



Estruturas de Dados 1

Disciplina 193704

Prof. Mateus Mendelson

mendelson@unb.br

Universidade de Brasília
Faculdade do Gama
Engenharia de Software



Listas encadeadas

1. Dados Simples

- Referência a um elemento, com domínio e conjunto de operações pré-definido:
 - ✓ Dados numéricos (por exemplo: inteiro, real)
 - ✓ Dados simbólicos (por exemplo: caractere)

2. Dados Compostos

- Referência a um conjunto de elementos com domínio e conjunto de operações a ser estabelecido:
 - ✓ Homogêneos (vetor, matriz, string)
 - ✓ Não necessariamente homogêneo (struct)
 - ✓ Conjuntos, formas de inclusão, acesso, organização, algoritmos eficientes para uso em conjuntos e organizações diversas (sequências, prioridade, hierárquicos, fluxos)

3. Tipo Abstrato de Dado (TAD)

- É um modelo matemático
- Abstrai detalhes após implementação, ou seja, realiza o encapsulamento do modelo matemático
- Implementado sobre um conjunto de elementos
 - ✓ Operações específicas
 - ✓ Semântica especial
 - ✓ A implementação usa as estruturas básicas de uma linguagem de programação formando uma nova estrutura
 - ✓ Dados simbólicos (por exemplo: caractere)

3. Tipo Abstrato de Dado (TAD)

- Exemplos de modelos matemáticos que poderiam ser implementados como TADs:
 - ✓ Ponto
 - ✓ Segmento de reta
 - ✓ Círculo
 - ✓ Números racionais
 - ✓ Números complexos
- Que operações poderiam ser implementadas para cada tipo?

3. Tipo Abstrato de Dado (TAD)

- Seja o TAD Ponto, um tipo de dado que representa um ponto no R^2 . Define-se, então, um tipo abstrato, o *Ponto*, e o conjunto de funções que operam sobre ele:

- ✓ *Cria*: cria um ponto com coordenadas x e y ;
- ✓ *Libera*: libera a memória alocada por um ponteiro;
- ✓ *Acessa*: retorna as coordenadas de um ponto;
- ✓ *Atribui*: escreve em memória novos valores às coordenadas; e
- ✓ *Distancia*: calcula a distância entre dois pontos.

- Implementem esse TAD de tal forma que todas as funções recebam *Ponto* por referência.

3. Tipo Abstrato de Dado (TAD)

- Quais as limitações de se trabalhar com vetores?
 - ✓ Declarar um vetor implica em especificar um tamanho fixo; e
 - ✓ Pode-se utilizar alocação dinâmica e guardar o endereço do primeiro elemento;
 - Problema: não podemos eliminar posições arbitrárias do vetor.
- Possível desperdício ou falta de memória.

4. Listas

- Estrutura que pode ter seu tamanho aumentado e diminuído conforme desejado.
- É possível inserir, acessar e remover elementos em posições arbitrárias.
- Matematicamente, uma lista é uma sequência de 0 ou mais elementos de um certo tipo de dado.
- Se uma lista contém n elementos, n é o tamanho da lista.
- É possível que a lista esteja vazia, com $n = 0$.

4. Listas

- Conjunto de operações de uma lista:
 - ✓ Inserir(x, p, L): inserir x na posição p da lista L ;
 - ✓ Deletar(p, L): deletar o elemento na posição p da lista L ;
 - ✓ ListaVazia(L): criar lista vazia;
 - ✓ Encontrar(x, L): encontrar em que posição da lista L está x ;
 - ✓ Posterior(p, L), Anterior(p, L): encontrar o elemento posterior/anterior ao elemento p da lista L ; e
 - ✓ ImprimirLista(L): imprimir todos os elementos de L .

