Algoritmos e Estruturas de Dados I

Profª Priscilla Abreu

priscilla.abreu@ime.uerj.br

2022.1

Algoritmos e Estruturas de Dados I

Roteiro da aula

- Listas Simplesmente Encadeadas (remoção)
- Listas Duplamente Encadeadas (introdução)

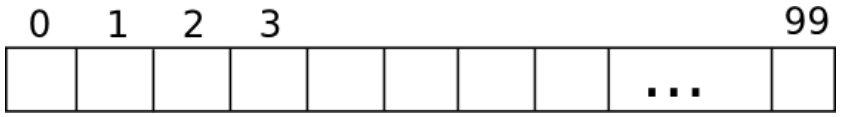
Revisando...

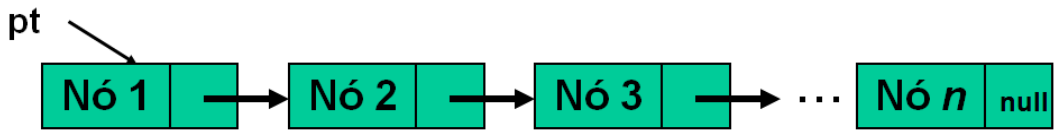
Algoritmos e Estruturas de Dados I

O tipo de armazenamento de uma lista linear pode ser classificado de acordo com a posição relativa na memória (contínua ou não) de cada dois nós consecutivos na lista.

Existem dois tipos de alocação:

- Alocação sequencial
- Alocação encadeada

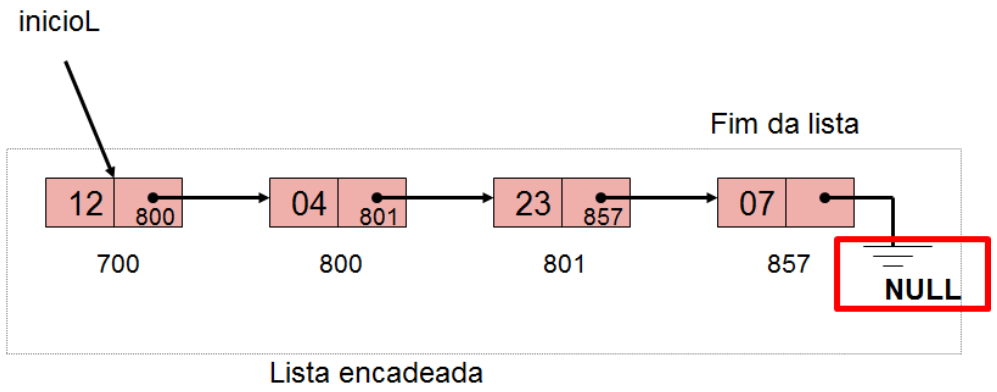




Algoritmos e Estruturas de Dados I

Alocação Dinâmica – Listas Encadeadas

- 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;
```

Algoritmos e Estruturas de Dados I

Alocação Dinâmica – Listas Encadeadas

```
#include <stdio.h>
#include <stdlib.h>
typedef struct no{
      int info;
      struct no *prox;
}no;
no *inicioL;
```

OPERAÇÕES:

- **Criar lista**
- **Lista vazia**
- **Inserir**
- **Percorrer**
- **Remover**

Algoritmos e Estruturas de Dados I

Alocação Dinâmica – Listas Encadeadas

```
void inicializa_lista () {
   inicioL = NULL;
}

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

Algoritmos e Estruturas de Dados I

Inserção em listas encadeadas

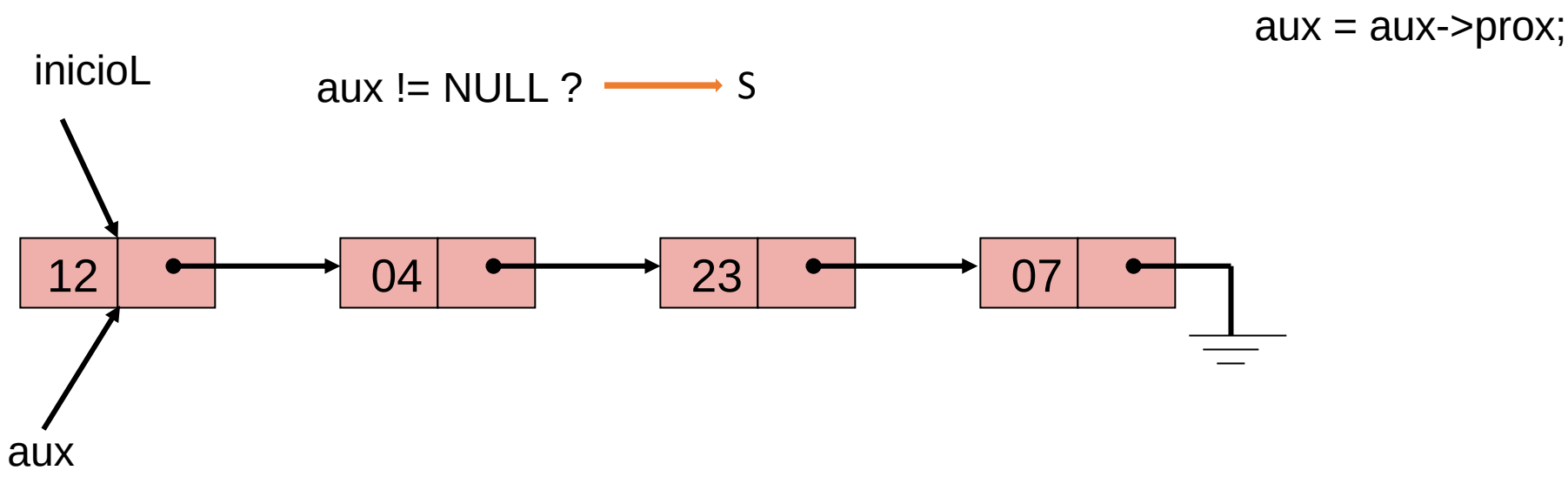
- Inserção no início

```
void inserir(int valor){
        no  *aux;
        aux = (no*) malloc(sizeof(no));
        if (aux != NULL){
                aux ->info= valor;
                aux -> prox = inicioL;
                inicioL = aux;
        }
}
```

