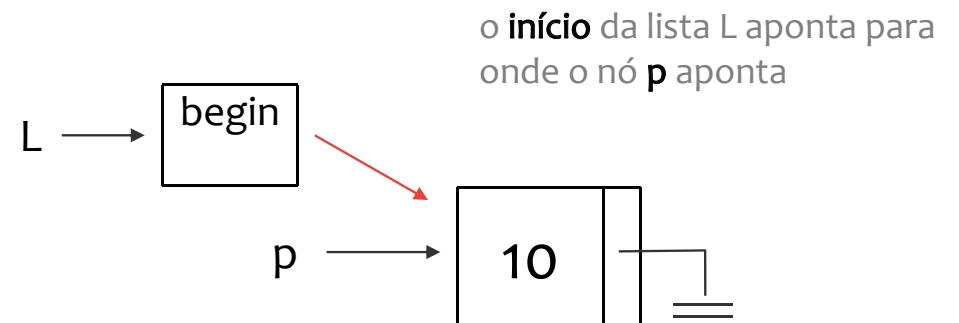
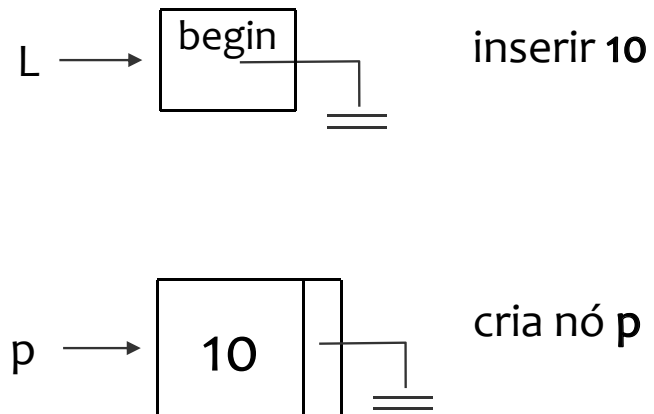
Lista Encadeada Simples

Diversos tipos de operações:

- Inserção na cabeça (início) da lista;
- Impressão dos Elementos da Lista
- **Inserção na cauda (fim) da lista;**
- Remover elementos da lista;
- Contar o número de elementos da Lista;
- Verificar se a lista está vazia e retornar verdadeiro/falso
- Retornar o primeiro elemento;
- Retornar o último elemento;
- Retornar um elemento na posição i

Inserção na cauda (fim) da lista

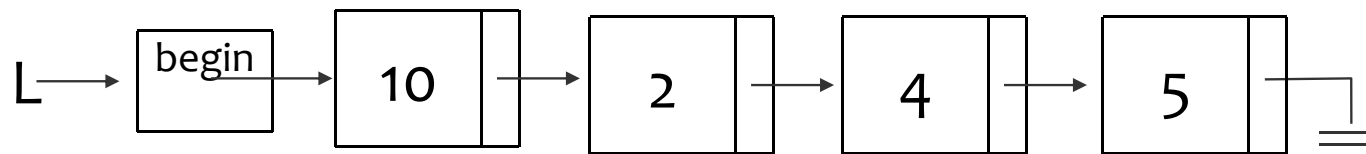
Caso 1: Lista está vazia



Inserção na cauda (fim) da lista

Caso 2: Lista possui elementos

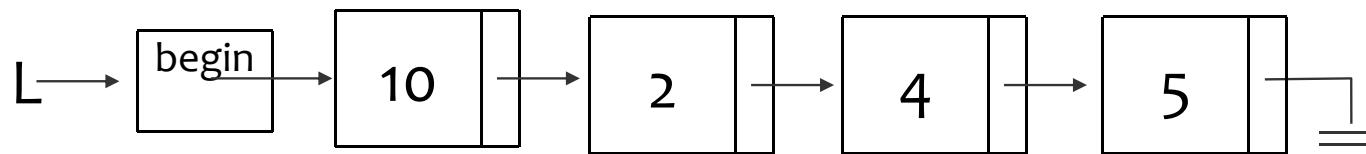
inserir 7



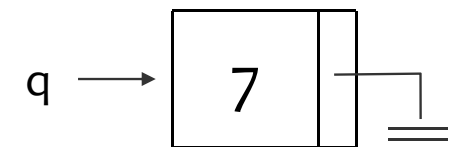
Inserção na cauda (fim) da lista

Caso 2: Lista possui elementos

inserir 7



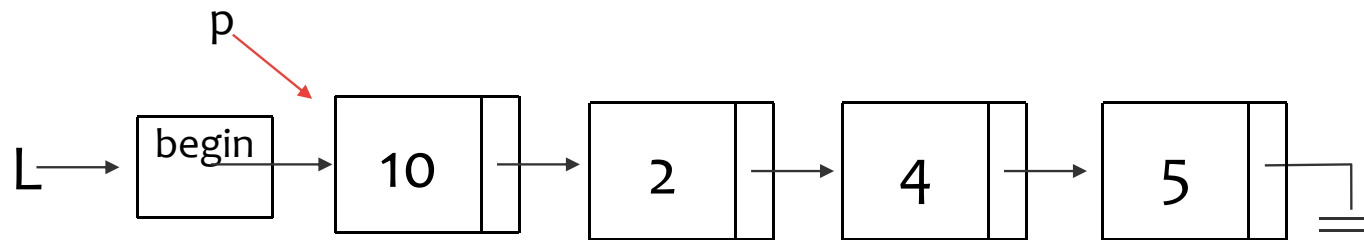
cria nó q



Inserção na cauda (fim) da lista

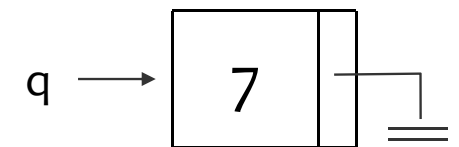
Caso 2: Lista possui elementos

inserir 7



O ponteiro **p** aponta, inicialmente, para o **nó inicial da lista**. Queremos posicionar **p** no **último** nós para então inserir o novo depois ($p \rightarrow \text{next} = q$)

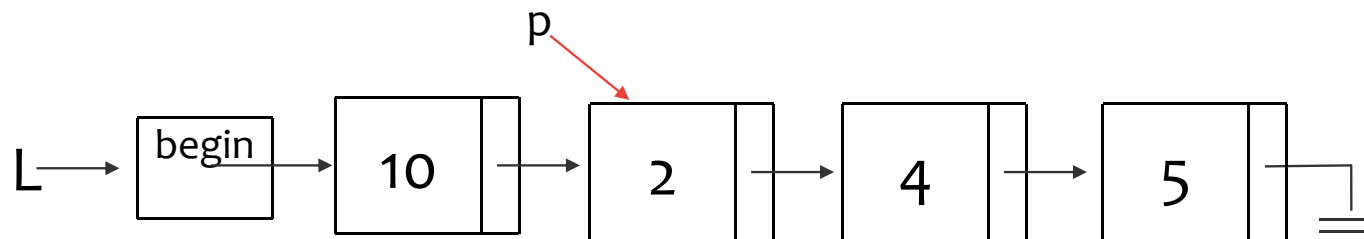
cria nó **q**



Inserção na cauda (fim) da lista

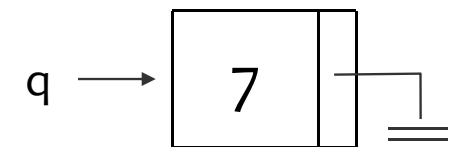
Caso 2: Lista possui elementos

inserir 7



O ponteiro **p** aponta, inicialmente, para o **nó inicial da lista**. Queremos posicionar **p** no **último** nós para então inserir o novo depois ($p \rightarrow \text{next} = q$)

