

Estruturas de Dados

José Ricardo Mello Viana

Aula 04
Listas Lineares

2010.2

- 1 Introdução
- 2 Conceito
- 3 Operações
- 4 Representações
 - Contiguidade
- 5 Implementação das operações
 - Acessar o k -ésimo elemento
 - Alterar o k -ésimo elemento

- Conjuntos de dados frequentemente se relacionam entre si
- Ex: leitura de índice pluviométrico
 - Função que expresse precipitação x tempo
 - Relação de ordem entre os elementos

- *Lista linear*: Estrutura de dados que preserva a relação de ordem total entre os elementos
- Composta de *nós*: podem conter dados primitivos ou compostos

- Conjunto de n nós, $n \geq 0$
 - x_1, x_2, \dots, x_n
- Organizados de forma a refletir a posição. Se $n > 0$:
 - x_1 é o primeiro
 - para $1 < k < n$, x_k é precedido de x_{k-1} e sucedido de x_{k+1}
 - x_n é o último elemento
- Se $n = 0$, diz-se que a lista é *vazia*

- Acessar o k -ésimo nó
- Inserir um nó após (ou antes) do k -ésimo elemento
- Remover o k -ésimo nó
- Concatenar duas listas
- Determinar o número de nós
- Localizar o nó que contém um dado valor

- Várias formas
- Escolha dependerá das operações usadas com mais frequência
 - Contiguidade física
 - Encadeamento de nós

- Sequencialidade da memória
- Armazenamento em endereços contíguos
- Se endereço de x_i é conhecido, o de x_{i+1} pode ser determinado

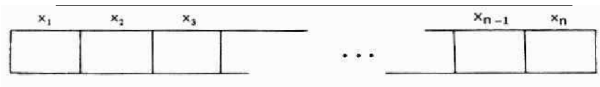


Figure: Esquema da representação por contiguidade física

- Mesma estrutura do vetor
- Lista linear é definida como:
 - *tipo* lista::vet[1..n] de tipo'
 - *tipo*' é o tipo de dado representado pelo nó
- Declaração da lista:
var x:lista
- Lista pode aumentar de tamanho
 - Usaremos um vetor de tamanho $m > n$

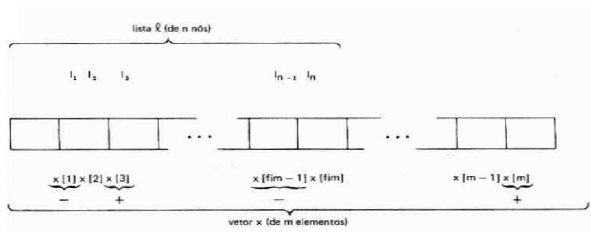


Figure: Esquema da representação por contiguidade física detalhado

- Lista 1 contida no vetor x
- $l_1 = x[1], l_2 = x[2], \dots, l_n = x[fim]$

- Construção única: $x[k]$
- Pode ocorrer: $k > fim$ ou $k \leq 0$
 - Tentativa de acessar nó não existente
 - Procedimento testará a validade de k
 - Parâmetro indicará validade ou não
 - Se *senal* = *falso* então $k > fim$ ou $k \leq 0$
 - Se *senal* = *verdadeiro* então $0 < k \leq fim$
 - Mesma convenção pode ser usada para as outras operações

```
proc acessar(x:lista; k, fim:int; sinal:bool; val:tipo')
  se  $k \leq 0$  ||  $k > fim$  então
    sinal  $\leftarrow$  false
  senão
    início
      val  $\leftarrow$  x[k]
      sinal  $\leftarrow$  true
    fim
```

```
proc alterar(x:lista; k, fim:int; sinal:bool; val:tipo')  
  se  $k \leq 0$  ||  $k > fim$  então  
    sinal  $\leftarrow$  false  
  senão  
    início  
       $x[k] \leftarrow val$   
      sinal  $\leftarrow$  true  
    fim
```