



# Estrutura de Dados

## Listas Simplesmente Encadeadas e Não Ordenadas

Prof. Thiago Caproni Tavares <sup>1</sup>    Paulo Muniz de Ávila <sup>2</sup>

<sup>1</sup>[thiago.tavares@ifsuldeminas.edu.br](mailto:thiago.tavares@ifsuldeminas.edu.br)

<sup>2</sup>[paulo.avila@ifsuldeminas.edu.br](mailto:paulo.avila@ifsuldeminas.edu.br)

## 1 Operações

- Inicialização
- Inserindo na Lista
  - Inserindo no Início da Lista
  - Inserindo no Fim da Lista
- Remoções

## 1 Operações

- Inicialização
- Inserindo na Lista
  - Inserindo no Início da Lista
  - Inserindo no Fim da Lista
- Remoções

## Operações

**1º Operação:**  
A lista está vazia

Lista
tam
inicio
0
NULL

**2º Operação:**  
Inserção do #9 no início da lista

Lista

tam	
1	
inicio	fin
500	500

500	prox
9	NULL



3º Operação:  
Inserção do #3 no início da lista

4º Operação:  
Inserção do #5 no fim da lista

Lista

índice	valor	prox
0	650	700
1	3	500
2	9	700
3	5	NULL

## 5º Operação: Inserção do #2 no fim da lista

Lista

tam	
<b>4</b>	
início	fim
<b>650</b>	<b>750</b>

```

graph LR
    N1[dado: 650, prox: 500] --> N2[dado: 500, prox: 700]
    N2 --> N3[dado: 700, prox: 750]
    N3 --> N4[dado: 750, prox: null]
    N4 --> end[||]
  
```

## 6º Operação: Remoção do #9

Lista	
tam	
3	
inicio	fin
650	750

## 7º Operação: Remoção do #3

Lista

tam	700
2	dado prox
inicio fim	5 750

750

dado prox	2 NULL
-----------	--------

|||

### 8º Operação: Remoção do #2

Lista

tam
1
inicio fim
700 700

700	dato	prox
5	NULL	→

## 1 Operações

### ■ Inicialização

### ■ Inserindo na Lista

- Inserindo no Início da Lista
- Inserindo no Fim da Lista

### ■ Remoções

# Inicialização

1º Operação:  
Inicializa a lista

```
void inicializar (Lista *lista){  
    lista->inicio = NULL;  
    lista->fim = NULL;  
    lista->tam = 0;  
}
```



# Inicialização

1º Operação:  
Inicializa a lista

```
void inicializar (Lista *lista){  
    lista->inicio = NULL;  
    lista->fim = NULL;  
    lista->tam = 0;  
}
```

Lista	
tam	lixo
inicio	fim
NULL	NULL

# Inicialização

1º Operação:  
Inicializa a lista

```
void inicializar (Lista *lista){  
    lista->inicio = NULL;  
    lista->fim = NULL;  
    lista->tam = 0;  
}
```

Lista	
tam	0
inicio	fim
NULL	NULL

## 1 Operações

- Inicialização

- Inserindo na Lista

  - Inserindo no Início da Lista

  - Inserindo no Fim da Lista

- Remoções

# Inserindo no Início da Lista

2º Operação:

Inserção do #9 no inicio da lista

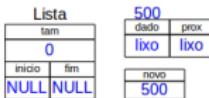
```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```



# Inserindo no Início da Lista

2º Operação:

Inserção do #9 no inicio da lista

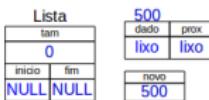
```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```



# Inserindo no Início da Lista

2º Operação:

Inserção do #9 no inicio da lista

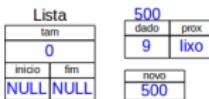
```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```



# Inserindo no Início da Lista

2º Operação:

Inserção do #9 no inicio da lista

```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```



# Inserindo no Início da Lista

2º Operação:

Inserção do #9 no inicio da lista

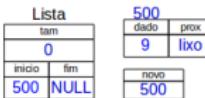
```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```



# Inserindo no Início da Lista

2º Operação:

Inserção do #9 no inicio da lista

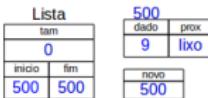
```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```



# Inserindo no Início da Lista

2º Operação:

Inserção do #9 no inicio da lista

```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```



# Inserindo no Início da Lista

2º Operação:

Inserção do #9 no início da lista

```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```



# Inserindo no Início da Lista

2º Operação:

Inserção do #9 no inicio da lista

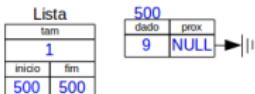
```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```



# Inserindo no Início da Lista

3º Operação:

Inserção do #3 no inicio da lista

```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```



# Inserindo no Início da Lista

3º Operação:

Inserção do #3 no inicio da lista

```
int inserirInicio(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
  
    if(novo == NULL)  
        return 0;  
  
    novo->dado = dado;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        novo->prox = lista->inicio;  
        lista->inicio = novo;  
    }  
  
    lista->tam++;  
    return 1;  
}
```



