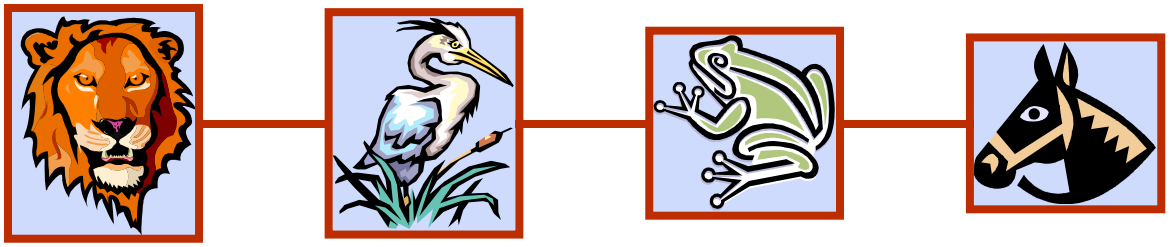
Listas encadeadas

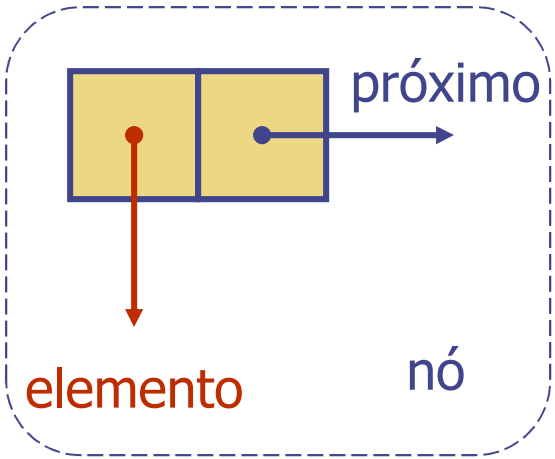


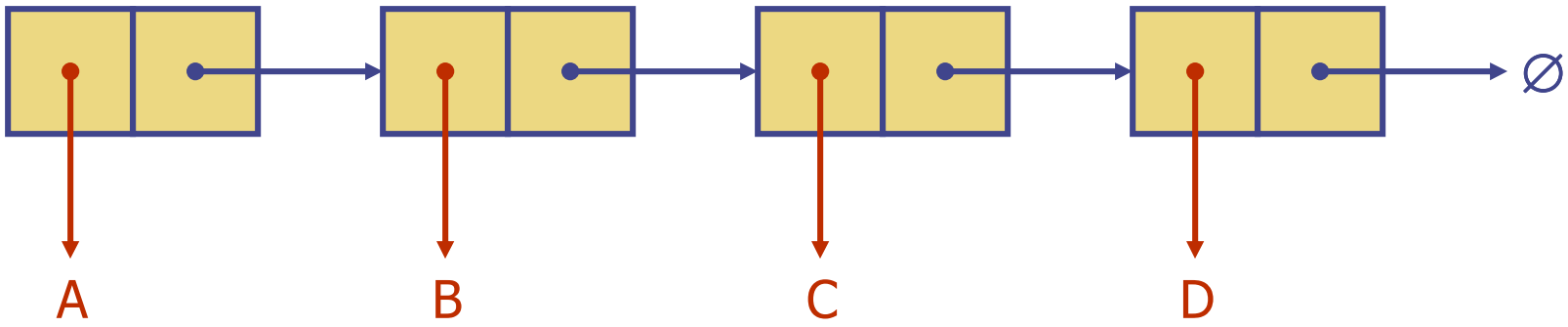
Roteiro

- Lista encadeada
- Lista duplamente encadeada

Lista Encadeada

- Uma lista encadeada é uma estrutura de dados concreta consistindo de uma sequência de nós
- Cada nó armazena
  - Um elemento
  - Uma *ligação* com o próximo nó