Algoritmos e Estruturas de Dados I

Percurso

Percurso – condição de parada para percorrer



Algoritmos e Estruturas de Dados I

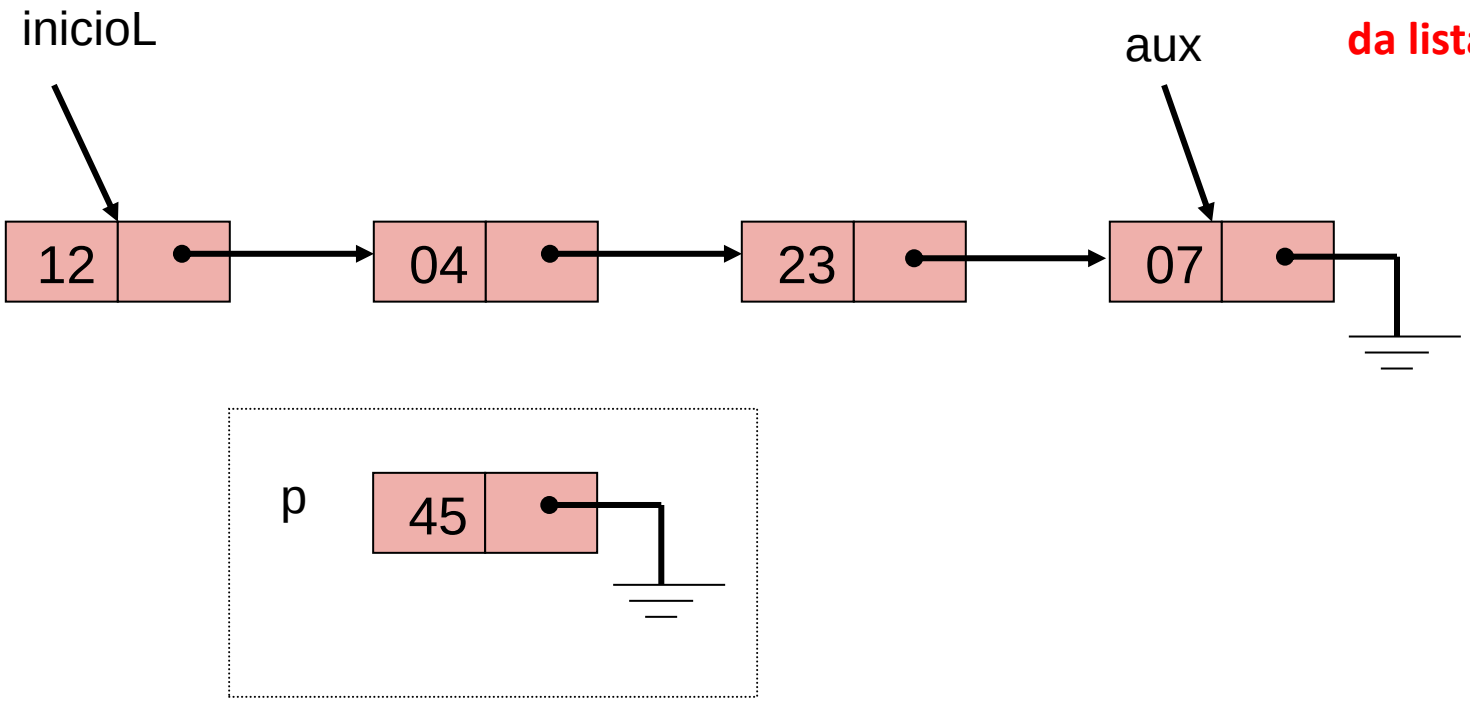
Percurso

```
void percorrer () {
        if(!lista_vazia()){
                no * aux;
                aux = inicioL;
                while (aux!= NULL) {
                        printf("%d", aux->info);
                        aux = aux->prox;
                }
        }
        else{
                printf("\n Lista vazia!\n");
        }
}
```

