

# Estruturas de Dados Elementares: LISTAS

Profª. Rose Yuri Shimizu

## Roteiro

- 1 ESTRUCTURAS DE DATOS ELEMENTARES
- 2 LISTAS ESTÁTICAS
- 3 LISTAS SIMPLEMENTE ENCADEADAS
- 4 LISTA DUPLAMENTE ENCADEADAS

- ## 2 LISTAS ESTÁTICAS

- #### 4 LISTA DUPLAMENTE ENCADEADAS

## Estrutura de Dados Elementares

- Estrutura de dados
  - ▶ Organizam uma coleção de dados
  - ▶ Possuem um conjunto de operações
- Elementar
  - ▶ Utilizados por outras estruturas
- Estrutura elementar: lista

## Roteiro

- 1 ESTRUCTURAS DE DATOS ELEMENTARES
- 2 LISTAS ESTÁTICAS
- 3 LISTAS SIMPLEMENTE ENCADEADAS
- 4 LISTA DUPLAMENTE ENCADEADAS

## LISTA ESTÁTICA

- Conjunto do **mesmo tipo** de dado
- Espaço **consecutivo** na memória RAM
- **Acesso aleatório**: qualquer posição pode ser acessada facilmente através de um **index**
- Nome → corresponde ao **endereço de memória**
- Tamanho **fixo** (stack) ou alocado **dinamicamente** (heap)
- Operações: <https://www.ime.usp.br/~pf/algoritmos/aulas/array.html>


## LISTA ESTÁTICA

- VANTAGEM: fácil acesso
- DESVANTAGEM: difícil manipulação
- Alternativa: LISTAS ENCADEADAS
- <https://www.ime.usp.br/~pf/algoritmos/aulas/lista.html>
- <https://www.ime.usp.br/~pf/mac0122-2002/aulas/llists.html>

## Roteiro

- 1 ESTRUCTURAS DE DATOS ELEMENTARES
- 2 LISTAS ESTÁTICAS
- 3 LISTAS SIMPLEMENTE ENCADEADAS
- 4 LISTA DUPLAMENTE ENCADEADAS

## LISTA SIMPLEMENTE ENCADEADAS

- Conjunto de nós ou células
  - Cada nó é tipo um contêiner que armazena **item** + **link (para outro nó)**
- 
- Mais adequado para **manipulações** do que acessos:
    - ▶ Maior eficiência para **rearranjar os itens** (reapontamentos)
    - ▶ **Não tem acesso direto** aos itens
  - Operações: buscar, inserir, remover



# LISTA SIMPLEMENTE ENCADEADAS

Nós da lista

```
1 //typedef struct item Item;  
2 typedef int Item;
```

```
1 typedef struct registro node;  
2 struct registro {  
3     Item info;  
4     node *prox;  
5 };
```



## LISTA SIMPLEMENTE ENCADEADAS

- Lista sem cabeça



Figura: Lista aponta para o primeiro item

- Lista com cabeça

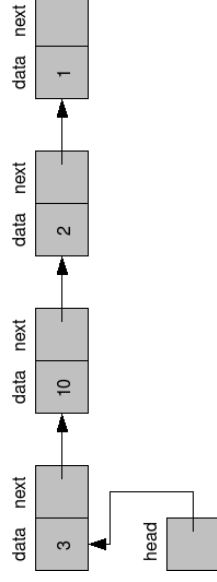


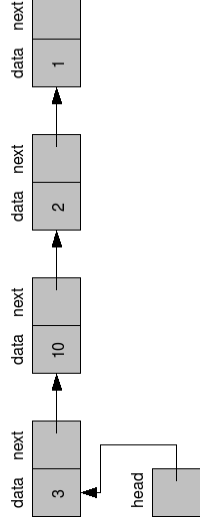
Figura: Item do tipo “cabeça”

- Outra possibilidade: com nó cabeça e cauda

# LISTA SIMPLEMENTE ENCADEADAS

## Lista com cabeça

- Primeiro nó: cabeça da lista
- Marca o início da lista
- Ou pode ser do mesmo tipo dos outros nós
  - ▶ Conteúdo é ignorado
- Ou pode-se criar um tipo específico
  - ▶ Aproveita para guardar metadados
  - ▶ Tamanho da lista e fim da lista, por exemplo
- Elementos da lista: a partir do segundo nó
- Fim da lista: último nó aponta para NULL



## LISTA SIMPLEMENTE ENCADEADAS

Possibilidades de tipos de nós cabeças

```
1 //igual ao restante da lista
2 typedef struct registro node;
3 struct registro {
4     Item info;
5     node *prox;
6 };
```

```
1 //especifico para cabeca
2 typedef struct cabeca head;
3 struct cabeca {
4     int num_itens;
5     node *prox;
6     node *ultimo;
7 };
```



## LISTA SIMPLEMENTE ENCADEADAS

- Implementado na STL (Standard Template Library) do C++
- Implementado na `<sys/queue.h>` da libc
- Algumas operações
  - ▶ Códigos na página da disciplina

## LISTAS SIMPLEMENTE ENCADEADAS

- 1 Escreva uma função que conte o número de células de uma lista encadeada. Faça duas versões: uma iterativa e uma recursiva.
- 2 Escreva uma função que concatene duas listas encadeadas. Faça duas versões: uma iterativa e uma recursiva.
- 3 Escreva uma função que insira uma nova célula com conteúdo  $x$  imediatamente depois da  $k$ -ésima célula de uma lista encadeada. Faça duas versões: uma iterativa e uma recursiva.
- 4 Escreva uma função que troque de posição duas células de uma mesma lista encadeada.

