ESTRUTURA DE DADOS

Prof.ª Priscilla Abreu

priscilla.braz@rj.senac.br

Estrutura de dados

Roteiro de Aula

- Objetivo da aula
- Listas encadeadas
- Exercícios

Estrutura de dados

ALOCAÇÃO DINÂMICA

Através desse modo de alocação dinâmica, conseguimos alocar espaços de memória em tempo de execução e modificar o espaço alocado, também em tempo de execução.

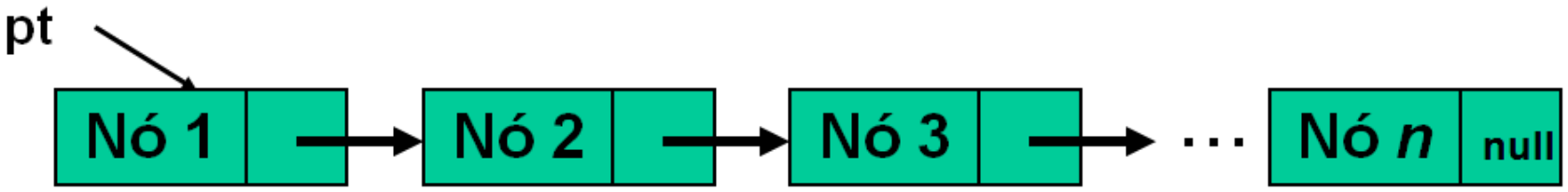
No entanto, todo o espaço alocado dinamicamente reservou um bloco de memória com endereços sequenciais, que foram manipulados como vetores.

Estrutura de dados

ALOCAÇÃO DINÂMICA

Com base no conceito de alocação dinâmica, veremos como alocar espaço de memória dinamicamente elemento a elemento, sem necessariamente os endereços alocados serem endereços sequenciais.

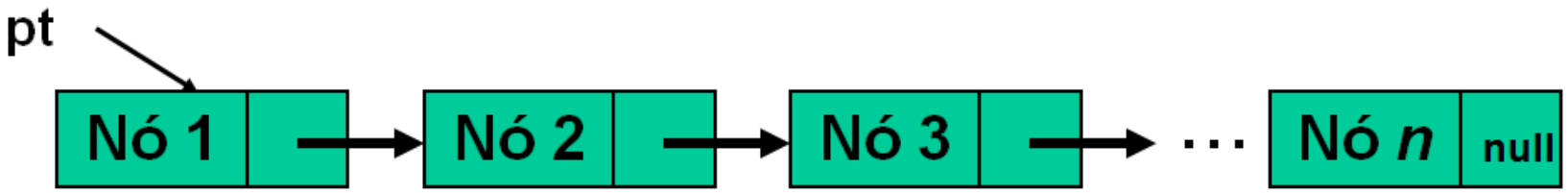
Utilizaremos LISTAS ENCADEADAS!



Estrutura de dados

INTRODUÇÃO

Com listas encadeadas temos estruturas de dados que podem crescer à medida que precisarmos armazenar novos elementos e diminuir à medida que precisarmos retirar elementos armazenados anteriormente.

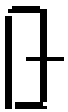


Estrutura de dados

LISTAS ENCADEADAS

- A proposta desse tipo de lista é que a estrutura esteja, inicialmente, vazia.
- A cada necessidade de inserção de um elemento na lista, um espaço de memória será solicitado para armazenar o novo dado e depois será adicionado na lista, através de alocação dinâmica.

Lista

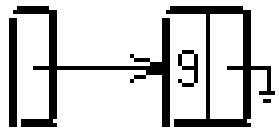


Estrutura de dados

LISTAS ENCADEADAS

- A proposta desse tipo de lista é que a estrutura esteja, inicialmente, vazia.
- A cada necessidade de inserção de um elemento na lista, um espaço de memória será solicitado para armazenar o novo dado e depois será adicionado na lista, através de alocação dinâmica.

Lista

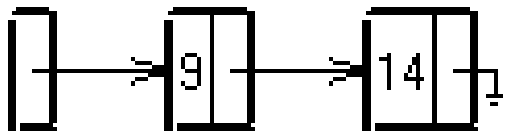


Estrutura de dados

LISTAS ENCADEADAS

- A proposta desse tipo de lista é que a estrutura esteja, inicialmente, vazia.
- A cada necessidade de inserção de um elemento na lista, um espaço de memória será solicitado para armazenar o novo dado e depois será adicionado na lista, através de alocação dinâmica.