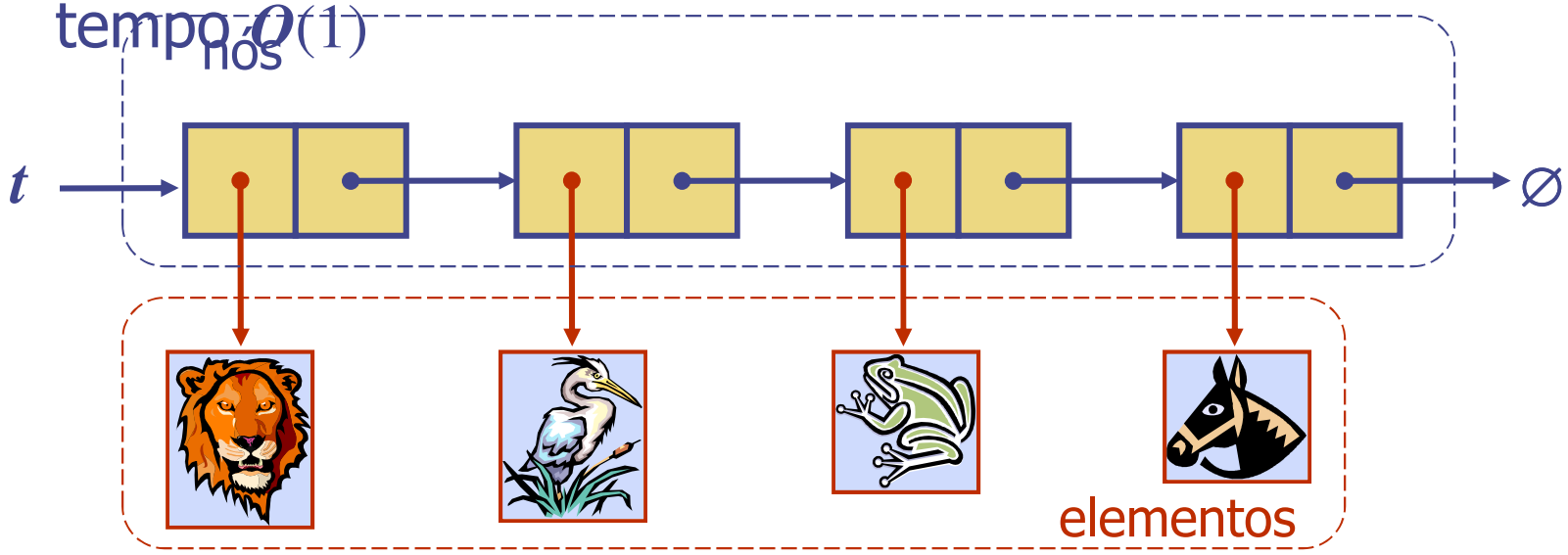


Classe No

```
public class No {
    private Object elemento;
    private No proximo;
    public Object getElemento() {
       return elemento;}
    public void setElemento(Object o){
            elemento = o;
    }
```

