



# Listas

## ESTRUTURA DE DADOS E ALGORITMOS I

Prof. Pila

# Alocação Estática: Vetor

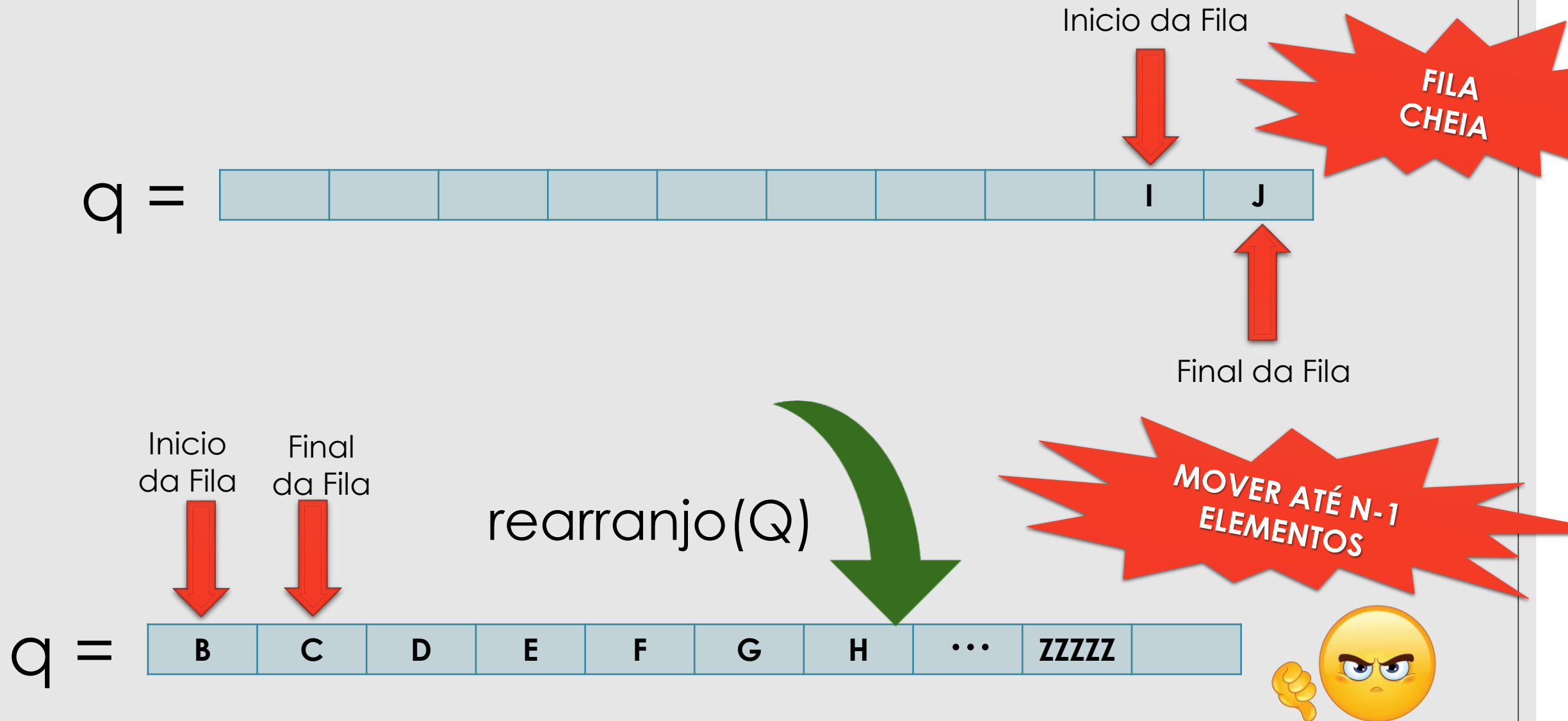
1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

1									
---	--	--	--	--	--	--	--	--	--

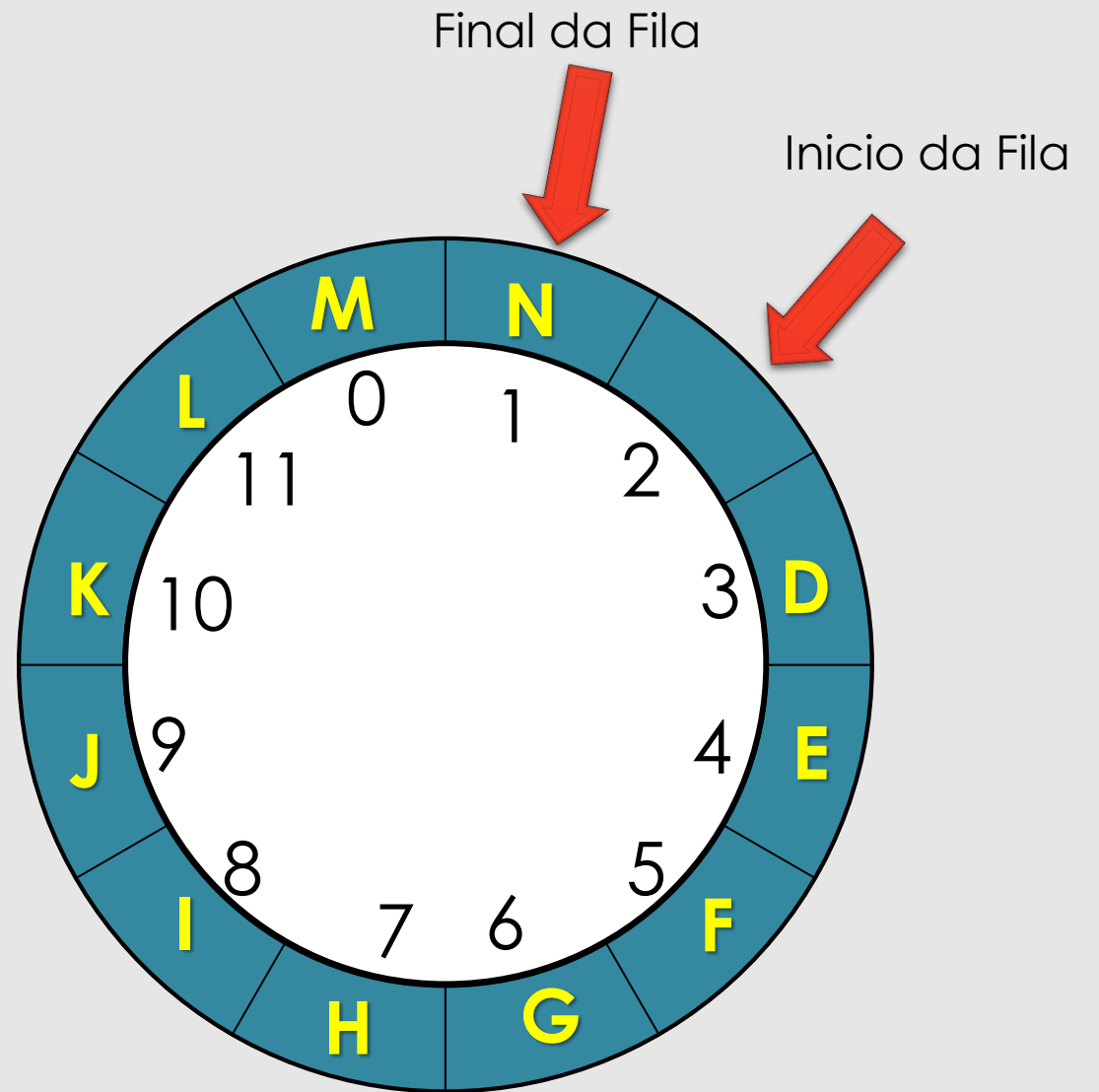
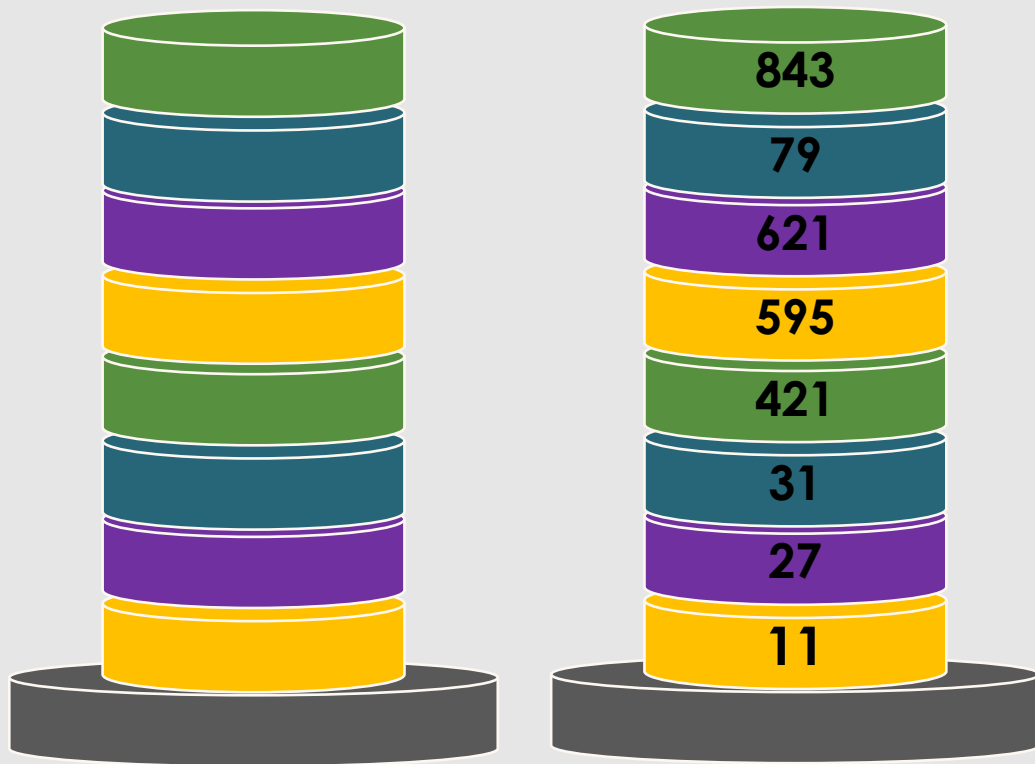
1	2	3							
---	---	---	--	--	--	--	--	--	--