# Inserindo no Início da Lista

3º Operação:

Inserção do #3 no inicio da lista

```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```



# Inserindo no Início da Lista

3º Operação:

Inserção do #3 no início da lista

```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```



# Inserindo no Início da Lista

3º Operação:

Inserção do #3 no inicio da lista

```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```



# Inserindo no Início da Lista

3º Operação:

Inserção do #3 no inicio da lista

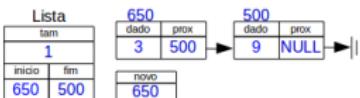
```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```



# Inserindo no Início da Lista

3º Operação:

Inserção do #3 no início da lista

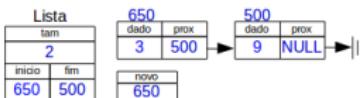
```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```



# Inserindo no Início da Lista

3º Operação:

Inserção do #3 no início da lista

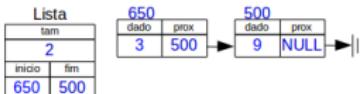
```
int inserirInicio(Lista *lista, int dado)
{
    cel *novo = malloc(sizeof(cel));

    if(novo == NULL)
        return 0;

    novo->dado = dado;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        novo->prox = lista->inicio;
        lista->inicio = novo;
    }

    lista->tam++;
    return 1;
}
```

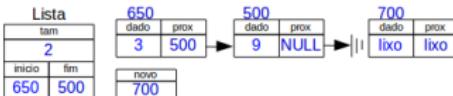


# Inserindo no Fim da Lista

4º Operação:

Inserção do #5 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```

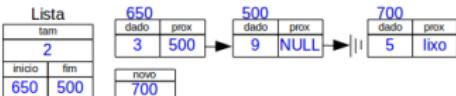


# Inserindo no Fim da Lista

4º Operação:

Inserção do #5 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```



## Inserindo no Fim da Lista

#### 4º Operação:

### Inserção do #5 no fim da lista

```
int inserirFim(Lista *lista, int dado){
    cel *novo = malloc(sizeof(cel));
    novo->dado = dado;

    if(novo == NULL)
        return 0;

    if(lista->inicio == NULL){
        lista->inicio = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }else{
        lista->fim->prox = novo;
        lista->fim = novo;
        lista->fim->prox = NULL;
    }

    lista->tam++;
    return 1;
}
```

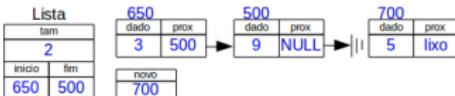


# Inserindo no Fim da Lista

4º Operação:

Inserção do #5 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```

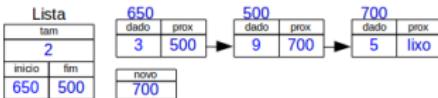


# Inserindo no Fim da Lista

4º Operação:

Inserção do #5 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```

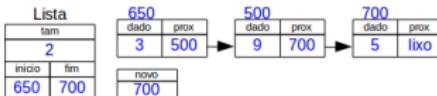


# Inserindo no Fim da Lista

4º Operação:

Inserção do #5 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```

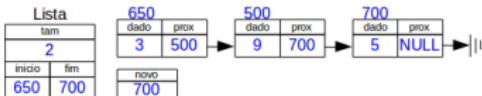


# Inserindo no Fim da Lista

4º Operação:

Inserção do #5 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```

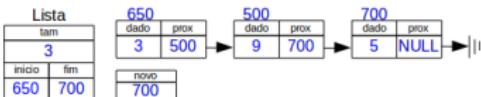


# Inserindo no Fim da Lista

4º Operação:

Inserção do #5 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```

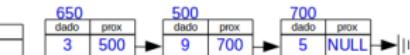


# Inserindo no Fim da Lista

4º Operação:

Inserção do #5 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```

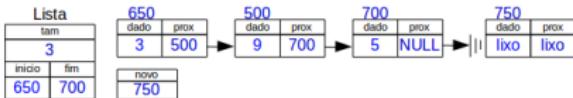


# Inserindo no Fim da Lista

5º Operação:

Inserção do #2 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```

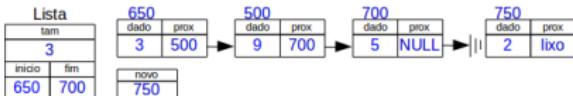


# Inserindo no Fim da Lista

5º Operação:

Inserção do #2 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```

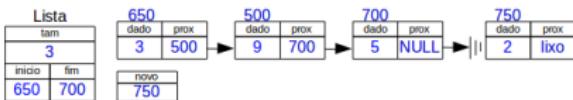


# Inserindo no Fim da Lista

5º Operação:

Inserção do #2 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```

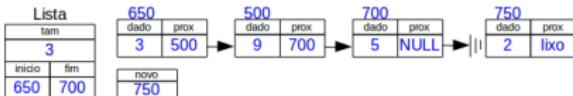


# Inserindo no Fim da Lista

5º Operação:

Inserção do #2 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```

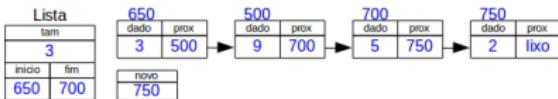


# Inserindo no Fim da Lista

5º Operação:

Inserção do #2 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```

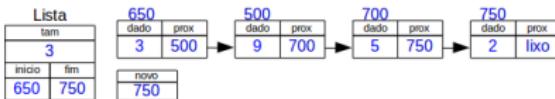


# Inserindo no Fim da Lista

5º Operação:

Inserção do #2 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```

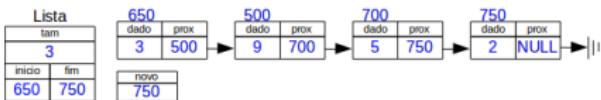


# Inserindo no Fim da Lista

5º Operação:

Inserção do #2 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```

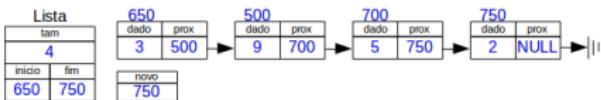


# Inserindo no Fim da Lista

5º Operação:

Inserção do #2 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```



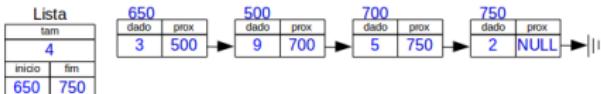
# Inserindo no Fim da Lista

5º Operação:

Inserção do #2 no fim da lista