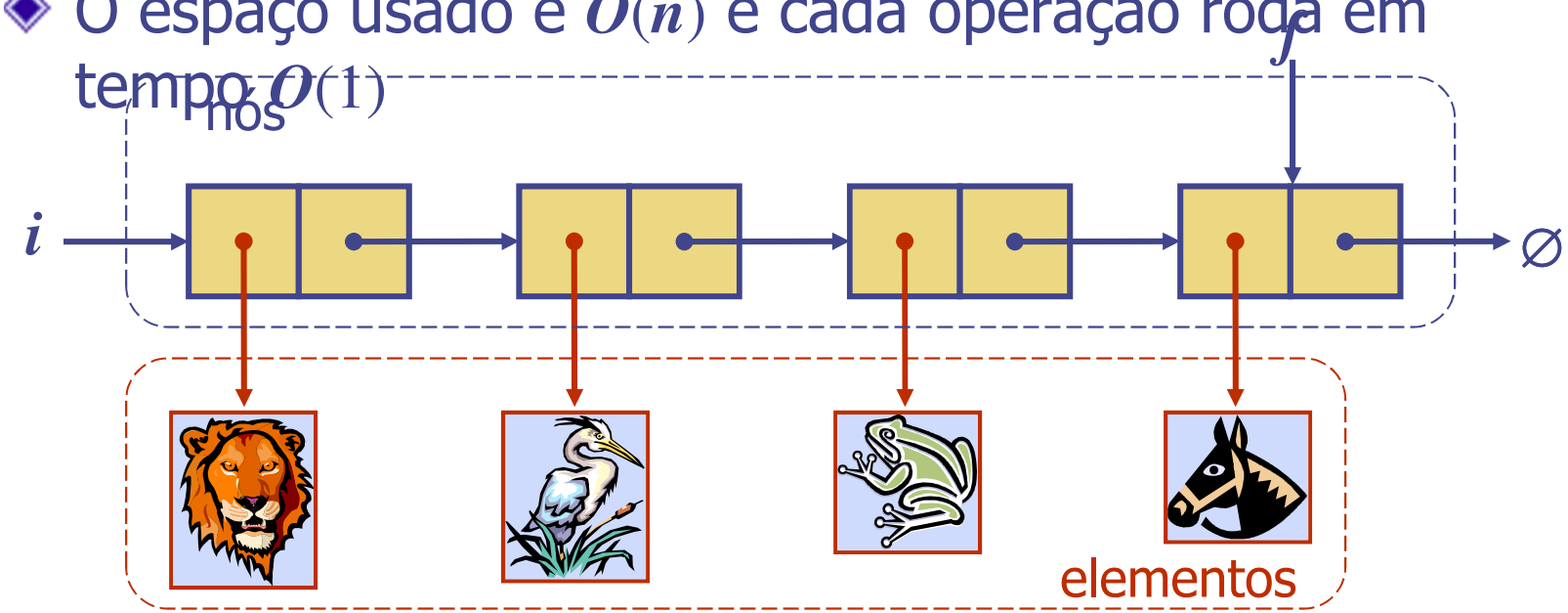
Pilhas com listas encadeadas

- Pode-se implementar uma pilha com uma lista encadeada
- O elemento do topo é armazenado no primeiro nó da lista
- O espaço usado é $O(n)$ e cada operação roda em tempo $O(1)$



Filas com listas encadeadas

- Pode-se implementar uma fila com uma lista encadeada
  - O elemento do início é o primeiro nó
  - O elemento do fim é o último nó
- O espaço usado é $O(n)$ e cada operação roda em tempo $O(1)$



TAD Posição

- O TAD Posição modela a noção de lugar no qual um dado da estrutura é armazenado
- Ele dá uma visão unificada das diversas formas de armazenar dados, tais como:
  - Uma célula em um *array*
  - Um nó em uma lista encadeada
- Possui apenas um método:
  - object element(): Retorna o elemento armazenado nesta posição

TAD Lista

- O TAD Lista modela uma sequência de elementos com suas posições
- Ele estabelece uma relação antes/depois entre as posições
- Métodos genéricos:
  - size(), isEmpty()
- Métodos de consulta:
  - isFirst(p), isLast(p)

Métodos de acesso:

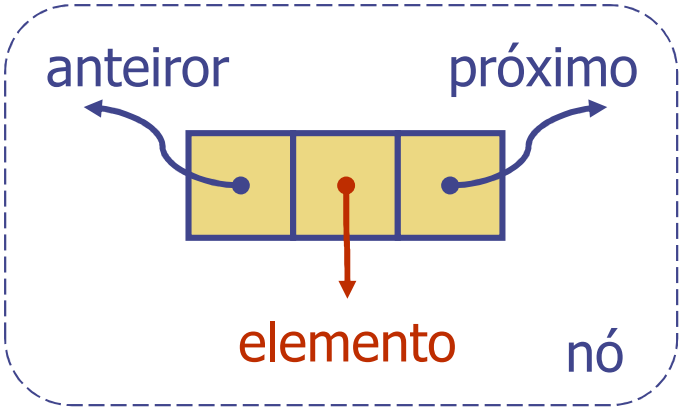
- first(), last()
- antes(p), depois(p)