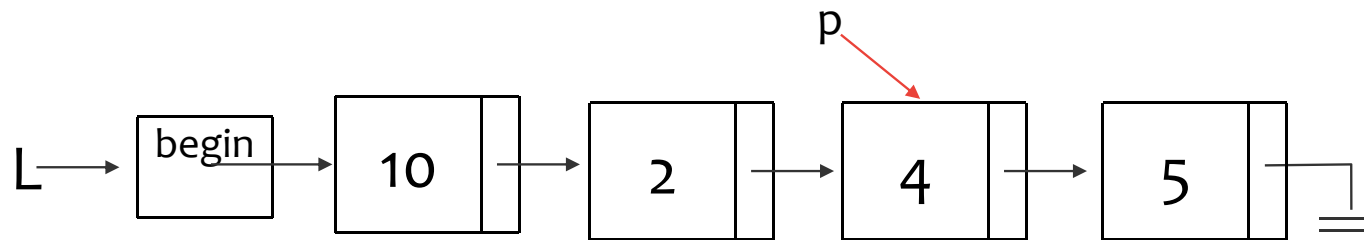
cria nó **q**



Inserção na cauda (fim) da lista

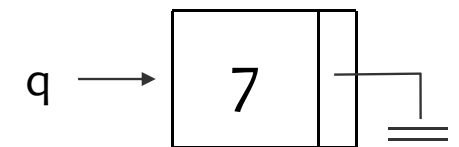
Caso 2: Lista possui elementos

inserir 7



O ponteiro **p** aponta, inicialmente, para o **nó inicial da lista**. Queremos posicionar **p** no **último** nós para então inserir o novo depois ($p \rightarrow \text{next} = q$)

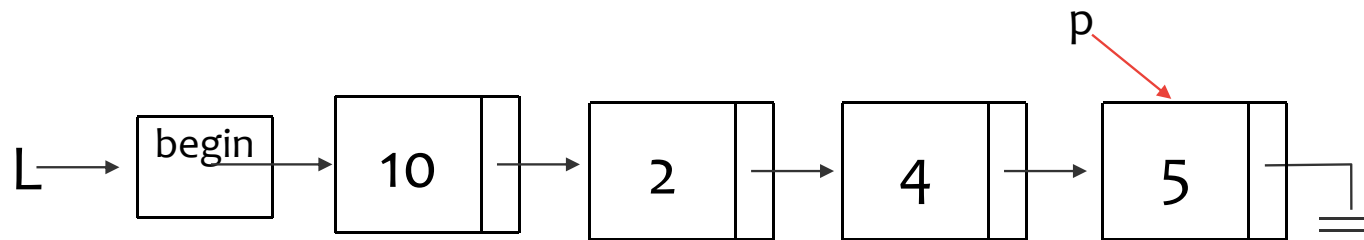
cria nó **q**



Inserção na cauda (fim) da lista

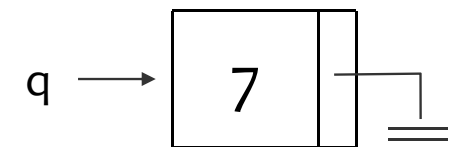
Caso 2: Lista possui elementos

inserir 7



O ponteiro **p** aponta, inicialmente, para o **nó inicial da lista**. Queremos posicionar **p** no **último** nós para então inserir o novo depois ($p \rightarrow \text{next} = q$)

cria nó **q**

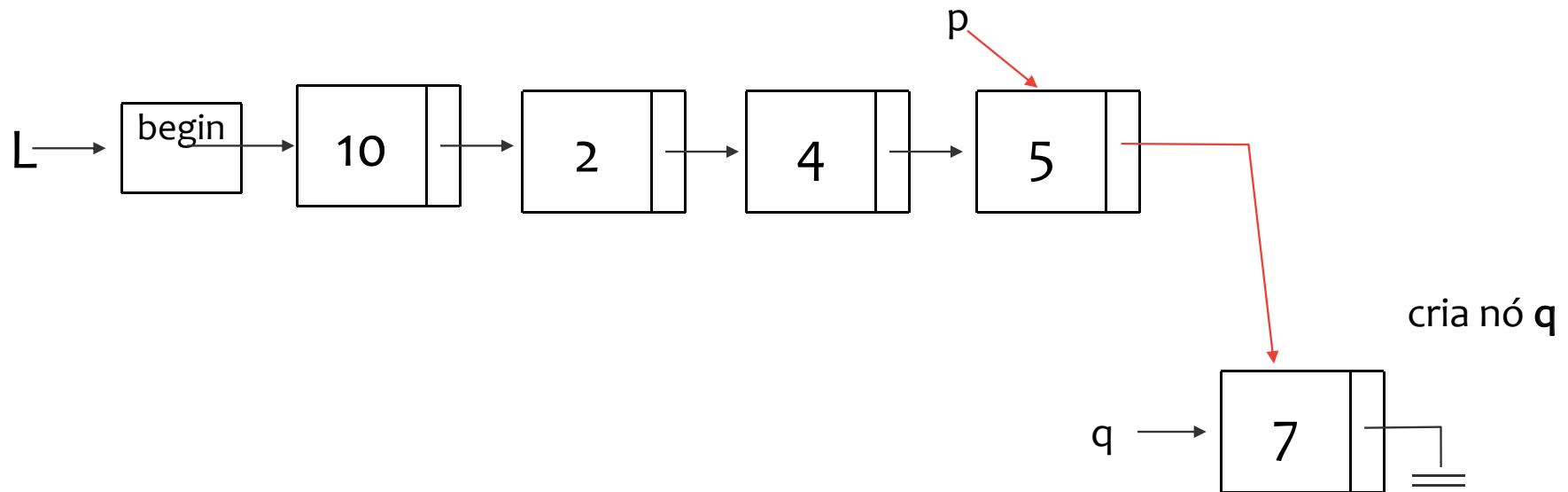


Inserção na cauda (fim) da lista

Caso 2: Lista possui elementos

inserir 7

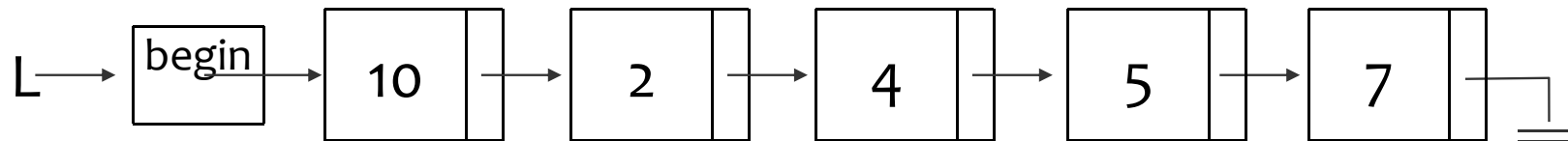
o próximo nó de **p** é **NULL**,
chegamos no nó final



o **próximo** de **p** aponta para **q**

Inserção na cauda (fim) da lista

Caso 2: Lista possui elementos



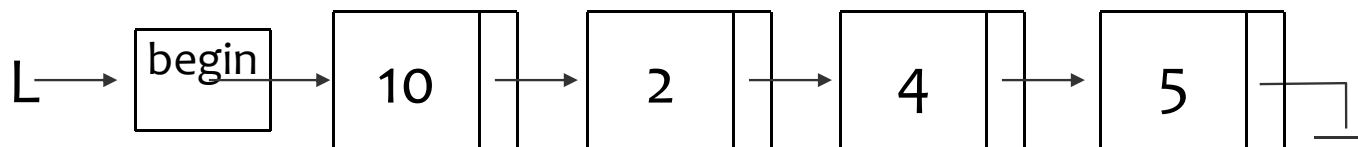
configuração final da
lista
após a inserção

Inserção na cauda (fim) da lista

Caso 2: Lista possui elementos

Problemas?