```
int inserirFim(Lista *lista, int dado){  
    cel *novo = malloc(sizeof(cel));  
    novo->dado = dado;  
  
    if(novo == NULL)  
        return 0;  
  
    if(lista->inicio == NULL){  
        lista->inicio = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }else{  
        lista->fim->prox = novo;  
        lista->fim = novo;  
        lista->fim->prox = NULL;  
    }  
  
    lista->tam++;  
    return 1;  
}
```

Lista	
tam	
inicio	fim
4	
650	750



## 1 Operações

- Inicialização
- Inserindo na Lista
  - Inserindo no Início da Lista
  - Inserindo no Fim da Lista
- Remoções

## Removendo Elementos

6º Operação:  
Remoção do #9

```

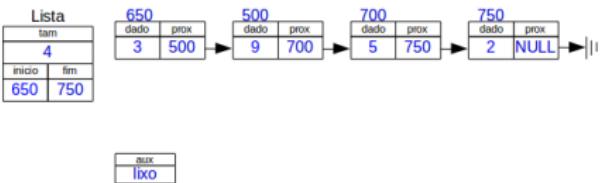
    cel *aux;
    cel *ant;
    int qtdDel = 0;

    aux = lista->inicio;
    ant = NULL;

    while(aux != NULL){
        if(aux->dado == dado){
            qtdDel++;
            lista->tam--;
            if(aux == lista->inicio){
                lista->inicio = aux->prox;
                free(aux);
                aux = lista->inicio;
            }else if(aux == lista->fim){
                ant->prox = NULL;
                lista->fim = ant;
                free(aux);
                aux = NULL;
            }else{
                ant->prox = aux->prox;
                free(aux);
                aux = ant->prox;
            }
        }else{
            ant = aux;
            aux = aux->prox;
        }
    }

    return qtdDel;
}

```



# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
4	
inicio fim	

dado	prox
3	500
650	750

dado	prox
9	700
5	750

dado	prox
5	750
2	NULL

dado	prox
750	



# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
4	
inicio fim	

650	750
3	500

650	500
9	700

500	700
5	750

700	750
2	NULL

750	NULL

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel
lixo	lixo	0

aux	ant	qtdDel

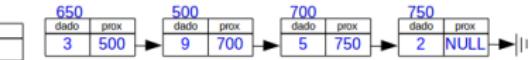

<tbl\_r cells="3" ix="2

# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista
tam
4
inicio fim
650 750



aux	ant	qtdDel
650	lixo	0

# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista
tam
4
inicio fim
650 750



aux	ant	qtdDel
650	NULL	0

# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
4	
início fim	

650	750
3	500

650	750
9	700

500	750
5	750

700	750
2	NULL

750	750
2	NULL

aux	ant	qtdDel
650	NULL	0

aux	ant	qtdDel
650	NULL	0

# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista
tam
4
inicio fim
650 750



aux	ant	qtdDel
650	NULL	0

# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }else{  
            ant = aux;  
            aux = aux->prox;  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
4	
início fim	

dados	prox
3	500
500	700
700	750
750	NULL

650 750

650 750

650 750

650 750

aux	ant	qtdDel
650	650	0

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

650 750

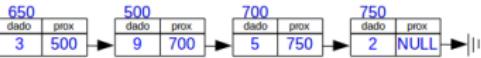
650 7

# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista
tam
4
inicio fim
650 750



aux	ant	qtdDel
500	650	0

# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista
tam
4
inicio fim
650 750



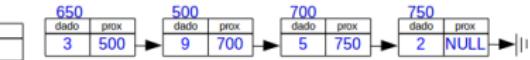
aux
500
ant 650
qtdDel 0

# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista  
tam  
4  
inicio fim  
650 750



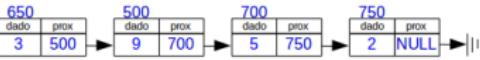
aux 500  
ant 650  
qtdDel 0

# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista
tam
4
inicio fim
650 750



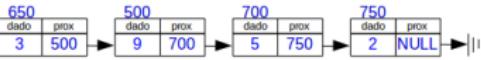
aux	ant	qtdDel
500	650	1

# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            [listा->tam--;  
             if(aux == lista->inicio){  
                 lista->inicio = aux->prox;  
                 free(aux);  
                 aux = lista->inicio;  
             }else if(aux == lista->fim){  
                 ant->prox = NULL;  
                 lista->fim = ant;  
                 free(aux);  
                 aux = NULL;  
             }else{  
                 ant->prox = aux->prox;  
                 free(aux);  
                 aux = ant->prox;  
             }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista
tam
3
inicio fim
650 750



aux	ant	qtdDel
500	650	1

# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista
tam
3
inicio fim
650 750



aux	ant	qtdDel
500	650	1

## Removendo Elementos

**6º Operação:**  
Remoção do #9