Algoritmos e Estruturas de Dados I

Inserção no final

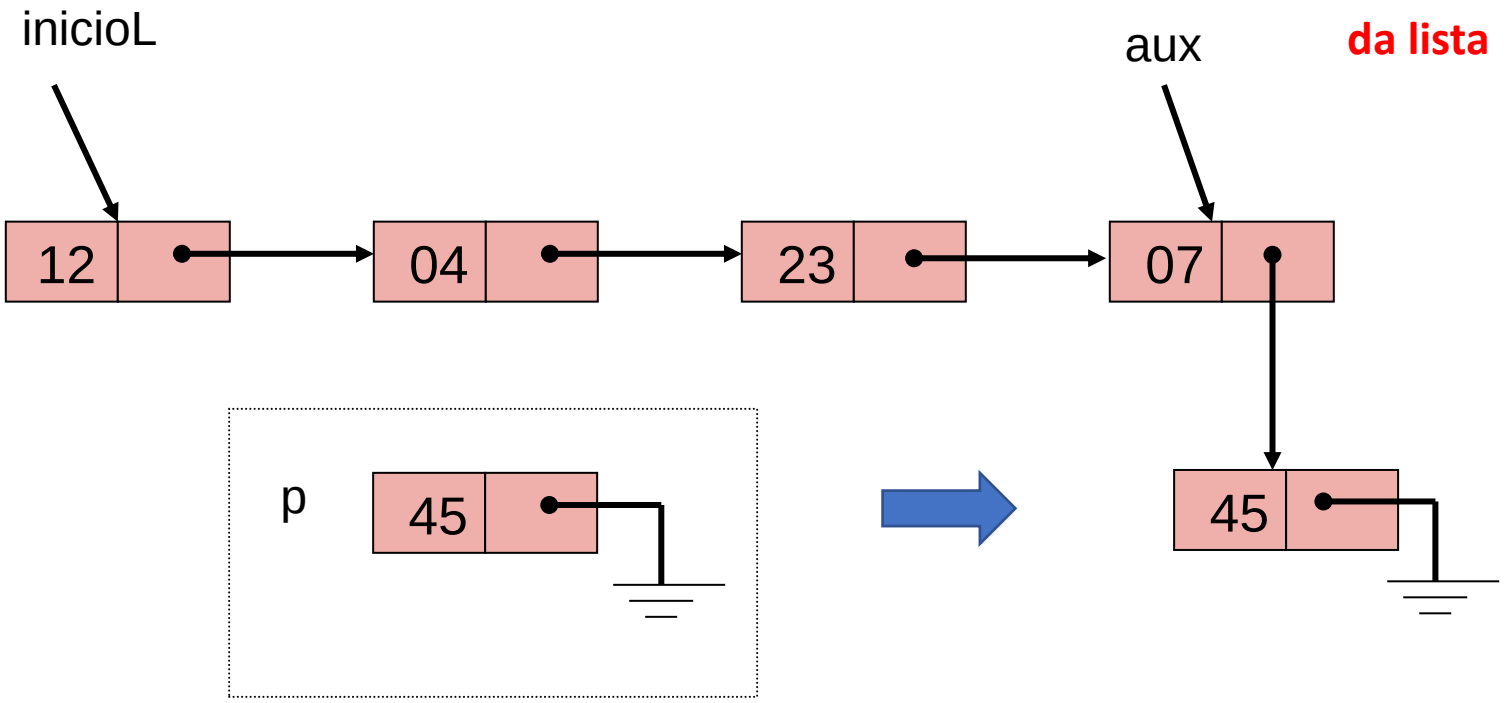
Ponteiro utilizado para encontrar o último elemento da lista



Algoritmos e Estruturas de Dados I

Inserção no final

Ponteiro utilizado para encontrar o último elemento da lista



Algoritmos e Estruturas de Dados I

Inserção no final

```
void inserir_fim (int valor) {
    no *aux, *p;
    aux = (no*) malloc(sizeof(no));
    if (aux != NULL){
            aux ->info= valor;
            aux -> prox = NULL;
            if (lista_vazia()){
                    inicioL=aux;
            }
            else{
                    p = inicioL;
                    while (p->prox != NULL)
                            p = p -> prox;
                    p->prox = aux;
            }
    }
}
```

Algoritmos e Estruturas de Dados I

EXERCÍCIO

Faça um programa para preenchimento de uma lista encadeada de números inteiros, utilizando as funções apresentadas. Você deve apresentar ao usuário um menu com as seguintes opções:

1- Inserir
2- Exibir a lista
3- Sair

Como remover um elemento específico da lista?

