

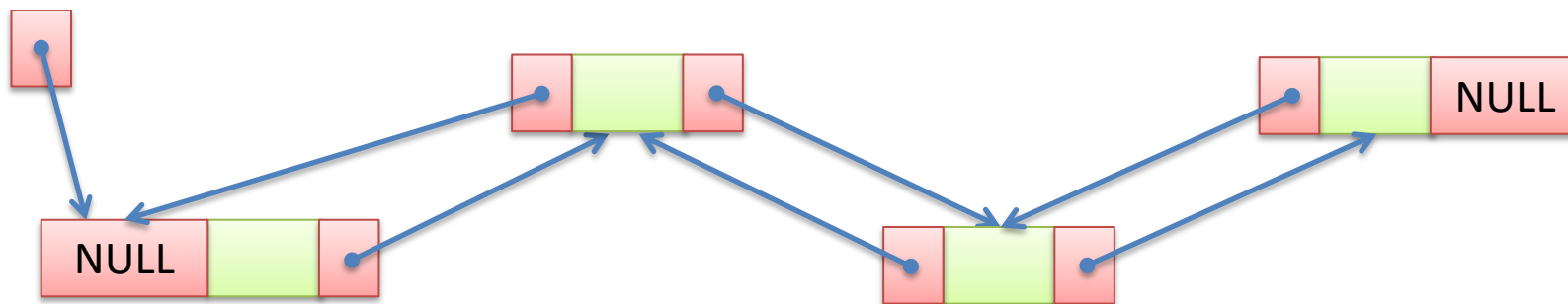
Estrutura de Dados e Algoritmos

Listas Duplamente Encadeadas

- Uma lista duplamente encadeada é uma estrutura composta por nós, onde cada nó tem uma informação e dois ponteiros: um para o próximo elemento da lista, outro para o elemento anterior;
- Conhecido o primeiro elemento da lista (cabeça) ou o último (calda), é possível acessar os demais elementos;

- Diferente das listas encadeadas, as listas duplamente encadeadas reduzem a complexidade da remoção de um elemento do final da lista de $O(n)$ para $O(1)$;
- Os nós de uma lista duplamente encadeada apontam tanto para o sucessor (campo prox) quanto para o antecessor (campo ant) do elemento;

Lista



```
struct TNo
```

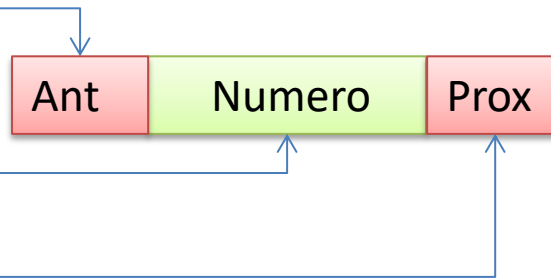
```
{
```

```
TNo *Ant;
```

```
int Numero;
```

```
TNo *Prox;
```

```
};
```

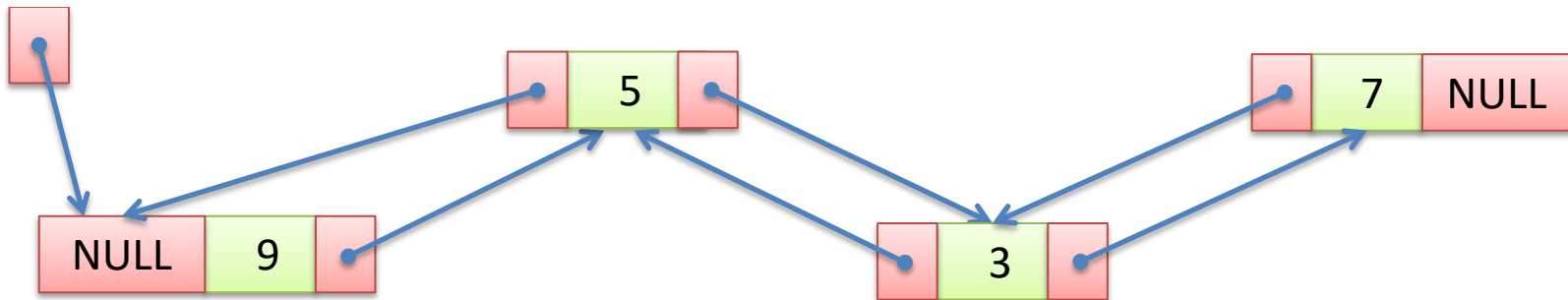


```
bool Vazia(TNo *pLista)
{
    if(pLista == NULL)
        return true;
    else
        return false;
}
```

Lista

NULL

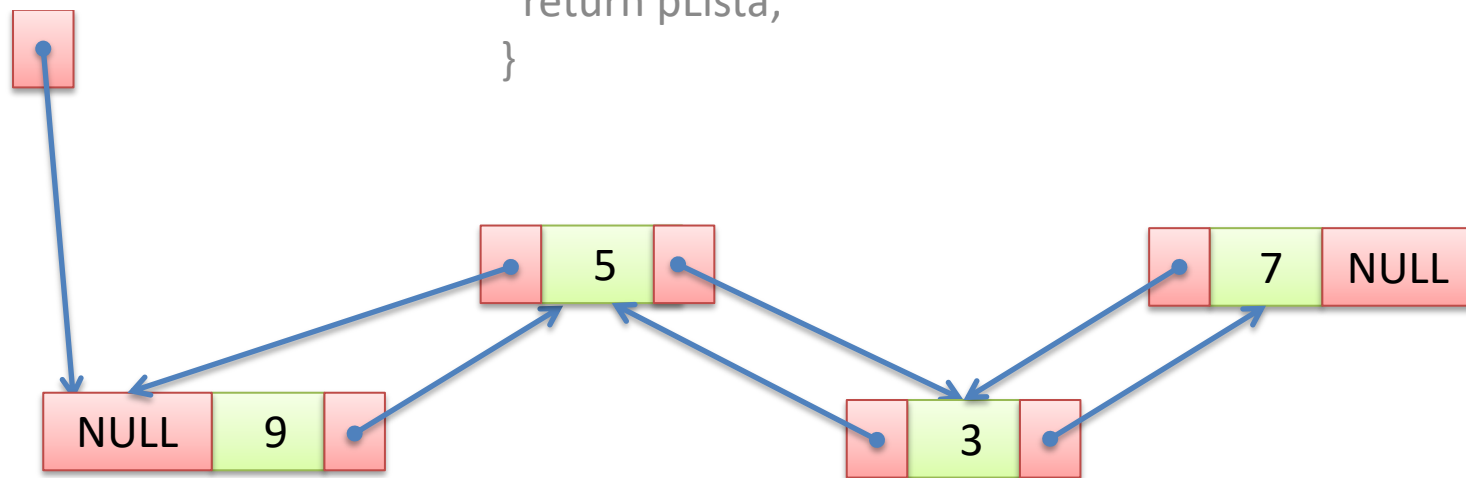
Lista



Lista Duplamente Encadeada Inclusão na Cabeça

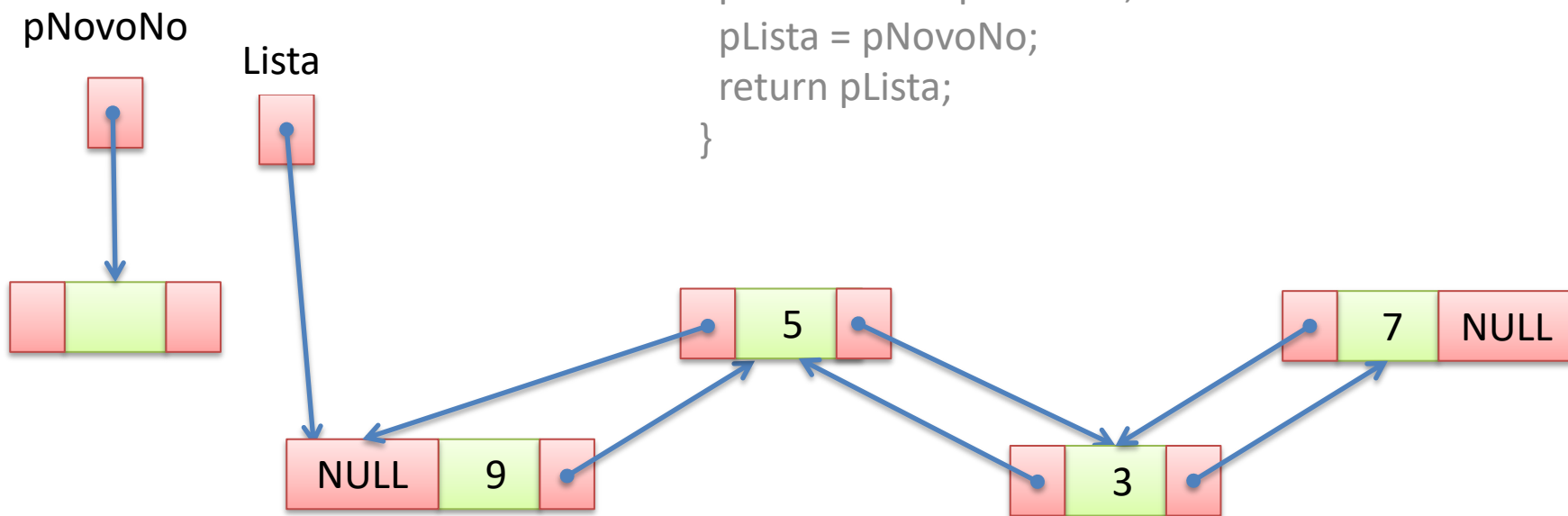
```
TNo *IncluiCabeça(TNo *pLista, int pValor)
{
    TNo *pNovoNo;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Ant = NULL;
    pNovoNo->Prox = pLista;
    pLista->Ant = pNovoNo;
    pLista = pNovoNo;
    return pLista;
}
```

Lista



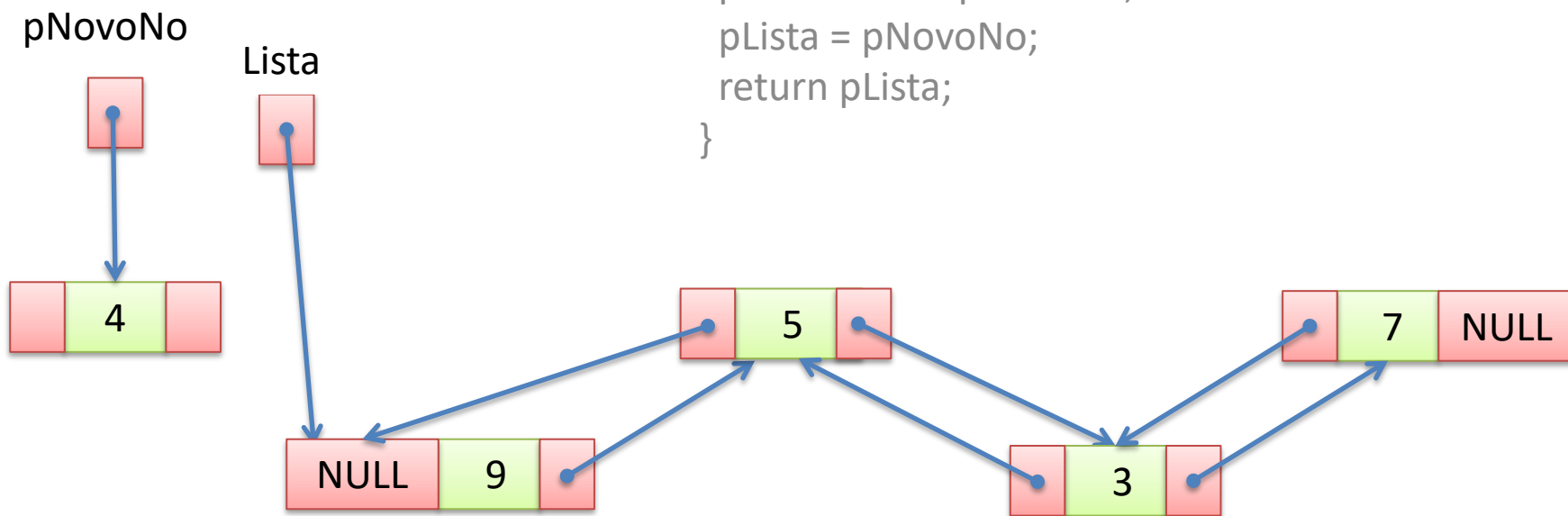
Lista Duplamente Encadeada Inclusão na Cabeça

```
TNo *IncluiCabeça(TNo *pLista, int pValor)
{
    TNo *pNovoNo;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Ant = NULL;
    pNovoNo->Prox = pLista;
    pLista->Ant = pNovoNo;
    pLista = pNovoNo;
    return pLista;
}
```



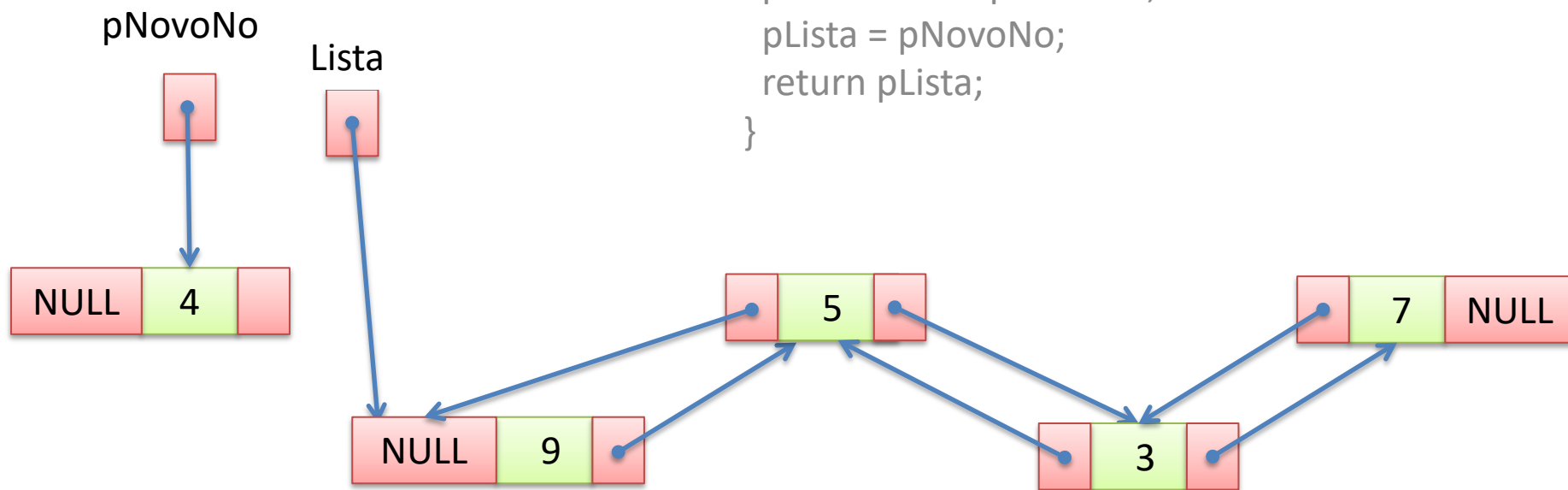
Lista Duplamente Encadeada Inclusão na Cabeça

```
TNo *IncluiCabeça(TNo *pLista, int pValor)
{
    TNo *pNovoNo;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Ant = NULL;
    pNovoNo->Prox = pLista;
    pLista->Ant = pNovoNo;
    pLista = pNovoNo;
    return pLista;
}
```



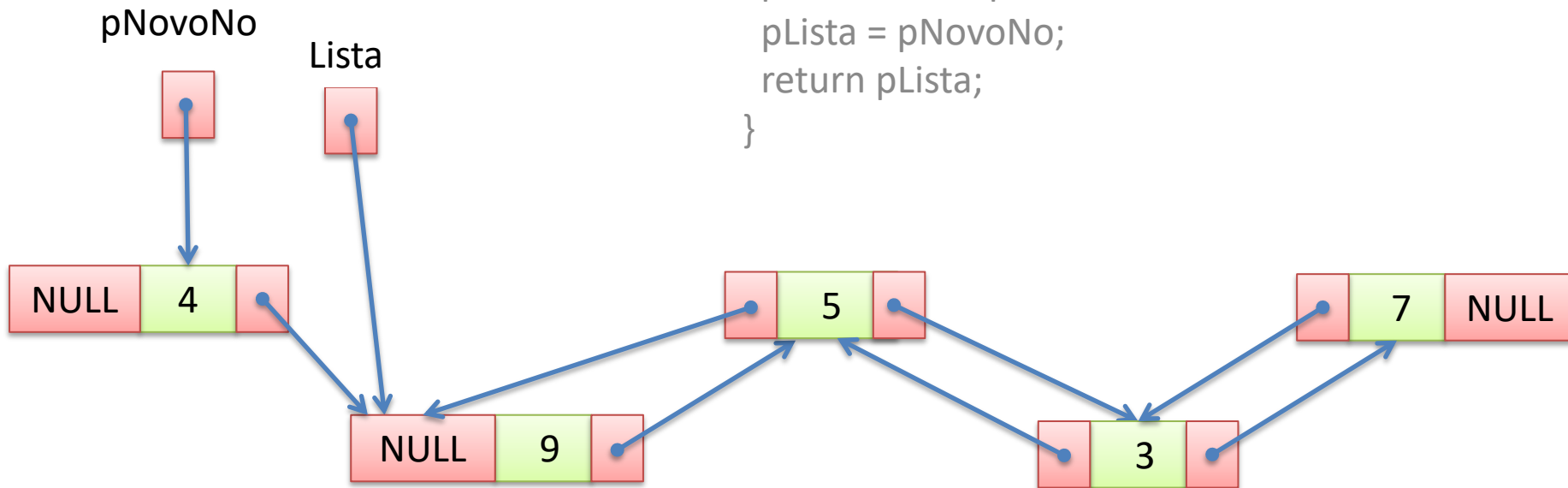
Lista Duplamente Encadeada Inclusão na Cabeça

```
TNo *IncluiCabeça(TNo *pLista, int pValor)
{
    TNo *pNovoNo;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Ant = NULL;
    pNovoNo->Prox = pLista;
    pLista->Ant = pNovoNo;
    pLista = pNovoNo;
    return pLista;
}
```



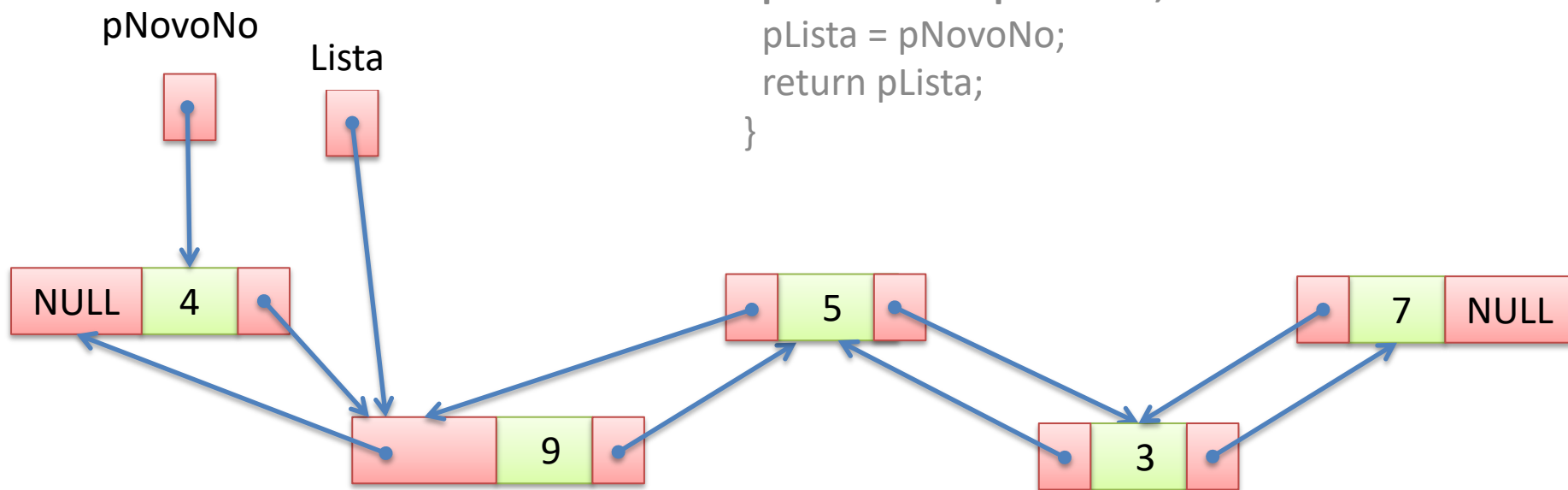
Lista Duplamente Encadeada Inclusão na Cabeça

```
TNo *IncluiCabeça(TNo *pLista, int pValor)
{
    TNo *pNovoNo;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Ant = NULL;
    pNovoNo->Prox = pLista;
    pLista->Ant = pNovoNo;
    pLista = pNovoNo;
    return pLista;
}
```



