Listas Duplamente Encadeadas

Introdução

- Lista encadeada: forma um encadeamento simples entre os elementos: cada elemento armazena um ponteiro para o próximo elemento da lista.

- Desta forma, não temos como percorrer eficientemente os elementos em ordem inversa, isto é, do final para o início da lista.

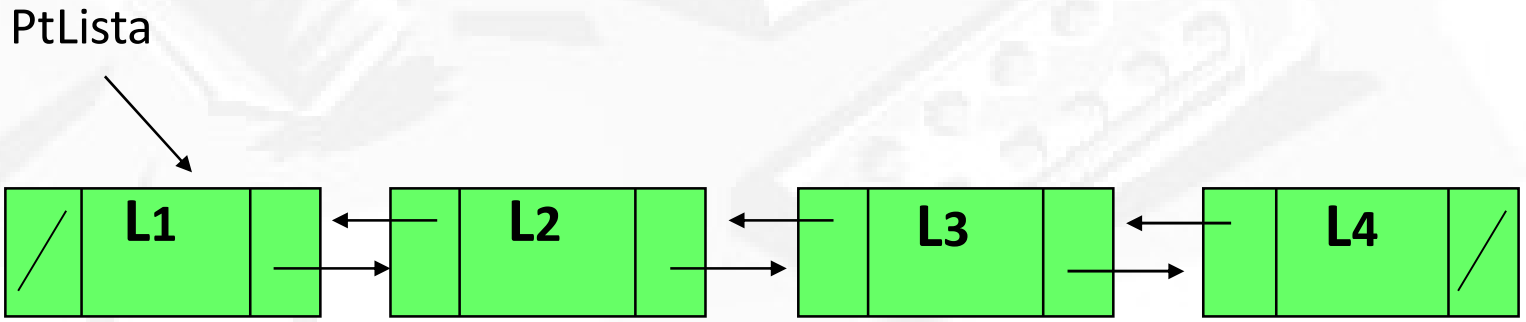
- O encadeamento simples também dificulta a retirada de um elemento da lista.
  - ✓ Mesmo se tivermos o ponteiro do elemento que desejamos retirar, temos que percorrer a lista, elemento por elemento, para encontrarmos o elemento anterior.

Definição

- Para solucionar esses problemas, podemos formar o que chamamos de ***listas duplamente encadeadas***.
- Cada elemento tem um ponteiro para o próximo elemento e um ponteiro para o elemento anterior.
- Desta forma, dado um elemento, podemos acessar ambos os elementos adjacentes: o próximo e o anterior.

Definição

- Se tivermos um ponteiro para o último elemento da lista, podemos percorrer a lista em ordem inversa, bastando acessar continuamente o elemento anterior, até alcançar o primeiro elemento da lista, que não tem elemento anterior (o ponteiro do elemento anterior vale `NULL`).



