ED – Notas de Aula 01

Capítulo 1 – Introdução

Ao concluir este capítulo, você deverá ser capaz de:

- Conhecer as formas mais básicas de organizar os dados na memória de um computador, de forma que essa organização reflita bem o problema que está sendo tratado e torne mais eficientes os algoritmos que manipulem esses dados.

Introdução

Na resolução de um problema através de um programa, a primeira providência é conceber um **algoritmo** adequado.

[Um algoritmo é um processo sistemático para a resolução de um problema, consistindo de uma seqüência de passos, ordenada e sem ambigüidades.]

A eficiência de um algoritmo está intimamente relacionada com a disposição, na memória, dos dados que são tratados pelo programa. Por exemplo, se freqüentemente enfrentamos o problema de descobrir o número do telefone de nossos conhecidos, é conveniente dispor de uma relação de números organizada em uma agenda. Se a organização for feita pela ordem alfabética, a agenda de fato ajuda. Se, porém, organizássemos nossa agenda pela ordem de altura das pessoas, com raras exceções, a agenda se tornaria difícil de manusear.

As estruturas de dados são formas de distribuir e relacionar os dados disponíveis, de modo a tornar mais eficientes os algoritmos que manipulam esses dados.

Vejamos alguns exemplos práticos

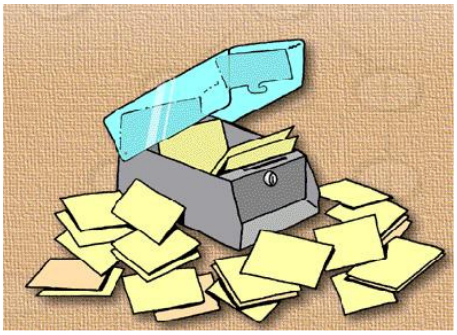
Situação 1

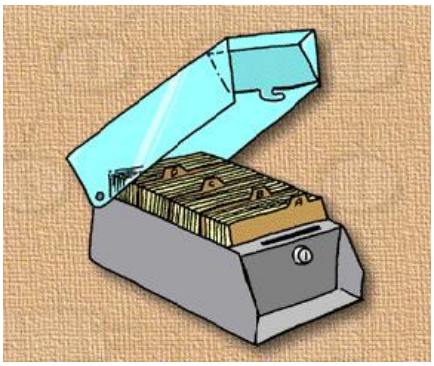
problema: Manipular um conjunto de fichas em um fichário.

solução: Organizar as fichas em ordem alfabética.

operações possíveis: Inserir ou retirar uma ficha, procurar uma ficha, etc.

estrutura de dados: LISTA – seqüência de elementos dispostos em alguma ordem.





Situação 2

problema: Organizar as pessoas que querem ser atendidas em um guichê.

solução: Colocar as pessoas em fila.

operações possíveis: Atender uma pessoa no guichê, entrar uma pessoa no final da fila. Não é permitido "furar" a fila, ou seja, entrar uma pessoa entre outras que já estão presentes.

estrutura de dados: FILA – seqüência de elementos dispostos em ordem, mas com uma regra para a entrada e saída dos elementos: o primeiro elemento que chega também é o primeiro que sai da estrutura.

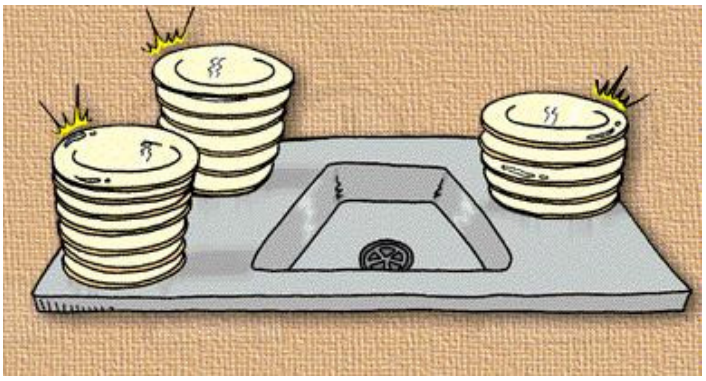
Situação 3

problema: Organizar um conjunto de pratos que estão sendo lavados, um a um, em um restaurante.

solução: Colocar os pratos empilhados.

operações possíveis: Colocar um prato limpo no alto da pilha, retirar um prato do alto da pilha. Não é permitido tirar um prato qualquer do meio da pilha.

estrutura de dados: PILHA - seqüência de elementos dispostos em ordem, mas com uma regra para a entrada e saída dos elementos: o último que chega é o primeiro a sair da estrutura.



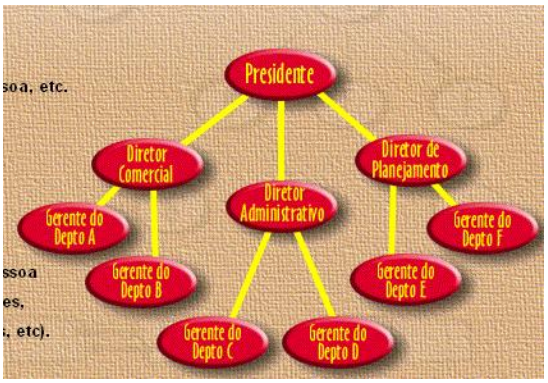
Situação 4

problema: Conseguir um modo de visualizar o conjunto de pessoas que trabalham em uma empresa, tendo em conta a função de cada pessoa.

solução: Construir um organograma da empresa.

operações possíveis: Inserir ou retirar certas funções, localizar uma pessoa, etc.

estrutura de dados: ÁRVORE – estrutura de dados que caracteriza uma relação de hierarquia entre os elementos: uma pessoa não pode pertencer a dois departamentos diferentes, cada diretoria tem os seus próprios departamentos, etc.



Situação 5

problema: Estabelecer um trajeto para percorrer todas as capitais de um país.

solução: Utilizar um mapa que indique as rodovias existentes, e estabelecer uma ordem possível para percorrer todas as cidades

operações possíveis: Encontrar um modo de percorrer todas as cidades, determinar o caminho mais curto para ir de uma cidade a outra, etc.

estrutura de dados: GRAFO – estrutura que organiza vários elementos, estabelecendo relações entre eles, dois a dois

Os exemplos vistos correspondem exatamente aos tipos básicos de estruturas de dados utilizadas em computação: listas, filas, pilhas, árvores e grafos.

As estruturas de dados são abstratas, isto é, devem ser imaginadas como uma forma de organizar e relacionar dados na memória do sistema, de modo a permitir certas operações sobre esses dados, independentemente do modo como essas estruturas são implementadas. Elas podem ser implementadas de diversos modos, dependendo da capacidade da máquina e dos recursos oferecidos pela linguagem que se utiliza.

Neste curso, vamos implementá-las sempre utilizando ponteiros, de modo que o gerenciamento da memória seja dinâmico. Isto significa que as estruturas podem ser criadas, ampliadas, reduzidas ou removidas da memória em tempo de execução. Desta forma otimiza-se o uso da memória disponível, em função do conjunto de dados que o programa está processando.