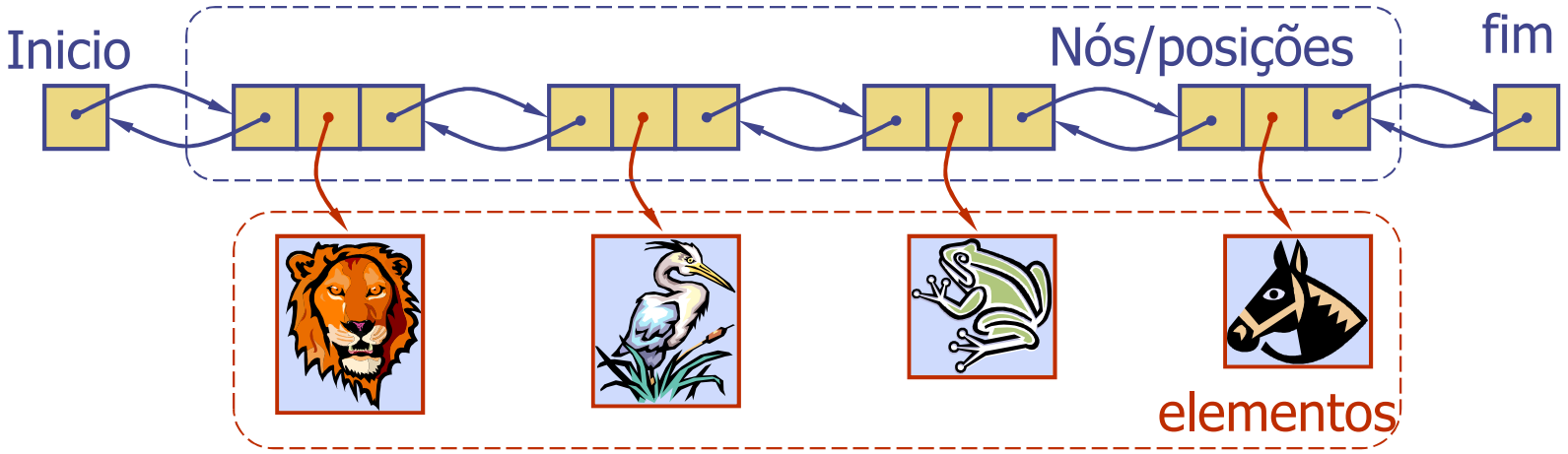
Métodos de atualização:

- replaceElement(p, o), swapElements(p, q)
- insertBefore(p, o), insertAfter(p, o),
- insertFirst(o), insertLast(o)
- remove(p)

Lista Duplamente encadeada

- Semelhante a lista encadeada
- Cada nó armazena:
  - elemento
  - Referência ao nó anterior
  - Referência ao próximo nó
- Nós especiais para o início e fim