Simulação

Algoritmos e Estruturas de Dados I

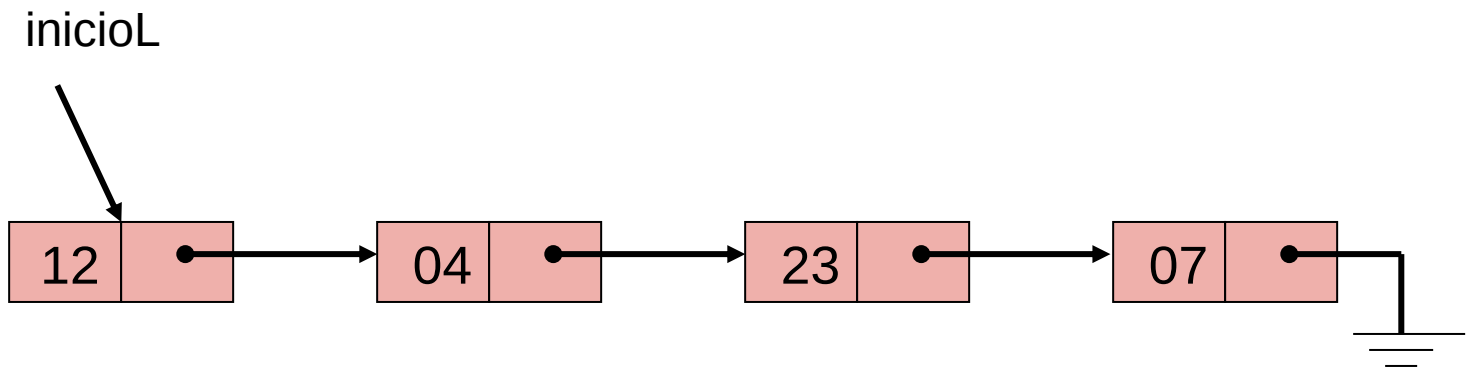
REMOÇÃO

Antes de remover um elemento é necessário verificar se há elementos na lista e procurar pelo elemento que deseja remover, além de obter informações de qual elemento está antes do elemento a ser removido, para que as referências dos ponteiros possam ser atualizadas.

Algoritmos e Estruturas de Dados I

REMOÇÃO: passos

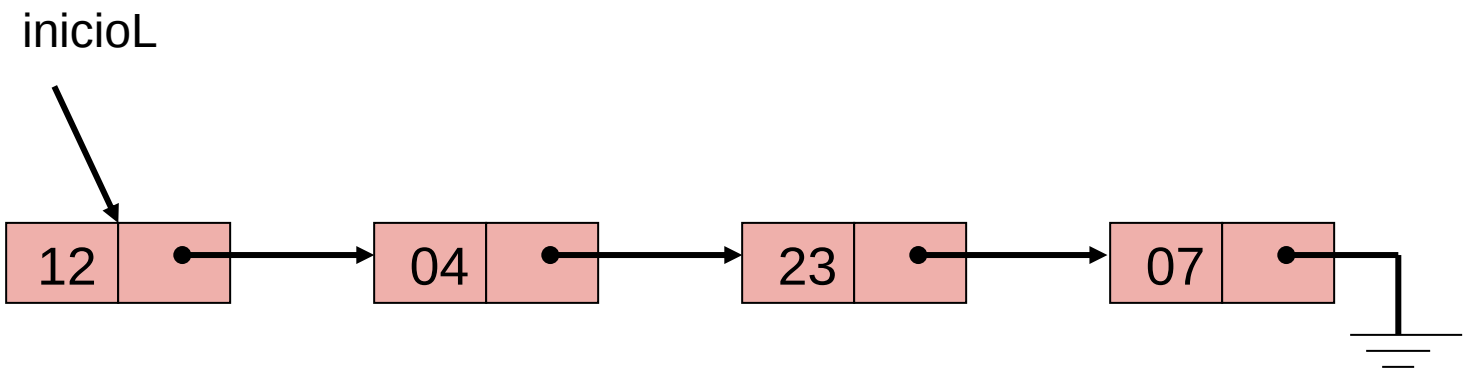
- Lista vazia???
- Busca pelo elemento;
- Referência do anterior ao elemento desejado;
- Casos de remoção.



