# Listas lineares



# Listas lineares

**Uma Lista Linear (LL) é uma seqüência de nodos**

**São as estruturas de mais simples manipulação**



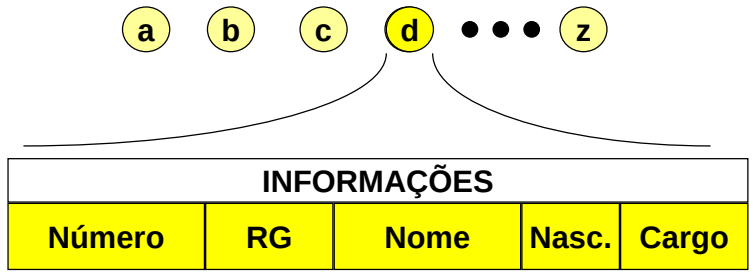
# Lista linear


- **Relação de ordem**
- **Linear - seqüencial**



# Estrutura dos nodos

- **Estrutura interna é abstraída**
- **Pode ter uma complexidade arbitrária**
- **Enfatizado o conjunto de relações existente**



# Definição matemática

Uma lista linear é uma coleção de n ≥ 0 nodos x[1], x[2], ... , x[n], cujas propriedades estruturais relevantes envolvem apenas as posições relativas lineares dos nodos:

n > 0 : x[1] é o primeiro nodo
x[n] é o último nodo

1 < k < n : x[k] é precedido por x[k-1]
e sucedido por x[k+1]

- n = 0 lista vazia
- Lista linear : seqüência de 0 ou mais nodos do mesmo tipo



# Exemplos de aplicações com listas

- notas de alunos
- cadastro de funcionários de uma empresa
- itens em estoque em uma empresa
- dias da semana
- vagões de um trem
- letras de uma palavra
- pessoas esperando ônibus
- cartas de baralho
- precipitações pluviométricas em um mês / dia



# Operações sobre listas lineares

- Criação de uma lista
- Destruição de uma lista
- Inserção de um nodo na i-ésima posição
- Exclusão do i-ésimo nodo
- Acesso ao i-ésimo nodo
- Alteração do i-ésimo nodo
- Combinação de duas ou mais listas
- Classificação da lista
- Cópia da lista
- Determinar cardinalidade da lista
- Localizar nodo através de info

Mais comuns ⇒ devem ser EFICIENTES !



# Representação física de uma lista linear

Representação física das relações existentes entre os nodos e não aquelas internas a eles

- Contigüidade física
- Encadeamento

Algoritmos para implementar operações sobre LL

# Listas lineares
# Contigüidade física



## Lista linear – TAD Genérico

- **Dados**
  - **????**
- **Operações**
  - **?????**



## Lista linear – TAD Genérico

- **Dados**
  - Tipo de dados para armazenar os elementos da lista
  - Componentes para controle da estrutura lista
- **Operações**
  - **inicializa():** Cria lista
  - **insere():** Insere um nodo na k-ésima posição da lista
  - **remove():** R[illegible]o nodo da lista
  - **consult[illegible]** [illegible]o nodo da lista
  - **altera():** S[illegible]ituí o nodo na k-ésima posição da lista por outro
  - **Destroi():** destrói a lista

