Estrutura de Dados e Algoritmos

Listas Encadeadas

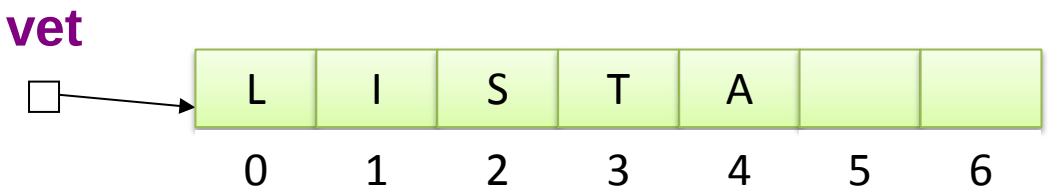
Prof. Nilton Correia da Silva

Listas

- Uma lista é uma representação de uma sequência de objetos na memória do computador;
- Cada elemento da lista é armazenado em um nó ou célula: o primeiro elemento na primeira célula, o segundo na segunda e assim por diante.

Listas Sequenciais

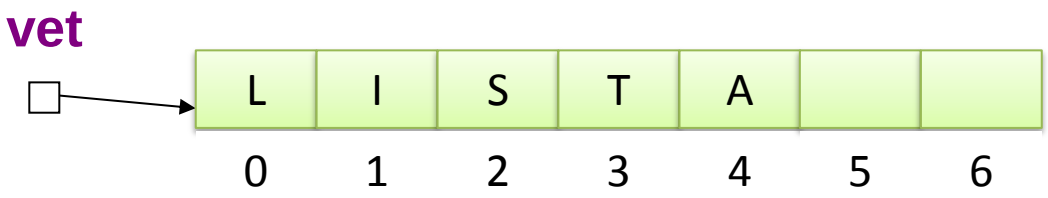
- Conjunto de itens organizados (vetor)
  - A organização é implícita (pela posição):



- vet representa a lista → A lista está armazenada em células contíguas da memória.

Listas Sequenciais

- Desvantagens: Quantidade de Nós pré-definida:
  - memória alocada sem uso;
  - impossibilidade de alocar mais memória.



- Solução:
  - Utilizar Estruturas de Dados que cresçam e diminuam na medida da necessidade → Estruturas Dinâmicas;
  - Alocação dinâmica de memória conforme demanda.

Listas Encadeadas

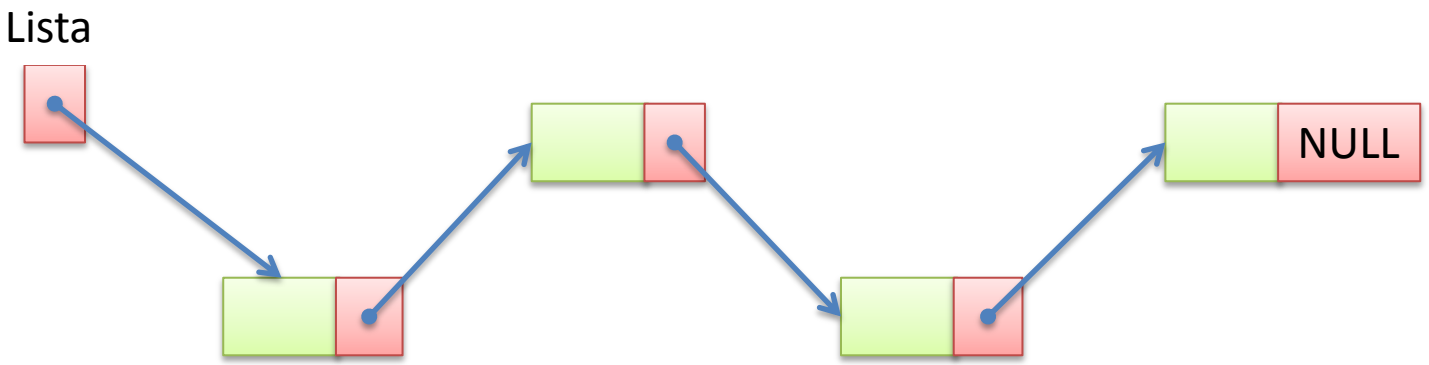
Estrutura de Uma Lista Encadeada

- Podem crescer e diminuir dinamicamente;
- Tamanho máximo não precisa ser definido previamente;
- Provêem flexibilidade, permitindo que os itens sejam rearrumados eficientemente;
- Também chamadas de Listas Ligadas.

- O acesso a um nó e feito de forma sequencial → O(n). Diferente de vetor → O(1);

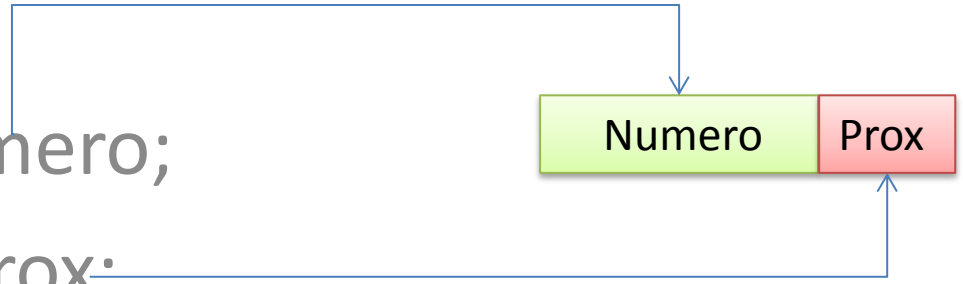
Estrutura de Uma Lista Encadeada

- Uma lista encadeada é uma sequência de nós não contíguos;
- Cada nó contém um objeto de algum tipo e, no mínimo, o endereço do nó seguinte;
- Uma lista encadeada deve ter um ponteiro para o início da lista (primeiro nó);



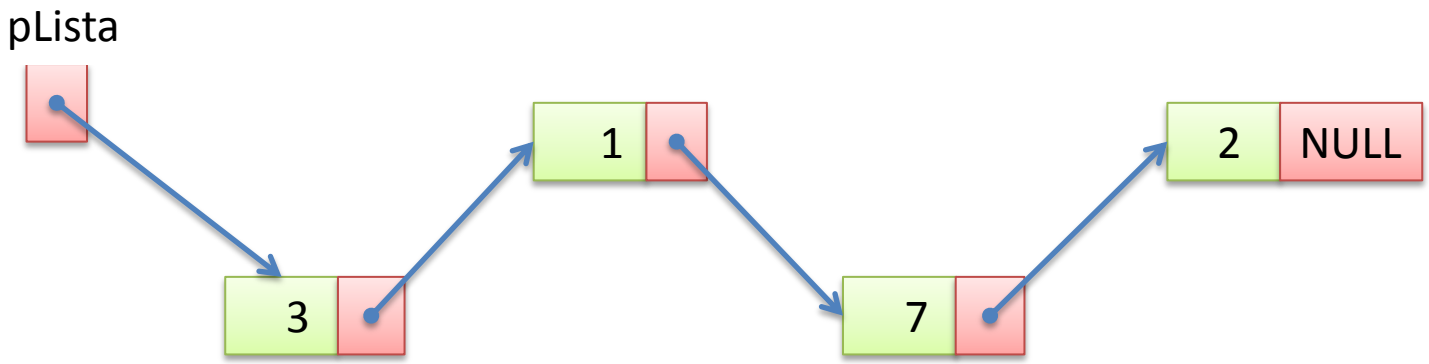
Lista Encadeada – Inclusão na Cabeça

```
struct TNo
{
  int Numero;
  Tno *Prox;
};
```