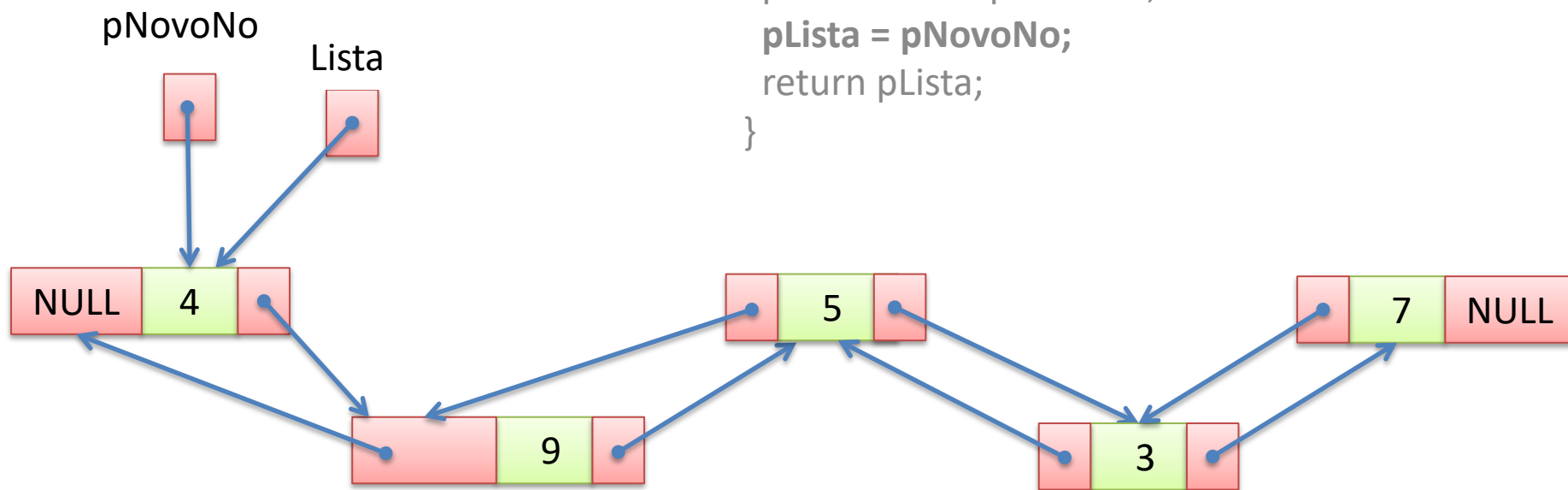
Lista Duplamente Encadeada Inclusão na Cabeça

```
TNo *IncluiCabeça(TNo *pLista, int pValor)
{
    TNo *pNovoNo;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Ant = NULL;
    pNovoNo->Prox = pLista;
    pLista->Ant = pNovoNo;
    pLista = pNovoNo;
    return pLista;
}
```



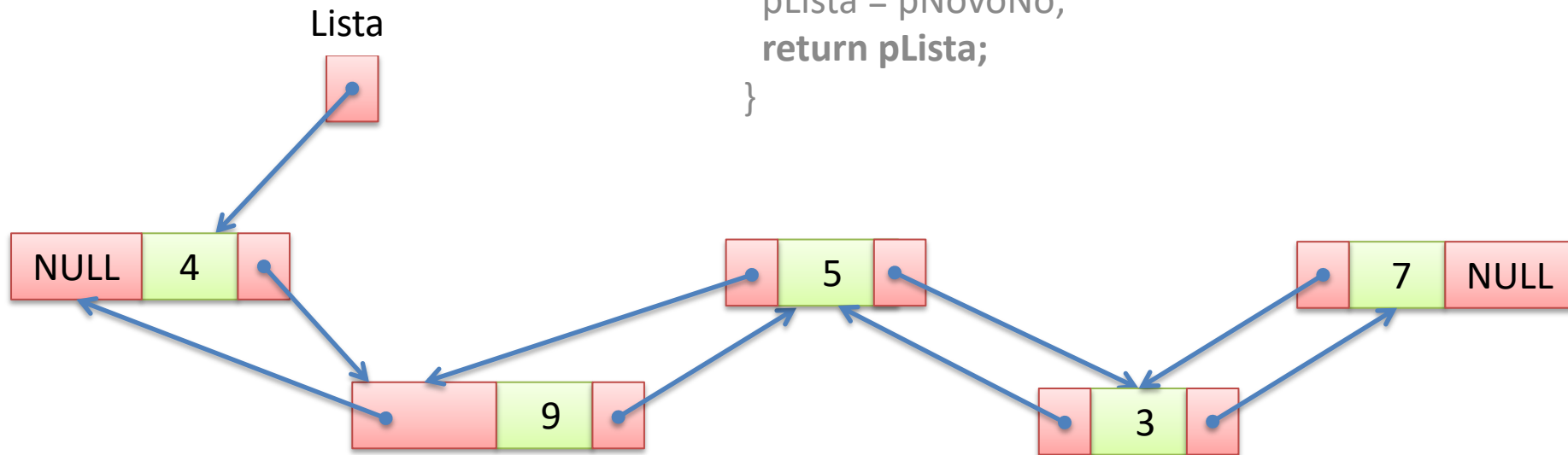
Lista Duplamente Encadeada Inclusão na Cabeça

```
TNo *IncluiCabeça(TNo *pLista, int pValor)
{
    TNo *pNovoNo;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Ant = NULL;
    pNovoNo->Prox = pLista;
    pLista->Ant = pNovoNo;
    pLista = pNovoNo;
    return pLista;
}
```



Lista Duplamente Encadeada Inclusão na Cabeça

```
TNo *IncluiCabeça(TNo *pLista, int pValor)
{
    TNo *pNovoNo;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Ant = NULL;
    pNovoNo->Prox = pLista;
    pLista->Ant = pNovoNo;
    pLista = pNovoNo;
    return pLista;
}
```

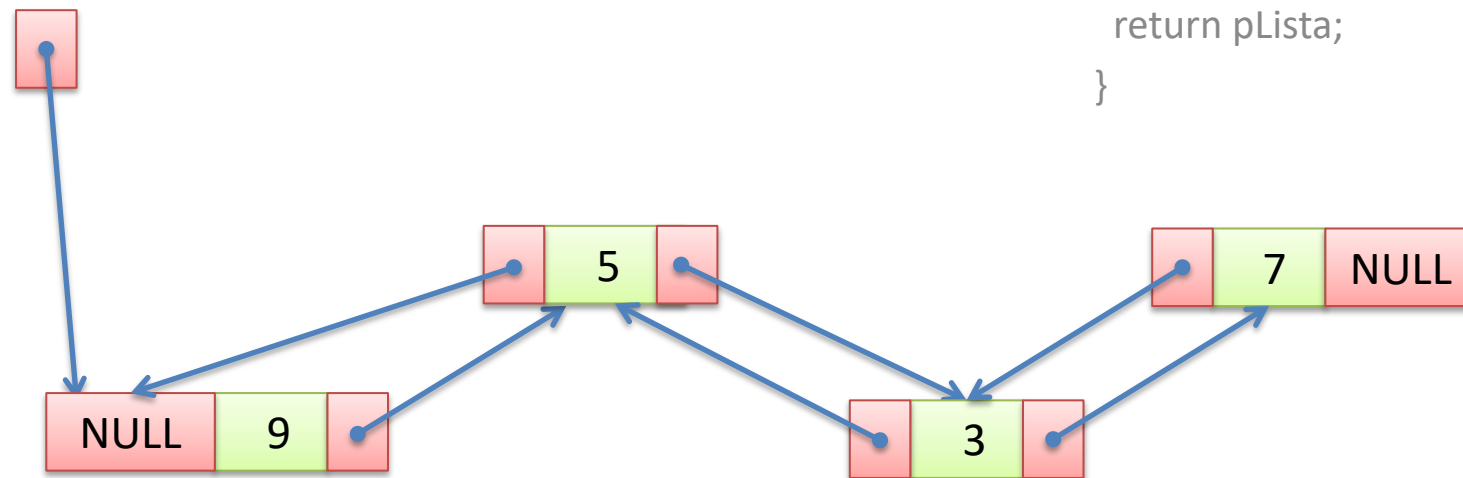


Lista Duplamente Encadeada

Inclusão na Calda

```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Prox = NULL;
    pAux = pLista;
    while (pAux->Prox != NULL)
        pAux = pAux->Prox;
    pAux->Prox = pNovoNo;
    pNovoNo->Ant = pAux;
    return pLista;
}
```

Lista

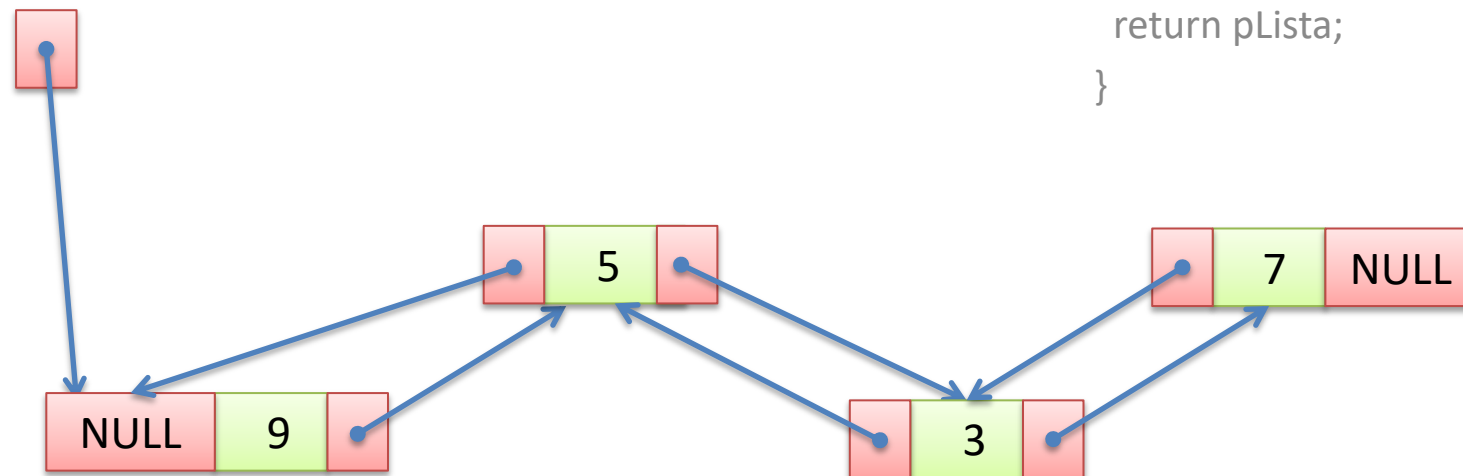


Lista Duplamente Encadeada

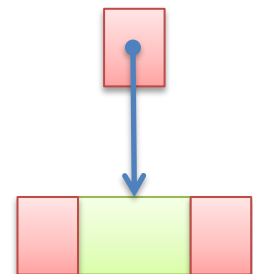
Inclusão na Calda

```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Prox = NULL;
    pAux = pLista;
    while (pAux->Prox != NULL)
        pAux = pAux->Prox;
    pAux->Prox = pNovoNo;
    pNovoNo->Ant = pAux;
    return pLista;
}
```

Lista



pNovoNo

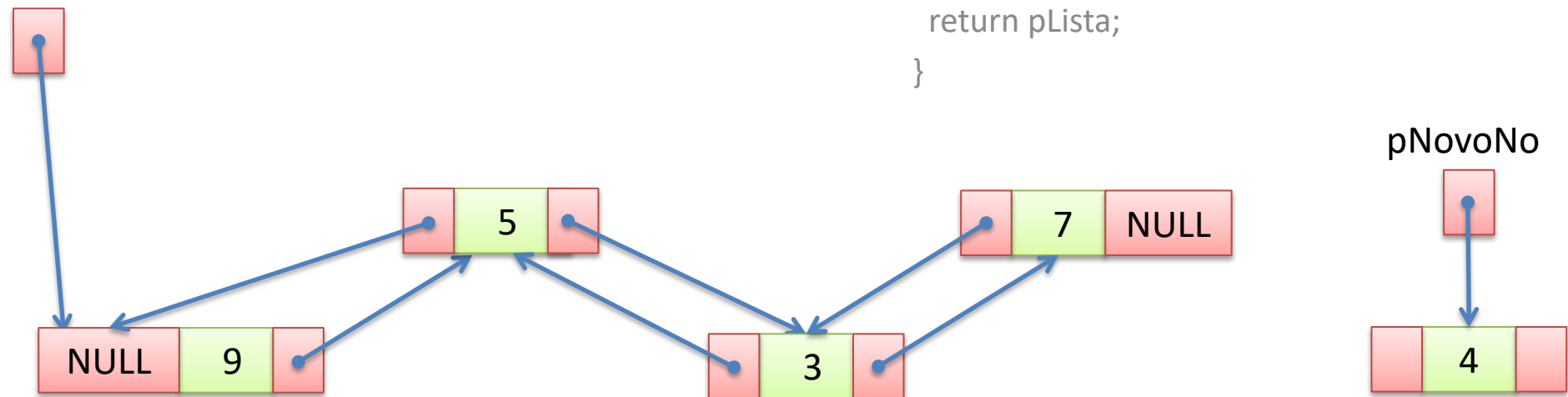


Lista Duplamente Encadeada

Inclusão na Calda

```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Prox = NULL;
    pAux = pLista;
    while (pAux->Prox != NULL)
        pAux = pAux->Prox;
    pAux->Prox = pNovoNo;
    pNovoNo->Ant = pAux;
    return pLista;
}
```

Lista



Listas Duplamente Encadeadas

Dr. Nilton Correia da Silva

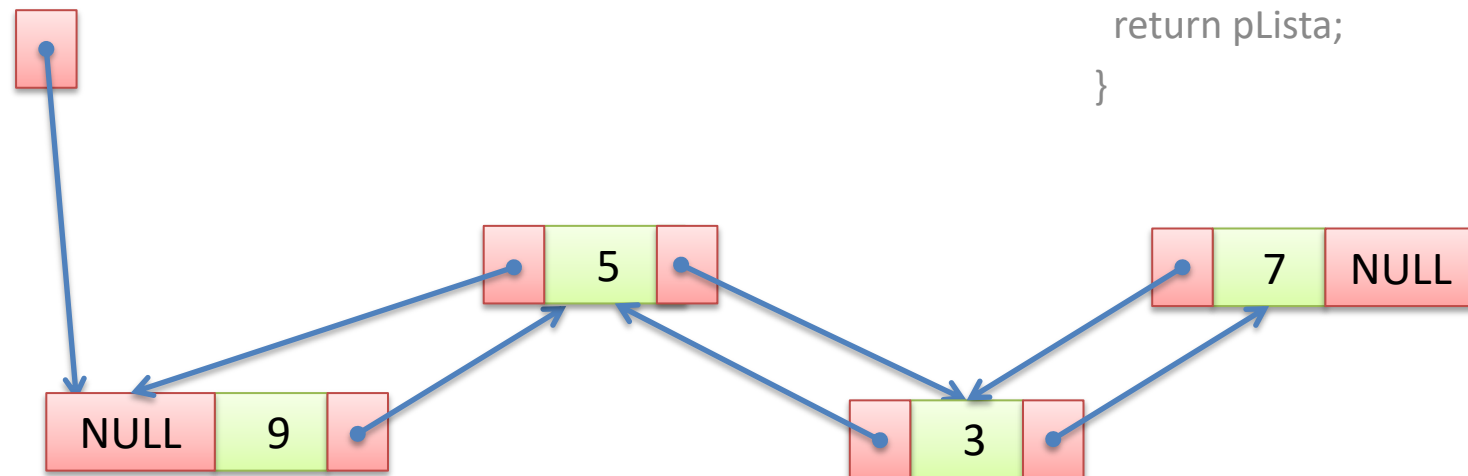
Estrutura de Dados e Algoritmos

Lista Duplamente Encadeada

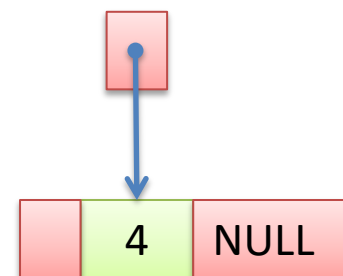
Inclusão na Calda

```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Prox = NULL;
    pAux = pLista;
    while (pAux->Prox != NULL)
        pAux = pAux->Prox;
    pAux->Prox = pNovoNo;
    pNovoNo->Ant = pAux;
    return pLista;
}
```

Lista



pNovoNo

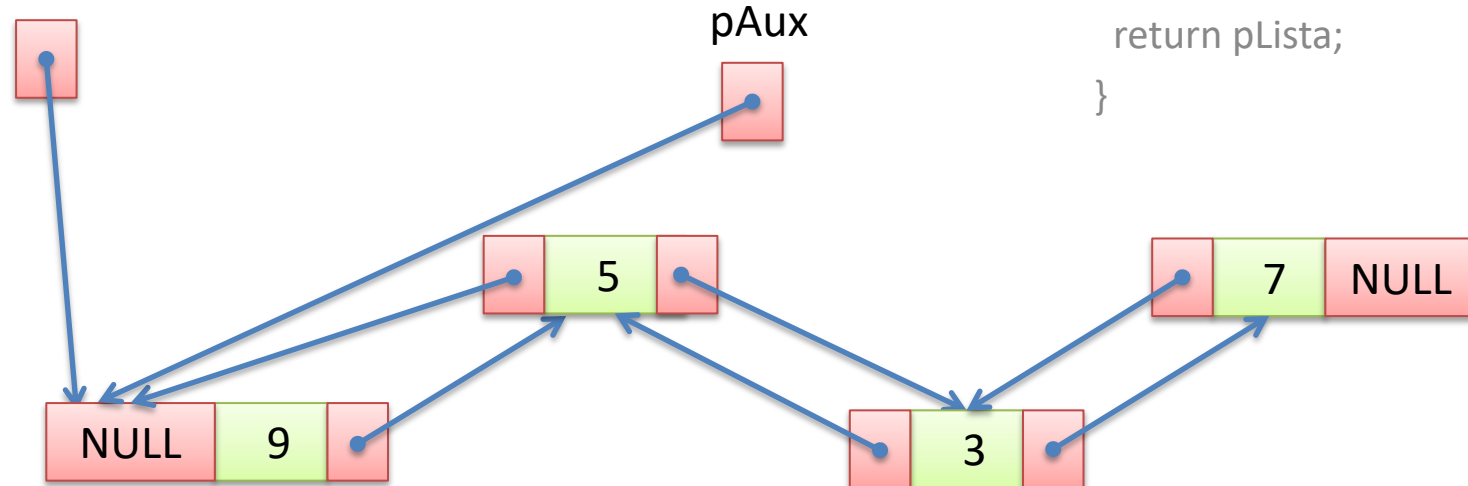


Lista Duplamente Encadeada

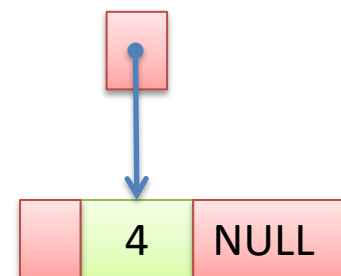
Inclusão na Calda

```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Prox = NULL;
    pAux = pLista;
    while (pAux->Prox != NULL)
        pAux = pAux->Prox;
    pAux->Prox = pNovoNo;
    pNovoNo->Ant = pAux;
    return pLista;
}
```

Lista



pNovoNo



Lista Duplamente Encadeada

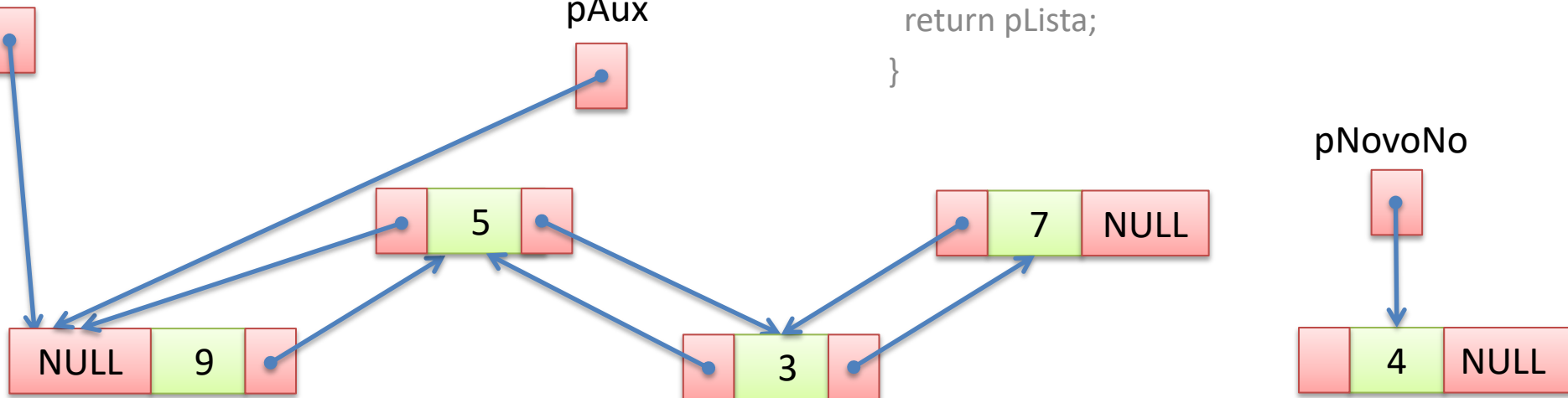
Inclusão na Calda

```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Prox = NULL;
    pAux = pLista;
    while (pAux->Prox != NULL)
        pAux = pAux->Prox;
    pAux->Prox = pNovoNo;
    pNovoNo->Ant = pAux;
    return pLista;
}
```