```

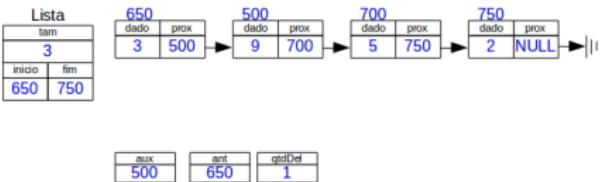
int remover(Lista *lista, int dado){
    cel *aux;
    cel *ant;
    int qtdDel = 0;

    aux = lista->inicio;
    ant = NULL;

    while(aux != NULL){
        if(aux->dado == dado){
            qtdDel++;
            lista->tam--;
            if(aux == lista->inicio){
                lista->inicio = aux->prox;
                free(aux);
                aux = lista->inicio;
            }else if(aux == lista->fim){
                ant->prox = NULL;
                lista->fim = ant;
                free(aux);
                aux = NULL;
            }else{
                ant->prox = aux->prox;
                free(aux);
                aux = ant->prox;
            }
        }else{
            ant = aux;
            aux = aux->prox;
        }
    }

    return qtdDel;
}

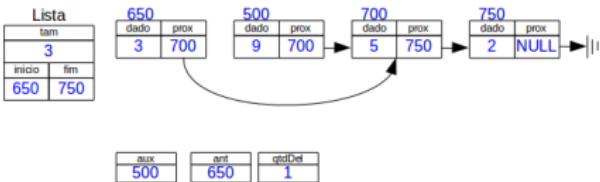
```



# Removendo Elementos

6º Operação:  
Remoção do #9

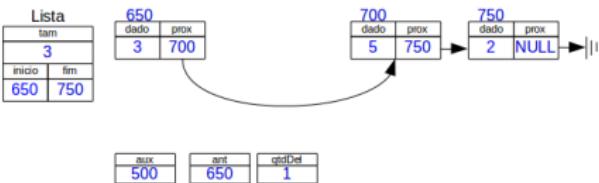
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

6º Operação:  
Remoção do #9

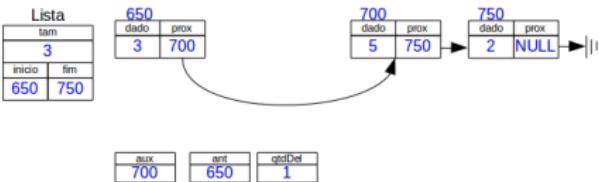
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
3	
inicio fim	

650	750
aux	ant

650	700
dado prox	

700	750
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

750	NULL
dado prox	

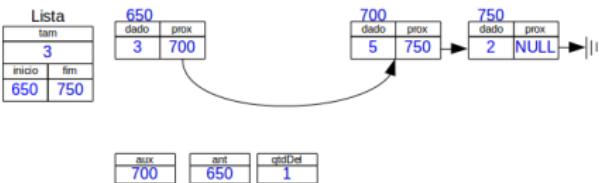
750	NULL
dado prox	

750	NULL
dado prox	

# Removendo Elementos

6º Operação:  
Remoção do #9

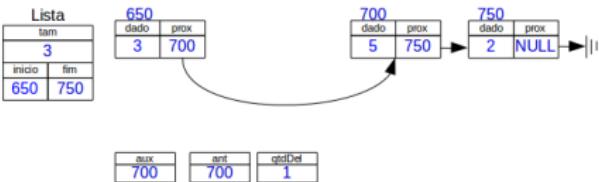
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

6º Operação:  
Remoção do #9

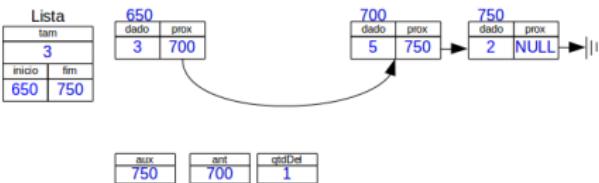
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

6º Operação:  
Remoção do #9

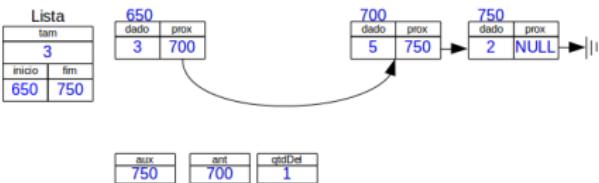
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

6º Operação:  
Remoção do #9

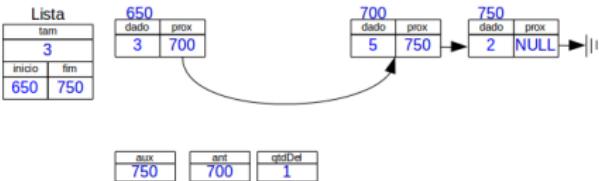
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

6º Operação:  
Remoção do #9

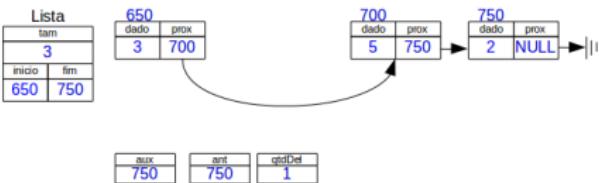
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

6º Operação:  
Remoção do #9

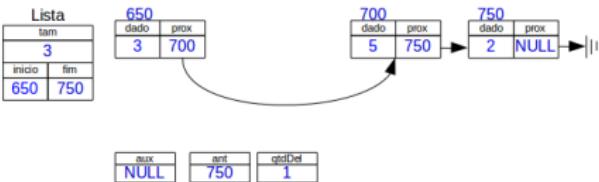
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

6º Operação:  
Remoção do #9

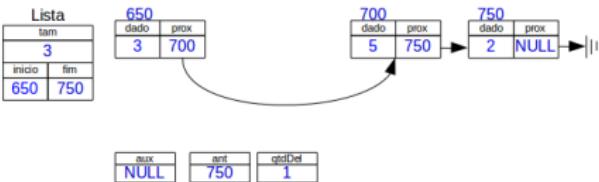
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