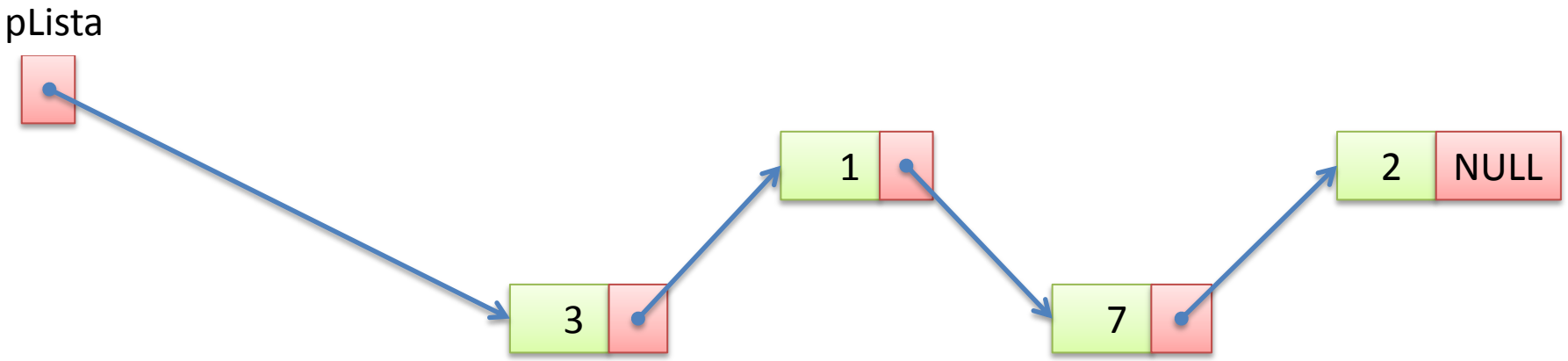
Lista Encadeada – Inclusão na Cabeça

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



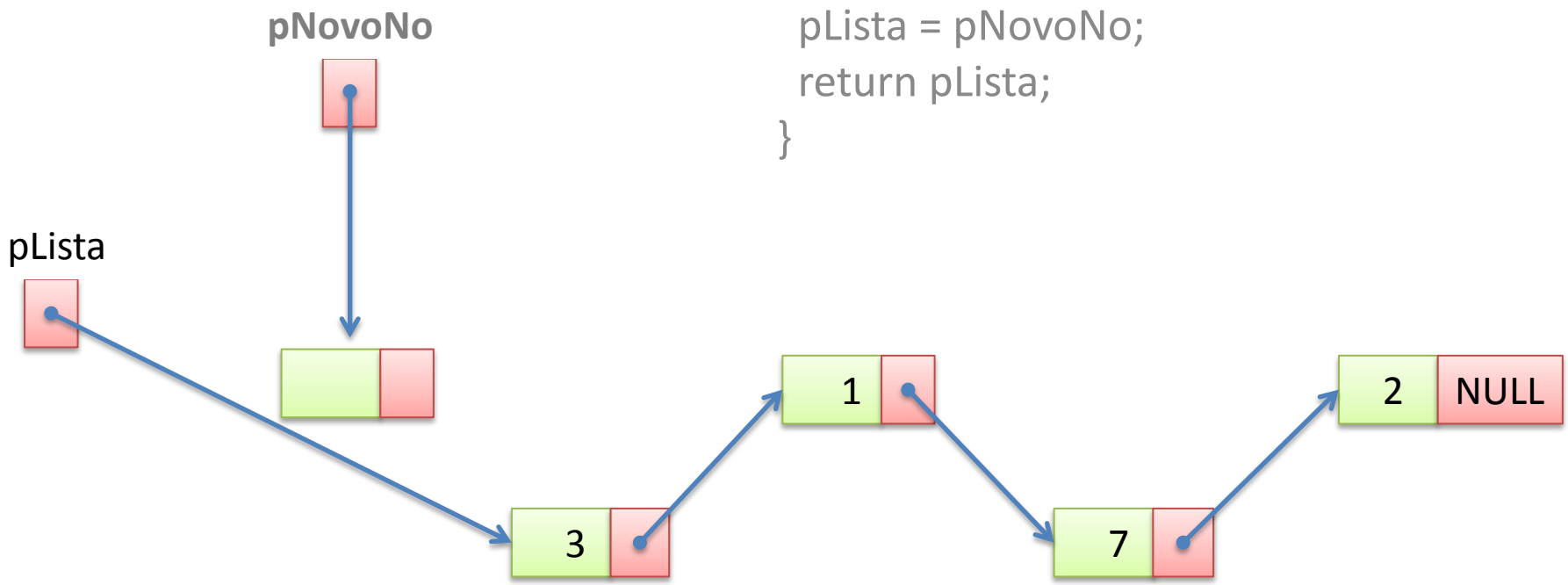
Lista Encadeada – Inclusão na Cabeça

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



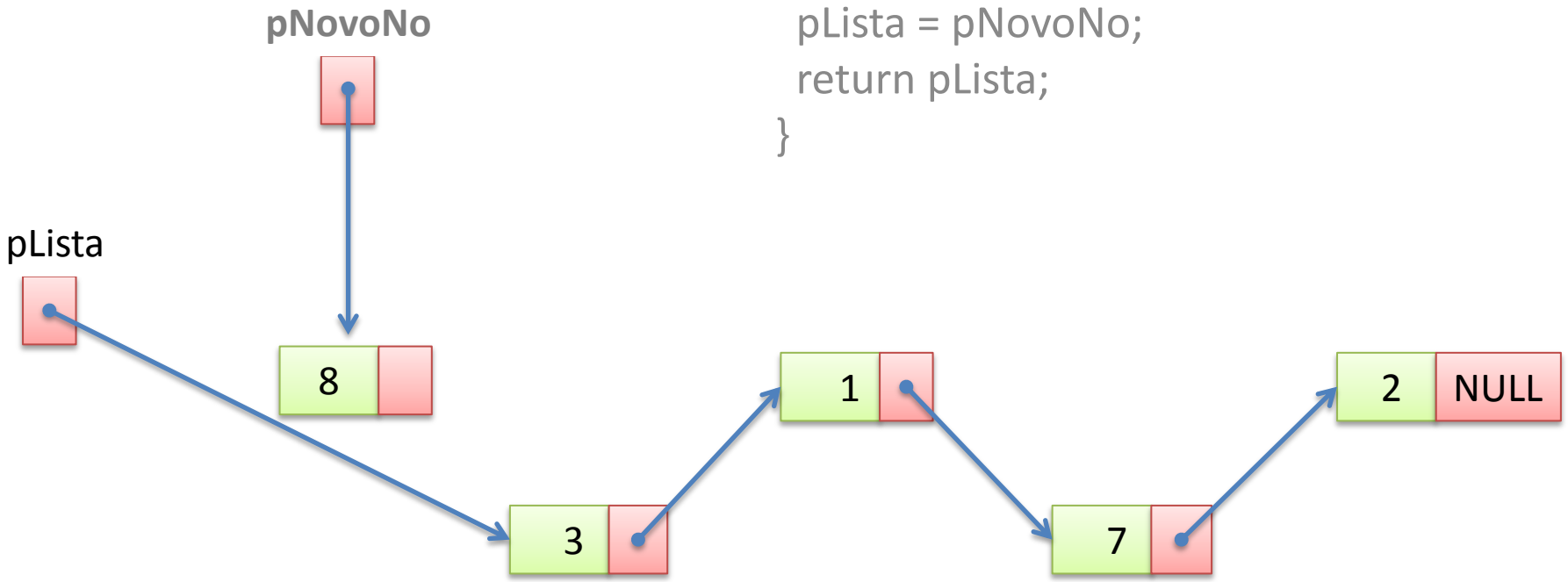
Lista Encadeada – Inclusão na Cabeça

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



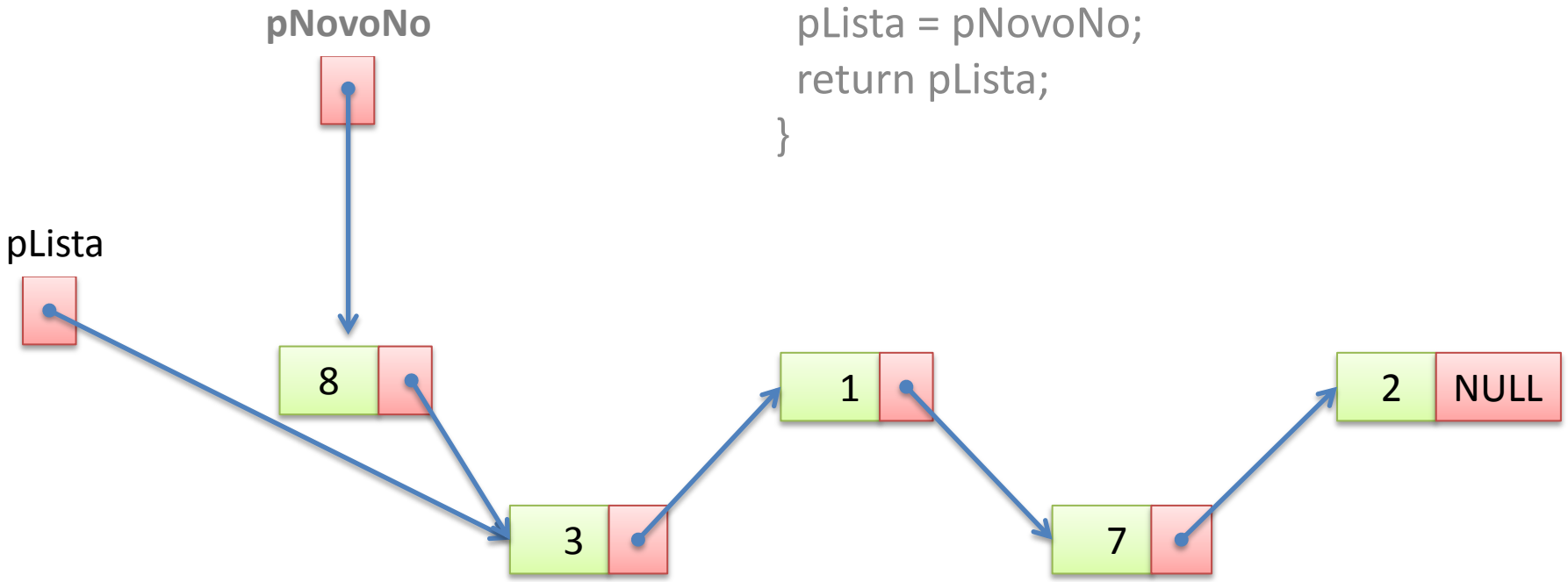
Lista Encadeada – Inclusão na Cabeça

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



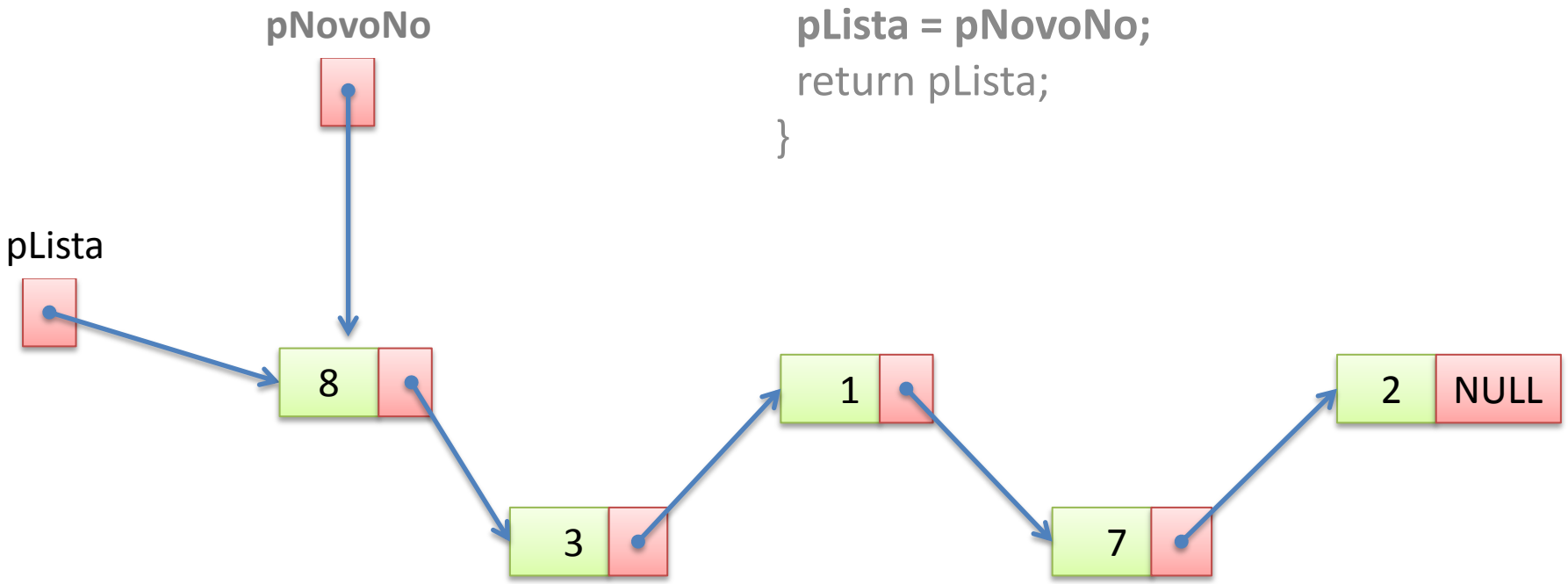
Lista Encadeada – Inclusão na Cabeça

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



Lista Encadeada – Inclusão na Cabeça

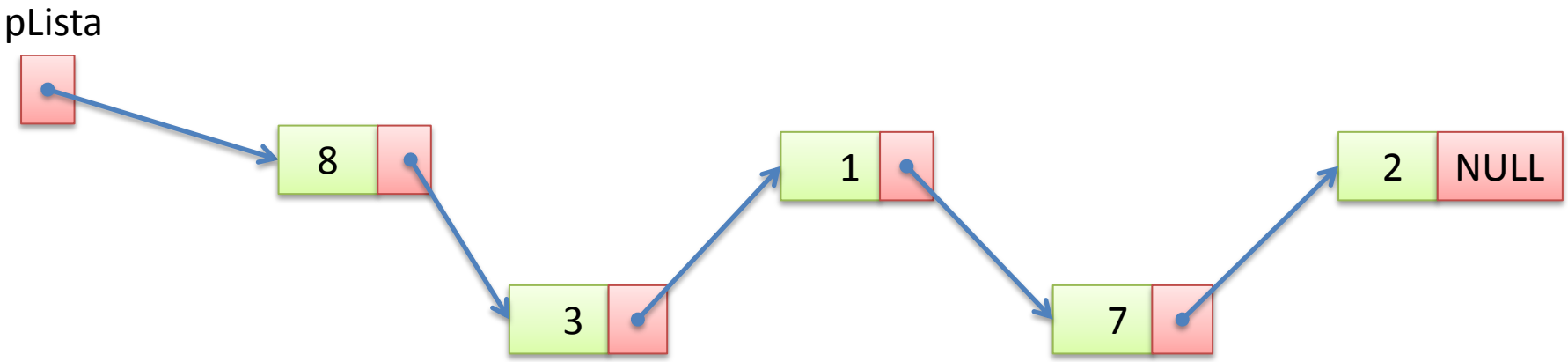
```
TNo *IncluiCabeca(TNo *pLista, int pValor)
{
  TNo *pNovoNo;
  pNovoNo = (TNo *) malloc(sizeof(TNo));
  pNovoNo->Numero = pValor;
  pNovoNo->Prox = pLista;
  pLista = pNovoNo;
  return pLista;
}
```



Lista Encadeada – Inclusão na Cabeça

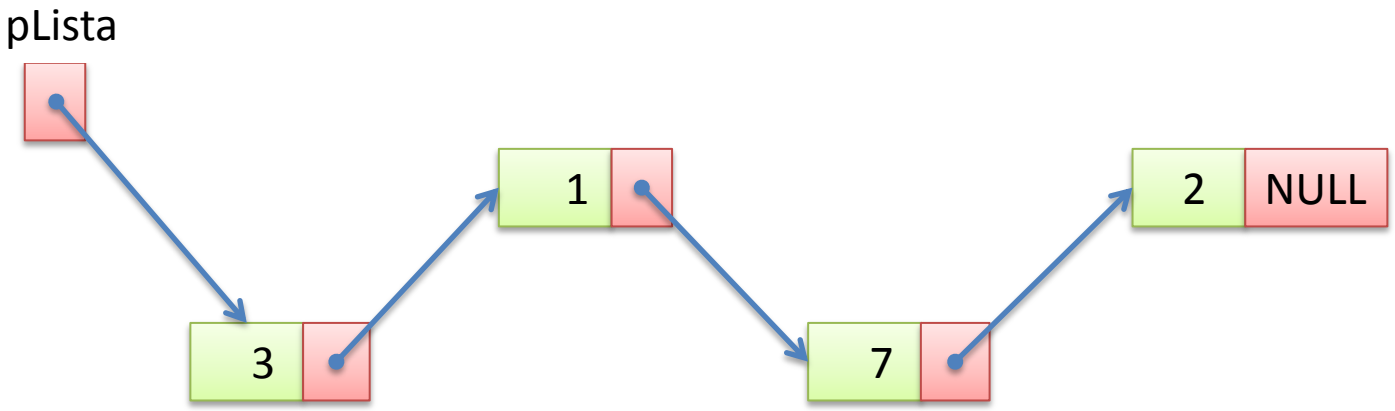
O(1)

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



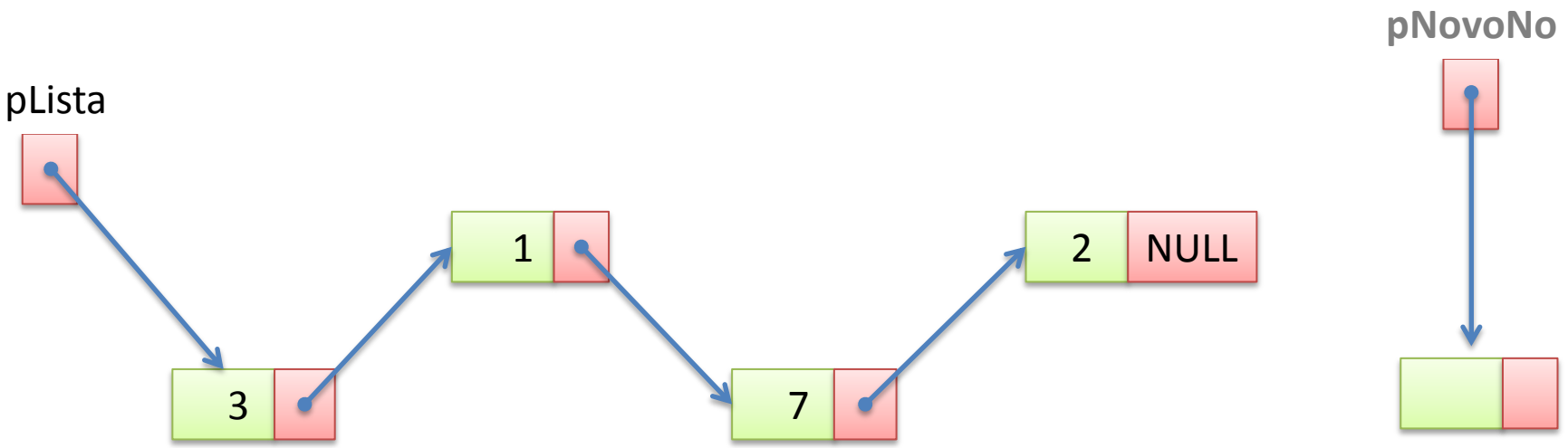
Lista Encadeada – Inclusão na Calda

```
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;
  return pLista;
}
```



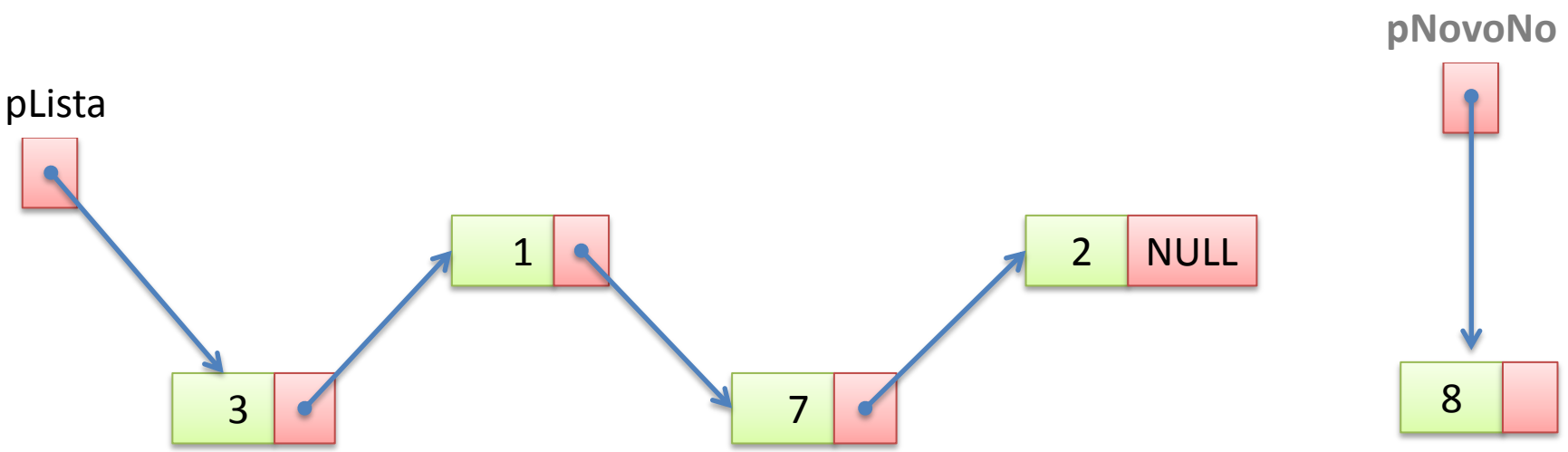
Lista Encadeada – Inclusão na Calda

```
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;
  return pLista;
}
```



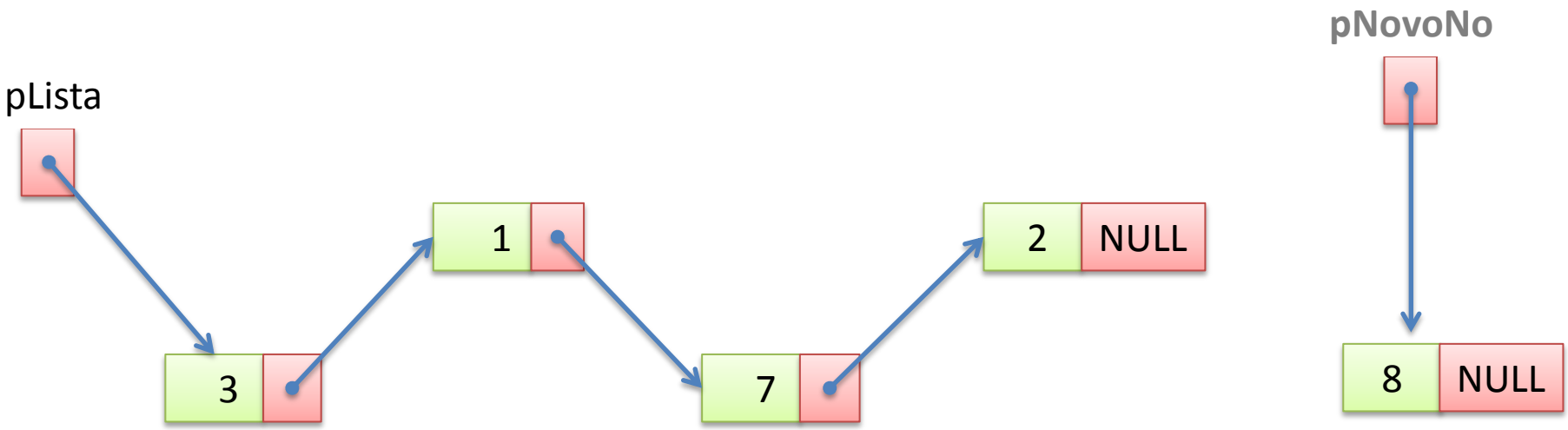
Lista Encadeada – Inclusão na Calda

```
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;
  return pLista;
}
```



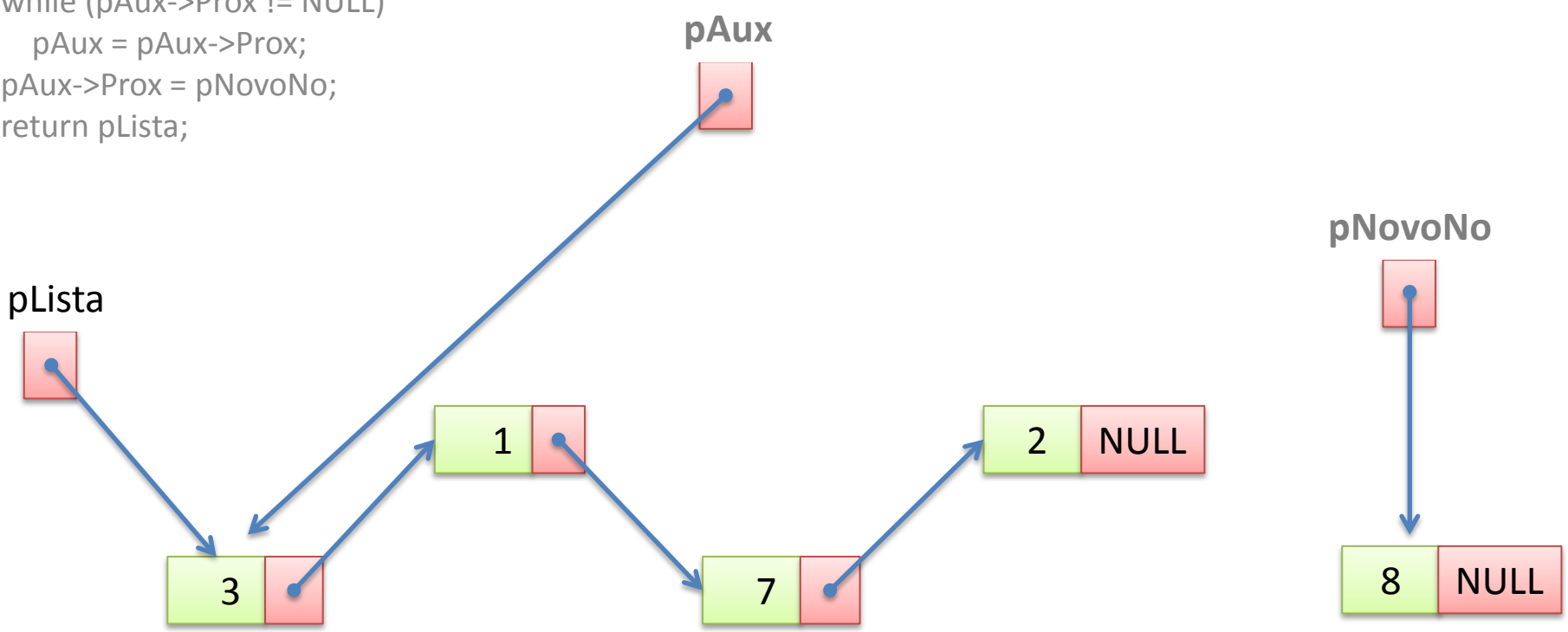
Lista Encadeada – Inclusão na Calda

```
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;
  return pLista;
}
```



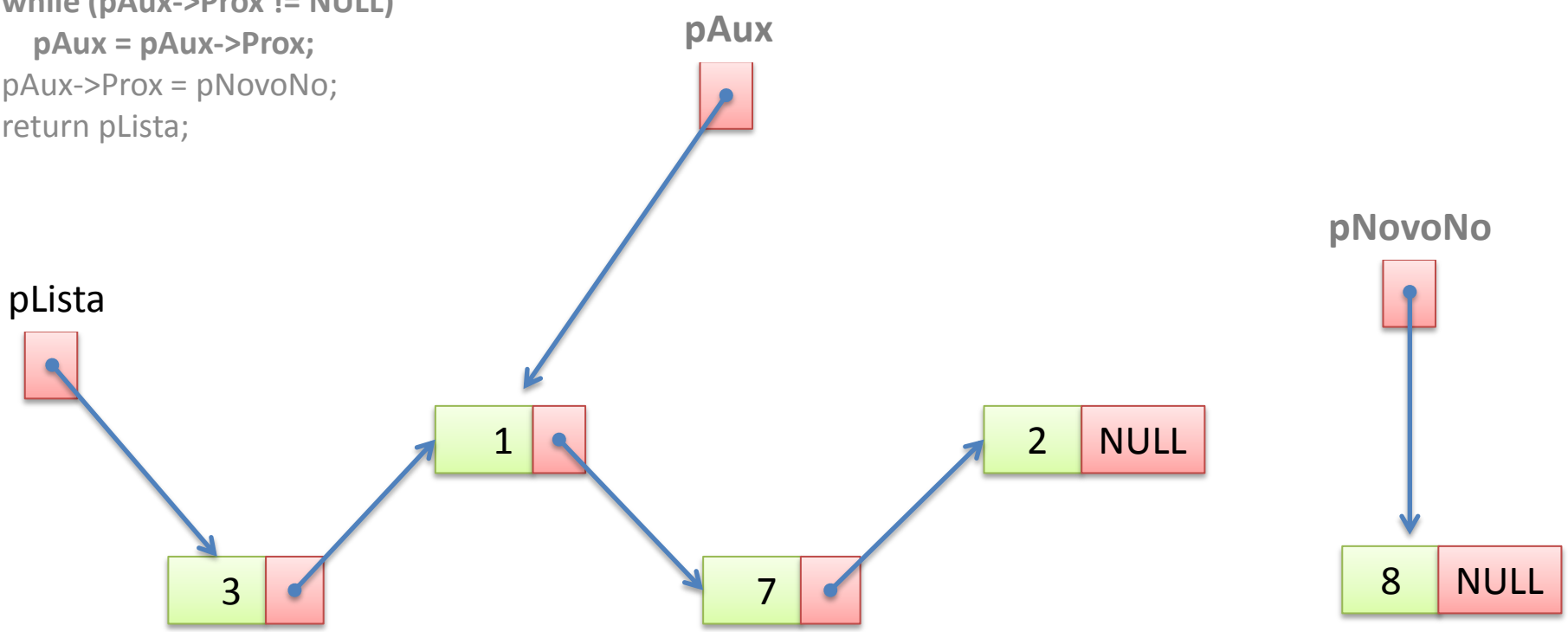
Lista Encadeada – Inclusão na Calda

```
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;
  return pLista;
}
```



