ESTRUTURA DE DADOS

Prof.ª Priscilla Abreu

priscilla.braz@rj.senac.br

Estrutura de dados

Roteiro de Aula

- Objetivo da aula
- Listas duplamente encadeadas
- Exercícios

Estrutura de dados

Objetivo da aula

Identificar situações em que são necessárias o uso de estruturas dinâmicas e encadeadas e compreender a manipulação dessas estruturas.

Competência:

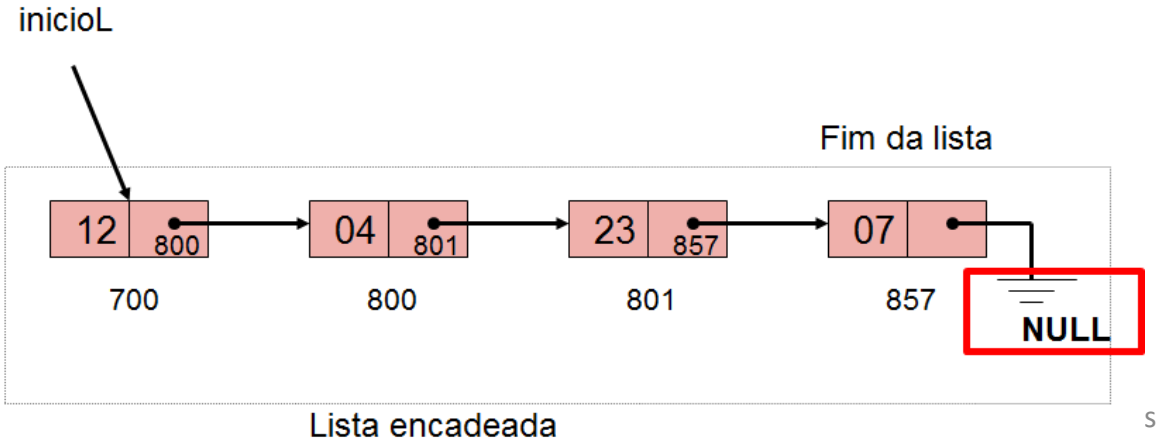
Desenvolver estruturas de dados lineares e não lineares.

REVISANDO...

Estrutura de dados

LISTAS SIMPLESMENTE ENCADEADAS - REVISÃO

- Nó da lista é representado por pelo menos dois campos:
  - a informação armazenada;
  - o ponteiro para o próximo elemento da lista.
- a lista é representada por um ponteiro para o primeiro nó;
- o campo próximo do último elemento é NULL.




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

no *inicioL;
```

LISTA DUPLAMENTE ENCADEADA

Estrutura de dados

LISTAS ENCADEADAS

Listas Simplesmente Encadeadas

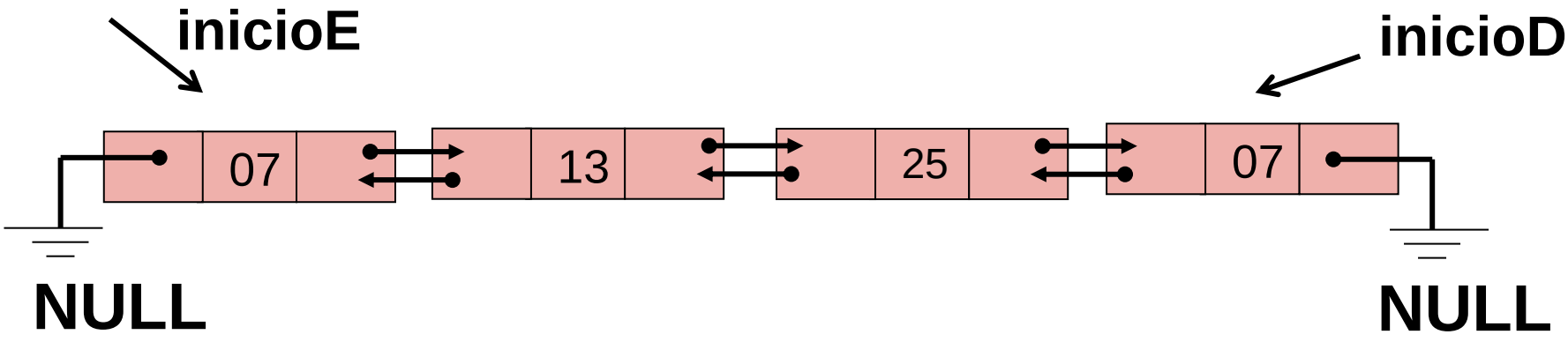
Limitações:

- Único sentido de percurso
- Necessidade de conhecer antecessor para inserir ou remover em uma k-ésima posição

Alternativa: Listas Duplamente Encadeadas!

Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

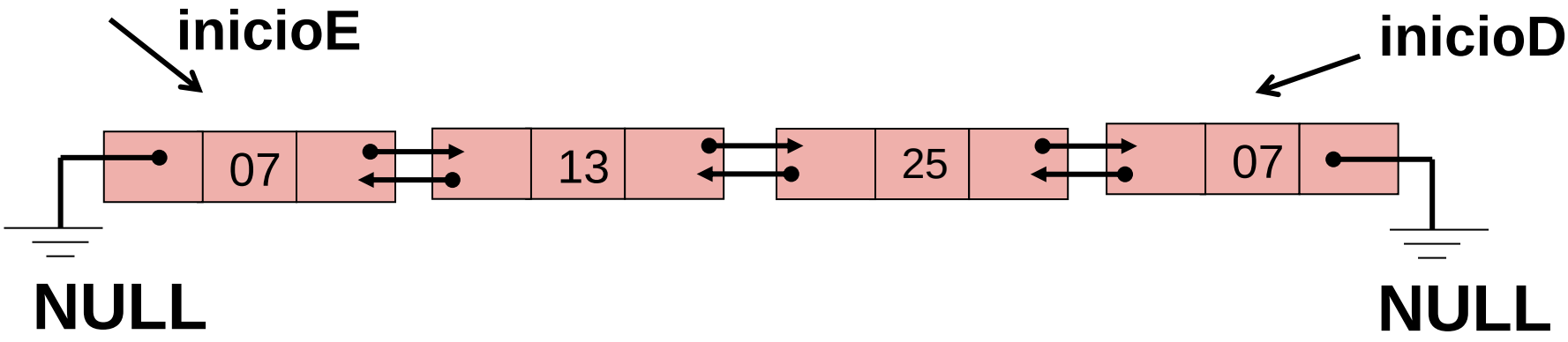


Estrutura básica da lista

Cada nó possui dois ponteiros: um para o elemento anterior e outro para o próximo elemento (ant e prox).

Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

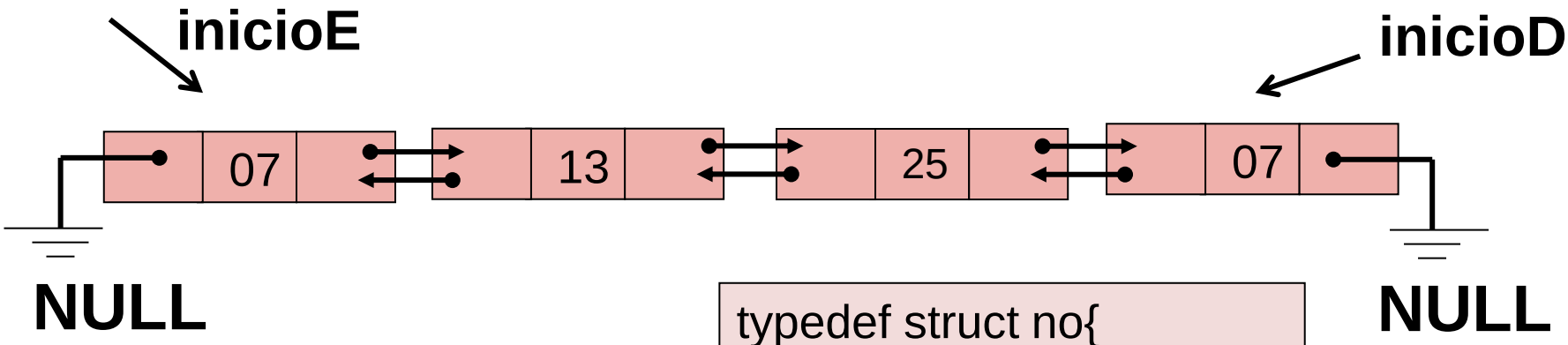


Estrutura básica da lista

Além disso, a lista possui uma referência para o início e outra para o final.

Estrutura de dados

LISTA DUPLAMENTE ENCADEADA



Estrutura básica da lista