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

# Alocação Estática: Pilhas e Filas



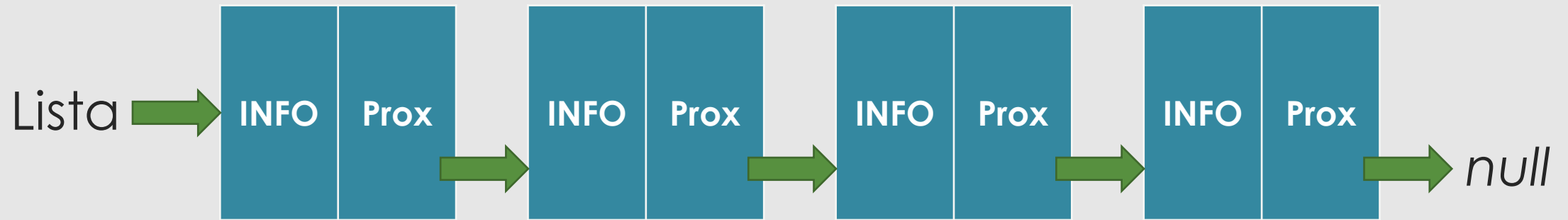
# Alocação Estática: Pilhas e Filas



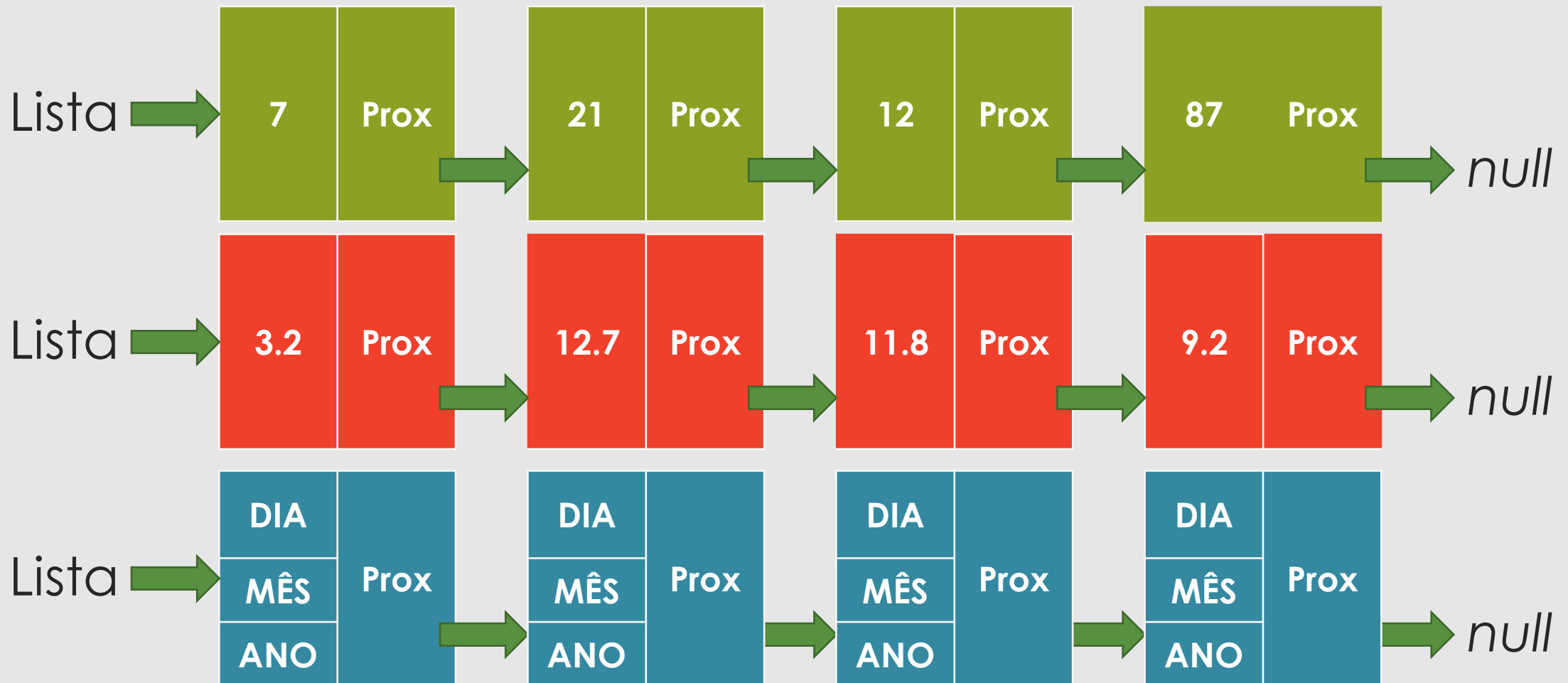
# Alocação Estática

- Tamanho é fixo !
- Mesmo sem uso da estrutura de dados, ocupa espaço de memória.
- Quando a estrutura está cheia, não é possível alocar mais espaço.
- Rearranjar os elementos da estrutura de dados demanda reorganizar a estrutura como um todo.

# Lista Ligada Linear



# Lista Ligada Linear: INFO pode, int, char, float, até uma **struct**

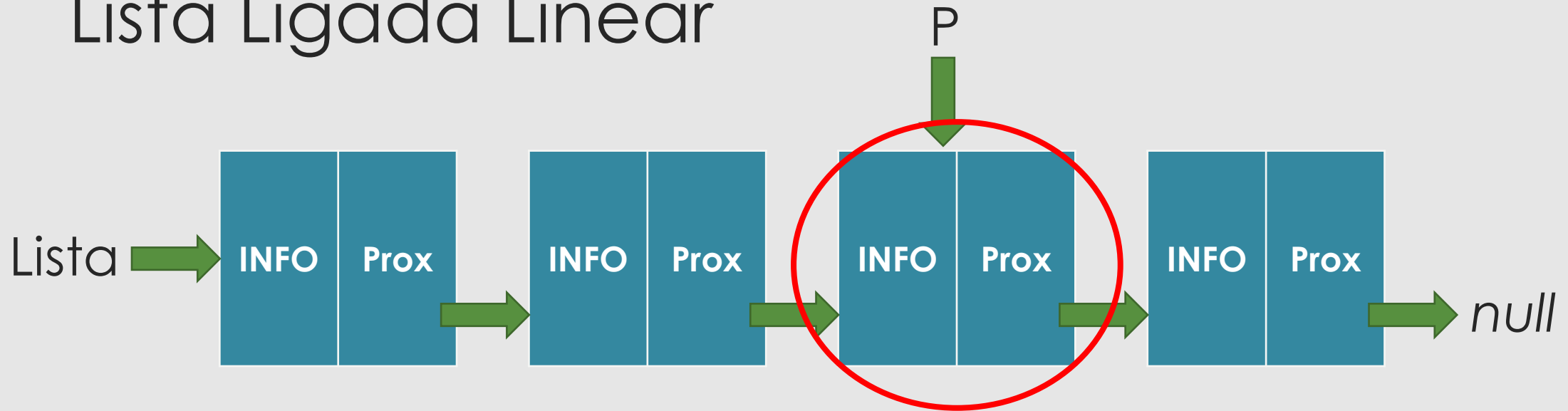


# Lista Ligada Linear

- Alocação é dinâmica !!! 😊
- Utiliza memória somente dos elementos presentes na estrutura de dados.
- Inserir um novo elemento na estrutura de dados, não é problema. Em qualquer posição.
- Não demanda rearranjar a estrutura de dados como um todo.
- Estrutura de dados vazia não ocupa espaço de memória.



# Lista Ligada Linear



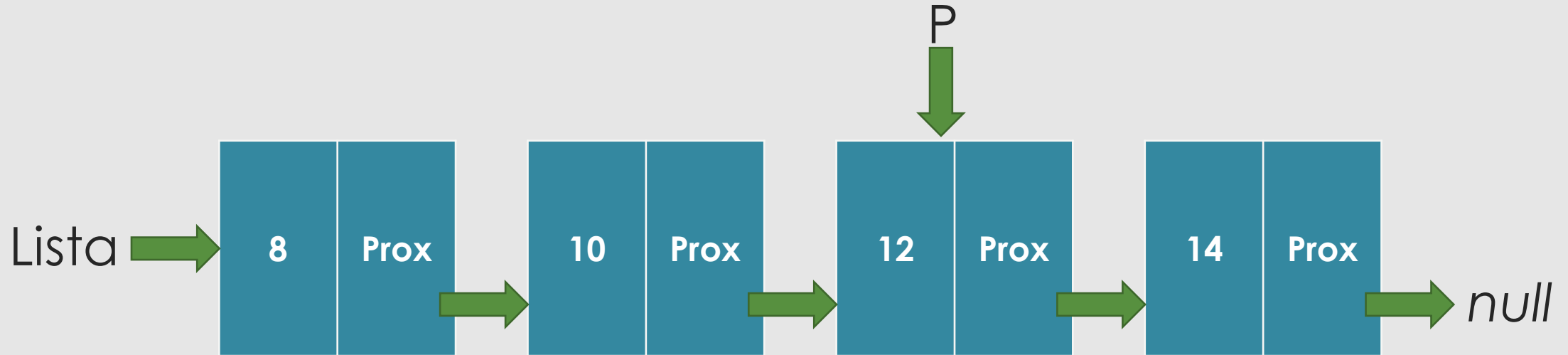
**Lista = *null*** : significa lista vazia

**P** : é um ponteiro para um **nó** da lista ligada

**P->INFO** : conteúdo (INFO) do nó

**P->Prox** : ponteiro para o próximo nó

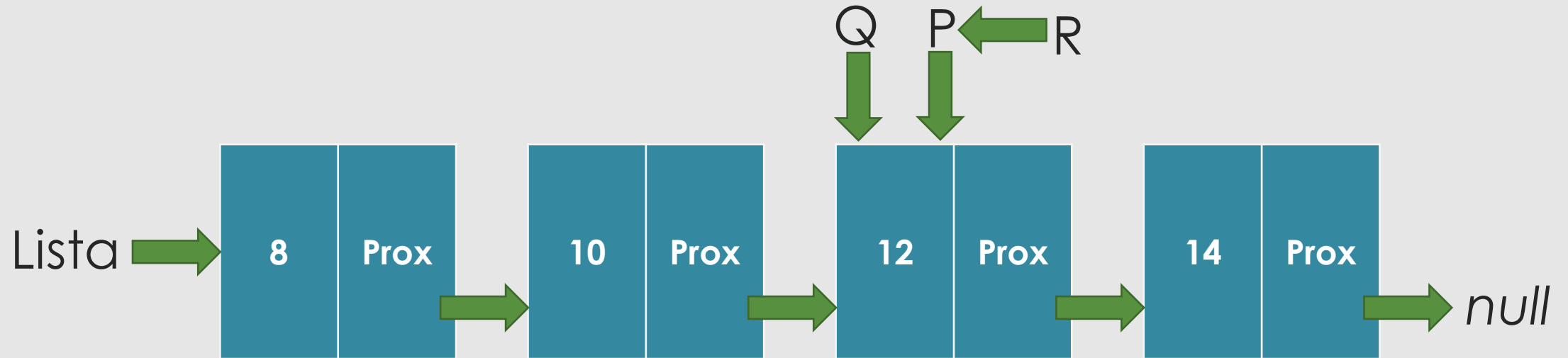
# Lista Ligada Linear



## RESPONDA

- Lista->INFO ?
- Lista->Prox ?
- P ?
- P->Info?
- P->Prox?
- P->Prox->Info?
- Lista->Prox->Prox->Info?