Excluir / Inserir



## Lista linear – TAD Genérico

- **Dados**
  - **????**
- **Operações**
  - **?????**

# Lista linear - Contigüidade física

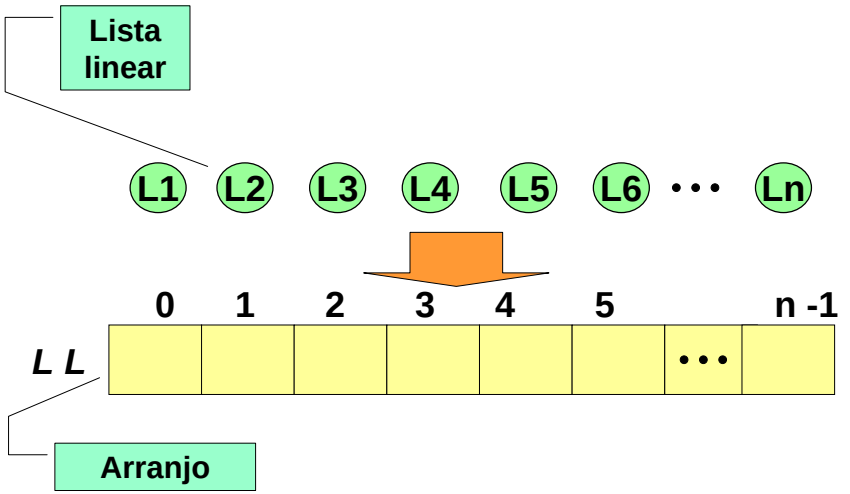
- Seqüencialidade da memória
- Endereços fisicamente adjacentes
  → nodos logicamente adjacentes

**Ex:** suporte da lista é um **arranjo de 1 dimensão**

- todos elementos do arranjo de tipo igual
- cada elemento 1 nodo
- tipo do elemento - tipo do nodo



# Lista linear - Contigüidade física






# Lista linear - TAD Genérico

- **Dados**
  - ?????
- **Operações**
  - ?????



# Lista linear - Contigüidade física

**declaração**

**arquivo.h**

```
typedef struct T_Produto {
        int cod;
        char nome[40];
        float preco;
}TProduto;
```

**aplicação**

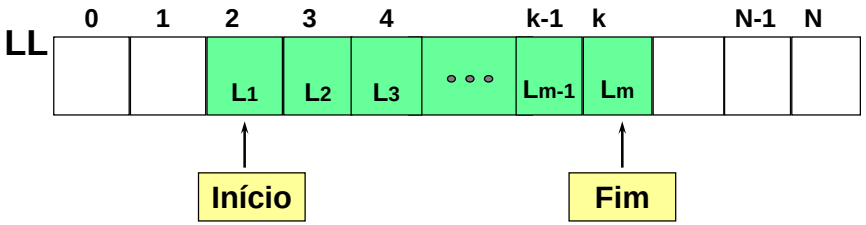
**programa.c**

```
#define MAX 5

TProduto Lista[MAX];
```

# Lista linear - Contigüidade física

**Início e final da lista diferentes do início e final do arranjo**

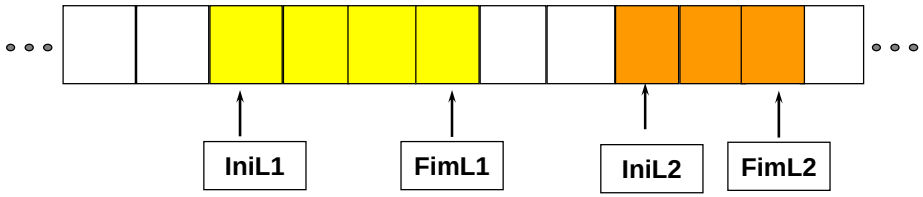
**Variáveis para identificar Início e Final da lista**






# Lista linear - Contigüidade física

**Um mesmo arranjo pode ser utilizado para mais de uma lista linear**






# Lista linear - TAD Genérico

- **Dados**

```
typedef struct T_Produto {
        int cod;
        char nome[40];
        float preco;
}TProduto;
```

  - início e final da lista
  - tamanho do arranjo (ou início e final do arranjo)
- **Operações**
  - **?????**



# Lista linear - TAD Genérico

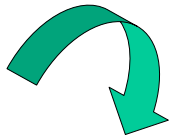
- **Dados**

```
typedef struct T_Produto {
        int cod;
        char nome[40];
        float preco;
}TProduto;

int inicio, fim, MAX;
```

- **Operações**
  - **?????**

# Operações sobre listas lineares em contigüidade física

- criar lista
- inserir nodo
- remover nodo
- alterar nodo
- ...