```
typedef struct no{
      int info;
      struct no *prox, *ant;
}no;

no *inicioE;
no *inicioD;
```

Estrutura de dados

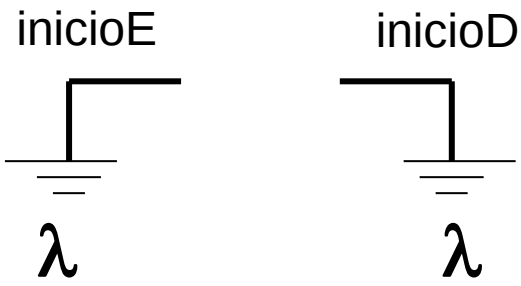
LISTA DUPLAMENTE ENCADEADA

```
#include <stdio.h>
#include <stdlib.h>
typedef struct no{
        int info;
        struct no *ant, *prox;
}no;
no *inicioE;
no *inicioD;
int main(){
}
```

Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

Inicializar lista




```
void inicializa_lista ()
{
        inicioD = NULL;
        inicioE  =  NULL;

}
```

Estrutura de dados

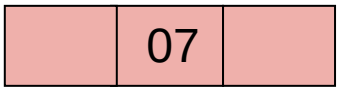
LISTA DUPLAMENTE ENCADEADA

Criar a estrutura de um nó para posterior inserção:

1. SOLICITAR ALOCAÇÃO

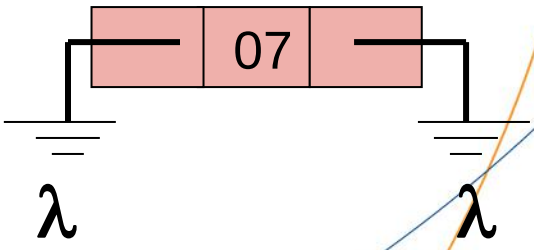
aux

2. ATRIBUIR VALOR AO CAMPO INFO



aux

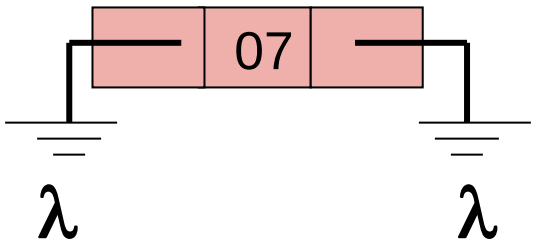
3. ATRIBUIR VALOR NULL AOS CAMPOS PROX E ANT.



Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

```
no* cria_no (int valor){
        no *aux;
        aux = (no*) malloc (sizeof (no));
        if (aux == NULL) {
                printf("ERRO!!!");
                exit(0);
        }
        else{
                aux -> info= valor;
                aux -> ant = NULL;
                aux -> prox = NULL;
                return aux;
        }
}
```



Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

