Listas Simplesmente Encadeadas

Disciplina: Estrutura de Dados

Luciano Moraes Da Luz Brum

Universidade Federal do Pampa – Unipampa – Campus Bagé

Email: lucianobrum18@gmail.com

Tópicos

- Revisão – Listas Lineares (contiguidade física).
- Listas Lineares encadeadas.
  - Motivação
  - Operações sobre listas e implementação;
  - Versões recursivas;
- Listas de tipos estruturados.
- Vantagens e desvantagens deste tipo de representação.

Estruturas de Dados

Revisão – Listas Lineares (contiguidade física).

- Ocupa espaço contíguo na memória.
- Acesso randômico aos elementos.
- Saber previamente o número máximo de elementos que vai ser armazenado.
- Operações:
  - Inserção.
  - Impressão.
  - Busca.
  - Exclusão .

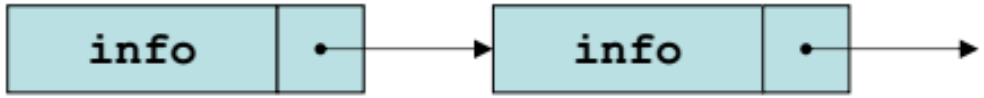
Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

Revisão – Listas Lineares (contiguidade física)
•Listas Lineares encadeadas.
•Motivação;
•Operações sobre listas e Implementação;
•Versões Recursivas;
•Listas de tipos estruturados.
•Vantagens e desvantagens deste tipo de representação.

Listas Encadeadas

➢ Motivação:

- Permite **Alocação Dinâmica de Memória**, ou seja, a lista cresce ou diminui com a execução do programa.
- Operações como inserção e remoção são mais simples.
- Isto é feito através de variáveis do tipo **ponteiro**, ou seja, um elemento aponta para o próximo (possui o endereço de memória do próximo elemento).
- São utilizadas para implementar outras estruturas de dados.



Listas Encadeadas

- Para cada novo elemento inserido na lista, aloca-se um espaço de memória para armazená-lo.
- Como não há garantias de que o espaço é contíguo, não há como acessar os elementos diretamente.
- Para percorrer os elementos da lista, é necessário guardar o encadeamento, armazenando, além dos dados, um ponteiro para o próximo elemento da lista.

Listas Encadeadas

- Uma lista encadeada é composta por nós e cada nó é representado por uma estrutura que contém:
  - A informação armazenada no nó;
  - Um ponteiro para o próximo elemento da lista;
- A lista é representada por um ponteiro para o primeiro elemento (ou nó).
- Do primeiro elemento, podemos acessar o segundo, e do segundo para o terceiro, etc.
- O último elemento possui um ponteiro para uma região de memória inválida, conhecida por NULL, sinalizando que não existem mais elementos na lista encadeada.

Revisão – Listas Lineares (contiguidade física)
•Listas Lineares encadeadas.
•Motivação;
•Operações sobre listas e Implementação;
•Versões Recursivas;
•Listas de tipos estruturados.
•Vantagens e desvantagens deste tipo de representação.

Listas Encadeadas

- Implementação da estrutura LISTA:

```
struct lista{
        int info;
        struct lista* prox;
};
typedef struct lista Lista;
```

Dado que a lista contém.

Ponteiro para o próximo elemento do tipo struct lista.

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
- Motivação;
- Operações sobre listas e Implementação;
- Versões Recursivas;
- Listas de tipos estruturados.
- Vantagens e desvantagens deste tipo de representação.

Listas Encadeadas


```
typedef struct elemento Elemento;

struct elemento{
    int info;
    Elemento *prox;
};
```

Uma forma alternativa.

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
- Motivação;
- Operações sobre listas e Implementação;
- Versões Recursivas;
- Listas de tipos estruturados.
- Vantagens e desvantagens deste tipo de representação.

Listas Encadeadas

- Das operações com listas encadeadas, veremos:
  - Criação da lista;
  - Inserção de elementos;
  - Busca de elementos;
  - Impressão dos elementos na tela;
  - Remoção de elementos;
  - Liberação da estrutura da memória;
  - Inserção em uma lista de elementos ordenada;
- Também veremos como lidar com listas de estruturas compostas (ex: tipo Aluno).

Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

```
Elemento *lst_cria(){
    return NULL;
}
```

Função que cria uma lista vazia, ou seja, o ponteiro para a lista, inicialmente vazia, aponta para NULL.

Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

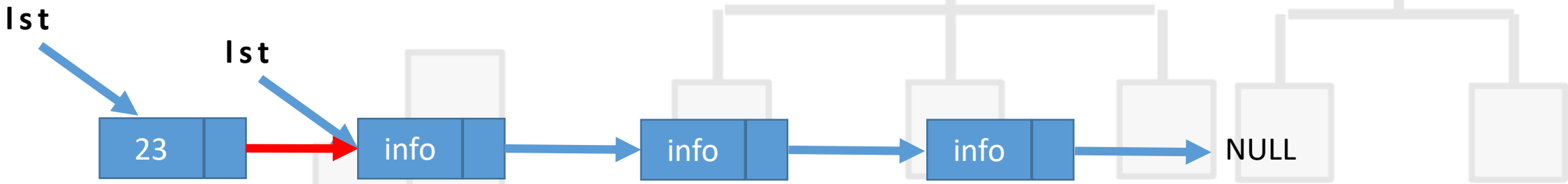
- Das operações com listas encadeadas, veremos:
  - ~~Criação da lista;~~
  - Inserção de elementos;
  - Busca de elementos;
  - Impressão dos elementos na tela;
  - Remoção de elementos;
  - Liberação da estrutura da memória;
  - Inserção em uma lista de elementos ordenada;
- Também veremos como lidar com listas de estruturas compostas (ex: tipo Aluno).

Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.


```
/*inserção no início: retorna a lista atualizada*/
Elemento *lst_insere (Elemento *lst, int val)
{
  Elemento *novo = (Elemento*)malloc(sizeof(Elemento));
  novo->info = val;
  novo->prox = lst;
  return novo;
}

lst = lst_insere(lst, 23); /*insere na lista o elemento 23*/
```



Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

```c
int main (void)
{
    Elemento *lst; /*declara uma lista não inicializada*/
    lst = lst_cria(); /*cria e inicializa lista como vazia*/

    lst = lst_insere(lst, 23); /*insere na lista o elemento 23*/
    lst = lst_insere(lst, 45); /*insere na lista o elemento 45*/
    ...
    return 0;
}

/*deve-se atualizar a
variável que representa a
lista a cada inserção de
um novo elemento*/
```

Listas Encadeadas

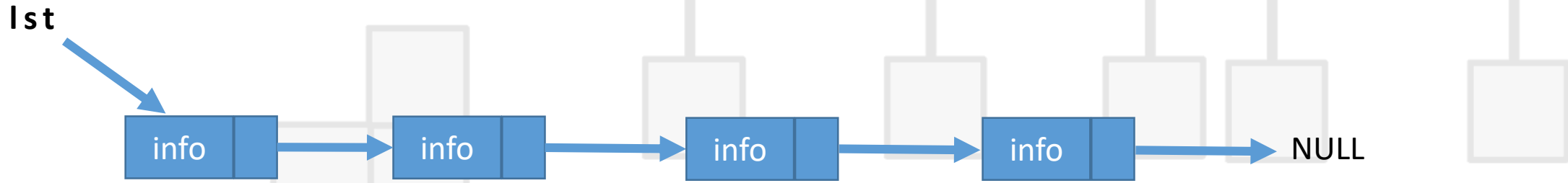
- Das operações com listas encadeadas, veremos:
  - ~~Criação da lista;~~
  - ~~Inserção de elementos;~~
  - Busca de elementos;
  - Impressão dos elementos na tela;
  - Remoção de elementos;
  - Liberação da estrutura da memória;
  - Inserção em uma lista de elementos ordenada;
- Também veremos como lidar com listas de estruturas compostas (ex: tipo Aluno).