5. Listas Encadeadas Simples

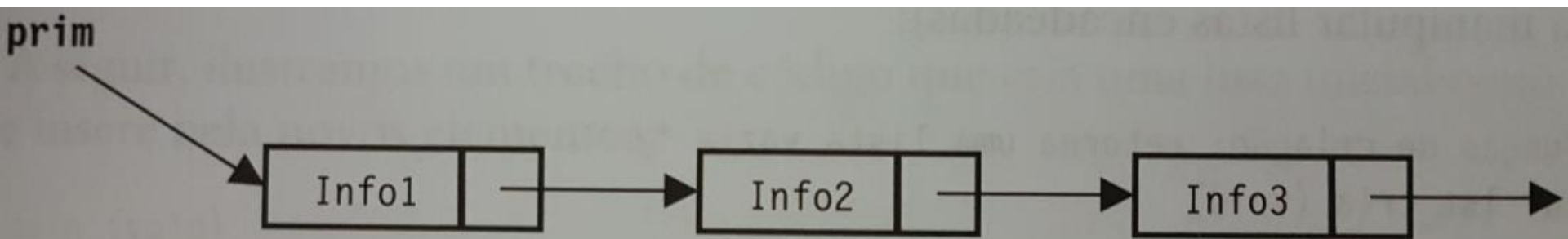
- Permite o armazenamento de um conjunto de dados de forma parecida com um vetor, porém os endereços de seus elementos não se encontram em endereços de memória contíguos.
- Para que haja o crescimento ou diminuição da lista, é necessário o uso de alocação dinâmica.
- Para gerenciar a TAD, é necessário o uso de ponteiros.

5. Listas Encadeadas Simples

- Para cada novo elemento inserido na lista, uma operação de alocação dinâmica é realizada.
- Cada elemento deve armazenar duas informações:
 - ✓ O conteúdo do elemento; e
 - ✓ Ponteiro para o próximo elemento.
 - Quando for o último elemento da lista, o valor do ponteiro é NULL.
- O acesso aos elementos é sequencial.

5. Listas Encadeadas Simples

- Representação de uma lista encadeada:



- Em C, como ficaria o código que define um elemento de uma lista (struct)?

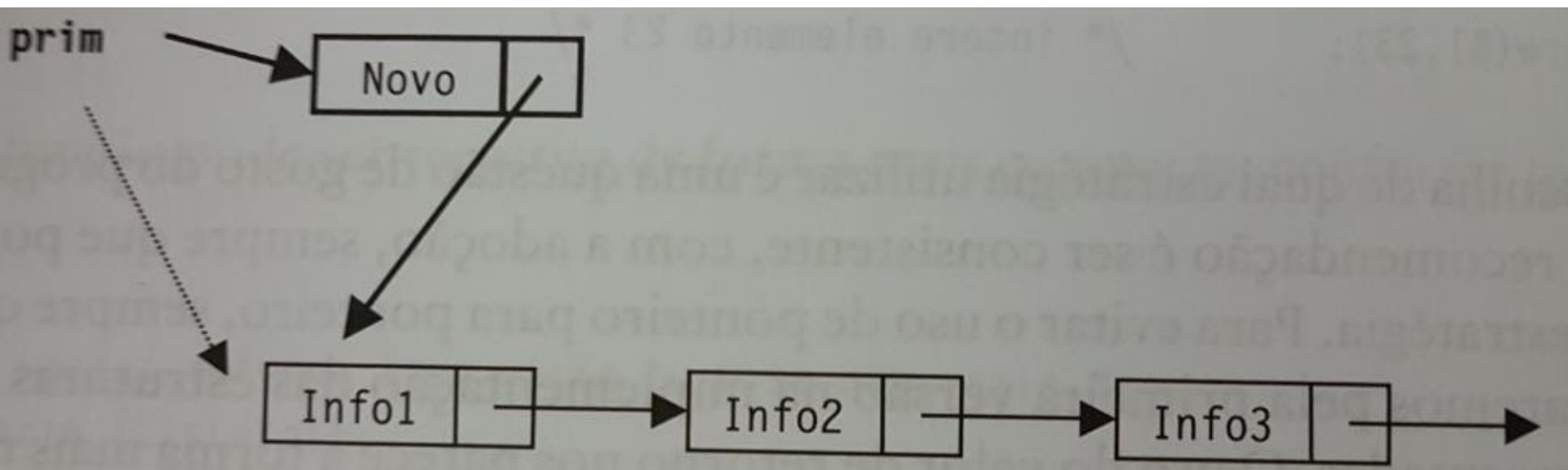
5. Listas Encadeadas Simples

- Listas com cabeça: o conteúdo da primeira célula é irrelevante, servindo apenas para marcar o início da lista. A primeira célula é a cabeça da lista.
- Listas sem cabeça: o conteúdo da primeira célula é tão relevante quanto as demais (vamos adotar esta abordagem).

5. Listas Encadeadas Simples

- Vamos codificar algumas funções, considerando uma lista que armazena números inteiros.

- ✓ Como seria a função que cria uma lista vazia?
- ✓ Como inserir um elemento no começo da lista?