Lista

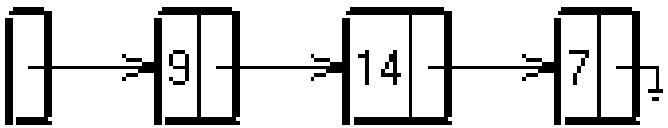


Estrutura de dados

LISTAS ENCADEADAS

- A proposta desse tipo de lista é que a estrutura esteja, inicialmente, vazia.
- A cada necessidade de inserção de um elemento na lista, um espaço de memória será solicitado para armazenar o novo dado e depois será adicionado na lista, através de alocação dinâmica.

Lista

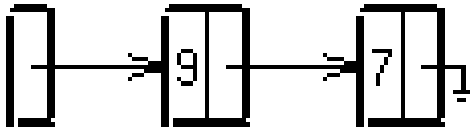


Estrutura de dados

LISTAS ENCADEADAS

- A proposta desse tipo de lista é que a estrutura esteja, inicialmente, vazia.
- A cada necessidade de inserção de um elemento na lista, um espaço de memória será solicitado para armazenar o novo dado e depois será adicionado na lista, através de alocação dinâmica.

Lista



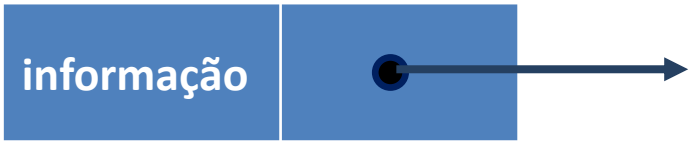
Estrutura de dados

LISTAS ENCADEADAS

– Desta forma, ao invés dos elementos estarem em sequência na memória, como na lista sequencial, os elementos podem ocupar quaisquer célula de memória.

– Para manter a relação de ordem entre os elementos, cada elemento indica qual é o seguinte, armazenando a referência (endereço) do próximo elemento.

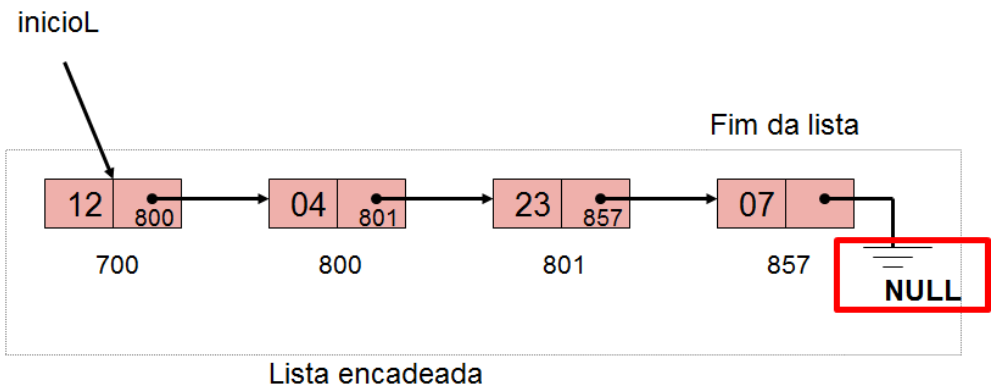
NÓ
Elemento da lista



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

```
#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**

Estrutura de dados

LISTAS ENCADEADAS

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

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

LISTAS ENCADEADAS

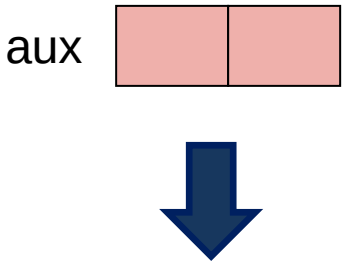
Inserção em listas encadeadas

- 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.

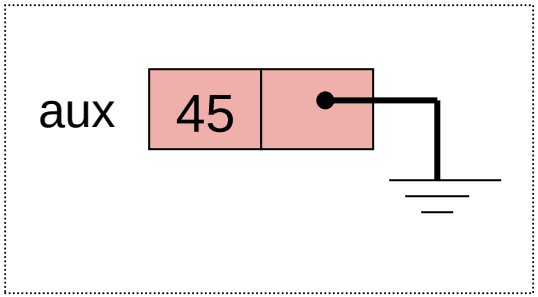
Estrutura de dados

LISTAS ENCADEADAS



CRIAR UM ELEMENTO DO TIPO NÓ:

```
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.

