ESTRUTURA DE DADOS

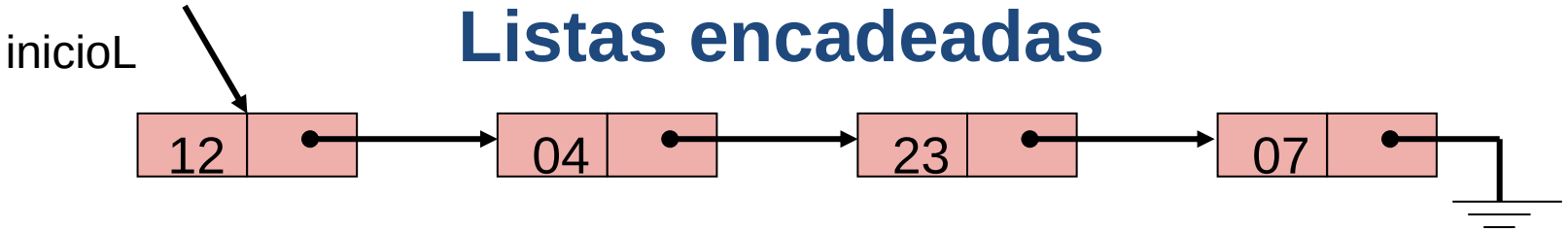
Prof.ª Priscilla Abreu

priscilla.braz@rj.senac.br

Estrutura de dados

INTRODUÇÃO

Nem sempre temos a certeza de quantas posições precisaremos para armazenar em uma lista!



Estrutura de dados

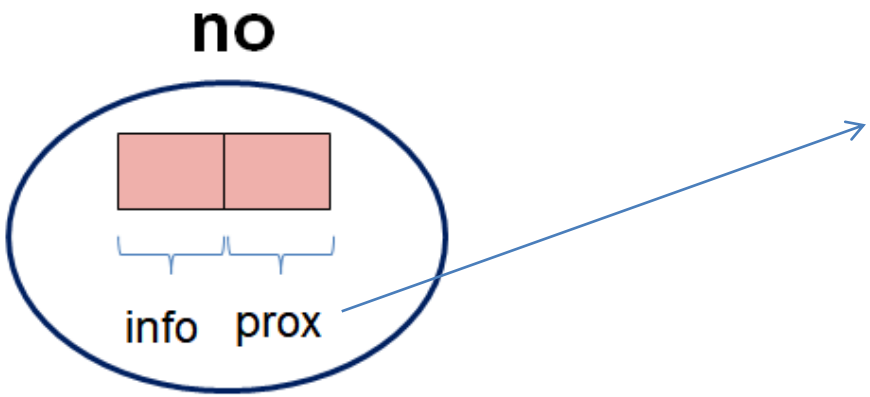
Listas encadeadas

- Possibilitam manter um conjunto de elementos relacionados, sem que todos tenham que ser declarados de uma só vez.
- No momento de cadastrar um elemento é que o **espaço necessário** para o armazenamento será solicitado.
- No entanto, não teremos garantia de que eles ficarão armazenados de modo **sequencial na memória.**

Estrutura de dados

Listas encadeadas

Cada elemento da lista precisa guardar, além do valor a ser armazenado na lista, uma indicação da localização do seu sucessor na lista.

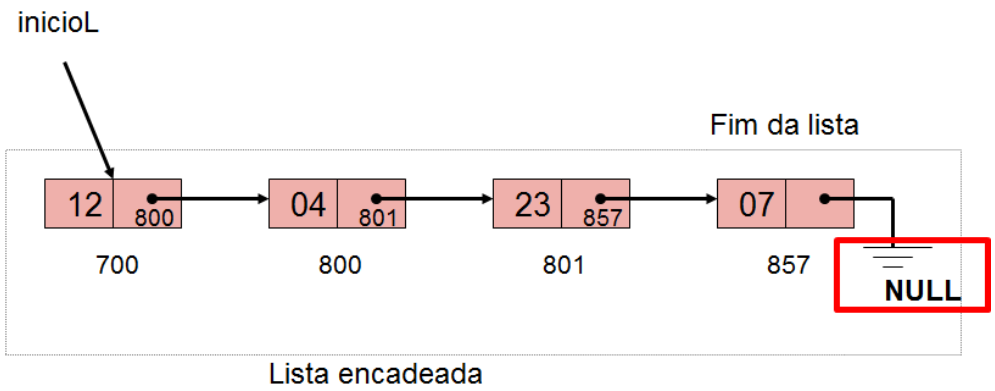


Ponteiro que armazena o endereço do próximo elemento da lista.

Estrutura de dados

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.

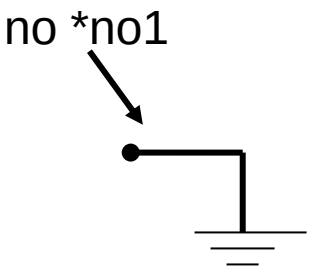



```
typedef struct no{
      int info;
      struct no *prox;
}no;

no *inicioL;
```

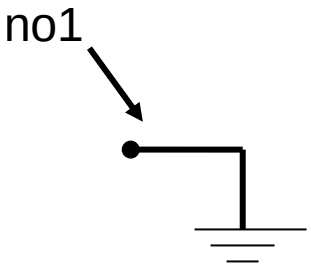
Estrutura de dados

LISTAS ENCADEADAS



Estrutura de dados

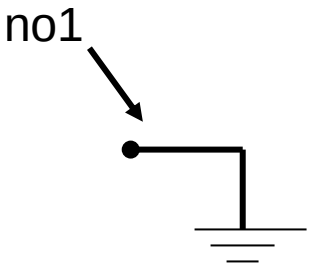
LISTAS ENCADEADAS



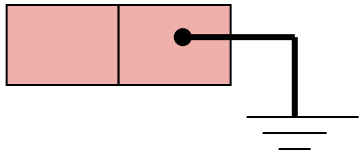

```
no1 = (no*)malloc(sizeof(no));
```

Estrutura de dados

LISTAS ENCADEADAS

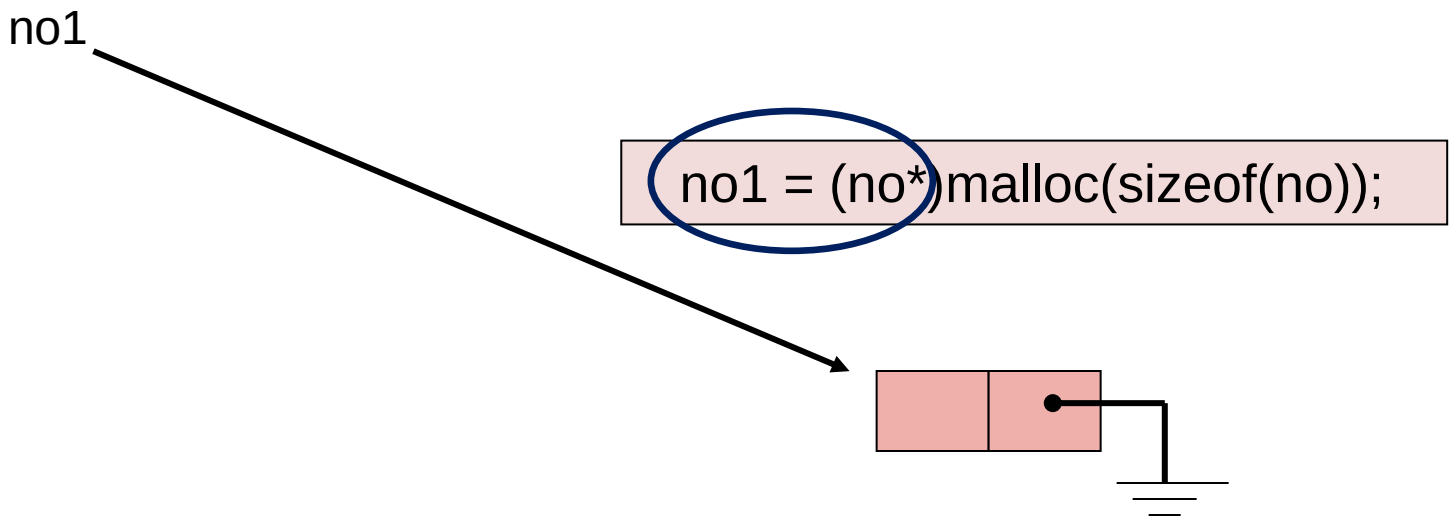



```
no1 = (no*)malloc(sizeof(no));
```



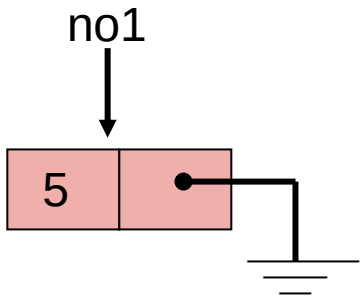
Estrutura de dados

LISTAS ENCADEADAS

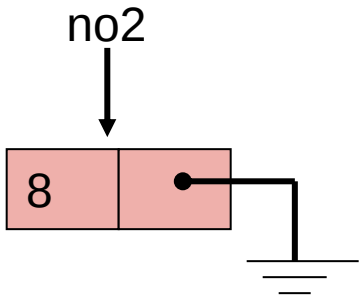


Estrutura de dados

LISTAS ENCADEADAS



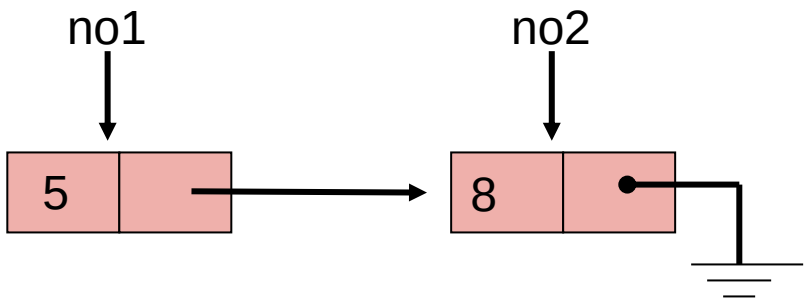

```
no1 = (no*)malloc(sizeof(no));
```




```
no2 = (no*)malloc(sizeof(no));
```