Estrutura de uma lista duplamente encadeada

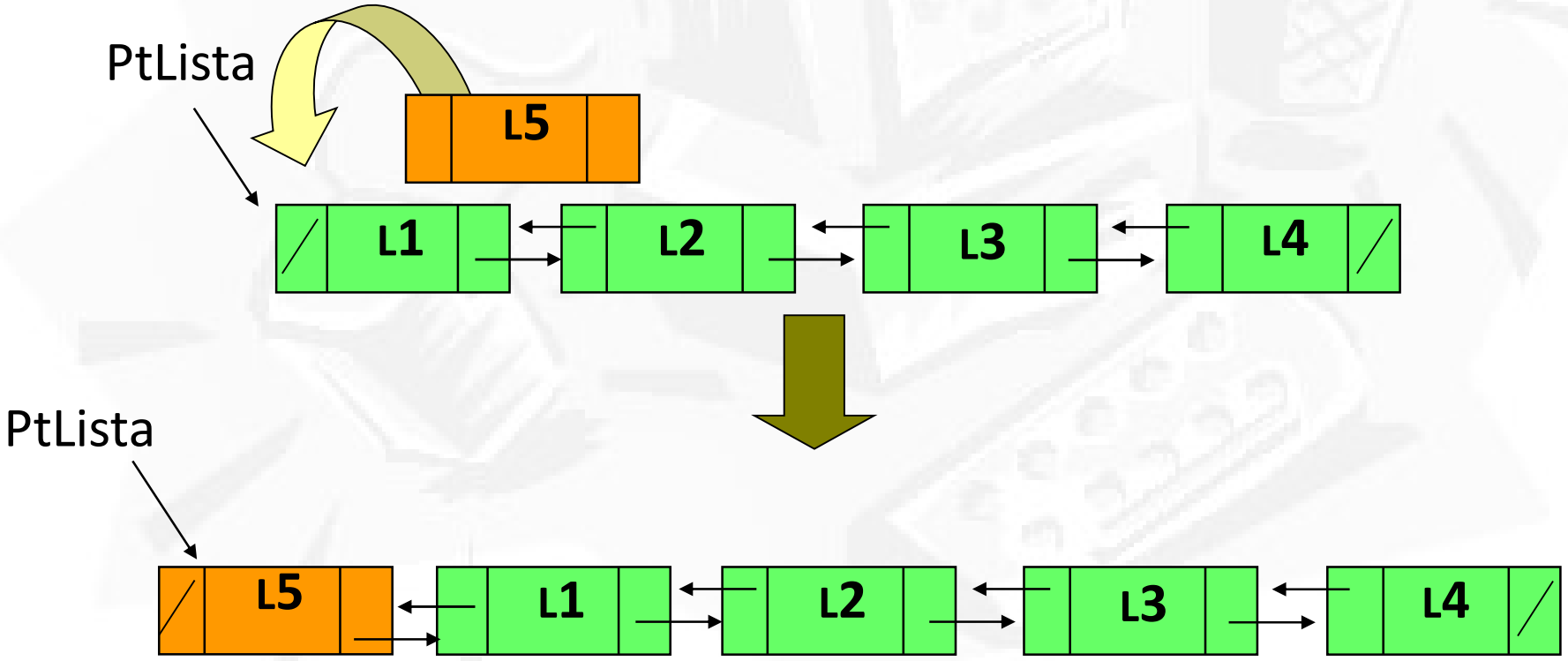
- O nó da lista pode ser representado pela estrutura abaixo e a lista pode ser representada utilizando o ponteiro para o primeiro nó.

```
typedef struct lista2 {
int info;
struct lista2* ant;
struct lista2* prox;
}TLista2;

typedef TLista2 *PLista2;
```

Inserção em lista duplamente encadeada

- O código a seguir mostra uma possível implementação da função que insere novos elementos no **início da lista.**



Inserção em lista duplamente encadeada

- O código a seguir mostra uma possível implementação da função que insere novos elementos no **início da lista.**

```
PLista2 insere_Inicio (PLista2 l, int v) {

 PLista2 novo = (PLista2) malloc(sizeof(TLista2));
 novo->info = v;
 novo->prox = l;
 novo->ant = NULL;
 if (l != NULL)
   l->ant = novo;
 return novo;
}
```

Inserção em lista duplamente encadeada

- O código a seguir mostra uma possível implementação da função que insere novos elementos no **FIM da lista.**

```
PLista2 Insere_Fim (PLista2 l, int v) {
PLista2 novo = (PLista2) malloc(sizeof(TLista2));
if (novo == NULL) return NULL;
novo->info = v;
novo->prox = NULL;
if (l == NULL) {    // sera o primeiro
 novo->ant = NULL;
 l = novo;
 return novo;
}
 else {
        PLista2 aux;
        aux = l;
        while(aux->prox != NULL)
        {
            aux = aux->prox;
        }
        aux->prox = novo;
        novo->ant = aux;
     }
 return l;
```

Busca em lista duplamente encadeada

- A função de busca recebe a informação referente ao elemento que queremos buscar e tem como valor de retorno o ponteiro do nó da lista que representa o elemento.

```
PLista2 busca (PLista2 l, int v){
PLista2 p;
for (p=l; p!=NULL; p=p->prox)
if (p->info == v)
   return p;
return NULL; /* não achou o elemento */
}
```