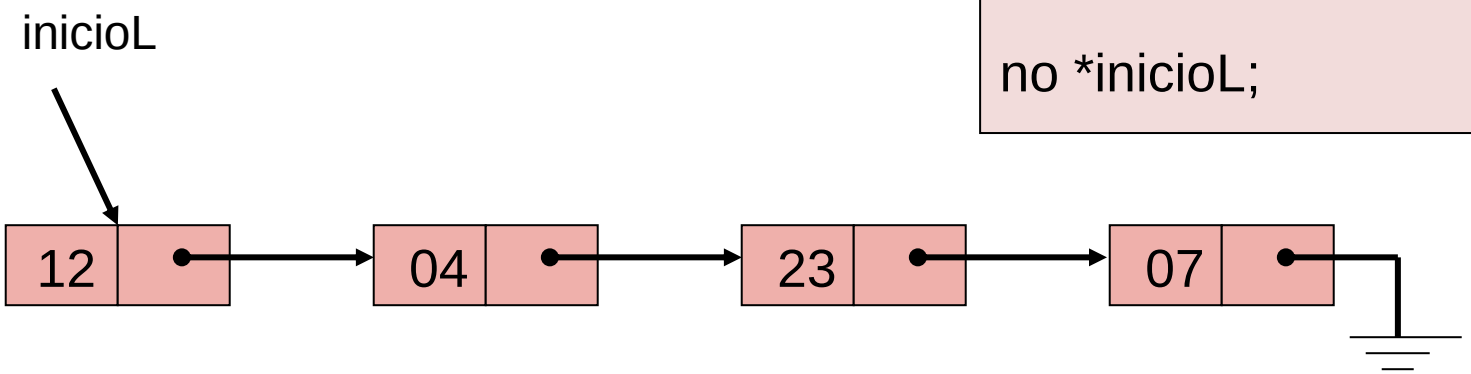
Estrutura de dados

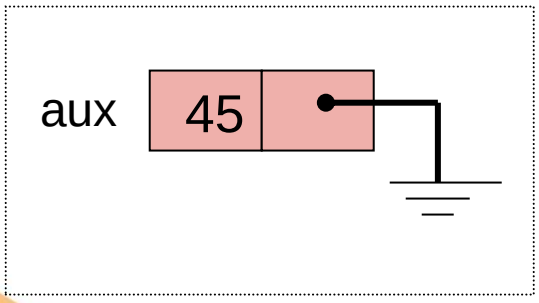
LISTAS ENCADEADAS

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

no *inicioL;
```

INSERÇÃO DE UM NÓ NO INÍCIO






```
cria_no(45);
```

LISTAS ENCADEADAS

Inserção em listas encadeadas