Estrutura de dados

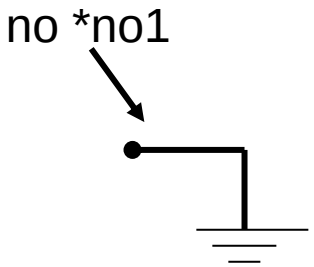
LISTAS ENCADEADAS



Estrutura de dados

endereço	valor		var
1001			
1002			**no1**
1003			
1004			
1005			
1006			
1007			

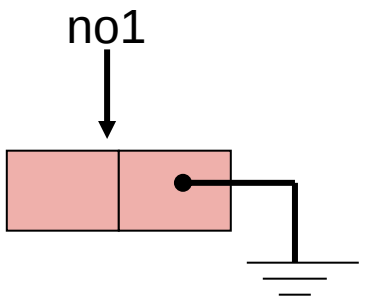
```
no *no1;
```



Estrutura de dados

endereço	valor		var
1001			
1002			**no1**
1003			
1004			
1005			
1006			
1007			

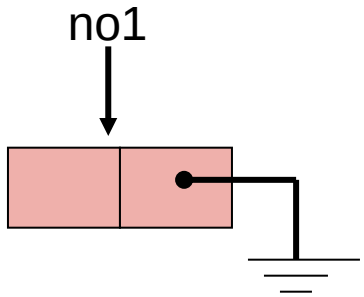
```
no1 = (no*)malloc(sizeof(no));
```



Estrutura de dados

endereço	valor		var
1001			
1002			**no1**
1003	**info**	**prox**	
1004			
1005			
1006			
1007			

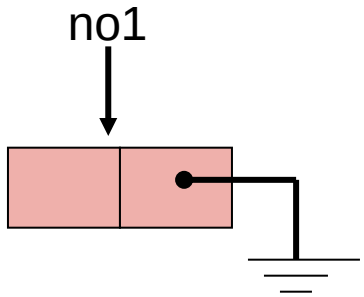
```
no1 = (no*)malloc(sizeof(no));
```



Estrutura de dados

endereço	valor		var
1001			
1002	1003		**no1**
1003	**info**	**prox**	
1004			
1005			
1006			
1007			

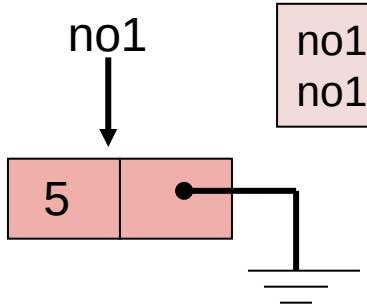
```
no1 = (no*)malloc(sizeof(no));
```



Estrutura de dados

endereço	valor		var
1001			
1002	1003		**no1**
1003	**info**	**prox**	
	5	NULL	
1004			
1005			
1006			
1007			

```
no1 = (no*)malloc(sizeof(no));
```

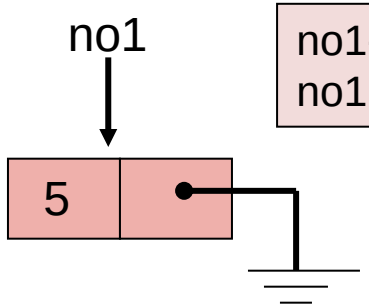



```
no1-> info = 5;
no1 -> prox = NULL;
```

Estrutura de dados

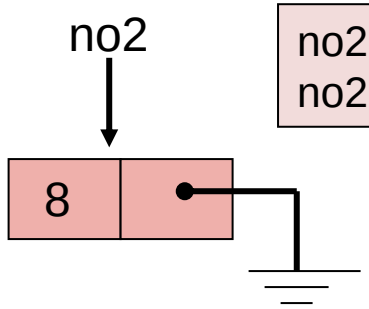
endereço	valor		var
1001			
1002	1003		**no1**
1003	**info**	**prox**	
	5	NULL	
1004			
1005			
1006			
1007			

```
no1 = (no*)malloc(sizeof(no));
```




```
no1-> info = 5;
no1 -> prox = NULL;
```

```
no2 = (no*)malloc(sizeof(no));
```

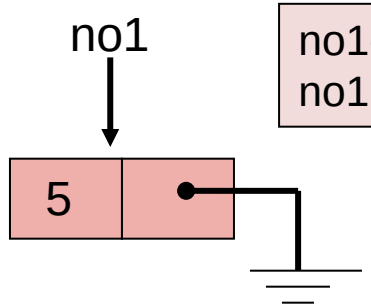



```
no2-> info = 8;
no2 -> prox = NULL;
```

Estrutura de dados

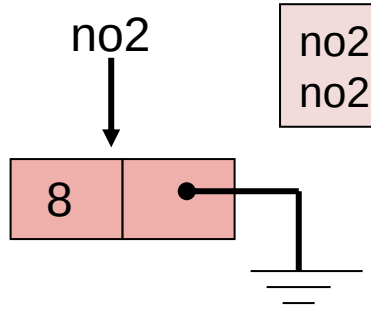
endereço	valor		var
1001			
1002	1003		**no1**
1003	**info**	**prox**	
	5	NULL	
1004			
1005	1007		**no2**
1006			
1007	**info**	**prox**	
	8	NULL	

```
no1 = (no*)malloc(sizeof(no));
```




```
no1-> info = 5;
no1 -> prox = NULL;
```

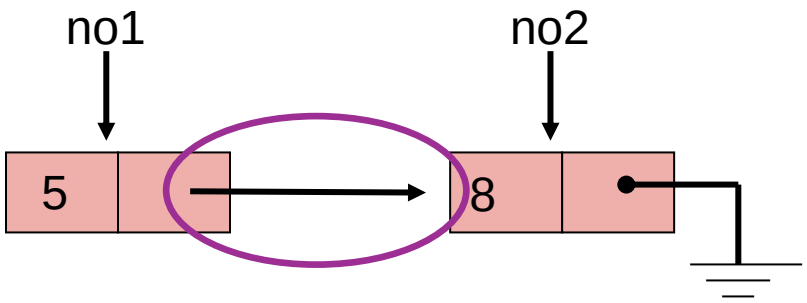
```
no2 = (no*)malloc(sizeof(no));
```




```
no2-> info = 8;
no2 -> prox = NULL;
```

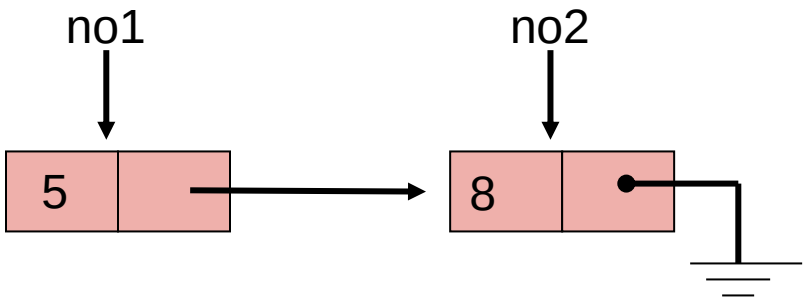
Estrutura de dados

endereço	valor		var
1001			
1002	1003		**no1**
1003	**info**	**prox**	
	5	NULL	
1004			
1005	1007		**no2**
1006			
1007	**info**	**prox**	
	8	NULL	



Estrutura de dados

endereço	valor		var
1001			
1002	1003		**no1**
1003	**info**	**prox**	
	5	**1007**	
1004			
1005	1007		**no2**
1006			
1007	**info**	**prox**	
	8	NULL	



Estrutura de dados

LISTAS ENCADEADAS

```
#include <stdio.h>
#include <stdlib.h>

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

no *inicioL;
```

OPERAÇÕES:

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

Estrutura de dados

LISTAS ENCADEADAS

- Inicializar a lista;
- Verificar se a lista está vazia;

Estrutura de dados

LISTAS ENCADEADAS

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

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

LISTAS ENCADEADAS

Inserção

- Inserção no início
- Inserção em posições aleatórias
- Inserção no final

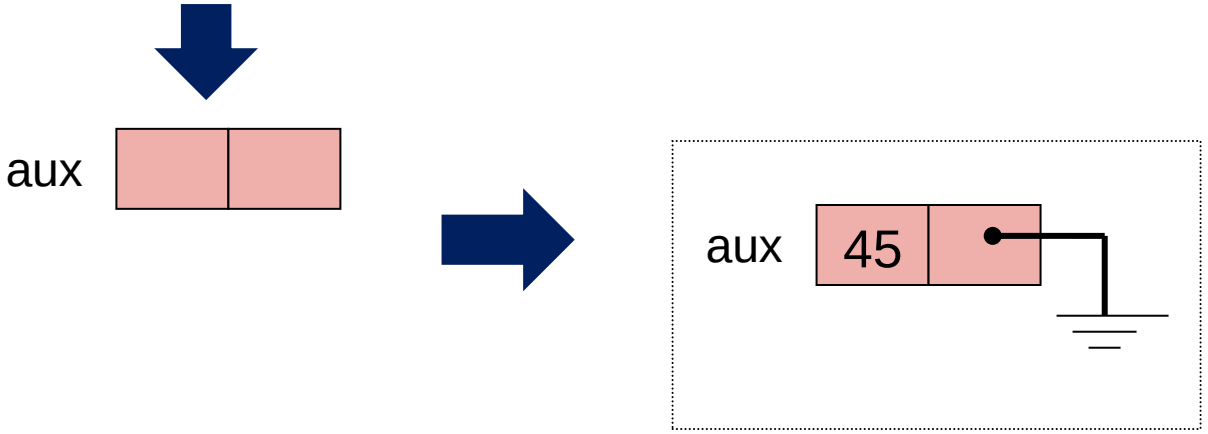
Antes de inserir um elemento na lista é necessário alocar um espaço para seu armazenamento.