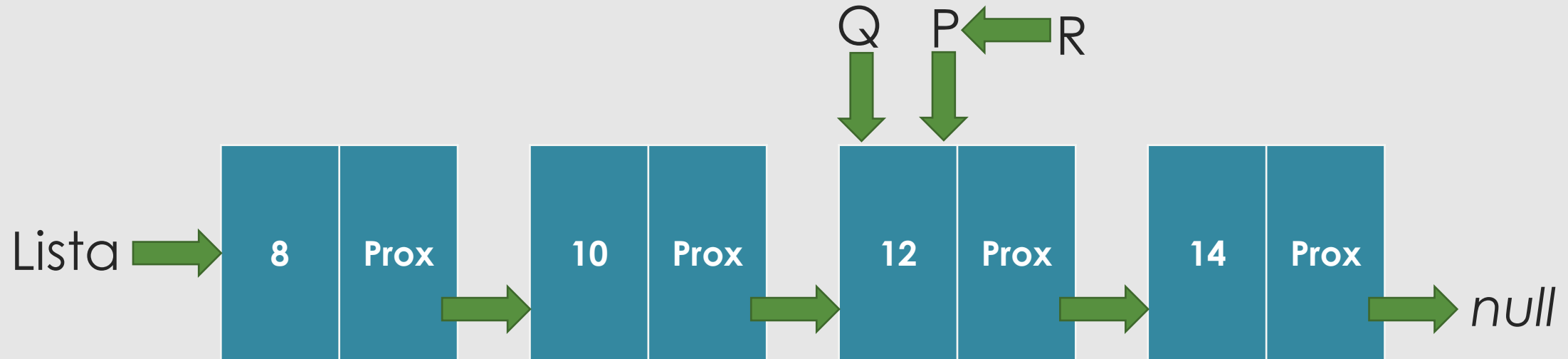
# Lista Ligada Linear



RESPONDA

- Q->Info?
- Q->Prox?
- R->Info?
- R->Prox?

# Lista Ligada Linear



- Quando falamos de ponteiro precisamos entender o que é *região de memória alocada* e o que é somente um ponteiro para essa região.
- Quando um *ponteiro aponta para outro não é o conteúdo, mas a mesma região de memória alocada dinamicamente.*

# Lista Ligada Linear: Definindo o nó

Nome 'no' é  
definido pelo  
programador

```
struct no {  
    int info;  
    struct no *prox;  
};
```

Conteúdo  
variável



Ponteiro  
para este  
tipo de  
estrutura

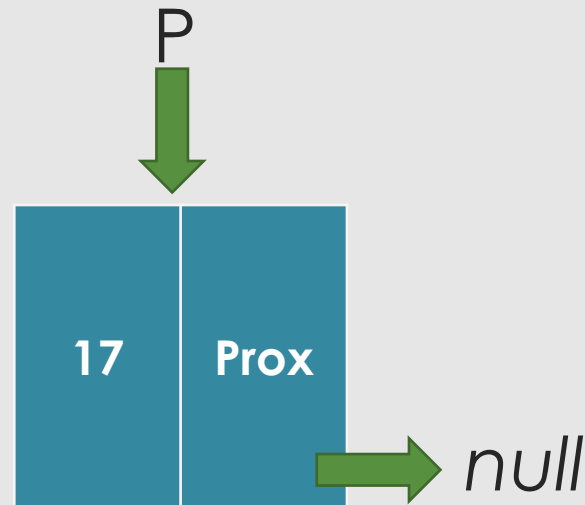
Variáveis deste tipo  
são declaradas  
como 'lista'

# Lista Ligada Linear: Construindo a lista

lista  null

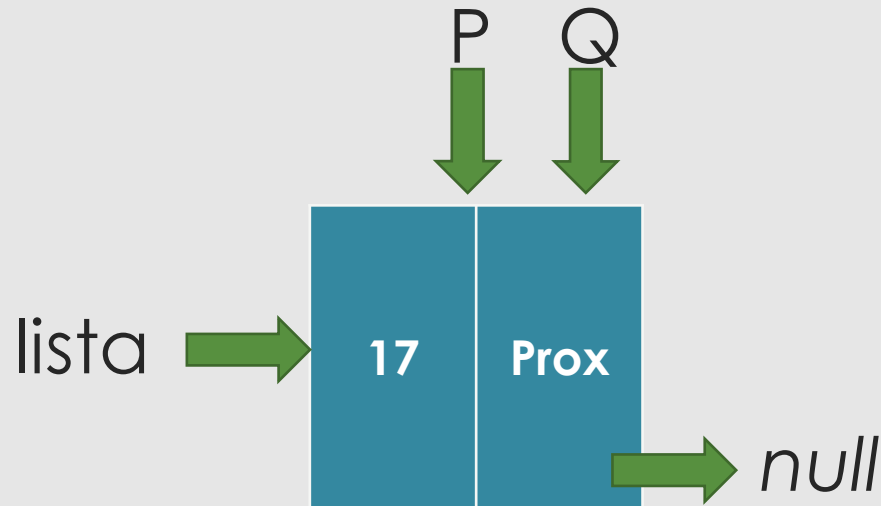
```
no *lista, *p, *q;  
lista = NULL;
```

```
p = (no *) malloc(sizeof(no));  
p->info = 17;  
p->prox = NULL;
```

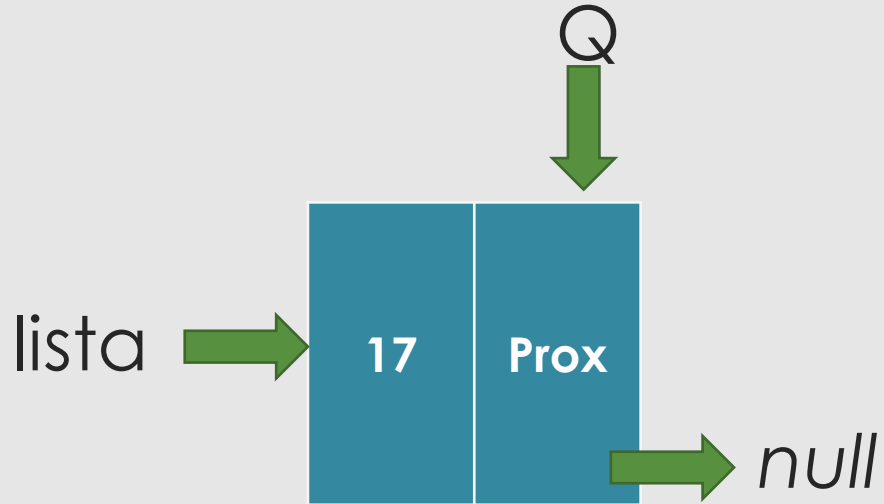


# Lista Ligada Linear: Construindo a lista

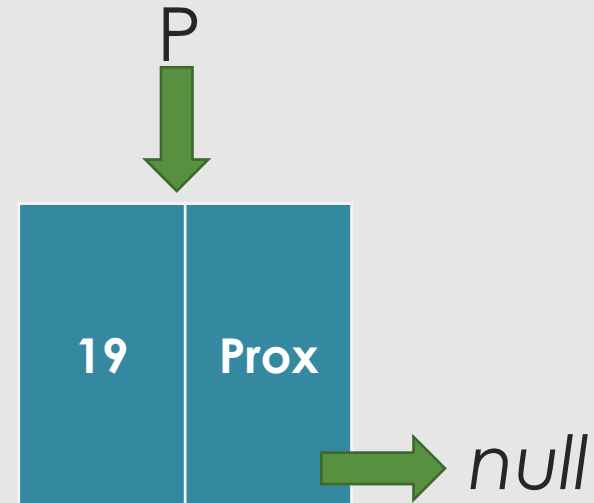
lista = p;  
q = p;



# Lista Ligada Linear: Construindo a lista

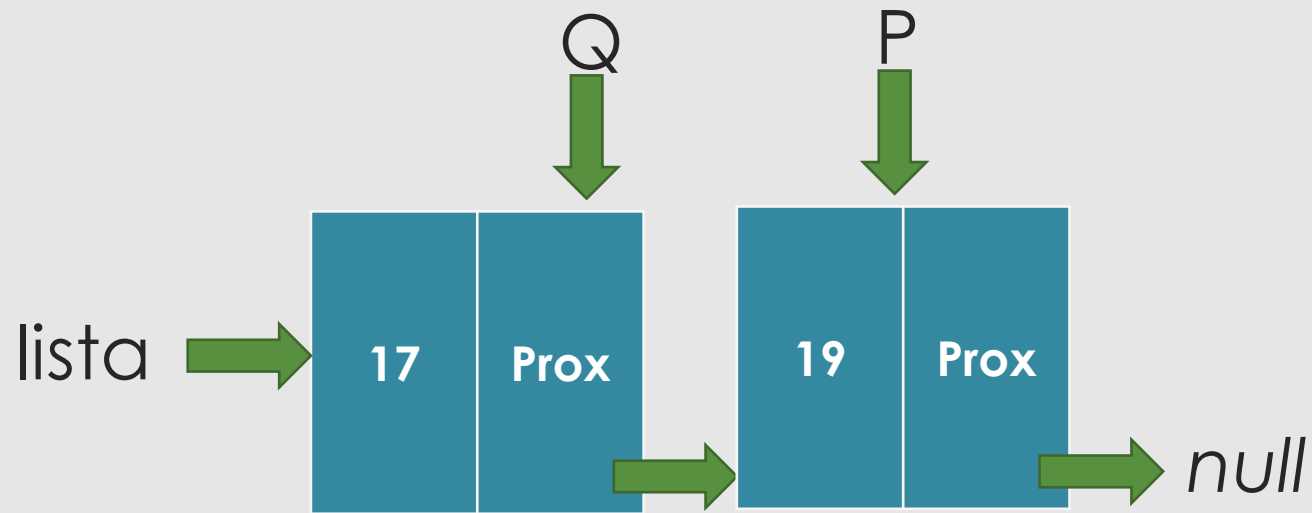


```
p = (no *) malloc(sizeof(no));  
p->info = 19;  
p->prox = NULL;
```