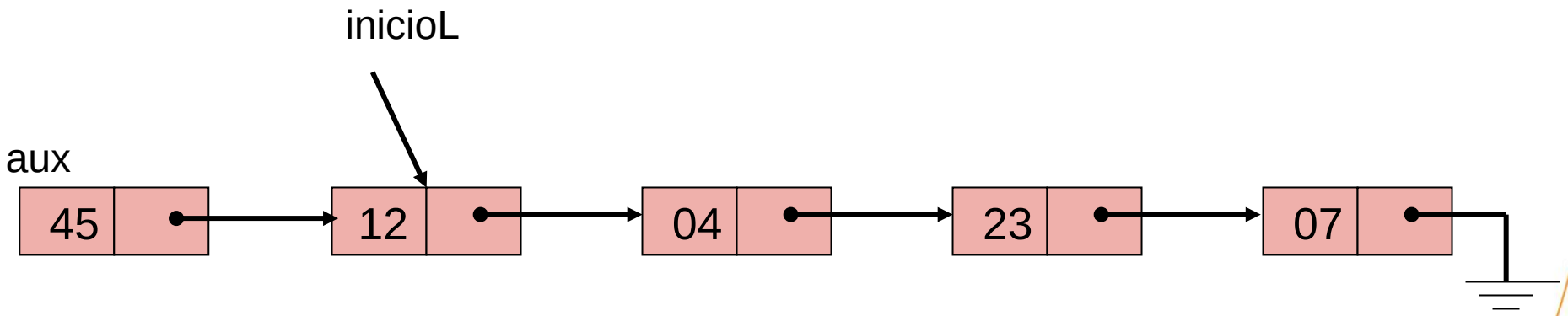
- Inserção no início

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



aux -> prox = inicioL

LISTAS ENCADEADAS

Inserção em listas encadeadas

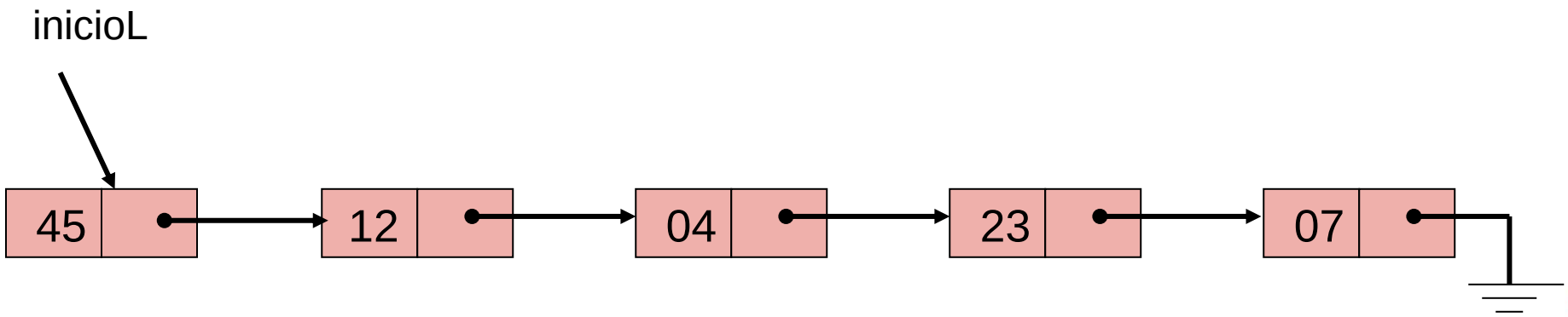
- Inserção no início

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.