Estrutura de dados

LISTAS ENCADEADAS

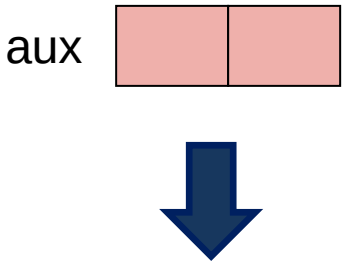
CRIAR UM ELEMENTO DO TIPO NÓ:

```
aux = (no*)malloc(sizeof(no));
```



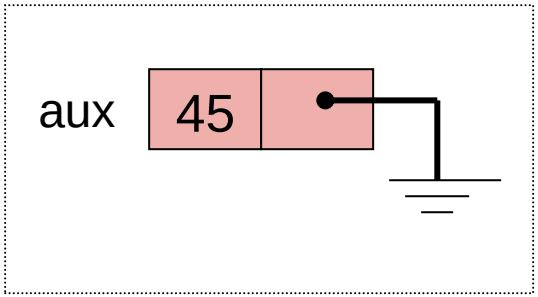
Estrutura de dados

LISTAS ENCADEADAS



CRIAR UM ELEMENTO DO TIPO NÓ:

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



LISTAS ENCADEADAS

Inserção em listas encadeadas

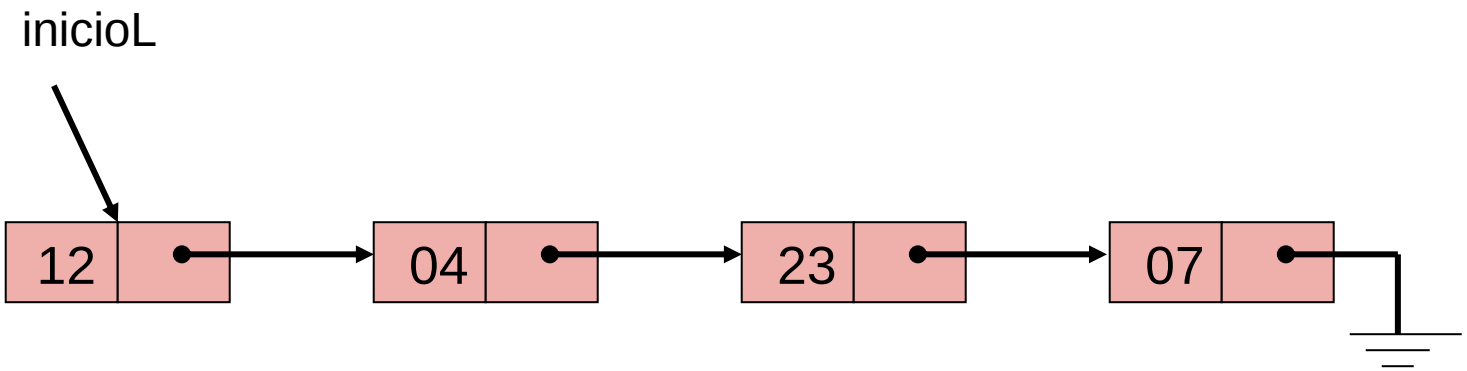
- Inserção no início

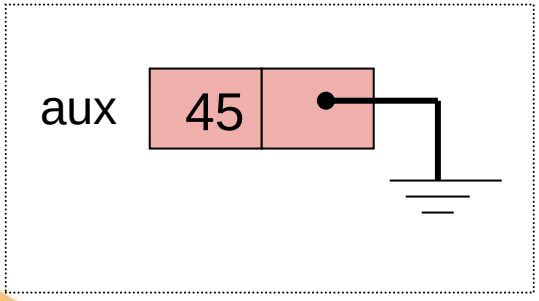
Inicialmente é preciso alocar um espaço do tipo nó, usando a função cria_no.

Estrutura de dados

LISTAS ENCADEADAS

INSERÇÃO DE UM NÓ NO INÍCIO






```
cria_no(45);
```

LISTAS ENCADEADAS

Inserção em listas encadeadas

- Inserção no início

Em seguida, o elemento criado (aux) precisa ser o primeiro da lista. Logo, ele deve indicar como seu próximo elemento, o que era até aquele momento o primeiro da lista (inicioL).

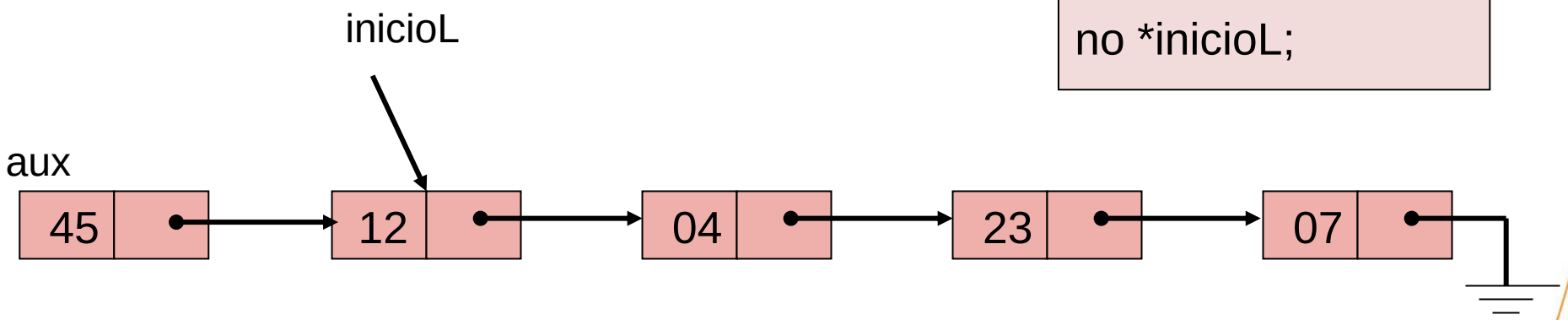
Estrutura de dados

LISTAS ENCADEADAS

INSERÇÃO DE UM NÓ NO INÍCIO

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

no *inicioL;
```




```
aux -> prox = inicioL
```

LISTAS ENCADEADAS

Inserção em listas encadeadas

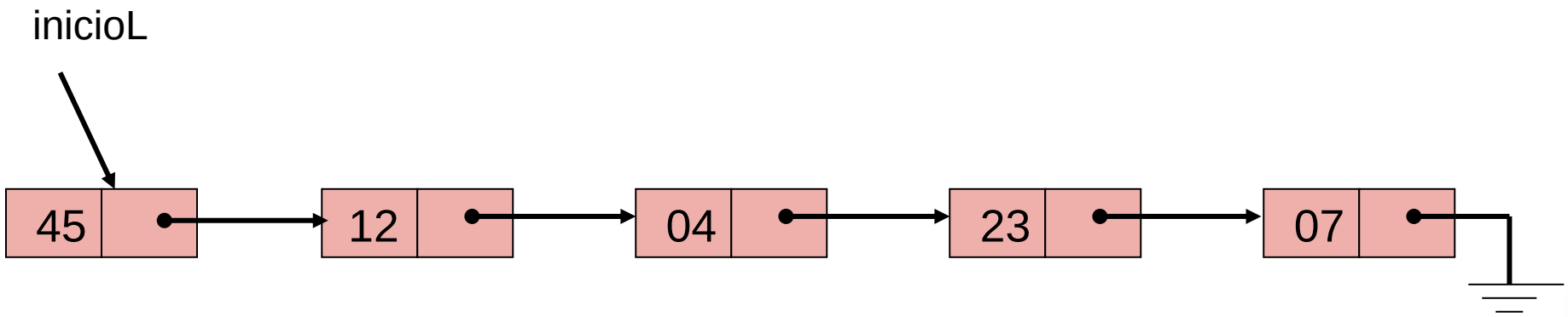
- Inserção no início

Agora, o ponteiro que indica o início da lista deve ser atualizado e passar a referenciar o elemento que acabou de ser inserido no início da lista.

Estrutura de dados

LISTAS ENCADEADAS

INSERÇÃO DE UM NÓ NO INÍCIO




```
inicioL = aux;
```

Estrutura de dados

LISTAS ENCADEADAS

INSERÇÃO DE UM NÓ NO INÍCIO