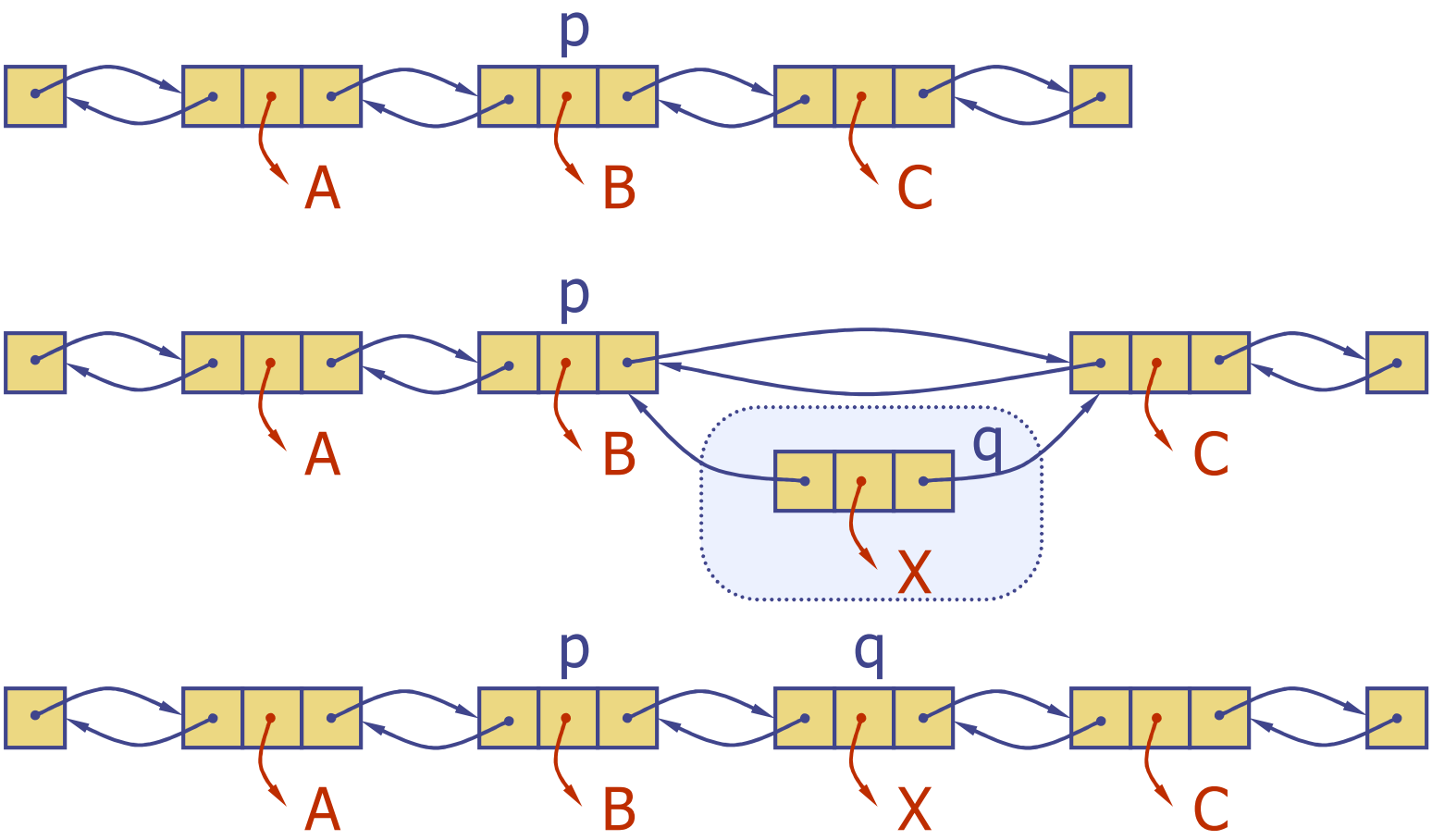


Classe No2

```
public class No {
        private Object elemento;
        private No anterior,proximo;
        public Object getElemento() {
                return elemento;
        }
        public void setElemento(Objecto){
                elemento = o;
        }
}
```