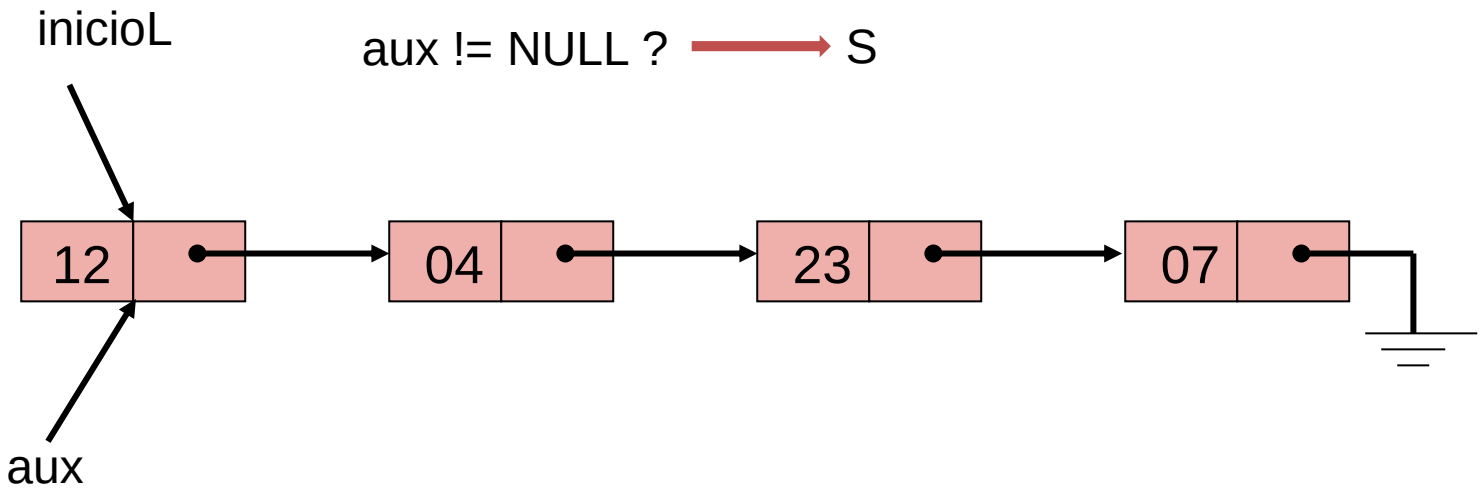
O final é encontrado ao atingir o valor NULL.

Estrutura de dados

LISTAS ENCADEADAS

aux = aux->prox;

CONDIÇÃO DE PARADA PARA PERCORRER

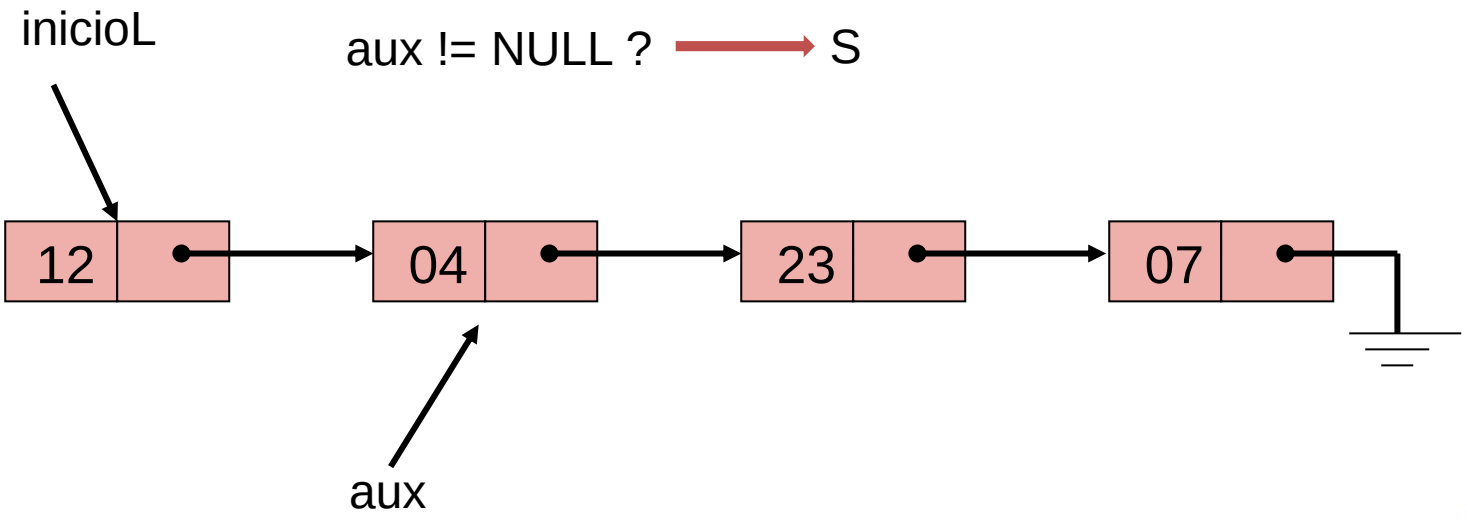


Estrutura de dados

LISTAS ENCADEADAS

aux = aux->prox;

CONDIÇÃO DE PARADA PARA PERCORRER