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

```c
/*função busca: busca um elemento na lista*/
Elemento *busca (Elemento *lst, int v)
{
    Elemento *p;
    for (p = lst; p != NULL; p = p->prox){
        if(p->info == v)
            return p;
    }
    return NULL; /*não achou o elemento*/
}
```



Listas Encadeadas

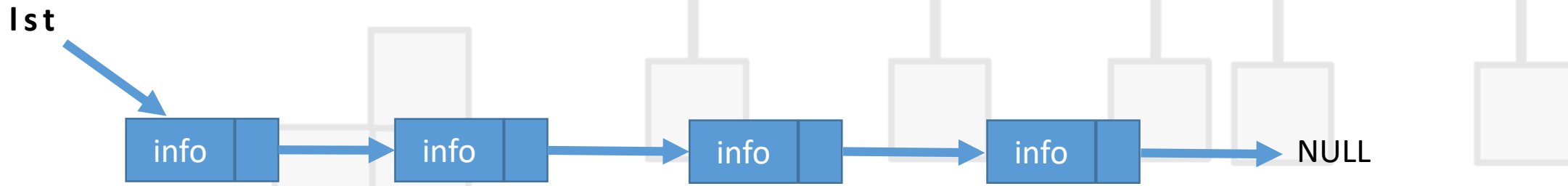
- Das operações com listas encadeadas, veremos:
  - ~~Criação da lista;~~
  - ~~Inserção de elementos;~~
  - ~~Busca de elementos;~~
  - Impressão dos elementos na tela;
  - Remoção de elementos;
  - Liberação da estrutura da memória;
  - Inserção em uma lista de elementos ordenada;
- Também veremos como lidar com listas de estruturas compostas (ex: tipo Aluno).

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

```c
/*função imprime: imprime valores dos elementos*/
void lst_imprime(Elemento *lst)
{
    Elemento *p;
    for (p = lst; p != NULL; p = p->prox)
        printf("info = %d\n", p->info);
}
```



Listas Encadeadas

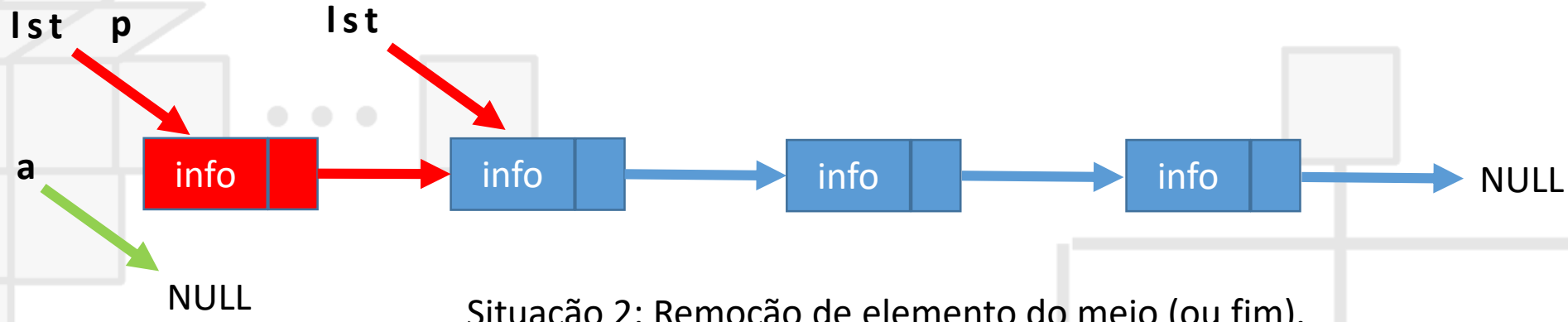
- Das operações com listas encadeadas, veremos:
  - ~~Criação da lista;~~
  - ~~Inserção de elementos;~~
  - ~~Busca de elementos;~~
  - ~~Impressão dos elementos na tela;~~
  - Remoção de elementos;
  - Liberação da estrutura da memória;
  - Inserção em uma lista de elementos ordenada;
- Também veremos como lidar com listas de estruturas compostas (ex: tipo Aluno).

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

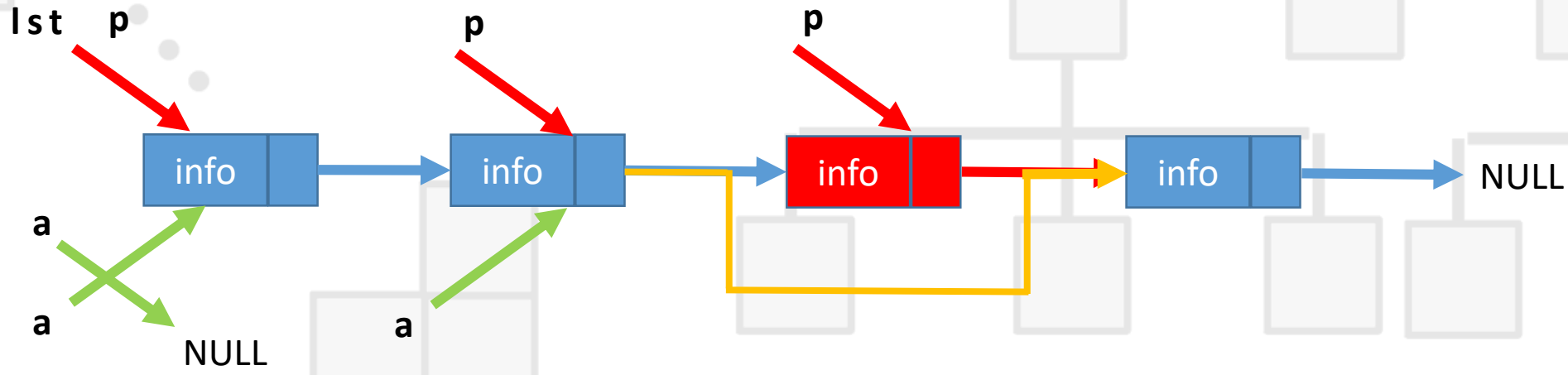
Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

Situação 1: Remoção de elemento do início.



Situação 2: Remoção de elemento do meio (ou fim).



Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

```c
/*função retira: retira elemento da lista*/
Elemento *lst_retira (Elemento *lst, int val)
{
    Elemento *a = NULL; /*ponteiro para o elemento anterior*/
    Elemento *p = lst; /*ponteiro para percorrer a lista*/

    /*procura elemento na lista, guardando o anterior*/
    while (p != NULL && p->info != val){
        a = p;
        p = p->prox;
    }
    /*verifica se achou o elemento*/
    if (p == NULL)
        return lst; /*não achou: retorna lista original*/

    /*retira elemento*/
    if (a == NULL){
        /*retira elemento do início*/
        lst = p->prox;
    }else{ /*retira elemento do meio da lista*/
        a->prox = p->prox;
    }
    free(p);
    return lst;
}
```

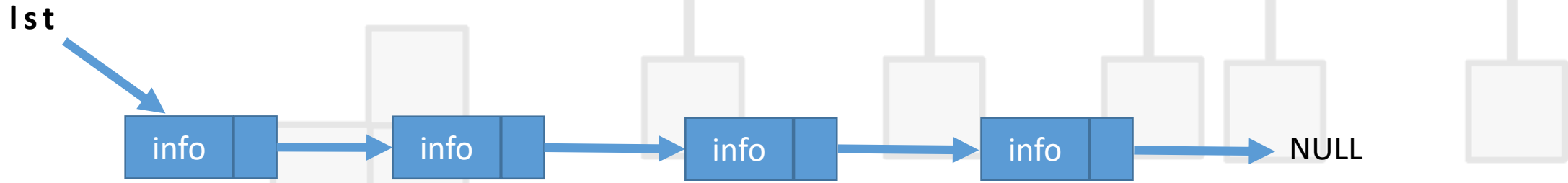
Listas Encadeadas

- Das operações com listas encadeadas, veremos:
  - ~~Criação da lista;~~
  - ~~Inserção de elementos;~~
  - ~~Busca de elementos;~~
  - ~~Impressão dos elementos na tela;~~
  - ~~Remoção de elementos;~~
  - Liberação da estrutura da memória;
  - Inserção em uma lista de elementos ordenada;
- Também veremos como lidar com listas de estruturas compostas (ex: tipo Aluno).

Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

```c
void lst_libera (Elemento *lst)
{
    Elemento*p = lst;
    while (p != NULL){
        Elemento *t = p->prox; /*guarda referência p/ proximo elemento*/
        free (p); /*libera a memória apontada por p*/
        p = t; /*faz p apontar para o próximo*/
    }
}
```



Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

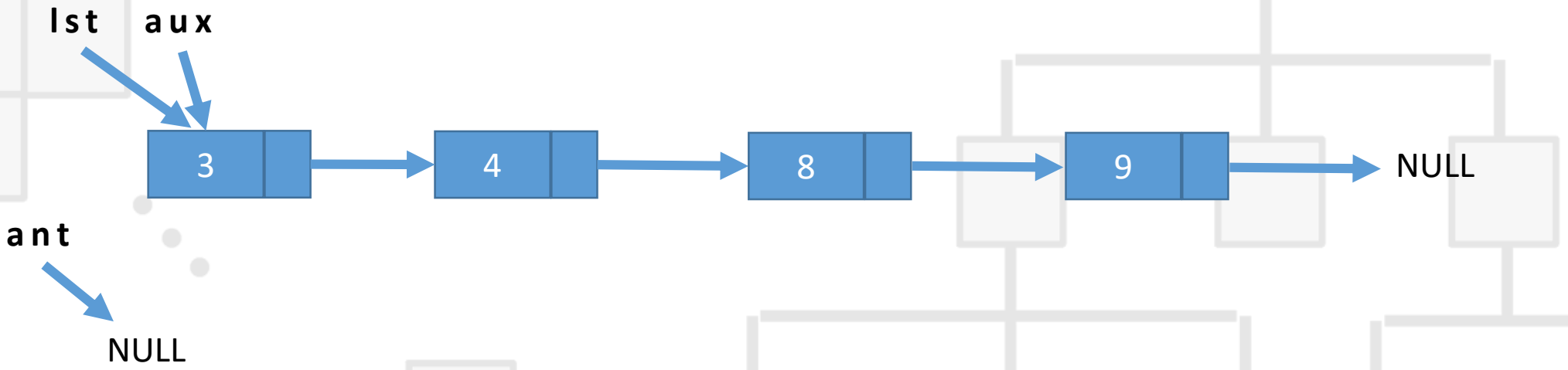
- Das operações com listas encadeadas, veremos:
  - ~~Criação da lista;~~
  - ~~Inserção de elementos;~~
  - ~~Busca de elementos;~~
  - ~~Impressão dos elementos na tela;~~
  - ~~Remoção de elementos;~~
  - ~~Liberação da estrutura da memória;~~
  - Inserção em uma lista de elementos ordenada; (Exercício)
- Também veremos como lidar com listas de estruturas compostas (ex: tipo Aluno).

Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

Objetivo: Inserir o 

na lista ordenada.

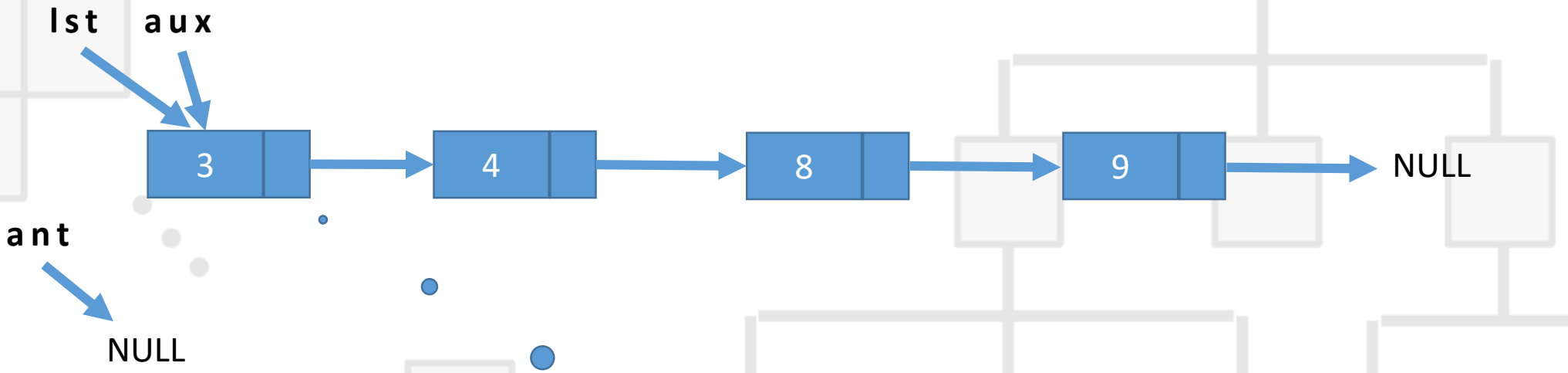


Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

Objetivo: Inserir o 

na lista ordenada.