```
int lista_vazia () {
        if (inicioD == NULL)
                return 1;
        return 0;
}
```

LISTA DUPLAMENTE ENCADEADA

INSERÇÃO NA LISTA:

Para inserir um elemento na lista precisamos pensar em alguns pontos:

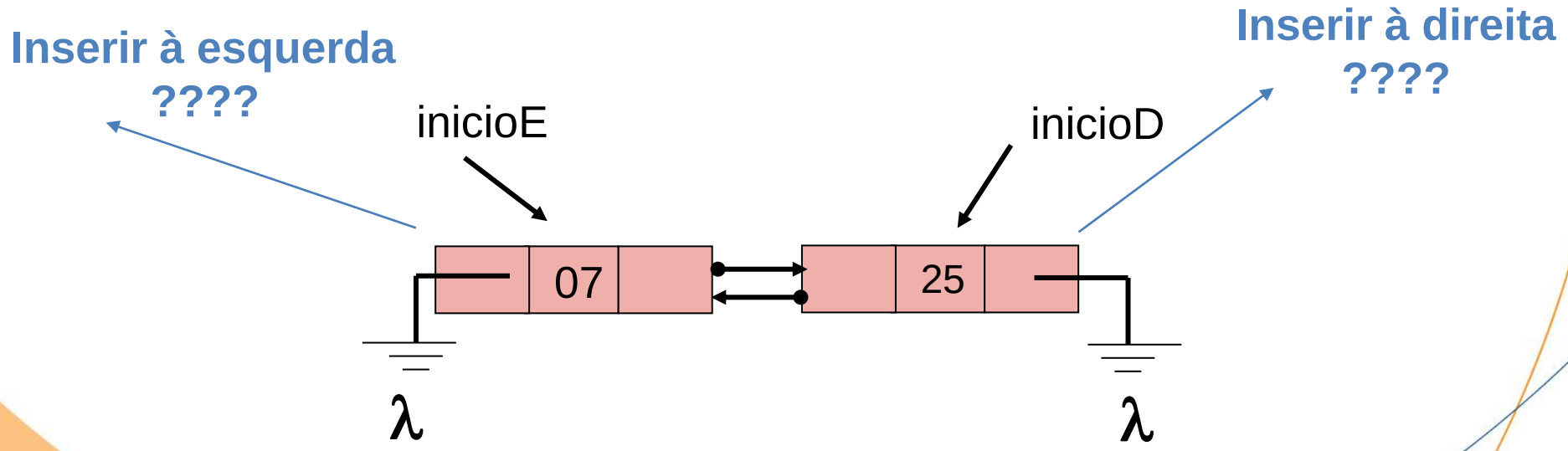
- É o primeiro elemento?
- Por qual lado ocorrerá a inserção?

Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

Inserir elemento na lista:

- Por qual lado ocorrerá a inserção?



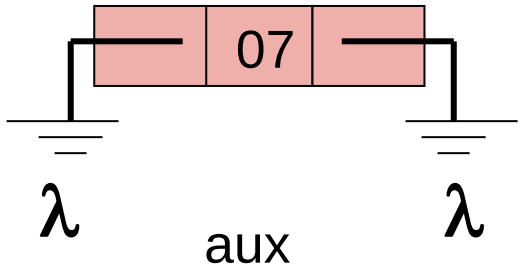
Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

Inserir elemento na lista:

Solicitar a alocação do espaço para um novo nó...

criar_no(valor)

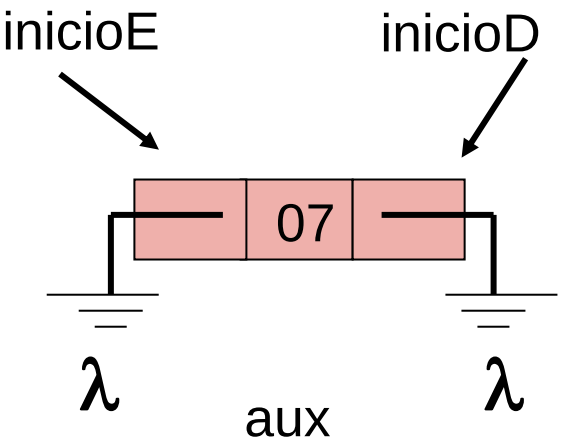


LISTA DUPLAMENTE ENCADEADA

Inserir elemento na lista:

- 1º elemento?

Apontar os ponteiros inicioD e inicioE para o elemento criado:



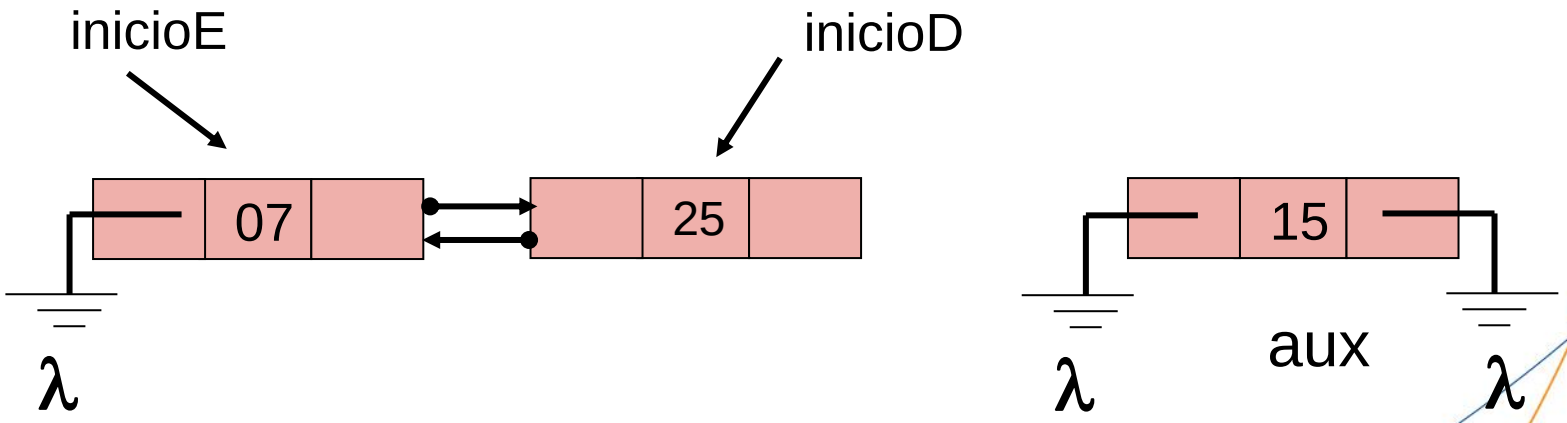
inicioD = aux
inicioE = aux

LISTA DUPLAMENTE ENCADEADA

Inserir elemento à direita:

Suponha que não seja o primeiro elemento.

Ligar os ponteiros entre o elemento da direita e o novo nó...



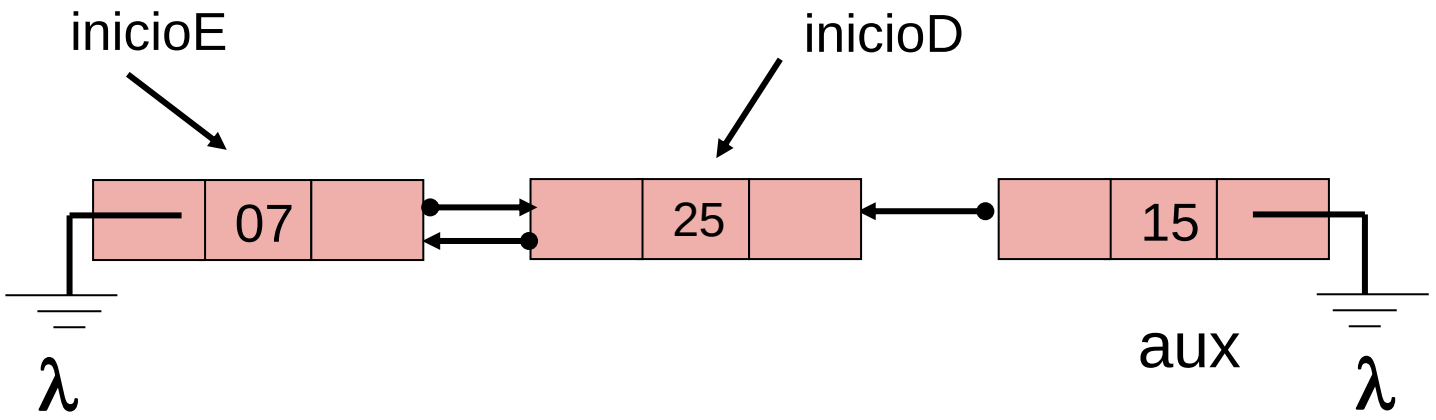
Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

Inserir elemento à direita:

Ligar os ponteiros entre o elemento da direita e o novo nó...

aux->ant = inicioD



Estrutura de dados

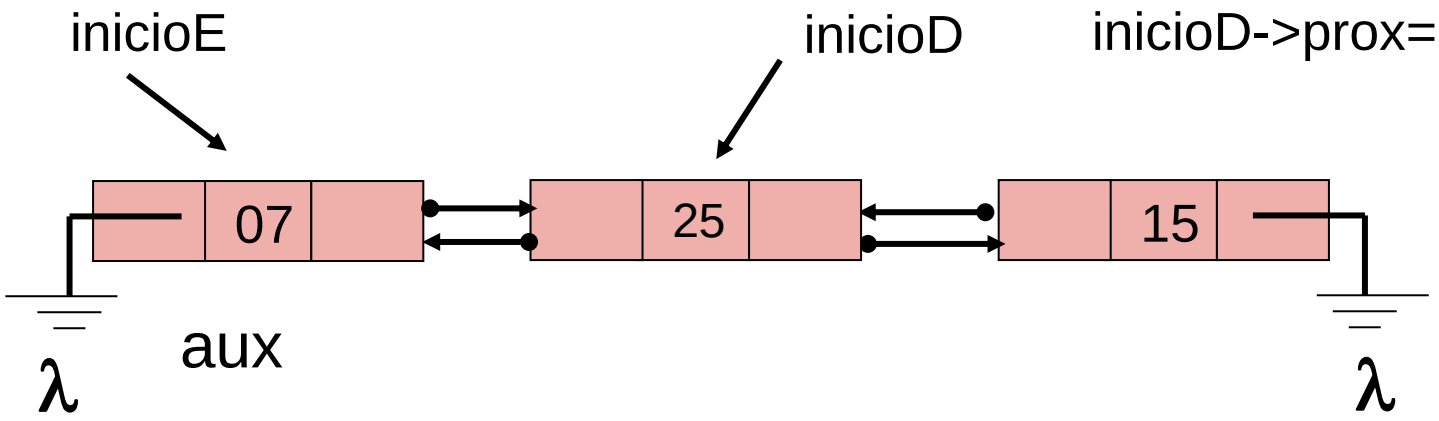
LISTA DUPLAMENTE ENCADEADA

Inserir elemento à direita:

Ligar os ponteiros entre o elemento da direita e o novo nó...

aux->ant = inicioD

inicioD->prox= aux



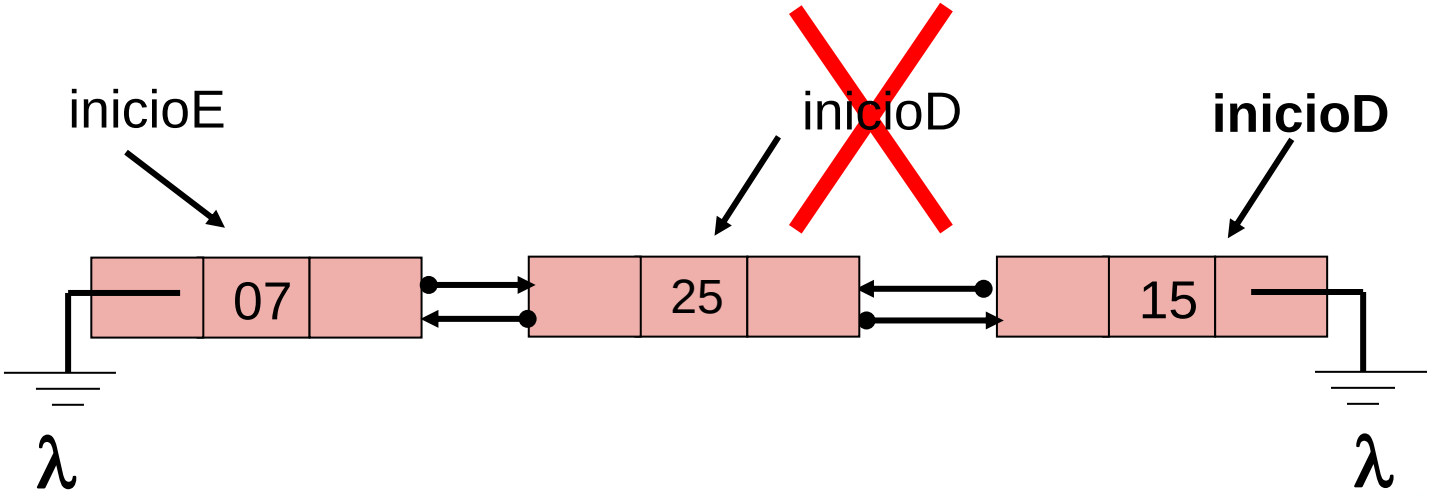
Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

Inserir elemento à direita:

Atualizar o ponteiro inicioD

inicioD = aux



Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

Inserir elemento à direita:

```
void inserir_Dir(int valor){
        no* aux;
        aux = cria_no(valor);
        if (lista_vazia()){
                inicioD = aux;
                inicioE = aux;
        }
        else{
                inicioD->prox = aux;
                aux->ant = inicioD;
                inicioD = aux;
        }
}
```

Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

IMPRESSÃO: Pela esquerda ou direita???

```
void percorrer_Esq () {
        no * aux;
        aux = inicioE;
        while (aux!= NULL) {
                printf("%d",aux->info);
                aux = aux->prox;
        }
}
```

Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

IMPRESSÃO: Pela esquerda ou direita???

```
void percorrer_Dir () {
        no * aux;
        aux = inicioD;
        while (aux!= NULL) {
                printf("%d",aux->info);
                aux = aux->ant;
        }
}
```

Estrutura de dados

EXERCÍCIO

1) Faça um procedimento para inserir um elemento na lista pela esquerda.

2) Faça uma função (buscaLDE) que realize a busca de um determinado valor de uma Lista Duplamente Encadeada e retorne ao programa que a chamou o nó que possui esse valor.

Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

REMOÇÃO NA LISTA:

Para remover um elemento na lista precisamos pensar em alguns pontos:

1. Lista está vazia?
2. O elemento a ser removido encontra-se na lista?

```
if (!lista_vazia()) {
    aux = busca_no(valor);
    ...
}
```

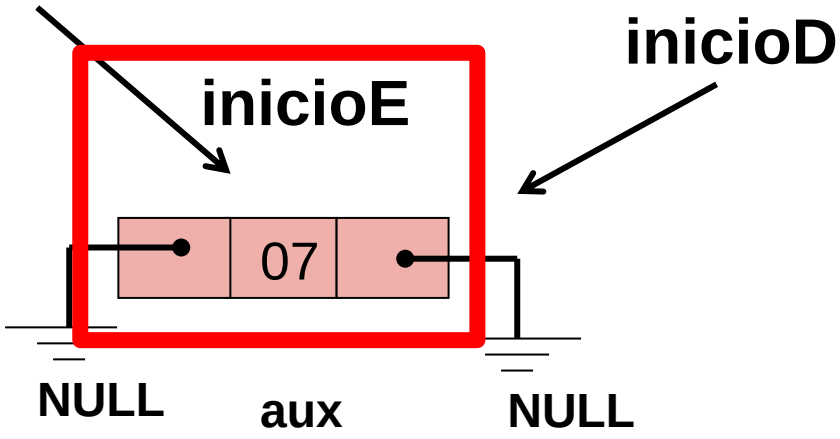
Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

REMOÇÃO NA LISTA:

3. A lista tem um único elemento?

```
aux = busca_no(valor);
...
inicioE = NULL;
inicioD = NULL;
free(aux);
...
```



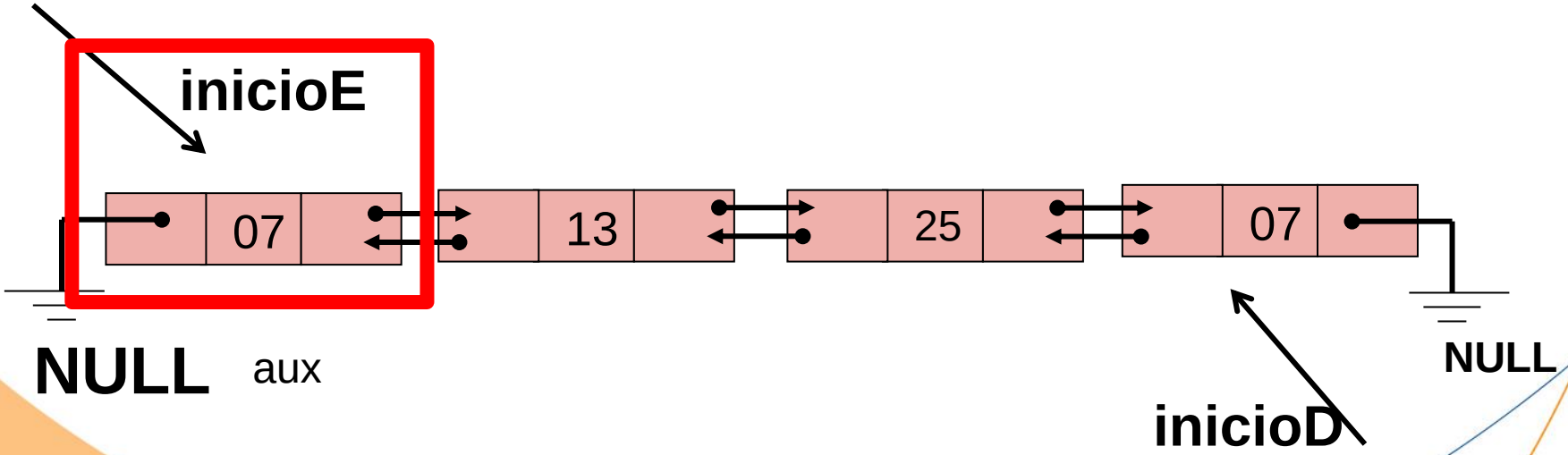
Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