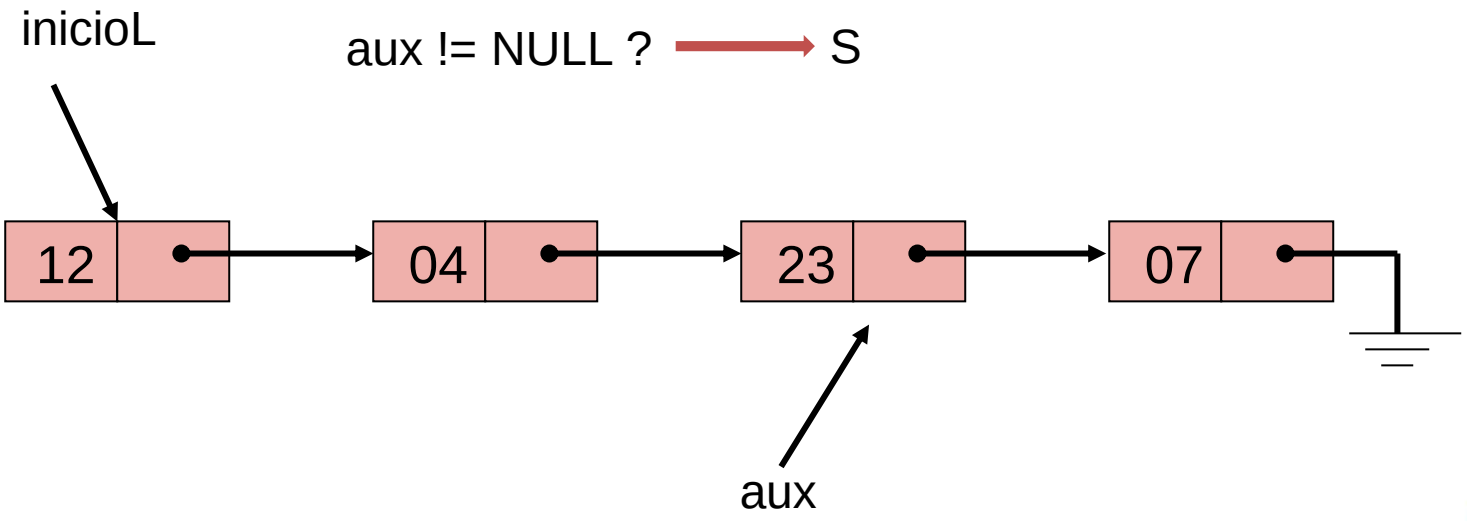


Estrutura de dados

LISTAS ENCADEADAS

aux = aux->prox;

CONDIÇÃO DE PARADA PARA PERCORRER

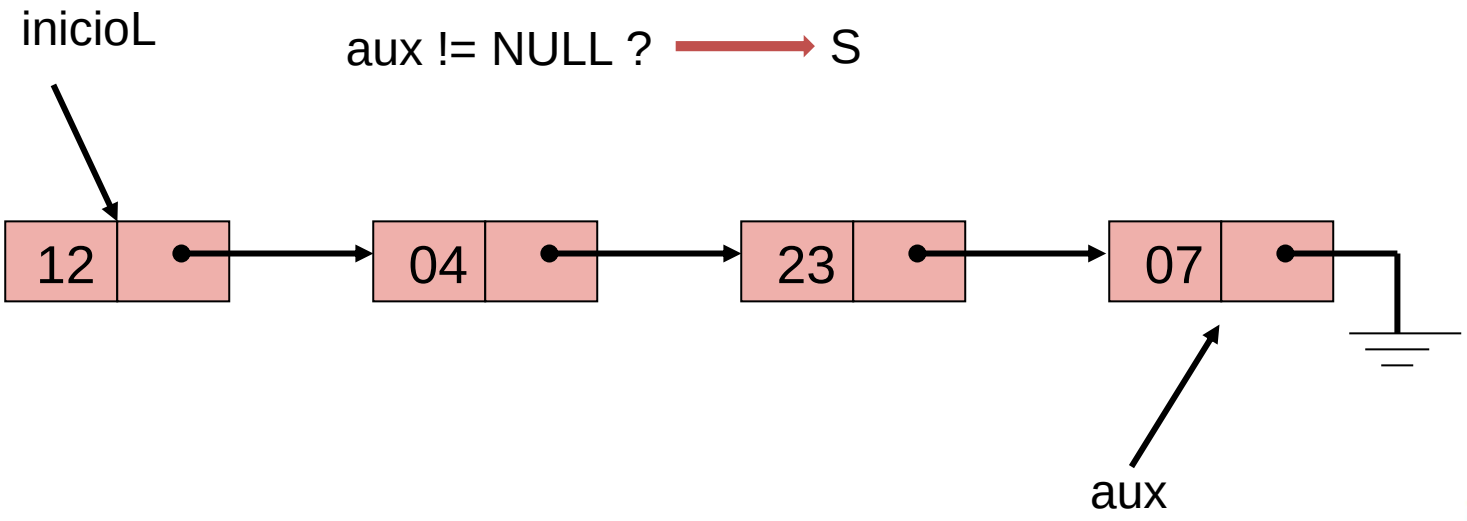


Estrutura de dados

LISTAS ENCADEADAS

aux = aux->prox;

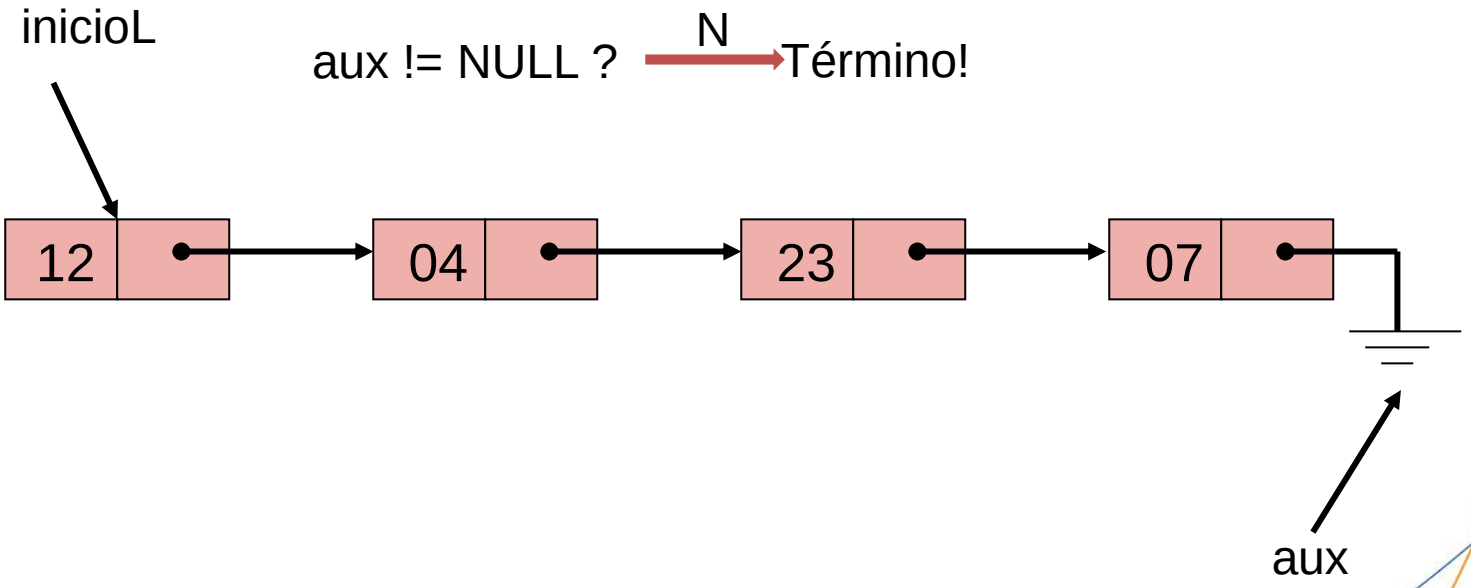
CONDIÇÃO DE PARADA PARA PERCORRER