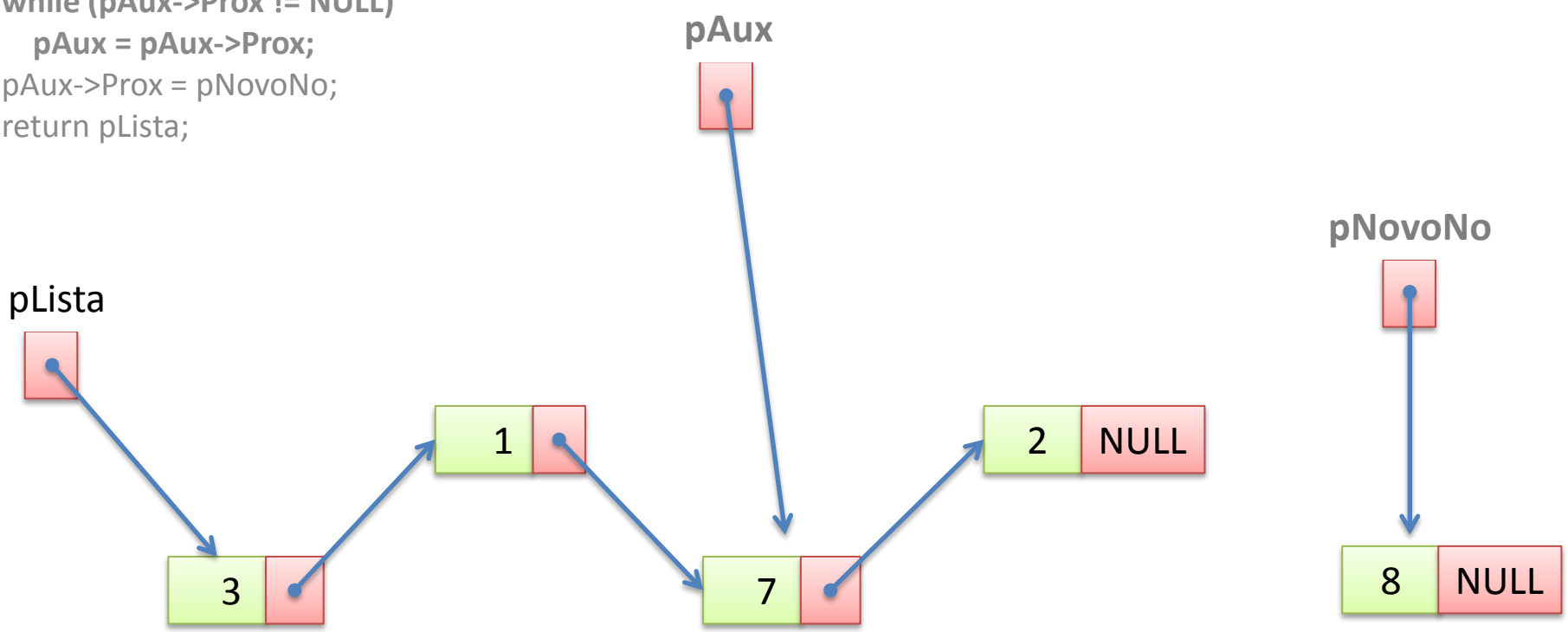
Lista Encadeada – Inclusão na Calda

```
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;
  return pLista;
}
```



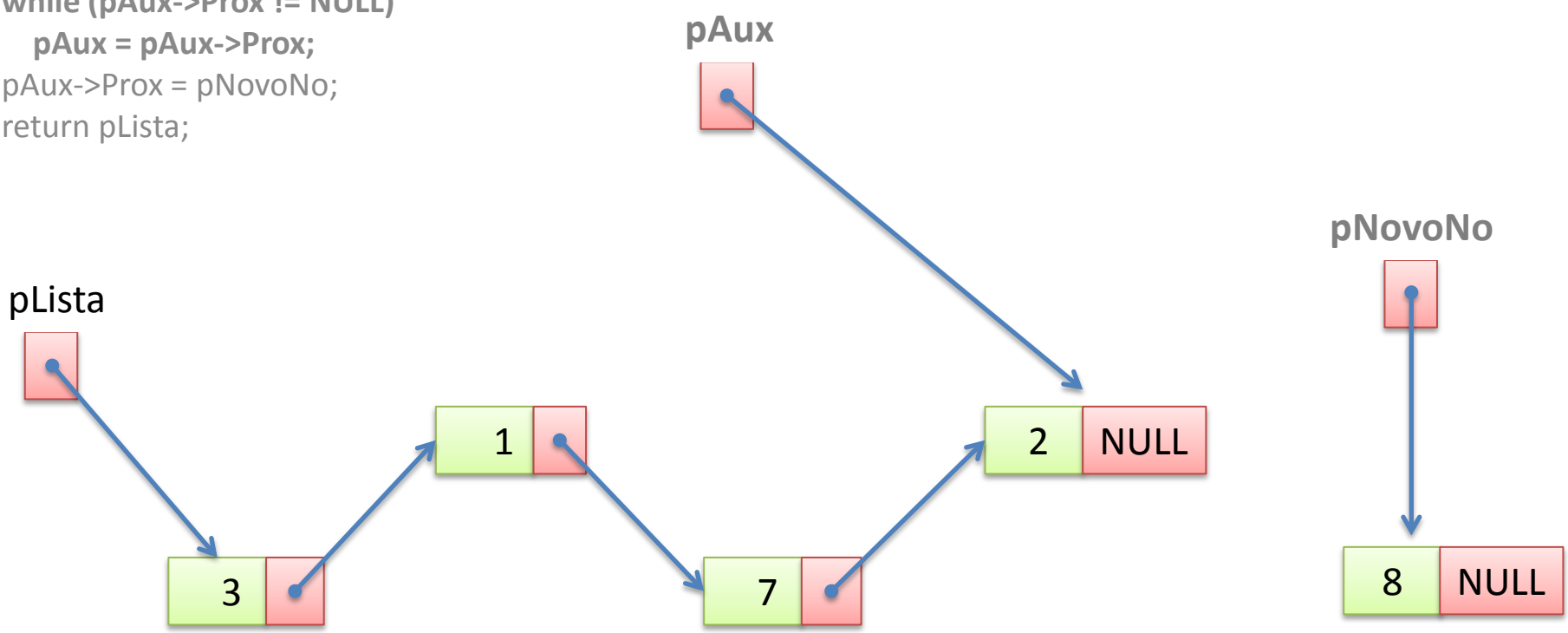
Lista Encadeada – Inclusão na Calda

```
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;
  return pLista;
}
```



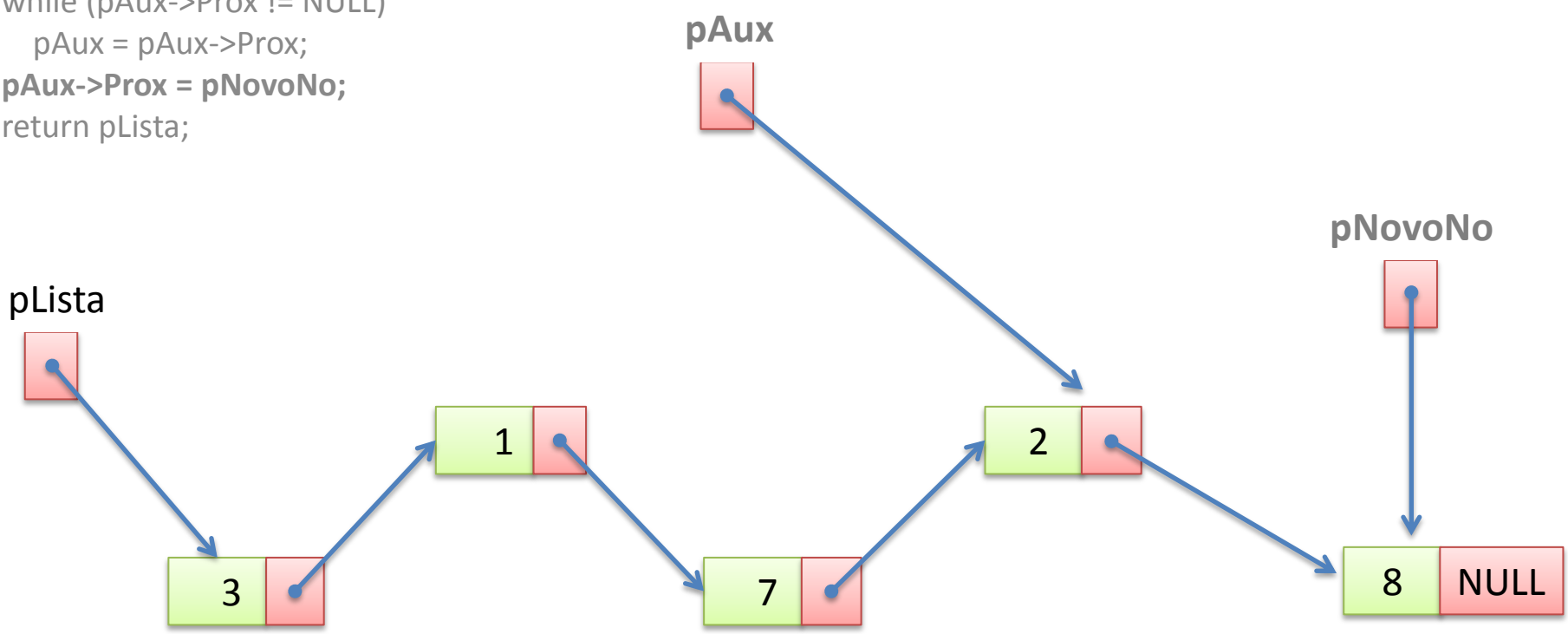
Lista Encadeada – Inclusão na Calda

```
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;
  return pLista;
}
```



Lista Encadeada – Inclusão na Calda

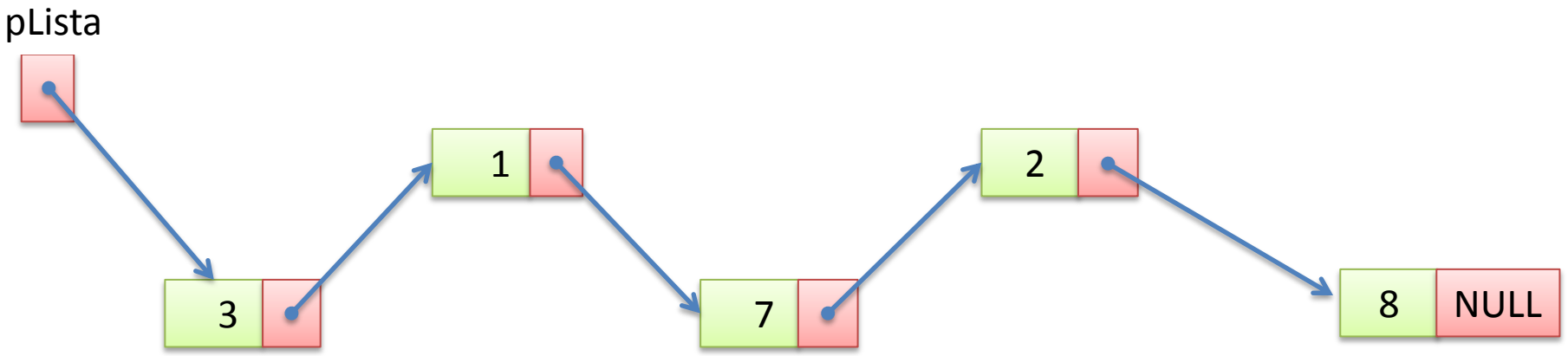
```
TNo *IncluiCalda(TNo *pLista, int pValor)
{
  TNo *pNovoNo, *pAux;
  pNovoNo = (TNo *) malloc(sizeof(TNo));
  pNovoNo->Numero = pValor;
  pNovoNo->Prox = NULL;
  pAux = pLista;
  while (pAux->Prox != NULL)
     pAux = pAux->Prox;
  pAux->Prox = pNovoNo;
  return pLista;
}
```



Lista 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;
  return pLista;
}
```

O(n)

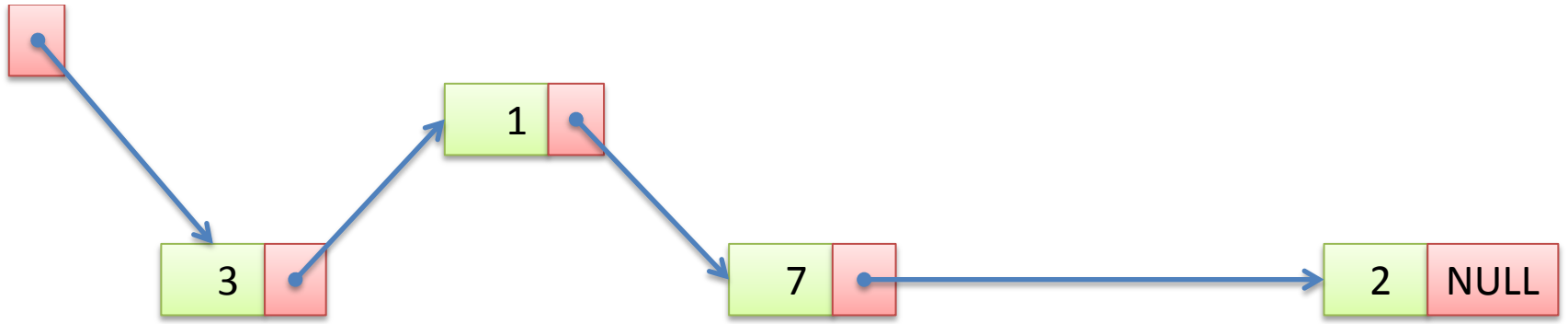


Lista Encadeada – Inclusão Antes de Uma Chave

```
TNo *IncluiAntes(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;
  pAux->Prox = pNovoNo;
  return pLista;
}
```

pChave = 2

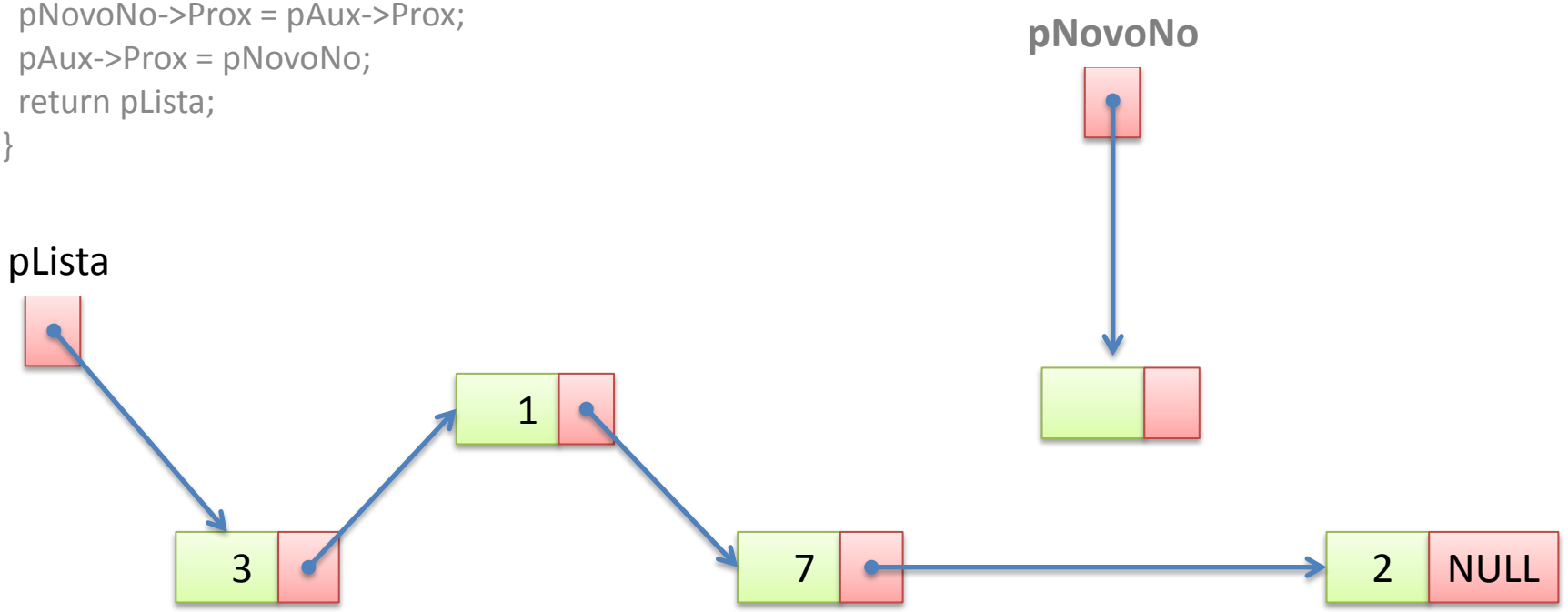
pLista



Lista Encadeada – Inclusão Antes de Uma Chave

```
TNo *IncluiAntes(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;
  pAux->Prox = pNovoNo;
  return pLista;
}
```

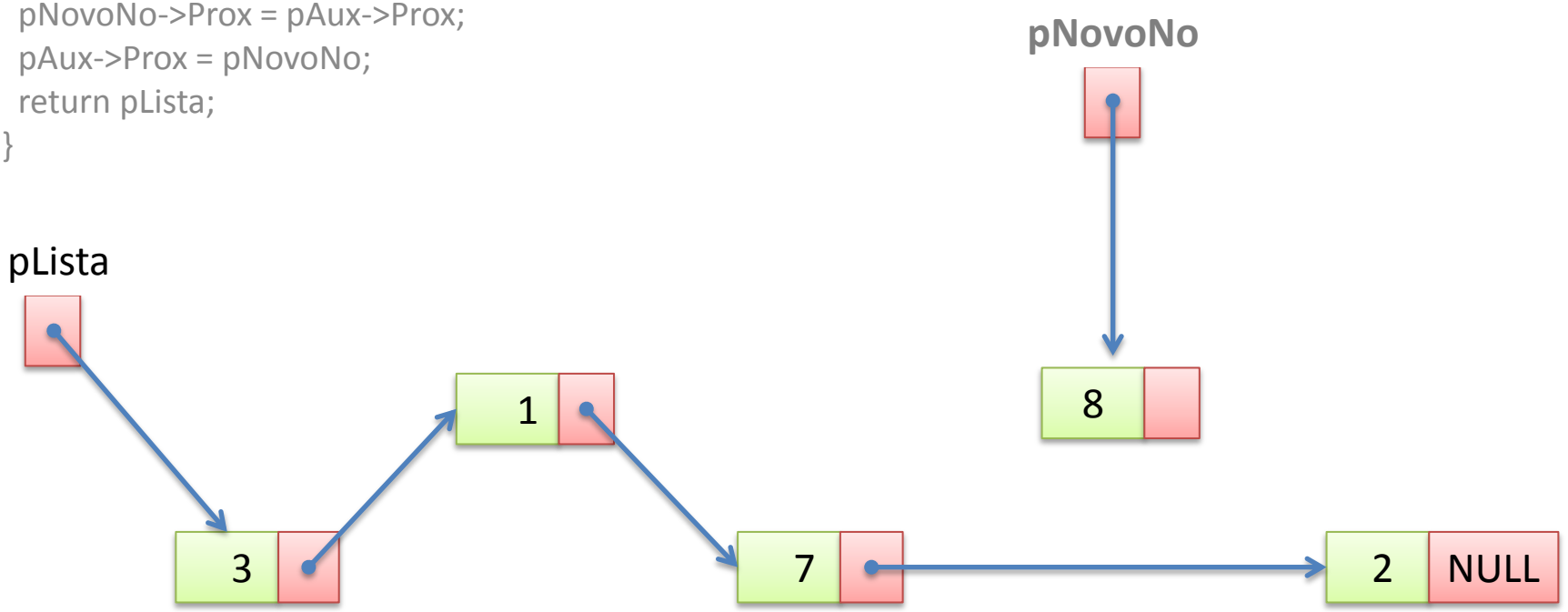
pChave = 2



Lista Encadeada – Inclusão Antes de Uma Chave