**Algoritmos**



# Lista linear – TAD Genérico

- **Dados**
  - **TIPO;**
  - início e final da lista
  - tamanho do arranjo (ou início e final do arranjo)
- **Operações**
  - **inicializa(???):** Cria lista
  - **insere(???):** Insere um nodo na k-ésima posição da lista
  - **remove(???):** Remove o k-ésimo nodo da lista
  - **consulta(???):** Consulta o k-ésimo nodo da lista
  - **Destroi(???):** destrói a lista

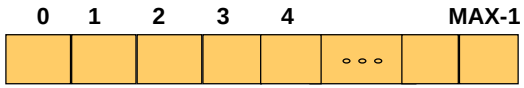


# Criação de uma lista linear implementada sobre um arranjo

- **declarar o arranjo**
- **inicializar variáveis que guardam início e final da lista**

**TProduto Lista[ ];**

***Ini = Fim = -1***




**Lista vazia**



# Lista linear – TAD Genérico

- **Dados**
  - **TIPO;**
  - início e final da lista
  - tamanho do arranjo (ou início e final do arranjo)
- **Operações**
  - **void inicializa (TProduto t[], int *inicio, int *fim);**
  - **insere(???):** Insere um nodo na k-ésima posição da lista
  - **remove(???):** Remove o k-ésimo nodo da lista
  - **consulta(???):** Consulta o k-ésimo nodo da lista
  - **destroi(???):** destrói a lista

# Inicializa

```
typedef struct T_Produto {
        int cod;
        char nome[40];
        float preco;
}TProduto;

TProduto Lista[MAX];
```

```
void inicializa ( TProduto t[], int *inicio, int *fim) {
    int i;

    for (i=0; i<MAX; i++) {
        strcpy(t[i].nome,"");
        t[i].cod=0;
        t[i].preco=0;
    }

    *inicio = -1;
    *fim = -1;
}
```

**Ex: Lista de informações de produtos de uma empresa**



# Lista linear – TAD Genérico

- **Dados**

```
typedef struct T_Produto {
   int cod;
         char nome[40];
         float preco;
}TProduto;

int inicio, fim, maximo;

TProduto Lista[MAX];
```

- **Operações**
  - void inicializa (TProduto t[], int *inicio, int *fim);
  - **insere(???):** Insere um nodo na k-ésima posição da lista
  - **remove(???):** Remove o k-ésimo nodo da lista
  - **consulta(???):** Consulta o k-ésimo nodo da lista
  - **Destroi(???):** destrói a lista

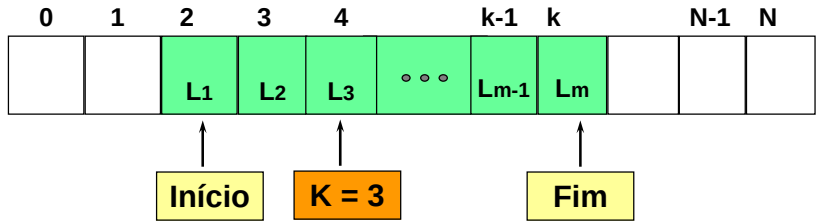


# Lista linear – TAD Genérico

- **Dados**
  - **TIPO**
  - início e final da lista
  - tamanho do arranjo (ou início e final do arranjo)
- **Operações**
  - void inicializa (TProduto t[], int *inicio, int *fim);
  - **insere(???):** Insere um nodo na k-ésima posição da lista
  - **remove(???):** Remove o k-ésimo nodo da lista
  - **consulta(???):** Consulta o k-ésimo nodo da lista
  - **destroi(???):** destrói a lista

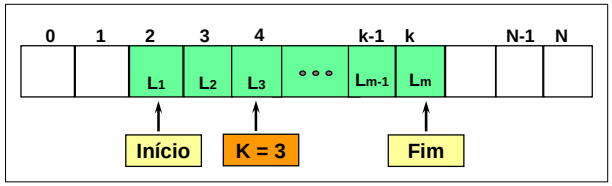


# Acessar o "k-ésimo" nodo de uma lista




**Acessar para**
- **Consulta a informações do nodo**
- **Alteração de algum campo de informação do nodo**

# Procedimento para consultar o "k-ésimo" nodo




- Recebe **posicao** (**ordem** do nodo na lista)
  - nome do arranjo ***LL***
  - índices ***Ini***, ***Fim*** de controle da lista
- Devolve código do nodo consultado ou -1 se não existir