Inserção

- A operação insertAfter(p, X), que retorna uma posição q



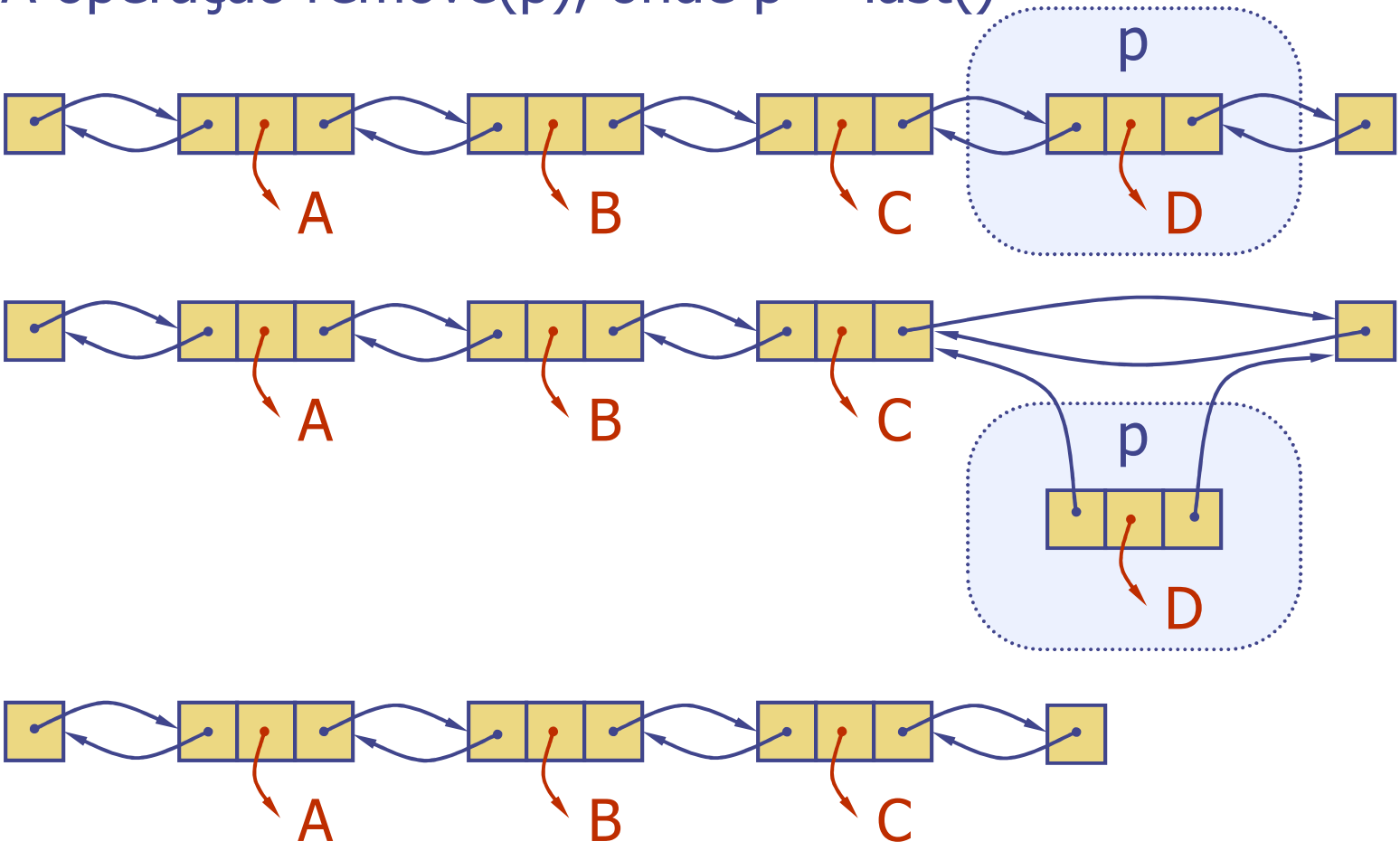
Algoritmo de inserção

Algoritmo insertAfter(p,e):

Criar novo nó v
v.setElement(e)
v.setPrev(p) {v referencia seu anterior}
v.setNext(p.getNext()) {v referencia seu posterior}
(p.getNext()).setPrev(v) {anterior do próximo de p agora é v}
p.setNext(v) {próximo de p é o novo nó v}
return v {A posição do elemento e}

Remoção

- A operação remove(p), onde p = last()



Algoritmo de remoção

Algoritmo remove(p):

t = p.element {Variável temporária para armazenar valor de retorno}
(p.getPrev()).setNext(p.getNext()) {"desreferenciando" p}
(p.getNext()).setPrev(p.getPrev())
p.setPrev(**null**) {invalidando a posição p}
p.setNext(**null**)
return t

Desempenho

- A implementação do TAD Lista usando uma lista duplamente encadeada:
  - O espaço usado pela lista com n elementos é $O(n)$
  - O espaço usado por cada posição na lista é $O(1)$
  - Todas as operações na lista são executadas em tempo $O(1)$
  - A operação element() do TAD Posição executa em tempo $O(1)$