- Precisamos percorrer todos os nós da lista até chegar no último nó —> **ineficiente**



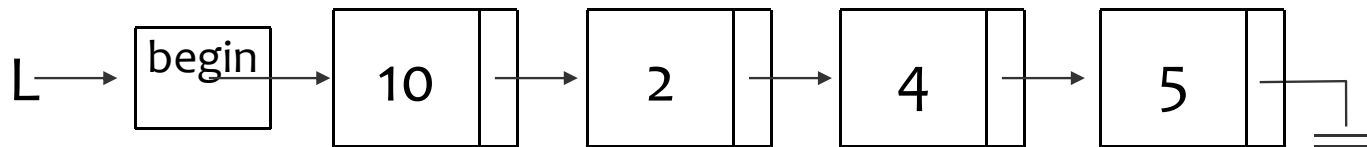
Inserção na cauda (fim) da lista

Caso 2: Lista possui elementos

Problemas?

- Precisamos percorrer todos os nós da lista até chegar no último nó —> **ineficiente**

Solução?



Inserção na cauda (fim) da lista

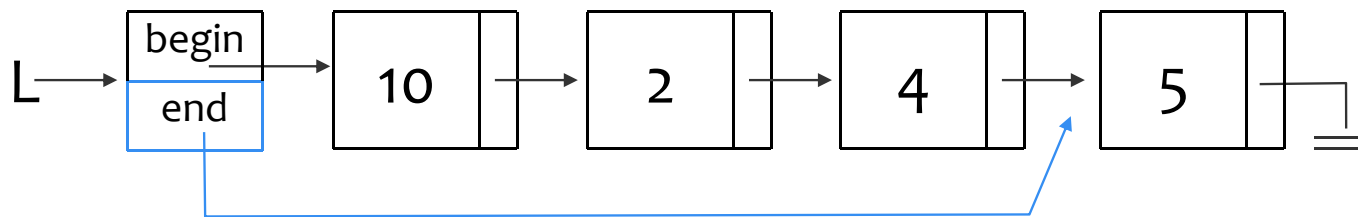
Caso 2: Lista possui elementos

Problemas?

- Precisamos percorrer todos os nós da lista até chegar no último nó —> **ineficiente**

Solução?

- Adicionamos, na struct de Lista, um ponteiro **end** para o **nó final da lista**;



Inserção na cauda (fim) da lista

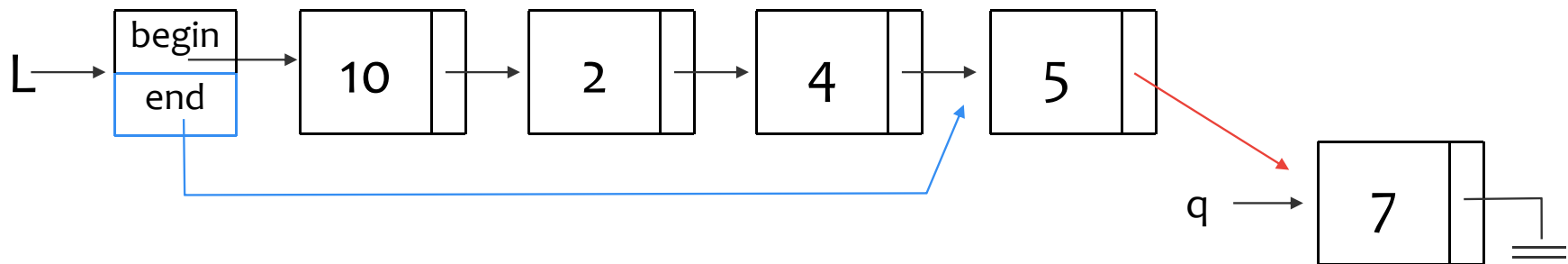
Caso 2: Lista possui elementos

Problemas?

- Precisamos percorrer todos os nós da lista até chegar no último nó —> **ineficiente**

Solução?

- Adicionamos, na struct de Lista, um ponteiro **end** para o **nó final da lista**;
- Assim, conseguimos adicionar o novo nó diretamente no final da lista



Inserção na cauda (fim) da lista

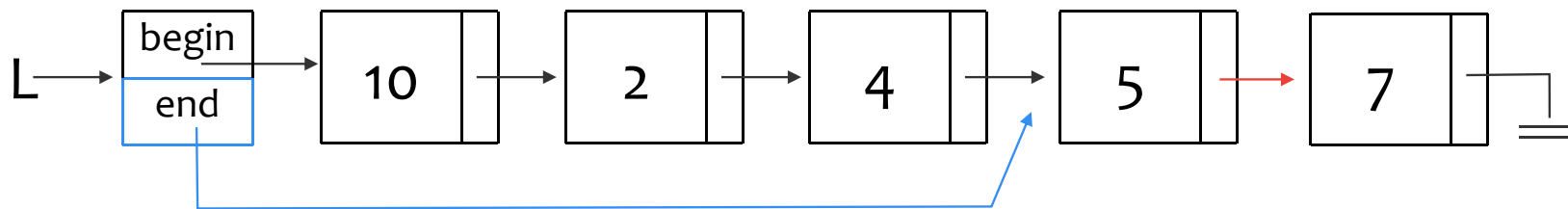
Caso 2: Lista possui elementos

Problemas?

- Precisamos percorrer todos os nós da lista até chegar no último nó —> **ineficiente**

Solução?

- Adicionamos, na struct de Lista, um ponteiro **end** para o **nó final da lista**;
- Assim, conseguimos adicionar o novo nó diretamente no final da lista



Inserção na cauda (fim) da lista

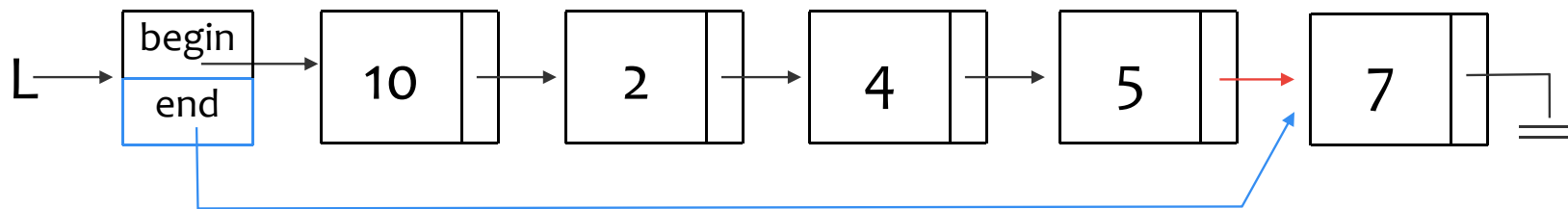
Caso 2: Lista possui elementos

Problemas?