6º Operação:  
Remoção do #9

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

6º Operação:  
Remoção do #9

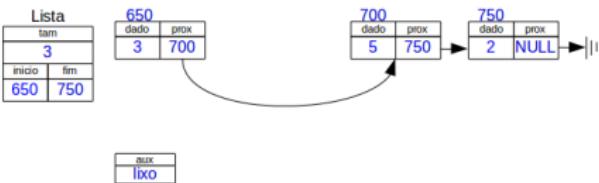
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

7º Operação:  
Remoção do #3

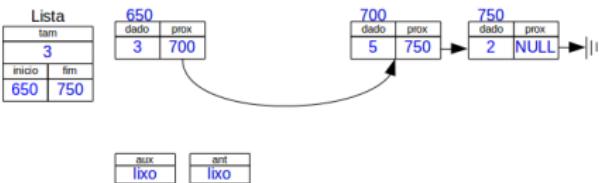
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

7º Operação:  
Remoção do #3

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

7º Operação:  
Remoção do #3

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista
tam 3
início fim 3 700
650 750

650
dado prox
3 700

aux
lixo
ant

ant
lixo
qtdDel

qtdDel
0

700
dado prox
5 750

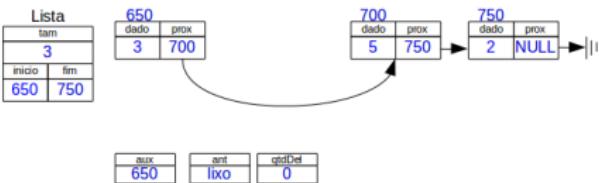
750
dado prox
2 NULL

--

# Removendo Elementos

7º Operação:  
Remoção do #3

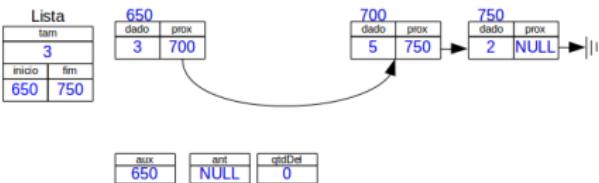
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

7º Operação:  
Remoção do #3

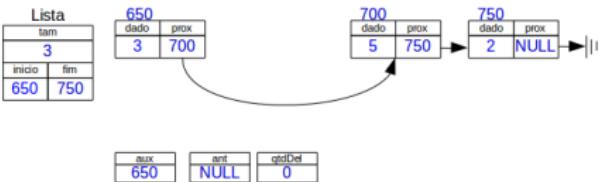
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

7º Operação:  
Remoção do #3

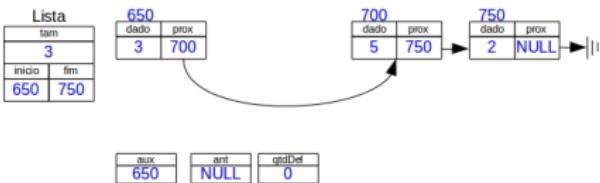
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

7º Operação:  
Remoção do #3

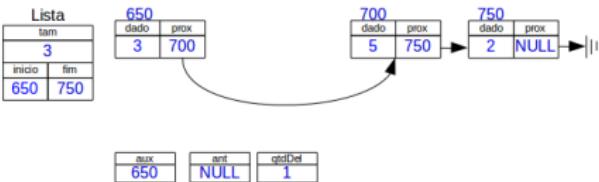
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

7º Operação:  
Remoção do #3

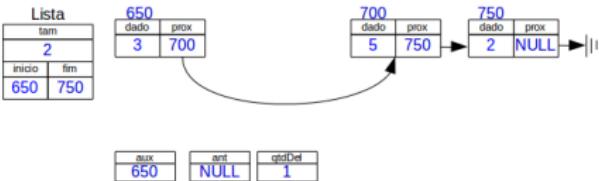
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

7º Operação:  
Remoção do #3

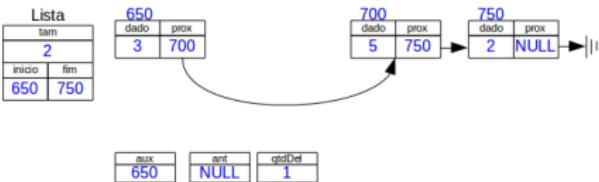
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            [list->inicio--];  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

7º Operação:  
Remoção do #3

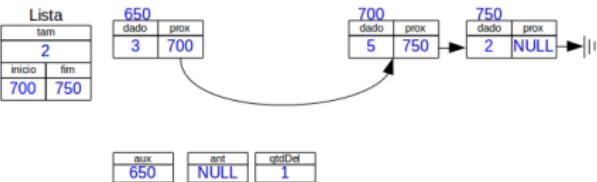
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

7º Operação:  
Remoção do #3

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

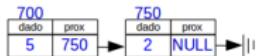
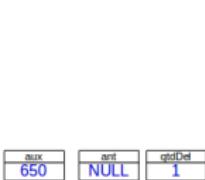


# Removendo Elementos

7º Operação:  
Remoção do #3

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

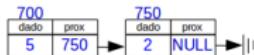
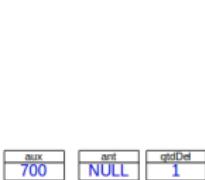
Lista	
tam	
2	
inicio	fim
700	750



# Removendo Elementos

7º Operação:  
Remoção do #3

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



## Removendo Elementos

7º Operação:  
Remoção do #3