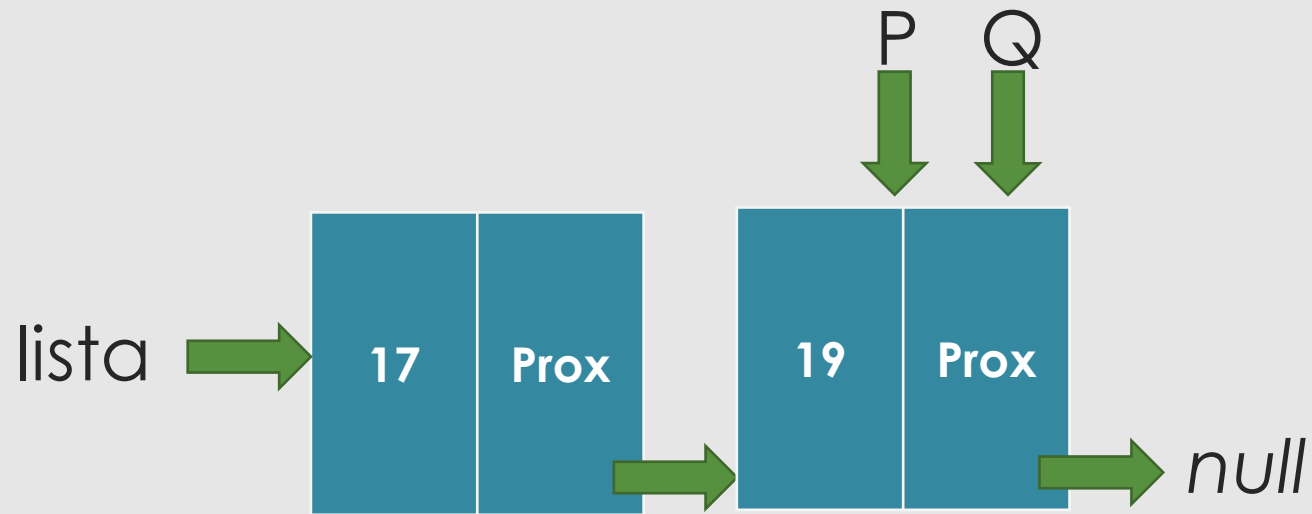


# Lista Ligada Linear: Construindo a lista



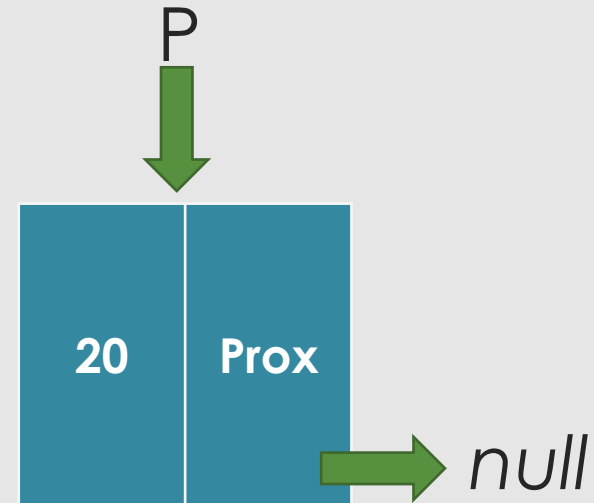
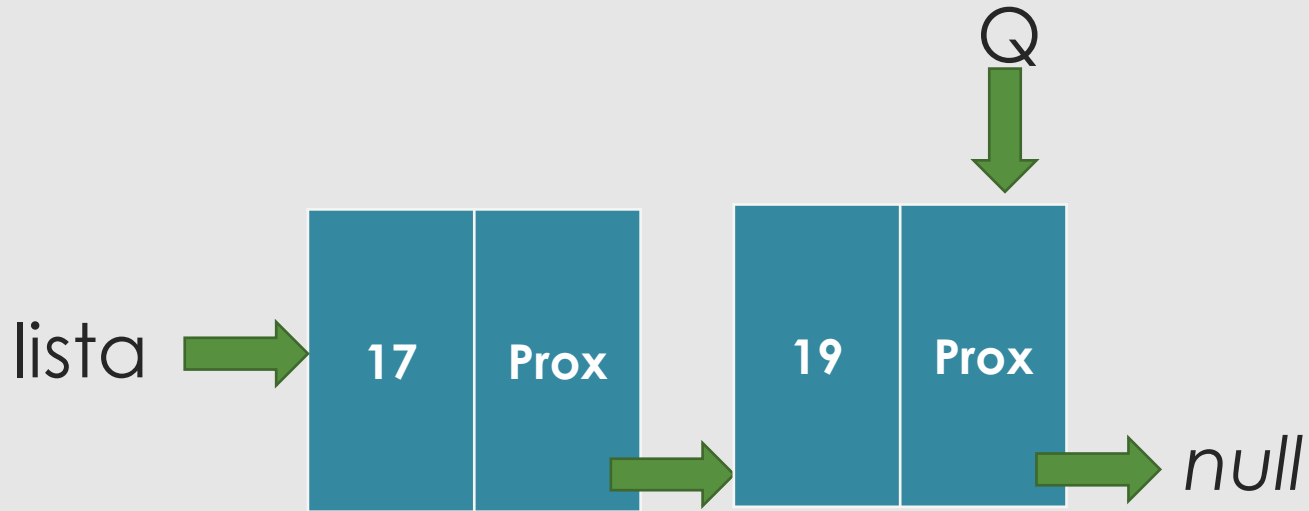
`q->prox = p;`

# Lista Ligada Linear: Construindo a lista



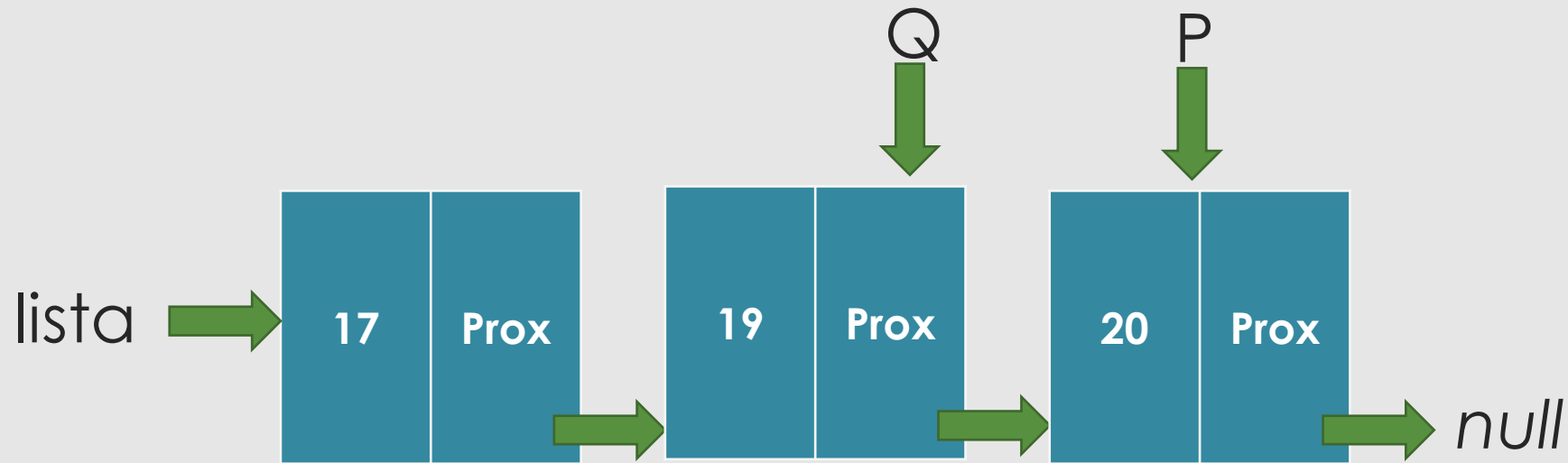
```
q = q->prox;
```

# Lista Ligada Linear: Construindo a lista



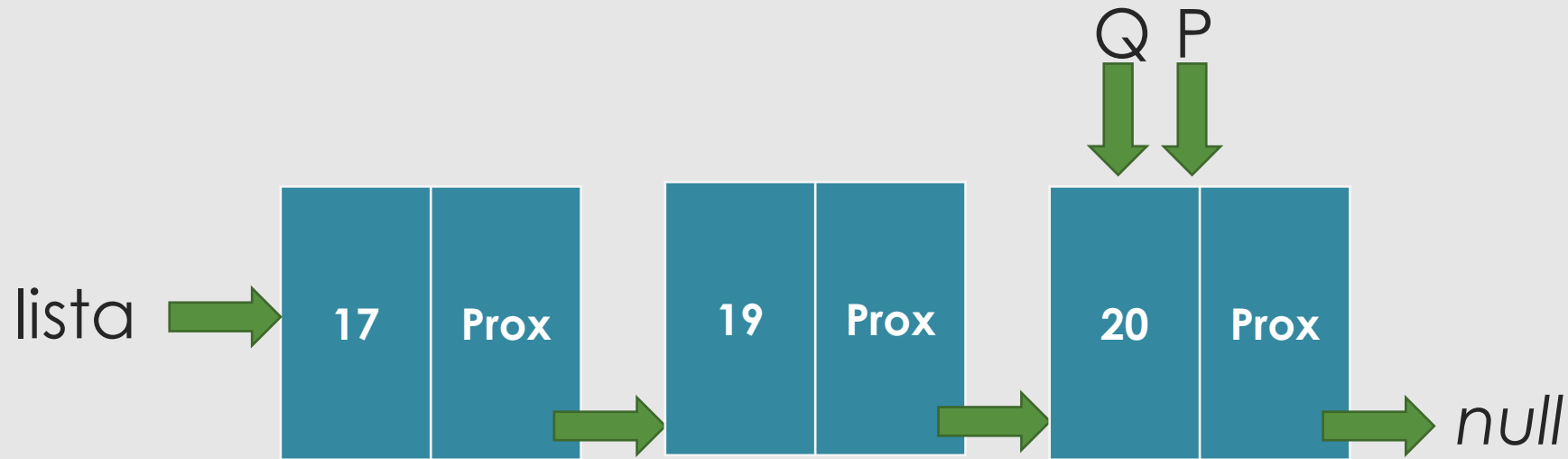
```
p = (no *) malloc(sizeof(no));  
p->info = 20;  
p->prox = NULL;
```

# Lista Ligada Linear: Construindo a lista



q->prox = p;