```
TNo *IncluiAntes(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;
  pAux->Prox = pNovoNo;
  return pLista;
}
```

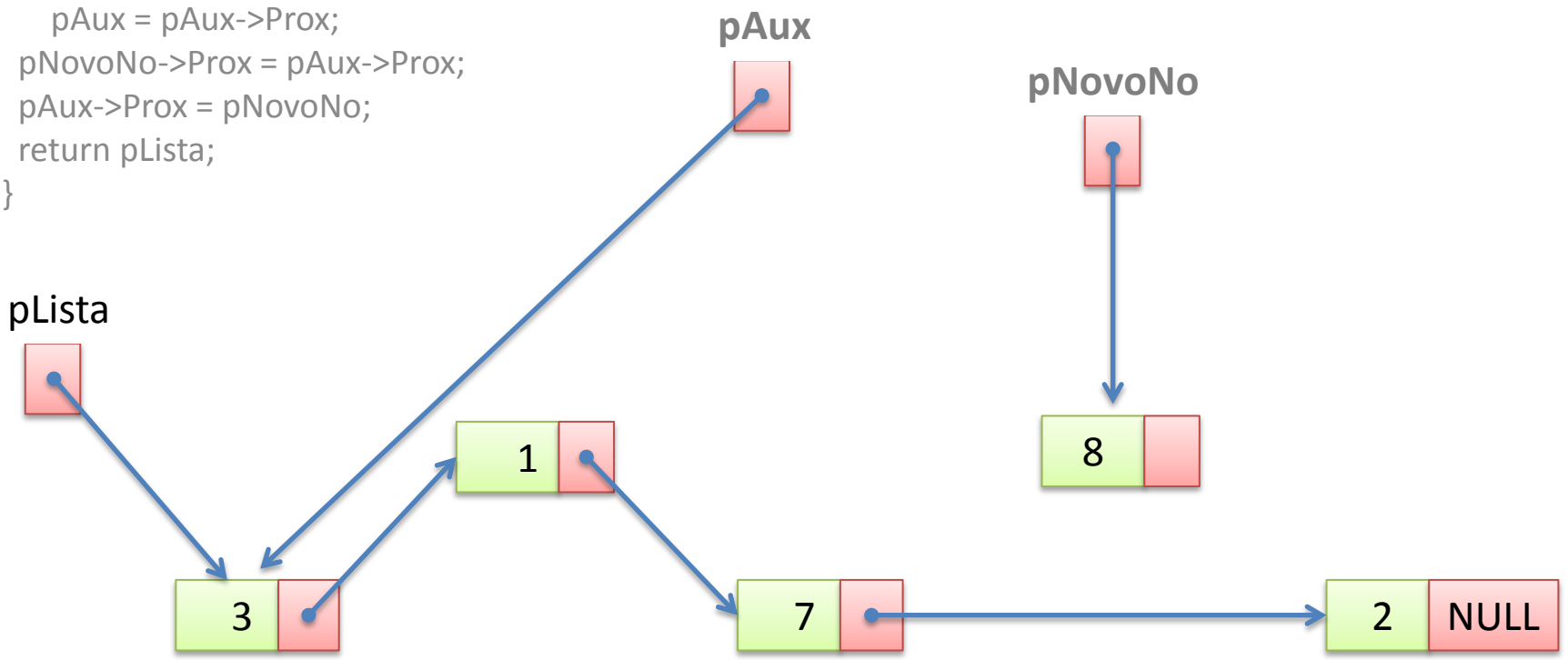
pChave = 2



Lista Encadeada – Inclusão Antes de Uma Chave

```
TNo *IncluiAntes(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;
  pAux->Prox = pNovoNo;
  return pLista;
}
```

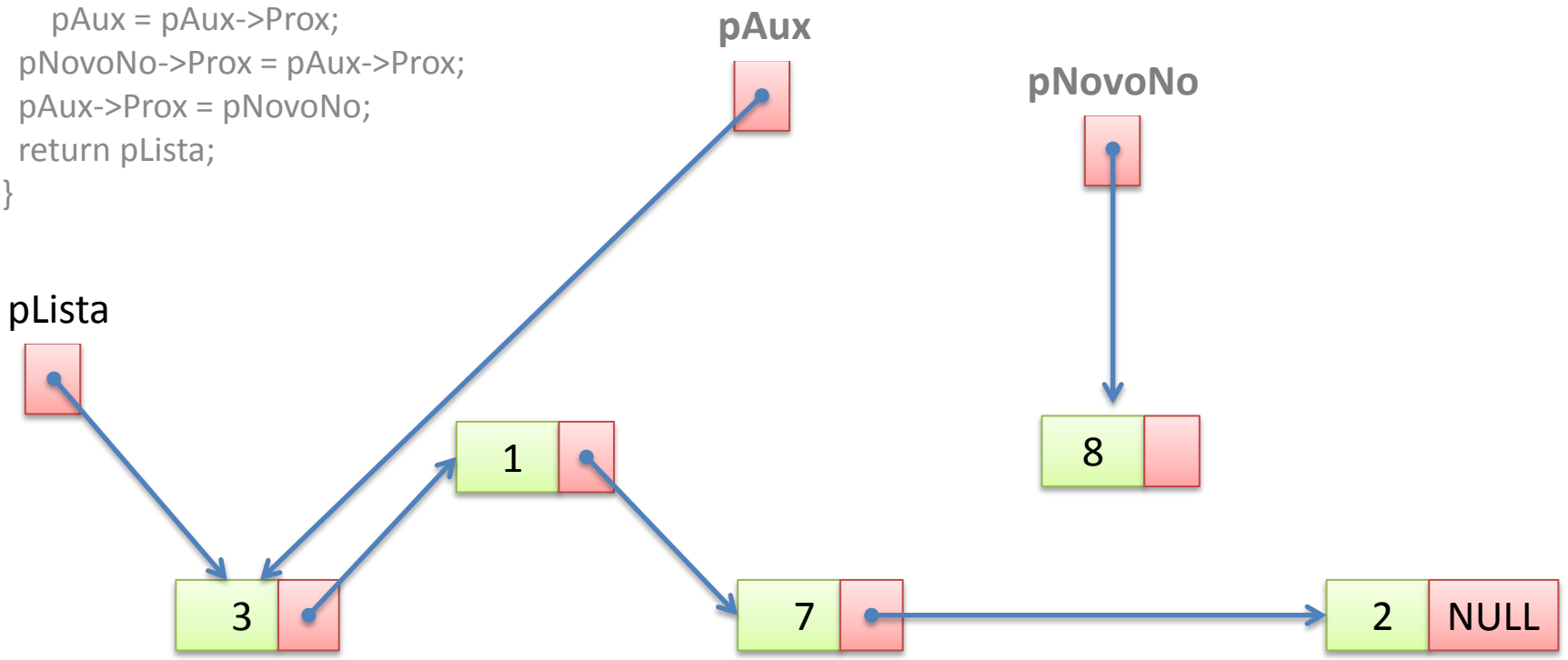
pChave = 2



Lista Encadeada – Inclusão Antes de Uma Chave

```
TNo *IncluiAntes(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;
  pAux->Prox = pNovoNo;
  return pLista;
}
```

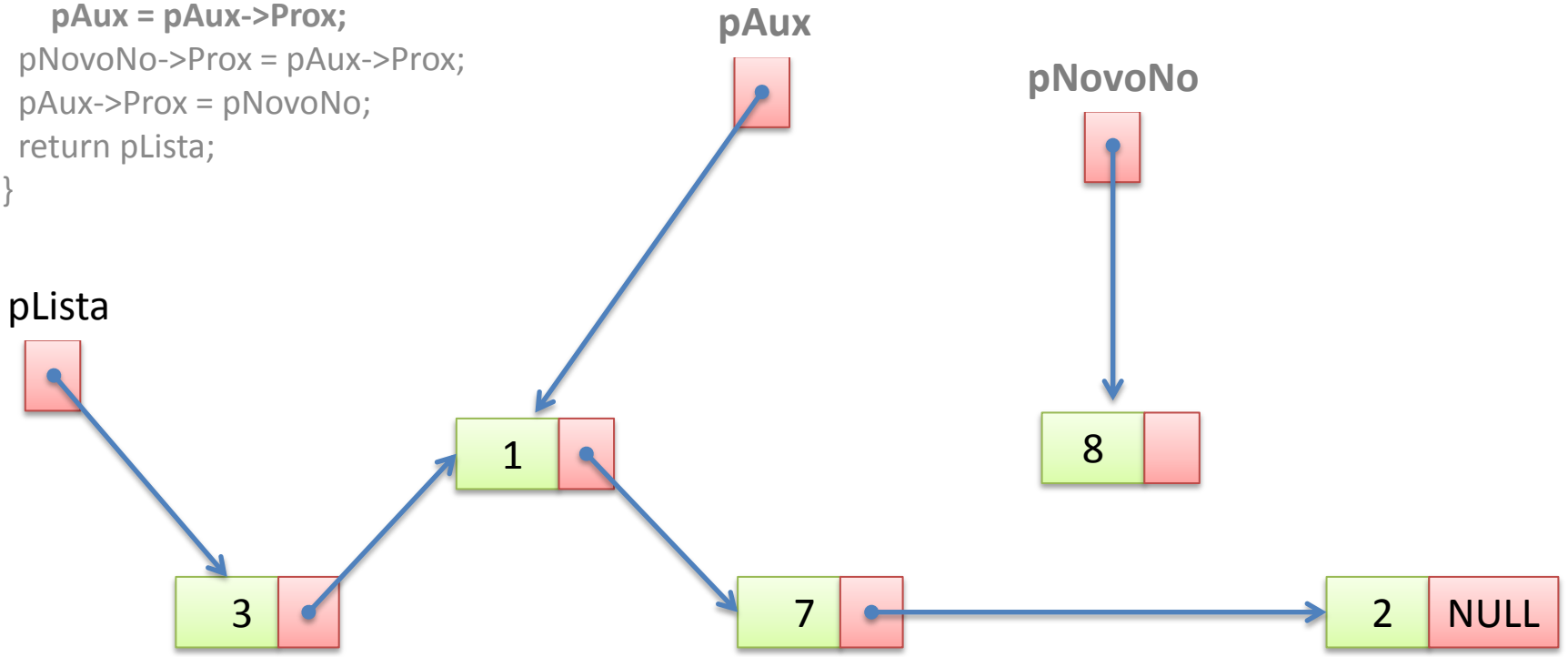
pChave = 2



Lista Encadeada – Inclusão Antes de Uma Chave

```
TNo *IncluiAntes(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;
  pAux->Prox = pNovoNo;
  return pLista;
}
```

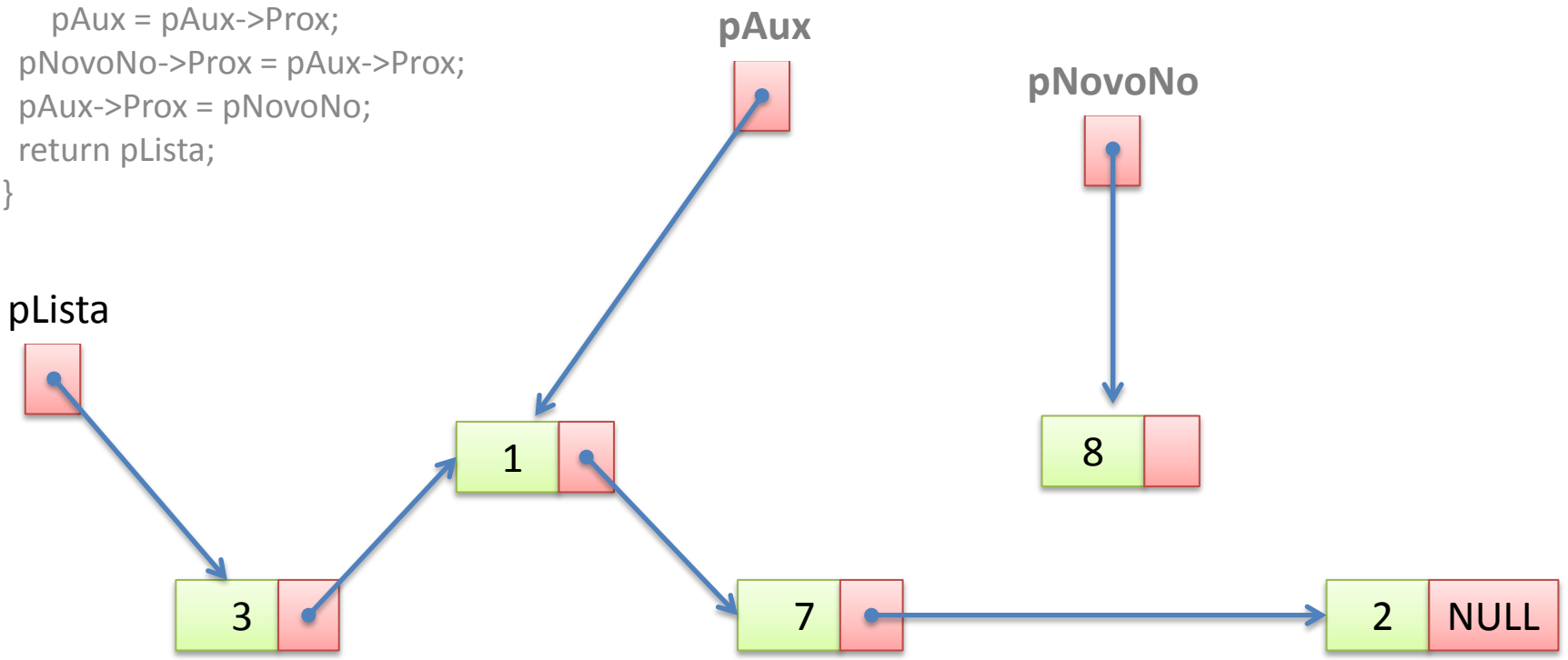
pChave = 2



Lista Encadeada – Inclusão Antes de Uma Chave

```
TNo *IncluiAntes(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;
  pAux->Prox = pNovoNo;
  return pLista;
}
```

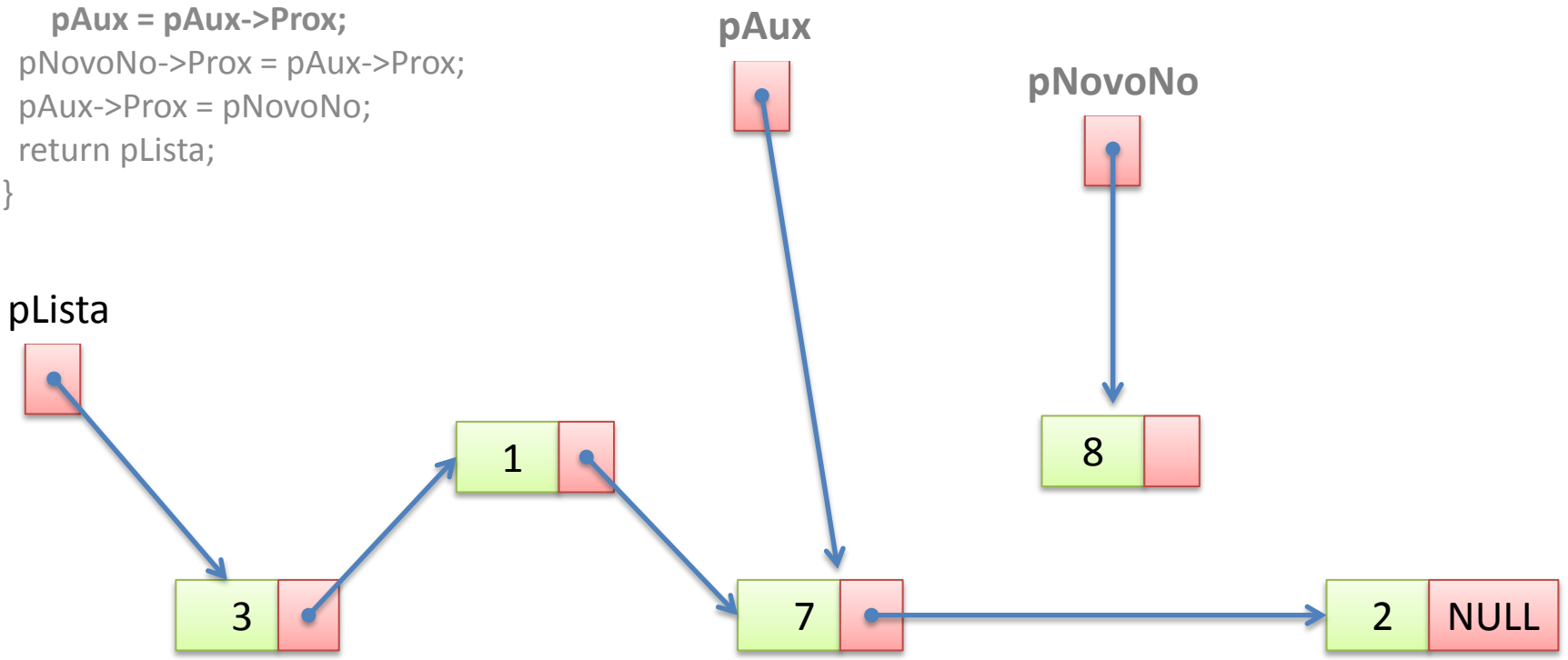
pChave = 2



Lista Encadeada – Inclusão Antes de Uma Chave