```

int remover(Lista *lista, int dado){
    cel *aux;
    cel *ant;
    int qtdDel = 0;

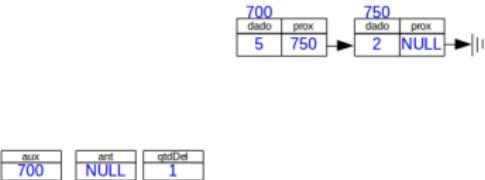
    aux = lista->inicio;
    ant = NULL;

    while(aux != NULL){
        if(aux->dado == dado){
            qtdDel++;
            lista->tam--;
            if(aux == lista->inicio){
                lista->inicio = aux->prox;
                free(aux);
                aux = lista->inicio;
            }else if(aux == lista->fim){
                ant->prox = NULL;
                lista->fim = ant;
                free(aux);
                aux = NULL;
            }else{
                ant->prox = aux->prox;
                free(aux);
                aux = ant->prox;
            }
        }else{
            ant = aux;
            aux = aux->prox;
        }
    }

    return qtdDel;
}

```

Lista	
tam	
inicio	fin
2	
700	750

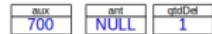
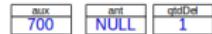
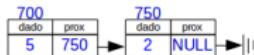
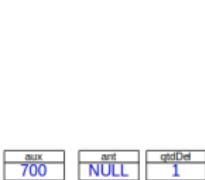


# Removendo Elementos

7º Operação:  
Remoção do #3

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
2	
inicio	fim
700	750

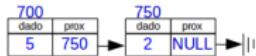
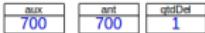


# Removendo Elementos

7º Operação:  
Remoção do #3

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }else{  
            ant = aux;  
            aux = aux->prox;  
        }  
    }  
  
    return qtdDel;  
}
```

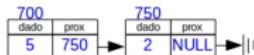
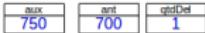
Lista	
tam	
2	
inicio	fim
700	750



# Removendo Elementos

7º Operação:  
Remoção do #3

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

7º Operação:  
Remoção do #3

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
2	
inicio	fim
700	750

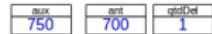
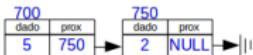