```
void inserir_ini (int valor) {
                    no* aux;
                    aux = cria_no(valor);
                    aux -> prox = inicioL;
                    inicioL = aux;
}
```

COMO PERCORRER A LISTA?

LISTAS ENCADEADAS

Percurso

Para percorrer uma lista precisamos de uma variável auxiliar. Tal variável será responsável por acessar cada elemento. Isso não pode ser feito com a variável **inicioL**, pois esta indica o início da lista e se modificarmos seu valor perderemos a referência de onde inicia a lista.

LISTAS ENCADEADAS

Percurso

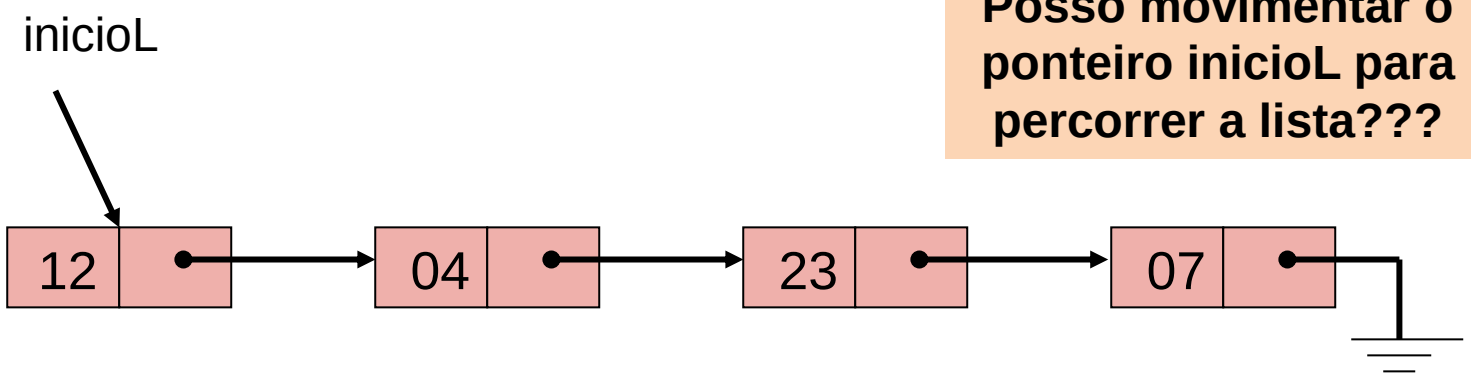
A variável auxiliar (aux) será inicializada com a referência de início da lista e irá percorrendo a lista até chegar ao seu final.

O final é encontrado ao atingir o valor NULL.

LISTAS ENCADEADAS

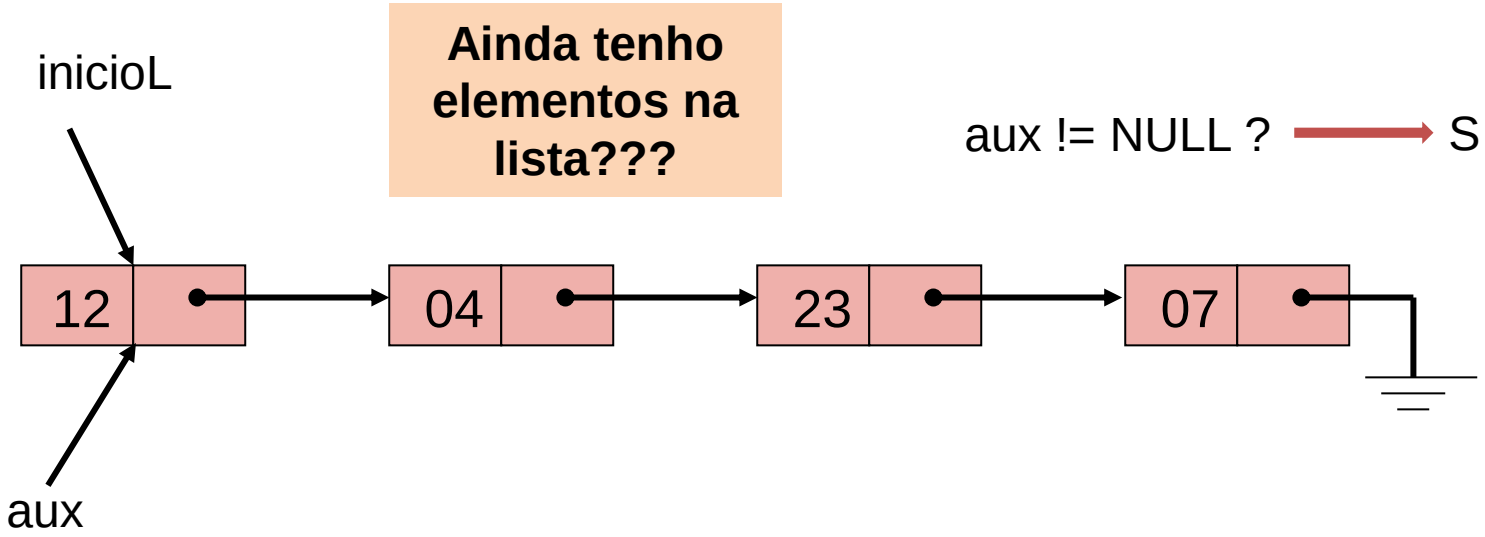
CONDIÇÃO DE PARADA PARA PERCORRER

Posso movimentar o ponteiro inicioL para percorrer a lista???



LISTAS ENCADEADAS

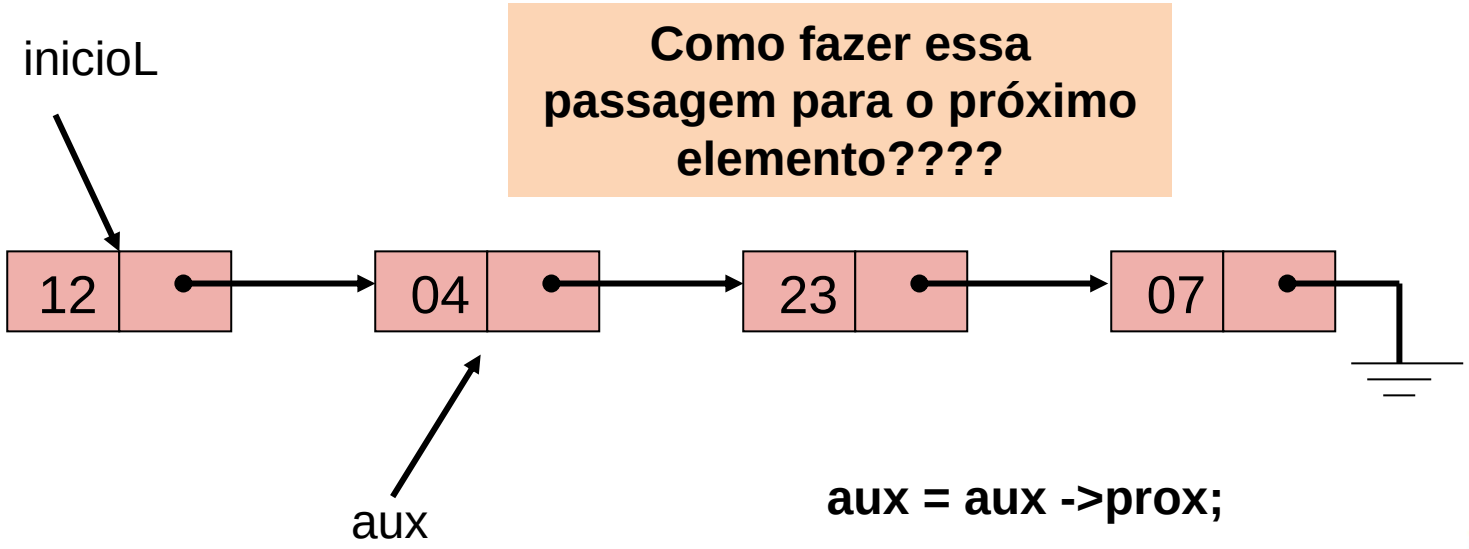
CONDIÇÃO DE PARADA PARA PERCORRER



Estrutura de dados

LISTAS ENCADEADAS

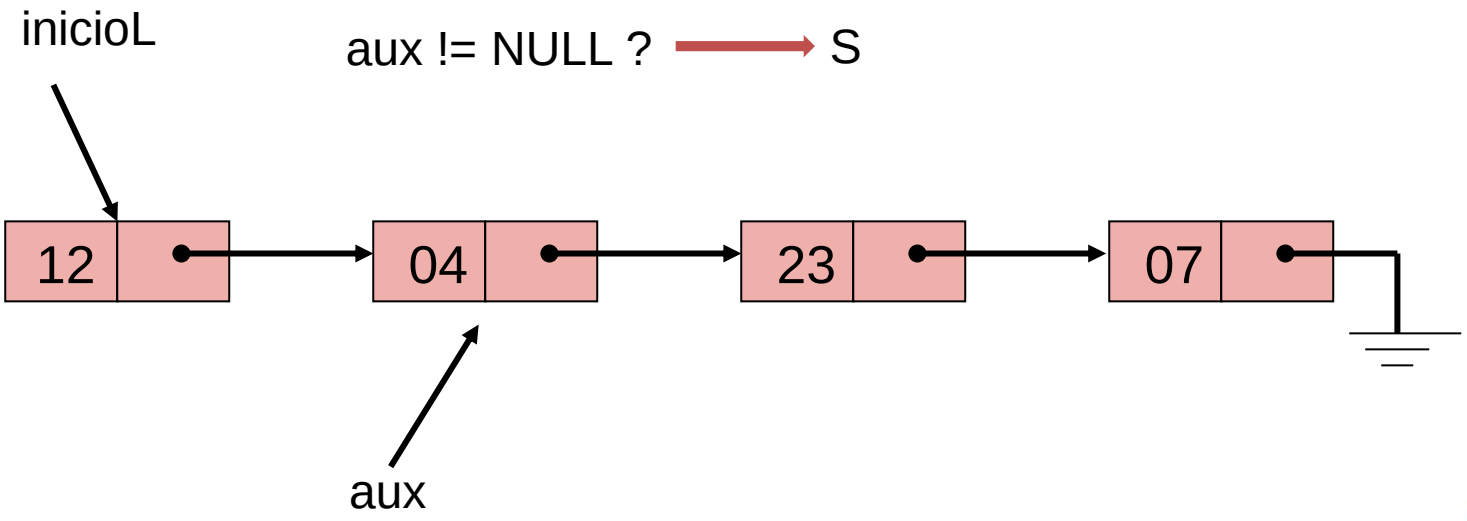
CONDIÇÃO DE PARADA PARA PERCORRER



Estrutura de dados

LISTAS ENCADEADAS

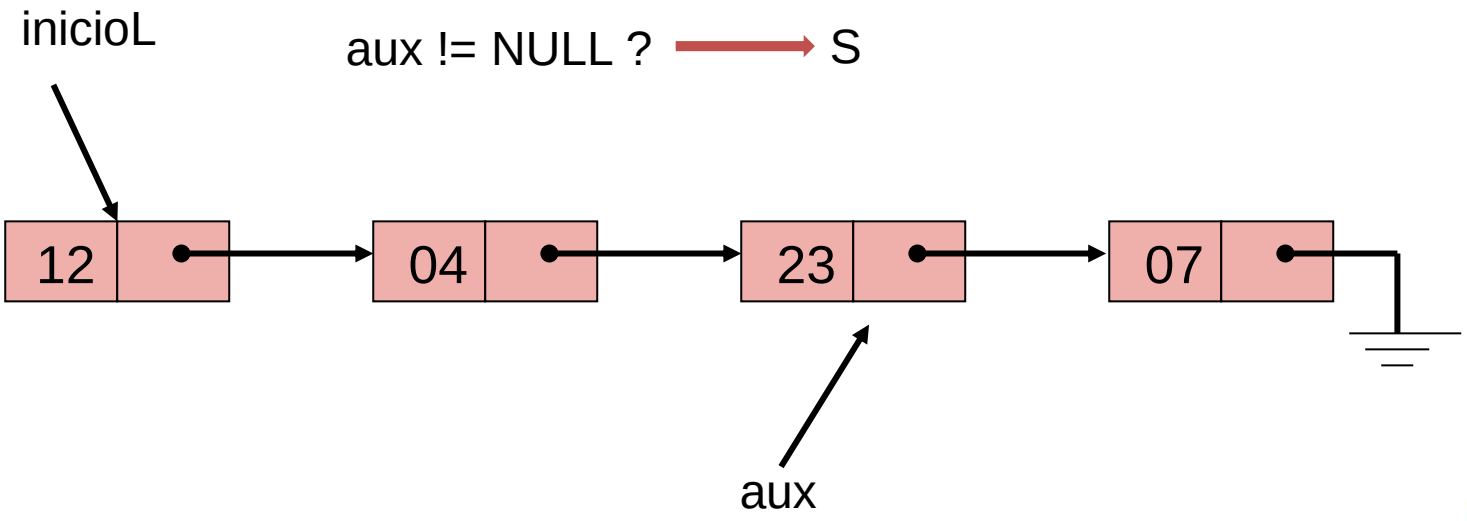
CONDIÇÃO DE PARADA PARA PERCORRER



Estrutura de dados

LISTAS ENCADEADAS

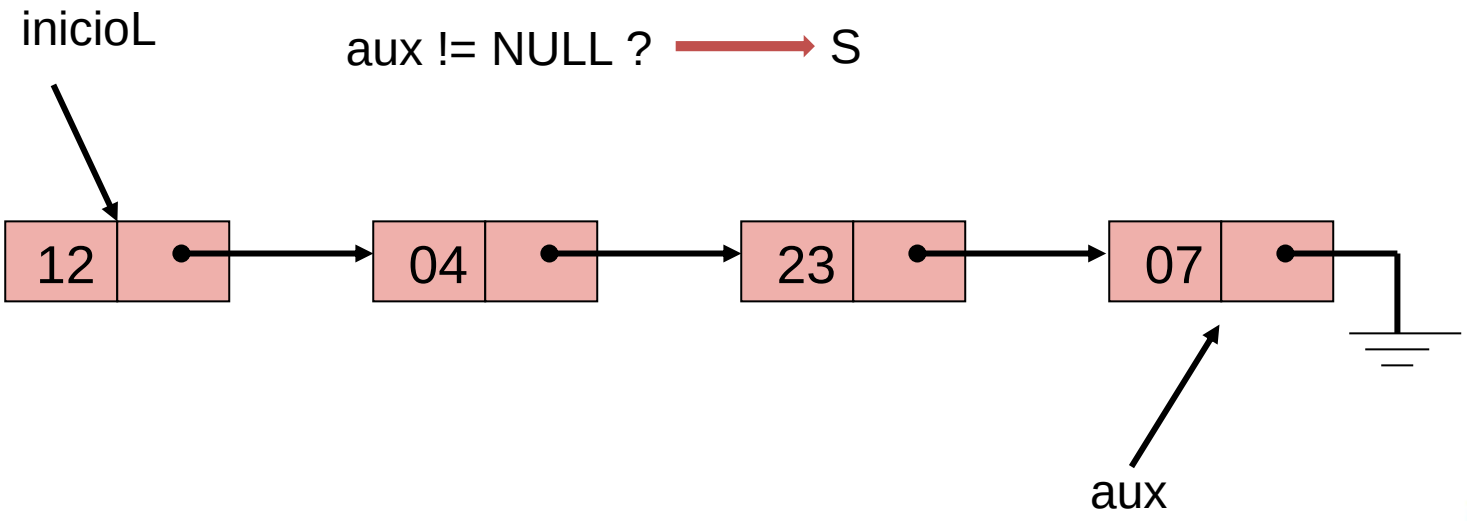
CONDIÇÃO DE PARADA PARA PERCORRER



Estrutura de dados

LISTAS ENCADEADAS

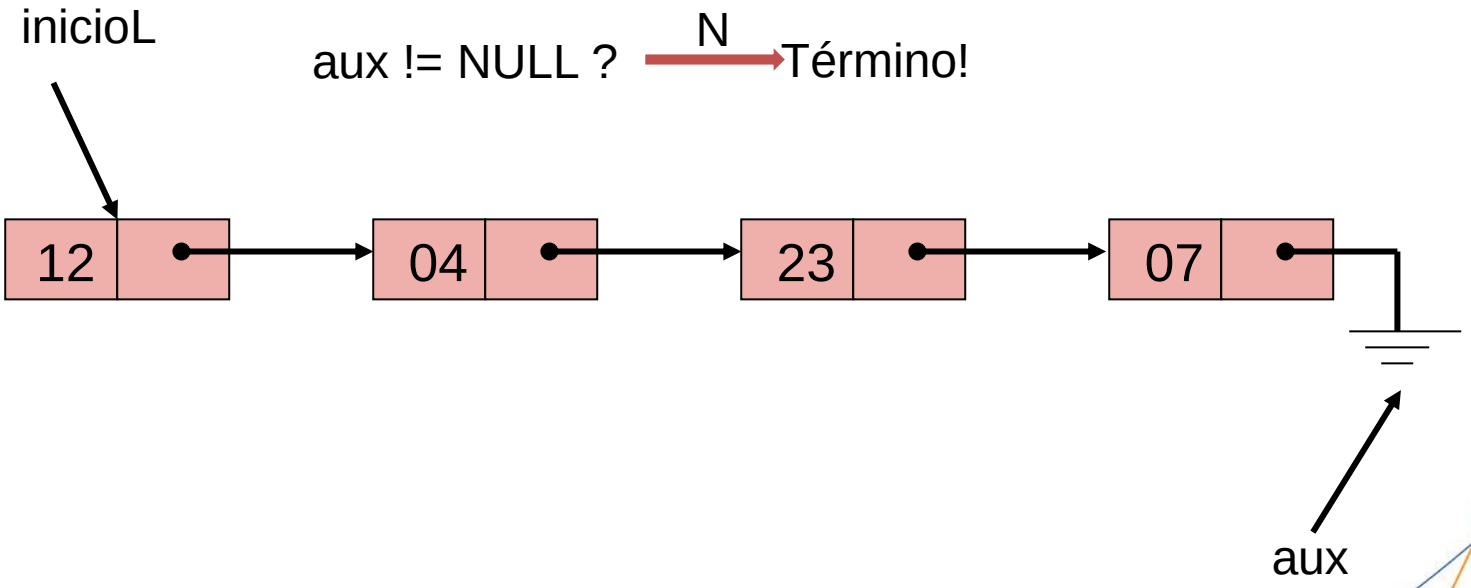
CONDIÇÃO DE PARADA PARA PERCORRER



Estrutura de dados

LISTAS ENCADEADAS

CONDIÇÃO DE PARADA PARA PERCORRER



Estrutura de dados

LISTAS ENCADEADAS

PERCORRER A LISTA