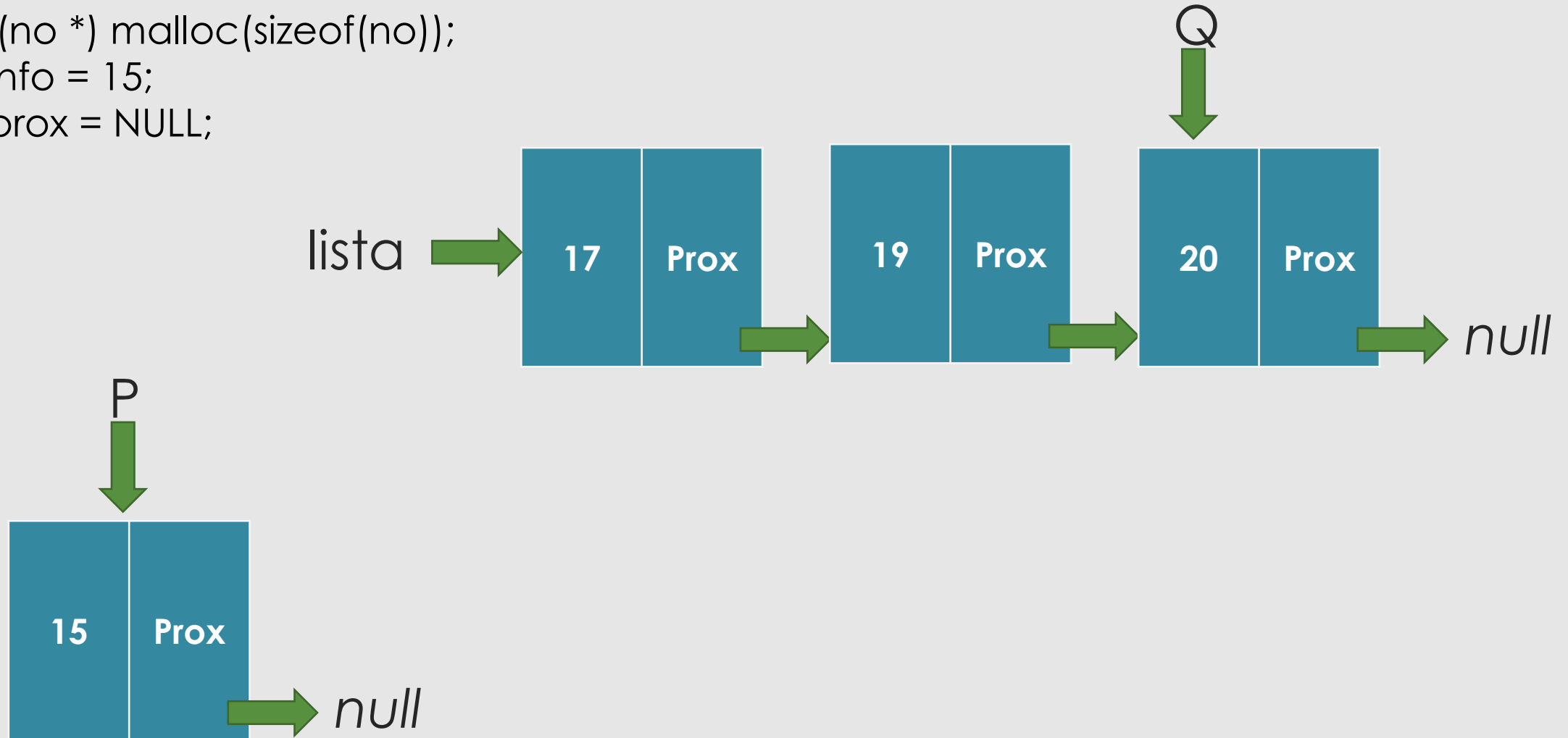
# Lista Ligada Linear: Construindo a lista



```
q = q->prox;
```

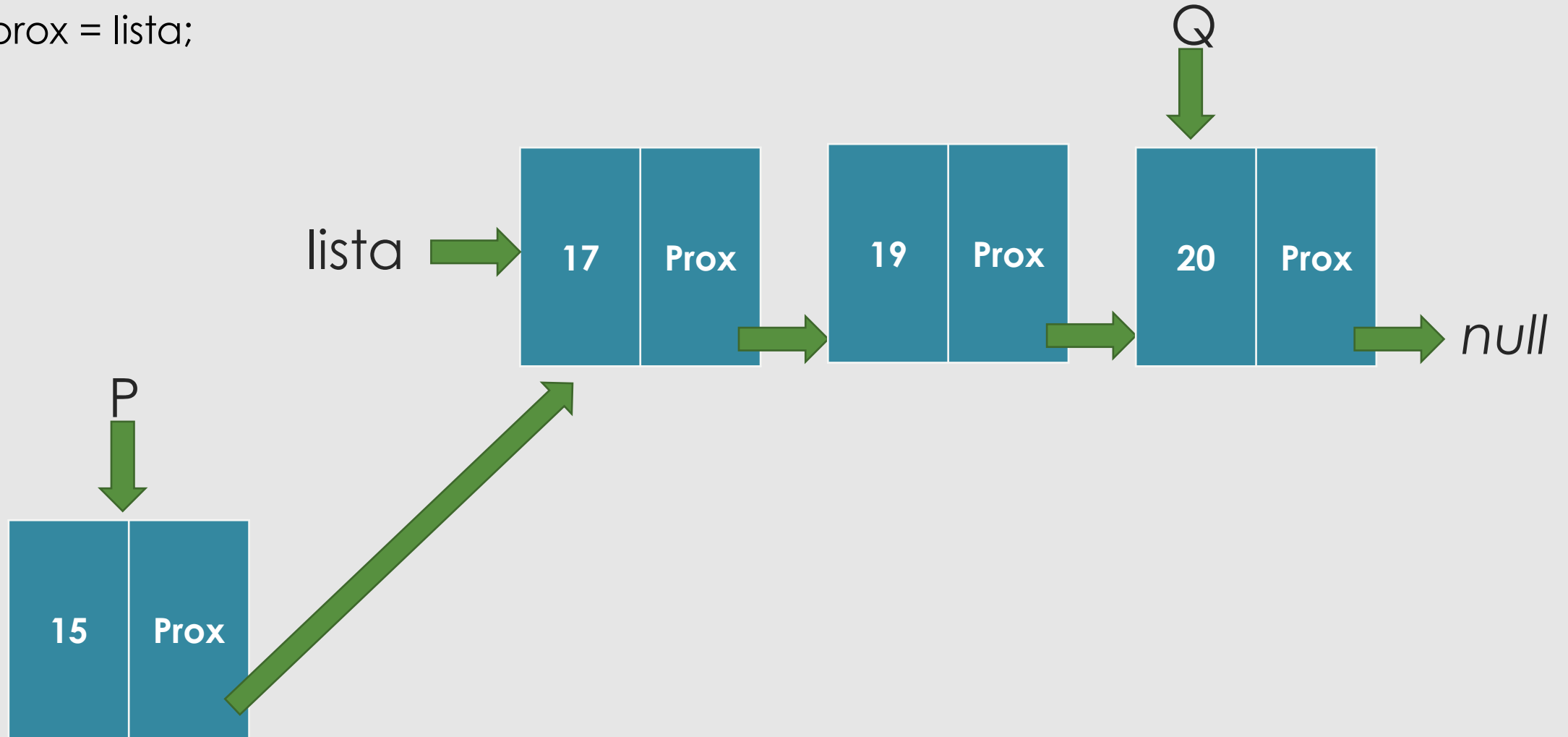
# Lista Ligada Linear: Inserção no início

```
p = (no *) malloc(sizeof(no));  
p->info = 15;  
p->prox = NULL;
```



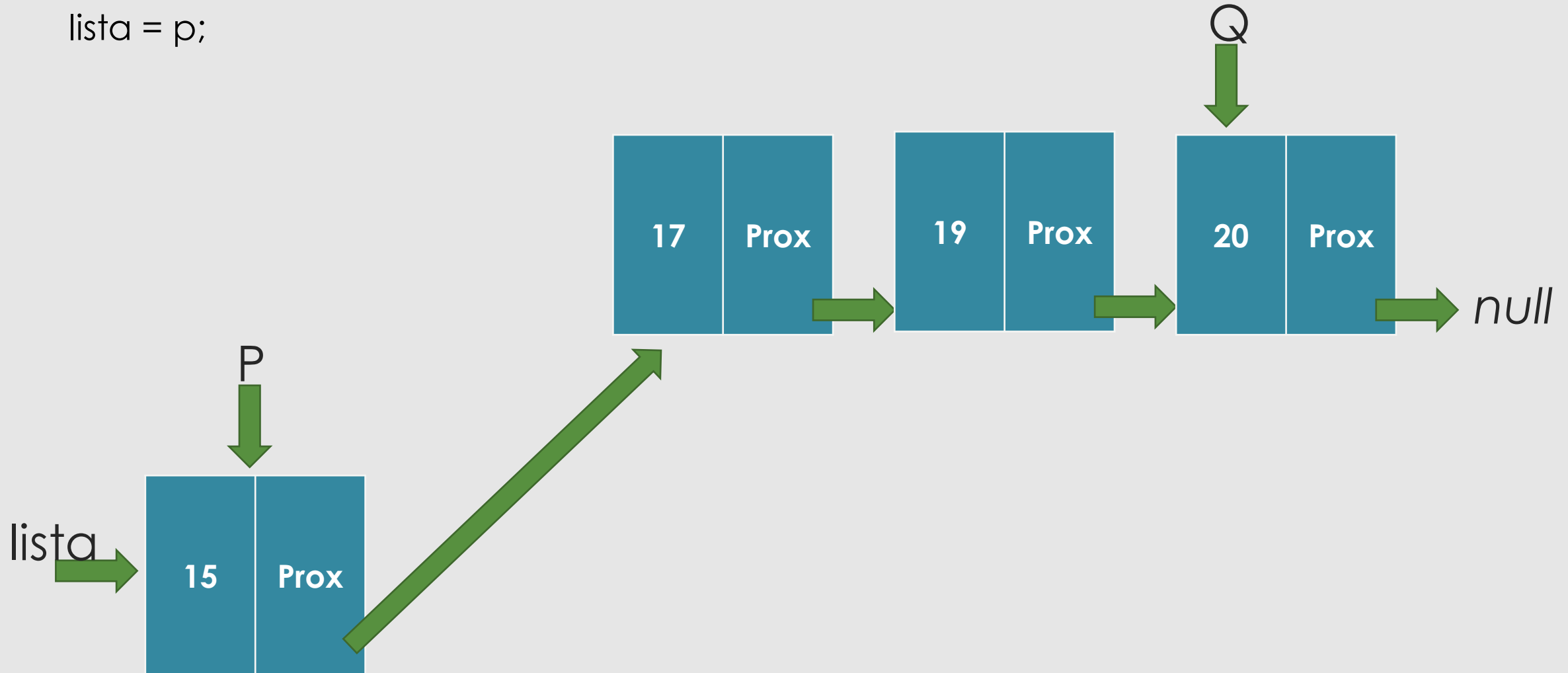
# Lista Ligada Linear: Inserção no início