# Removendo Elementos

7º Operação:  
Remoção do #3

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

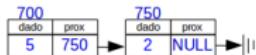
Lista	
tam	
2	
inicio	fim
700	750



# Removendo Elementos

7º Operação:  
Remoção do #3

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }else{  
            ant = aux;  
            aux = aux->prox;  
        }  
    }  
  
    return qtdDel;  
}
```



## Removendo Elementos

## 7º Operação: Remoção do #3

```

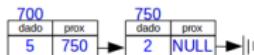
int remover(Lista *lista, int dado){
    cel *aux;
    cel *ant;
    int qtdDel = 0;

    aux = lista->inicio;
    ant = NULL;

    while(aux != NULL){
        if(aux->dado == dado){
            qtdDel++;
            lista->tam--;
            if(aux == lista->inicio){
                lista->inicio = aux->prox;
                free(aux);
                aux = lista->inicio;
            }else if(aux == lista->fim){
                ant->prox = NULL;
                lista->fim = ant;
                free(aux);
                aux = NULL;
            }else{
                ant->prox = aux->prox;
                free(aux);
                aux = ant->prox;
            }
        }else{
            ant = aux;
            aux = aux->prox;
        }
    }

    return qtdDel;
}

```



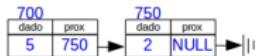
# Removendo Elementos

7º Operação:  
Remoção do #3

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
2	
inicio	fim
700	750

aux	ant	qtdDel
NULL	750	1

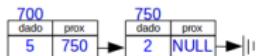


# Removendo Elementos

7º Operação:  
Remoção do #3

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
2	
inicio fim	
700	750

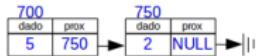


# Removendo Elementos

8º Operação:  
Remoção do #2

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

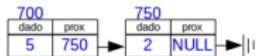
Lista	
tam	
2	
inicio	fim
700	750



# Removendo Elementos

8º Operação:  
Remoção do #2

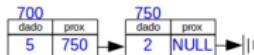
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

8º Operação:  
Remoção do #2

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



## Removendo Elementos

## 8º Operação: Remoção do #2

```

    int remover(Lista *lista, int dado){
        cel = aux;
        cel = ant;
        int qtdDel = 0;

        aux = lista->inicio;
        ant = NULL;

        while(aux != NULL){
            if(aux->dado == dado){
                qtdDel++;
                lista->stam--;
                if(aux == lista->inicio){
                    lista->inicio = aux->prox;
                    free(aux);
                    aux = lista->inicio;
                }else if(aux == lista->fim){
                    ant->prox = NULL;
                    lista->fim = ant;
                    free(aux);
                    aux = NULL;
                }else{
                    ant->prox = aux->prox;
                    free(aux);
                    aux = ant->prox;
                }
            }else{
                ant = aux;
                aux = aux->prox;
            }
        }

        return qtdDel;
    }
}

```

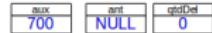
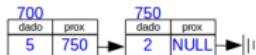
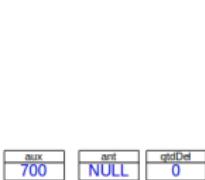


# Removendo Elementos

8º Operação:  
Remoção do #2

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

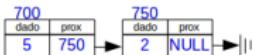
Lista	
tam	
2	
inicio	fim
700	750



# Removendo Elementos

8º Operação:  
Remoção do #2

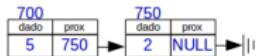
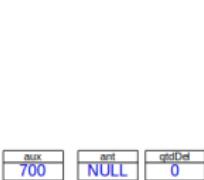
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

8º Operação:  
Remoção do #2

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

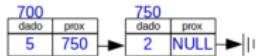


# Removendo Elementos

8º Operação:  
Remoção do #2

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }else{  
            ant = aux;  
            aux = aux->prox;  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
2	
inicio	fim
700	750



## Removendo Elementos

## 8º Operação: Remoção do #2

```

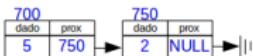
int remover(Lista *lista, int dado){
    cel *aux;
    cel *ant;
    int qtdDel = 0;

    aux = lista->inicio;
    ant = NULL;

    while(aux != NULL){
        if(aux->dado == dado){
            qtdDel++;
            lista->tam--;
            if(aux == lista->inicio){
                lista->inicio = aux->prox;
                free(aux);
                aux = lista->inicio;
            }else if(aux == lista->fim){
                ant->prox = NULL;
                lista->fim = ant;
                free(aux);
                aux = NULL;
            }else{
                ant->prox = aux->prox;
                free(aux);
                aux = ant->prox;
            }
        }else{
            ant = aux;
            aux = aux->prox;
        }
    }

    return qtdDel;
}

```

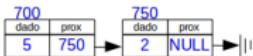


# Removendo Elementos

8º Operação:  
Remoção do #2

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

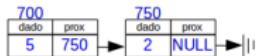
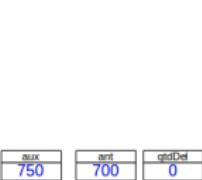
Lista	
tam	
2	
inicio	fim
700	750



# Removendo Elementos

8º Operação:  
Remoção do #2

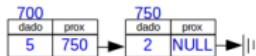
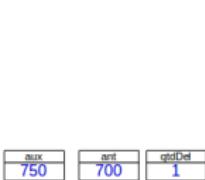
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

8º Operação:  
Remoção do #2

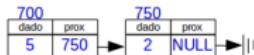
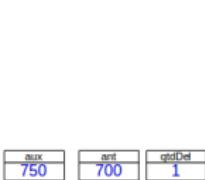
```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```