```
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");
     }
}
```

E para inserir um elemento ao final da lista, como fazer?

LISTAS ENCADEADAS

Inserção no final

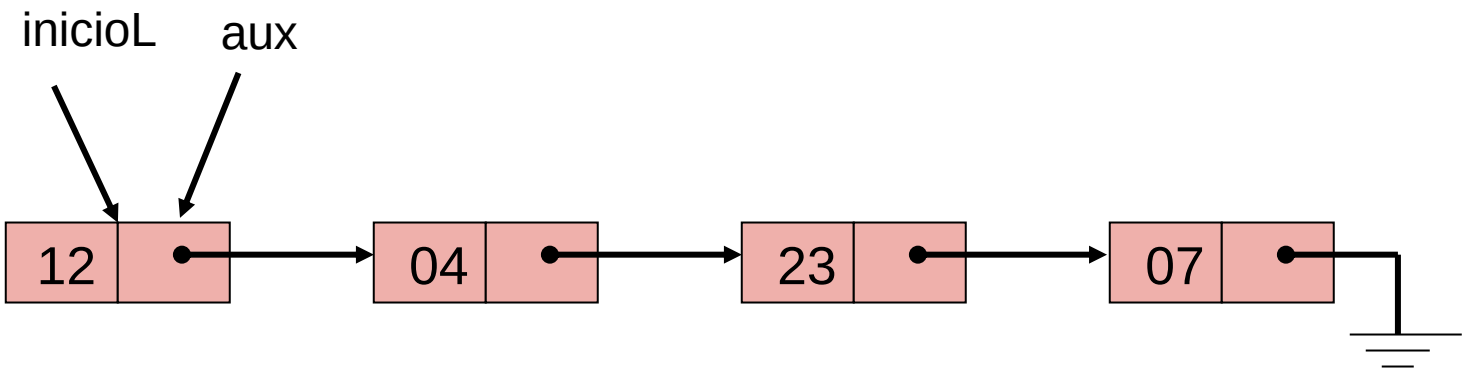
A variável auxiliar (aux) será inicializada com a referência de início da lista e irá percorrendo a lista até chegar no último elemento, sem chegar no valor NULL.

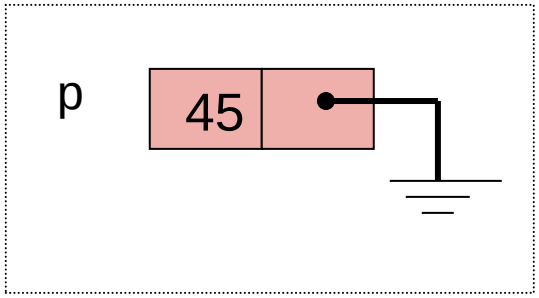
Estando com a referência desse último elemento, já é possível ligar seu campo **prox** ao nó a ser inserido na lista.

Estrutura de dados

LISTAS ENCADEADAS

INSERÇÃO DE UM NÓ NO FIM





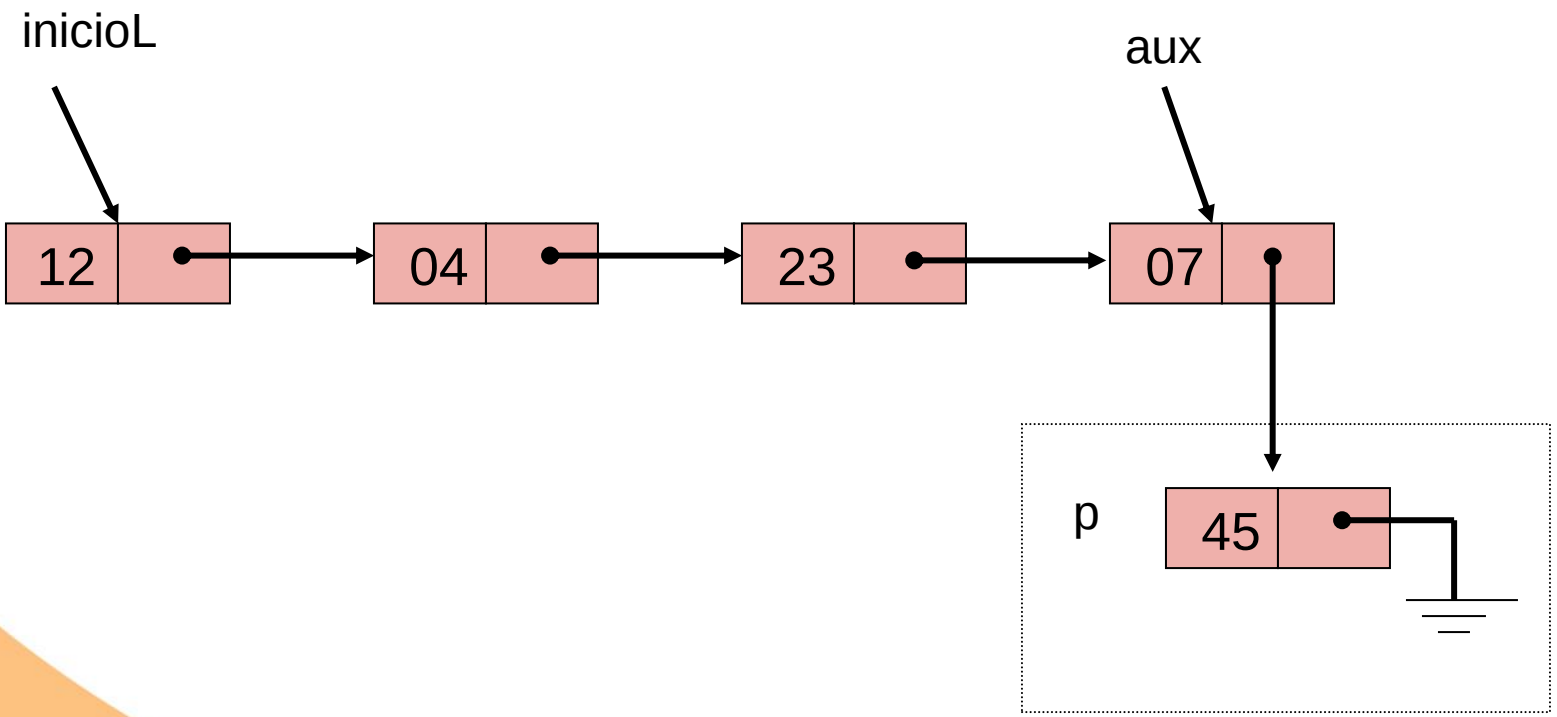