- Precisamos percorrer todos os nós da lista até chegar no último nó —> **ineficiente**

Solução?

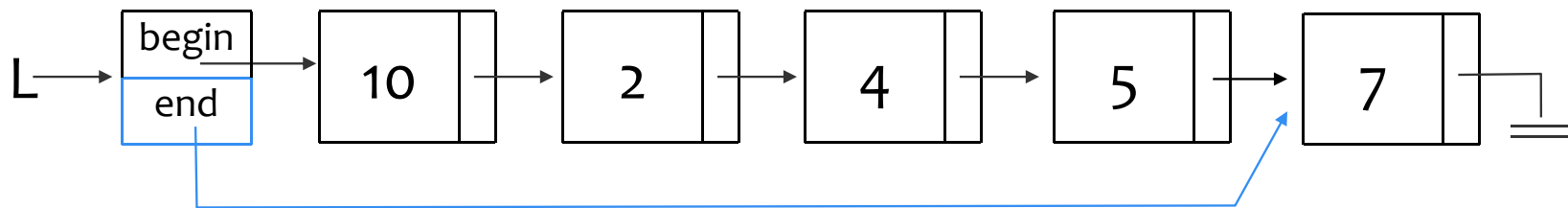
- Adicionamos, na struct de Lista, um ponteiro **end** para o **nó final da lista**;
- Assim, conseguimos adicionar o novo nó diretamente no final da lista
- Teremos que adaptar o **caso 1** da função de **inserção no início** da lista



Inserção na cauda (fim) da lista

```
typedef struct _node {  
    int val;  
    struct _node *next;  
} Node;
```

```
typedef struct _linked_list {  
    Node *begin;  
    Node *end;  
} LinkedList;
```



Lista Encadeada Simples

Diversos tipos de operações:

- Inserção na cabeça (início) da lista;
- Impressão dos Elementos da Lista
- **Inserção na cauda (fim) da lista;**
- Remover elementos da lista;
- Contar o número de elementos da Lista;
- Verificar se a lista está vazia e retornar verdadeiro/falso
- Retornar o primeiro elemento;
- Retornar o último elemento;
- Retornar um elemento na posição i

Lista Encadeada Simples

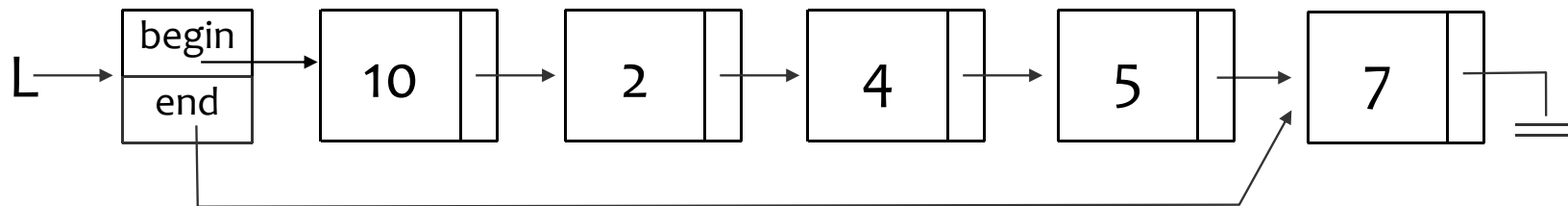
Diversos tipos de operações:

- Inserção na cabeça (início) da lista;
- Impressão dos Elementos da Lista
- Inserção na cauda (fim) da lista;
- **Remover elementos da lista;**
- Contar o número de elementos da Lista;
- Verificar se a lista está vazia e retornar verdadeiro/falso
- Retornar o primeiro elemento;
- Retornar o último elemento;
- Retornar um elemento na posição i