```
TNo *IncluiAntes(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;
  pAux->Prox = pNovoNo;
  return pLista;
}
```

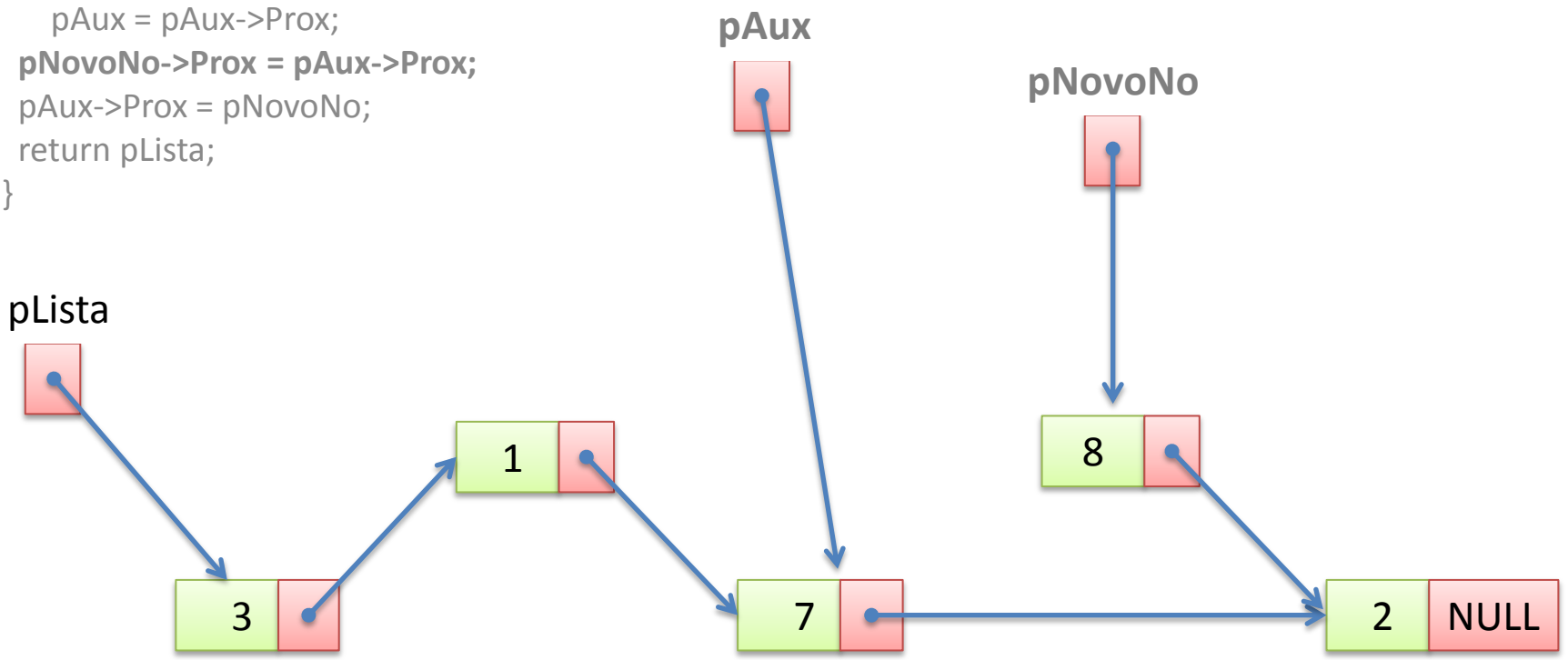
pChave = 2



Lista Encadeada – Inclusão Antes de Uma Chave

```
TNo *IncluiAntes(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;
  pAux->Prox = pNovoNo;
  return pLista;
}
```

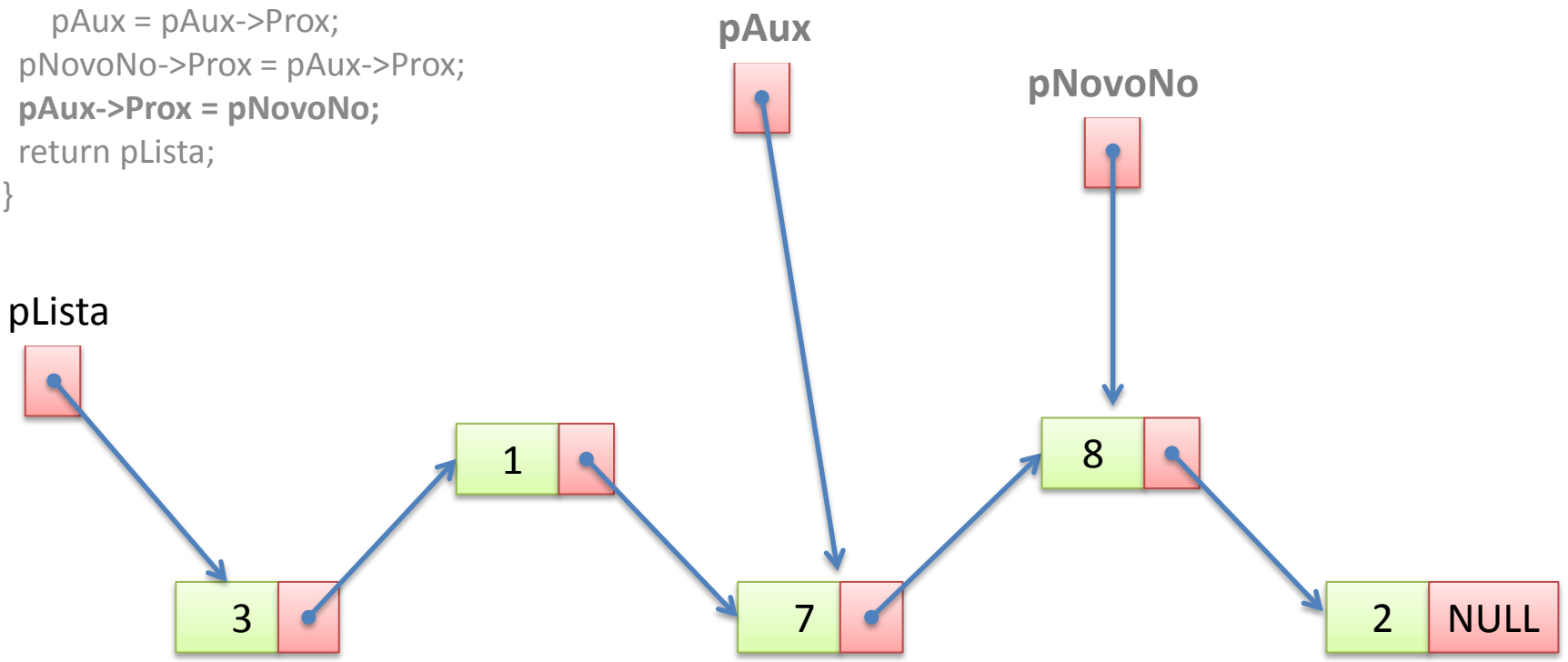
pChave = 2



Lista Encadeada – Inclusão Antes de Uma Chave

```
TNo *IncluiAntes(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;
  pAux->Prox = pNovoNo;
  return pLista;
}
```

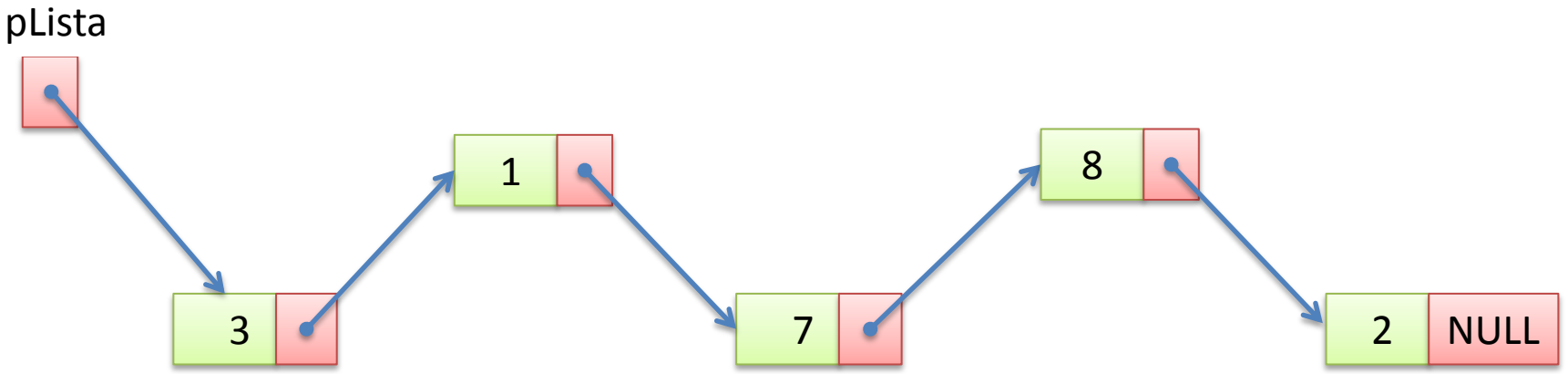
pChave = 2



Lista Encadeada – Inclusão Antes de Uma Chave

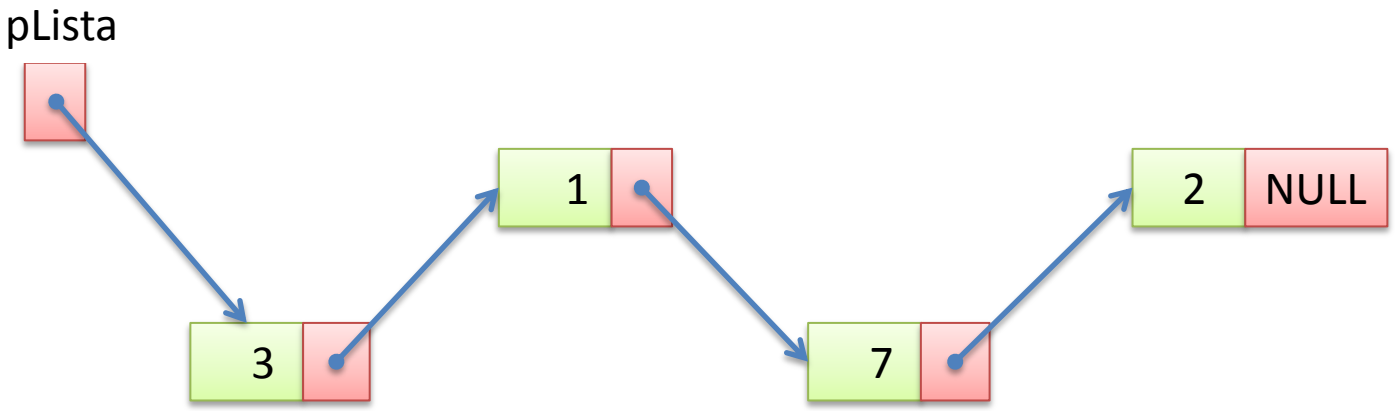
```
TNo *IncluiAntes(TNo *pLista, int pChave, int pValor)
{
  TNo *pNovoNo, *pAux;
  pNovoNo = (TNo *) malloc(sizeof(TNo));
  pNovoNo->Numero = pValor;
  pAux = pLista;
  while (pAux->Prox->Valor != pChave)
     pAux = pAux->Prox;
  pNovoNo->Prox = pAux->Prox;
  pAux->Prox = pNovoNo;
  return pLista;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

O(n)



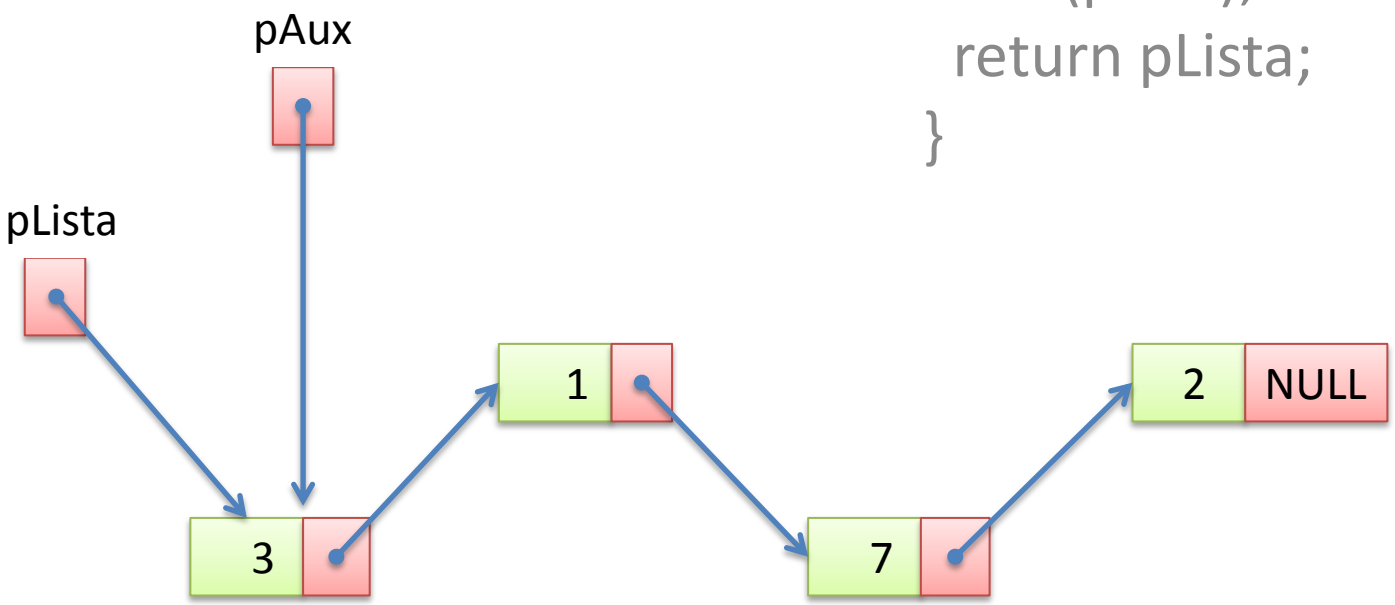
Lista Encadeada – Exclusão na Cabeça

```
TNo *ExcluiCabeca(TNo *pLista)
{
  TNo *pAux;
  pAux = pLista;
  pLista = pLista->Prox;
  free(pAux);
  return pLista;
}
```



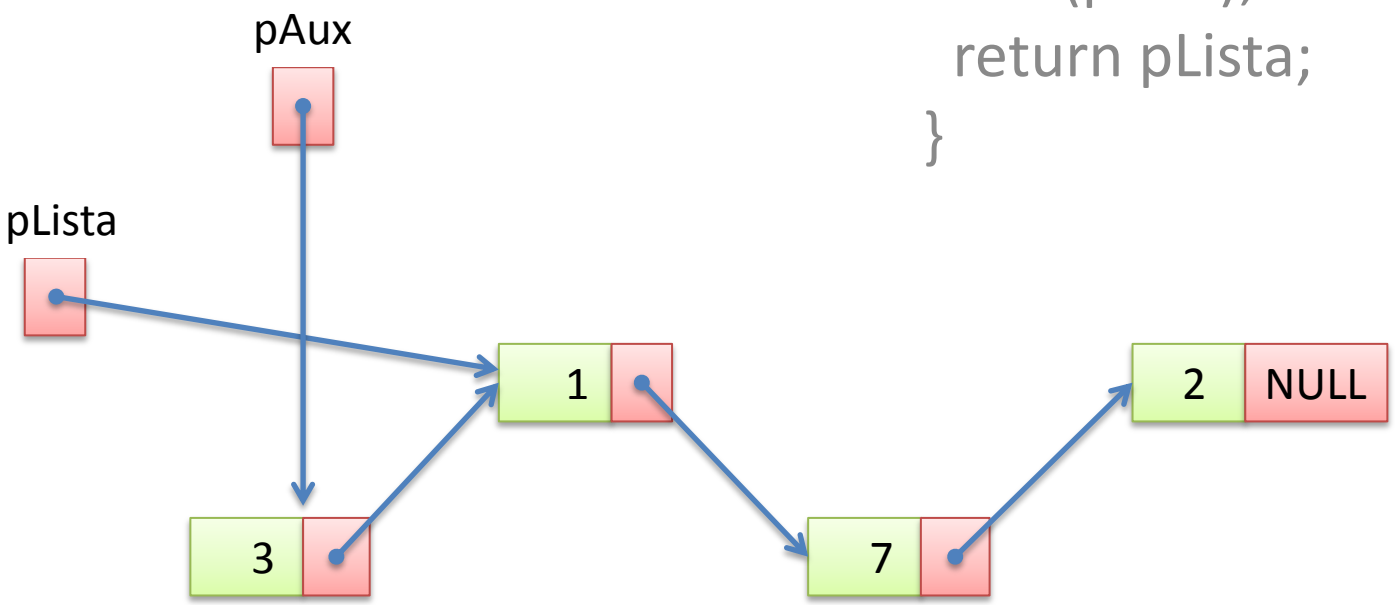
Lista Encadeada – Exclusão na Cabeça

```
TNo *ExcluiCabeca(TNo *pLista)
{
  TNo *pAux;
  pAux = pLista;
  pLista = pLista->Prox;
  free(pAux);
  return pLista;
}
```



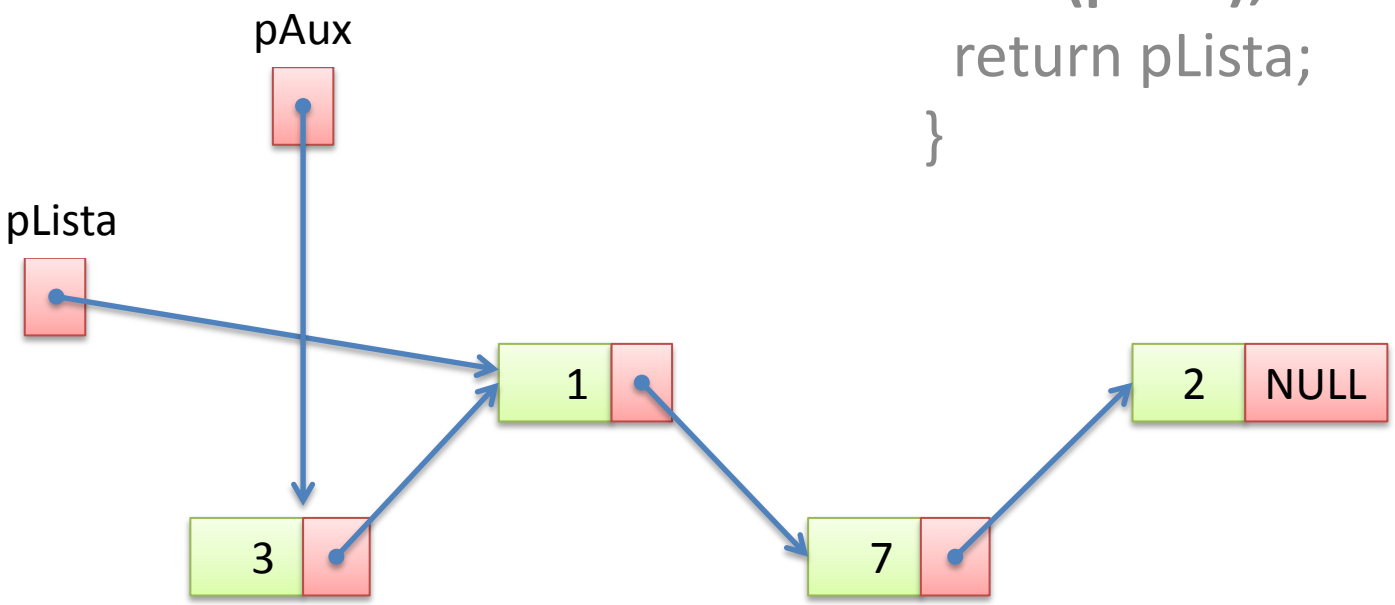
Lista Encadeada – Exclusão na Cabeça

```
TNo *ExcluiCabeca(TNo *pLista)
{
  TNo *pAux;
  pAux = pLista;
  pLista = pLista->Prox;
  free(pAux);
  return pLista;
}
```