```
cria_no(45);
```

Estrutura de dados

LISTAS ENCADEADAS

Ponteiro utilizado para encontrar o último elemento da lista

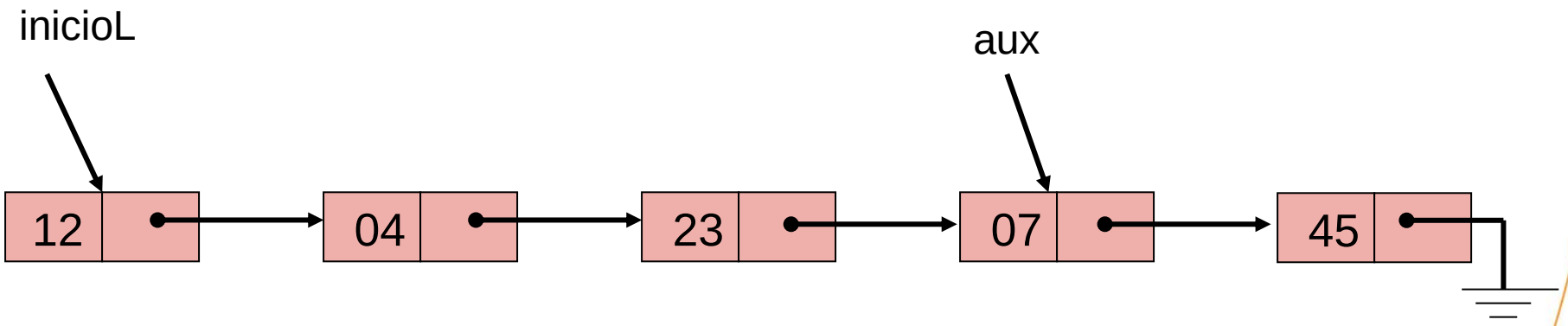
INSERÇÃO DE UM NÓ NO FIM



Estrutura de dados

LISTAS ENCADEADAS

INSERÇÃO DE UM NÓ NO FIM