Algoritmos e Estruturas de Dados I

REMOÇÃO - casos

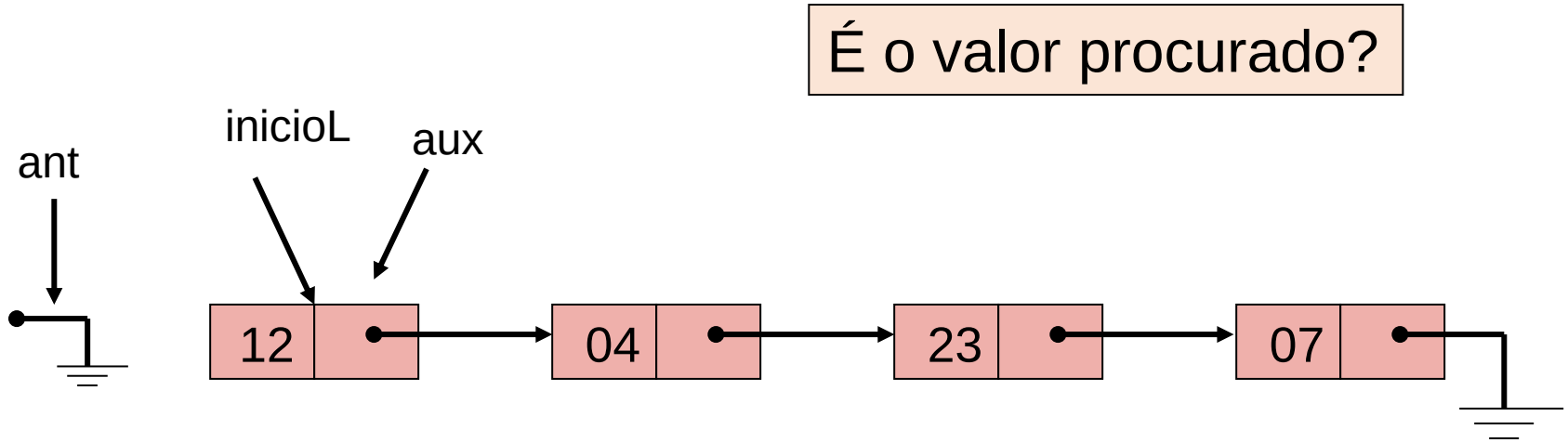
- Posição aleatória
- 1ª posição



Algoritmos e Estruturas de Dados I

REMOÇÃO - casos

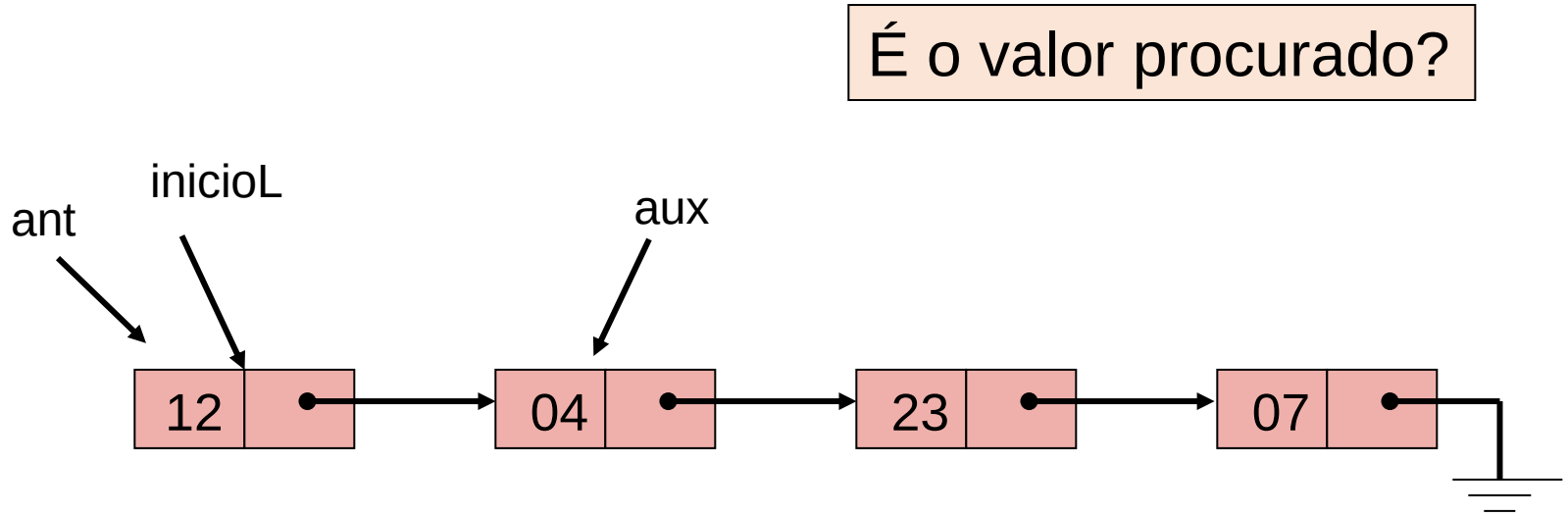
- Exemplo:
  - Valor procurado: 23



Algoritmos e Estruturas de Dados I

REMOÇÃO - casos

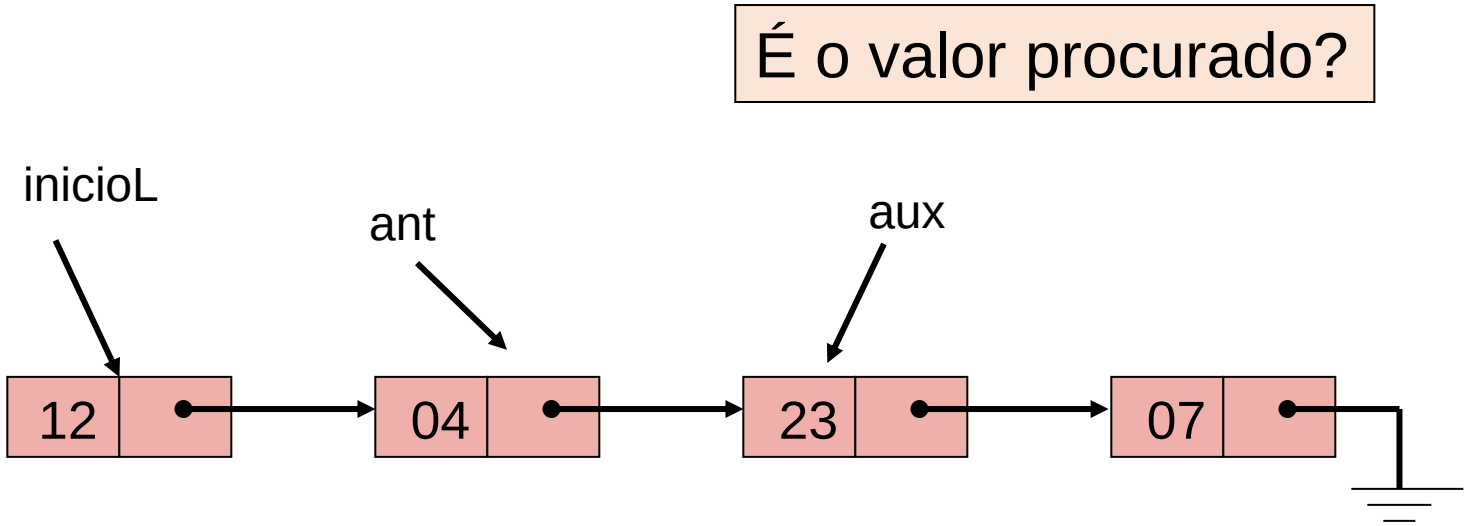
- Exemplo:
  - Valor procurado: 23