Lista

pAux

pNovoNo

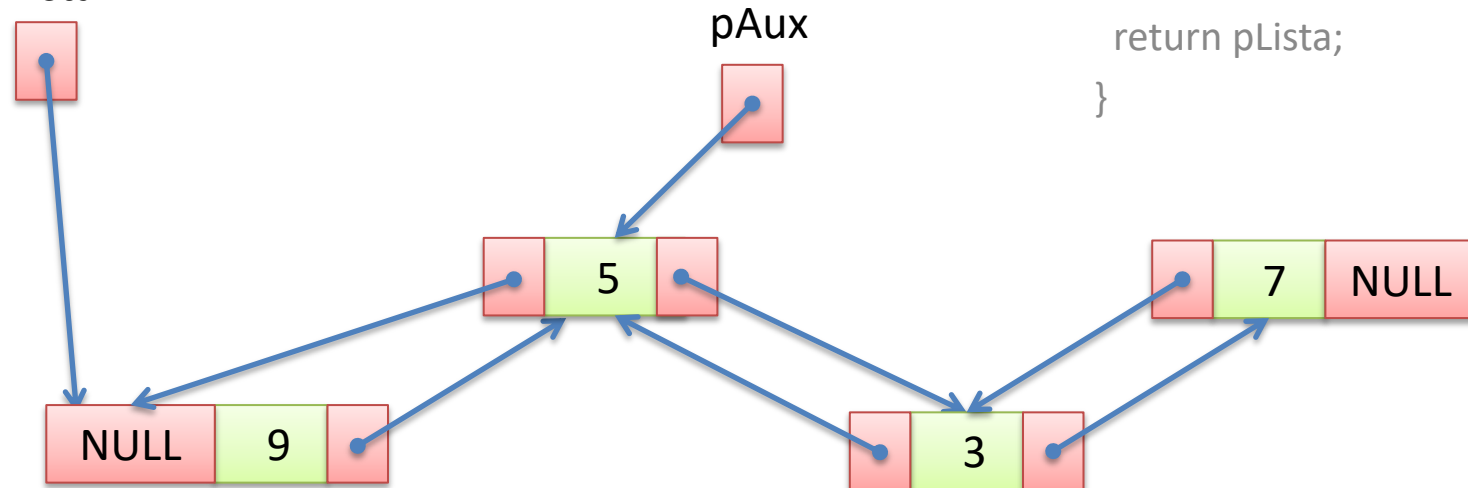


Lista Duplamente Encadeada

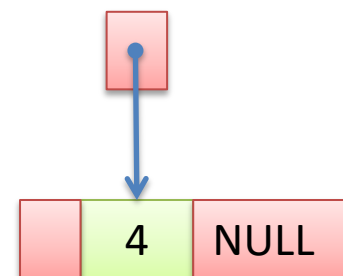
Inclusão na Calda

```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Prox = NULL;
    pAux = pLista;
    while (pAux->Prox != NULL)
        pAux = pAux->Prox;
    pAux->Prox = pNovoNo;
    pNovoNo->Ant = pAux;
    return pLista;
}
```

Lista



pNovoNo

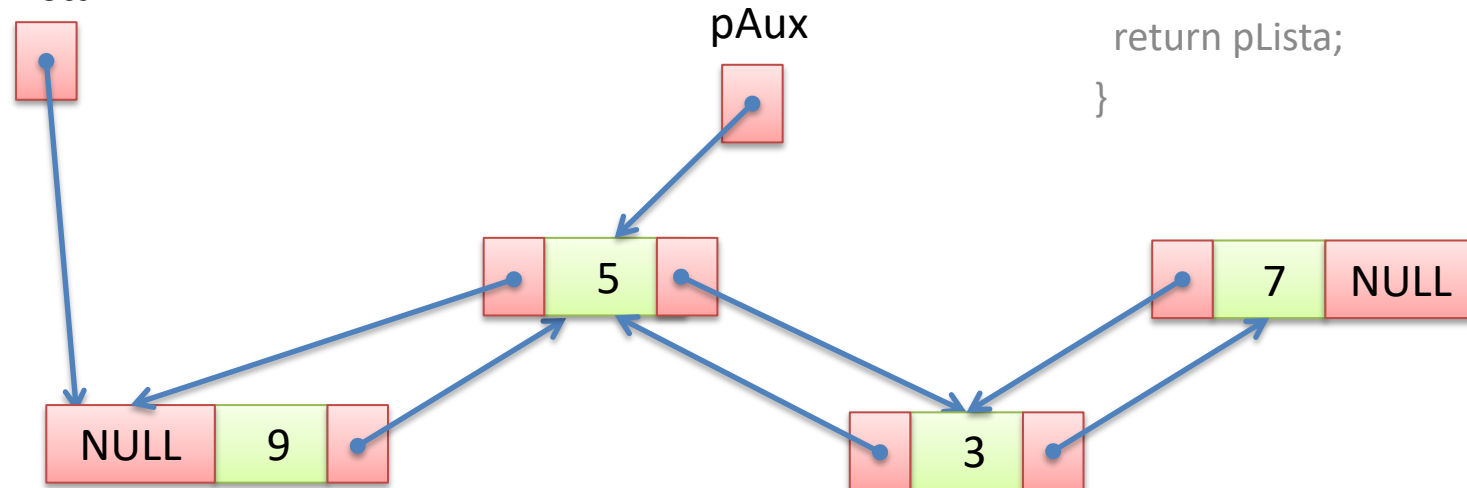


Lista Duplamente Encadeada

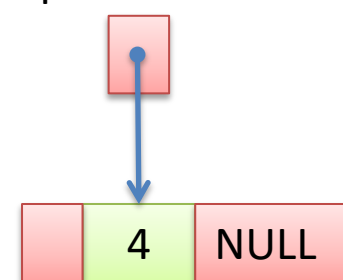
Inclusão na Calda

```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Prox = NULL;
    pAux = pLista;
    while (pAux->Prox != NULL)
        pAux = pAux->Prox;
    pAux->Prox = pNovoNo;
    pNovoNo->Ant = pAux;
    return pLista;
}
```

Lista



pNovoNo

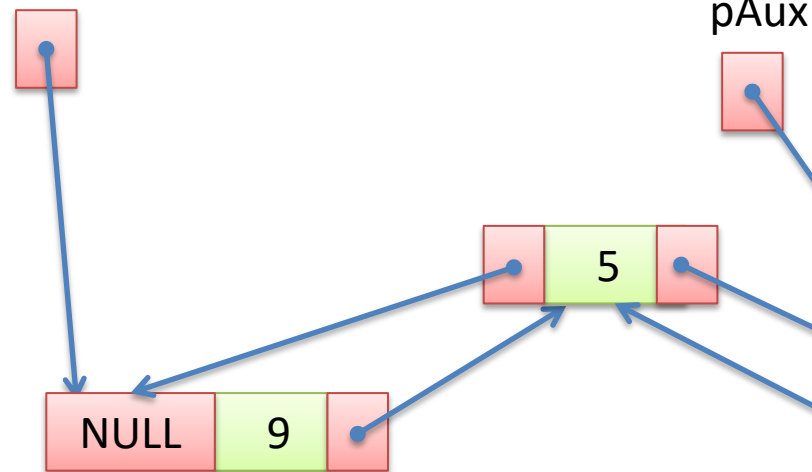


Lista Duplamente Encadeada

Inclusão na Calda

```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Prox = NULL;
    pAux = pLista;
    while (pAux->Prox != NULL)
        pAux = pAux->Prox;
    pAux->Prox = pNovoNo;
    pNovoNo->Ant = pAux;
    return pLista;
}
```

Lista



Listas Duplamente Encadeadas

Dr. Nilton Correia da Silva

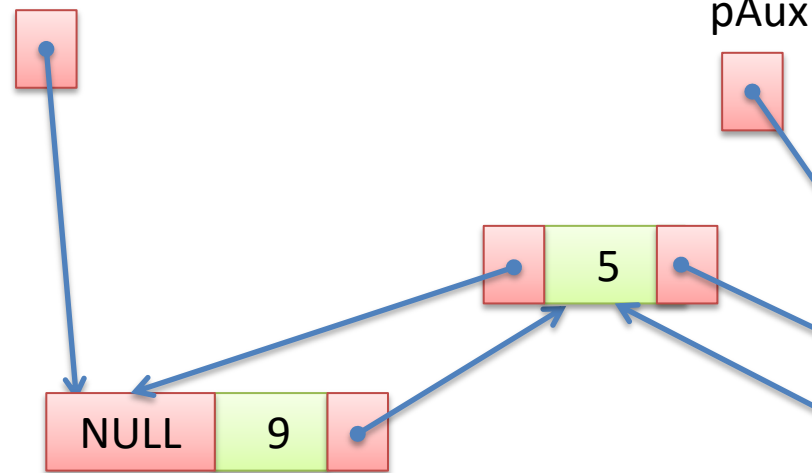
Estrutura de Dados e Algoritmos

Lista Duplamente Encadeada

Inclusão na Calda

```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Prox = NULL;
    pAux = pLista;
    while (pAux->Prox != NULL)
        pAux = pAux->Prox;
    pAux->Prox = pNovoNo;
    pNovoNo->Ant = pAux;
    return pLista;
}
```

Lista

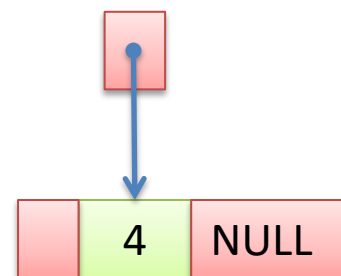


Listas Duplamente Encadeadas

Dr. Nilton Correia da Silva

Estrutura de Dados e Algoritmos

pNovoNo

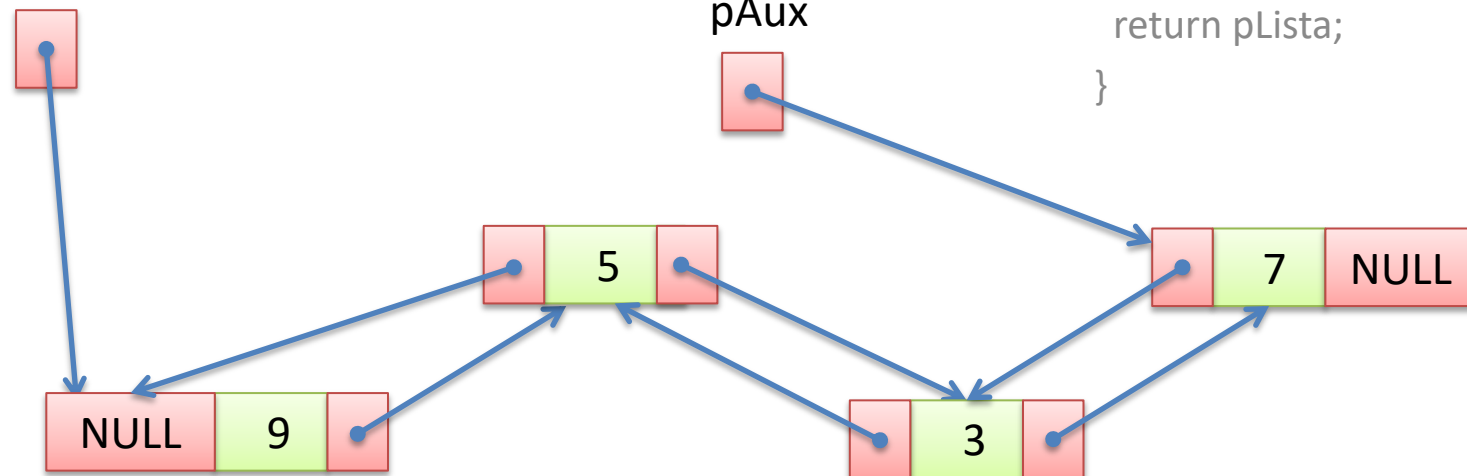


Lista Duplamente Encadeada

Inclusão na Calda

```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Prox = NULL;
    pAux = pLista;
    while (pAux->Prox != NULL)
        pAux = pAux->Prox;
    pAux->Prox = pNovoNo;
    pNovoNo->Ant = pAux;
    return pLista;
}
```

Lista



Listas Duplamente Encadeadas

Dr. Nilton Correia da Silva

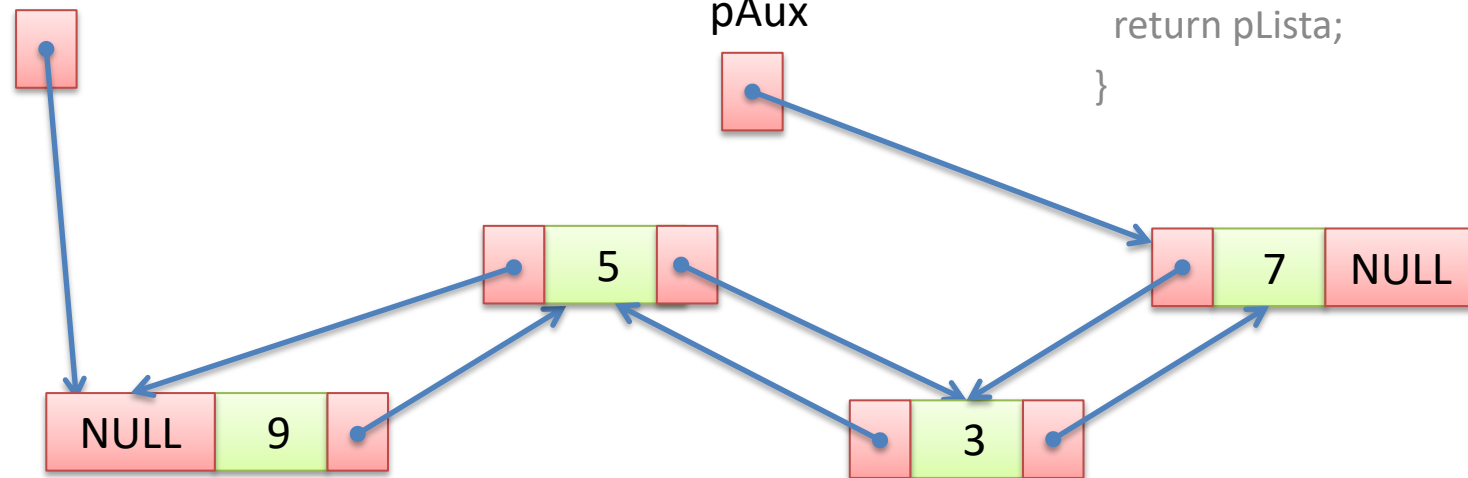
Estrutura de Dados e Algoritmos

Lista Duplamente Encadeada

Inclusão na Calda

```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Prox = NULL;
    pAux = pLista;
    while (pAux->Prox != NULL)
        pAux = pAux->Prox;
    pAux->Prox = pNovoNo;
    pNovoNo->Ant = pAux;
    return pLista;
}
```

Lista



Listas Duplamente Encadeadas

Dr. Nilton Correia da Silva

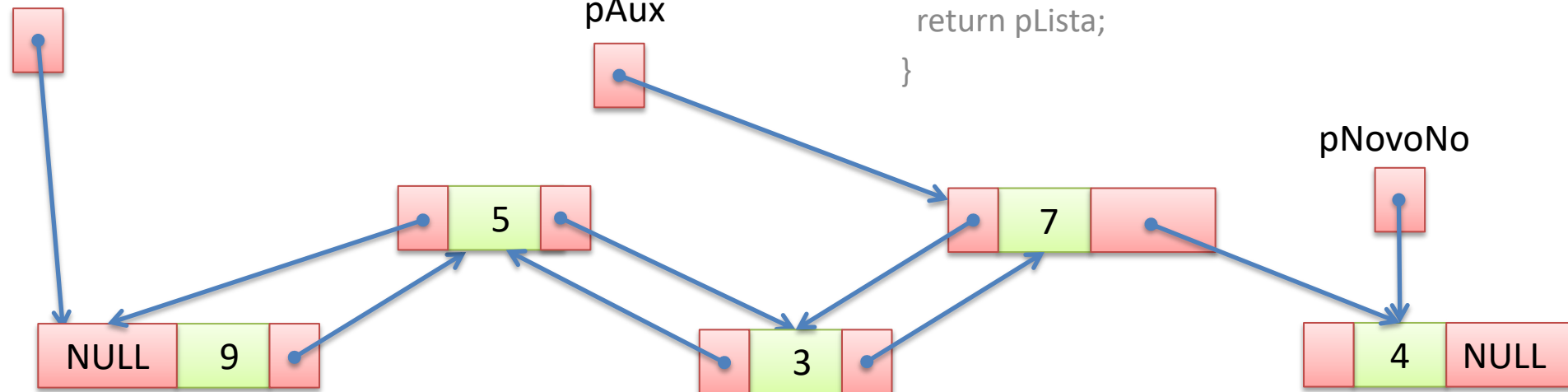
Estrutura de Dados e Algoritmos

Lista Duplamente Encadeada

Inclusão na Calda

```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Prox = NULL;
    pAux = pLista;
    while (pAux->Prox != NULL)
        pAux = pAux->Prox;
    pAux->Prox = pNovoNo;
    pNovoNo->Ant = pAux;
    return pLista;
}
```

Lista

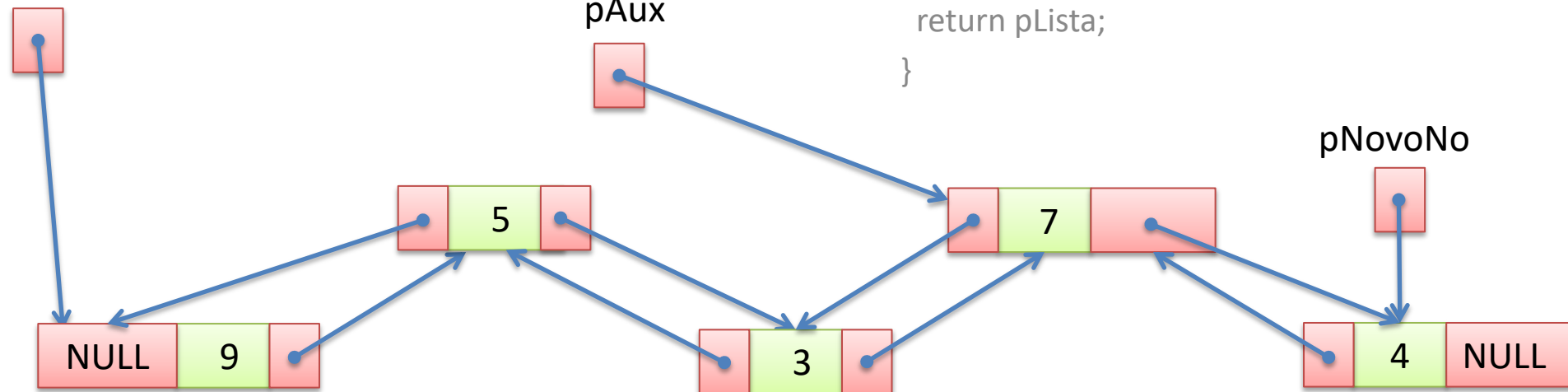


Lista Duplamente Encadeada

Inclusão na Calda

```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Prox = NULL;
    pAux = pLista;
    while (pAux->Prox != NULL)
        pAux = pAux->Prox;
    pAux->Prox = pNovoNo;
    pNovoNo->Ant = pAux;
    return pLista;
}
```

Lista

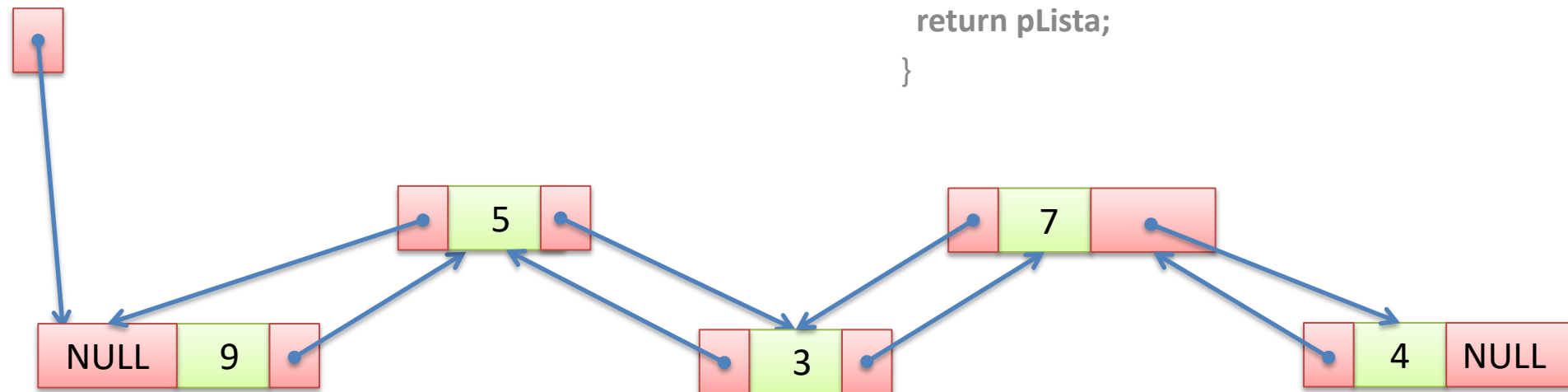


Lista Duplamente Encadeada

Inclusão na Calda

```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pNovoNo->Prox = NULL;
    pAux = pLista;
    while (pAux->Prox != NULL)
        pAux = pAux->Prox;
    pAux->Prox = pNovoNo;
    pNovoNo->Ant = pAux;
    return pLista;
}
```