Remoção de Elementos da Lista

Caso 1: Elemento está na **cabeça (início)** da lista

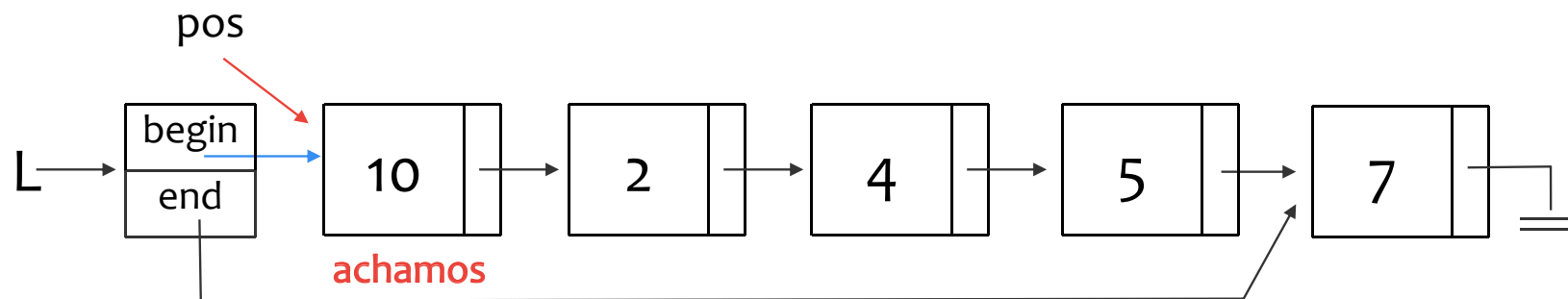
remover 10



Remoção de Elementos da Lista

Caso 1: Elemento está na **cabeça (início)** da lista

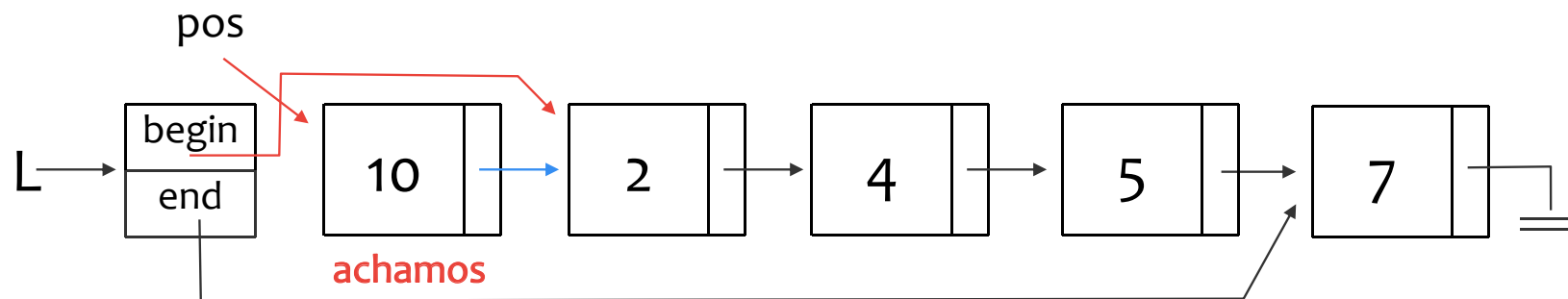
remover 10



Remoção de Elementos da Lista

Caso 1: Elemento está na **cabeça (início)** da lista

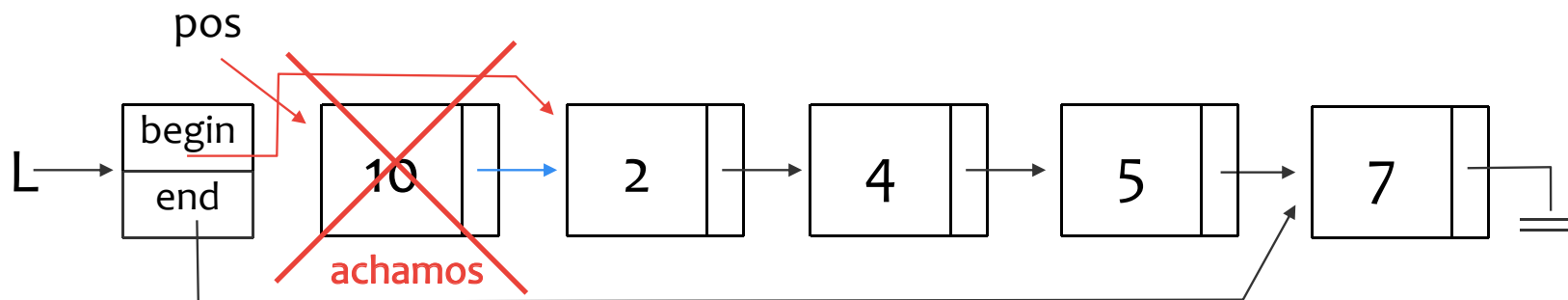
remover 10



Remoção de Elementos da Lista

Caso 1: Elemento está na **cabeça (início)** da lista

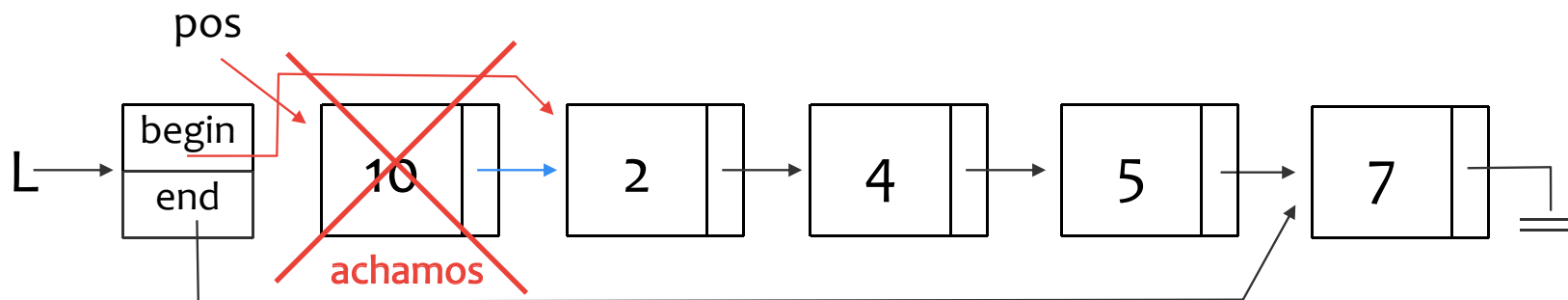
remover 10



Remoção de Elementos da Lista

Caso 1: Elemento está na **cabeça (início)** da lista

remover 10

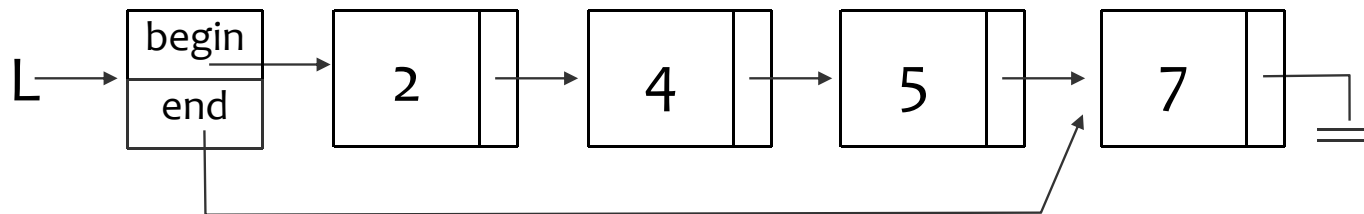


poderíamos continuar percorrendo
a lista para remover outros
possíveis nós 10

Remoção de Elementos da Lista

Caso 1: Elemento está na **cabeça (início)** da lista