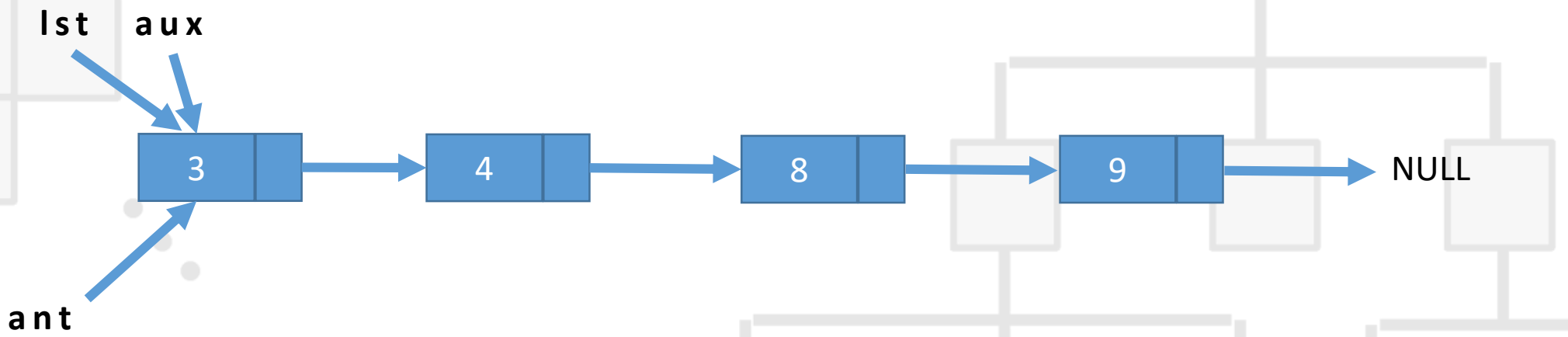
6 é maior que 3? Se sim, ant aponta para aux e aux aponta para aux->prox.

Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

Objetivo: Inserir o 

na lista ordenada.

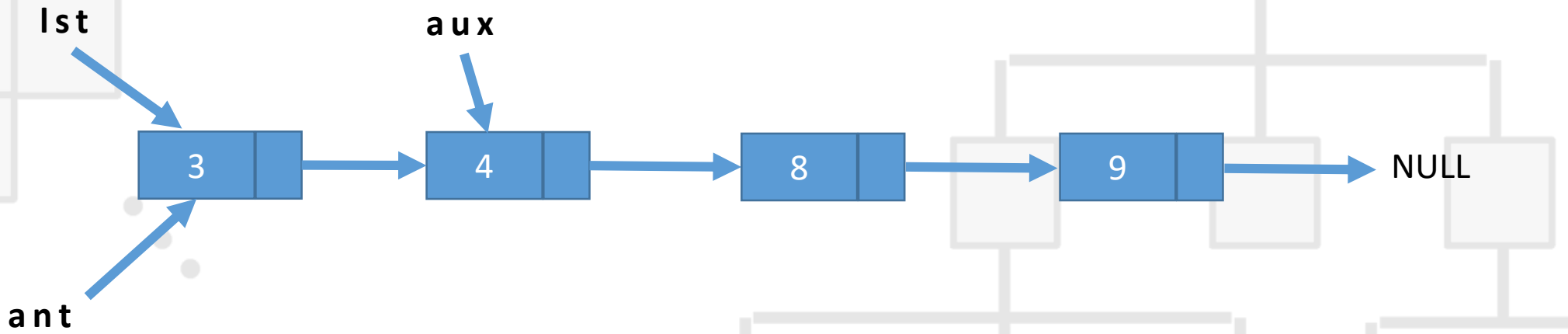


Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

Objetivo: Inserir o 

na lista ordenada.

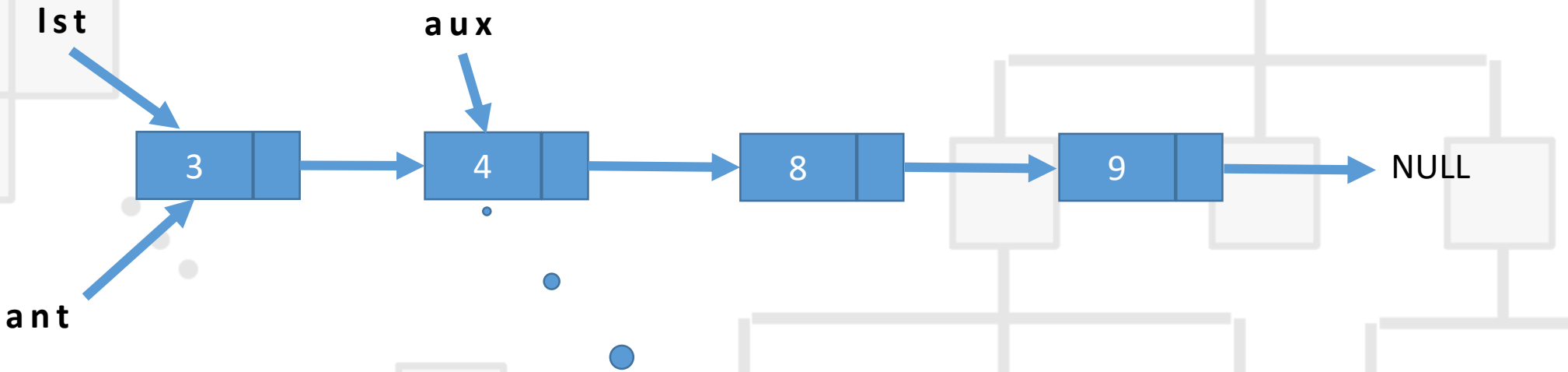


Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

Objetivo: Inserir o 

na lista ordenada.



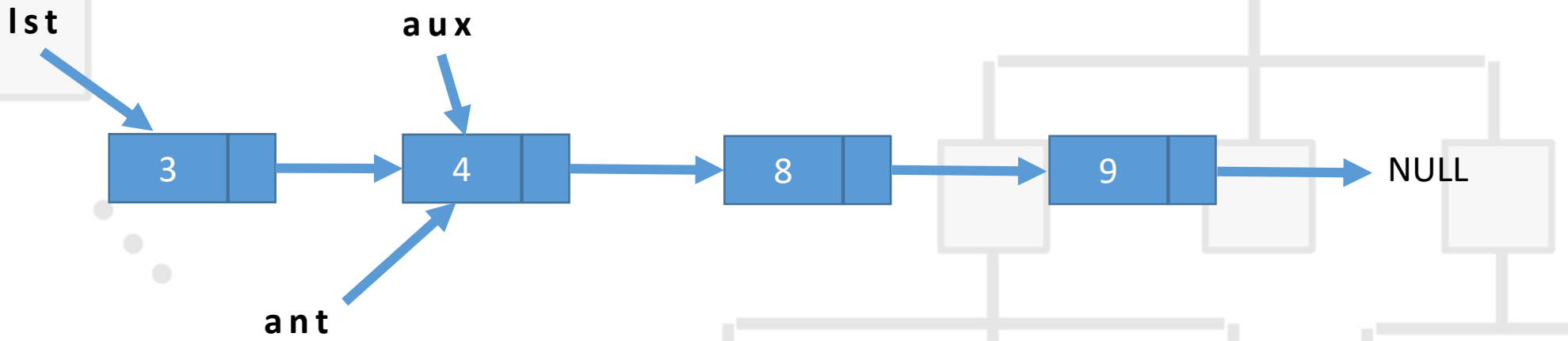
6 é maior que 4? Se sim, ant aponta para aux e aux aponta para aux->prox.

Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

Objetivo: Inserir o 

na lista ordenada.