Lista



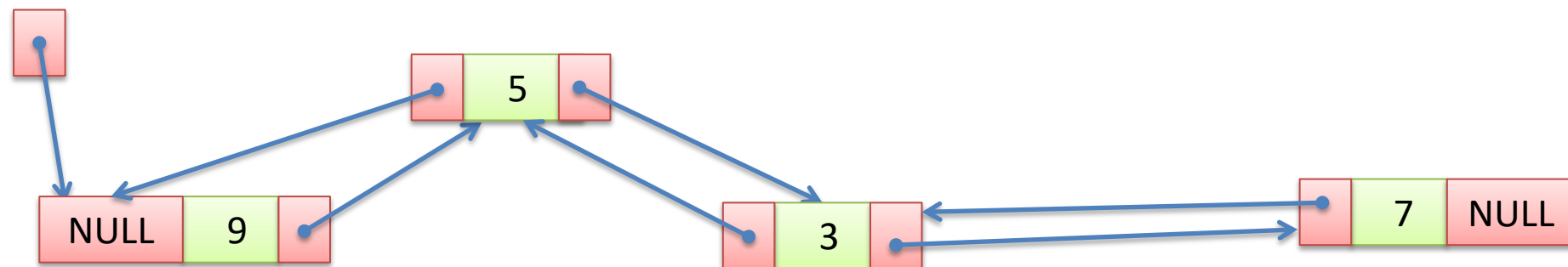
Lista Duplamente Encadeada Inclusão Depois de Uma Chave

```
TNo *IncluiDepois(TNo *pLista, int pChave, int pValor)
```

```
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pAux = pLista;
    while (pAux->Prox->Valor != pChave)
        pAux = pAux->Prox;
    pNovoNo->Prox = pAux->Prox;
    pNovoNo->Ant = pAux;
    pAux->Prox = pNovoNo;
    pNovoNo->Prox->Ant = pNovoNo;
    return pLista;
}
```

pChave = 3

Lista



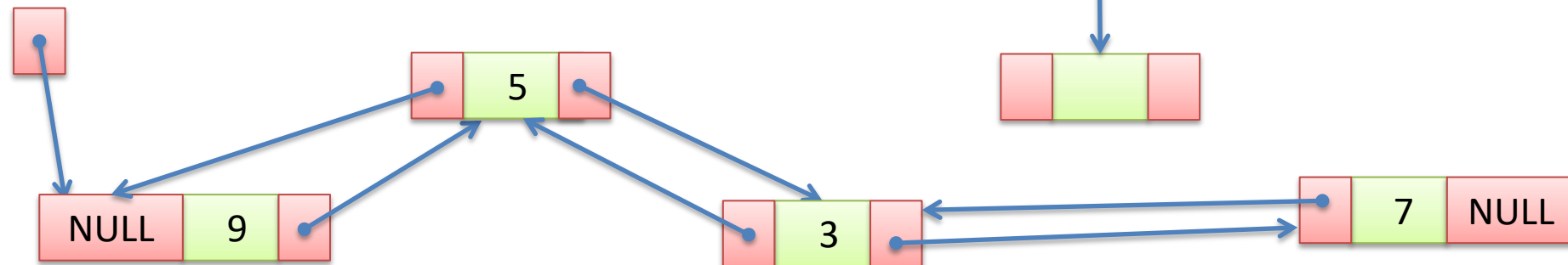
Lista Duplamente Encadeada Inclusão Depois de Uma Chave

```
TNo * IncluiDepois(TNo *pLista, int pChave, int pValor)
```

```
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pAux = pLista;
    while (pAux->Prox->Valor != pChave)
        pAux = pAux->Prox;
    pNovoNo->Prox = pAux->Prox;
    pNovoNo->Ant = pAux;
    pAux->Prox = pNovoNo;
    pNovoNo->Prox->Ant = pNovoNo;
    return pLista;
}
```

pChave = 3

Lista

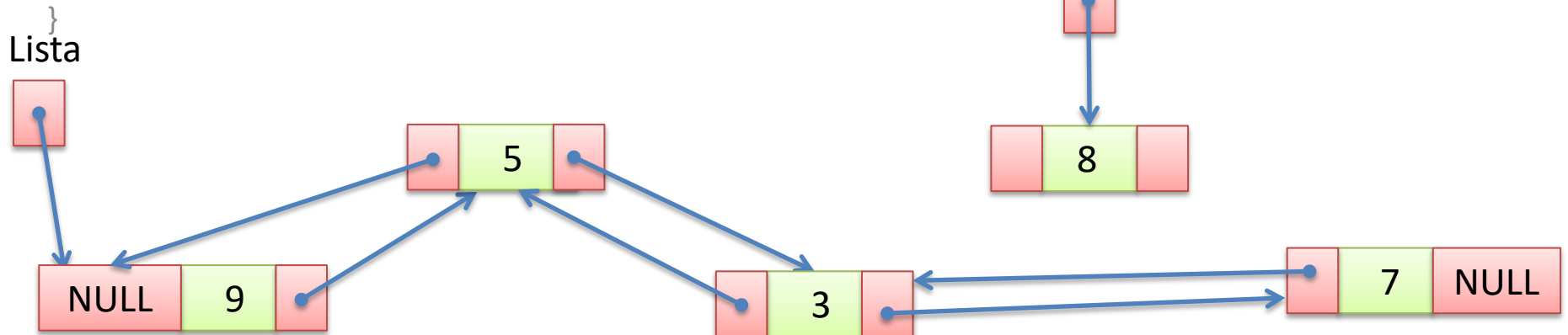


Lista Duplamente Encadeada Inclusão Depois de Uma Chave

```
TNo * IncluiDepois(TNo *pLista, int pChave, int pValor)
```

```
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pAux = pLista;
    while (pAux->Prox->Valor != pChave)
        pAux = pAux->Prox;
    pNovoNo->Prox = pAux->Prox;
    pNovoNo->Ant = pAux;
    pAux->Prox = pNovoNo;
    pNovoNo->Prox->Ant = pNovoNo;
    return pLista;
}
```

pChave = 3

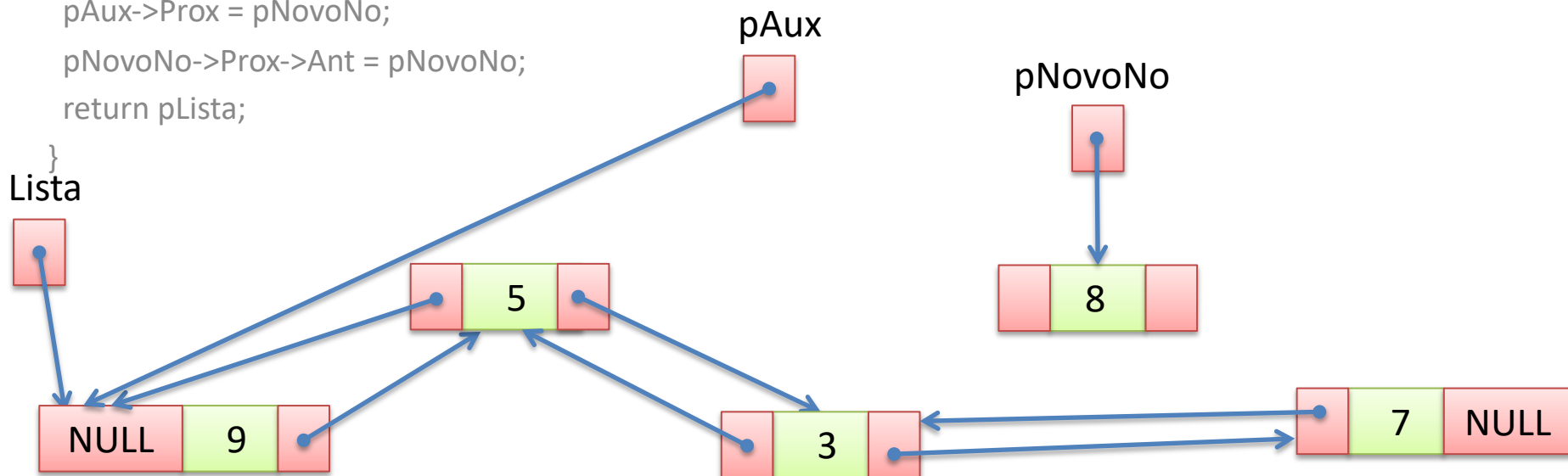


Lista Duplamente Encadeada Inclusão Depois de Uma Chave

```
TNo * IncluiDepois(TNo *pLista, int pChave, int pValor)
```

```
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pAux = pLista;
    while (pAux->Prox->Valor != pChave)
        pAux = pAux->Prox;
    pNovoNo->Prox = pAux->Prox;
    pNovoNo->Ant = pAux;
    pAux->Prox = pNovoNo;
    pNovoNo->Prox->Ant = pNovoNo;
    return pLista;
}
```

pChave = 3

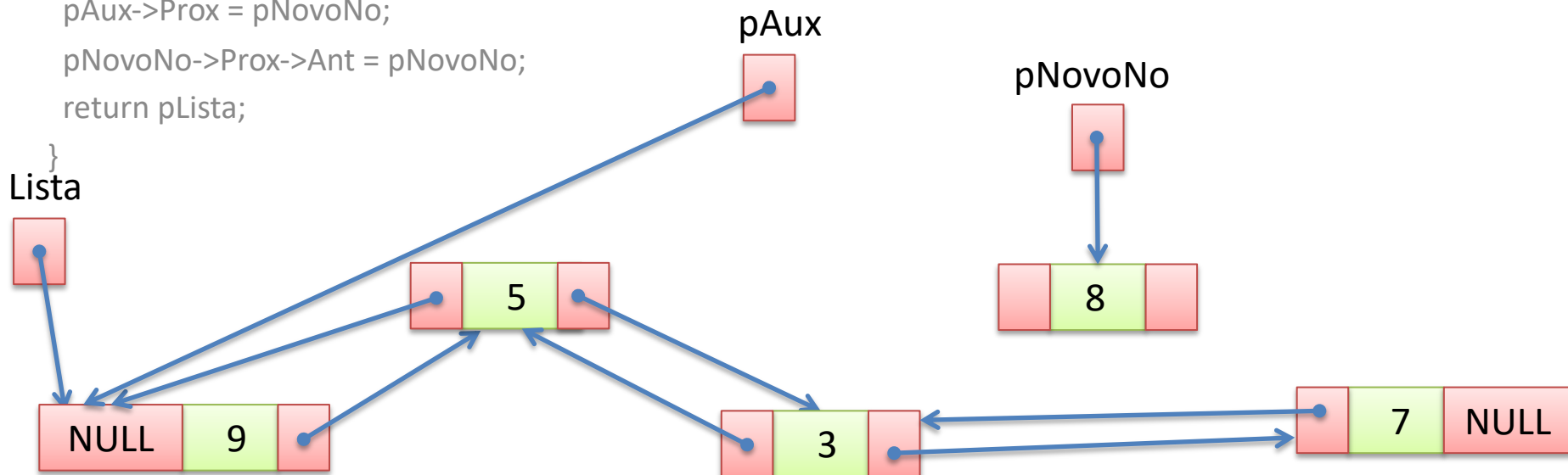


Lista Duplamente Encadeada Inclusão Depois de Uma Chave

```
TNo * IncluiDepois(TNo *pLista, int pChave, int pValor)
```

```
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pAux = pLista;
    while (pAux->Prox->Valor != pChave)
        pAux = pAux->Prox;
    pNovoNo->Prox = pAux->Prox;
    pNovoNo->Ant = pAux;
    pAux->Prox = pNovoNo;
    pNovoNo->Prox->Ant = pNovoNo;
    return pLista;
}
```

pChave = 3

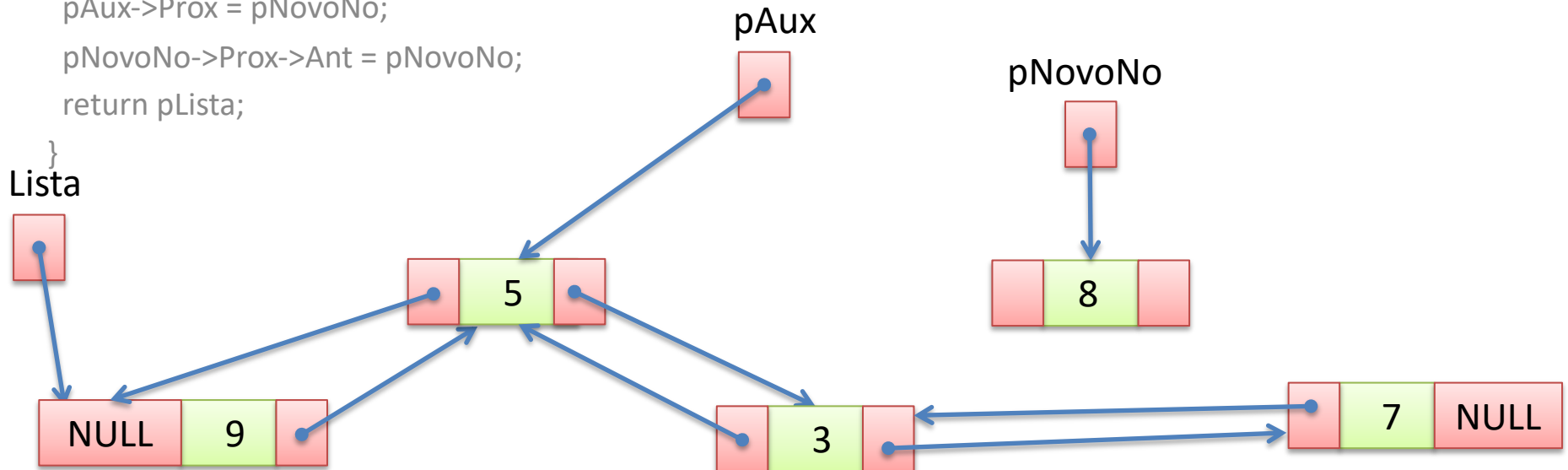


Lista Duplamente Encadeada Inclusão Depois de Uma Chave

```
TNo * IncluiDepois(TNo *pLista, int pChave, int pValor)
```

```
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pAux = pLista;
    while (pAux->Prox->Valor != pChave)
        pAux = pAux->Prox;
    pNovoNo->Prox = pAux->Prox;
    pNovoNo->Ant = pAux;
    pAux->Prox = pNovoNo;
    pNovoNo->Prox->Ant = pNovoNo;
    return pLista;
}
```

pChave = 3

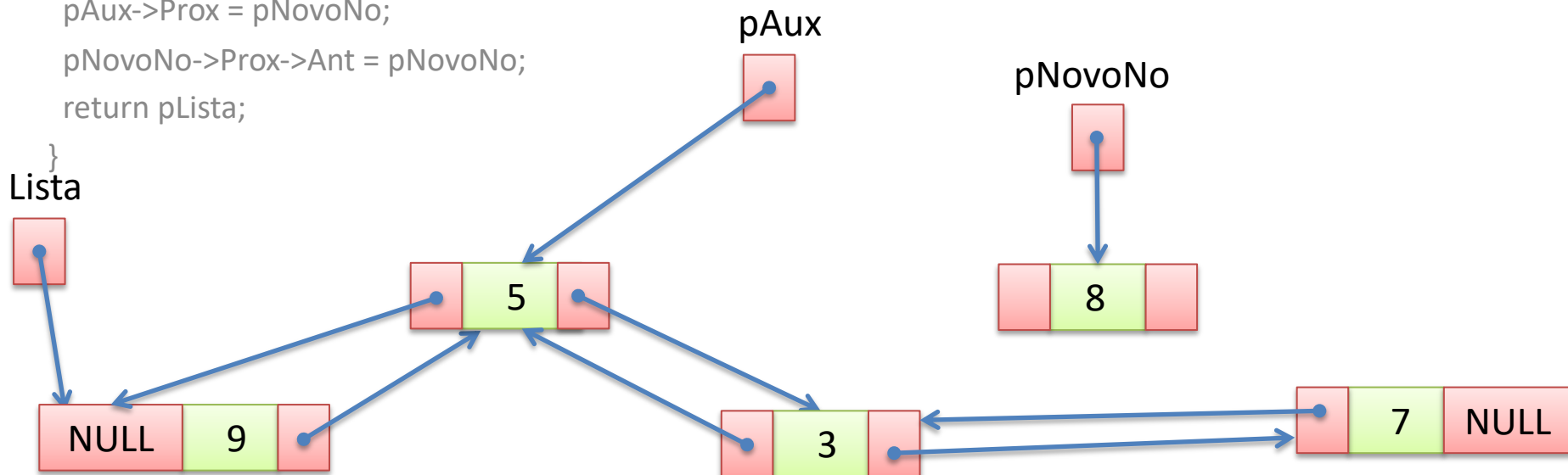


Lista Duplamente Encadeada Inclusão Depois de Uma Chave

```
TNo * IncluiDepois(TNo *pLista, int pChave, int pValor)
```

```
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pAux = pLista;
    while (pAux->Prox->Valor != pChave)
        pAux = pAux->Prox;
    pNovoNo->Prox = pAux->Prox;
    pNovoNo->Ant = pAux;
    pAux->Prox = pNovoNo;
    pNovoNo->Prox->Ant = pNovoNo;
    return pLista;
}
```

pChave = 3

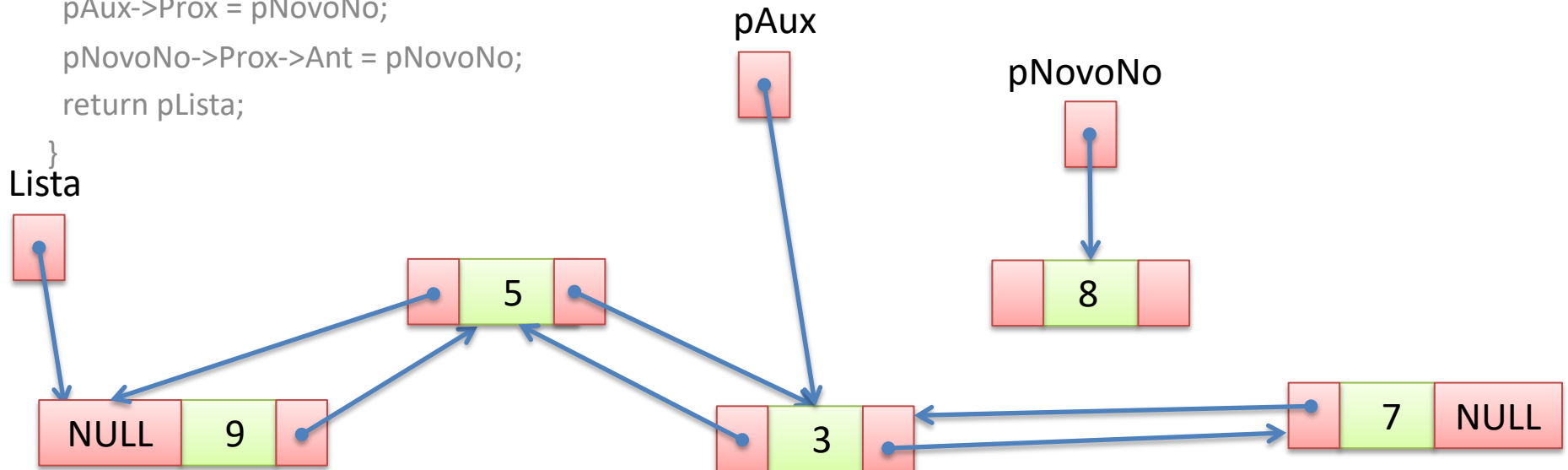


Lista Duplamente Encadeada Inclusão Depois de Uma Chave

```
TNo * IncluiDepois(TNo *pLista, int pChave, int pValor)
```

```
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pAux = pLista;
    while (pAux->Prox->Valor != pChave)
        pAux = pAux->Prox;
    pNovoNo->Prox = pAux->Prox;
    pNovoNo->Ant = pAux;
    pAux->Prox = pNovoNo;
    pNovoNo->Prox->Ant = pNovoNo;
    return pLista;
}
```