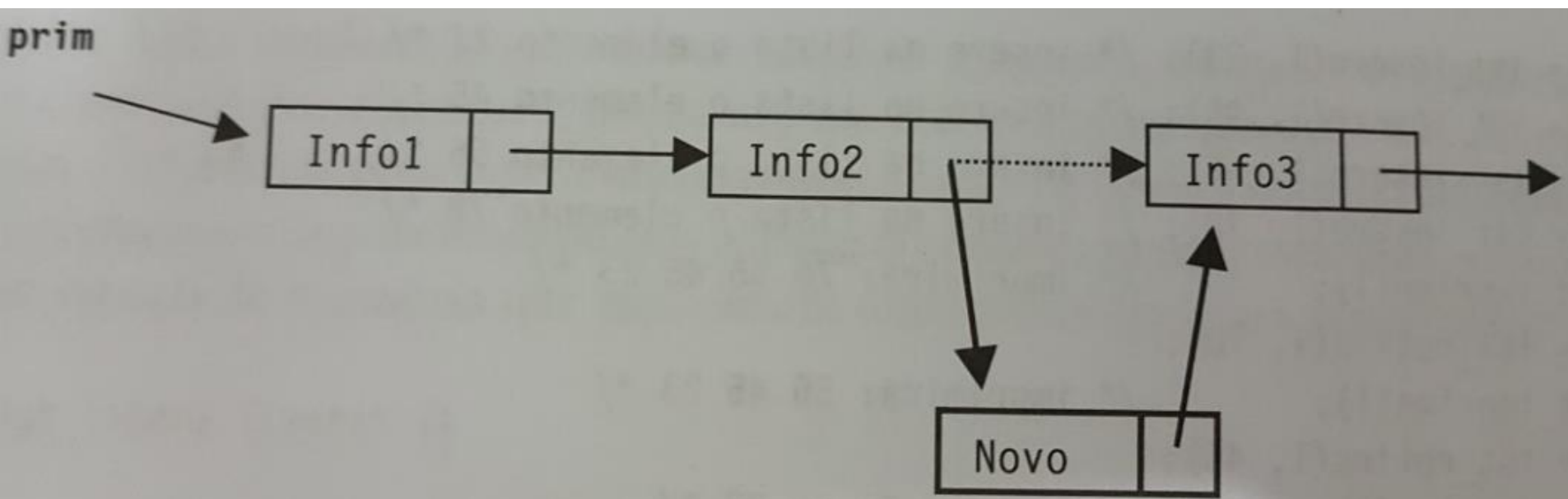


- ✓ Como imprimir uma lista na tela?

5. Listas Encadeadas Simples

- Vamos codificar algumas funções, considerando uma lista que armazena números inteiros.

✓ Como inserir um elemento no meio da lista?

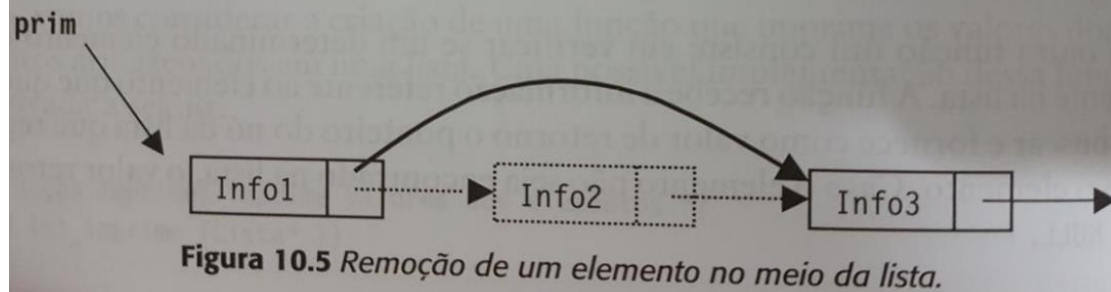
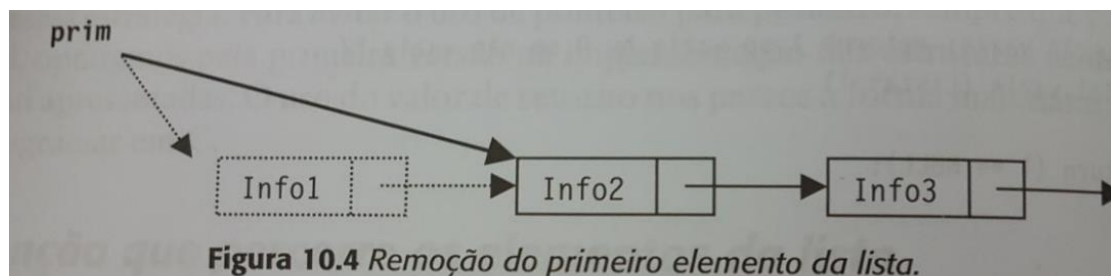


✓ Como buscar um elemento na lista, retornando o endereço dele?