---

**Testar:**

- se o **k**-ésimo nodo faz parte da lista
  - pode ser o primeiro (índice *Ini*) ...
  - até o último (índice *Fim - Ini + 1*)
- se a lista não está vazia



# Lista linear - TAD Genérico

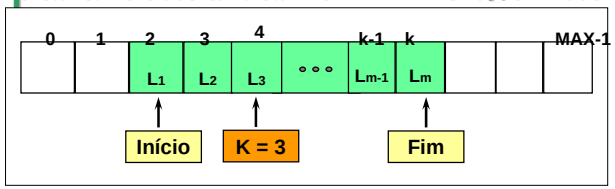
- **Dados**

```
typedef struct T_Produto {
  int cod;
        char nome[40];
        float preco;
}TProduto;

int inicio, fim, maximo;

TProduto Lista[MAX];
```

- **Operações**
  - void inicializa (TProduto t[], int *inicio, int *fim);
  - **insere(???):** Insere um nodo na k-ésima posição da lista
  - **remove(???):** Remove o k-ésimo nodo da lista
  - int consulta ( TProduto t[], int inicio, int fim, int posicao)
  - **Destroi(???):** destrói a lista



# Procedimento para consultar o "k-ésimo" nodo



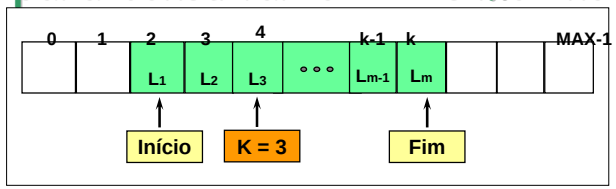

```
int consulta ( TProduto t[], int inicio, int fim, int posicao) {

  if ( (posicao > ??? )   ||        (posicao < ???))
      return -1;
  else
      return t[inicio+posicao-1].cod;
}
```

depois do fim

antes do início



# Procedimento para consultar o "k-ésimo" nodo




```
int consulta ( TProduto t[], int inicio, int fim, int posicao) {

  if ( (posicao > fim - inicio + 1 ) || (posicao < 1))
      return -1;
  else
      return t[inicio+posicao-1].cod;
}
```

depois do fim

antes do início

# Lista linear - TAD Genérico

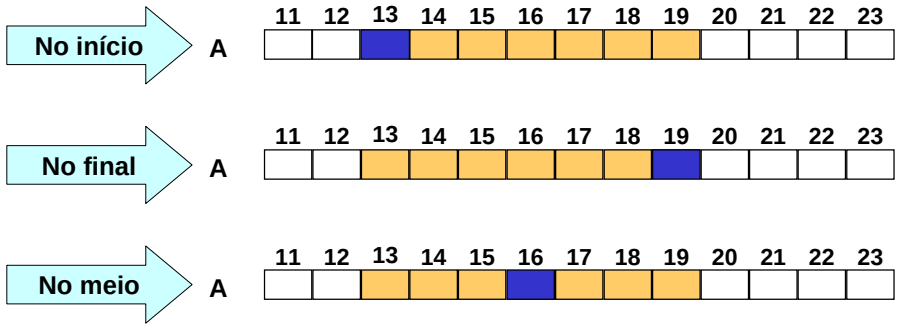
- **Dados**

```
typedef struct T_Produto {
   int cod;
         char nome[40];
         float preco;
}TProduto;

int inicio, fim, maximo;

TProduto Lista[MAX];
```

- **Operações**
  - void inicializa (TProduto t[], int *inicio, int *fim);
  - **insere(???):** Insere um nodo na k-ésima posição da lista
  - **remove(???):** Remove o k-ésimo nodo da lista
  - int consulta ( TProduto t[], int inicio, int fim, int posicao)
  - **Destroi(???):** destrói a lista