```
aux->prox = p;
```

Estrutura de dados

LISTAS ENCADEADAS

INSERÇÃO DE UM NÓ NO FIM

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

Estrutura de dados

LISTAS ENCADEADAS

EXERCÍCIO 1

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

LISTAS ENCADEADAS

REMOÇÃO

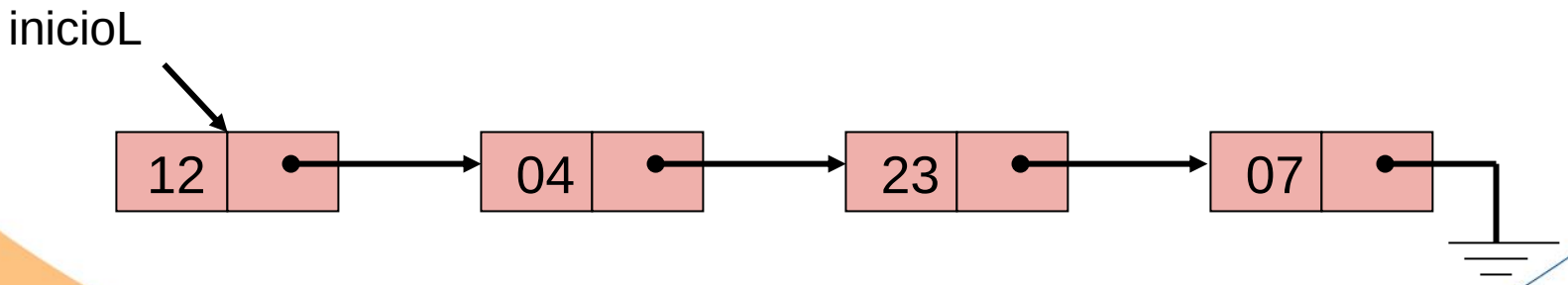
Antes de remover um elemento é necessário verificar se há elementos na lista e procurar pelo elemento que deseja remover, além de obter informações de qual elemento está antes do elemento a ser removido, para que as referências dos ponteiros possam ser atualizadas.

Estrutura de dados

LISTAS ENCADEADAS

REMOÇÃO: passos

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

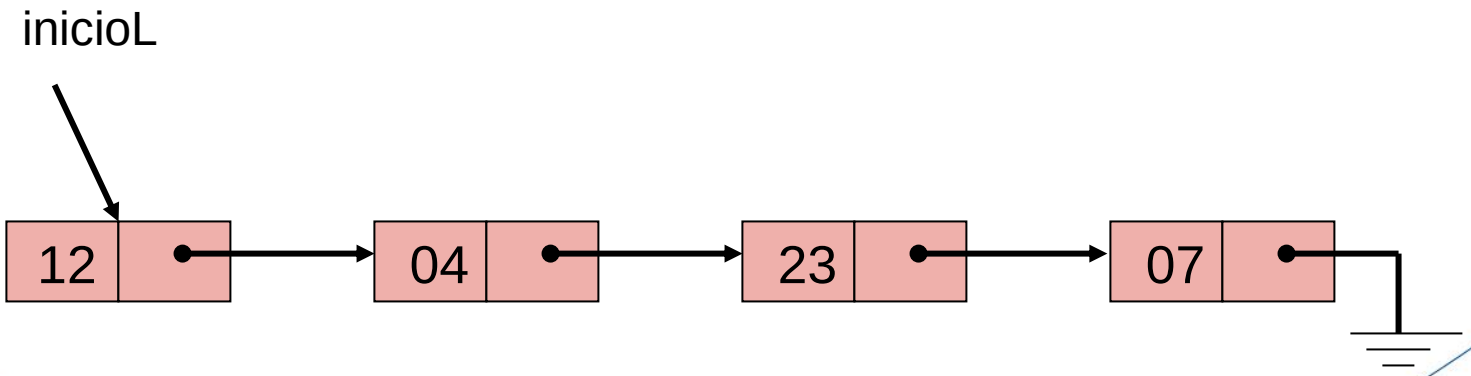


Estrutura de dados

LISTAS ENCADEADAS

REMOÇÃO - casos

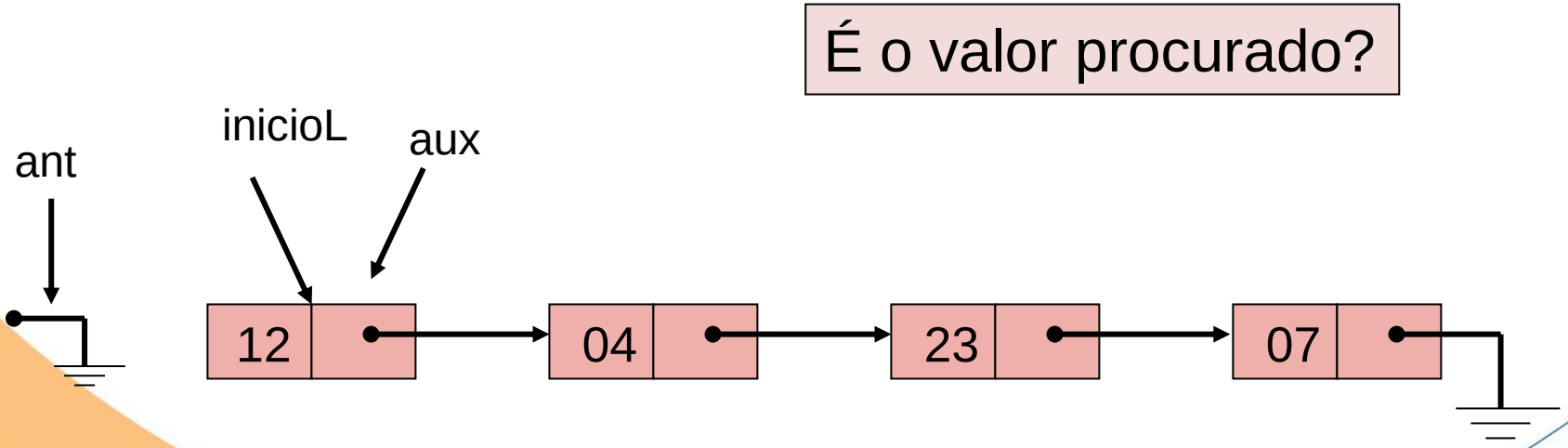
- Posição aleatória
- 1ª posição



LISTAS ENCADEADAS

REMOÇÃO - casos

- Exemplo:
  - Valor procurado: 23



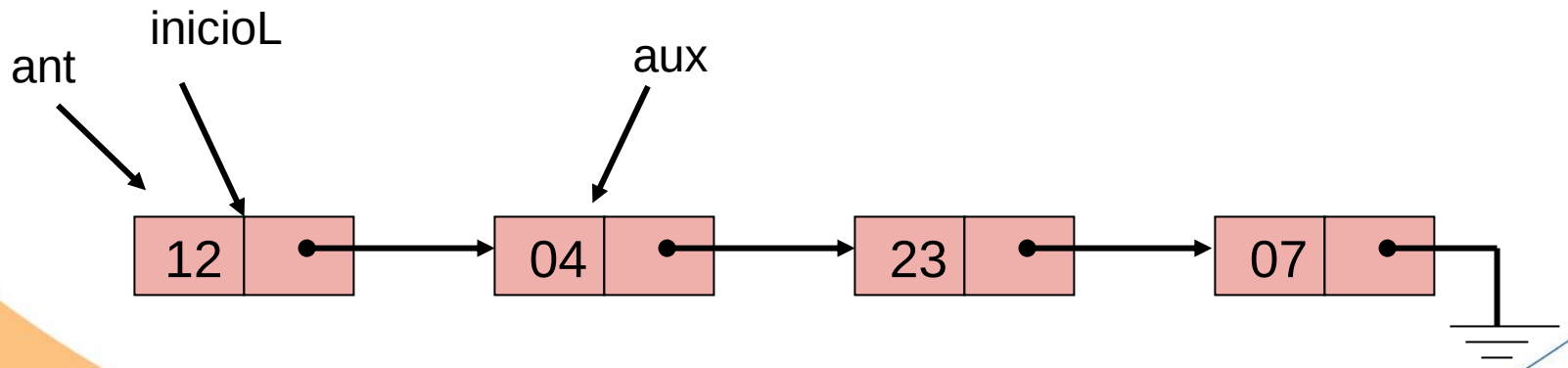
Estrutura de dados

LISTAS ENCADEADAS

REMOÇÃO - casos

- Exemplo:
  - Valor procurado: 23

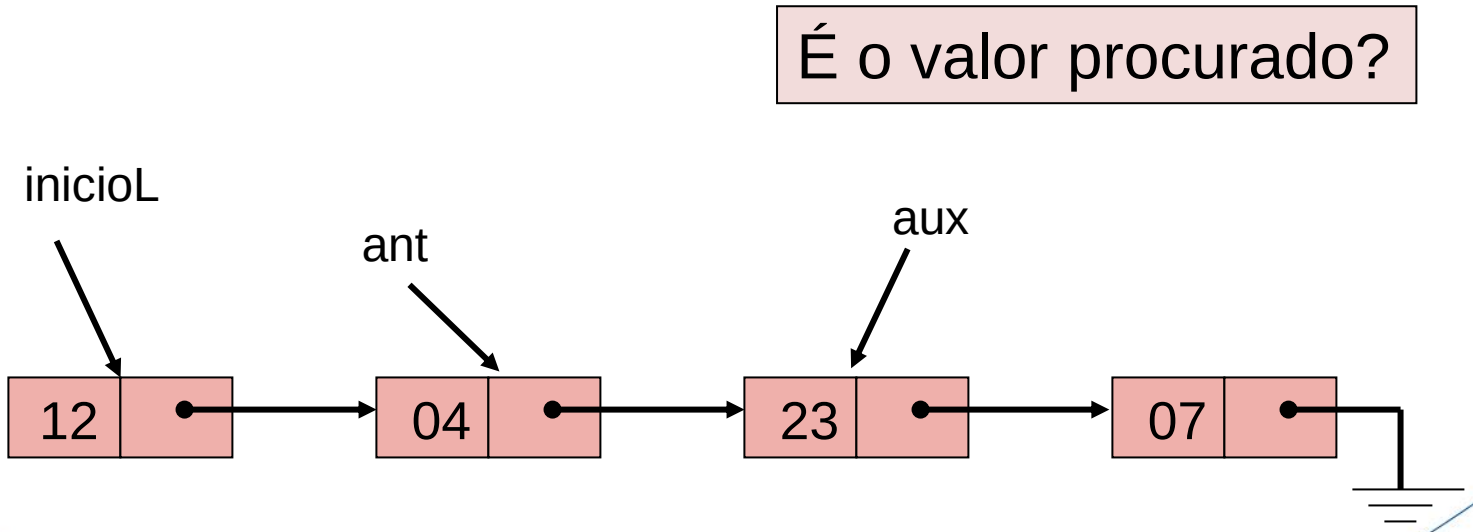
É o valor procurado?



LISTAS ENCADEADAS

REMOÇÃO - casos

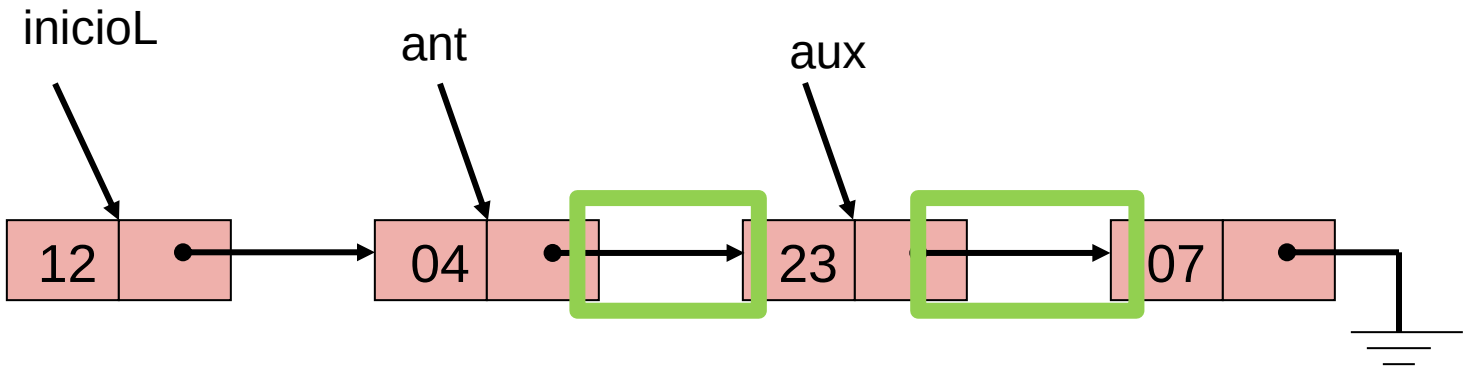
- Exemplo:
  - Valor procurado: 23



Estrutura de dados

LISTAS ENCADEADAS

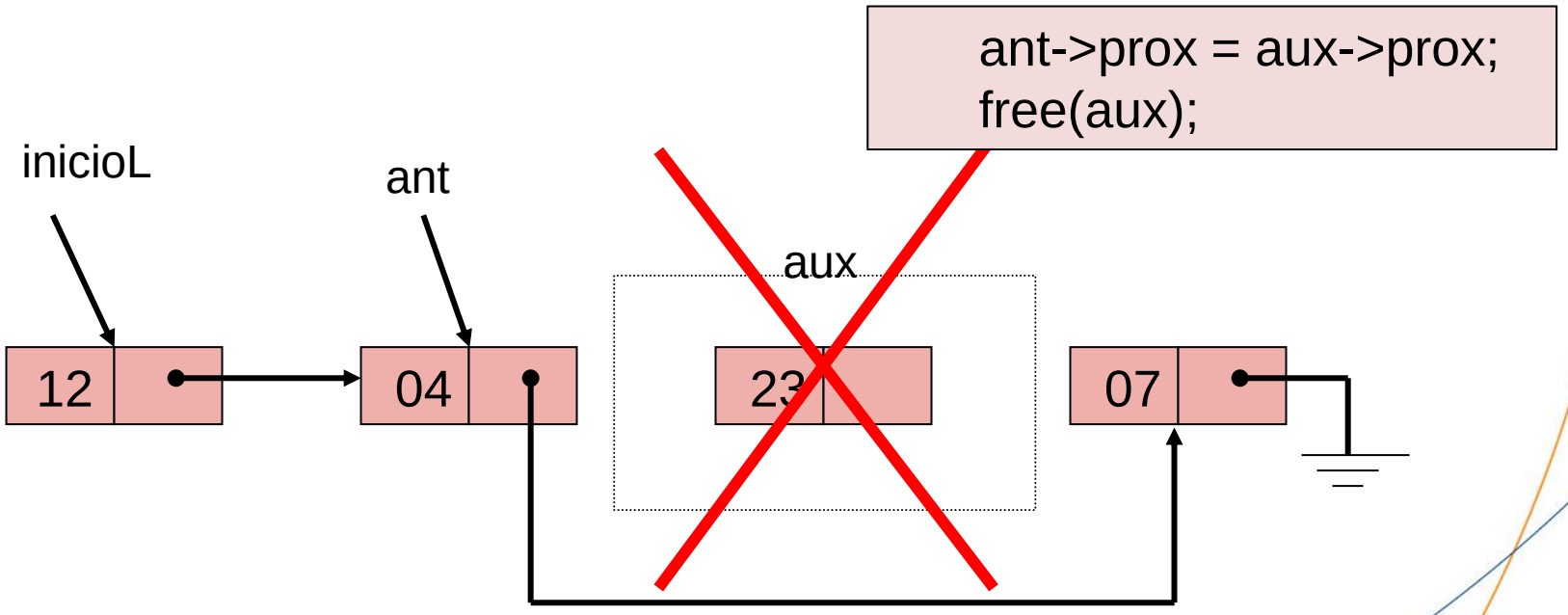
REMOÇÃO DE UM NÓ DO MEIO DA LISTA



Estrutura de dados

LISTAS ENCADEADAS

REMOÇÃO DE UM NÓ DO MEIO DA LISTA



LISTAS ENCADEADAS

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.

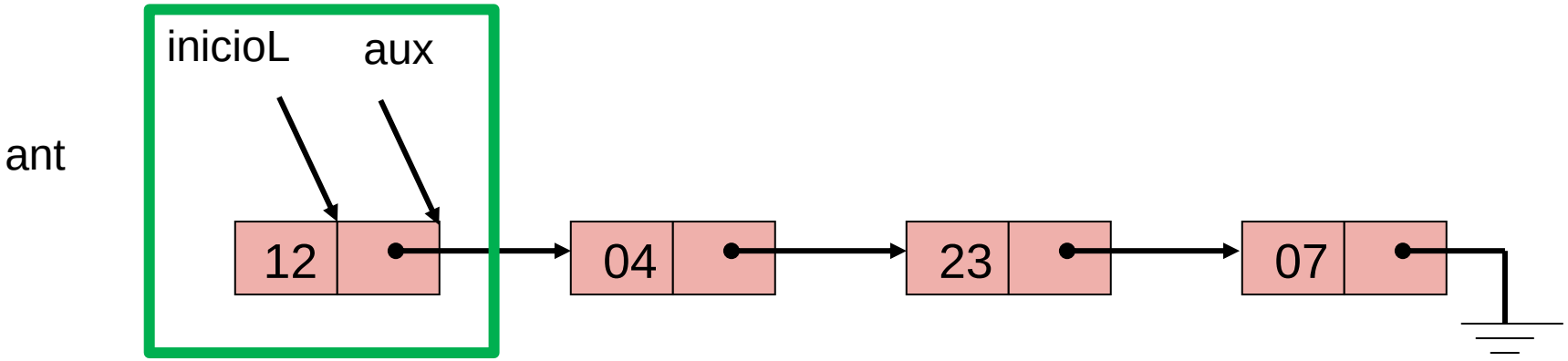
```
ant->prox = aux->prox;
free(aux);
```

Estrutura de dados

LISTAS ENCADEADAS

REMOÇÃO DE UM NÓ NO INÍCIO DA LISTA

- Primeira posição:



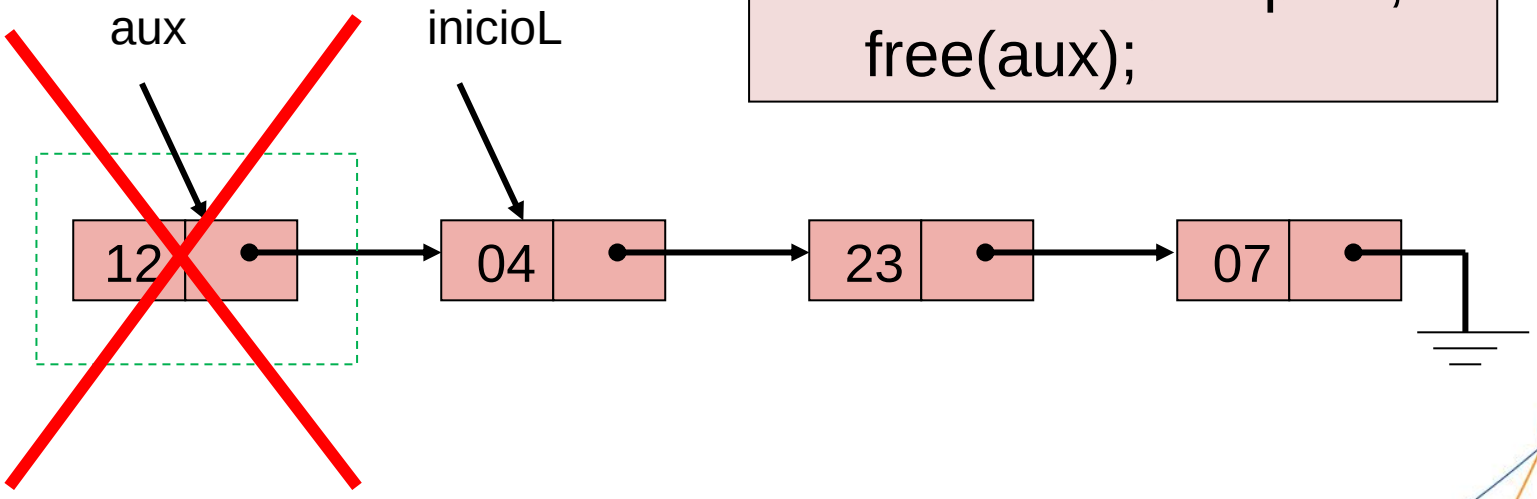
Estrutura de dados

LISTAS ENCADEADAS

REMOÇÃO DE UM NÓ NO INÍCIO DA LISTA

- Primeira posição:

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

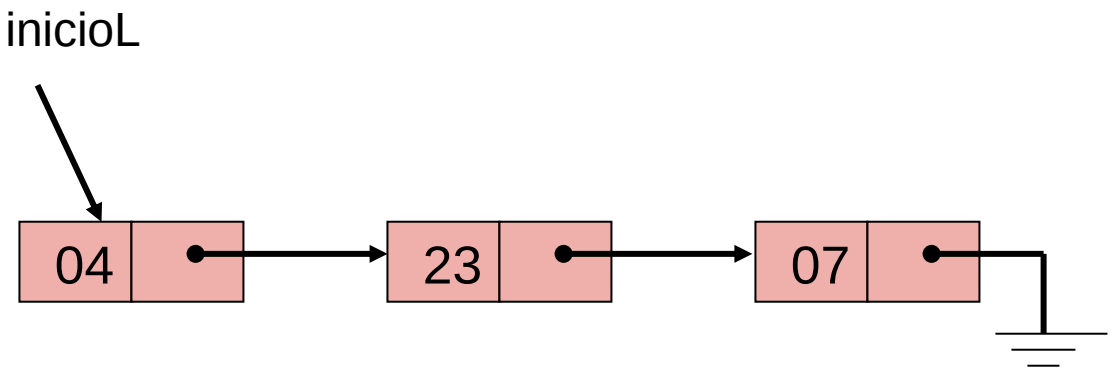


Estrutura de dados

LISTAS ENCADEADAS

REMOÇÃO DE UM NÓ NO INÍCIO DA LISTA

- Primeira posição:



Estrutura de dados

LISTAS ENCADEADAS

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;
        }
```