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

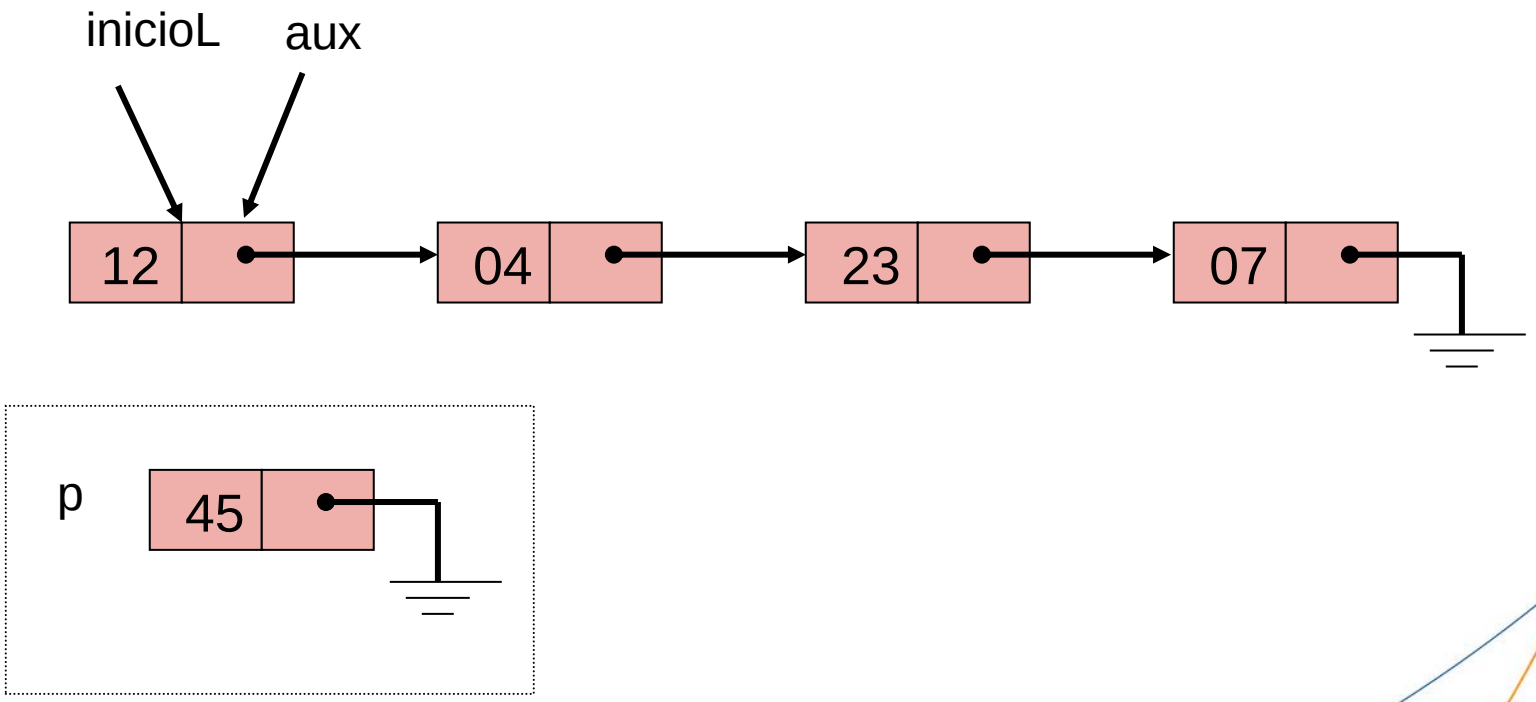
A variável auxiliar (aux) será inicializada com a referência de início da lista e irá percorrendo a lista até chegar no último elemento, sem chegar no valor NULL.

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