remover 10

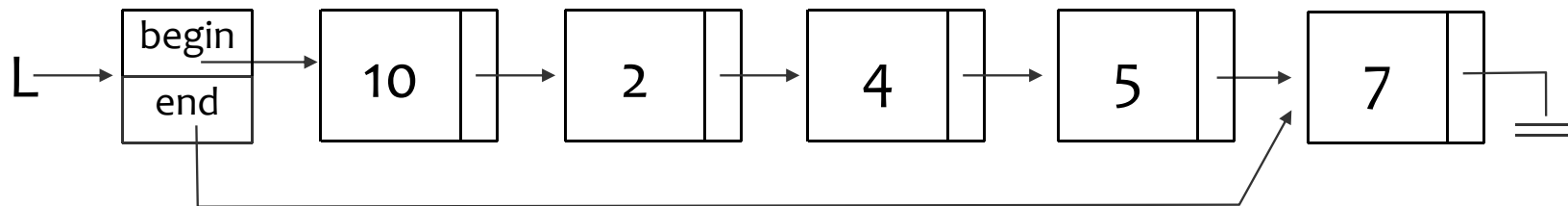


configuração final da lista
após a inserção

Remoção de Elementos da Lista

Caso 2: Elemento está no **meio** da lista

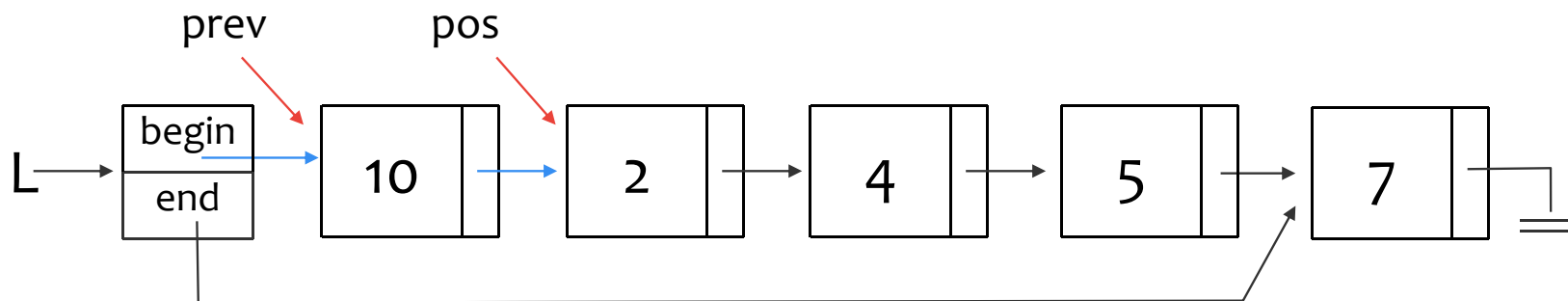
remover 4



Remoção de Elementos da Lista

Caso 2: Elemento está no **meio** da lista

remover 4



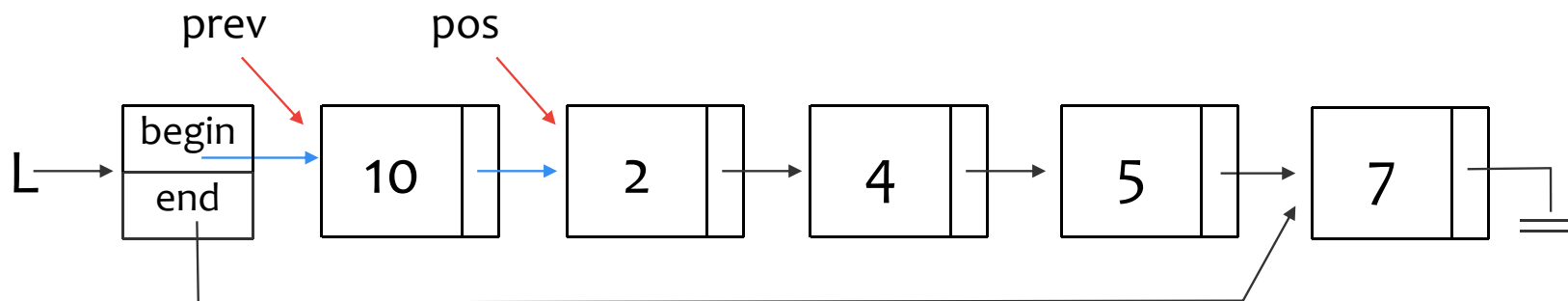
```
prev = L->inicio;  
pos = L->inicio->prox;
```

além do ponteiro **pos** que checa o **elemento atual**, precisaremos de um outro ponteiro **prev** apontando para o **nó anterior**

Remoção de Elementos da Lista

Caso 2: Elemento está no **meio** da lista

remover 4



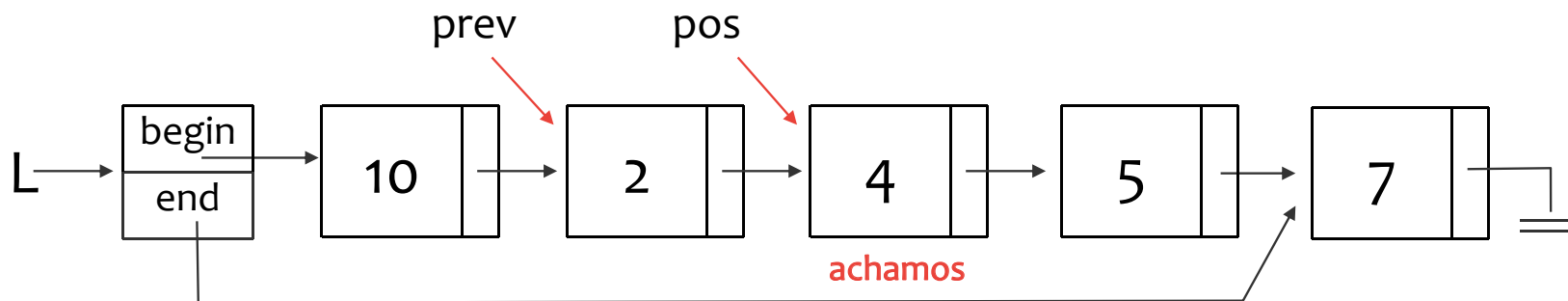
```
prev = L->inicio;  
pos = L->inicio->prox;
```

enquanto **não achou** o elemento e a lista **não chegou ao fim** ($pos == \text{NULL}$),
mova os dois ponteiros para os próximos nós