pChave = 3

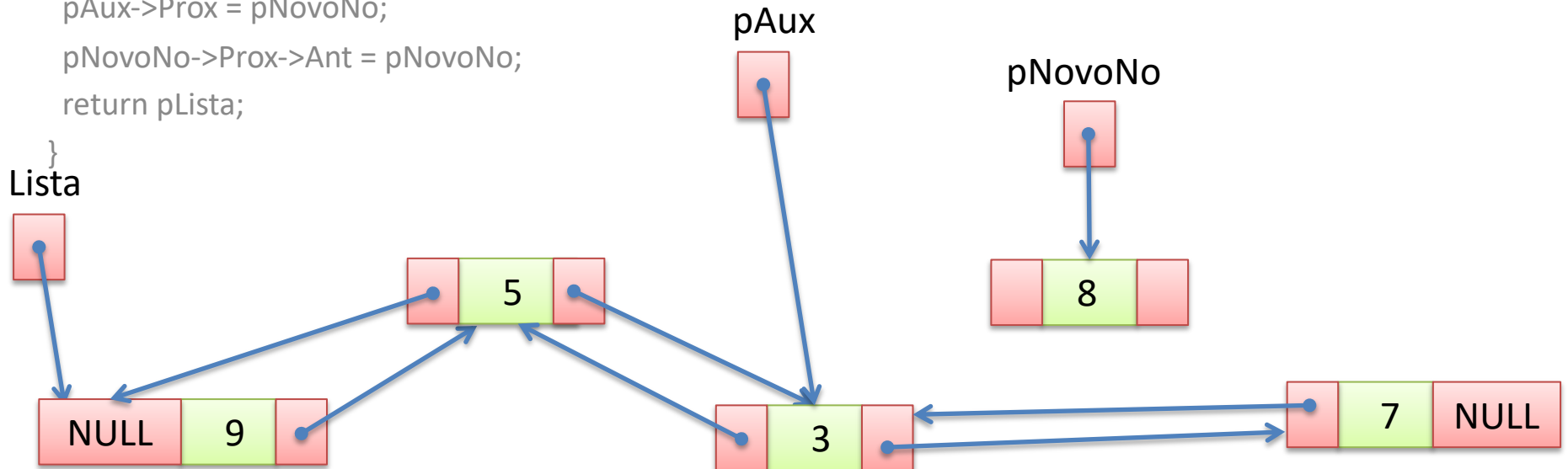


Lista Duplamente Encadeada Inclusão Depois de Uma Chave

```
TNo * IncluiDepois(TNo *pLista, int pChave, int pValor)
```

```
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pAux = pLista;
    while (pAux->Valor != pChave)
        pAux = pAux->Prox;
    pNovoNo->Prox = pAux->Prox;
    pNovoNo->Ant = pAux;
    pAux->Prox = pNovoNo;
    pNovoNo->Prox->Ant = pNovoNo;
    return pLista;
}
```

pChave = 3



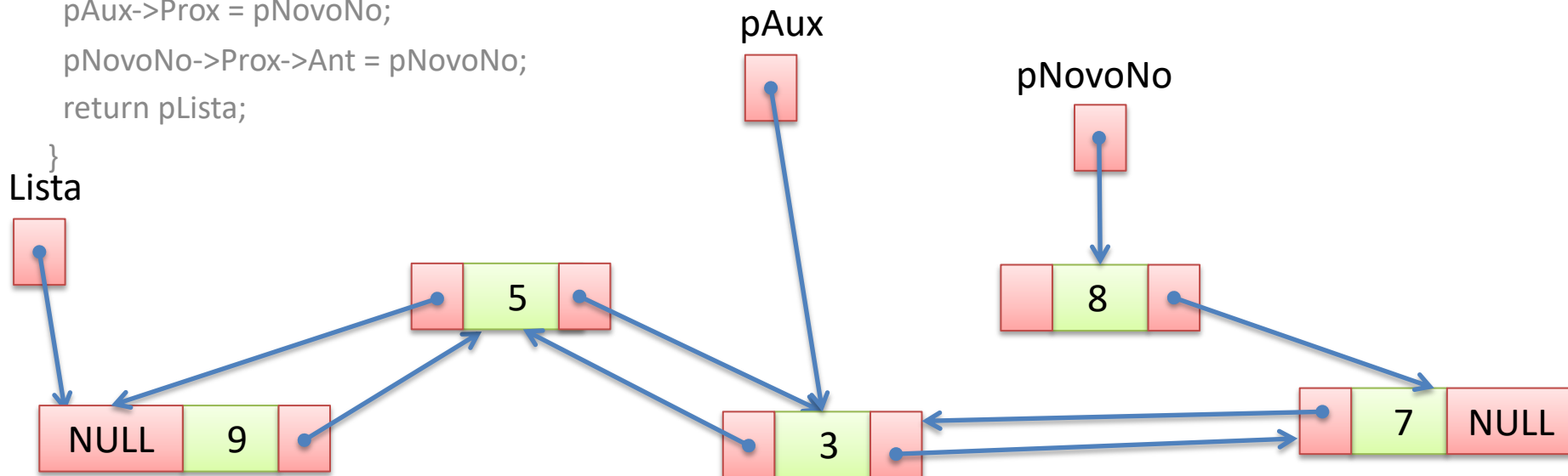
Lista Duplamente Encadeada Inclusão Depois de Uma Chave

```
TNo * IncluiDepois(TNo *pLista, int pChave, int pValor)
```

```
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pAux = pLista;
    while (pAux->Prox->Valor != pChave)
        pAux = pAux->Prox;
    pNovoNo->Prox = pAux->Prox;
    pNovoNo->Ant = pAux;
    pAux->Prox = pNovoNo;
    pNovoNo->Prox->Ant = pNovoNo;
    return pLista;
}
```

pChave = 3

Lista

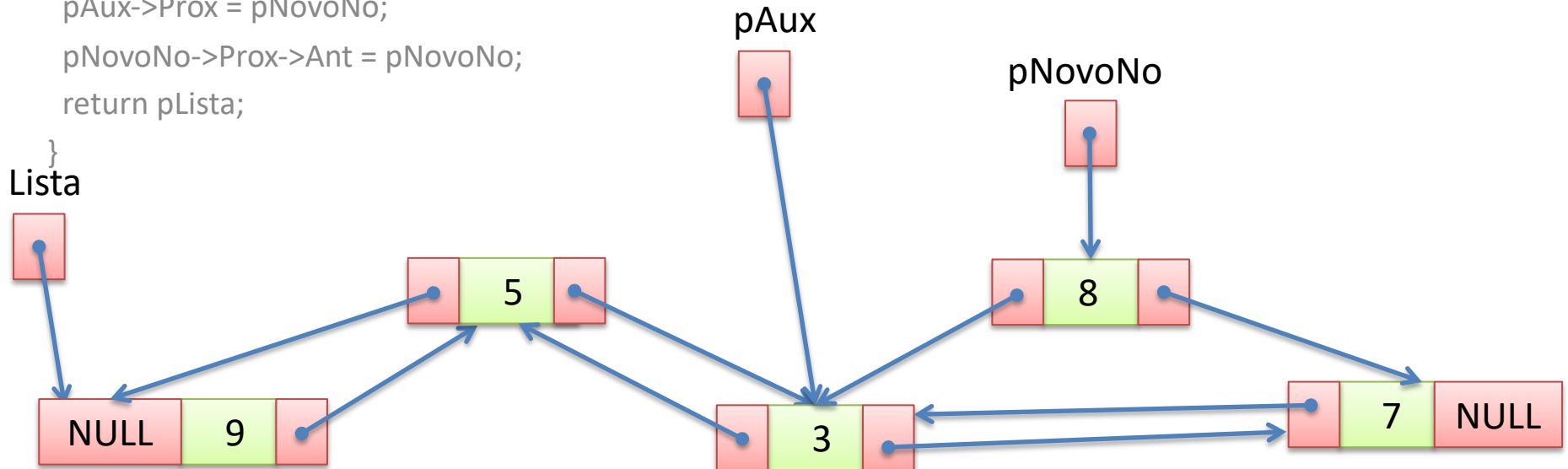


Lista Duplamente Encadeada Inclusão Depois de Uma Chave

```
TNo * IncluiDepois(TNo *pLista, int pChave, int pValor)
```

```
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pAux = pLista;
    while (pAux->Prox->Valor != pChave)
        pAux = pAux->Prox;
    pNovoNo->Prox = pAux->Prox;
    pNovoNo->Ant = pAux;
    pAux->Prox = pNovoNo;
    pNovoNo->Prox->Ant = pNovoNo;
    return pLista;
}
```

pChave = 3

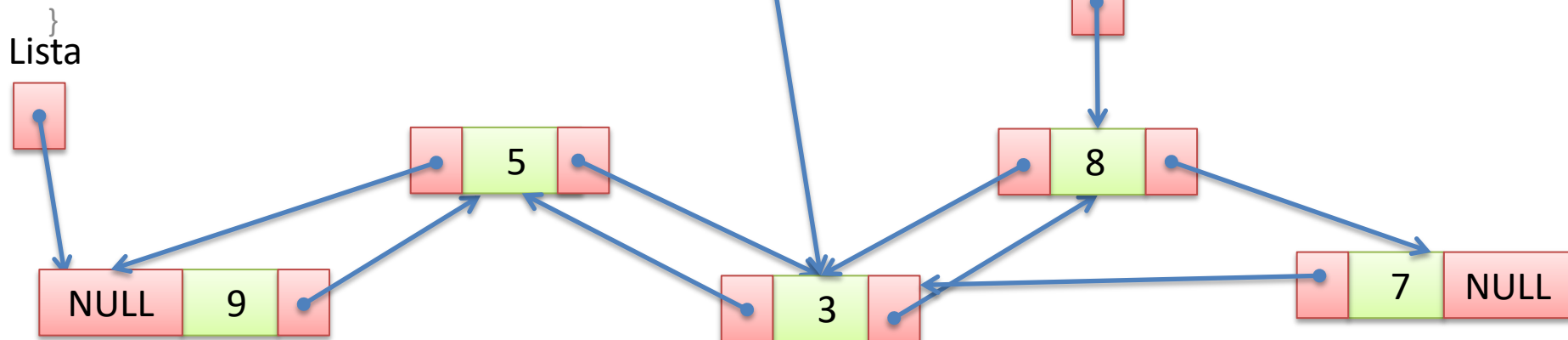


Lista Duplamente Encadeada Inclusão Depois de Uma Chave

```
TNo * IncluiDepois(TNo *pLista, int pChave, int pValor)
```

```
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pAux = pLista;
    while (pAux->Prox->Valor != pChave)
        pAux = pAux->Prox;
    pNovoNo->Prox = pAux->Prox;
    pNovoNo->Ant = pAux;
    pAux->Prox = pNovoNo;
    pNovoNo->Prox->Ant = pNovoNo;
    return pLista;
}
```

pChave = 3

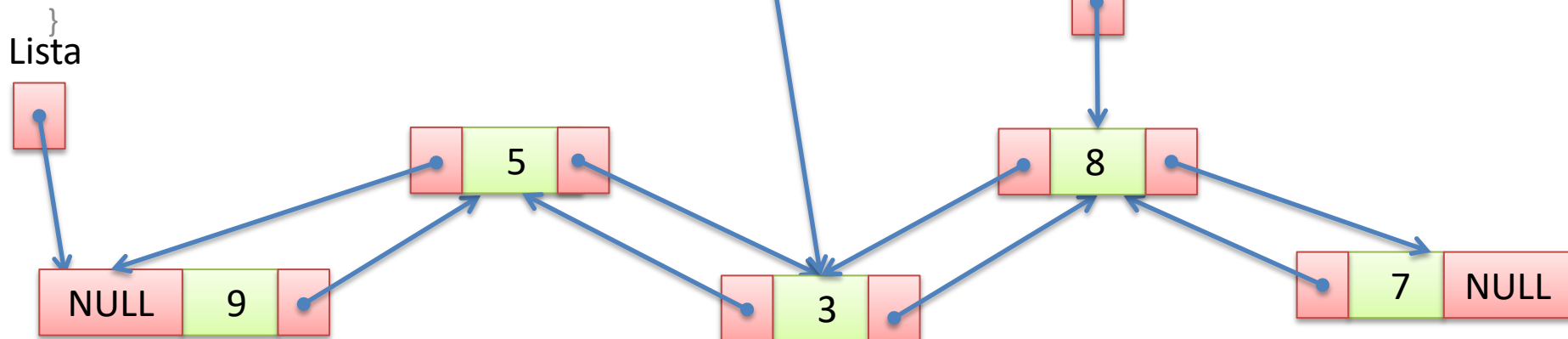


Lista Duplamente Encadeada Inclusão Depois de Uma Chave

```
TNo * IncluiDepois(TNo *pLista, int pChave, int pValor)
```

```
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pAux = pLista;
    while (pAux->Prox->Valor != pChave)
        pAux = pAux->Prox;
    pNovoNo->Prox = pAux->Prox;
    pNovoNo->Ant = pAux;
    pAux->Prox = pNovoNo;
    pNovo->Prox->Ant = pNovoNo;
    return pLista;
}
```

pChave = 3



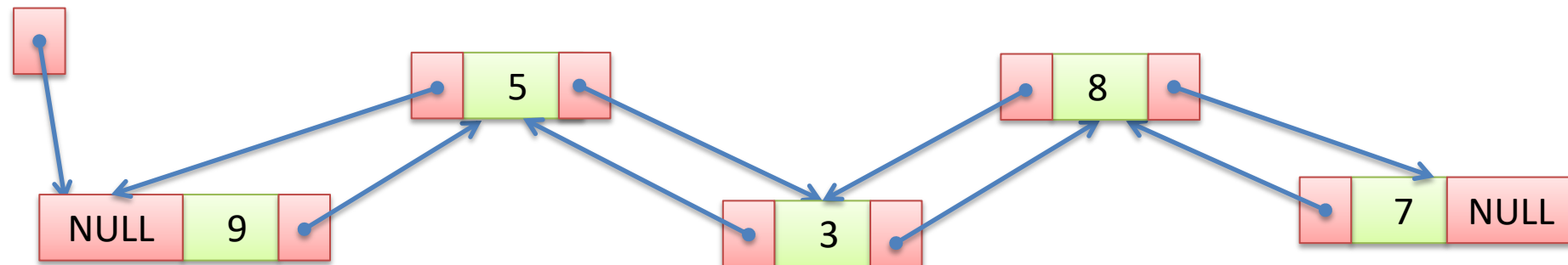
Lista Duplamente Encadeada Inclusão Depois de Uma Chave

```
TNo * IncluiDepois(TNo *pLista, int pChave, int pValor)
```

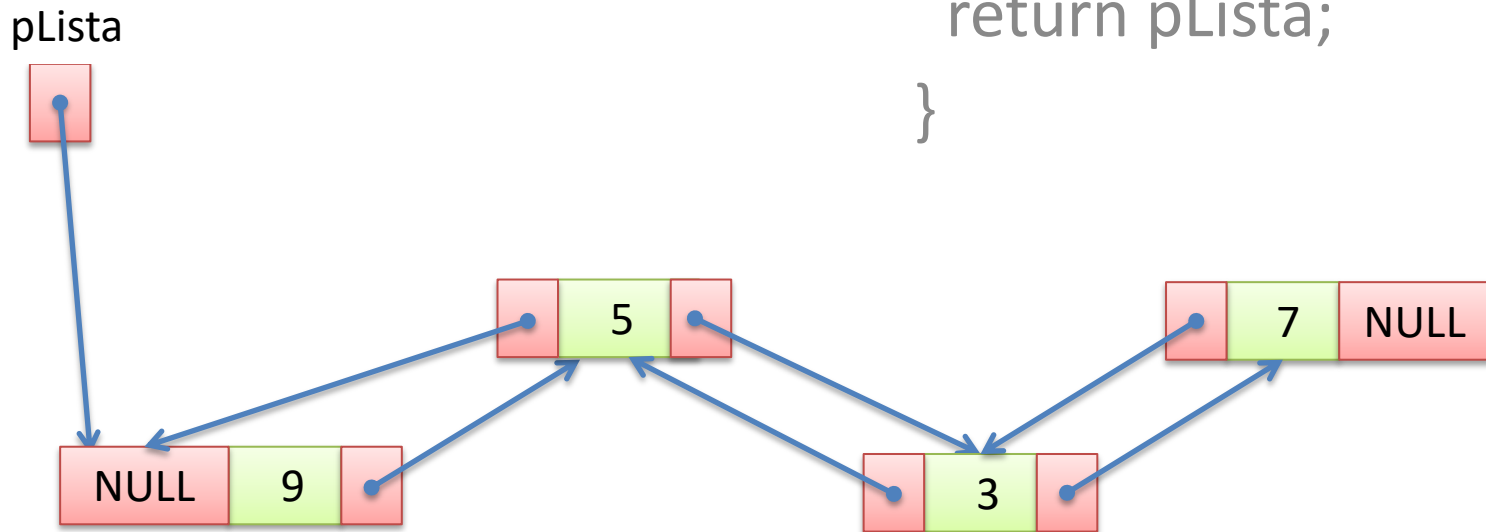
```
{
    TNo *pNovoNo, *pAux;
    pNovoNo = (TNo *) malloc(sizeof(TNo));
    pNovoNo->Numero = pValor;
    pAux = pLista;
    while (pAux->Prox->Valor != pChave)
        pAux = pAux->Prox;
    pNovoNo->Prox = pAux->Prox;
    pNovoNo->Ant = pAux;
    pAux->Prox = pNovoNo;
    pNovo->Prox->Ant = pNovoNo;
    return pLista;
}
```

pChave = 3

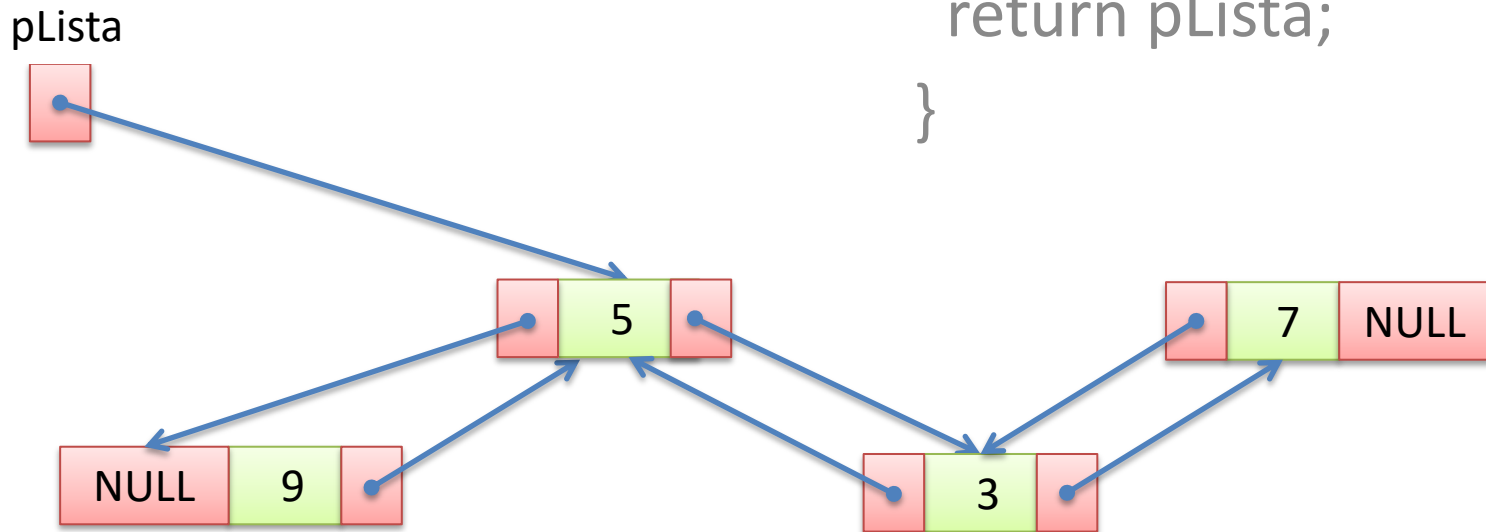
pLista



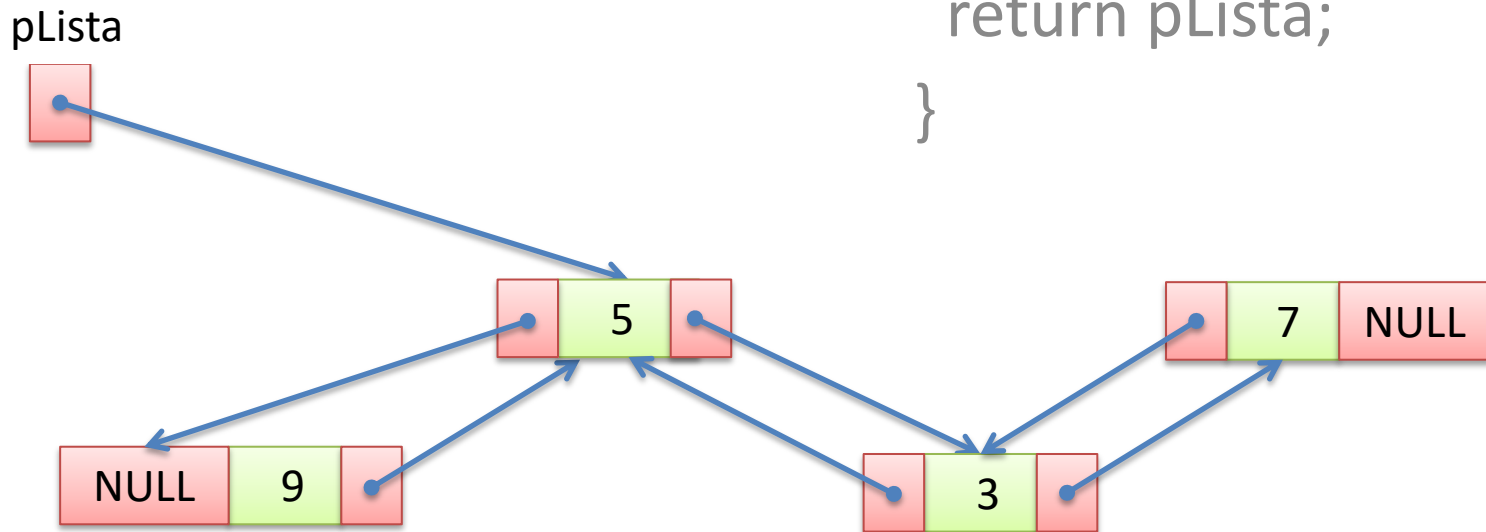
```
TNo *ExcluiCabeça(TNo *pLista)
{
    pLista = pLista->Prox;
    free(pLista->Ant);
    pLista->Ant = NULL;
    return pLista;
}
```