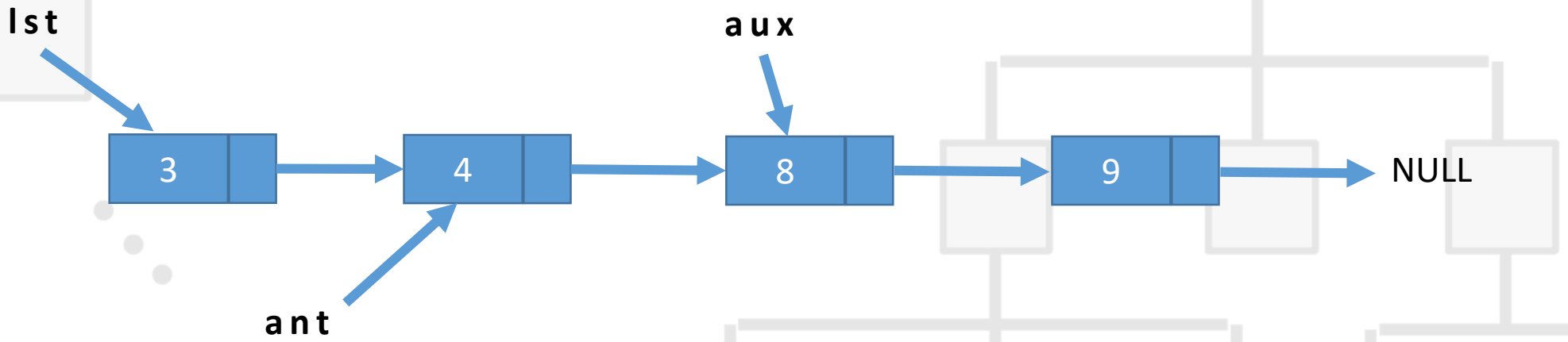


Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

Objetivo: Inserir o 

na lista ordenada.

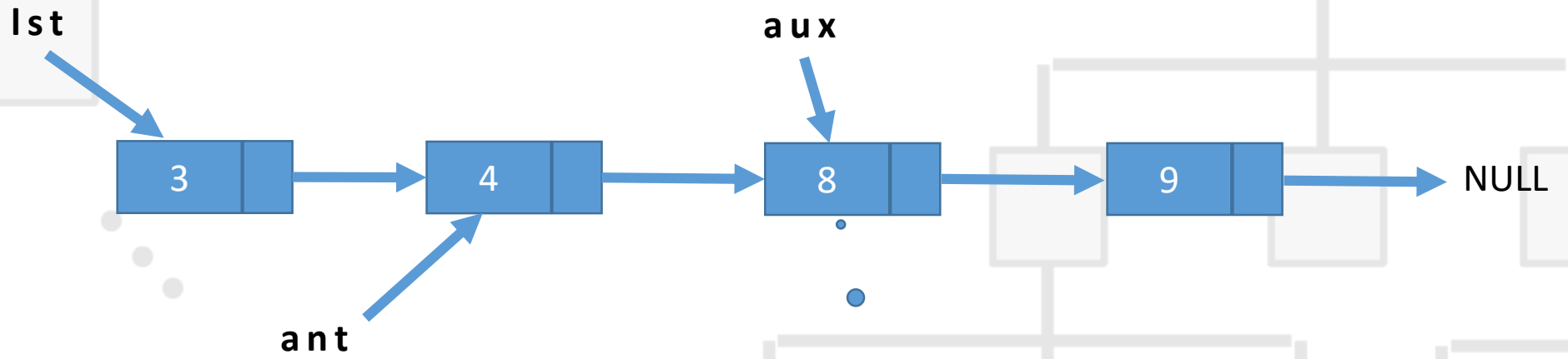


Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

Objetivo: Inserir o 

na lista ordenada.



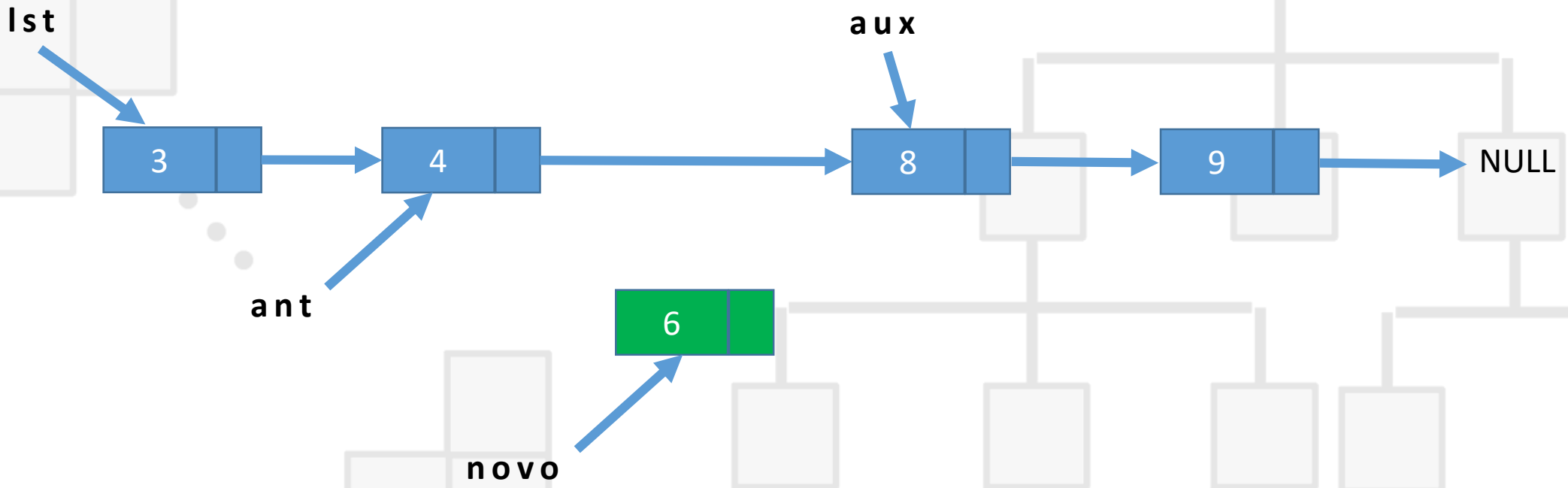
6 é maior que 8? NÃO. É onde vamos inserir o 6 (entre o 4 e o 8).

Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

Objetivo: Inserir o 

na lista ordenada.

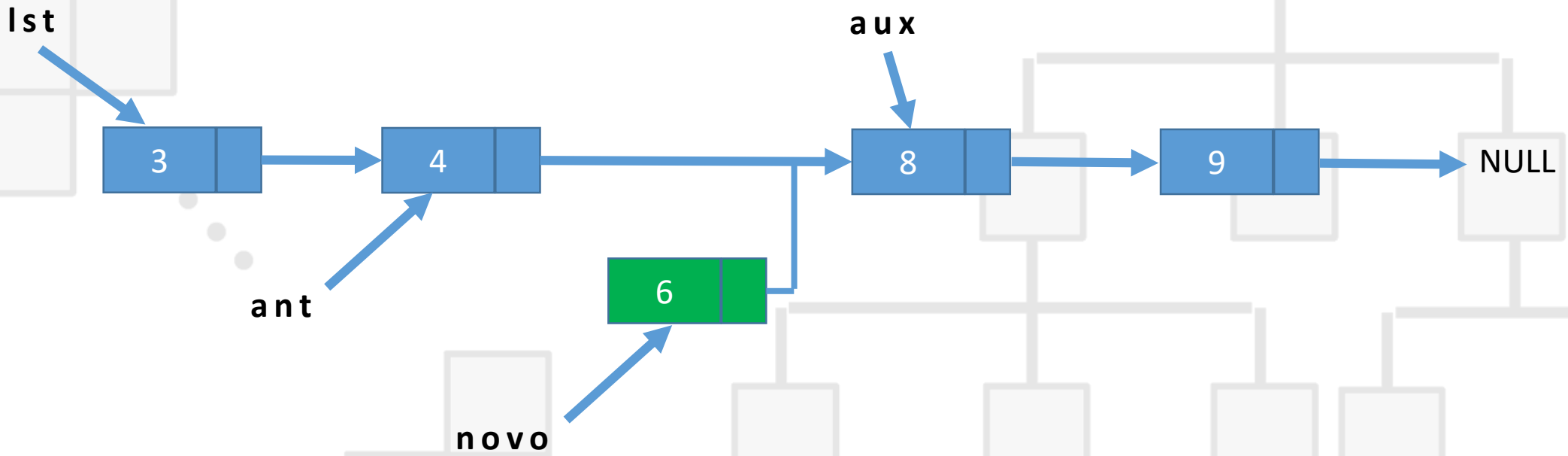


Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

Objetivo: Inserir o 

na lista ordenada.