`p->prox = lista;`



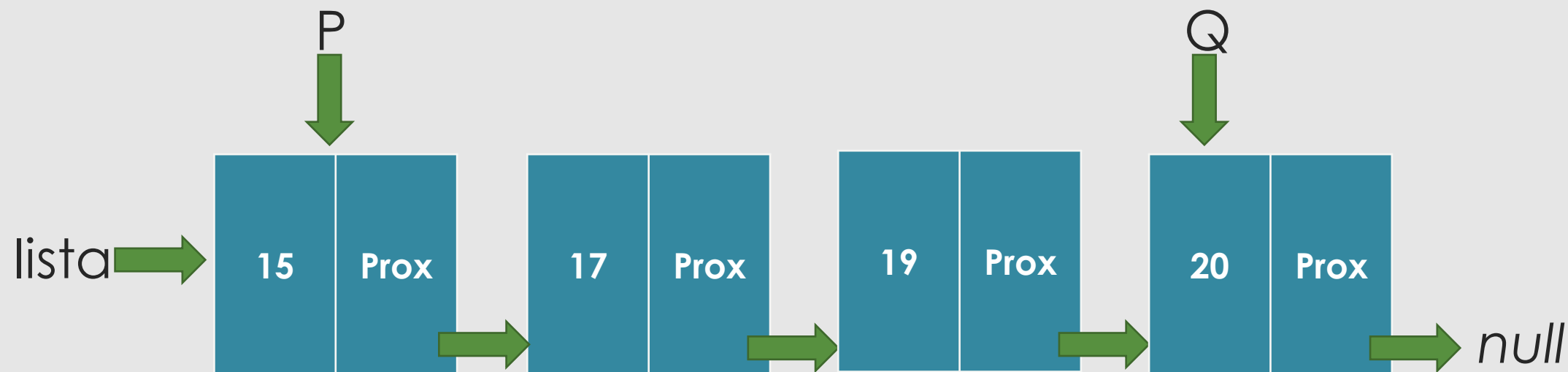
# Lista Ligada Linear: Inserção no início

lista = p;

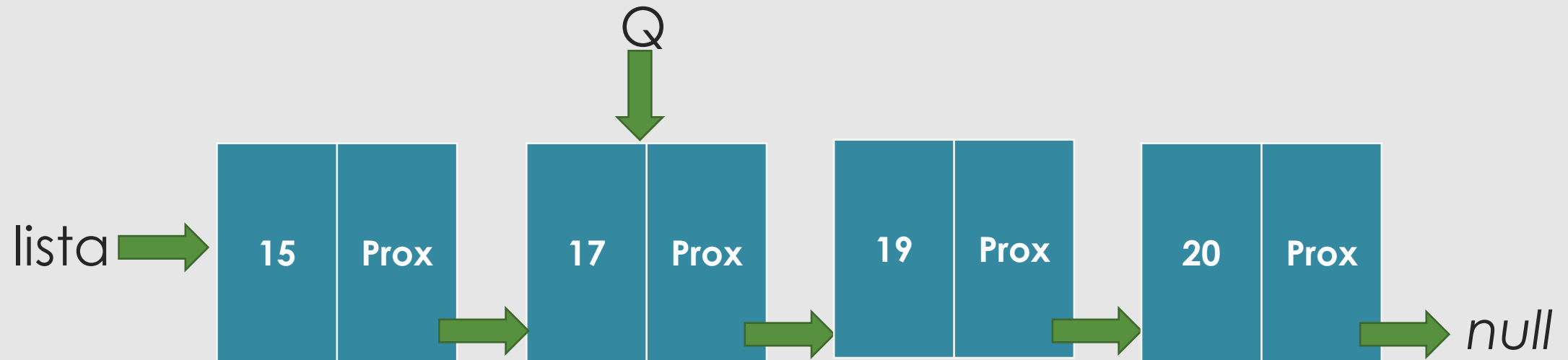




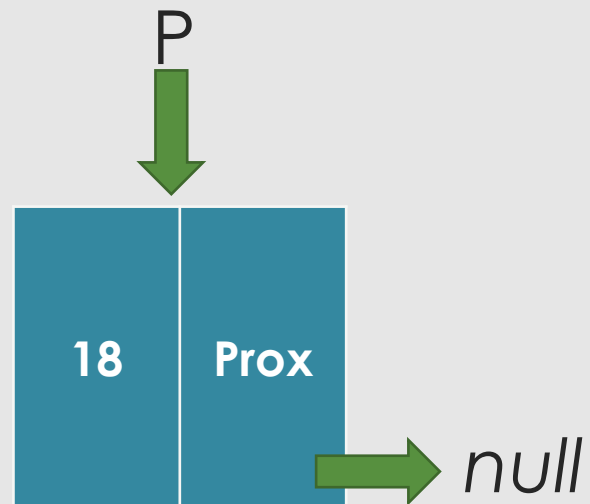
# Lista Ligada Linear: Inserção no início



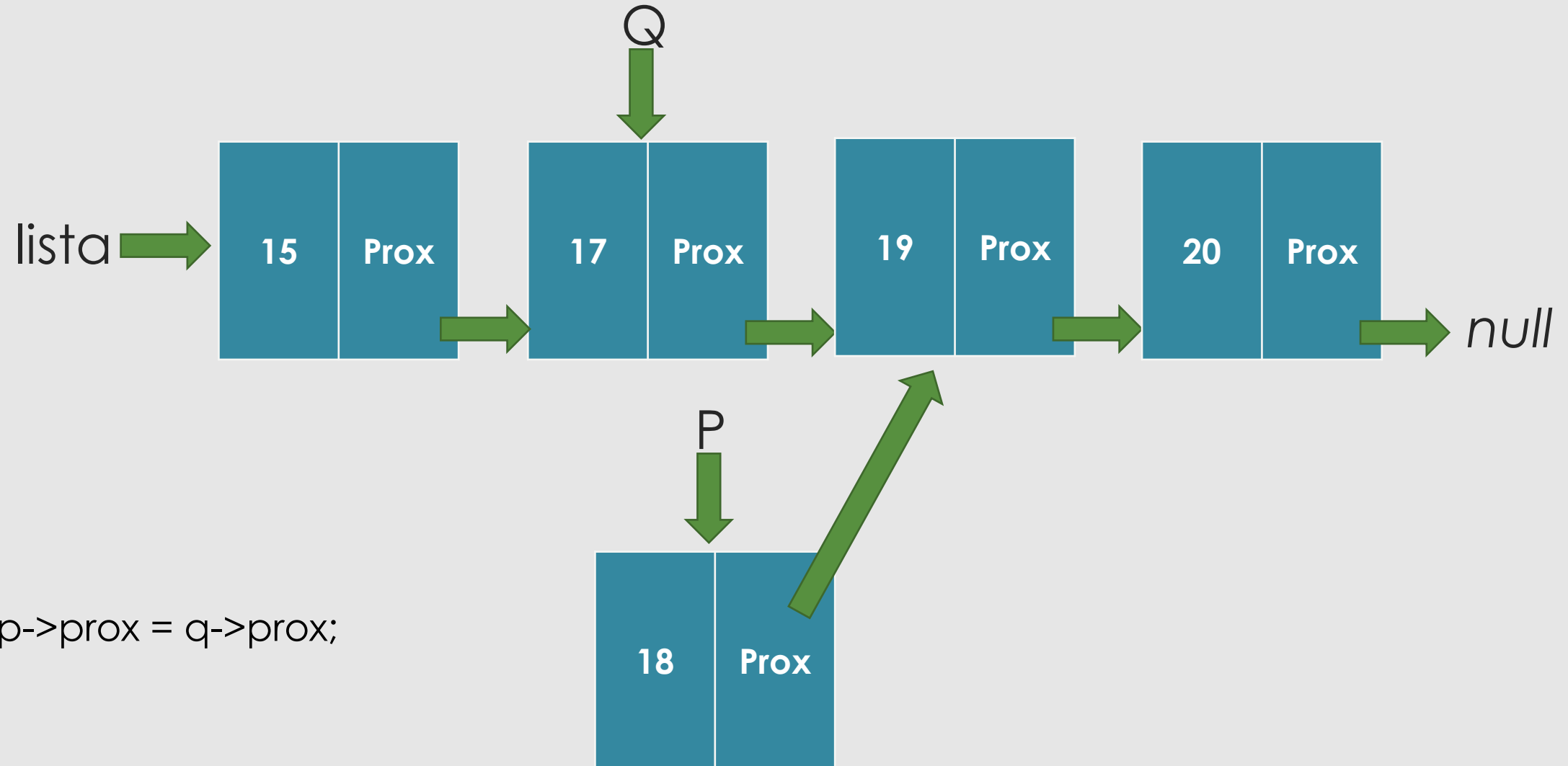
# Lista Ligada Linear: Inserção no meio



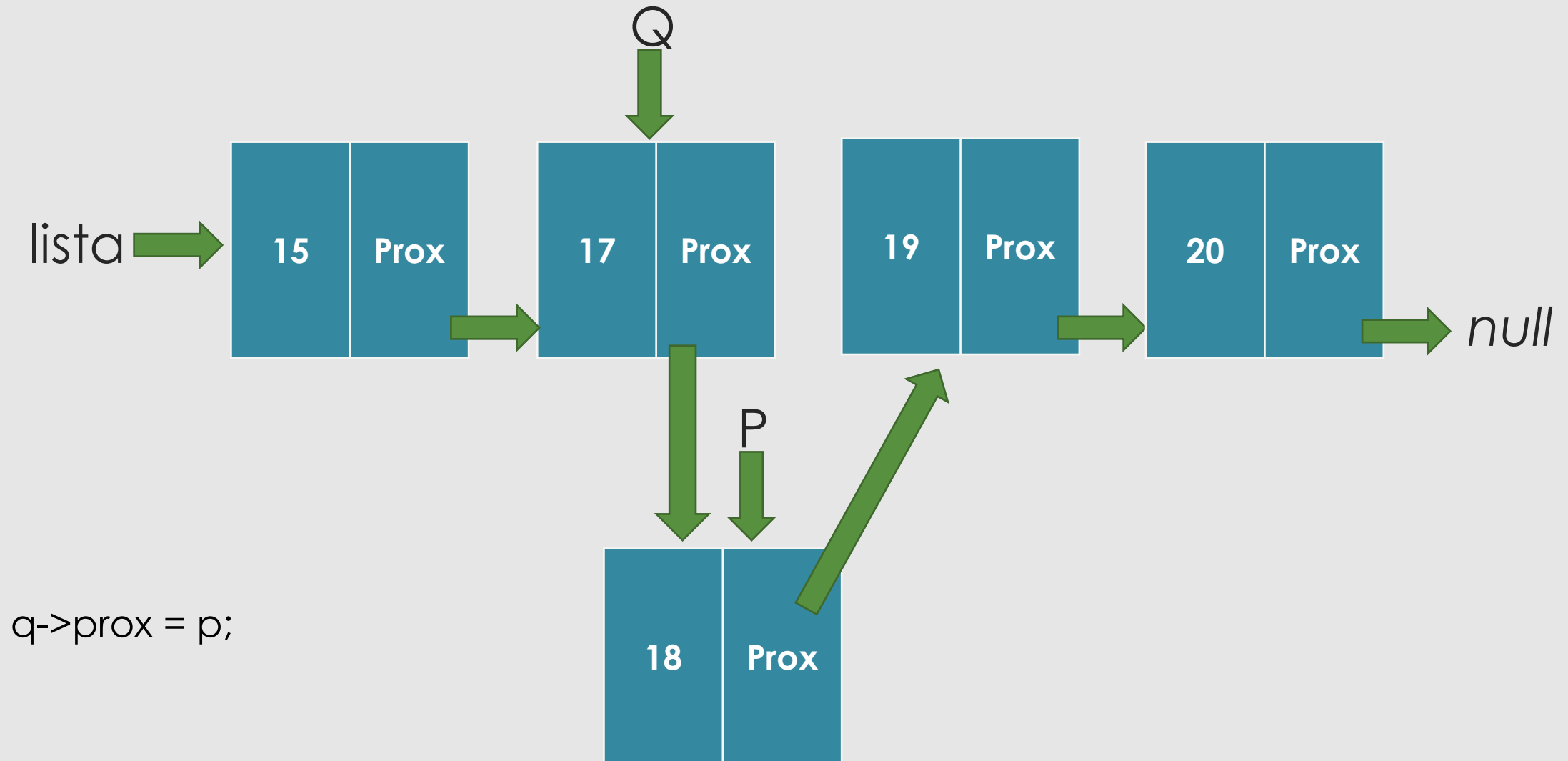
```
p = (no *) malloc(sizeof(no));  
p->info = 18;  
p->prox = NULL;
```