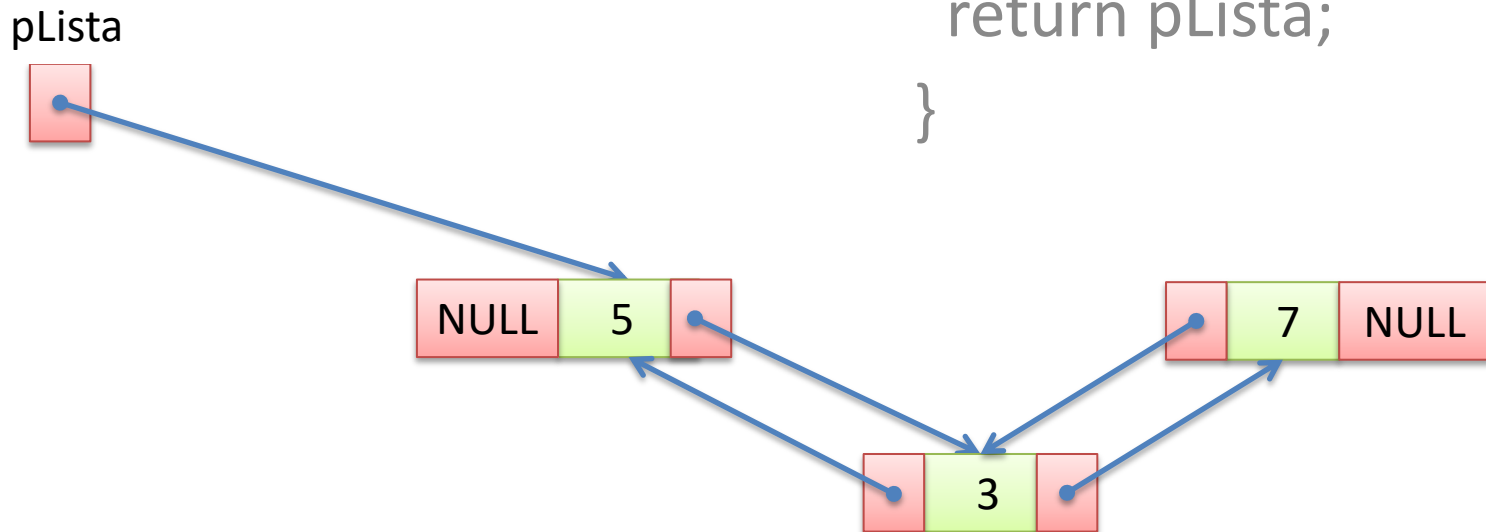
```
TNo *ExcluiCabeça(TNo *pLista)
{
    pLista = pLista->Prox;
    free(pLista->Ant);
    pLista->Ant = NULL;
    return pLista;
}
```



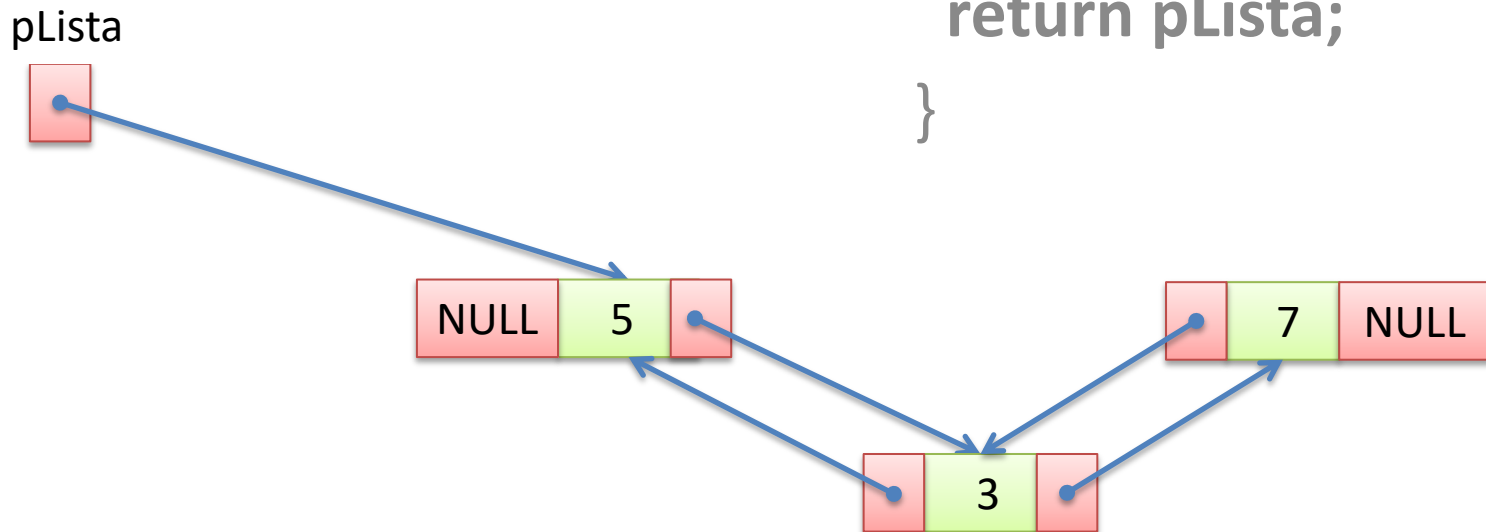
```
TNo *ExcluiCabeça(TNo *pLista)
{
    pLista = pLista->Prox;
    free(pLista->Ant);
    pLista->Ant = NULL;
    return pLista;
}
```



```
TNo *ExcluiCabeça(TNo *pLista)
{
    pLista = pLista->Prox;
    free(pLista->Ant);
    pLista->Ant = NULL;
    return pLista;
}
```

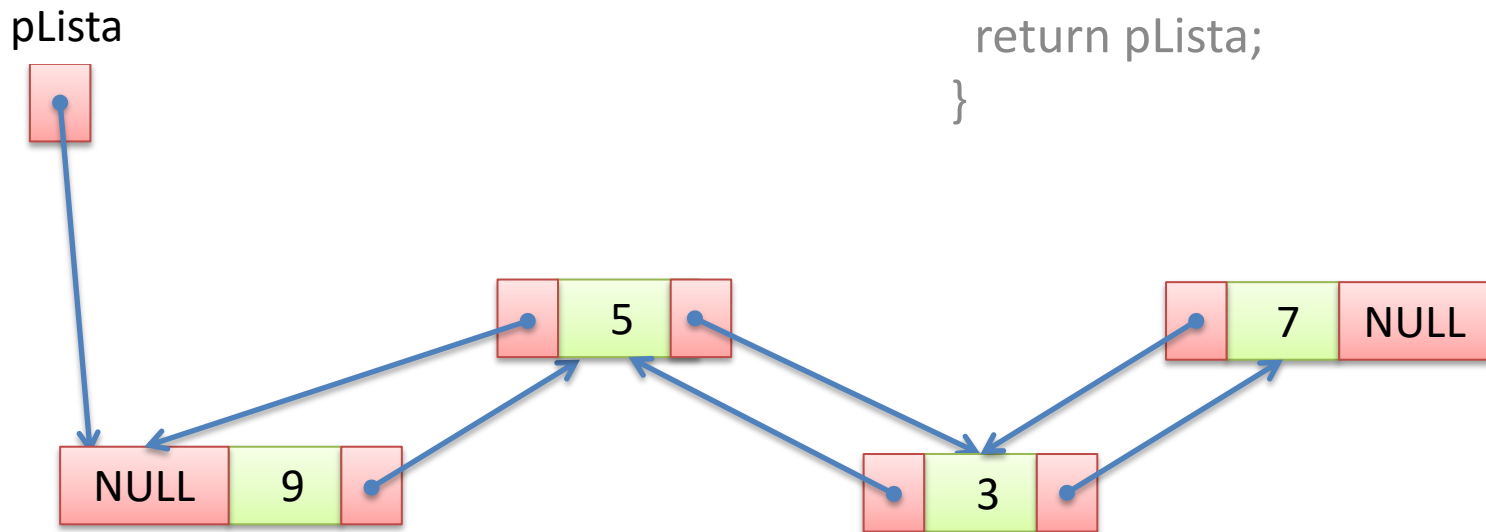


```
TNo *ExcluiCabeça(TNo *pLista)
{
    pLista = pLista->Prox;
    free(pLista->Ant);
    pLista->Ant = NULL;
    return pLista;
}
```



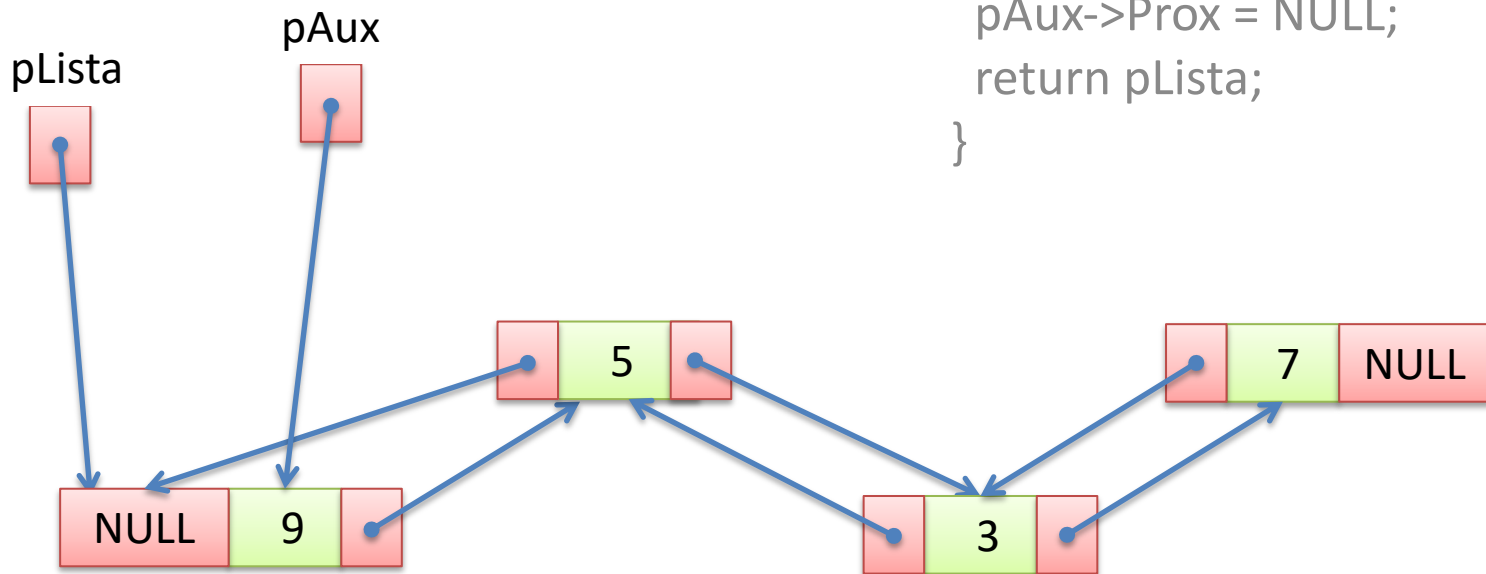
Lista Duplamente Encadeada Exclusão na Calda

```
TNo *ExcluiCalda(TNo *pLista)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Prox->Prox != NULL)
        pAux = pAux->Prox;
    free(pAux->Prox);
    pAux->Prox = NULL;
    return pLista;
}
```



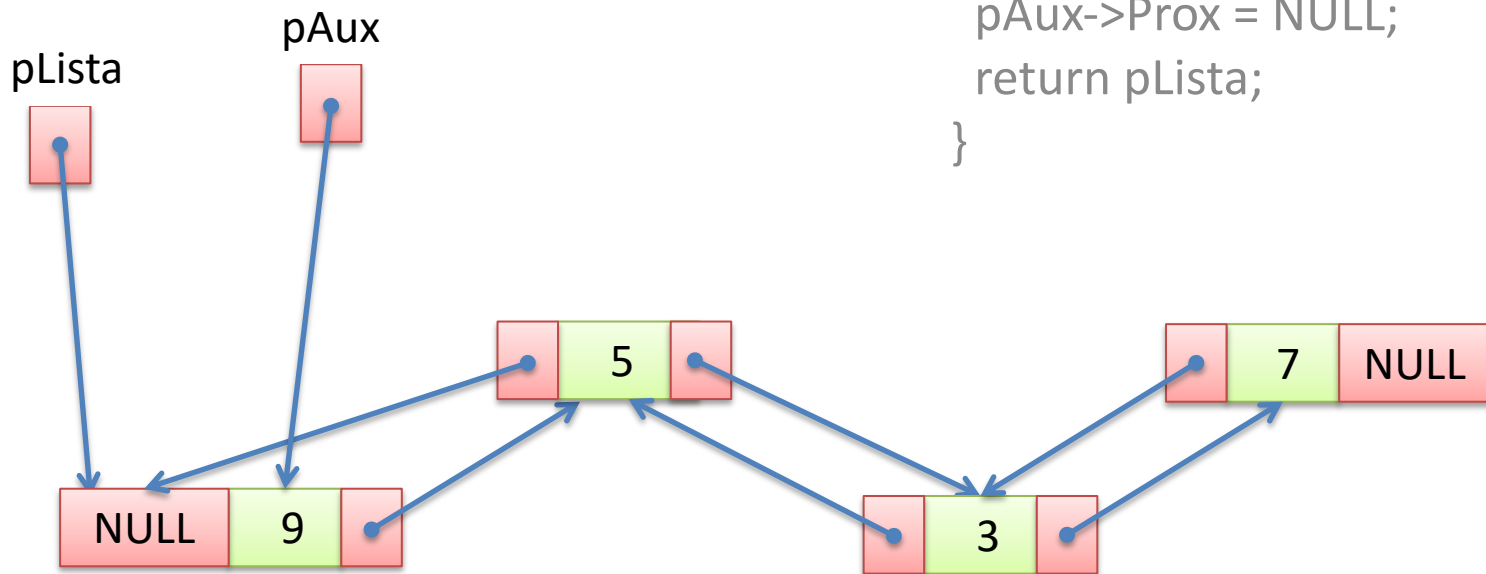
Lista Duplamente Encadeada Exclusão na Calda

```
TNo *ExcluiCalda(TNo *pLista)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Prox->Prox != NULL)
        pAux = pAux->Prox;
    free(pAux->Prox);
    pAux->Prox = NULL;
    return pLista;
}
```



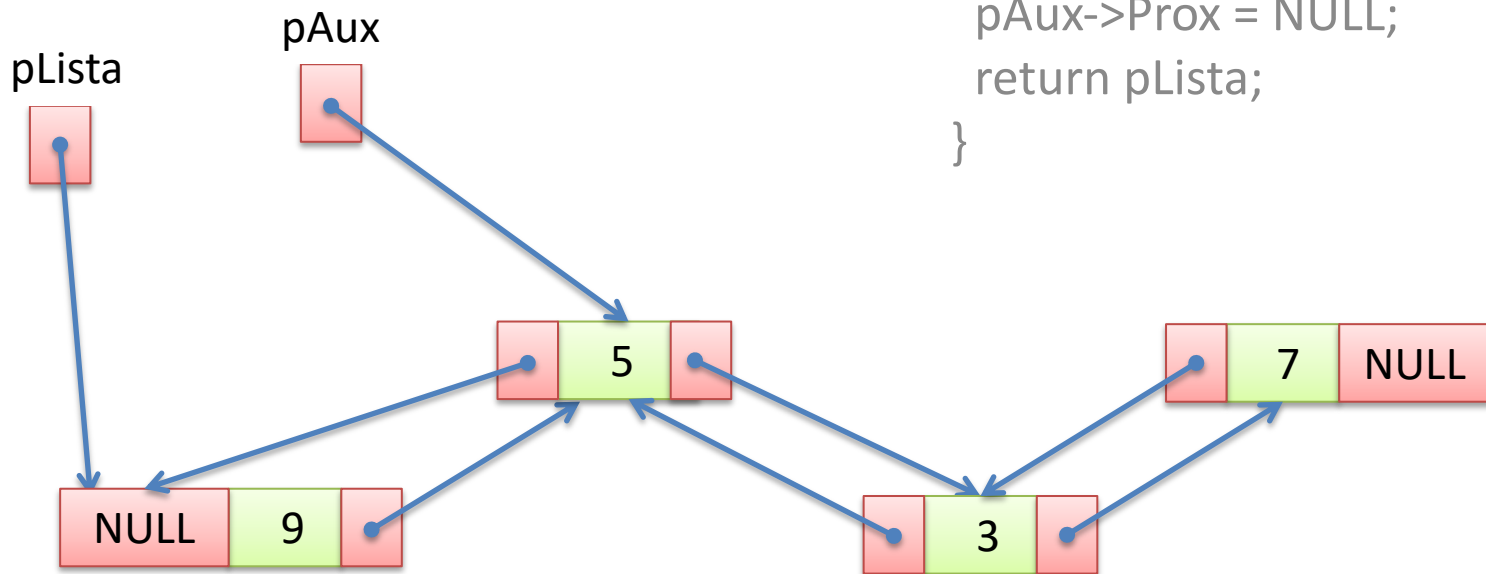
Lista Duplamente Encadeada Exclusão na Calda

```
TNo *ExcluiCalda(TNo *pLista)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Prox->Prox != NULL)
        pAux = pAux->Prox;
    free(pAux->Prox);
    pAux->Prox = NULL;
    return pLista;
}
```



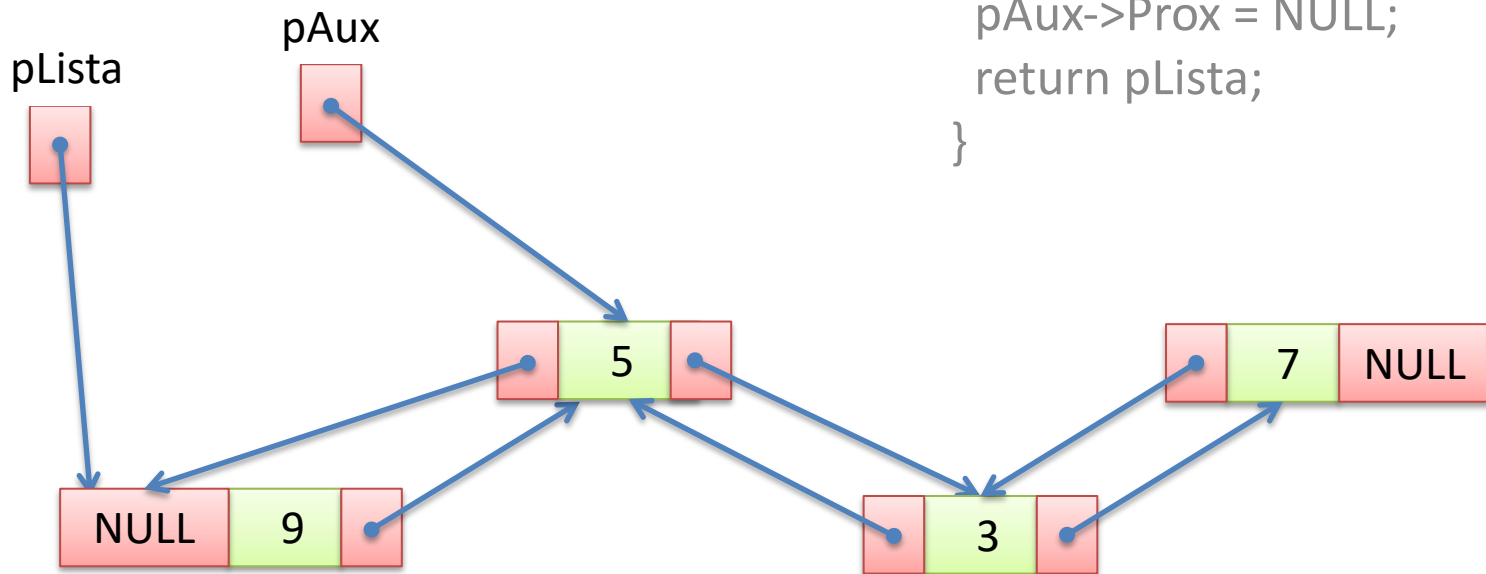
Lista Duplamente Encadeada Exclusão na Calda

```
TNo *ExcluiCalda(TNo *pLista)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Prox->Prox != NULL)
        pAux = pAux->Prox;
    free(pAux->Prox);
    pAux->Prox = NULL;
    return pLista;
}
```