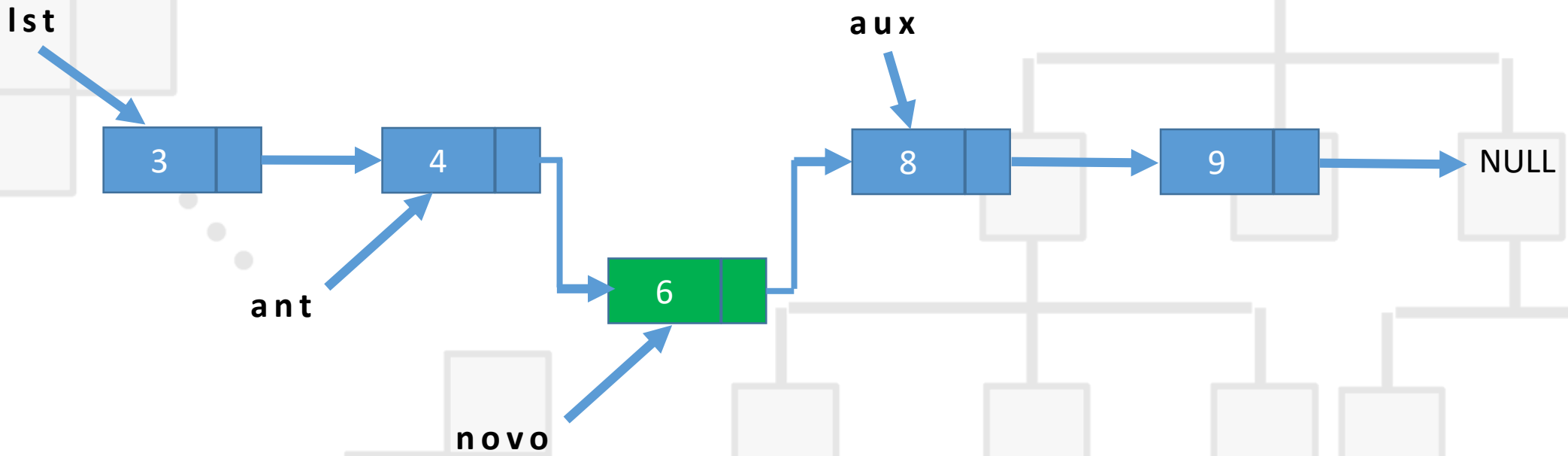


Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

Objetivo: Inserir o 

na lista ordenada.



Listas Encadeadas

- Das operações com listas encadeadas, veremos:
  - ~~Criação da lista;~~
  - ~~Inserção de elementos;~~
  - ~~Busca de elementos;~~
  - ~~Impressão dos elementos na tela;~~
  - ~~Remoção de elementos;~~
  - ~~Liberação da estrutura da memória;~~
  - ~~Inserção em uma lista de elementos ordenada; (Exercício)~~
- Também veremos como lidar com listas de estruturas compostas (ex: tipo Aluno).

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
- Listas de tipos estruturados.
- Vantagens e desvantagens deste tipo de representação.

Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

- A informação de cada nó de uma lista encadeada pode ser mais complexa, sem alterar o encadeamento dos elementos.
- As funções apresentadas para manipular listas de inteiros podem ser modificadas para tratar listas de outros tipos.
- O campo da informação pode ser representado por um ponteiro para uma estrutura, em lugar da estrutura em si.
- A estrutura da lista sempre será:
  - Um ponteiro para informação.
  - Um ponteiro para o próximo nó da lista.

Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
•Listas Lineares encadeadas.
•Motivação;
•Operações sobre listas e Implementação;
•Versões Recursivas;
•Listas de tipos estruturados.
•Vantagens e desvantagens deste tipo de representação.

```
struct retangulo{
    float a;
    float h;
};
typedef struct retangulo Retangulo;

struct elemento{
    Retangulo *info;
    struct elemento *prox;
};
typedef struct elemento Elemento;
```

campo da informação representado por um ponteiro para uma estrutura, em lugar da estrutura em si

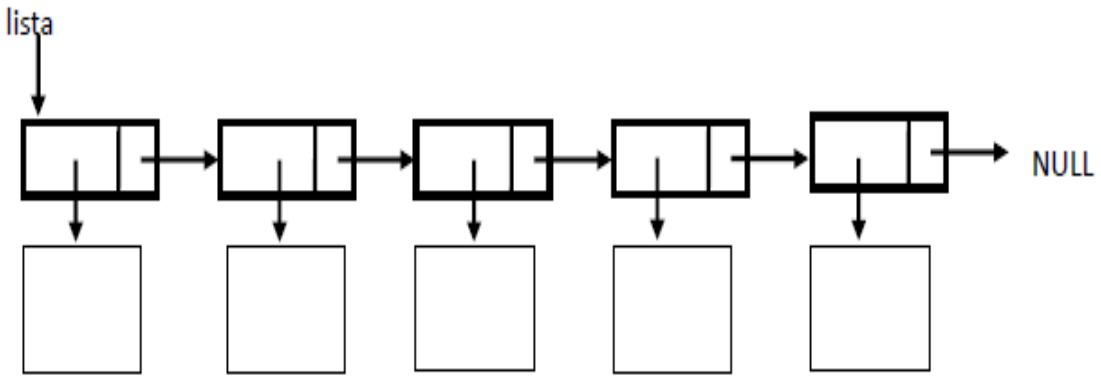
Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.


```
Elemento *aloca (float a, float h)
{
    Retangulo *r = (Retangulo*)malloc(sizeof(Retangulo));
    Elemento *p = (Elemento*)malloc(sizeof(Elemento));
    r->b = b;
    r->h = h;
    p->info = r;
    p->prox = NULL;
    return p;
}
```

Para alocar um nó, são necessárias duas alocações dinâmicas:
uma para criar a estrutura do retângulo e outra para criar a estrutura do nó.

O valor da base associado a um nó **p** seria acessado por: **p->info->b**.



Listas Encadeadas

Revisão – Listas Lineares (contiguidade física)
- Listas Lineares encadeadas.
  - Motivação;
  - Operações sobre listas e Implementação;
  - Versões Recursivas;
  - Listas de tipos estruturados.
  - Vantagens e desvantagens deste tipo de representação.