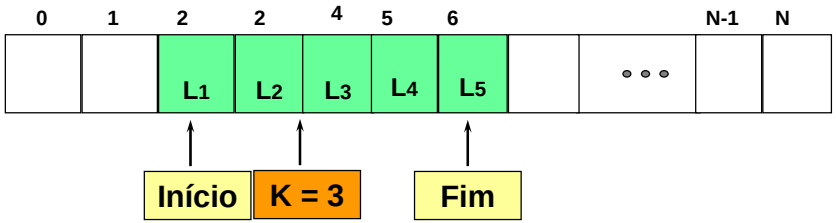
# Inserir novo nodo em LL - contigüidade física

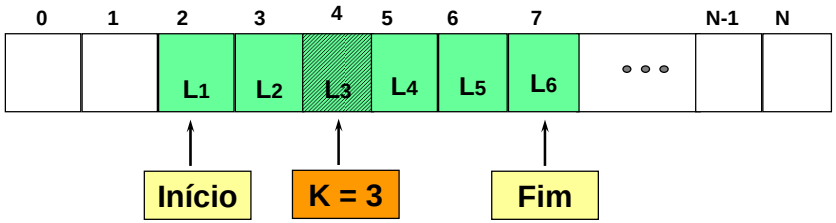



**Observar o preenchimento das estruturas...**



# Inserção de novo nodo na "k-ésima" posição

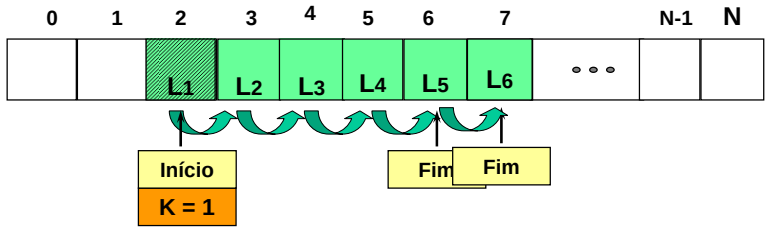


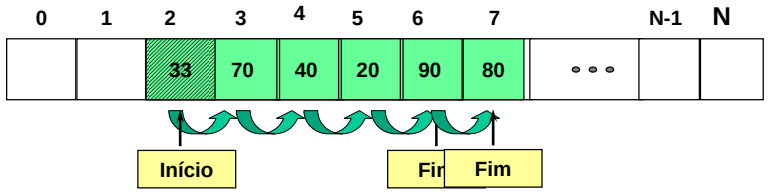



# Inserção de novo nodo na "k-ésima" posição

- **Inserir como primeiro nodo**

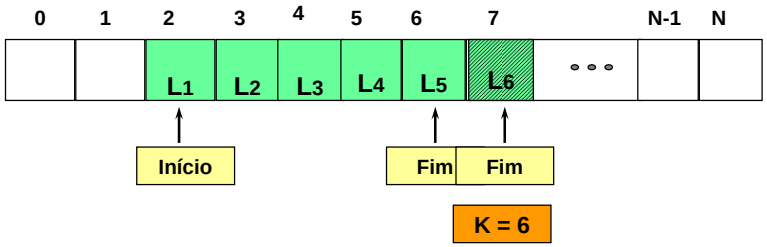



- **Ex:**



## Inserção de novo nodo na "k-ésima" posição

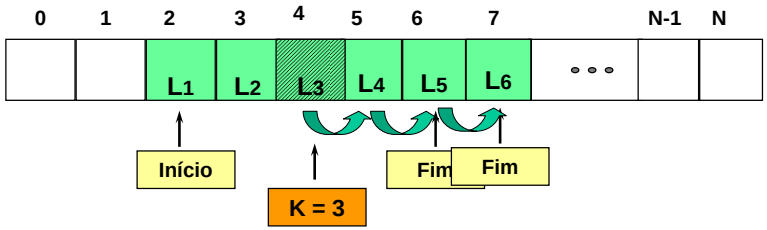
- **Inserir como último nodo**






## Inserção de novo nodo na "k-ésima" posição

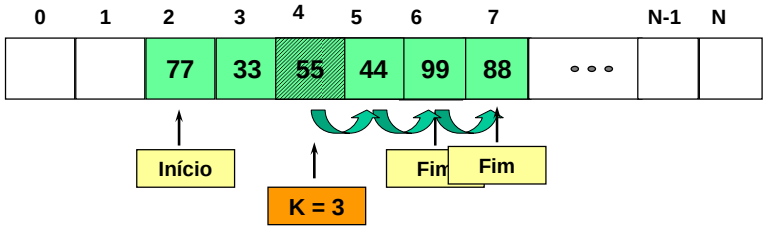
- **Inserir no meio da lista linear**





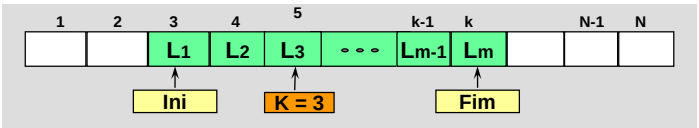

## Inserção de novo nodo na "k-ésima" posição

- **Ex: Inserir no meio da lista linear**






## Inserção de novo nodo na "k-ésima" posição




***void insere ( TProduto t[], int *inicio, int *fim, int posicao)***

**Analisar:**

- **lista vazia (*Ini* e *Fim* = -1), só se for K = 1 (primeiro da lista)**
- ***Ini* = 0 (início do arranjo) e *Fim* = MAX-1 (final do arranjo) - não tem mais espaço para inserir (insucesso)**
- **pode inserir como primeiro, no meio, ou logo após o último ( K )**
- **se *Fim* = MAX-1 e tem espaço antes, deslocar nodos para frente**

# Lista linear – TAD Genérico

- **Dados**

```
typedef struct T_Produto {
   int cod;
         char nome[40];
