Estrutura de Dados

Rafael D. Ribeiro, M.Sc.
rafaeldiasribeiro@gmail.com
http://www.rafaeldiasribeiro.com.br

Estrutura de Dados
Recursividade

Exemplo:

- 5 ! = 5 x 4 !
- 4 ! = 4 x 3 !
- 3 ! = 3 X 2 !
- 2 ! = 2 x 1 !
- 1 ! = 1 x 0 !
- Fatorial de 0 = 1

Estrutura de Dados
Recursividade

algoritmo 1.3: fatorial (não recursivo)

$fat[0] := 1$

para $j := 1, \ldots, n$ faça

$fat[j] := j \times fat[j-1]$

Estrutura de Dados
Recursividade

algoritmo 1.2: fatorial (recursivo)

função $fatorial(i)$

$fat(i) :=$ se $i \leq 1$ então 1 senão $i \times fat(i-1)$

Estrutura de Dados

"As estruturas de dados estudam as relações lógicas existentes entre os dados (ex: relação linear, relação hierárquica ..) e serão manipuladas através de operações sobre os dados (ex: consultar uma informação em um conjunto de fichas de alunos, remover um assinante de uma lista de assinantes, apagar um diretório em uma árvore de diretórios e subdiretórios...).

A escolha da estrutura de dados a ser utilizada refletirá diretamente na construção de uma solução mais ou menos eficiente para o problema."

Estrutura de Dados

Definição de lista linear : Uma **lista linear (ou tabela)** pode ser definida como uma sequência de n >= 0 elementos, denominados **nós**, L_0, L_1, ..., L_{N-1}, tais que as propriedades estruturais decorrem, inicialmente, das posições relativas dos elementos da lista.

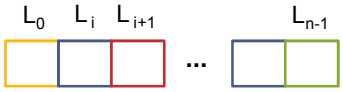
Temos : Se n >= 0 :

L_0 é o primeiro elemento

L_{n-1} é o último elemento

L_{i+1} é o sucessor de L_i e

L_i é o antecessor de L_{i+1}.



Estrutura de Dados

Características de uma lista linear:

- possui 0 ou mais nós;
- todos os elementos podem ser acessados
- podemos incluir ou excluir qualquer elemento
- agrupa informações referentes a um conjunto de elementos que relacionam-se entre si .

EXEMPLOS:

- Pessoas esperando ônibus
- Letras de uma palavra
- Palavras de uma frase

Estrutura de Dados

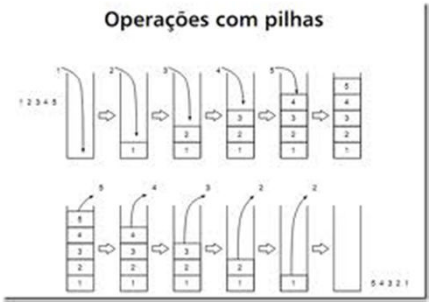
Operações que podem ser realizadas sobre listas:

- Acessar um elemento qualquer da lista
- Inserir um elemento em uma posição específica
- Remover um elemento de uma posição específica
- Combinar duas listas
- Dividir uma lista
- Determinar o total de elementos em uma lista
- Procurar determinado elemento na lista
- Apagar lista
- Outras...

Estrutura de Dados

Operações de acesso, inserção e remoção, restrita a estremos da lista. Casos especiais:

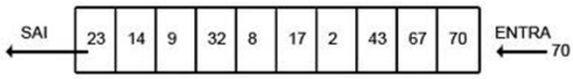
- Pilha:
  - Lista linear onde todas as inserções, remoções e acessos são realizados em um único extremo.
  - Lista LIFO (Last In / First Out)



Estrutura de Dados

Operações de acesso, inserção e remoção, restrita a estremos da lista. Casos especiais:

- Fila:
  - Lista linear onde todas as inserções são feitas em um certo extremo e todas as remoções e acessos são feitas no outro extremo.
  - Lista FIFO (First In / First Out)