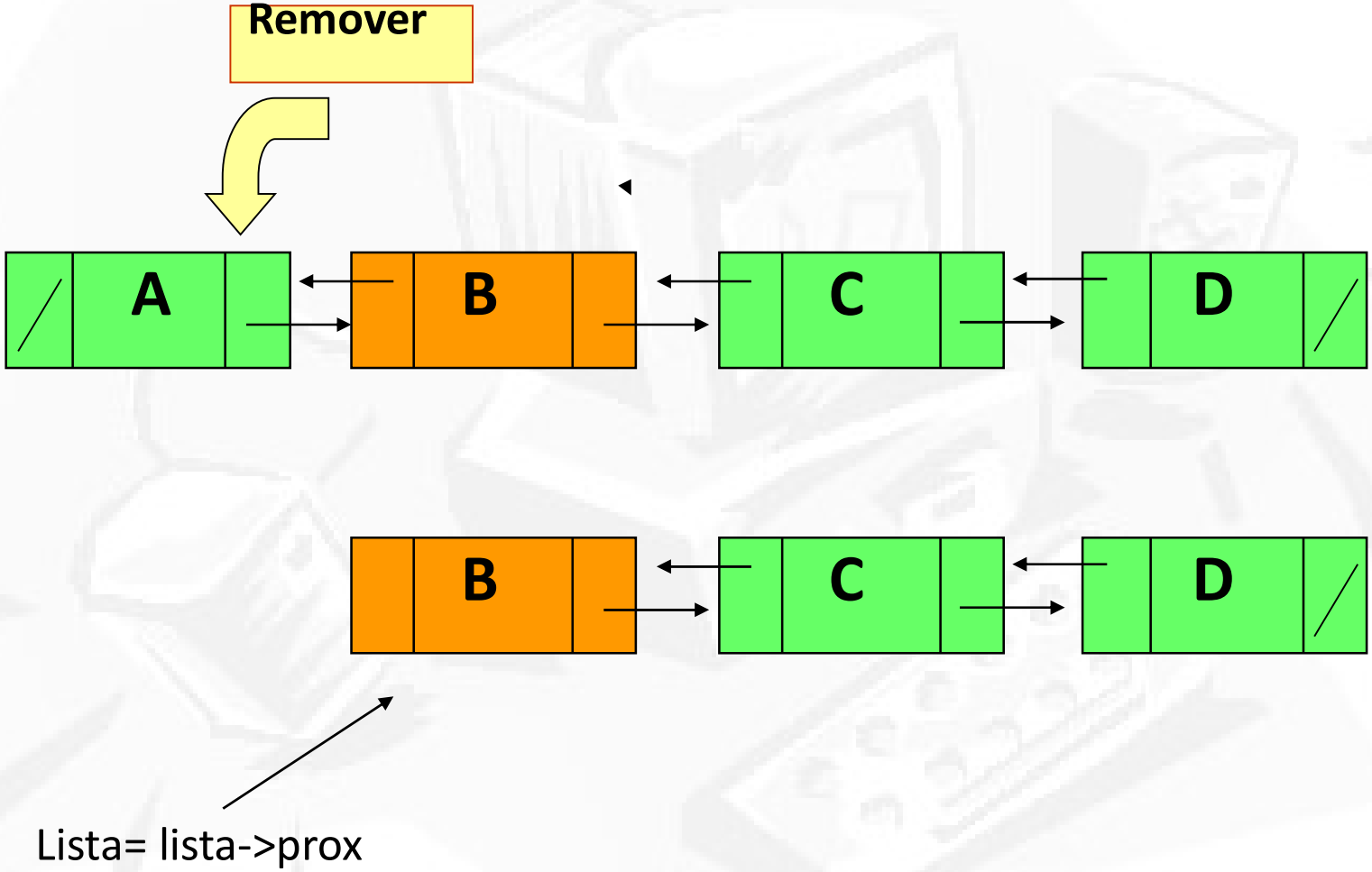
Remoção em lista duplamente encadeada

- Podemos retirar um elemento da lista conhecendo apenas o ponteiro para esse elemento.