# Removendo Elementos

8º Operação:  
Remoção do #2

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

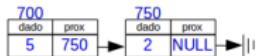


# Removendo Elementos

8º Operação:  
Remoção do #2

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
1	
inicio	fim
700	750



## Removendo Elementos

8º Operação:  
Remoção do #2

```

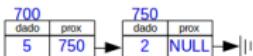
int remover(Lista *lista, int dado){
    cel *aux;
    cel *ant;
    int qtdDel = 0;

    aux = lista->inicio;
    ant = NULL;

    while(aux != NULL){
        if(aux->dado == dado){
            qtdDel++;
            lista->tam--;
            if(aux == lista->inicio){
                lista->inicio = aux->prox;
                free(aux);
                aux = lista->inicio;
            } else if(aux == lista->fim){
                ant->prox = NULL;
                lista->fim = ant;
                free(aux);
                aux = NULL;
            } else{
                ant->prox = aux->prox;
                free(aux);
                aux = ant->prox;
            }
        } else{
            ant = aux;
            aux = aux->prox;
        }
    }

    return qtdDel;
}

```

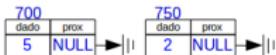


# Removendo Elementos

8º Operação:  
Remoção do #2

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
1	
inicio	fim
700	750

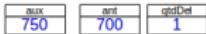
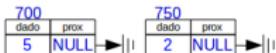


# Removendo Elementos

8º Operação:  
Remoção do #2

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
1	
inicio	fim
700	700



# Removendo Elementos

8º Operação:  
Remoção do #2

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
inicio	fim
1	
700	700

dado	prox
5	NULL



# Removendo Elementos

8º Operação:  
Remoção do #2

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
inicio	fim
1	
700	700

dado	prox
5	NULL

aux	ant	qtdDel
NULL	700	1

# Removendo Elementos

8º Operação:  
Remoção do #2

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
dados	prox
1	
700	700

700	dado	prox
	5	NULL

aux	ant	qtdDel
NULL	700	1

# Removendo Elementos

8º Operação:  
Remoção do #2

```
int remover(Lista *lista, int dado){  
    cel *aux;  
    cel *ant;  
    int qtdDel = 0;  
  
    aux = lista->inicio;  
    ant = NULL;  
  
    while(aux != NULL){  
        if(aux->dado == dado){  
            qtdDel++;  
            lista->tam--;  
            if(aux == lista->inicio){  
                lista->inicio = aux->prox;  
                free(aux);  
                aux = lista->inicio;  
            }else if(aux == lista->fim){  
                ant->prox = NULL;  
                lista->fim = ant;  
                free(aux);  
                aux = NULL;  
            }else{  
                ant->prox = aux->prox;  
                free(aux);  
                aux = ant->prox;  
            }  
        }  
    }  
  
    return qtdDel;  
}
```

Lista	
tam	
inicio	fim
1	
700	700



Obrigado pela atenção!!!  
[thiago.tavares@ifsuldeminas.edu.br](mailto:thiago.tavares@ifsuldeminas.edu.br)



-  ASCENCIO, A.; CAMPOS, E. de. *Fundamentos da programação de computadores: algoritmos, Pascal, C/C++ e Java*. Pearson Prentice Hall, 2008. ISBN 9788576051480. Disponível em: <<https://books.google.com.br/books?id=p-mTPgAACAAJ>>.
-  C: A Reference Manual. Pearson Education, 2007. ISBN 9788131714409. Disponível em: <<https://books.google.com.br/books?id=Wt2NEypdGNIC>>.
-  DAMAS, L. *LINGUAGEM C*. LTC. ISBN 9788521615194. Disponível em: <<https://books.google.com.br/books?id=22-vPgAACAAJ>>.
-  FEOFILOFF, P. *Algoritmos Em Linguagem C*. CAMPUS - RJ, 2009. ISBN 9788535232493. Disponível em: <<http://books.google.com.br/books?id=LfUQai78VQgC>>.
-  KERNIGHAN, B.; RITCHIE, D. *C: a linguagem de programação padrão ANSI*. Campus, 1989. ISBN 9788570015860. Disponível em: <<https://books.google.com.br/books?id=aVWrQwAACAAJ>>.

-  LOPES, A.; GARCIA, G. *Introdução à programação: 500 algoritmos resolvidos*. Campus, 2002. ISBN 9788535210194. Disponível em: <<https://books.google.com.br/books?id=Rd-LPgAACAAJ>>.
-  MIZRAHI, V. *Treinamento em linguagem C*. Pearson Prentice Hall, 2008. ISBN 9788576051916. Disponível em: <<https://books.google.com.br/books?id=7xt7PgAACAAJ>>.
-  SCHILDT, H.; MAYER, R. *C completo e total*. Makron, 1997. ISBN 9788534605953. Disponível em: <<https://books.google.com.br/books?id=PbI0AAAACAAJ>>.