```
aux = busca_no(valor);
...
```

REMOÇÃO NA LISTA:

4. **O elemento a ser removido é o primeiro da esquerda?**



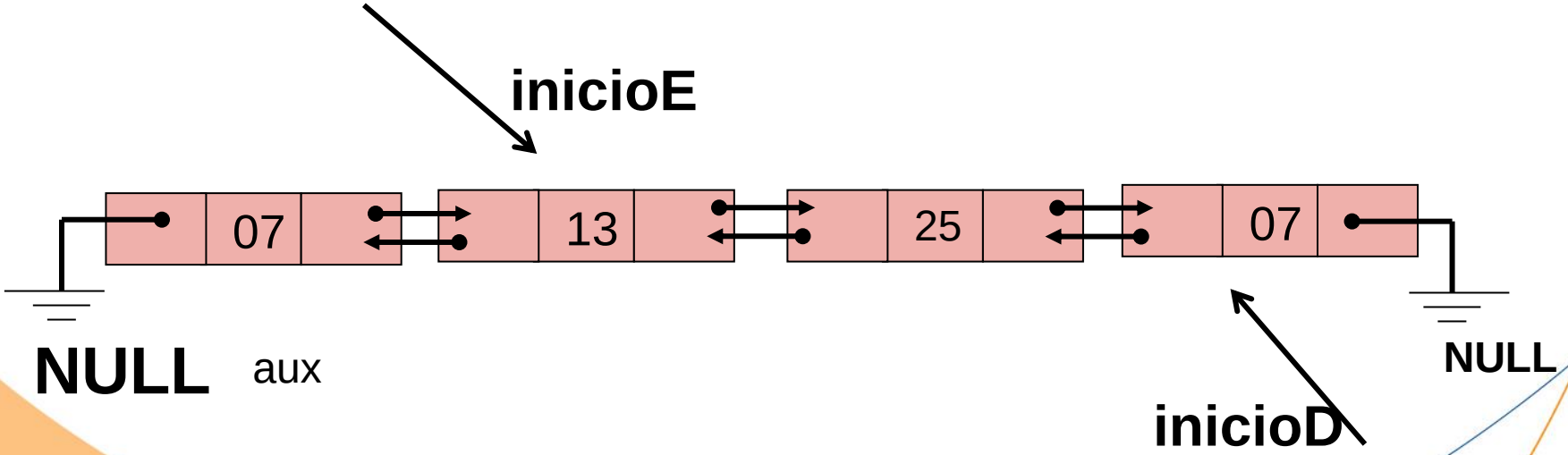
Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

```
aux = busca_no(valor);
...
inicioE = aux->prox;
```

REMOÇÃO NA LISTA:

4. **O elemento a ser removido é o primeiro da esquerda?**



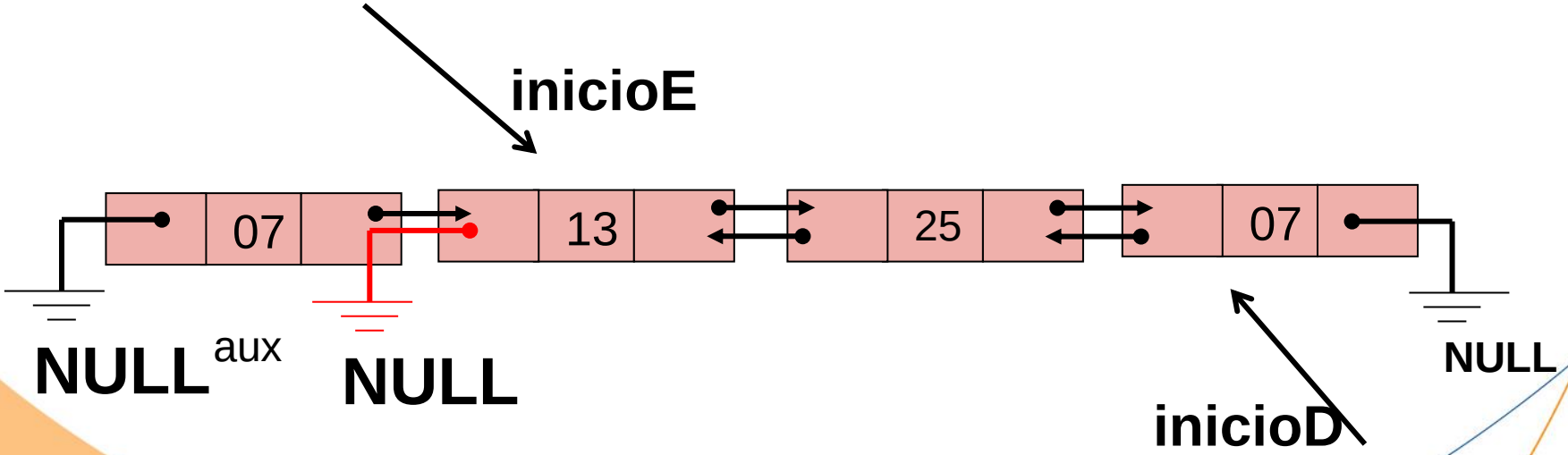
Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

```
aux = busca_no(valor);
...
inicioE = aux->prox;
inicioE->ant = NULL;
```

REMOÇÃO NA LISTA:

4. **O elemento a ser removido é o primeiro da esquerda?**



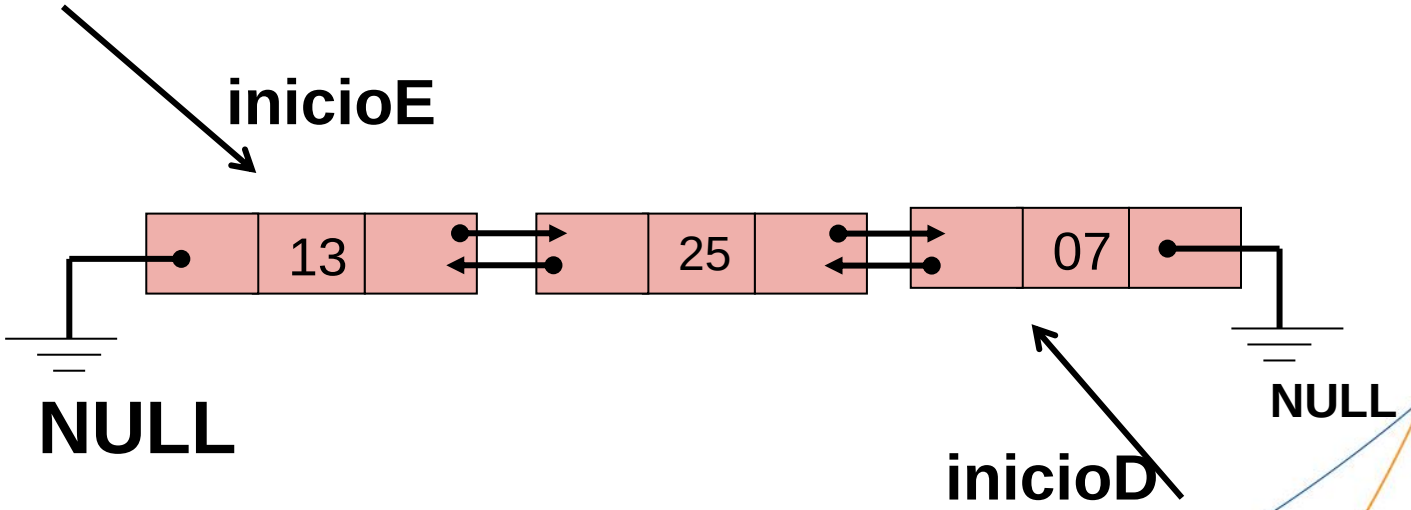
Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