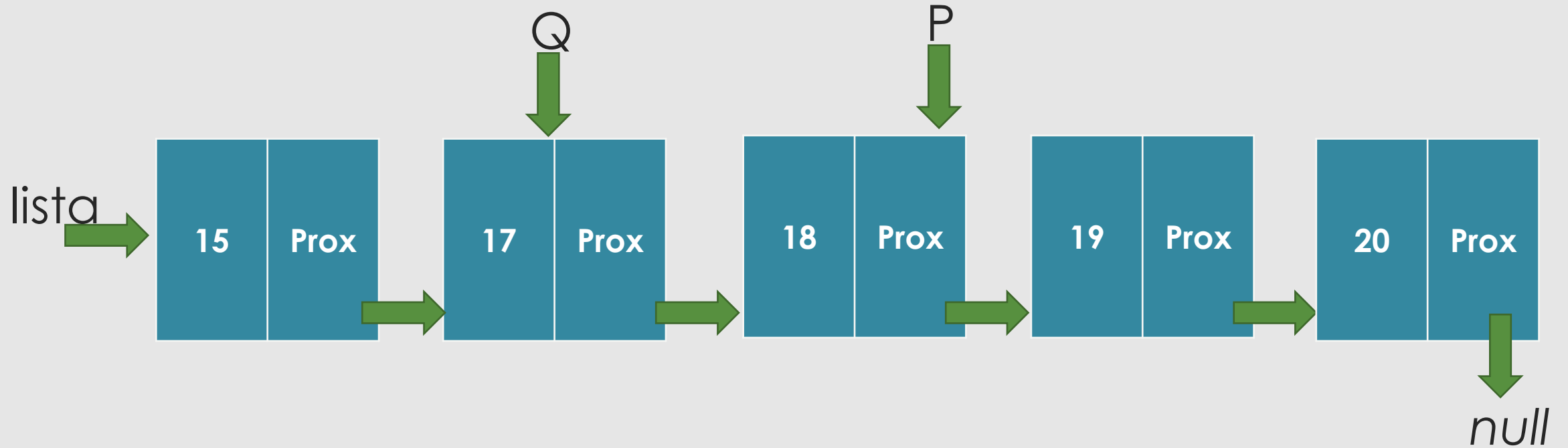
# Lista Ligada Linear: Inserção no meio



# Lista Ligada Linear: Inserção no meio

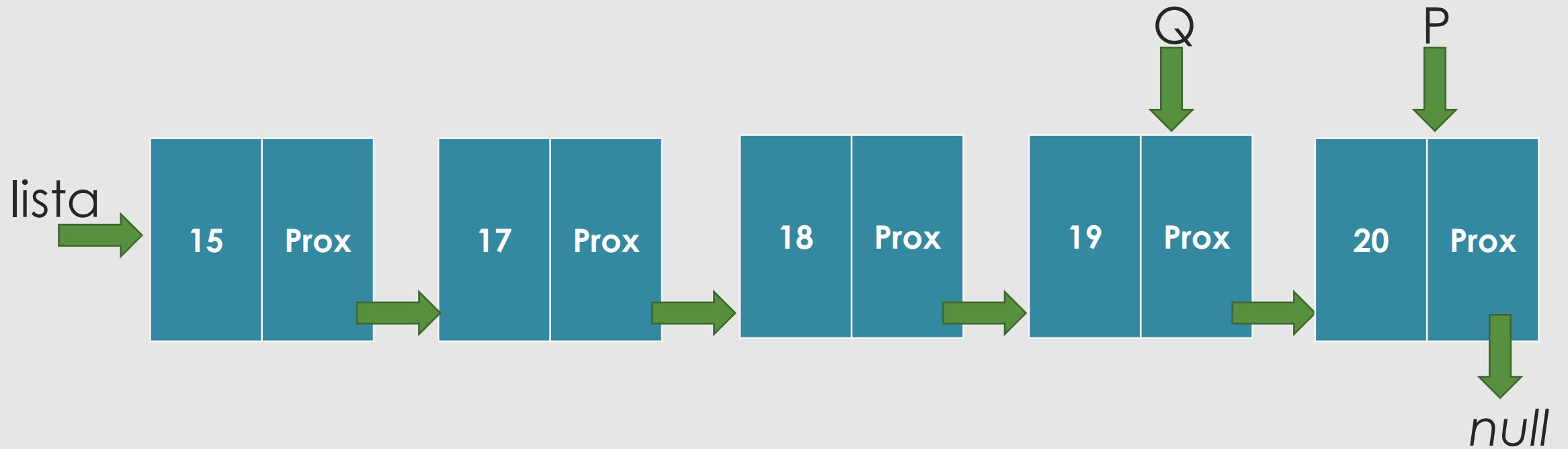


# Lista Ligada Linear: Inserção no meio



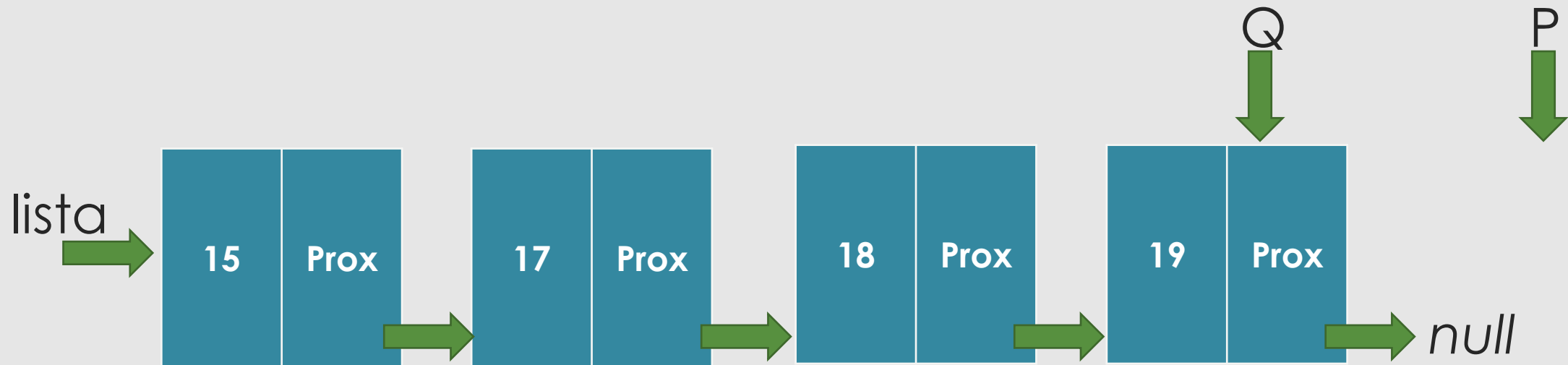
RESULTADO FINAL

# Lista Ligada Linear: Remoção no final

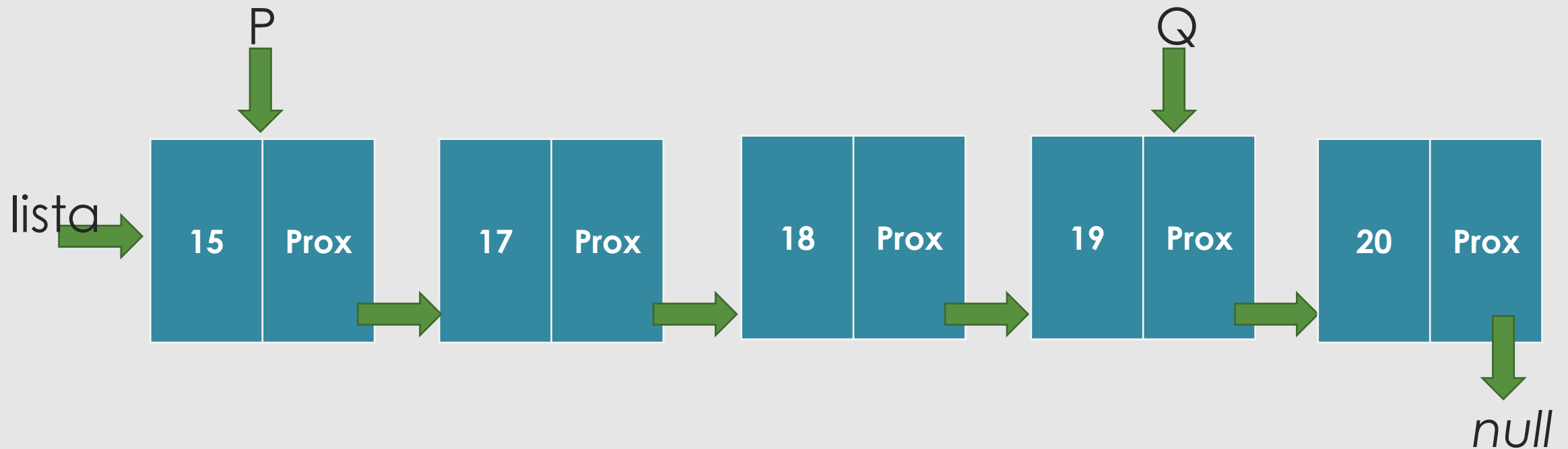


```
q->prox = NULL;  
free(P)
```

# Lista Ligada Linear: Remoção no final



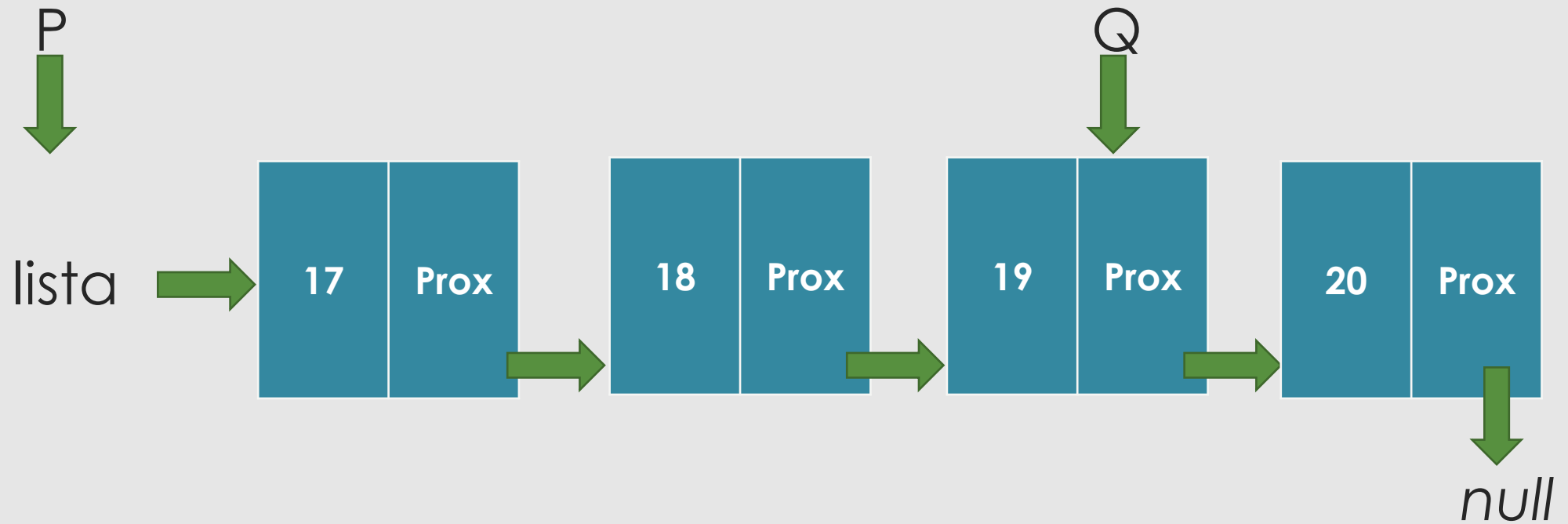
# Lista Ligada Linear: Remoção no início



```
lista => p->prox;  
free(p);
```

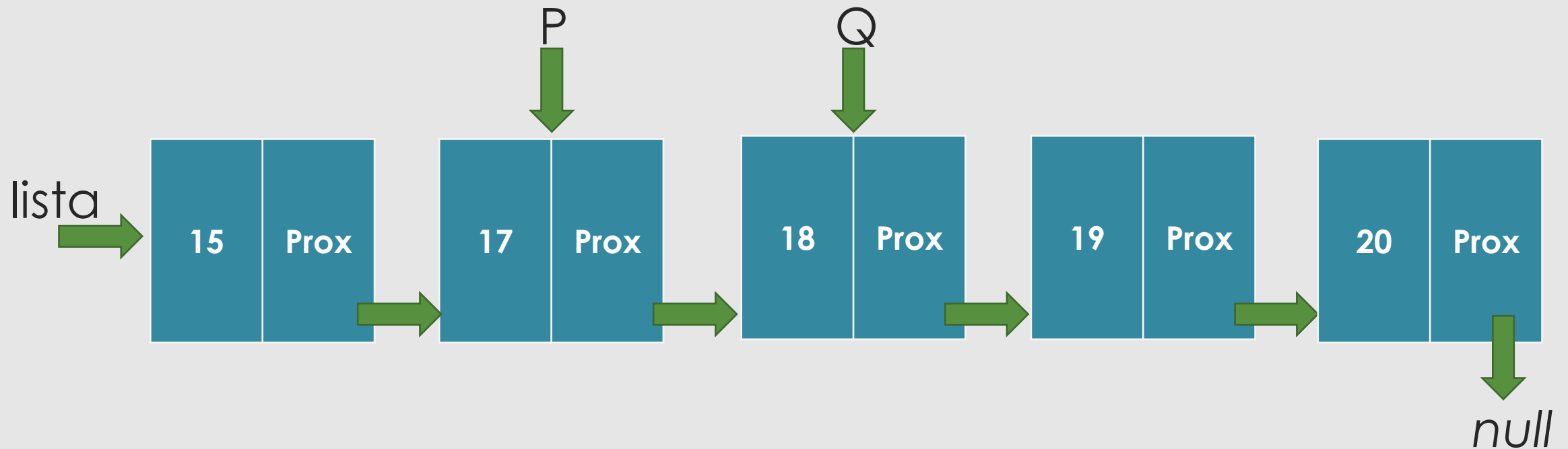


# Lista Ligada Linear: Remoção no início



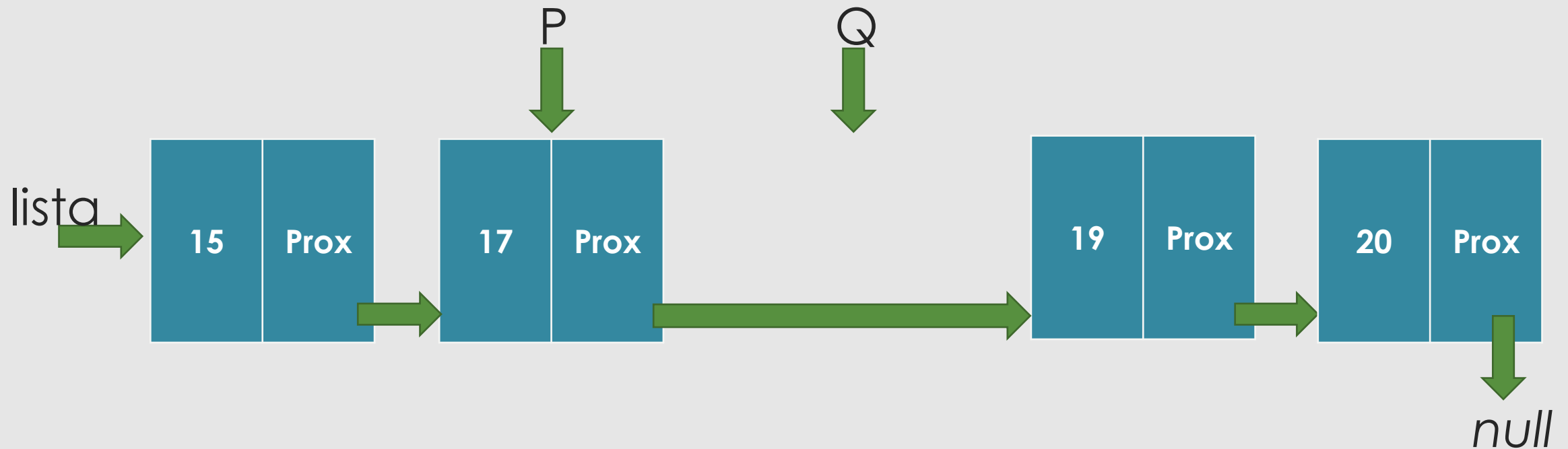
```
lista => p->prox;  
free(p);
```