         float preco;
}TProduto;

int inicio, fim, maximo;

TProduto Lista[MAX];
```

- **Operações**
  - void inicializa (TProduto t[], int *inicio, int *fim);
  - void insere (TProduto t[], int *inicio, int *fim, int posicao);
  - **remove(???):** Remove o k-ésimo nodo da lista
  - int consulta ( TProduto t[], int inicio, int fim, int posicao)
  - **Destroi(???):** destrói a lista



```
void insere ( TProduto t[], int *inicio, int *fim, int posicao) {
  int i;
  if ( ((*inicio == 0) && (*fim == MAX-1)) || //não tem espaço
       (posicao > *fim - *inicio + 2 ) || //posição inválida
       (posicao < 1) || //posição inválida
       ((*inicio == -1) && (posicao != 1 )) ) { //lista vazia, só pode ser o primeiro
          printf("erro - posicao invalida\n");
          return ;
        }
  else if (*inicio ==-1) {
         *inicio = ???;
         *fim = ???;
        }
        else if (*fim != MAX-1) {
            for (i=*fim; i >= *inicio + posicao -1; i--)
                  t[???] = t[???];
            *fim = ???;
      }
         else {
        for (i=*inicio; i <= *inicio + posicao-1; i++)
                  t[???] = t[???];
            *inicio = ???;
      }
      /* Lendo os dados*/
      printf("Codigo: "); scanf("%d", &t[*inicio+posicao-1].cod);
      printf("Nome: "); scanf ("%s", t[*inicio+posicao-1].nome);
      printf("Preco: "); scanf ("%f", &t[*inicio+posicao-1].preco);
}
```