5. Listas Encadeadas Simples

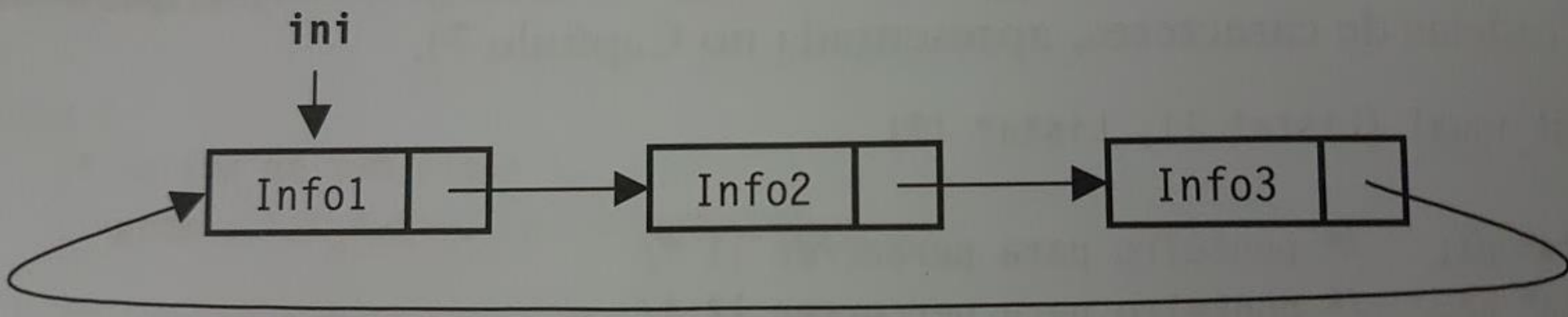
- Vamos codificar algumas funções, considerando uma lista que armazena números inteiros.

- ✓ Como remover o primeiro elemento?
- ✓ Como remover um elemento no meio da lista?



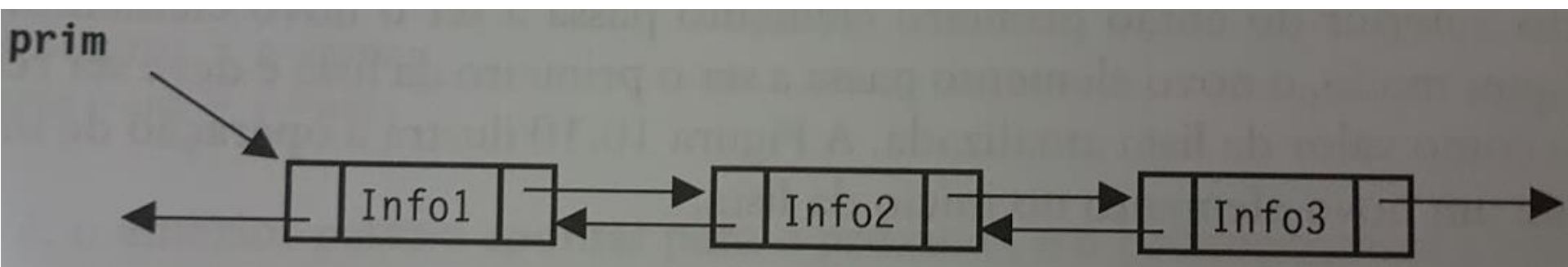
6. Listas Circulares

- Cada elemento da lista é igual ao elemento da lista com encadeamento simples.
- Não se faz distinção entre o primeiro e o último elemento da lista.
- Para testar se a lista foi totalmente percorrida, deve-se guardar o endereço do primeiro elemento lido e compará-lo com o endereço do elemento sendo lido.



7. Listas Duplamente Encadeadas

- Cada elemento, em vez de armazenar apenas o endereço do próximo, armazena o endereço do próximo e do elemento anterior a ele.
- Dessa forma, é possível percorrer a lista em dois sentidos.



8. Exemplo: listas heterogêneas

- Código!