Algoritmos e Estruturas de Dados I

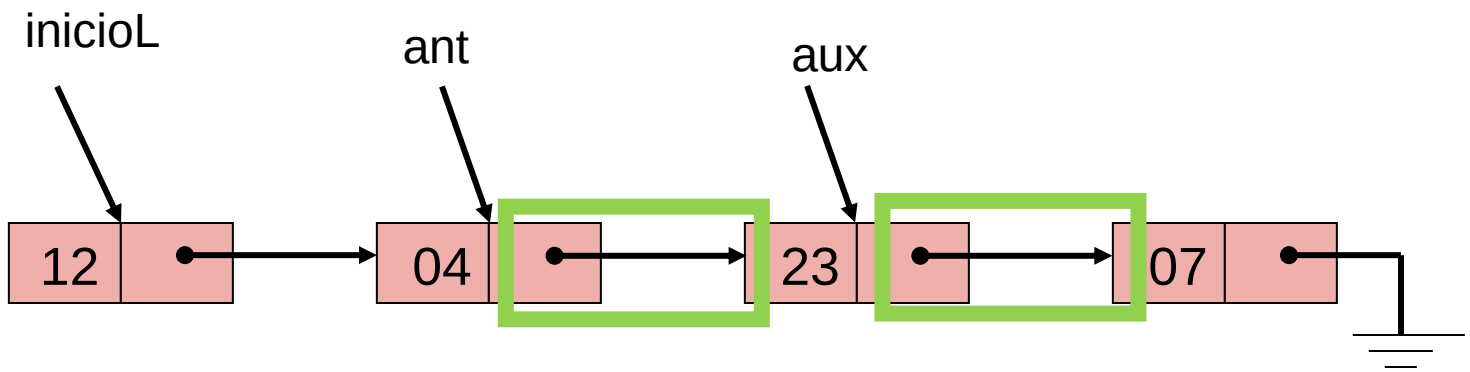
REMOÇÃO - casos

- Exemplo:
  - Valor procurado: 23



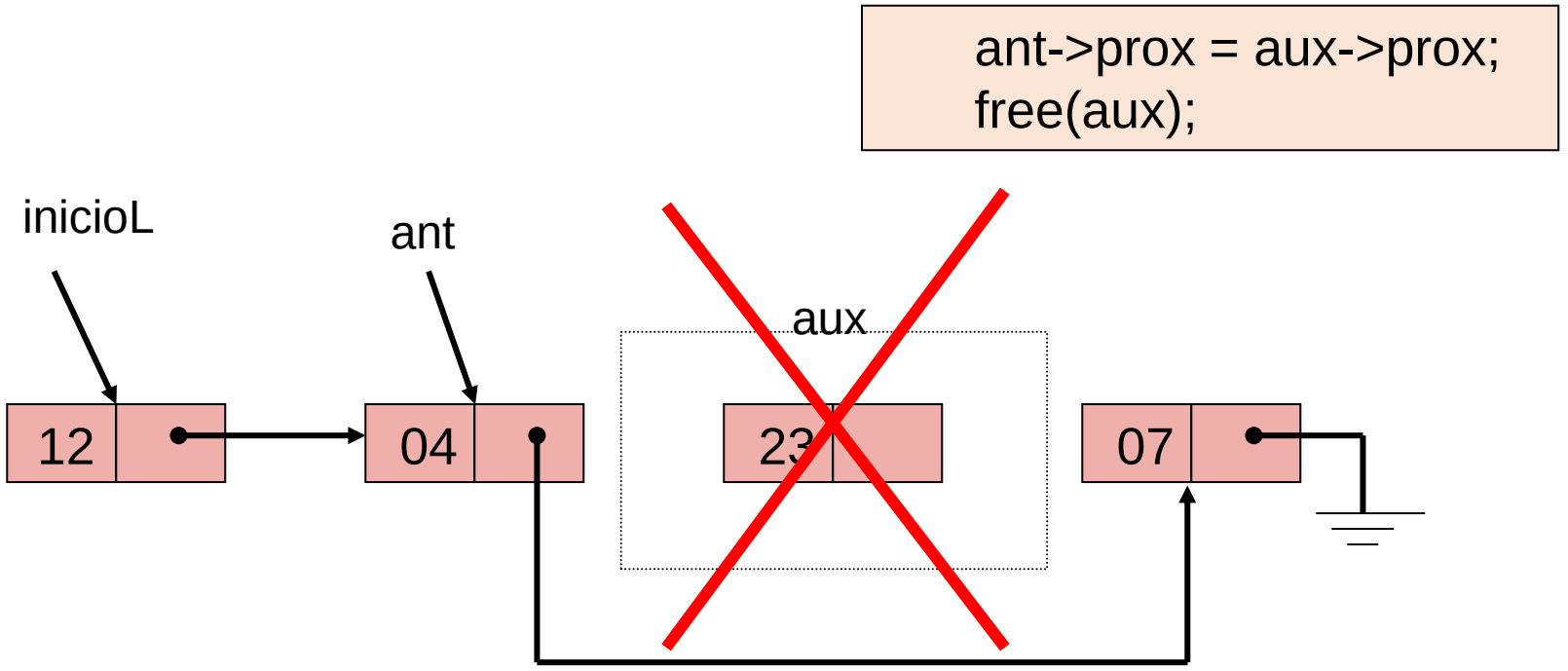
Algoritmos e Estruturas de Dados I

REMOÇÃO DE UM NÓ DO “MEIO” DA LISTA



Algoritmos e Estruturas de Dados I

REMOÇÃO DE UM NÓ DO “MEIO” DA LISTA



Algoritmos e Estruturas de Dados I