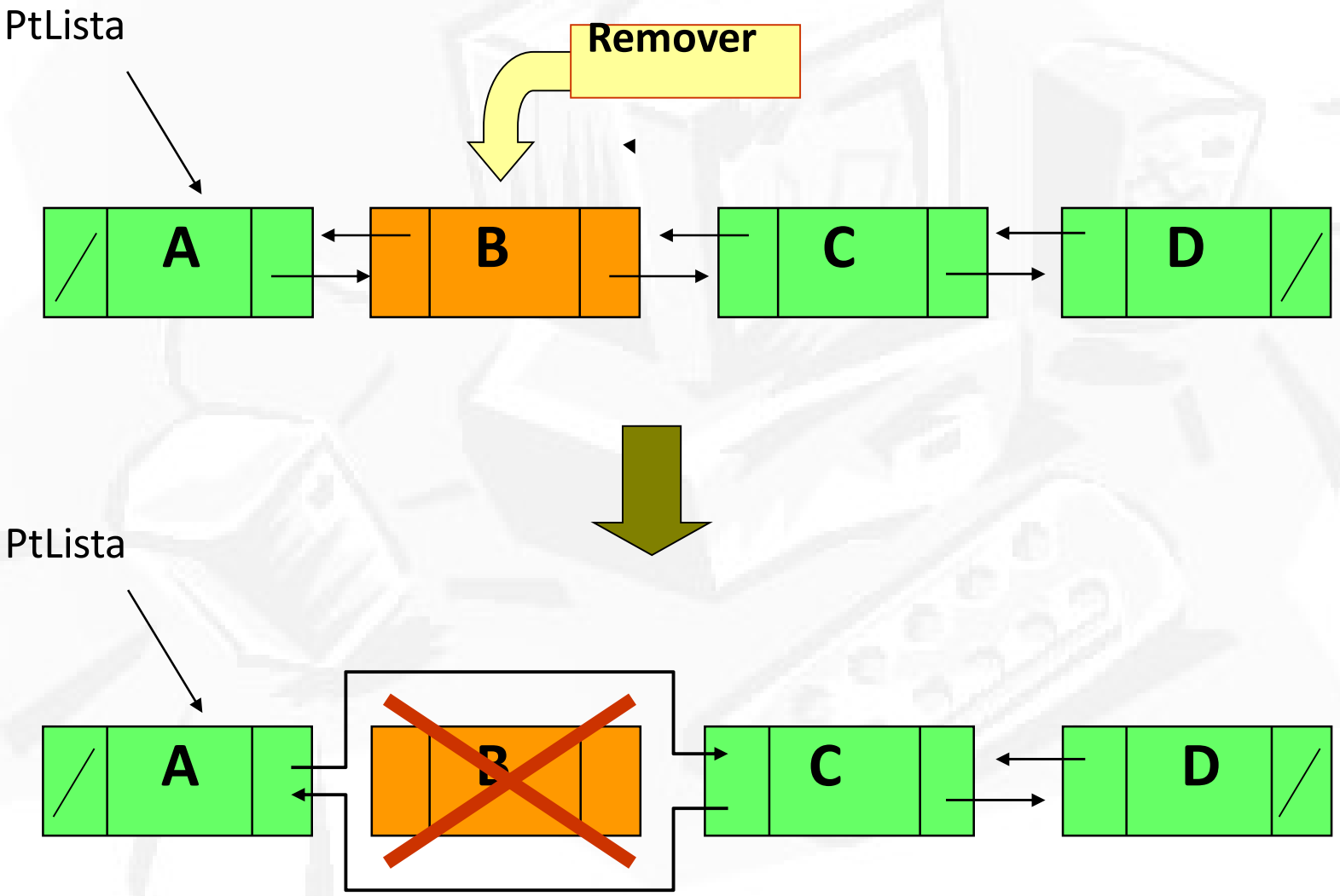
- Usar a função de busca para localizar o elemento
- acertar o encadeamento duplo, liberando o elemento ao final.

- Faça a remoção!!!