```
aux = busca_no(valor);
...
inicioE = aux->prox;
inicioE->ant = NULL;
free(aux);
```

REMOÇÃO NA LISTA:

4. **O elemento a ser removido é o primeiro da esquerda?**



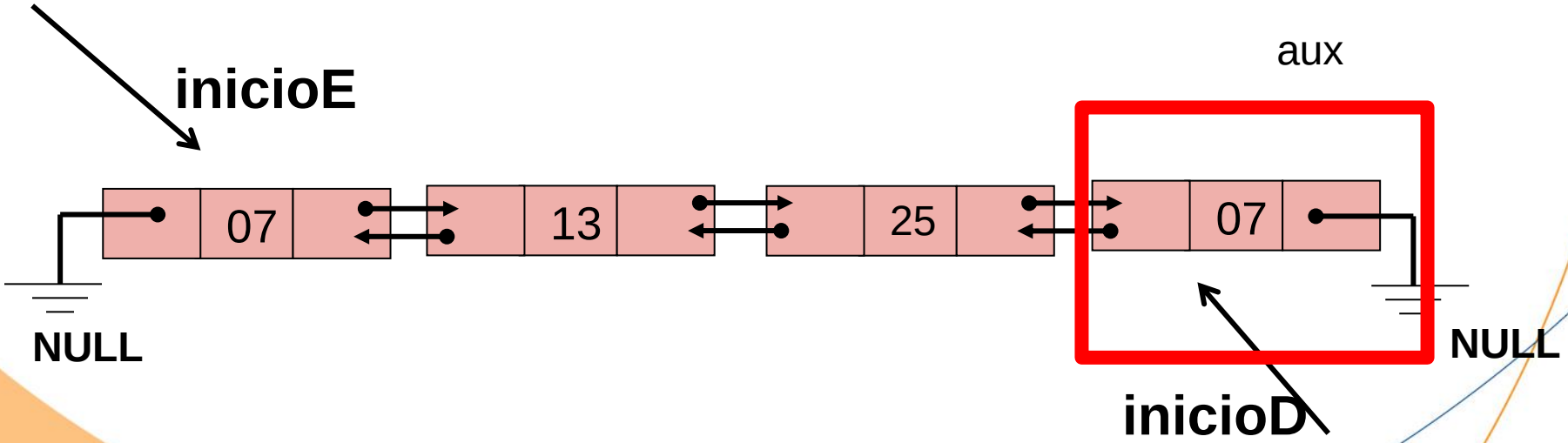
Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

```
aux = busca_no(valor);
...
```

REMOÇÃO NA LISTA:

5. **O elemento a ser removido é o primeiro da direita?**



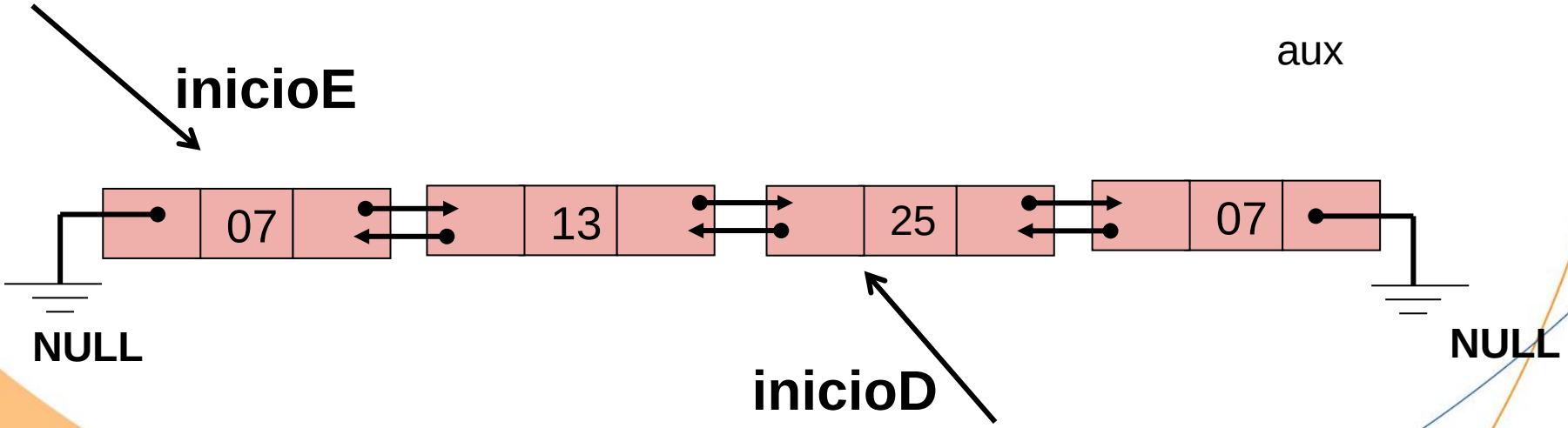
Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

```
aux = busca_no(valor);
...
inicioD = aux->ant;
```

REMOÇÃO NA LISTA:

5. **O elemento a ser removido é o primeiro da direita?**



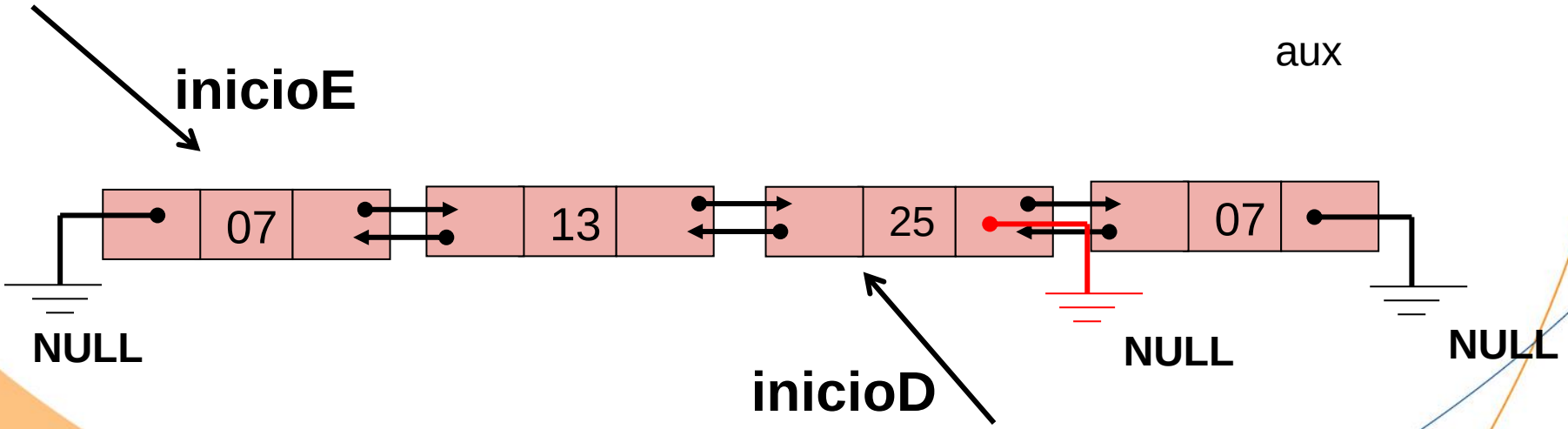
Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