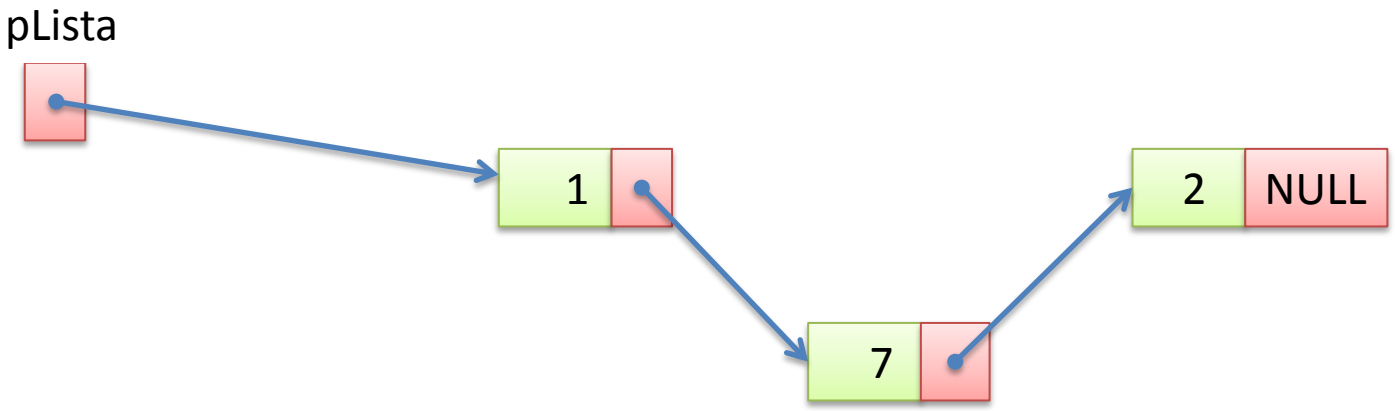
Lista Encadeada – Exclusão na Cabeça

```
TNo *ExcluiCabeca(TNo *pLista)
{
  TNo *pAux;
  pAux = pLista;
  pLista = pLista->Prox;
  free(pAux);
  return pLista;
}
```



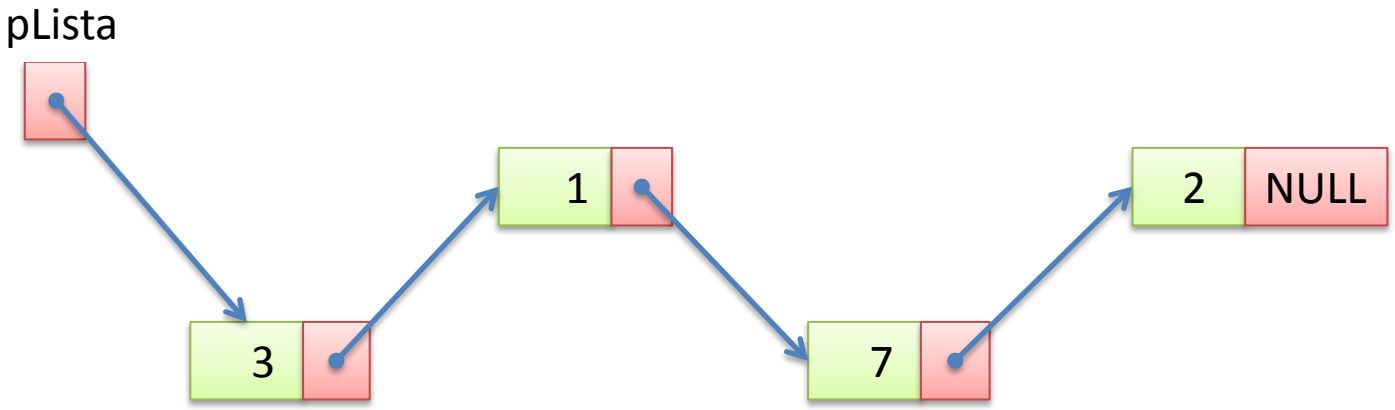
Lista Encadeada – Exclusão na Cabeça

```
TNo *ExcluiCabeca(TNo *pLista)
{
  TNo *pAux;
  pAux = pLista;
  pLista = pLista->Prox;
  free(pAux);
  return pLista;
}
```