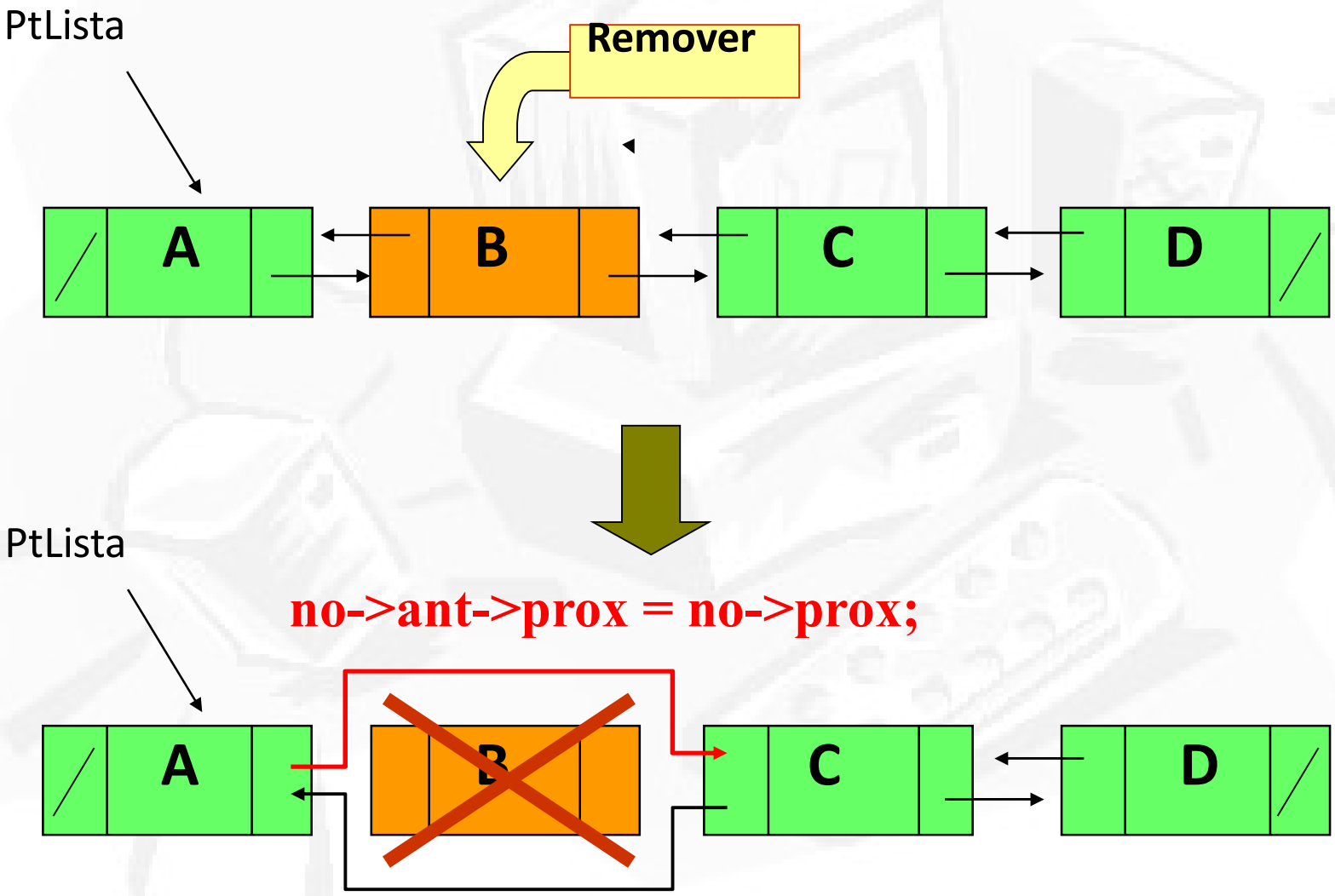
Remoção em lista duplamente encadeada



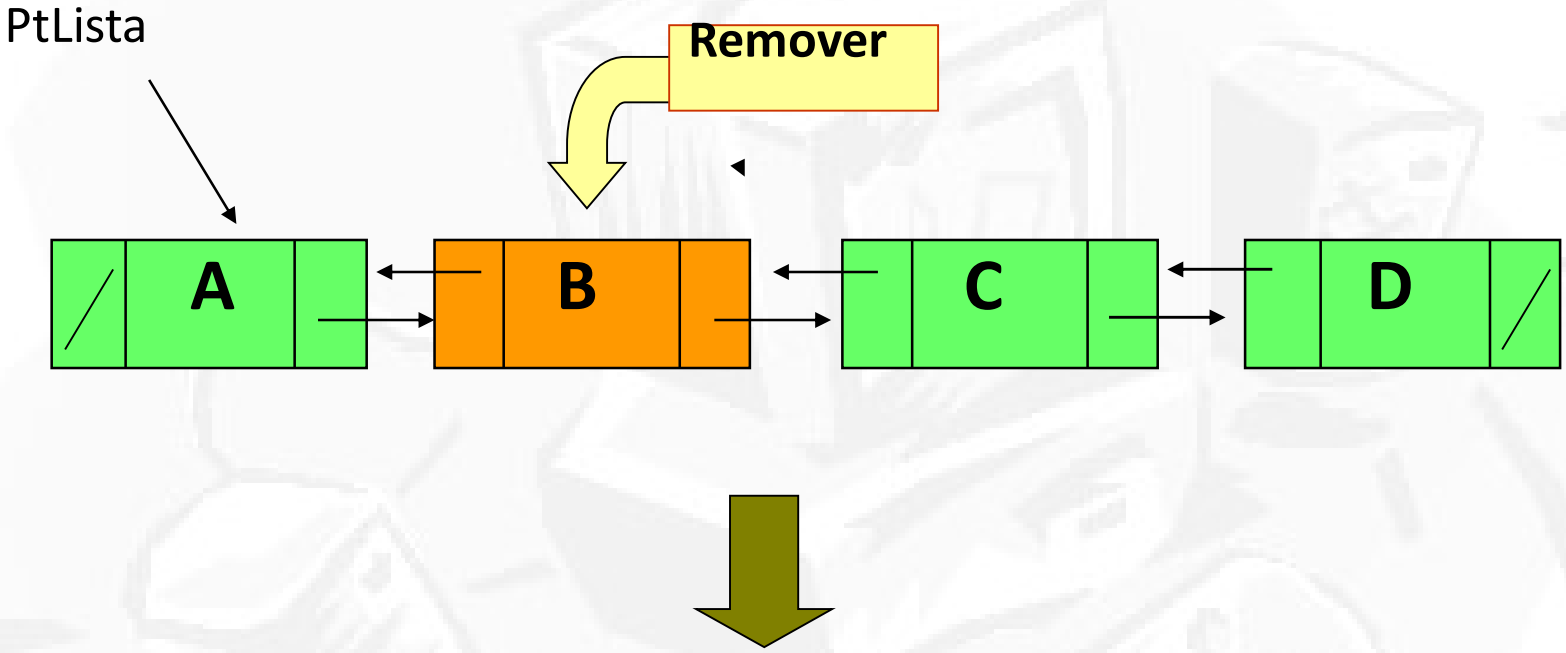
Remoção em lista duplamente encadeada