# Lista Ligada Linear: Remoção no meio



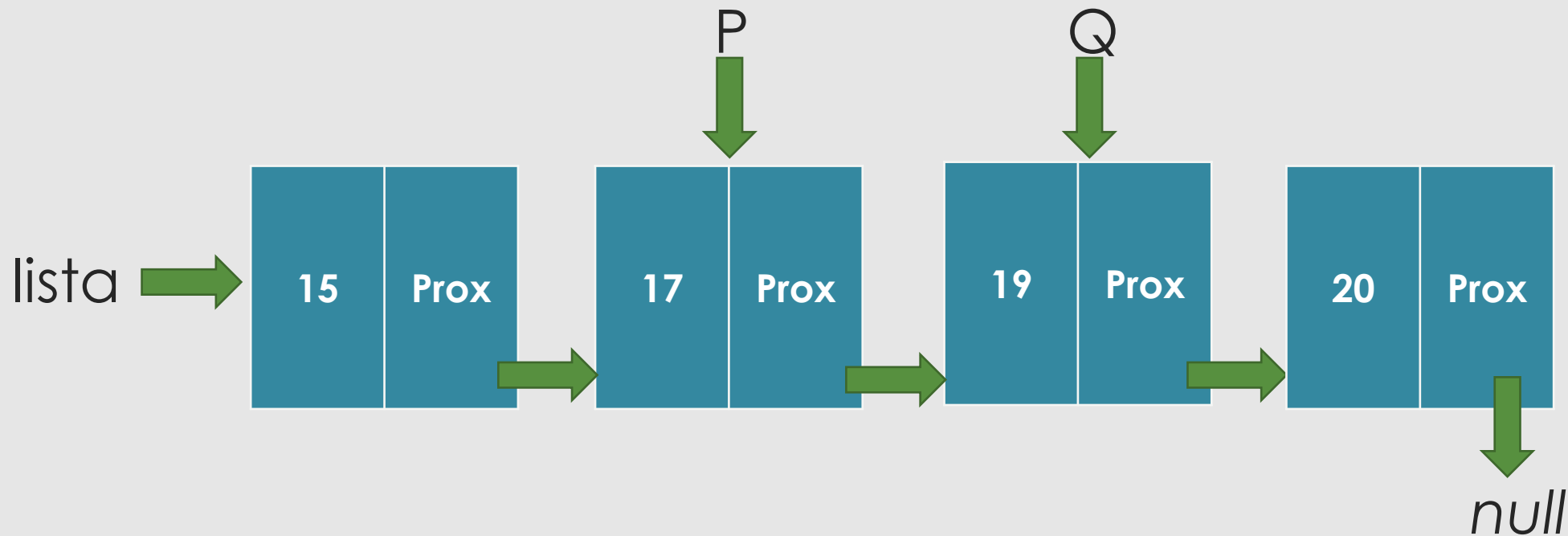
```
p->prox = q->prox;  
free(q);
```

# Lista Ligada Linear: Remoção no meio



```
p->prox = q->prox;  
free(q);
```

# ATENÇÃO



## LEIS ETERNAS

- ▶ Jamais perca a referência do ponteiro inicial que marca o início da lista.
- ▶ Jamais perca a referência do null que marca o final da lista.

The background is a complex, abstract composition of overlapping, semi-transparent geometric shapes, primarily triangles and quadrilaterals. The colors are vibrant and varied, including shades of blue, green, yellow, red, and grey. The shapes are layered in a way that creates a sense of depth and movement. In the center, there is a dark grey rectangular box containing the text.

OBRIGADO PELA AUDIÊNCIA