Insere o primeiro

Deslocamento para o fim

Desloca para o início



```
void insere ( TProduto t[], int *inicio, int *fim, int posicao) {
  int i;
  if ( ((*inicio == 0) && (*fim == MAX-1)) || //não tem espaço
       (posicao > *fim - *inicio + 2 ) || //posição inválida
       (posicao < 1) || //posição inválida
       ((*inicio == -1) && (posicao != 1 )) ) { //lista vazia, só pode ser o primeiro
          printf("erro - posicao invalida\n");
          return ;
        }
  else if (*inicio ==-1) {
         *inicio = 0;
         *fim = 0;
        }
        else if (*fim != MAX-1) {
            for (i=*fim; i >= *inicio + posicao -1; i--)
                  t[i+1] = t[i];
            *fim = *fim + 1;
      }
         else {
        for (i=*inicio; i <= *inicio + posicao-1; i++)
                  t[i-1] = t[i];
            *inicio = *inicio - 1;
      }
      /* Lendo os dados*/
      printf("Codigo: "); scanf("%d", &t[*inicio+posicao-1].cod);
      printf("Nome: "); scanf ("%s", t[*inicio+posicao-1].nome);
      printf("Preco: "); scanf ("%f", &t[*inicio+posicao-1].preco);
}
```

Insere o primeiro

Deslocamento para o fim
*(tem espaço no fim)*

Desloca para o início

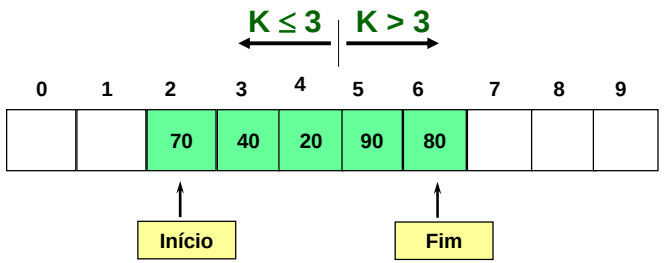


# Inserção de novo nodo na "k-ésima" posição

## Otimizar:

**deslocar nodos da metade para o fim se a inserção for na primeira metade da lista, e da metade para o início se for na segunda metade**

→ **desde que haja espaço no lado considerado**



# Lista linear - TAD Genérico

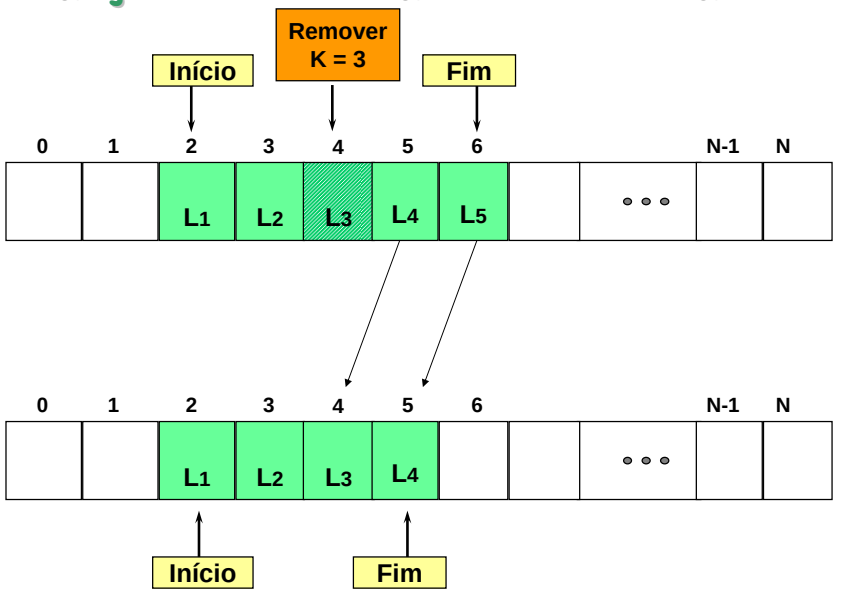
- **Dados**

```
typedef struct T_Produto {
   int cod;
         char nome[40];
         float preco;
}TProduto;

int inicio, fim, maximo;

TProduto Lista[MAX];
```

- **Operações**
  - void inicializa (TProduto t[], int *inicio, int *fim);
  - void insere (TProduto t[], int *inicio, int *fim, int posicao);
  - **remove(???):** Remove o k-ésimo nodo da lista
  - int consulta ( TProduto t[], int inicio, int fim, int posicao)
  - **Destroi(???):** destrói a lista