```
aux = busca_no(valor);
...
inicioD = aux->ant;
inicioD->prox = NULL;
```

REMOÇÃO NA LISTA:

5. **O elemento a ser removido é o primeiro da direita?**



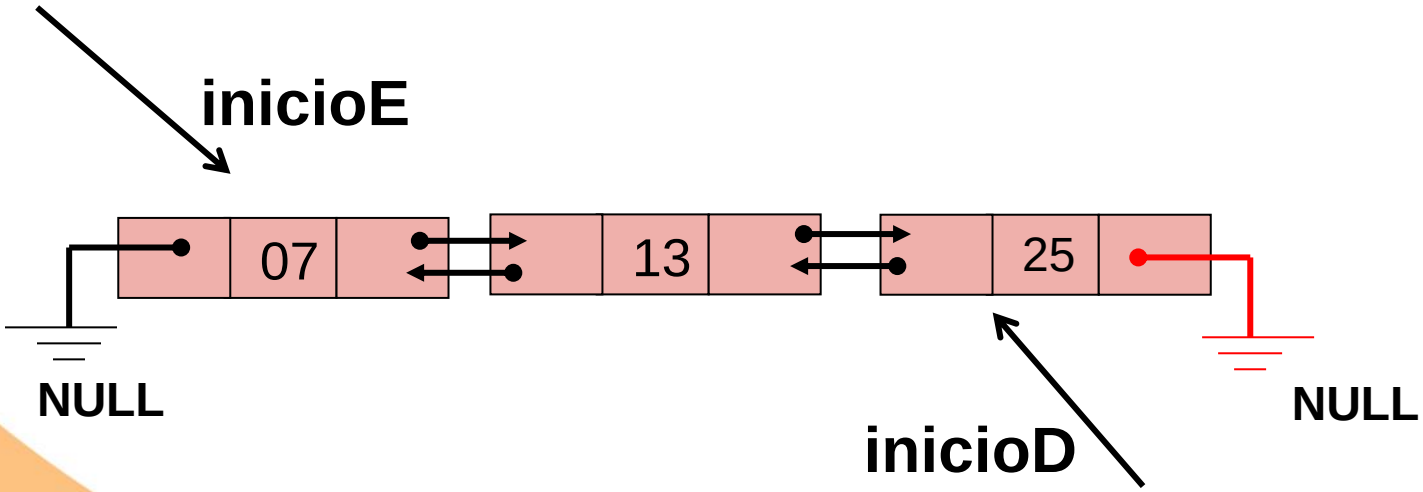
Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

```
aux = busca_no(valor);
...
inicioD = aux->ant;
inicioD->prox = NULL;
free(aux);
```

REMOÇÃO NA LISTA:

5. **O elemento a ser removido é o primeiro da direita?**



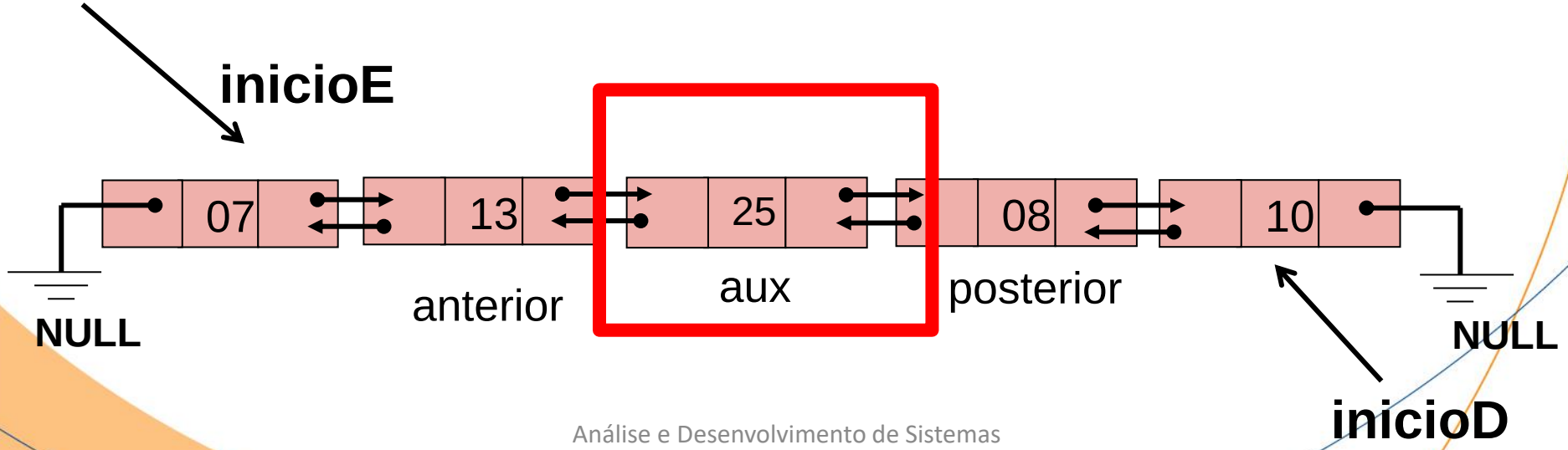
Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

```
aux = busca_no(valor);
anterior = aux->ant;
posterior = aux->prox;
```

REMOÇÃO NA LISTA:

6. **O elemento a ser removido está em qualquer outra posição?**



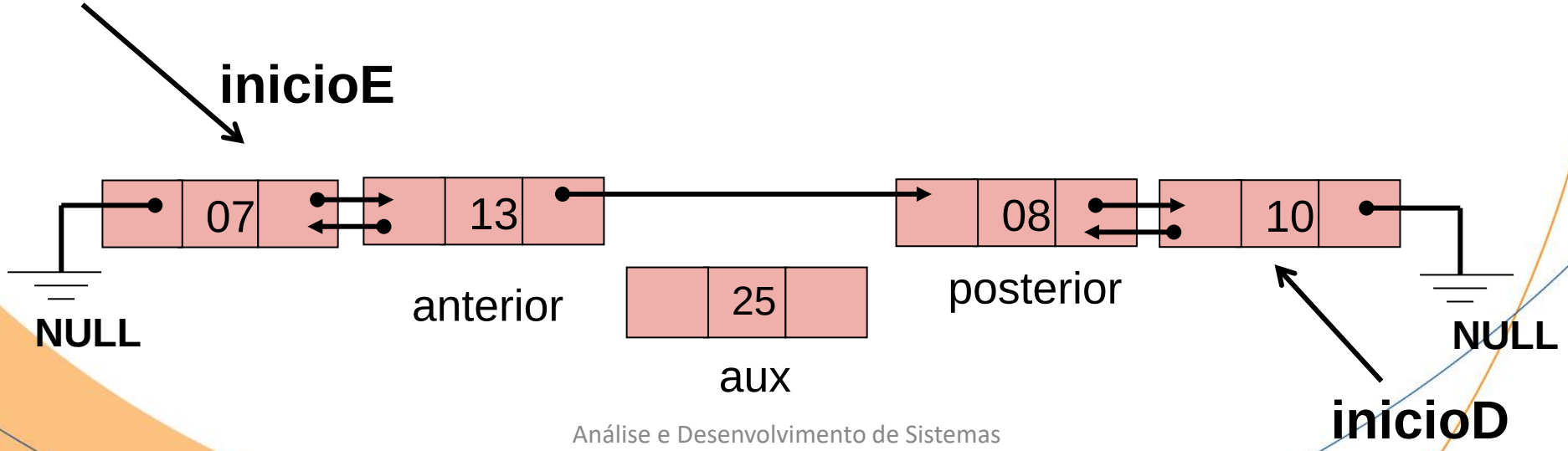
Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