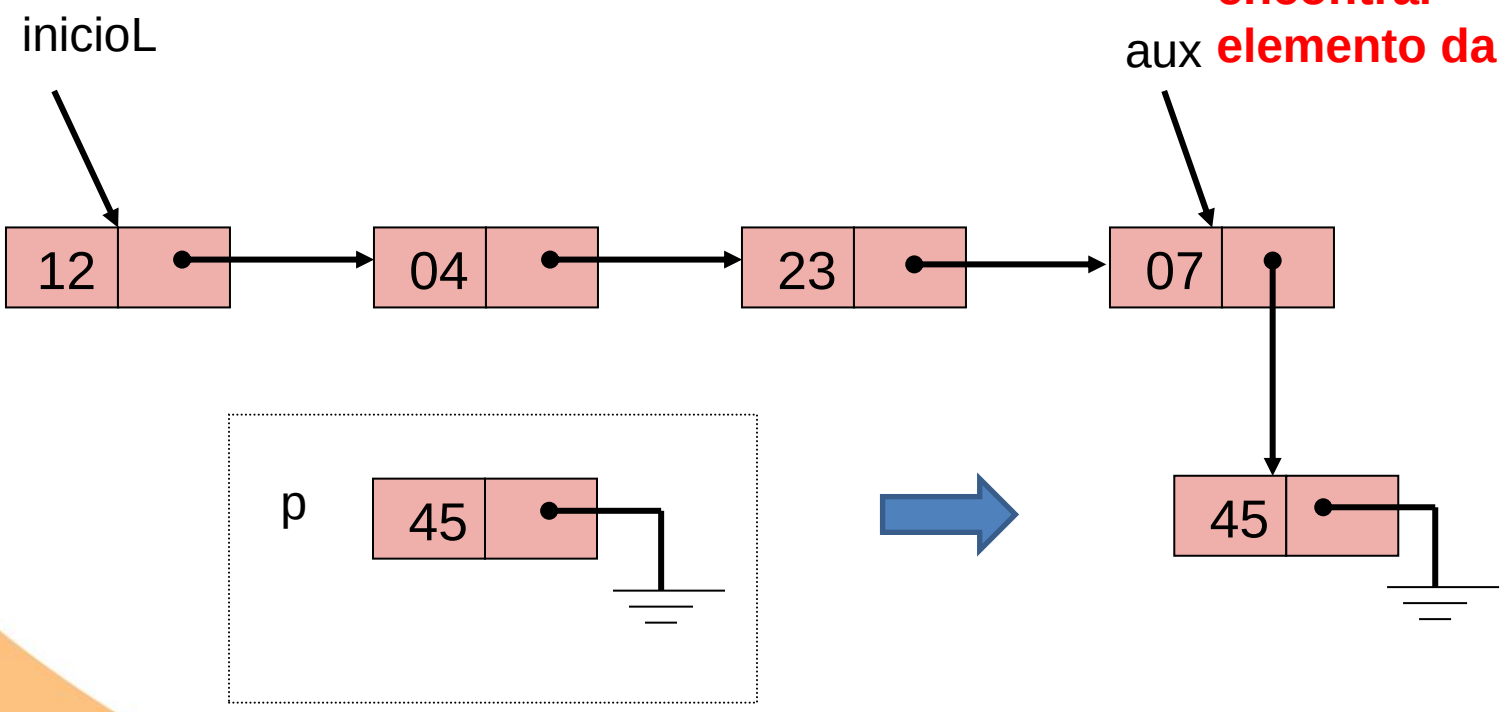


Estrutura de dados

LISTAS ENCADEADAS

INSERÇÃO DE UM NÓ NO FIM

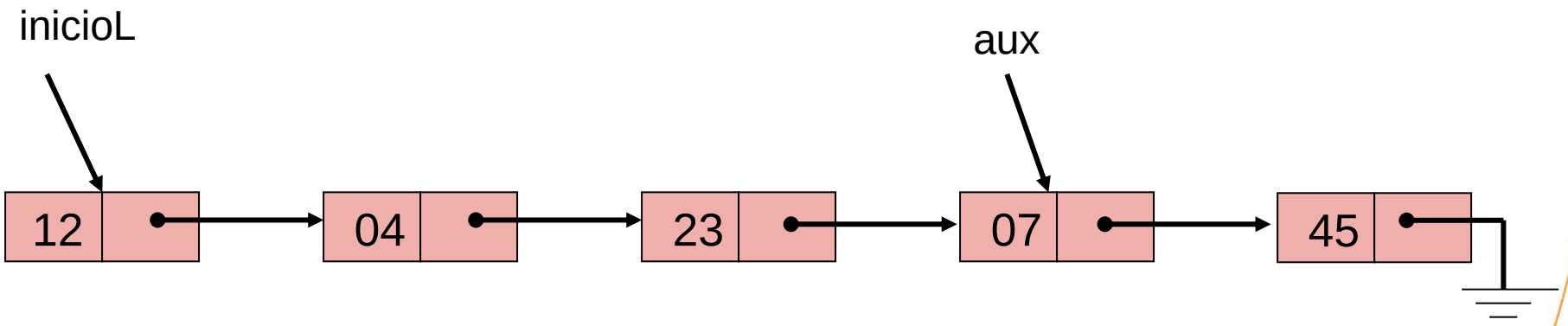
Ponteiro utilizado para encontrar o último elemento da lista



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;
        }
}
```

DÚVIDAS?