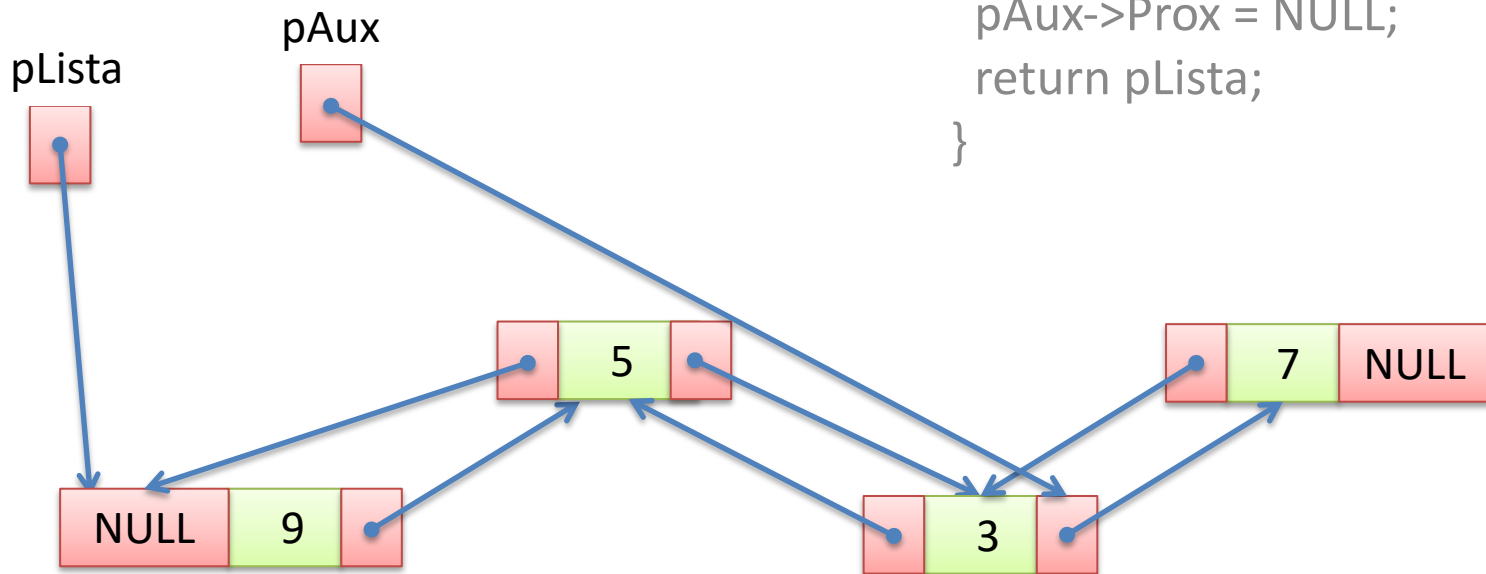
Lista Duplamente Encadeada Exclusão na Calda

```
TNo *ExcluiCalda(TNo *pLista)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Prox->Prox != NULL)
        pAux = pAux->Prox;
    free(pAux->Prox);
    pAux->Prox = NULL;
    return pLista;
}
```



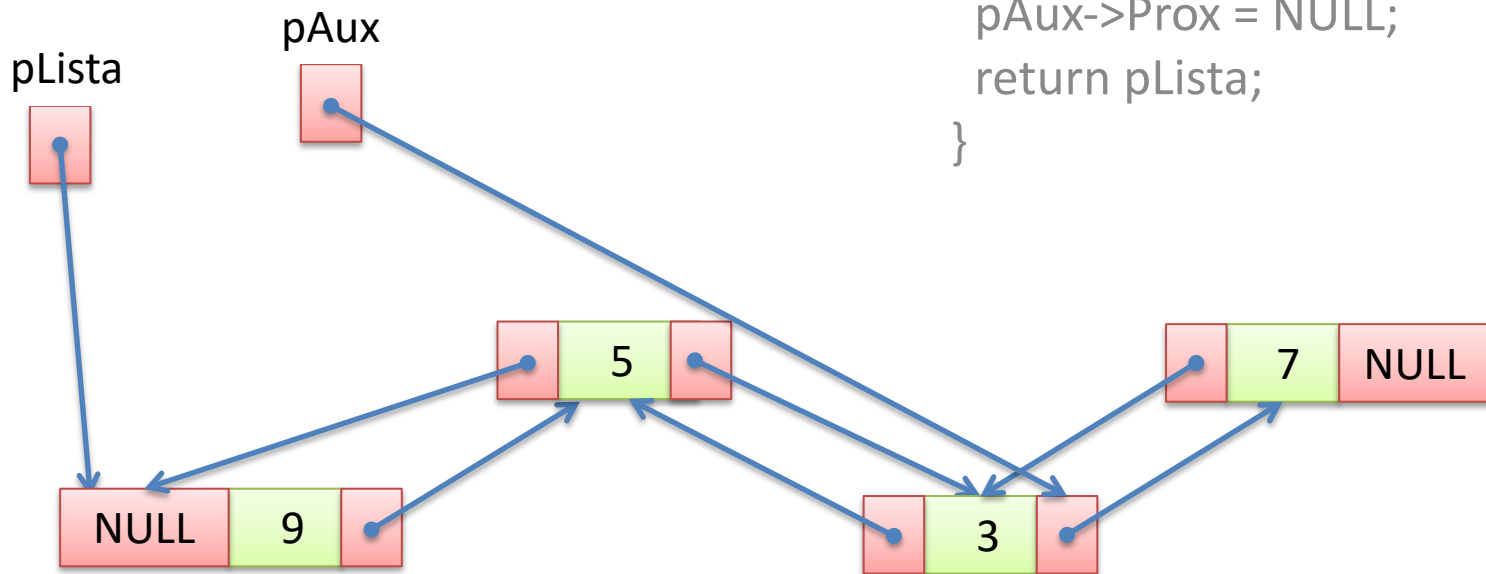
Lista Duplamente Encadeada Exclusão na Calda

```
TNo *ExcluiCalda(TNo *pLista)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Prox->Prox != NULL)
        pAux = pAux->Prox;
    free(pAux->Prox);
    pAux->Prox = NULL;
    return pLista;
}
```



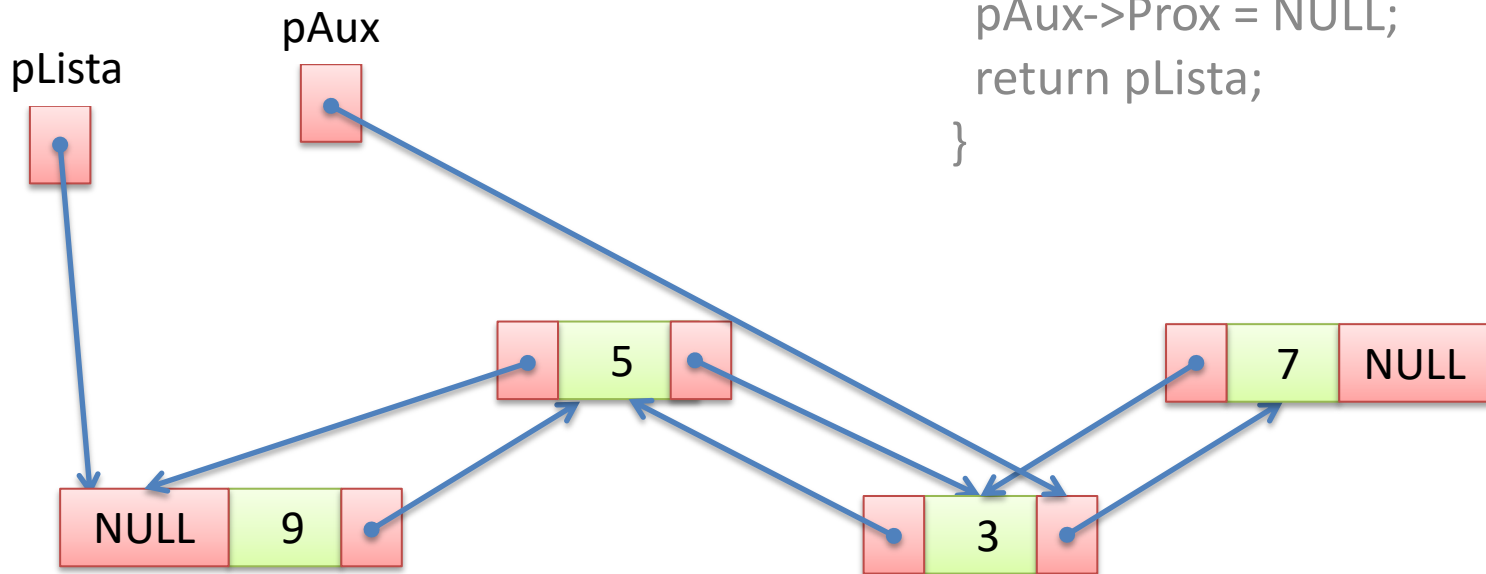
Lista Duplamente Encadeada Exclusão na Calda

```
TNo *ExcluiCalda(TNo *pLista)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Prox->Prox != NULL)
        pAux = pAux->Prox;
    free(pAux->Prox);
    pAux->Prox = NULL;
    return pLista;
}
```



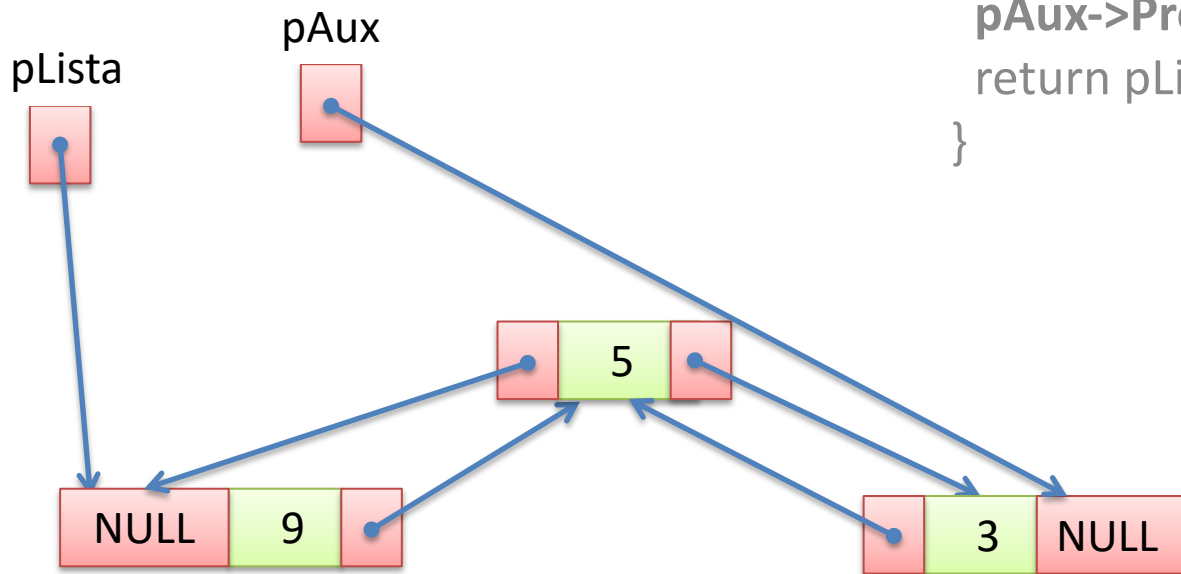
Lista Duplamente Encadeada Exclusão na Calda

```
TNo *ExcluiCalda(TNo *pLista)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Prox->Prox != NULL)
        pAux = pAux->Prox;
    free(pAux->Prox);
    pAux->Prox = NULL;
    return pLista;
}
```



Lista Duplamente Encadeada Exclusão na Calda

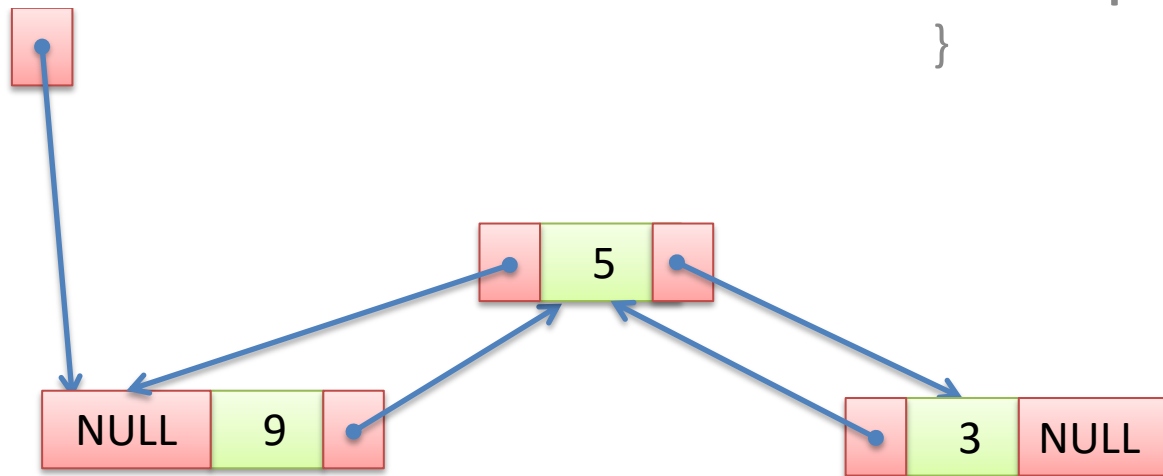
```
TNo *ExcluiCalda(TNo *pLista)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Prox->Prox != NULL)
        pAux = pAux->Prox;
    free(pAux->Prox);
    pAux->Prox = NULL;
    return pLista;
}
```



Lista Duplamente Encadeada Exclusão na Calda

```
TNo *ExcluiCalda(TNo *pLista)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Prox->Prox != NULL)
        pAux = pAux->Prox;
    free(pAux->Prox);
    pAux->Prox = NULL;
    return pLista;
}
```

pLista



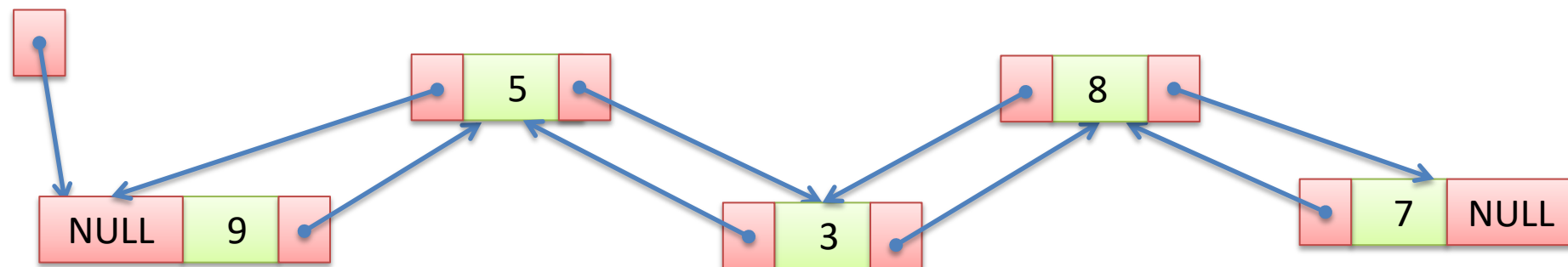
Lista Duplamente Encadeada

Exclusão de Uma Chave

pChave = 3

```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Numero != pChave)
        pAux = pAux->Prox;
    pAux->Ant->Prox = pAux->Prox;
    pAux->Prox->Ant = pAux->Ant;
    free(pAux);
    return pLista;
}
```

pLista

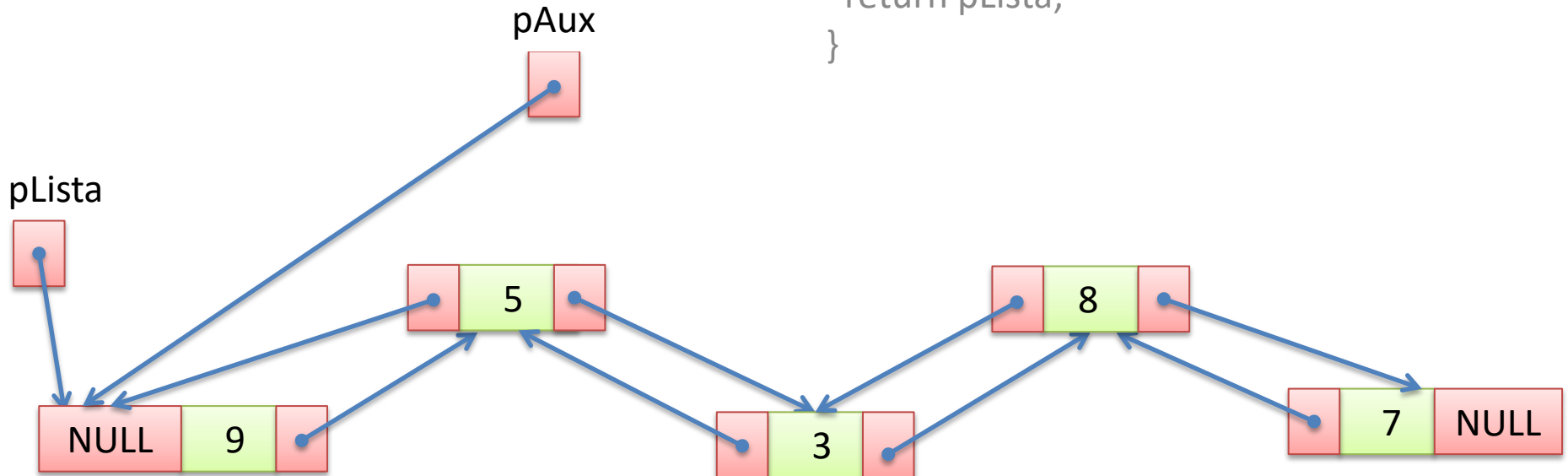


Lista Duplamente Encadeada

Exclusão de Uma Chave

pChave = 3

```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Numero != pChave)
        pAux = pAux->Prox;
    pAux->Ant->Prox = pAux->Prox;
    pAux->Prox->Ant = pAux->Ant;
    free(pAux);
    return pLista;
}
```

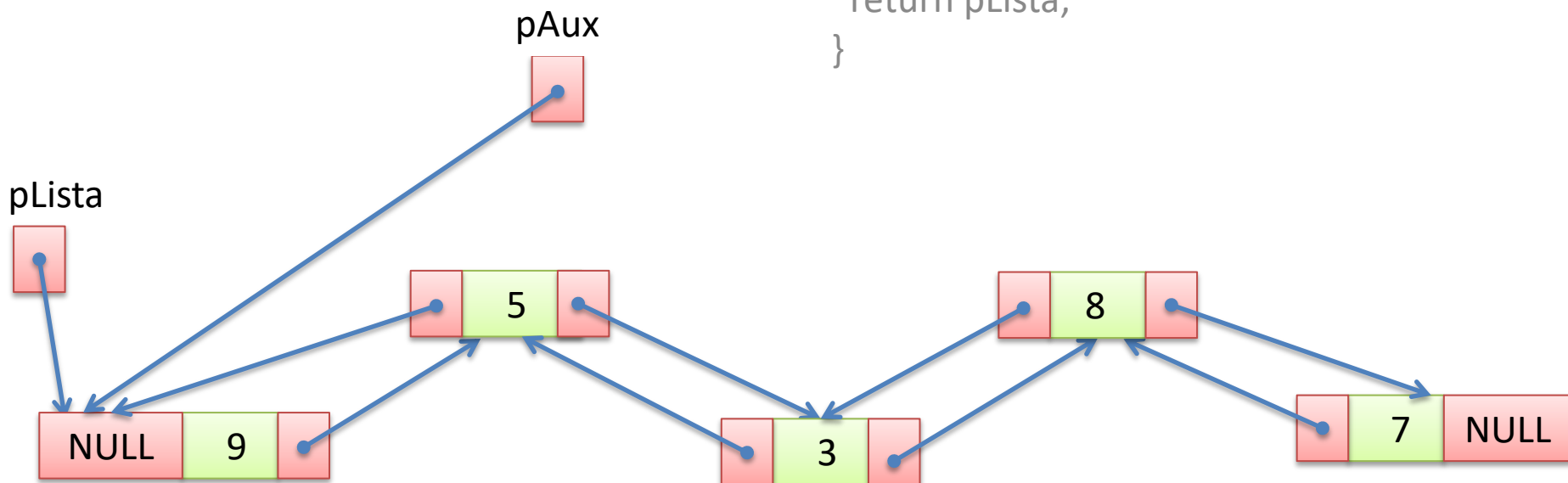


Lista Duplamente Encadeada

Exclusão de Uma Chave

pChave = 3

```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Numero != pChave)
        pAux = pAux->Prox;
    pAux->Ant->Prox = pAux->Prox;
    pAux->Prox->Ant = pAux->Ant;
    free(pAux);
    return pLista;
}
```