## LISTAS SIMPLEMENTE ENCADEADAS

- 1 **Altura.** A altura de uma célula c em uma lista encadeada é a **distância entre c e o fim da lista**. Escreva uma função que calcule a altura de uma dada célula.
- 2 **Profundidade.** A profundidade de uma célula c em uma lista encadeada é **distância entre o início da lista e c**. Escreva uma função que calcule a profundidade de uma dada célula.
- 3 Escreva uma função que inverta a ordem das células de uma lista encadeada (a primeira passa a ser a última, a segunda passa a ser a penúltima etc.). Faça isso sem usar espaço auxiliar, apenas alterando ponteiros. Dê duas soluções: uma iterativa e uma recursiva.

## LISTAS SIMPLEMENTE ENCADEADAS

- 1 Escreva uma função que encontre uma célula com **conteúdo mínimo**. Faça duas versões: uma iterativa e uma recursiva.
- 2 Escreva uma função para remover de uma lista encadeada todas as células que contêm *y*.
- 3 Escreva uma função que verifique se **duas listas encadeadas são iguais**, ou melhor, se têm o mesmo conteúdo. Faça duas versões: uma iterativa e uma recursiva.
- 4 Listas de strings. Este exercício trata de listas encadeadas que contêm strings ASCII (cada célula contém uma string). Escreva uma função que verifique se uma lista desse tipo está em ordem lexicográfica. As células são do seguinte tipo:

```
1 typedef struct reg {  
2     char *str; struct reg *prox;  
3 } celula ;
```

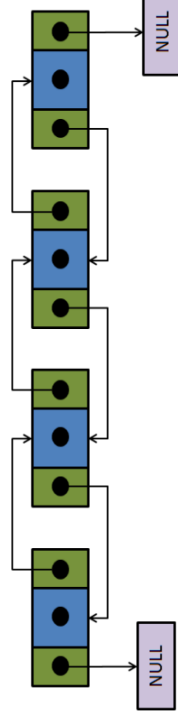


## Roteiro

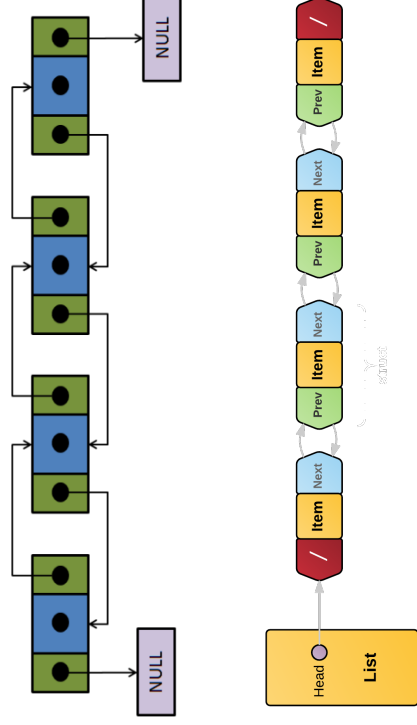
- 1 ESTRUCTURAS DE DATOS ELEMENTARES
- 2 LISTAS ESTÁTICAS
- 3 LISTAS SIMPLEMENTE ENCADEADAS
- 4 LISTA DUPLAMENTE ENCADEADAS

## Estrutura elementar: LISTA DUPLAMENTE ENCADEADAS

- Armazena a informação do nó anterior e posterior
- Muito útil quando ocorrem muitas inserções e remoções, principalmente de elementos intermediários.
- Variações:
  - ▶ Pode ser implementado sem cabeça da lista (head), com cabeça da lista (head) e, com cabeça da lista (head) e a cauda (tail)
    - ★ Cabeça/Cauda do mesmo tipo dos nós da lista: elemento anterior da cabeça aponta sempre para NULL enquanto que no nó cauda quem aponta para NULL é próximo
  - ▶ Circular: conhecida como circular pois o primeiro elemento aponta para o último e vice-versa, formando assim um círculo lógico.



## Estrutura elemental: LISTA DUPLAMENTE ENCADEADAS



## Estrutura elementar: LISTA DUPLAMENTE ENCADEADAS

### Lista com cabeça

```
1 typedef struct registro node;  
2 struct registro {  
3     Item info;  
4     node *ant;  
5     node *prox;  
6 };
```

```
1 typedef struct cabeca head;  
2 struct cabeca {  
3     int num_itens;  
4     node *prox;  
5     node *ultimo;  
6 };
```

## Estrutura elementar: LISTA DUPLAMENTE ENCADEADAS

Lista com cabeça

- Implementado na STL (Standard Template Library) do C++
- Implementado na `<sys/queue.h>` da libc
- Algumas operações
  - ▶ Códigos na página da disciplina