Remoção de Elementos da Lista

Caso 2: Elemento está no **meio** da lista

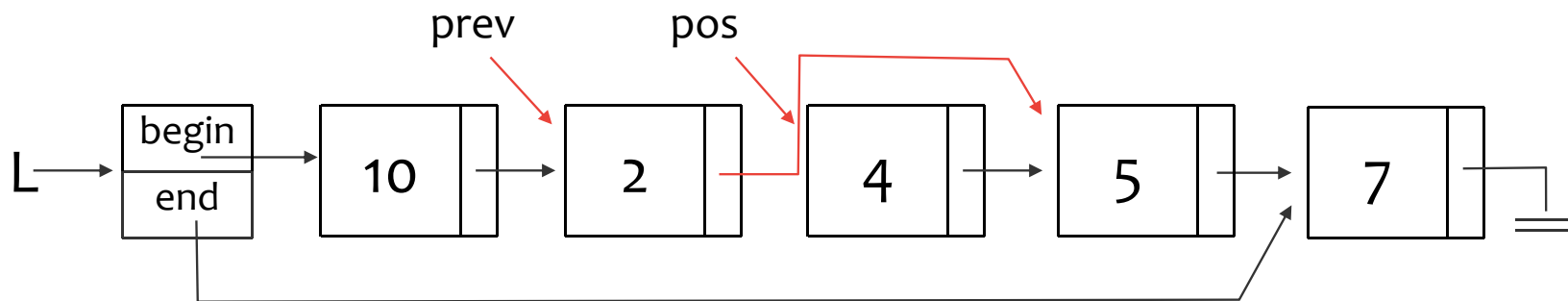
remover 4



Remoção de Elementos da Lista

Caso 2: Elemento está no **meio** da lista

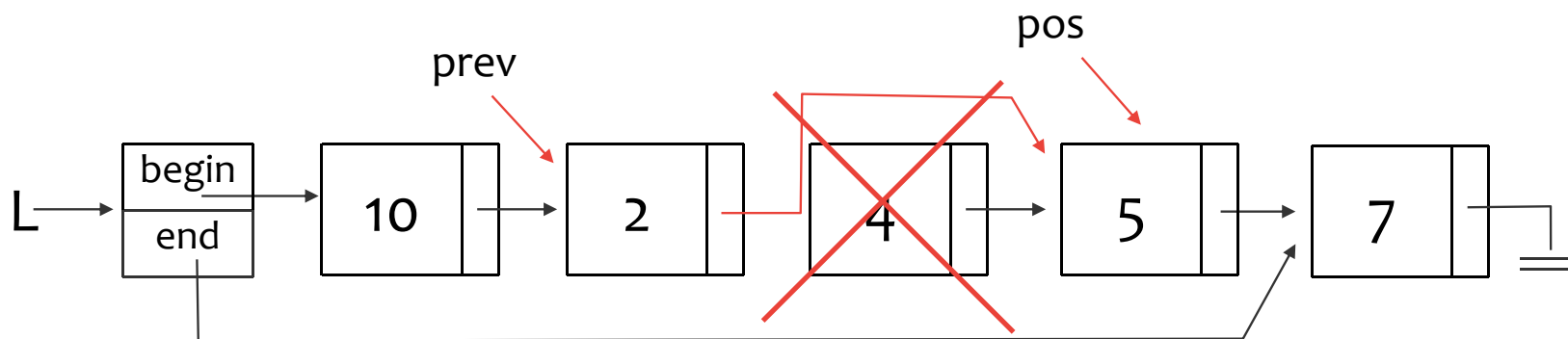
remover 4



Remoção de Elementos da Lista

Caso 2: Elemento está no **meio** da lista

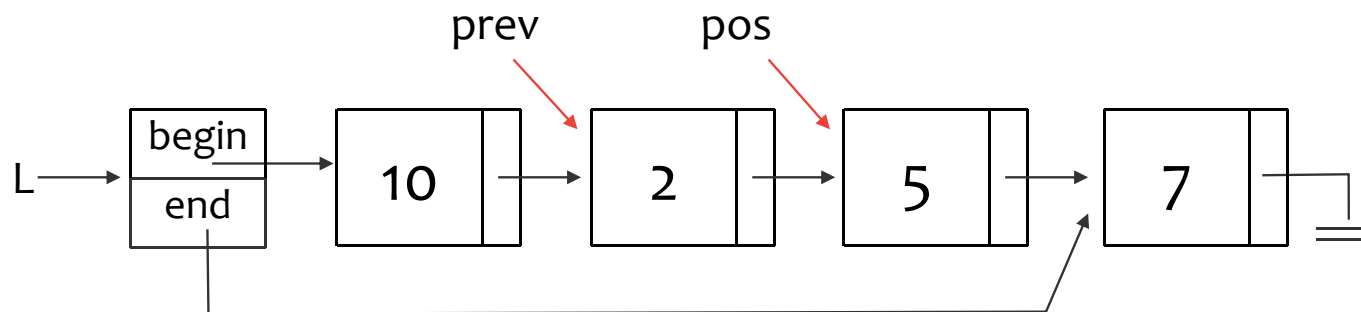
remover 4



Remoção de Elementos da Lista

Caso 2: Elemento está no **meio** da lista

remover 4

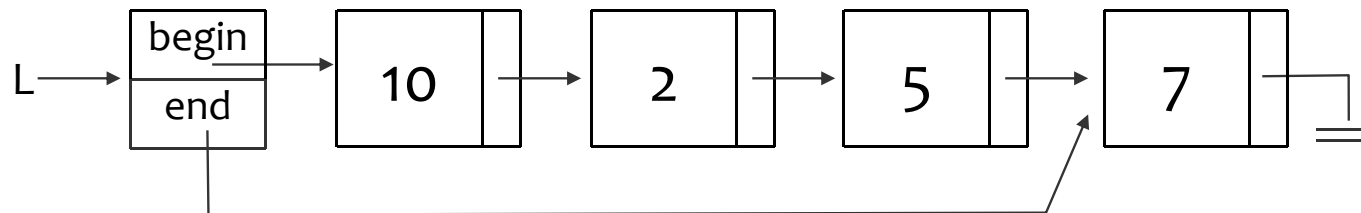


poderíamos continuar percorrendo
a lista para remover outros
possíveis nós 4

Remoção de Elementos da Lista

Caso 2: Elemento está no **meio** da lista

remover 4

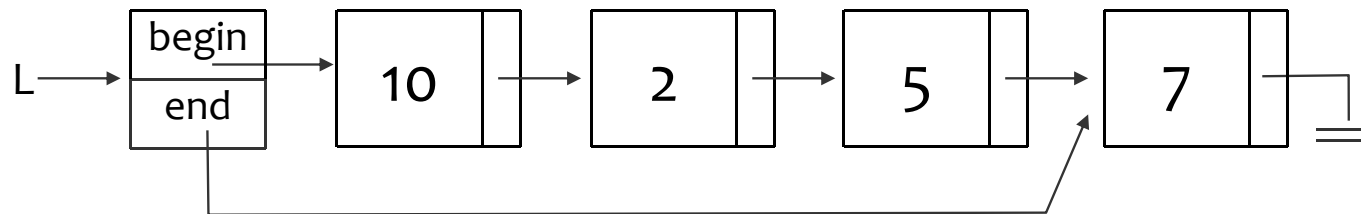


configuração final da lista
após a inserção

Remoção de Elementos da Lista

Caso 3: Elemento está na **cauda (final)** da lista

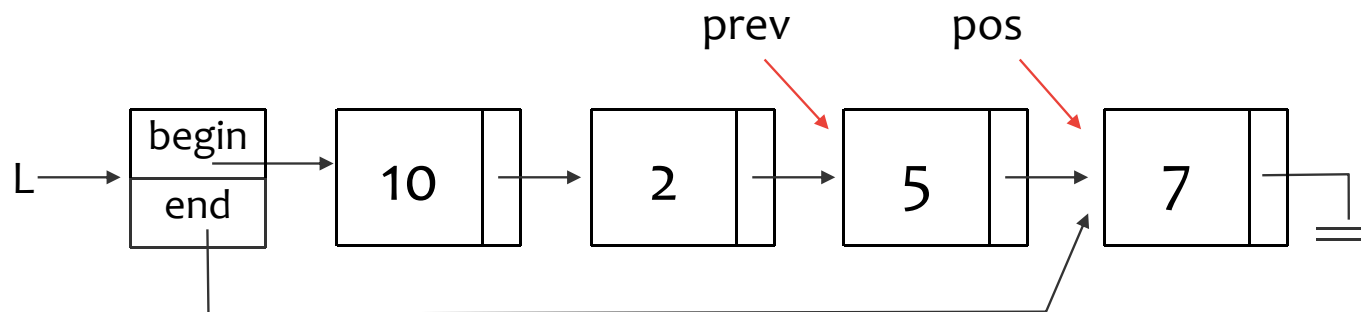
remover 7



Remoção de Elementos da Lista

Caso 3: Elemento está na **cauda (final)** da lista

remover 7

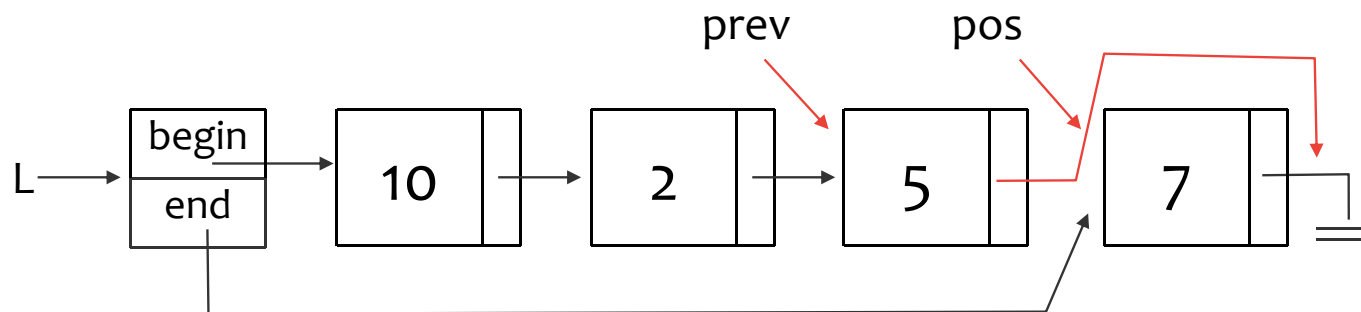


Suponha que já percorremos a lista inteira e chegamos no nó final

Remoção de Elementos da Lista

Caso 3: Elemento está na **cauda (final)** da lista

remover 7

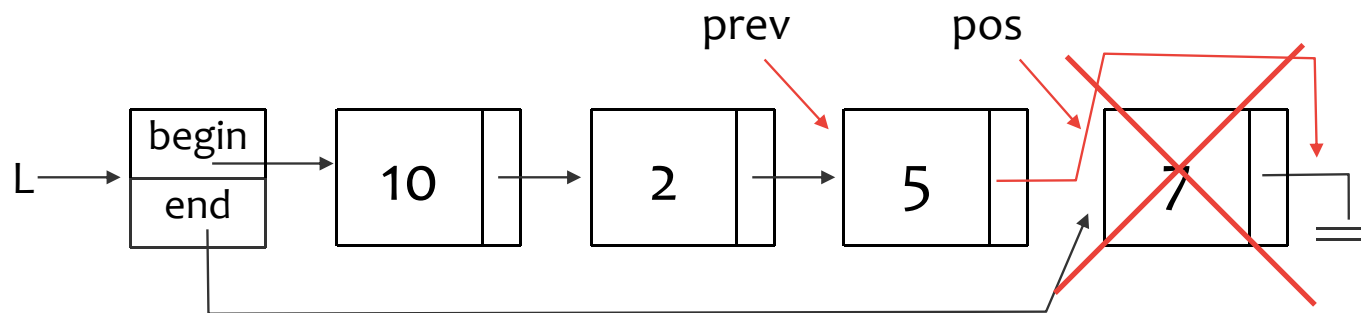


Suponha que já percorremos a lista inteira e chegamos no nó final

Remoção de Elementos da Lista

Caso 3: Elemento está na **cauda (final)** da lista

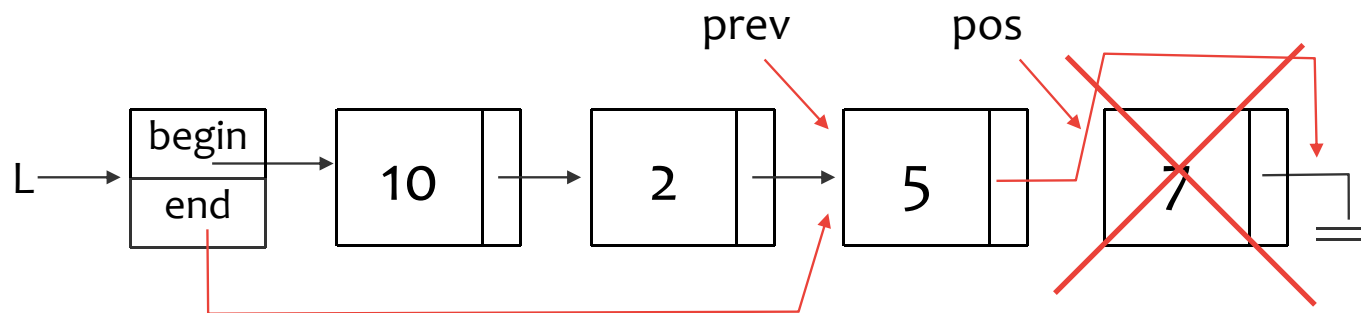
remover 7