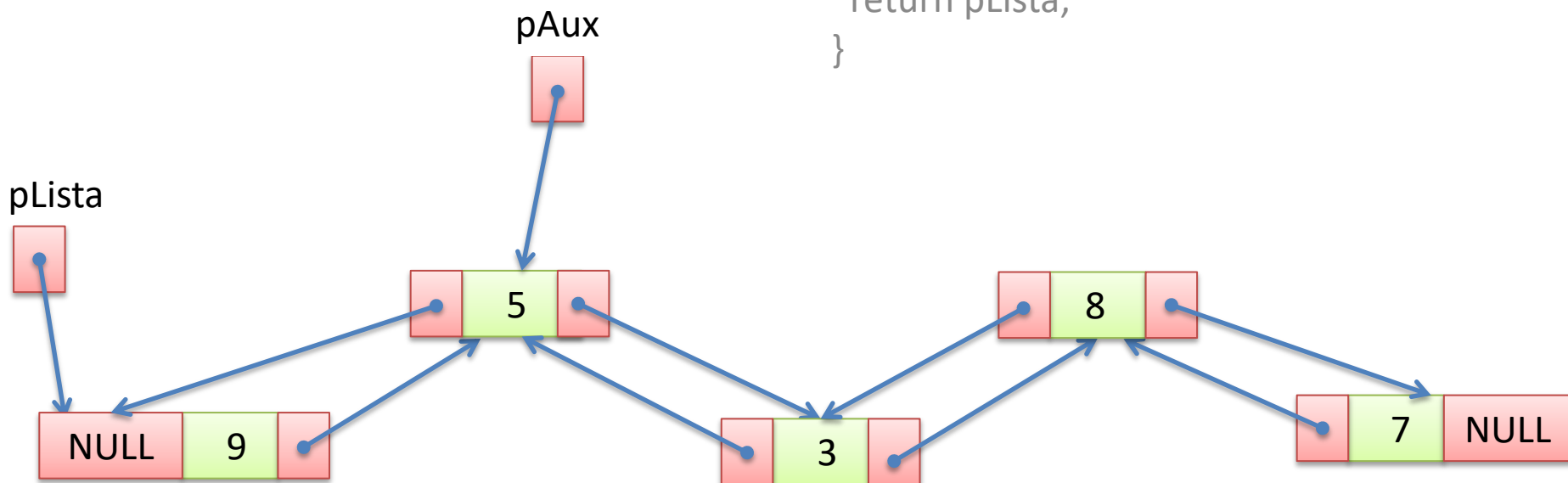


Lista Duplamente Encadeada

Exclusão de Uma Chave

pChave = 3

```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Numero != pChave)
        pAux = pAux->Prox;
    pAux->Ant->Prox = pAux->Prox;
    pAux->Prox->Ant = pAux->Ant;
    free(pAux);
    return pLista;
}
```

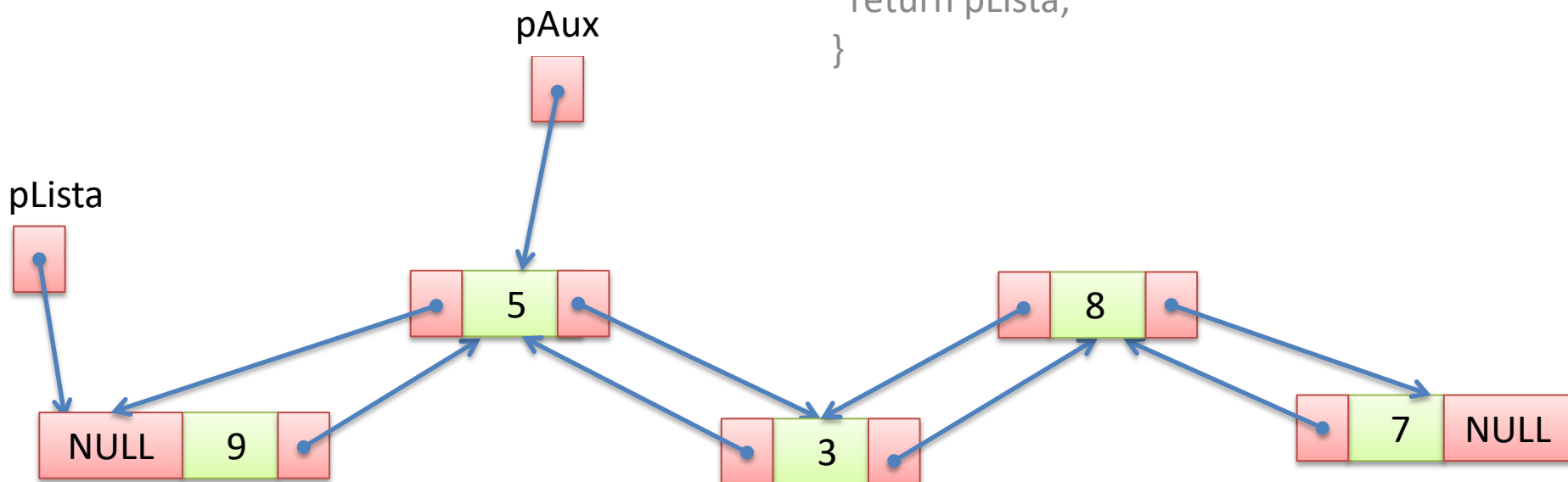


Lista Duplamente Encadeada

Exclusão de Uma Chave

pChave = 3

```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Numero != pChave)
        pAux = pAux->Prox;
    pAux->Ant->Prox = pAux->Prox;
    pAux->Prox->Ant = pAux->Ant;
    free(pAux);
    return pLista;
}
```

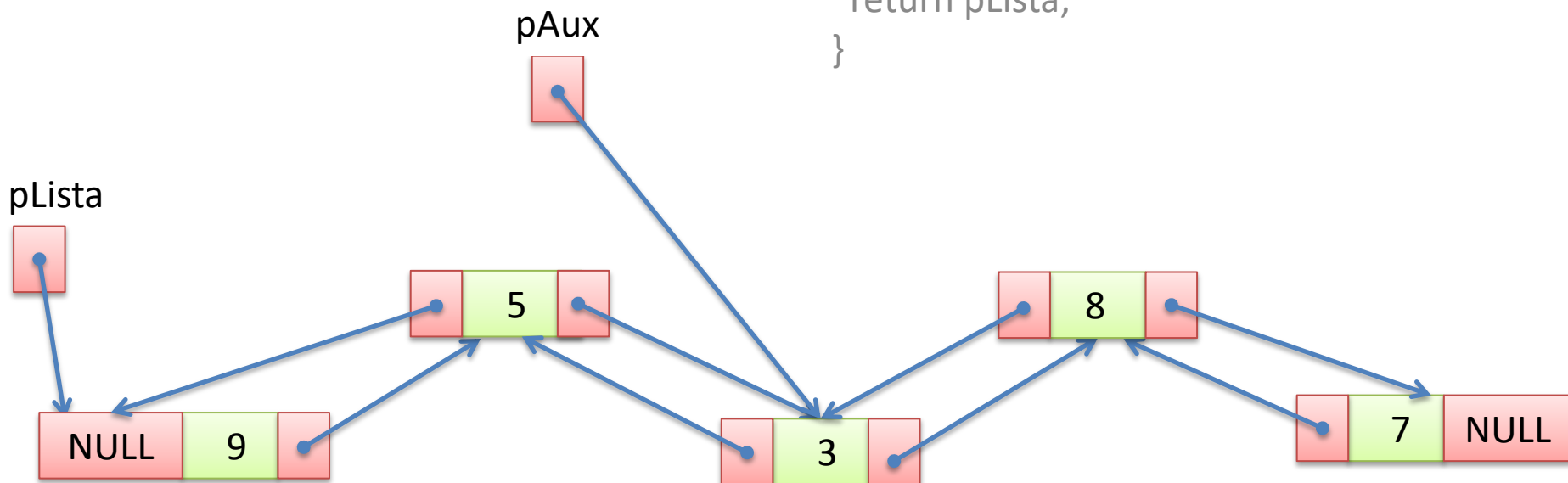


Lista Duplamente Encadeada

Exclusão de Uma Chave

pChave = 3

```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Numero != pChave)
        pAux = pAux->Prox;
    pAux->Ant->Prox = pAux->Prox;
    pAux->Prox->Ant = pAux->Ant;
    free(pAux);
    return pLista;
}
```

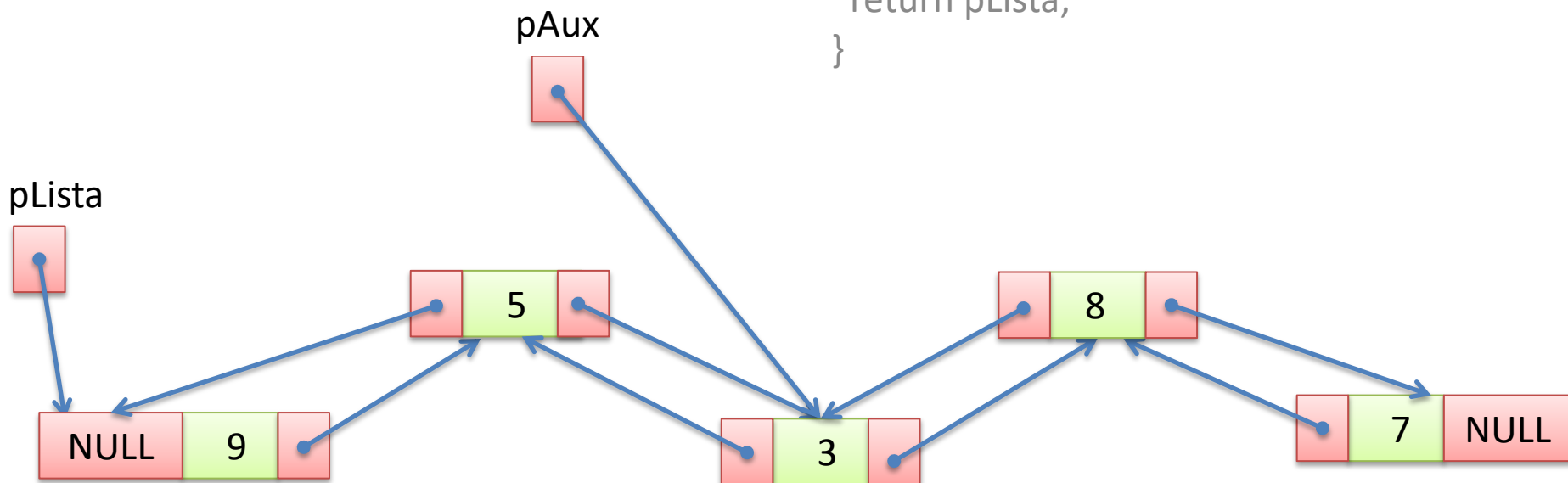


Lista Duplamente Encadeada

Exclusão de Uma Chave

pChave = 3

```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Numero != pChave)
        pAux = pAux->Prox;
    pAux->Ant->Prox = pAux->Prox;
    pAux->Prox->Ant = pAux->Ant;
    free(pAux);
    return pLista;
}
```

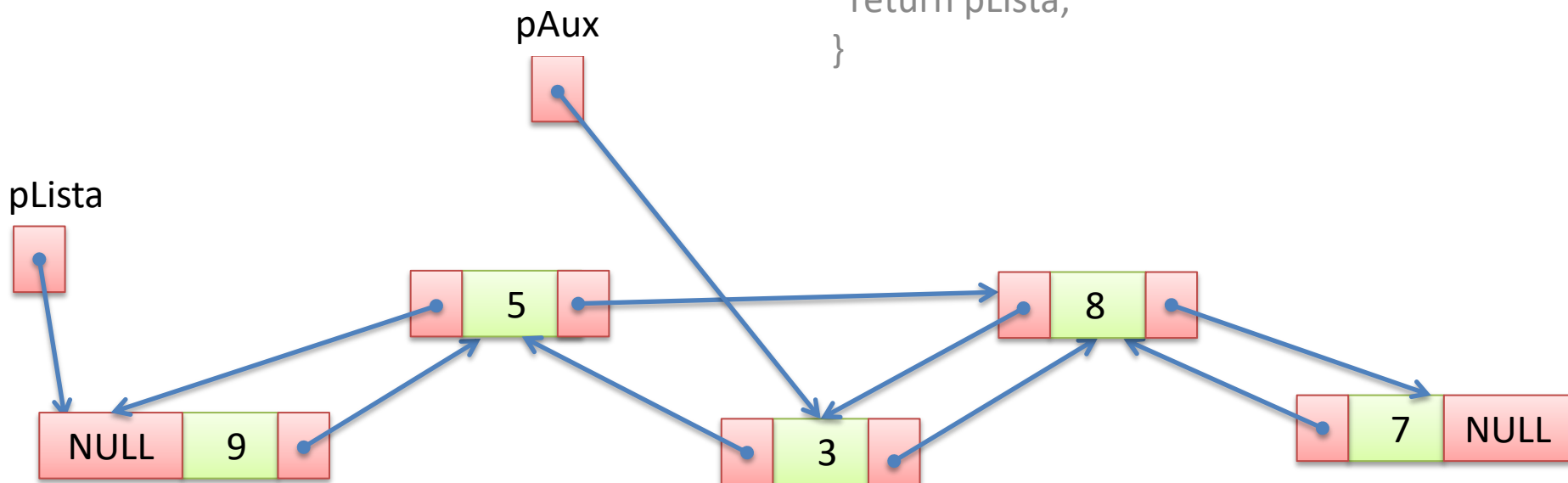


Lista Duplamente Encadeada

Exclusão de Uma Chave

pChave = 3

```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Numero != pChave)
        pAux = pAux->Prox;
    pAux->Ant->Prox = pAux->Prox;
    pAux->Prox->Ant = pAux->Ant;
    free(pAux);
    return pLista;
}
```

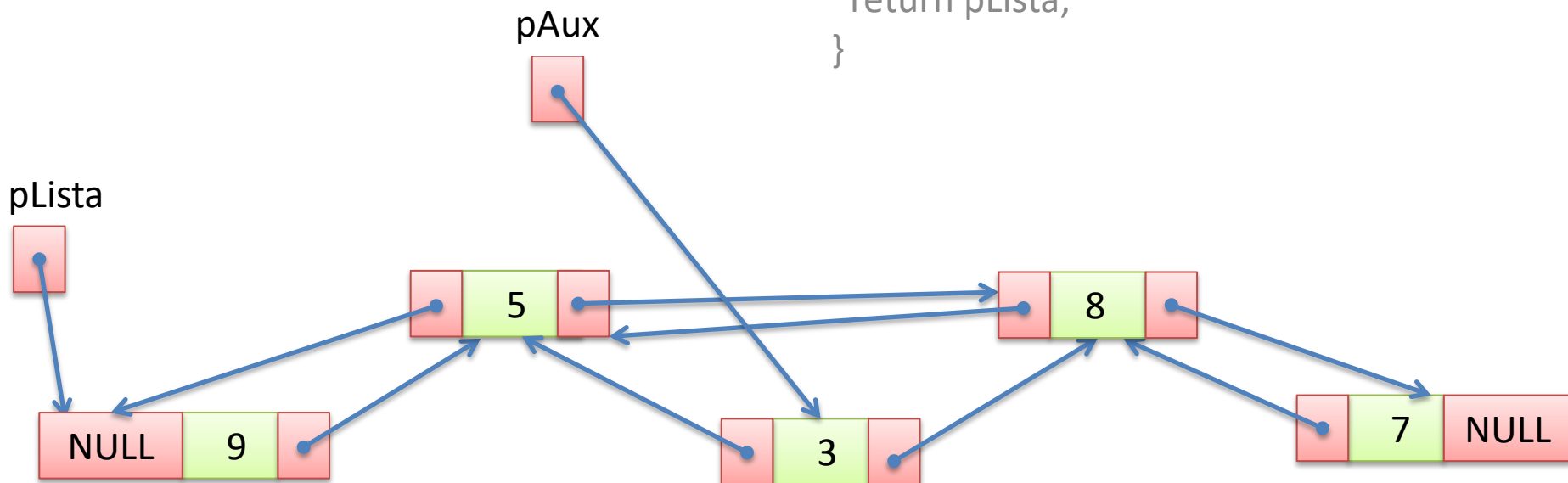


Lista Duplamente Encadeada

Exclusão de Uma Chave

pChave = 3

```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Numero != pChave)
        pAux = pAux->Prox;
    pAux->Ant->Prox = pAux->Prox;
    pAux->Prox->Ant = pAux->Ant;
    free(pAux);
    return pLista;
}
```

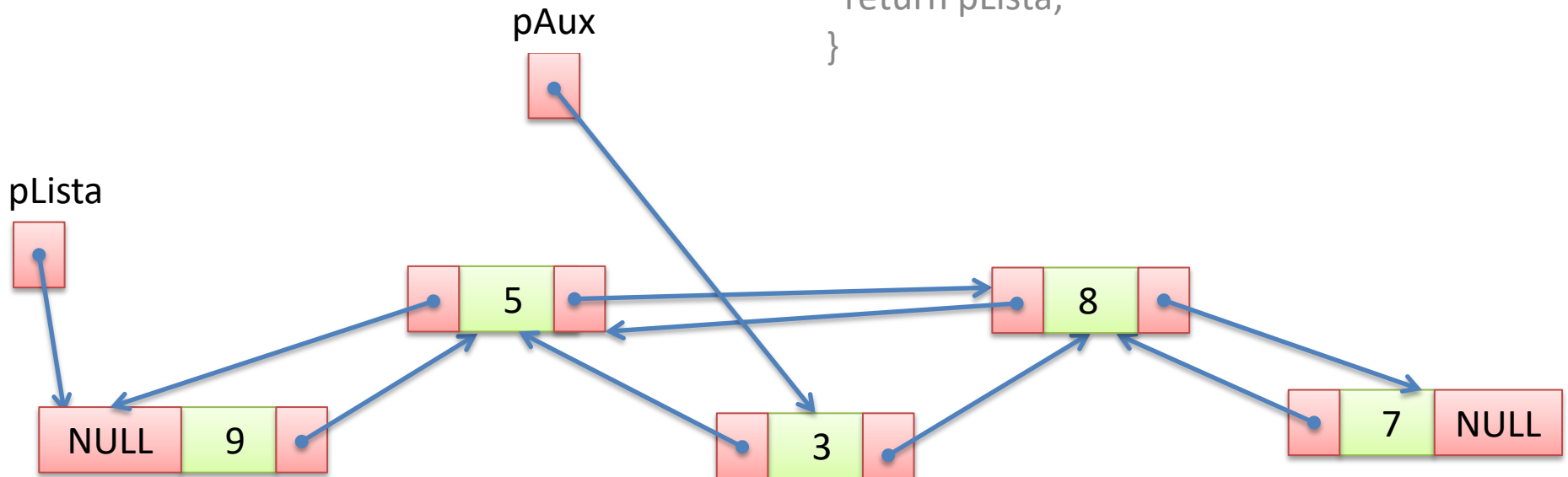


Lista Duplamente Encadeada

Exclusão de Uma Chave

pChave = 3

```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Numero != pChave)
        pAux = pAux->Prox;
    pAux->Ant->Prox = pAux->Prox;
    pAux->Prox->Ant = pAux->Ant;
    free(pAux);
    return pLista;
}
```

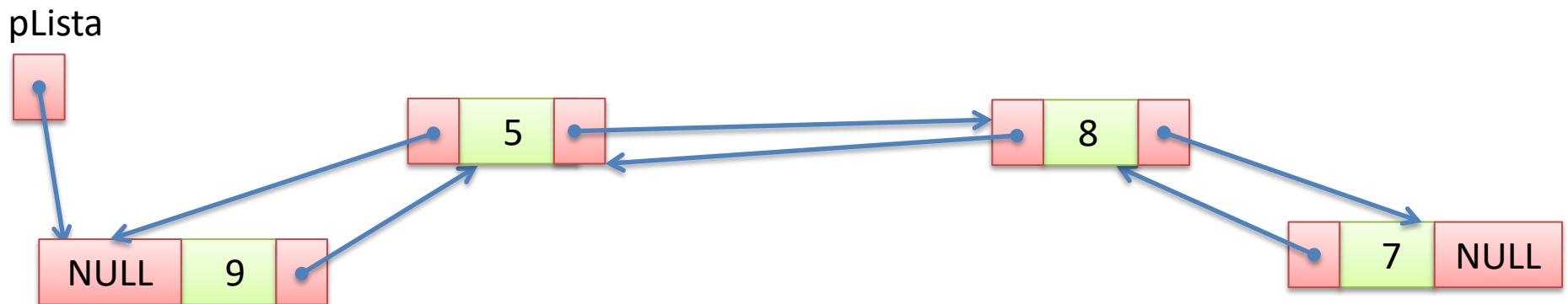


Lista Duplamente Encadeada

Exclusão de Uma Chave

pChave = 3

```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
    TNo *pAux;
    pAux = pLista;
    while (pAux->Numero != pChave)
        pAux = pAux->Prox;
    pAux->Ant->Prox = pAux->Prox;
    pAux->Prox->Ant = pAux->Ant;
    free(pAux);
    return pLista;
}
```



struct TNo

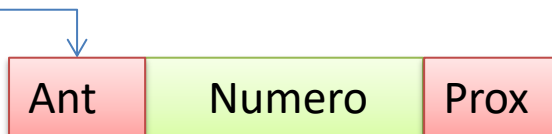
{

TNo *Ant;

int Numero;

TNo *Prox;

};



struct TLista

{

TNo *Primeiro;

int Qtde;

TNo *Ultimo;

};



Lista Duplamente Encadeada