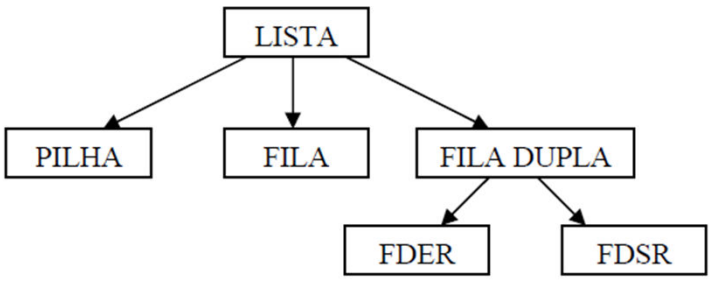


Estrutura de Dados

Operações de acesso, inserção e remoção, restrita a estremos da lista. Casos especiais:

- Fila Dupla:
  - Lista linear onde todas as inserções, remoções e acessos são feitas em qualquer extremo
  - Lista DEQUE (Double – Ennded Queue)
  - Podem ser:
    - Fila dupla de entrada restrita (inserções restritas a um único estremo)
    - Fila dupla de saída restrita (remoções restritas a um único estremo)

Estrutura de Dados



Estrutura de Dados

Alocação de Memória

- A alocação de memória do ponto de vista do programador se divide em 4 categorias:

	Sequencial	Encadeada
Estática	Estática Sequencial	Estática Encadeada
Dinâmica	Dinâmica Sequencial	Dinâmica Encadeada

Estrutura de Dados

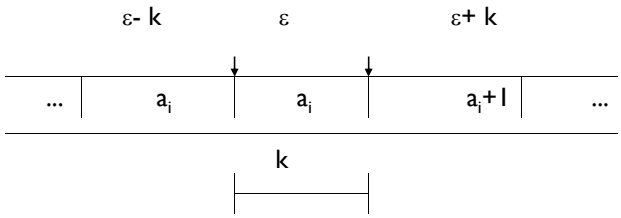
Alocação Estática x Dinâmica

- Quantidade total de memória utilizada pelos dados é previamente conhecida e definida de modo imutável, no próprio código-fonte do programa, não variando durante toda a execução do programa.
- Capacidade de criar novas variáveis enquanto executa, isto é, se áreas de memória que não foram declaradas no programa passam a existir durante a sua execução.

Estrutura de Dados

Alocação Sequencial

- Consiste em colocar os elementos alocáveis em células de memória consecutivas, um após o outro.



Assim, se cada célula tem um endereço único ε e utiliza *k* bytes, temos :

- Endereço(a_i) = ε
- Endereço (a_i-1) = ε-k
- Endereço(a_i+1) = ε+k

Estrutura de Dados

Alocação Sequencial: Vantagens e desvantagens

- Dado o endereço inicial da área alocada e o índice *i* de um elemento qualquer da lista, pode-se acessá-lo imediatamente, com um simples e rápido cálculo.

- Precisa-se movimentar os elementos, de modo a abrir espaço para inserção, ou de modo a ocupar o espaço liberado por um elemento que foi removido.

Estrutura de Dados

Alocação Encadeada

- Os elementos podem ocupar quaisquer células de memória (não necessariamente consecutivas) e, para manter a relação de ordem linear, juntamente com cada elemento é armazenado o endereço do próximo elemento da lista.

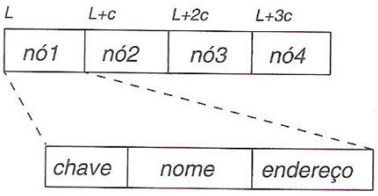
- Os elementos são armazenados em blocos de memória denominados **nodos**, sendo que cada nodo é composto por dois campos: um para armazenar dados e outro para armazenar endereço.