Remoção de Elementos da Lista

Caso 3: Elemento está na **cauda (final)** da lista

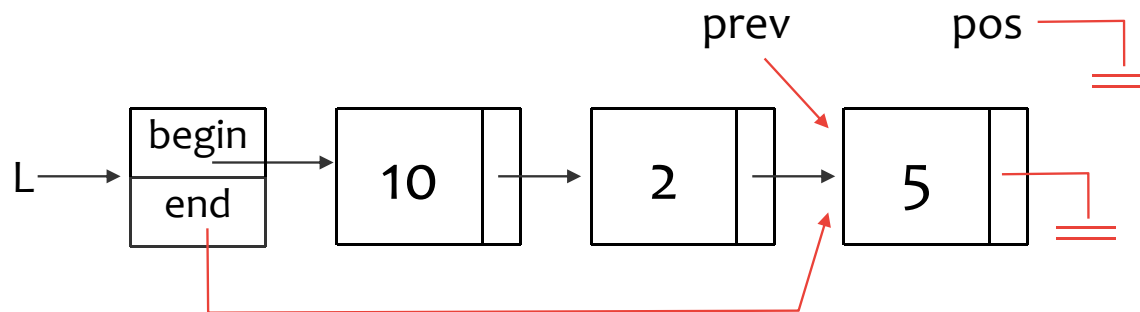
remover 7



Remoção de Elementos da Lista

Caso 3: Elemento está na **cauda (final)** da lista

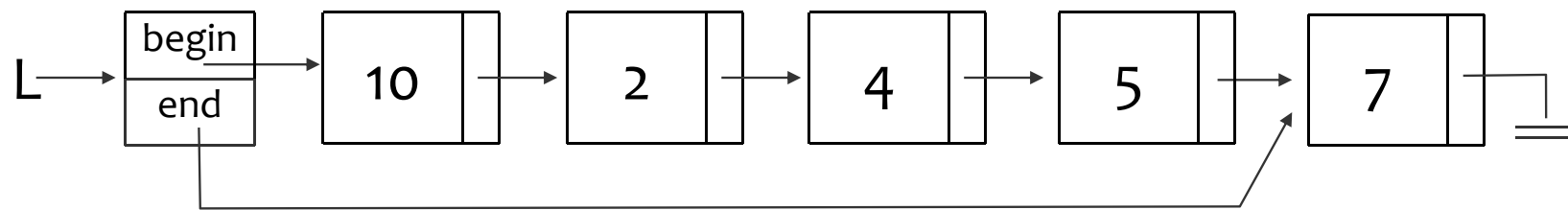
remover 7



Lista Encadeada Simples

Diversos tipos de operações:

- Inserção na cabeça (início) da lista;
- Impressão dos Elementos da Lista
- Inserção na cauda (fim) da lista;
- Remover elementos da lista;
- Contar o número de elementos da Lista;
- Verificar se a lista está vazia e retornar verdadeiro/falso
- Retornar o primeiro elemento;
- Retornar o último elemento;
- Retornar um elemento na posição i



Exercícios

- Contar o número de elementos da Lista;
- Inserir um elemento em uma dada posição
- Inverter uma lista
- Copiar/clonar uma lista;
- Apagar todos os elementos da lista
- Concatenar de duas listas;
- Ordenação de uma lista;
- Buscar um dado elemento na lista e retornar seu ponteiro;
- Inserção ordenada;

Array x Lista Encadeada

Arrays

Vantagens

- Acesso direto ($v[i]$): **rápido**
- Dados **contíguos** na memória
- Percorrer os elementos do array é **rápido**

Desvantagens

- Tamanho do array é fixado (não é flexível)

Listas Encadeadas

Vantagens

- Tamanho é flexível;

Desvantagens

- Acesso de um elemento é **sequencial**;
 - É necessário comparar elemento por elemento;
- Dados não contíguos na memória;
- Percorrer a lista inteira é **lento**;