Estrutura de dados

LISTAS ENCADEADAS

REMOÇÃO DE UM NÓ DO MEIO DA LISTA

```
        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?

LISTAS ENCADEADAS

EXERCÍCIO 2

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.

LISTAS ENCADEADAS

EXERCÍCIO 3

Complemente o menu do programa do 1º exercício para que permita ao usuário também buscar por um valor na lista e remover um elemento.

Lista Simplesmente Encadeada

1- Inserir
2- Buscar um valor
3- Remover um valor
4- Exibir a lista
5- Sair

LISTAS ENCADEADAS

EXERCÍCIO 4

Implemente um programa que defina uma lista simplesmente encadeada e que insira quantos elementos o usuário desejar. Ao final, o programa deve exibir o maior e o menor elementos dessa lista. Para isso, você deve implementar um procedimento que execute essa operação.

ESTRUTURA DE DADOS

CRONOGRAMA FINAL

- **07/06:**
  - Correção exercícios Listas Simplesmente Encadeadas
  - Listas Duplamente Encadeadas
- **11/06:** Horário para esclarecimento de dúvidas sobre o seminário (sábado)
- **14/06:**
  - Atividade Listas Simplesmente Encadeadas
  - Planejamento Seminário Árvores e Grafos
- **21/06:** Seminário – parte 1
- **28/06:**
  - Seminário – parte 2
  - Atividade individual (árvores e grafos)
- **02/07:** Atividade de recuperação (sábado)
- **05/07:** Feedback final e recuperação final

ESTRUTURA DE DADOS

ATIVIDADE EM GRUPO – SEMINÁRIO

O material para cada grupo será disponibilizado no dia 07/06.

Etapa 1 (14/06): Preparação para o seminário.

- A segunda parte da aula será reservada para que os grupos possam esclarecer suas dúvidas com a professora.
- É fundamental o estudo prévio do conteúdo para que possam aproveitar melhor o tempo de aula para a elaboração do material e esclarecimento de dúvidas.

ESTRUTURA DE DADOS

ATIVIDADE EM GRUPO – SEMINÁRIO

Etapa 2 (21/06 e 28/06): Apresentação do seminário.

- Tema principal: árvores e grafos;
- A turma será dividida em 4 grupos;
- 28/06: Atividade individual sobre árvores e grafos.

ESTRUTURA DE DADOS

ATIVIDADE EM GRUPO – SEMINÁRIO

Etapa 3 (02/07): Atividade recuperação

05/07:

- Feedback final
- Atividade de recuperação