```
aux = busca_no(valor);
anterior = aux->ant;
posterior = aux->prox;
anterior->prox = posterior;
```

REMOÇÃO NA LISTA:

6. **O elemento a ser removido está em qualquer outra posição?**



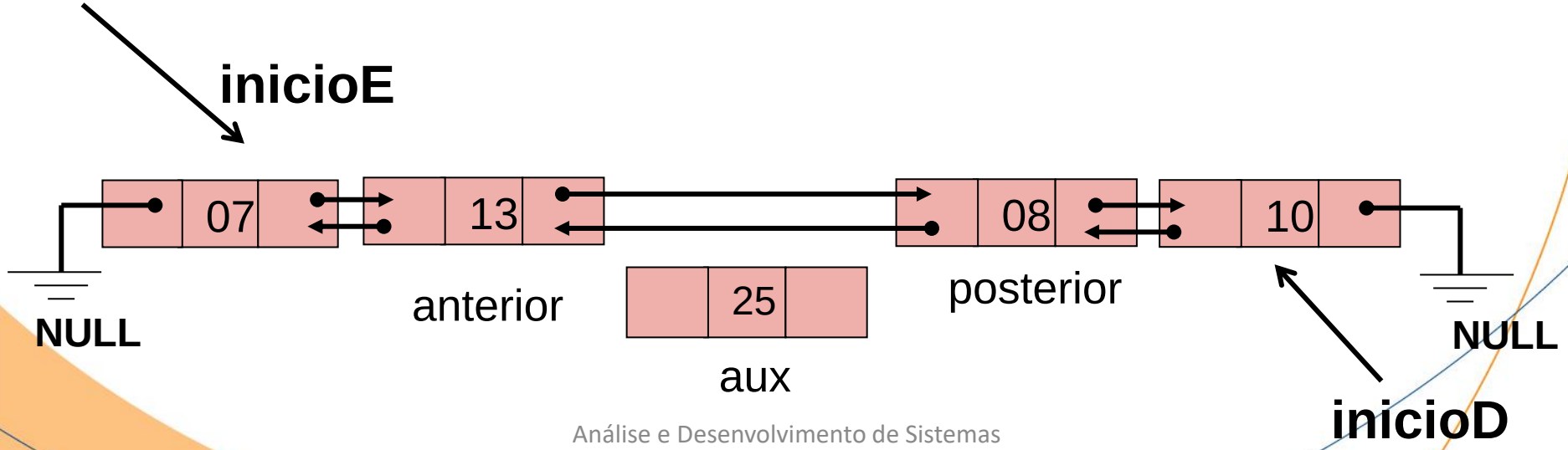
Estrutura de dados

LISTA DUPLAMENTE ENCADEADA

```
aux = busca_no(valor);
anterior = aux->ant;
posterior = aux->prox;
anterior->prox = posterior;
posterior->ant = anterior;
```

REMOÇÃO NA LISTA:

6. **O elemento a ser removido está em qualquer outra posição?**



Estrutura de dados

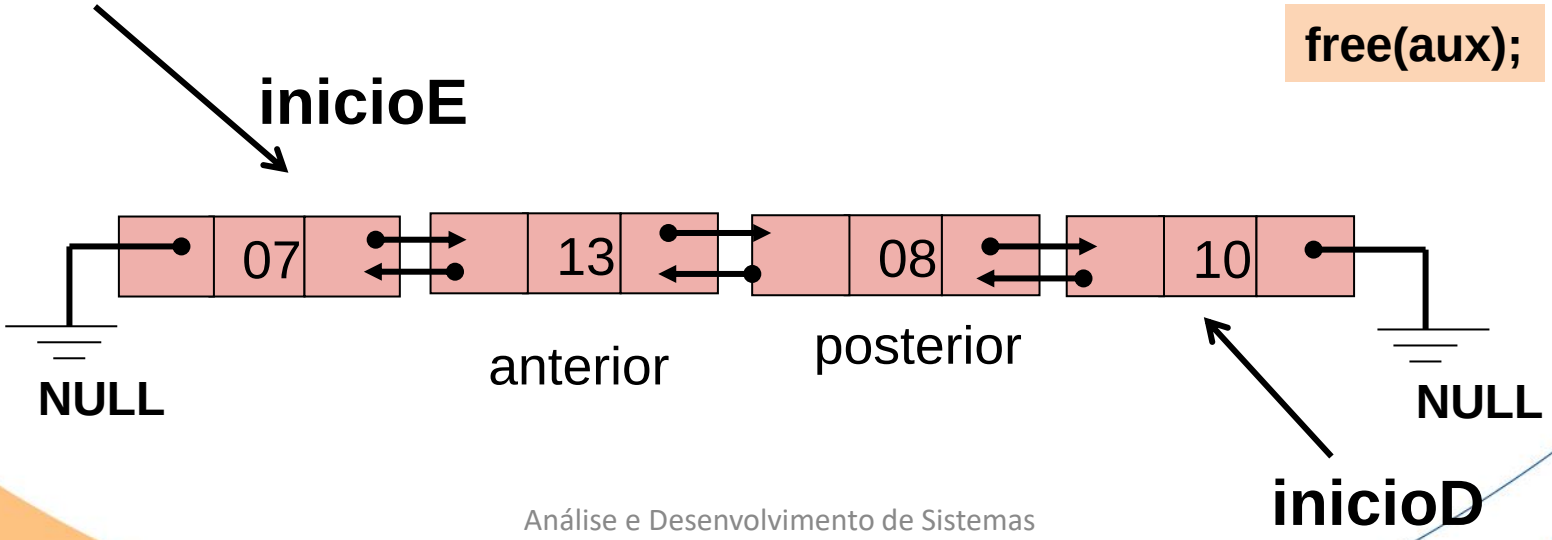
LISTA DUPLAMENTE ENCADEADA

```
aux = busca_no(valor);
anterior = aux->ant;
posterior = aux->prox;
anterior->prox = posterior;
posterior->ant = anterior;
```

REMOÇÃO NA LISTA:

6. **O elemento a ser removido está em qualquer outra posição?**

free(aux);



Estrutura de dados

LISTAS DUPLAMENTE ENCADEADAS

```
void remover (int valor) {
    no *aux;
    if (lista_vazia())
        printf("Lista vazia!");
    else{
        aux = buscaLDE(valor);
        if (aux == NULL)
            printf("Elemento não está na lista!");
        else{
```

Função que busca pelo elemento a ser removido.

Não encontrou o elemento.

Estrutura de dados

LISTAS DUPLAMENTE ENCADEADAS

```
if (inicioE == inicioD)  {
        inicioD = NULL;
        inicioE=NULL;
}
else if (aux->ant == NULL){
        inicioE = aux->prox;
        inicioE->ant=NULL;
}
```

Único elemento

Primeiro elemento da esquerda

Estrutura de dados

LISTAS DUPLAMENTE ENCADEADAS

Primeiro elemento da direita

Qualquer outra posição

```
        else
            if (aux->prox == NULL) {
                inicioD = aux->ant;
                inicioD->prox = NULL;
            }
            else {
                no *anterior = aux->ant;
                no *posterior = aux->prox;
                anterior->prox = posterior;
                posterior->ant = anterior;
            }
    free(aux);
    }
}
```

Estrutura de dados

LISTAS DUPLAMENTE ENCADEADAS

- Lista que possibilita manipulação nos dois sentidos, através dos ponteiros *ant* e *prox*;
- Útil quando é preciso percorrer a lista na ordem inversa;
- Remoção de um elemento não precisa guardar anterior;
- Remoção de um elemento cujo ponteiro é informado não precisar percorrer a lista toda;
- Um conjunto maior de ligações precisam ser atualizadas.

DÚVIDAS?