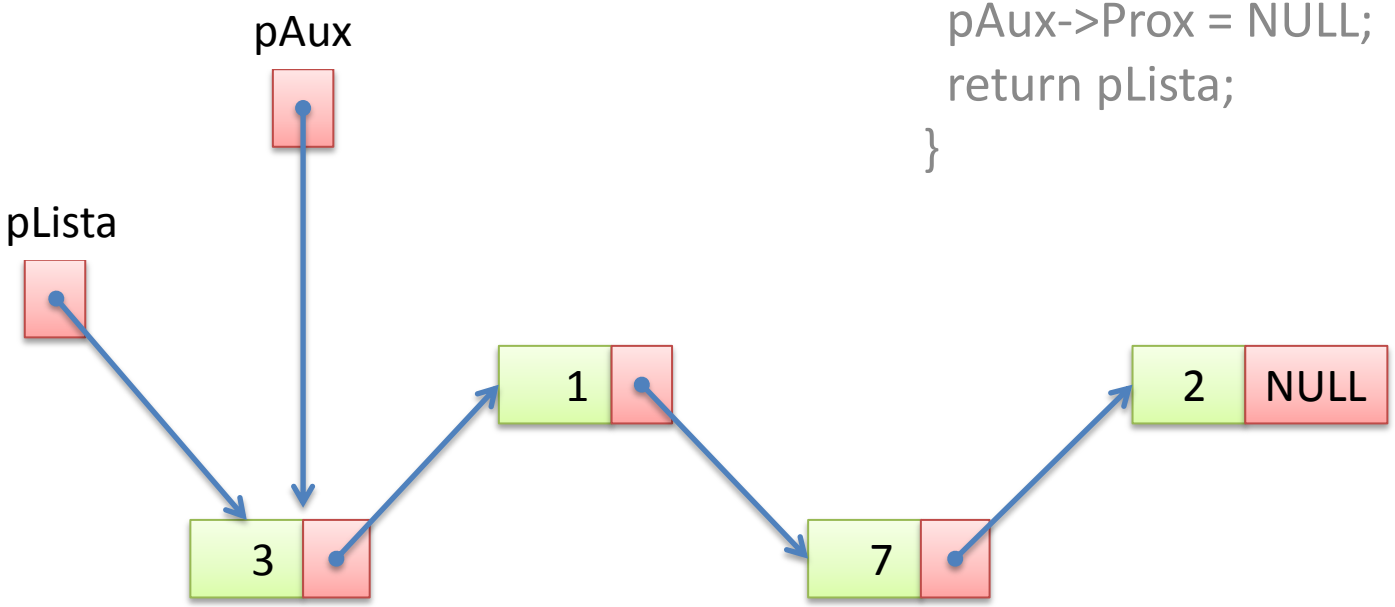
Lista Encadeada – Exclusão na Calda

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



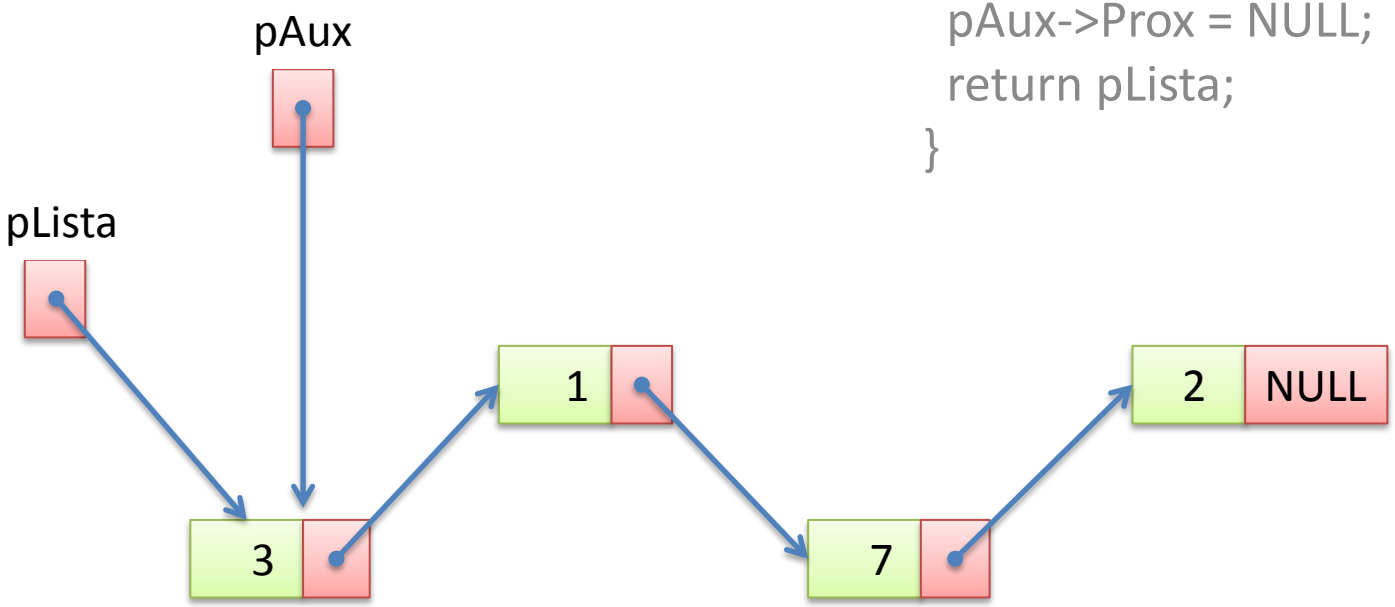
Lista Encadeada – Exclusão na Calda

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



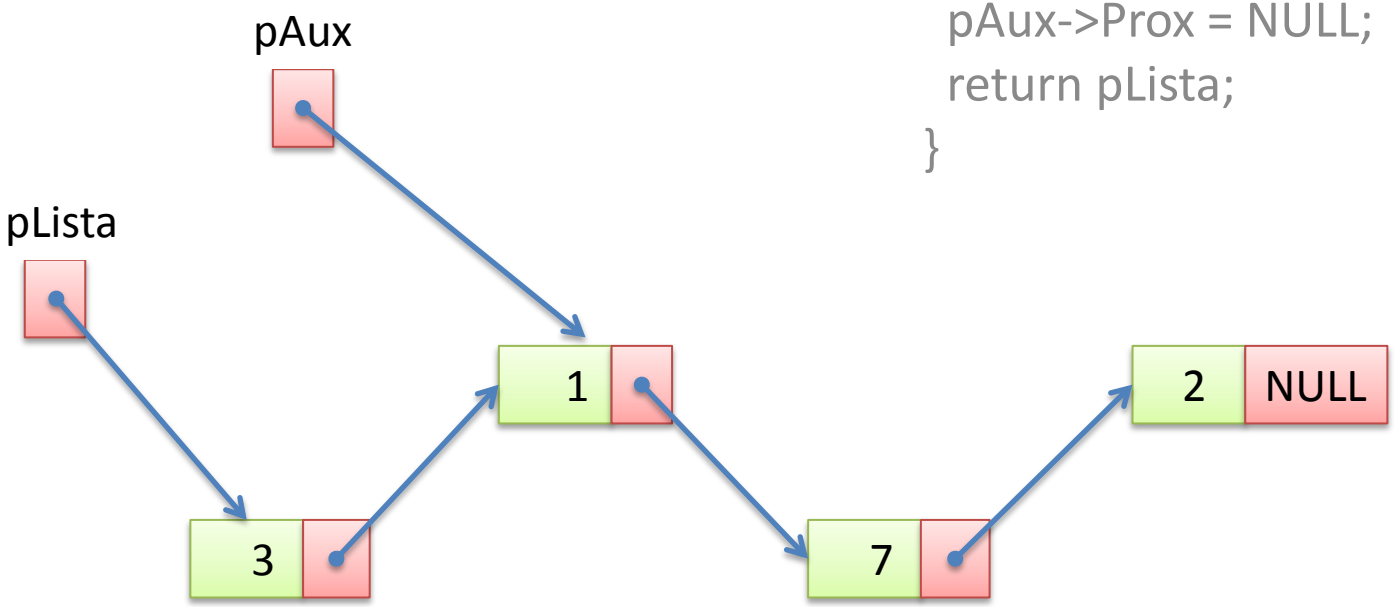
Lista Encadeada – Exclusão na Calda

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



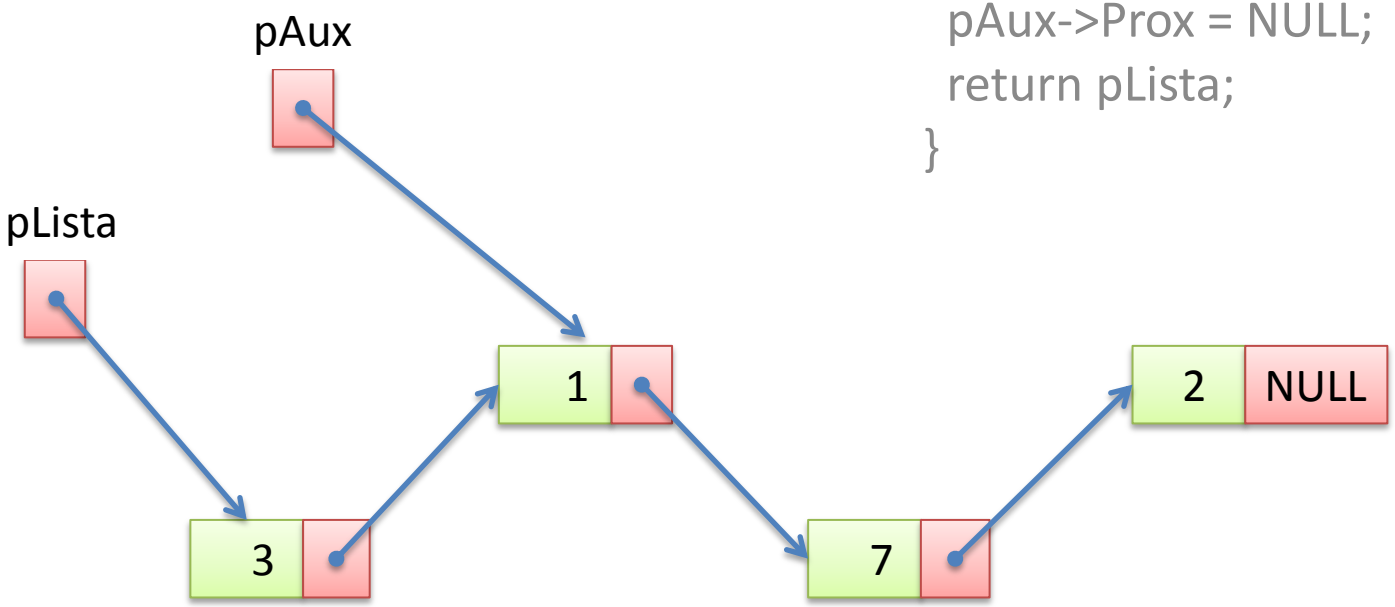
Lista Encadeada – Exclusão na Calda

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



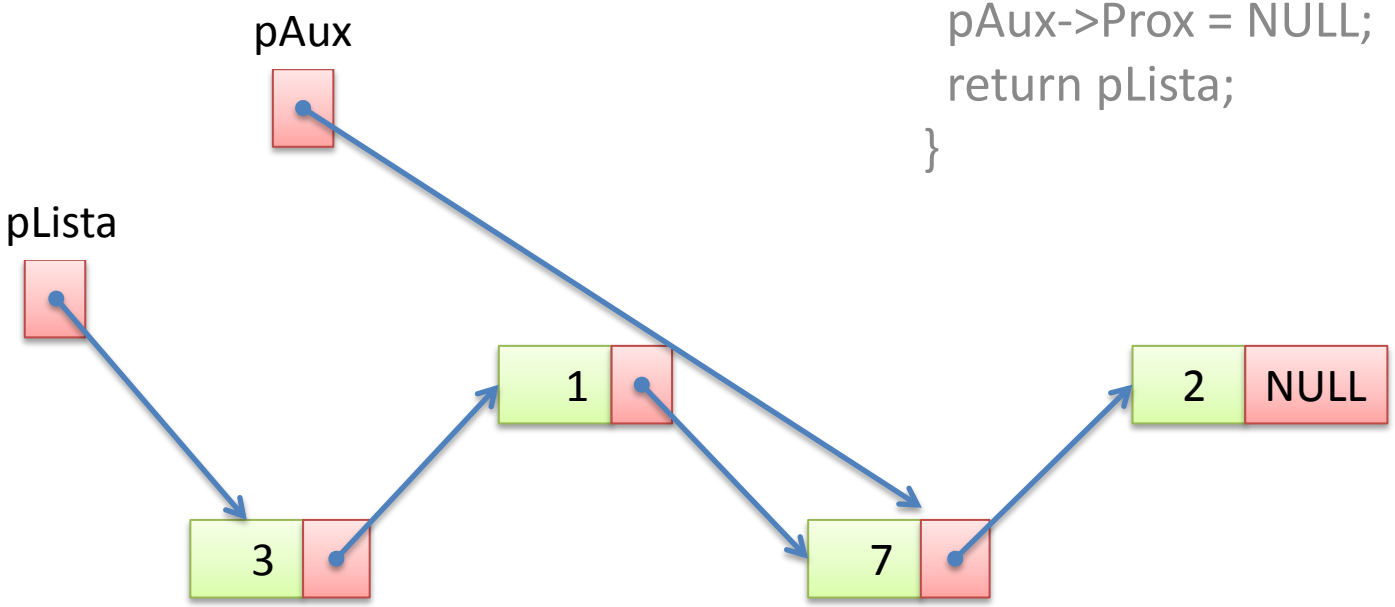
Lista Encadeada – Exclusão na Calda

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



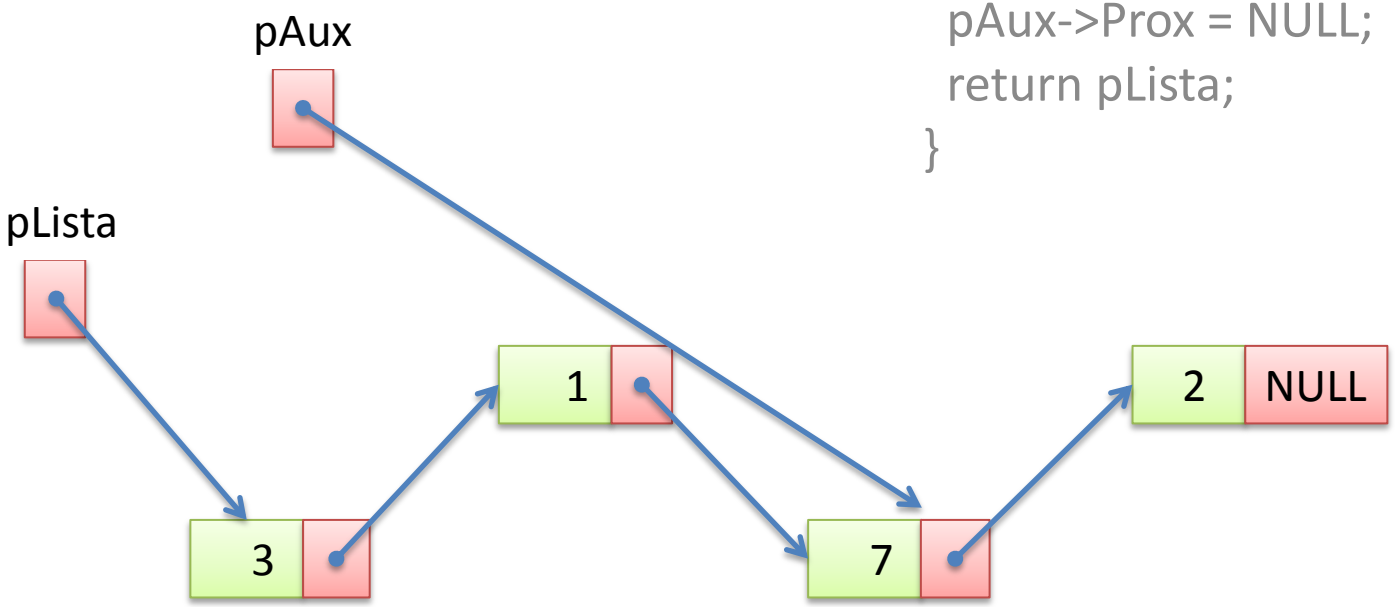
Lista Encadeada – Exclusão na Calda

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



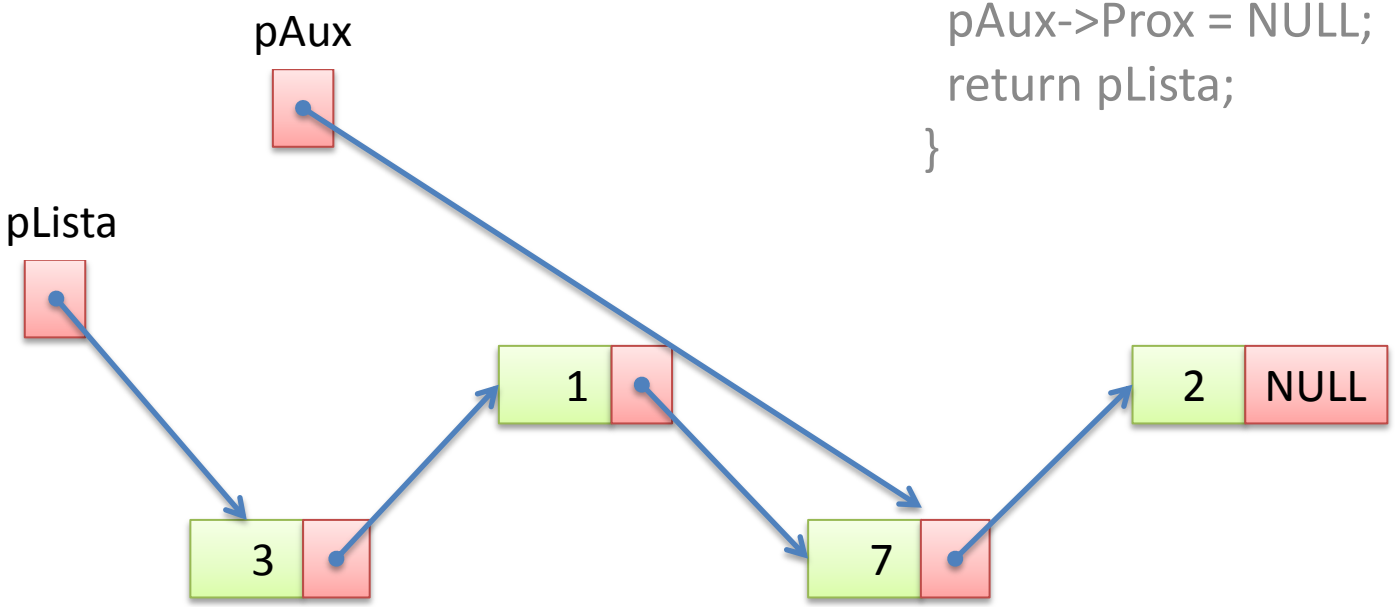
Lista Encadeada – Exclusão na Calda

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



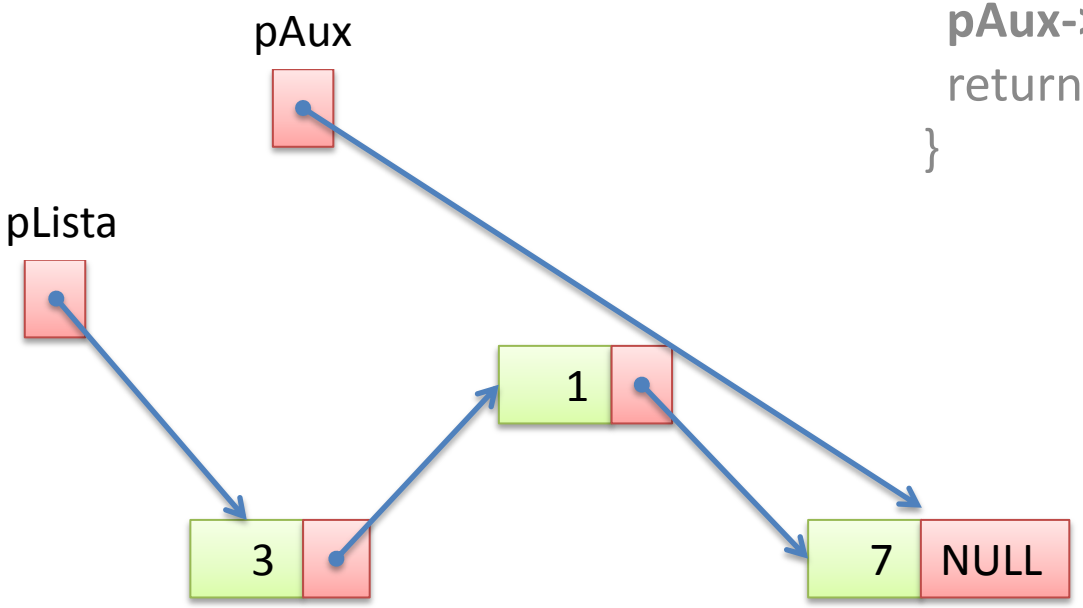
Lista Encadeada – Exclusão na Calda

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



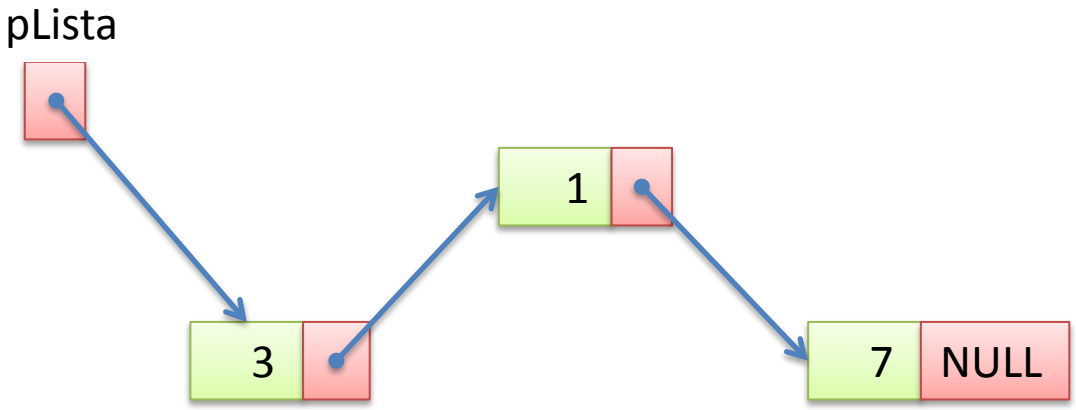
Lista Encadeada – Exclusão na Calda

```
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 Encadeada – Exclusão na Calda

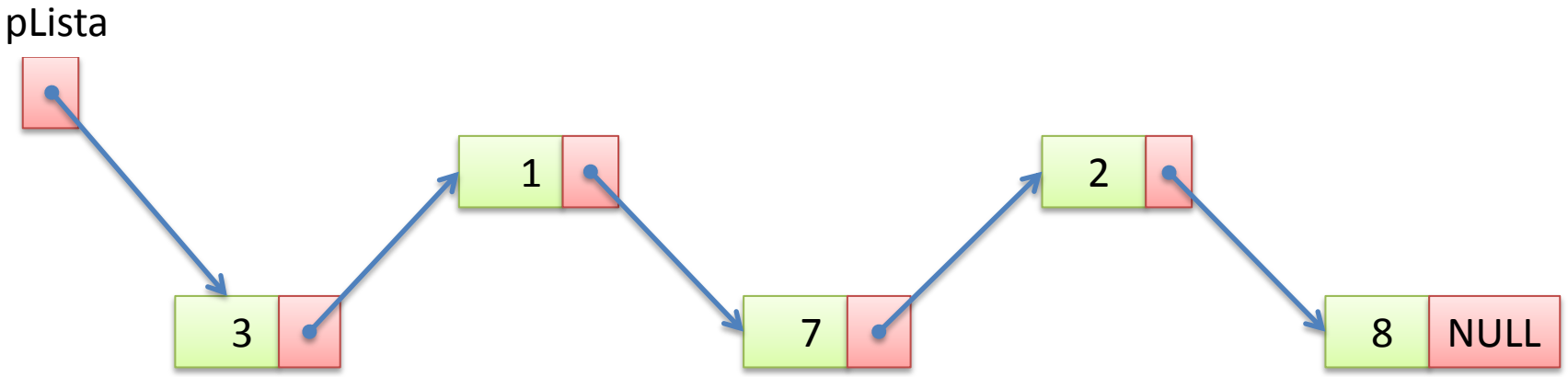
```
TNo *ExcluiCalda(TNo *pLista)
{
  TNo *pAux;
  pAux = pLista;
  while (pAux->Prox->Prox != NULL)
     pAux = pAux->Prox;
  free(pAux->Prox);
  pAux->Prox = NULL;
  return pLista;
}
```



Lista Encadeada – Exclusão de Uma Chave

pChave = 7

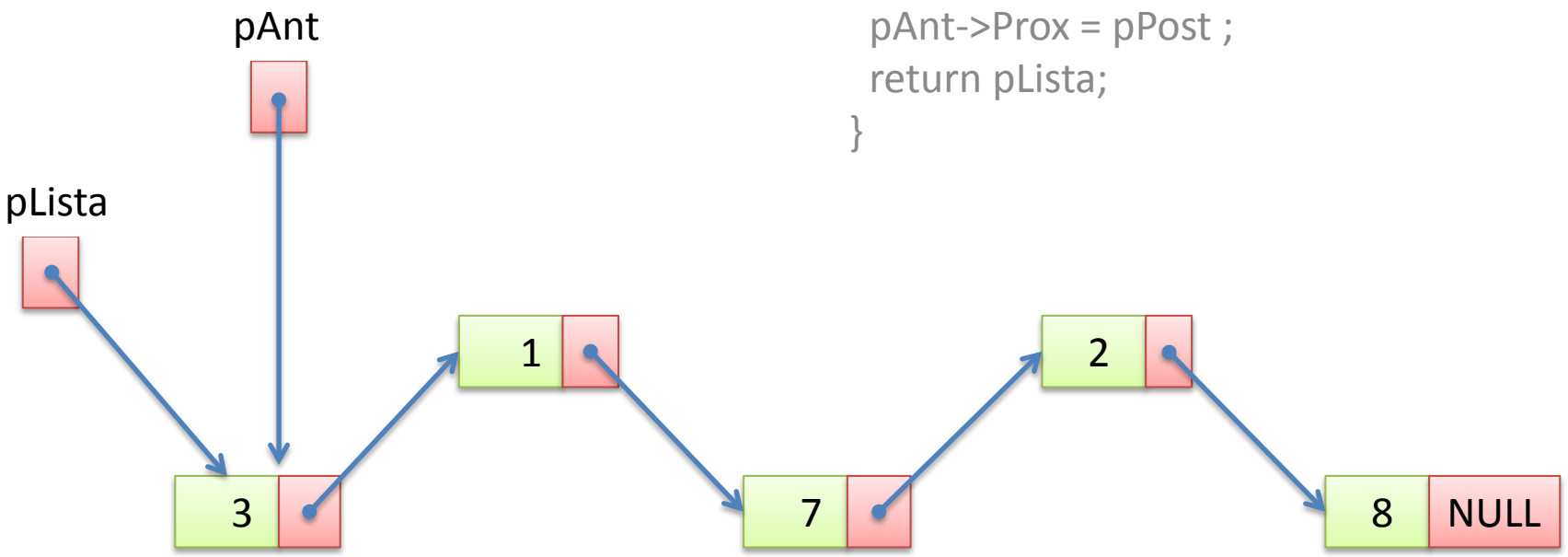
```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
  TNo *pAnt, *pPost;
  pAnt = pLista;
  while (pAnt->Prox->Numero != pChave)
     pAnt = pAnt->Prox;
  pPost = pAnt->Prox->Prox;
  free(pAnt->Prox);
  pAnt->Prox = pPost ;
  return pLista;
}
```



Lista Encadeada – Exclusão de Uma Chave

pChave = 7

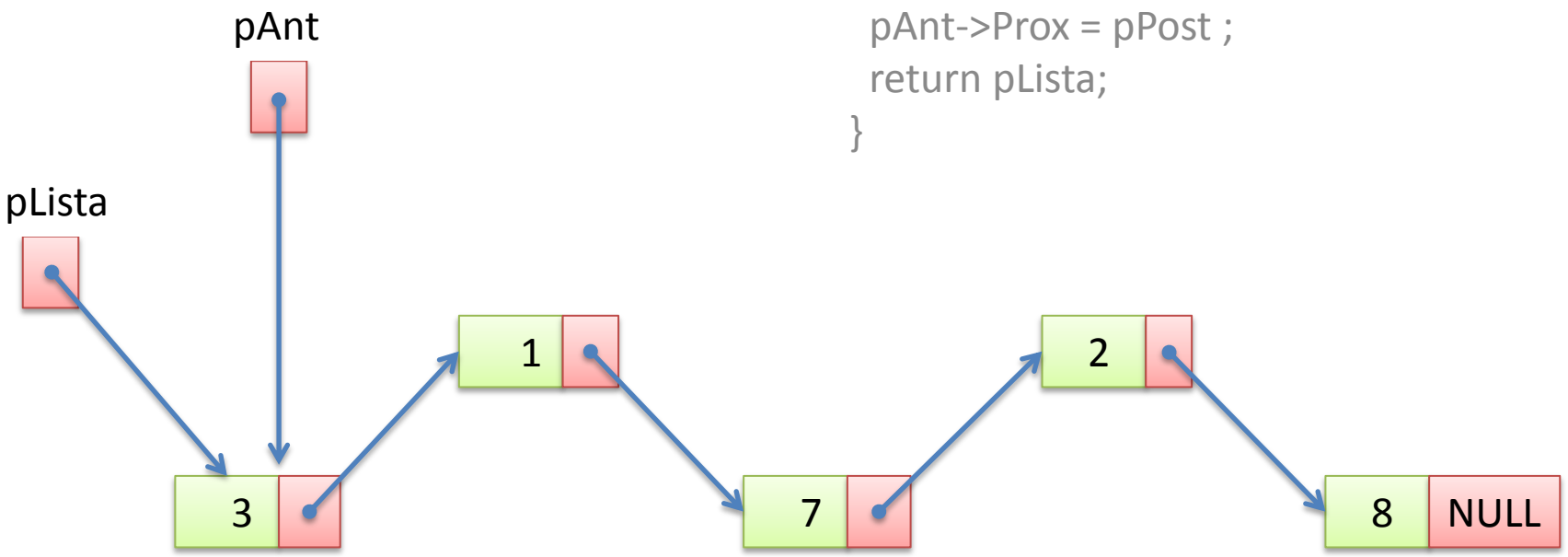
```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
  TNo *pAnt, *pPost;
  pAnt = pLista;
  while (pAnt->Prox->Numero != pChave)
     pAnt = pAnt->Prox;
  pPost = pAnt->Prox->Prox;
  free(pAnt->Prox);
  pAnt->Prox = pPost ;
  return pLista;
}
```



Lista Encadeada – Exclusão de Uma Chave

pChave = 7

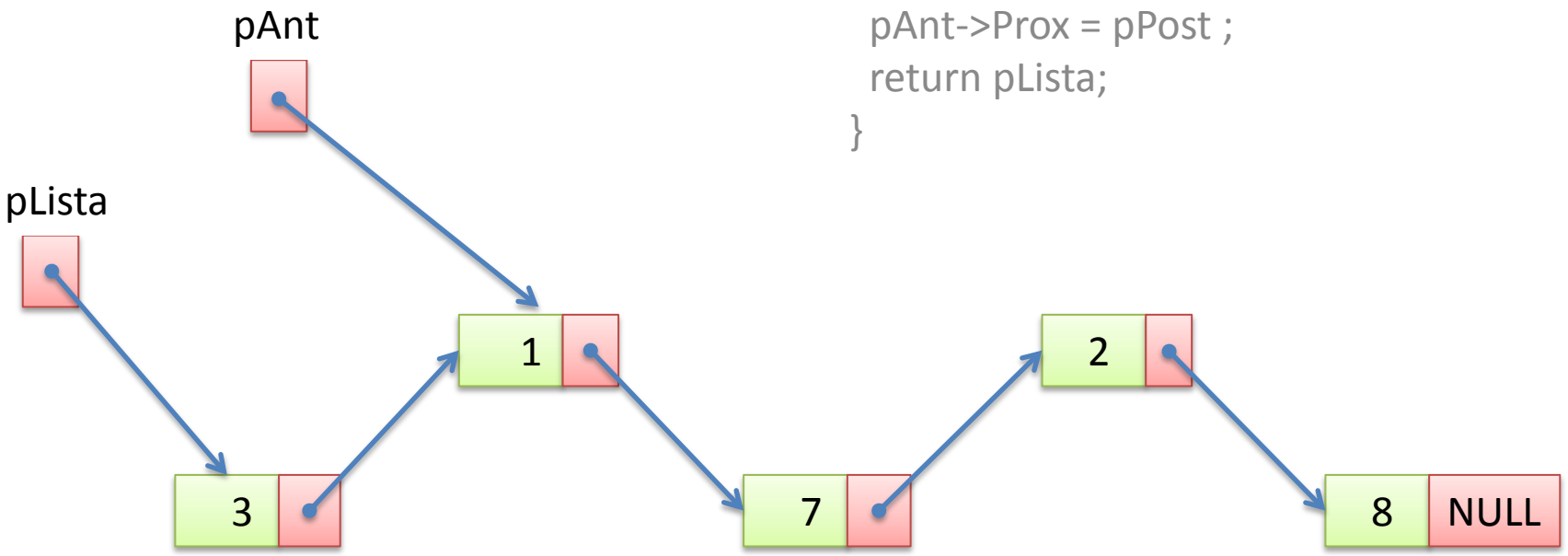
```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