Remoção em lista duplamente encadeada

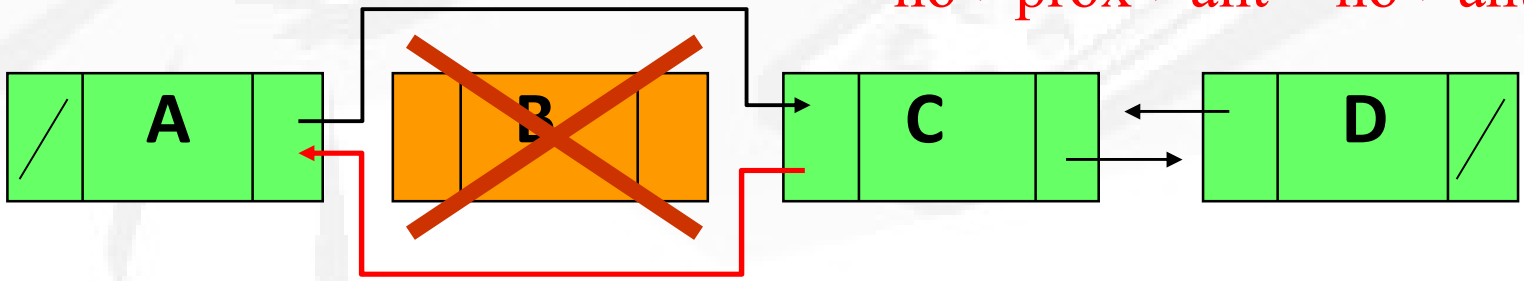


Remoção em lista duplamente encadeada



no->ant->prox = no->prox;

if(no->prox != NULL)
no->prox->ant = no->ant;