# Remoção do k-ésimo nodo de uma LL






# Lista linear - TAD Genérico

- **Dados**

```
typedef struct T_Produto {
   int cod;
         char nome[40];
         float preco;
}TProduto;

int inicio, fim, maximo;

TProduto Lista[MAX];
```

- **Operações**
  - void inicializa (TProduto t[], int *inicio, int *fim);
  - void insere (TProduto t[], int *inicio, int *fim, int posicao);
  - int remove (TProduto t[], int *inicio, int *fim, int posicao);
  - int consulta ( TProduto t[], int inicio, int fim, int posicao)
  - **Destroi(???):** destrói a lista



# Remoção do k-ésimo nodo de uma LL

```
int remove ( TProduto t[], int *inicio, int *fim, int posicao) {
  int i, rem=0;

  if ( (posicao > *fim - *inicio + 1 ) || (posicao < 1))
      return -1;
  else
  {
     rem =  ???;
     for (i=*inicio+posicao-1; i <*fim; i++)
        t[???] = t[???];

     strcpy(t[*fim].nome,"");
     t[*fim].cod=0;
     t[*fim].preco=0;
     *fim = *fim -1;
     return rem;
   }
}
```

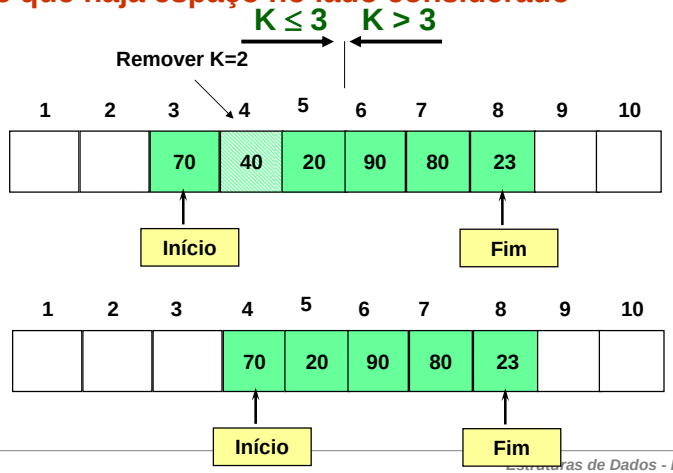
# Remoção do k-ésimo nodo de uma LL

```
int remove ( TProduto t[], int *inicio, int *fim, int posicao) {
  int i, rem=0;

  if ( (posicao > *fim - *inicio + 1 ) || (posicao < 1))
      return -1;
  else
  {
     rem =  t[*inicio+posicao-1].cod;
     for (i=*inicio+posicao-1; i <*fim; i++)
        t[i] = t[i+1];

     strcpy(t[*fim].nome,"");
     t[*fim].cod=0;
     t[*fim].preco=0;
     *fim = *fim -1;
     return rem;
   }
}
```

# Otimização da remoção do k-ésimo nodo

**Deslocar nodos da metade para baixo se o nodo a ser removido estiver na primeira metade da lista, e da metade para cima se estiver na segunda metade**

→ **desde que haja espaço no lado considerado**






# Lista linear – TAD Genérico

- **Dados**

```
typedef struct T_Produto {
   int cod;
         char nome[40];
         float preco;
}TProduto;

int inicio, fim, maximo;

TProduto Lista[MAX];
```

- **Operações**
  - void inicializa (TProduto t[], int *inicio, int *fim);
  - void insere (TProduto t[], int *inicio, int *fim, int posicao);
  - void remove (TProduto t[], int *inicio, int *fim, int posicao);
  - int consulta ( TProduto t[], int inicio, int fim, int posicao)
  - void destroi (TProduto t[], int *inicio, int *fim);



# Destrói

```
void destroi ( TProduto t[], int *inicio, int *fim) {
    int i;

    *inicio = -1;
    *fim = -1;
}
```