  TNo *pAnt, *pPost;
  pAnt = pLista;
  while (pAnt->Prox->Numero != pChave)
     pAnt = pAnt->Prox;
  pPost = pAnt->Prox->Prox;
  free(pAnt->Prox);
  pAnt->Prox = pPost ;
  return pLista;
}
```



Lista Encadeada – Exclusão de Uma Chave

pChave = 7

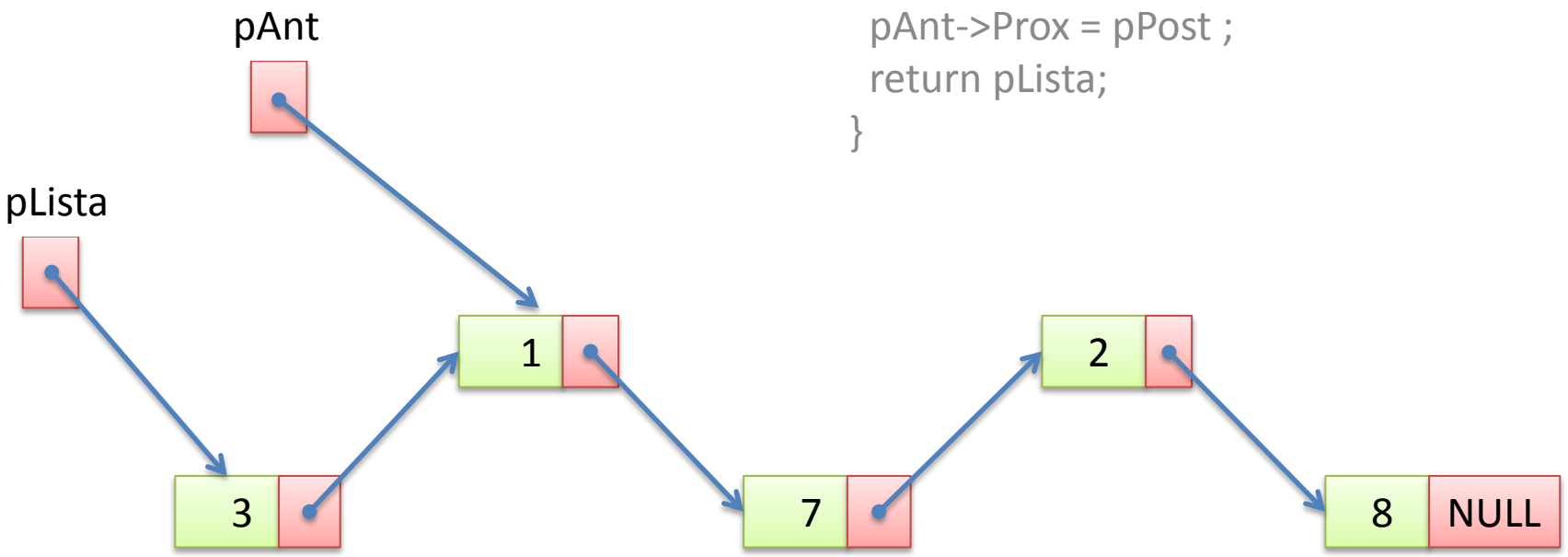
```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
  TNo *pAnt, *pPost;
  pAnt = pLista;
  while (pAnt->Prox->Numero != pChave)
     pAnt = pAnt->Prox;
  pPost = pAnt->Prox->Prox;
  free(pAnt->Prox);
  pAnt->Prox = pPost ;
  return pLista;
}
```



Lista Encadeada – Exclusão de Uma Chave

pChave = 7

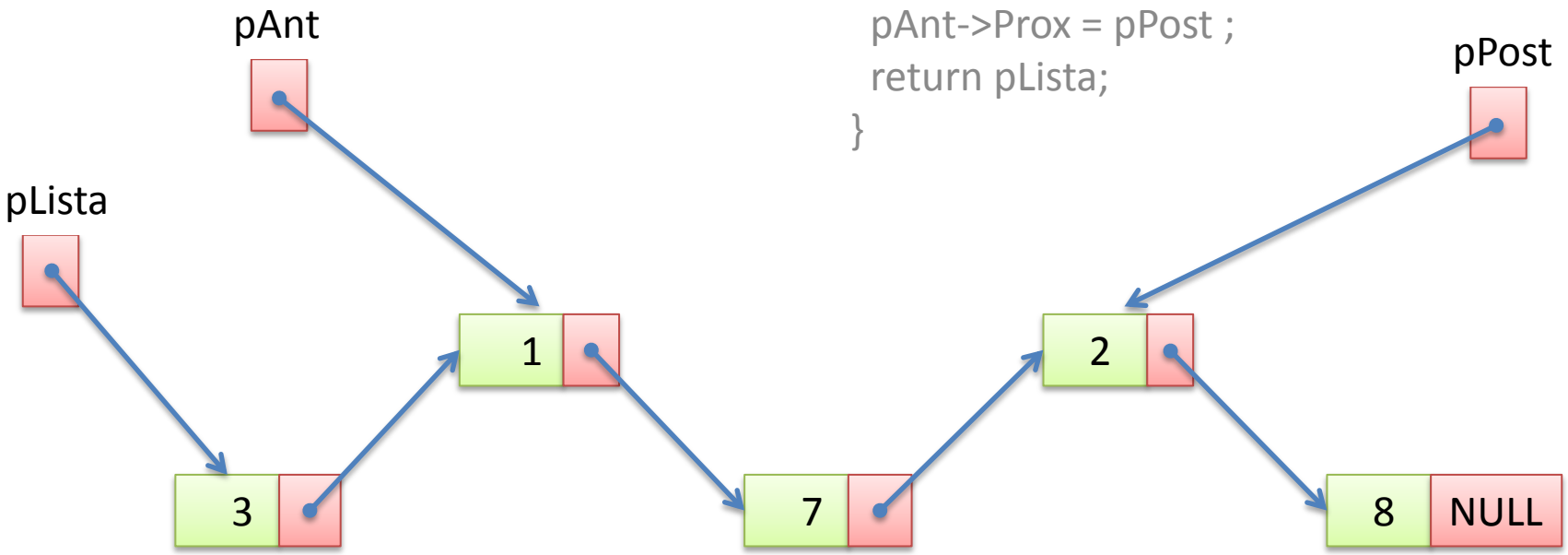
```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
  TNo *pAnt, *pPost;
  pAnt = pLista;
  while (pAnt->Prox->Numero != pChave)
     pAnt = pAnt->Prox;
  pPost = pAnt->Prox->Prox;
  free(pAnt->Prox);
  pAnt->Prox = pPost ;
  return pLista;
}
```



Lista Encadeada – Exclusão de Uma Chave

pChave = 7

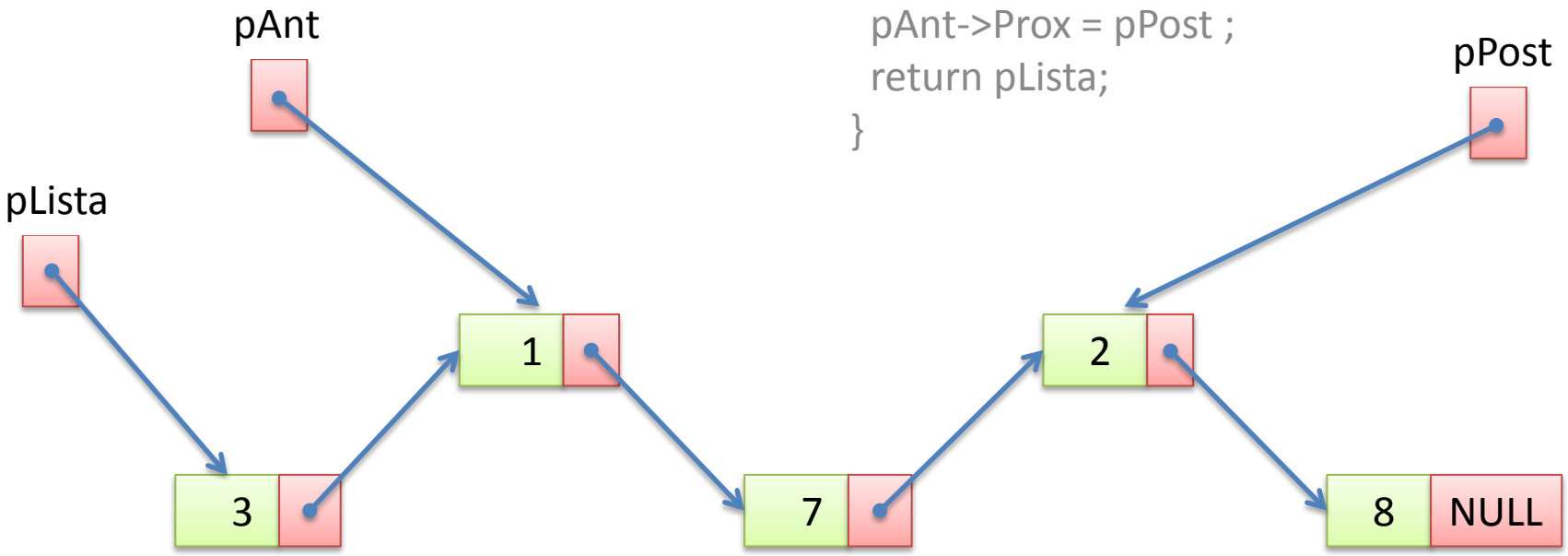
```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
  TNo *pAnt, *pPost;
  pAnt = pLista;
  while (pAnt->Prox->Numero != pChave)
     pAnt = pAnt->Prox;
  pPost = pAnt->Prox->Prox;
  free(pAnt->Prox);
  pAnt->Prox = pPost ;
  return pLista;
}
```



Lista Encadeada – Exclusão de Uma Chave

pChave = 7

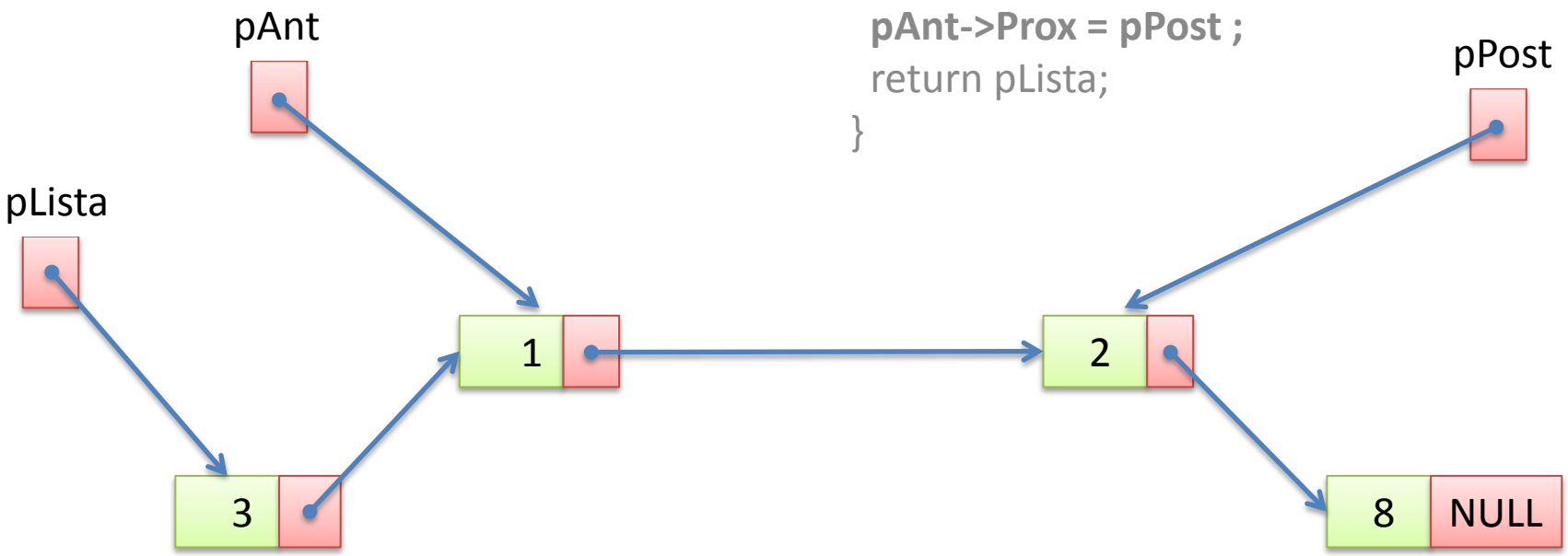
```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
  TNo *pAnt, *pPost;
  pAnt = pLista;
  while (pAnt->Prox->Numero != pChave)
     pAnt = pAnt->Prox;
  pPost = pAnt->Prox->Prox;
  free(pAnt->Prox);
  pAnt->Prox = pPost ;
  return pLista;
}
```



Lista Encadeada – Exclusão de Uma Chave

pChave = 7

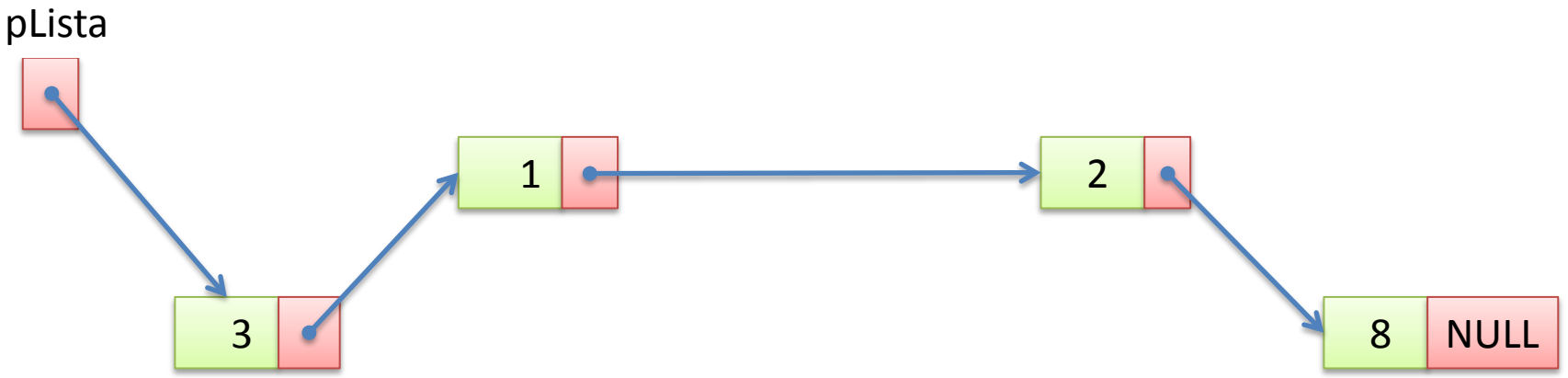
```
TNo *ExcluiChave(TNo *pLista, int pChave)
{
  TNo *pAnt, *pPost;
  pAnt = pLista;
  while (pAnt->Prox->Numero != pChave)
     pAnt = pAnt->Prox;
  pPost = pAnt->Prox->Prox;
  free(pAnt->Prox);
  pAnt->Prox = pPost ;
  return pLista;
}
```



Lista Encadeada – Exclusão de Uma Chave

pChave = 7

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

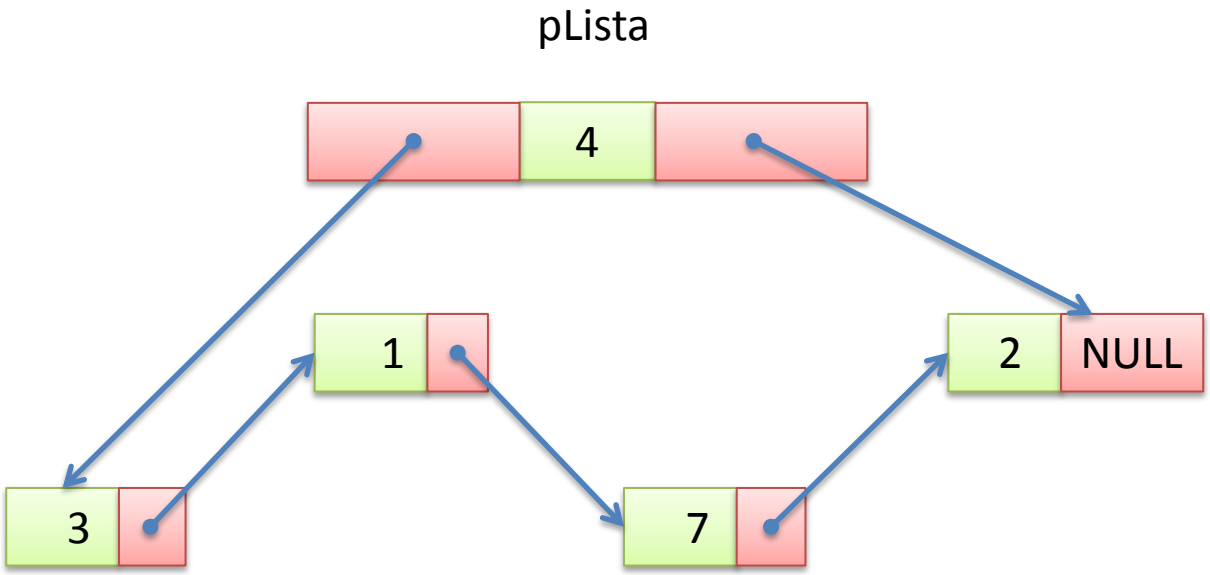


Lista Encadeada

```
struct TNo
{
  int Numero;
  Tno *Prox;
};
```


```
struct TLista
{
  TNo *Primeiro;
  int Qtde;
  TNo *Ultimo;
};
```

Lista Encadeada



Prática...

- **No ambiente virtual (Moodle):**
  - **Prática ListasEncadeadas.pdf**