Estrutura de Dados

Alocação Encadeada

Endereço	Conteúdo		
L=3FFA	a_1	1C34	←Primeiro elemento, acessível a partir de L
1C34	a_2	BD2F	←Note que o segundo não ocupa o endereço consecutivo à a_1
BD2F	a_3	AC12	
	...		
1000	a_{n-2}	3A7B	← Cada nodo armazena um elemento e um endereço do próximo elemento da lista
	...		
14F6	a_{n-1}	5D4A	
5D4A	a_n	null	← Último elemento da cadeia, o endereço nulo indica que o elemento não tem um sucessor.

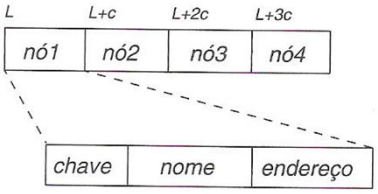
Estrutura de Dados
Listas Lineares em Alocação Sequencial



Lista Não Ordenada

```
algoritmo 2.1: Busca de um elemento na lista ℒ
função busca1(x)
    i := 1;   busca1 := 0
    enquanto i ≤ n faça
        se ℒ[i].chave = x então
            busca1 := i                    % chave encontrada
            i := n + 1
        senão i := i + 1                   % pesquisa prossegue
```

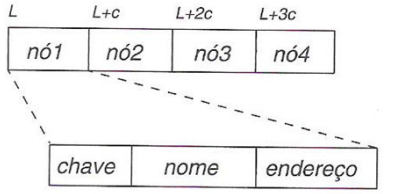
Estrutura de Dados
Listas Lineares em Alocação Sequencial



Lista Não Ordenada

```
algoritmo 2.2: Busca de um elemento na lista ℒ
função busca(x)
    ℒ[n + 1].chave := x;   i := 1
    enquanto ℒ[i].chave ≠ x faça
        i := i + 1
    se i ≠ n + 1 então
        busca := i                         % elemento encontrado
    senão busca := 0                       % elemento não encontrado
```

Estrutura de Dados
Listas Lineares em Alocação Sequencial



Lista Ordenada

algoritmo 2.3: Busca de um elemento na lista $\mathcal{L}$, ordenada

```
função busca-ord(x)
    L[n + 1].chave := x;    i := 1
    enquanto L[i].chave < x faça
            i := i + 1
    se i = n + 1 ou L[i].chave ≠ x então
            busca-ord := 0
    senão busca-ord := i
```

Estrutura de Dados
Listas Lineares em Alocação Sequencial

Lista Ordenada – **Busca Binária**

algoritmo 2.4: Busca binária

```
função busca-bin(x)
    inf := 1;    sup := n;    busca-bin := 0
    enquanto inf ≤ sup faça
            meio := ⌊(inf + sup)/2⌋          % índice a ser buscado
            se L[meio].chave = x então
                    busca-bin := meio        % elemento encontrado
                    inf := sup + 1
            senão se L[meio].chave < x então
                            inf := meio + 1
                    senão sup := meio − 1
```

Estrutura de Dados
Listas Lineares em Alocação Sequencial

Inserção

algoritmo 2.5: Inserção de um nó na lista $\mathcal{L}$

```
se n < M então
        se busca(x) = 0 então
                L[n + 1] := novo-valor
                n := n + 1
        senão "elemento já existe na tabela"
senão overflow
```

Estrutura de Dados
Listas Lineares em Alocação Sequencial

Remoção

algoritmo 2.6: Remoção de um nó da lista $\mathcal{L}$

```
se n ≠ 0 então
        indice := busca(x)
        se indice ≠ 0 então
                valor-recuperado := L[indice]
                para i := indice, n − 1 faça
                        L[i] := L[i + 1]
                n := n − 1
        senão "elemento não se encontra na tabela"
senão underflow
```

Estrutura de Dados

Bibliografia das Notas de Aula

Estruturas de Dados e Seus Algoritmos - Jayme Luiz Szwarcfiter, Lilian Markenzon LTC Editora - 2 ° Ed.