- A vantagem da representação (utilizando ponteiros) é que, independente da informação armazenada na lista, a estrutura do nó é sempre composta por um ponteiro para a informação e um ponteiro para o próximo nó da lista.

- A representação da informação por um ponteiro nos permite construir listas heterogêneas, isto é, listas em que as informações armazenadas diferem de nó para nó. Diversas aplicações precisam construir listas heterogêneas, pois necessitam agrupar elementos afins mas não necessariamente iguais.

Versões Recursivas de Listas

Revisão – Listas Lineares (contiguidade física)
•Listas Lineares encadeadas.
•Motivação;
•Operações sobre listas e Implementação;
•Versões Recursivas;
•Listas de tipos estruturados.
•Vantagens e desvantagens deste tipo de representação.

- Exemplo: função recursiva para imprimir uma lista.
  - Se a lista for vazia, não imprima nada, caso contrário:
    - Imprima a informação associada ao primeiro nó, dada por **lst->info**
    - Imprima a sub-lista, dada por **lst->prox**, chamando recursivamente a função.

```
/* Função imprime recursiva */
void lst_imprime_rec (Elemento* lst)
{
   if ( ! lst_vazia(lst)) {
      /* imprime primeiro elemento */
      printf("info: %d\n",lst->info);
      /* imprime sub-lista         */
      lst_imprime_rec(lst->prox);
   }
}
```

Versões Recursivas de Listas

- Exemplo: função recursiva para imprimir uma lista.
  - O que acontece se invertermos as linhas do ***printf*** e o ***lst_imprime_rec?***

```
void lst_imprime_rec (Elemento* lst)
{
   if ( ! lst_vazia(lst)) {
      /* imprime primeiro elemento */
      printf("info: %d\n",lst->info);
      /* imprime sub-lista         */
      lst_imprime_rec(lst->prox);
   }
}
```

```
void lst_imprime_rec (Elemento* lst)
{
   if ( !lst_vazia(lst) ) {
      /* imprime sub-lista         */
      lst_imprime_rec(lst->prox);
      /* imprime ultimo elemento   */
      printf("info: %d\n",lst->info);

   }
}
```

Versões Recursivas de Listas

- Exemplo: função para retirar um elemento da lista:
  - Retire o elemento, se ele for o primeiro da lista (ou da sub-lista), caso contrário:
    - Chame a função recursivamente.

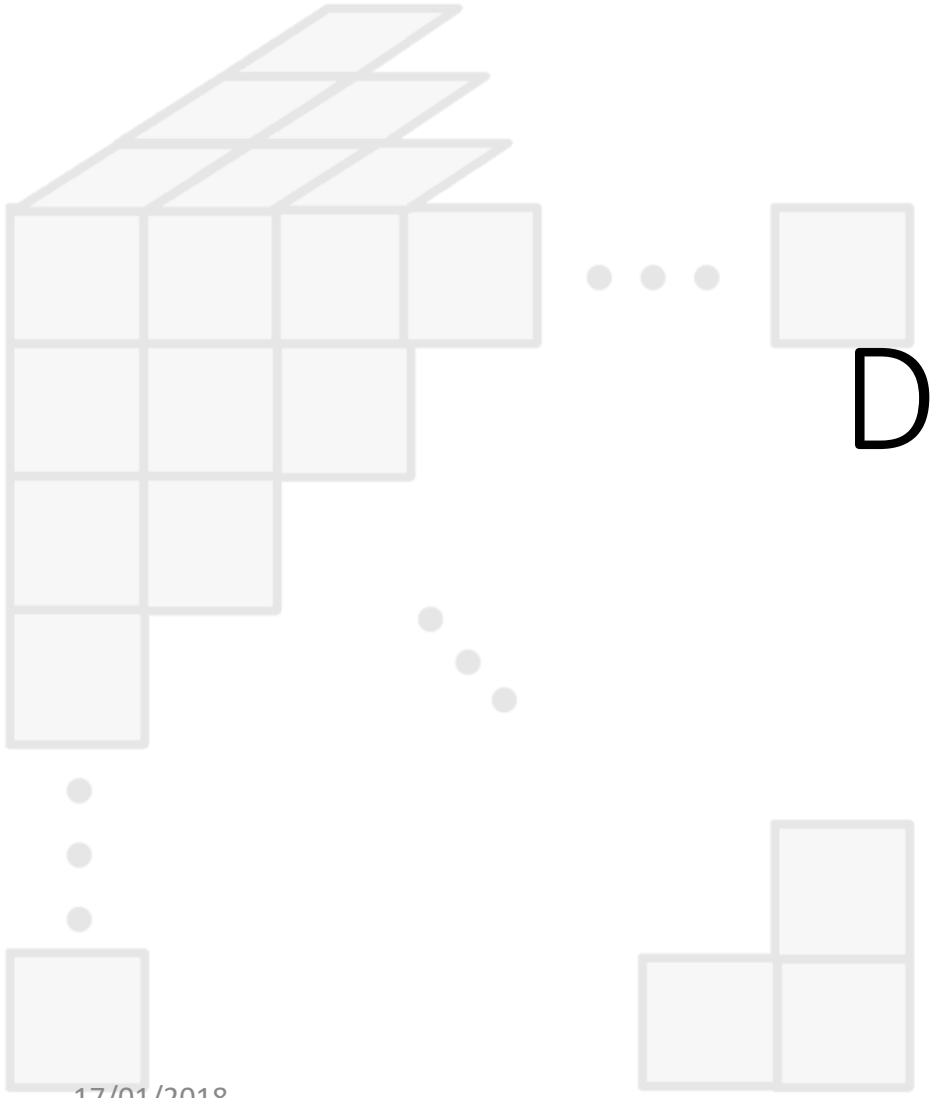
```
/* Função retira recursiva */
Elemento* lst_retira_rec (Elemento* lst, int val)
{
   if (!lst_vazia(lst)) {
      /* verifica se elemento a ser retirado é o primeiro */
      if (lst->info == val) {
         Elemento* t = lst;/* temporário para poder liberar */
         lst = lst->prox;
         free(t);
      }
      else {
         /* retira de sub-lista */
         lst->prox = lst_retira_rec(lst->prox,val);
      }
   }
   return lst;
}
```