REMOÇÃO DE UM NÓ DO MEIO DA LISTA

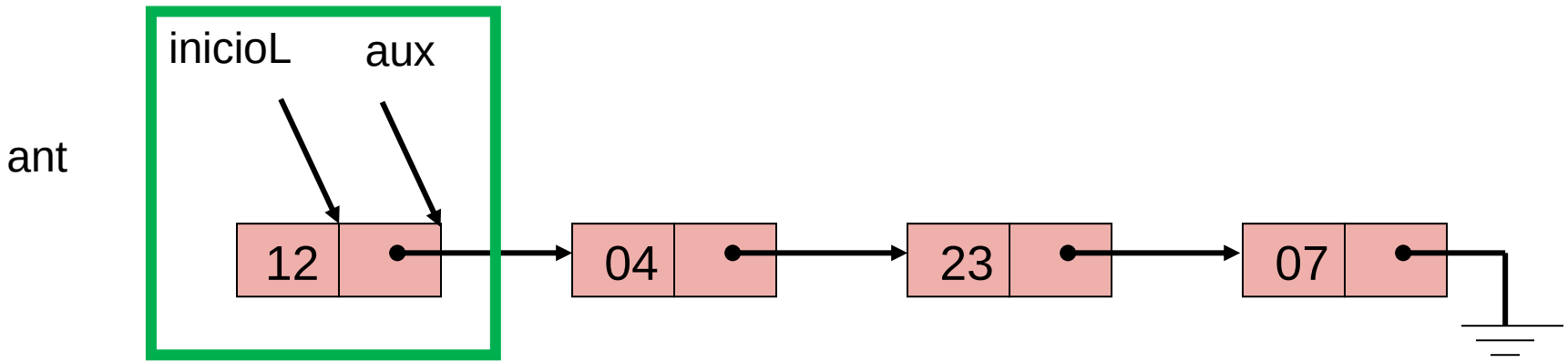
- Guardar o endereço do elemento que será removido em um ponteiro auxiliar aux;
- Fazer o nó ant apontar para o que o aux aponta;
- Liberar aux.

```
ant->prox = aux->prox;
free(aux);
```

Algoritmos e Estruturas de Dados I

REMOÇÃO DE UM NÓ NO INÍCIO DA LISTA

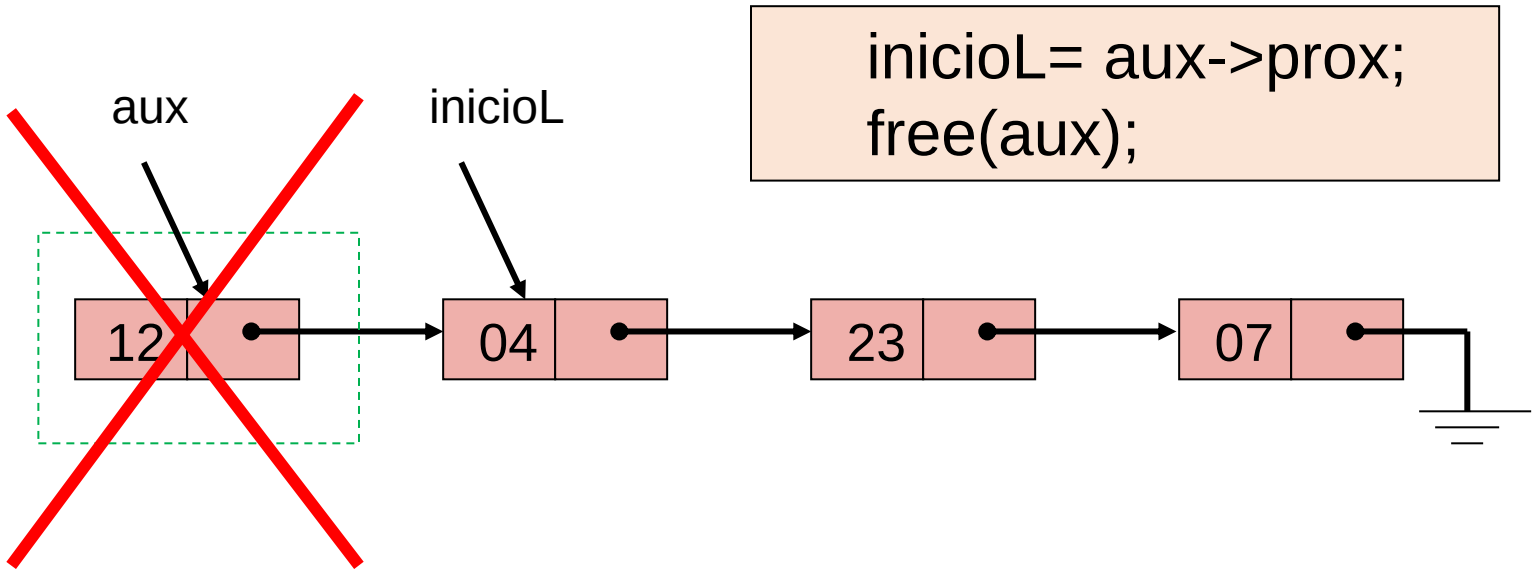
- Primeira posição:



Algoritmos e Estruturas de Dados I

REMOÇÃO DE UM NÓ NO INÍCIO DA LISTA

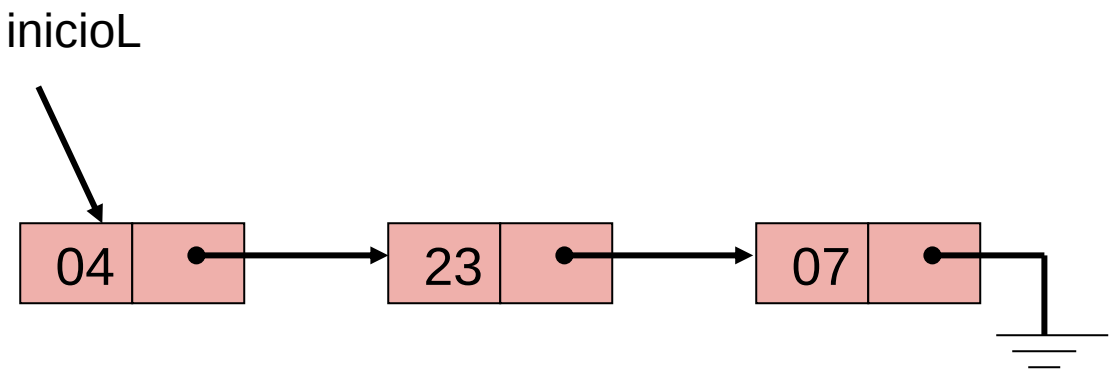
- Primeira posição:



Algoritmos e Estruturas de Dados I

REMOÇÃO DE UM NÓ NO INÍCIO DA LISTA

- Primeira posição:



Algoritmos e Estruturas de Dados I

REMOÇÃO

```
void remover(int valor){
   no* ant = NULL;
   no* aux = inicioL;
   if (!listaVazia() ) {
   //procura elemento na lista, guardando o anterior
   while (aux!=NULL)&&(aux->info!=valor) {
          ant = aux;
          aux = aux -> prox;
          }
```

Algoritmos e Estruturas de Dados I

REMOÇÃO

```
            if (aux == NULL)
                        printf("Elemento não encontrado!");
            else{  /* retira elemento */
                        if (ant == NULL)
                                    inicioL= aux-> prox;
                        else
                                    ant -> prox = aux-> prox;
                        free(aux);
            }
      }
      else
            printf("Lista vazia!");
}
```

DÚVIDAS???

Algoritmos e Estruturas de Dados I

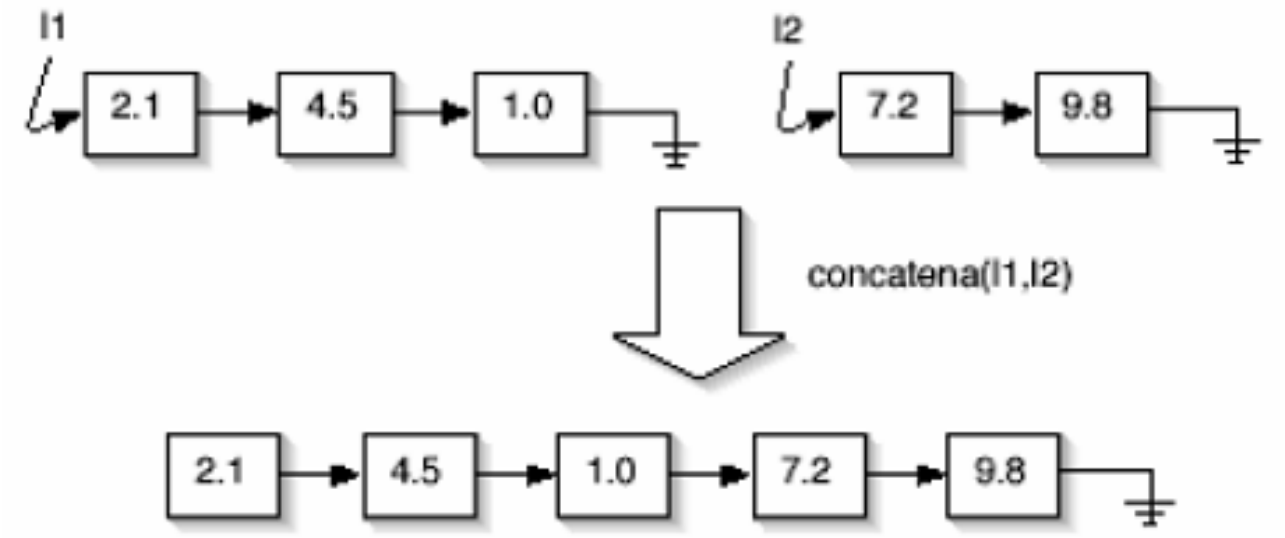
EXERCÍCIO

Faça uma função que permita buscar um elemento em uma lista encadeada, para que possa ser utilizada nas funções de inserção ou remoção de um elemento, quando necessário. A função deve retornar o endereço do elemento encontrado.

Algoritmos e Estruturas de Dados I

EXERCÍCIO

Faça uma função/procedimento que, a partir de duas listas encadeadas L1 e L2, crie uma lista L3 que será a concatenação das listas L1 e L2.



FIM