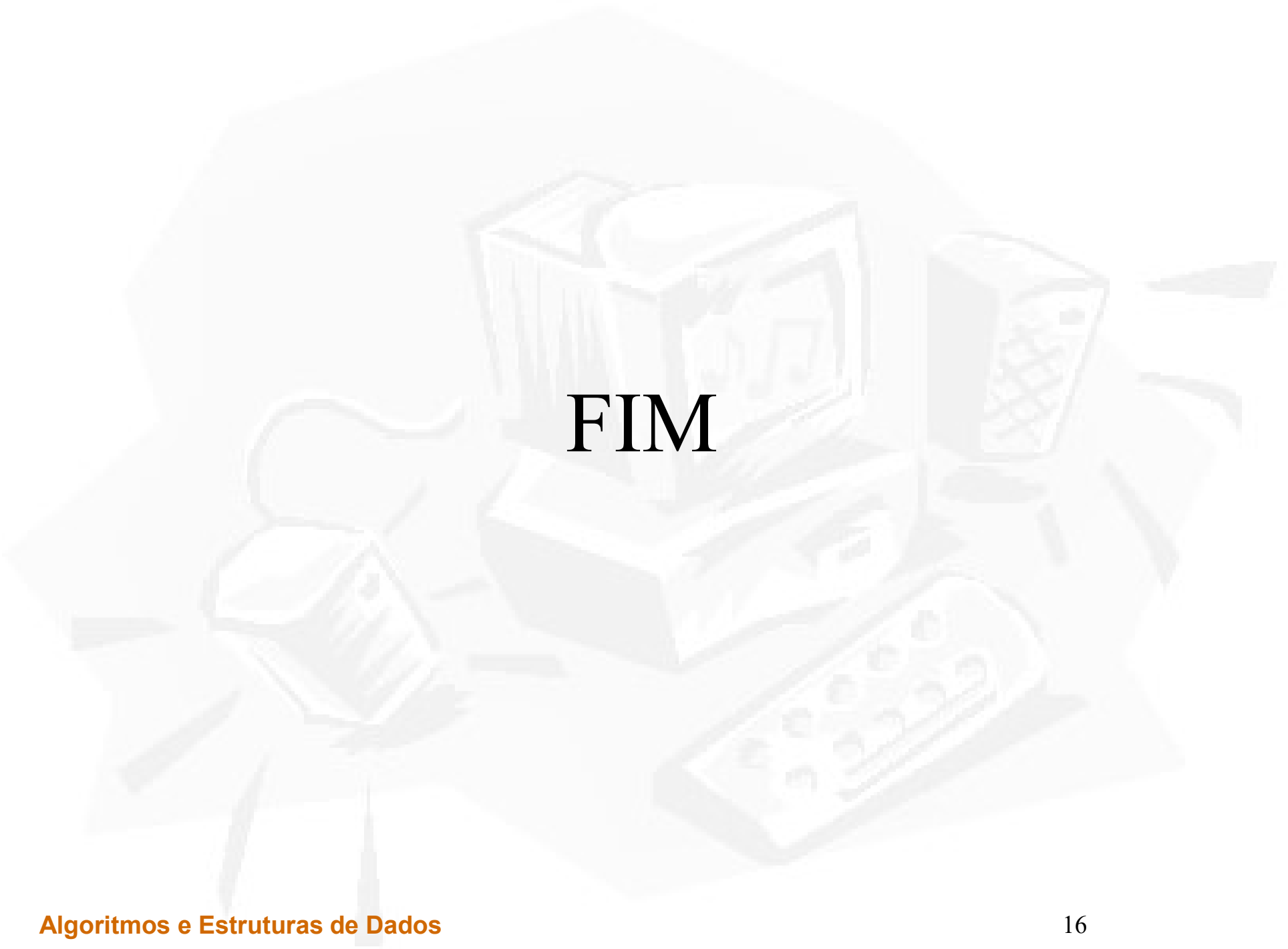


- Escreva uma função que remova de uma lista duplamente encadeada todos os elementos que contêm o valor de y.

```
PLista2 remove_lista(PLista2 li, int mat){
if(li == NULL)     return 0;
   PLista2 no=li;
   while(no != NULL && no->info != mat){
    no = no->prox;  }

if(no == NULL)   return 0;  //não achou
   if(no->ant == NULL)//remover o primeiro
      li = no->prox;
   else
      no->ant->prox = no->prox;

   if(no->prox != NULL) // remove do meio ou fim
       no->prox->ant = no->ant;
   free(no);
  return li;
```



FIM