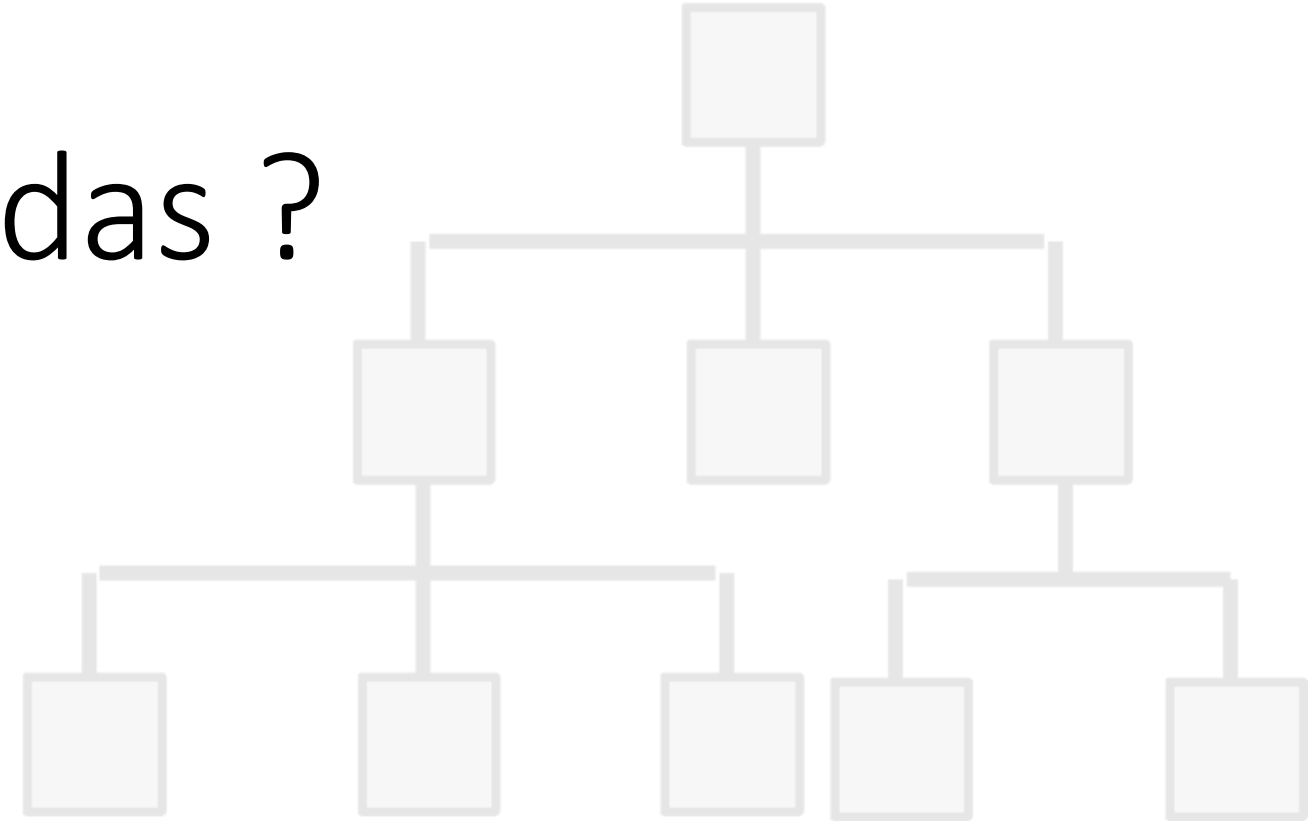
é necessário re-atribuir o valor de **`lst->prox`** na chamada recursiva, já que a função pode alterar a sub-lista

Listas Encadeadas

- **Vantagens:**
  - Lista cresce indeterminadamente, enquanto houver memória livre (Alocação Dinâmica de Memória);
  - As operações de inserção e remoção de elementos não exige a movimentação dos demais elementos (como era feito na lista linear com contiguidade física).
  - Não há desperdício de memória, só usa memória se for necessário.
- **Desvantagens:**
  - Determinar o número de elementos da lista, pois para tanto deve-se percorrer toda a lista;
  - Não é possível acessar diretamente um elemento pela sua posição, pois só é conhecido o primeiro elemento da lista;
  - Acessar o último elemento da lista, pois para acessá-lo, deve-se “visitar” todos os intermediários (na lista linear com contiguidade física, bastava saber o índice do elemento).

Dúvidas ?

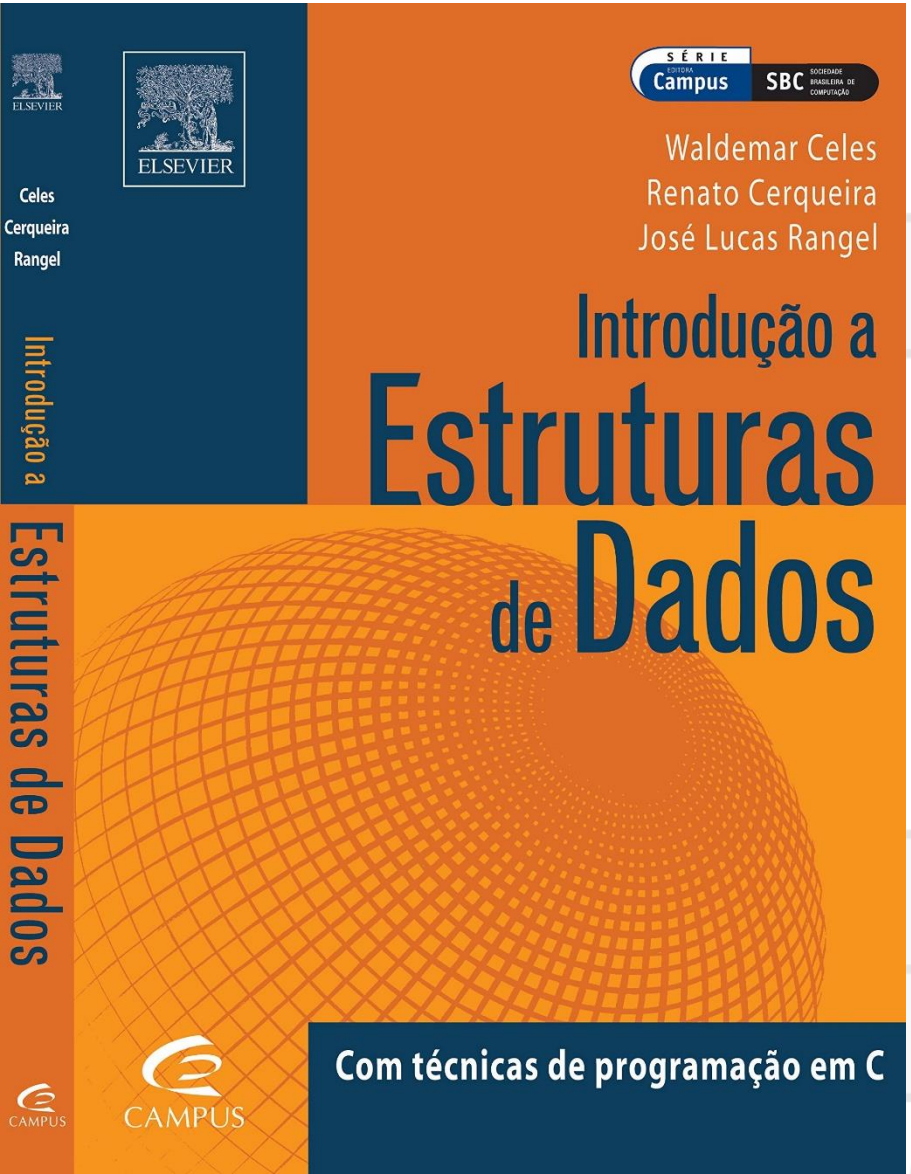




Exercício

- Construa um programa que tenha as seguintes opções:
  - Criar uma lista;
  - Incluir dados em uma lista de número inteiros, mantendo-a ordenada (usar recursividade na inserção);
  - Exclua dados da lista. Este dado deve ser informado pelo usuário;
  - Liberar estrutura da lista da memória (usar recursividade);

Referências



- CELES, Waldemar; CERQUEIRA, Renato; RANGEL, José Lucas. Introdução a Estruturas de Dados ~~c~~om técnicas de programação em C. Rio de Janeiro: Elsevier (Campus), 2004. 4ª Reimpressão. 294 p.