CONCEITOS DE ESTRUTURAS DE DADOS

# Conceitos Gerais:

Estrutura de dados é uma forma de organizar os dados na memória do computador deforma que as informações sejam disponibilizadas mais rapidamente ao processador.

Existem duas preocupações essenciais quando for definir uma estrutura de dados, sendo elas:

* Como os dados são armazenados
* Quais operações serão realizadas nos dados armazenados

As estruturas de dados são independentes de sua implementação. (ADT)

As soluções de problemas são realizadas através de estruturas de dados e algoritmos, que devem ser eficientes tanto em tempo de execução quando em uso de memória, dando prioridade no tempo gasto pelo algoritmo.

As estruturas de dados podem ser divididas em **Lineares** (Arrays, listas encadeadas, pilhas e filas) e **hierárquicas** (árvores e grafos). Cada estrutura se adequa especialmente a tipos de problemas específicos.

# Mais importantes estruturas de dados:

são a estrutura de dados mais comum usada para armazenar dados. Se você deseja inserir um item em uma posição específica que já está ocupada por algum elemento, você deve deslocar todos os itens uma posição para a direita da posição que o novo elemento deve ser inserido, e então, inserir o novo item. O tempo gasto pela operação de inserção depende do tamanho do array e em qual posição o item está sendo inserido. O mesmo acontece com a exclusão de um item.

**Listas** recebem memória em tempo de execução, logo, não há desperdício de memória. A lista vinculada, em termos de desempenho, é mais lenta que a matriz porque não há acesso direto aos elementos da lista vinculada. Já a lista encadeada provou ser uma estrutura de dados útil quando o número de elementos a serem armazenados não é conhecido antecipadamente.

**Pilhas** são estruturas de dados de estratégia do tipo last-in-first-out, isto significa que o elemento armazenado por último será removido primeiro.

A **fila** é uma estrutura de dados do tipo first-in-first-out. O elemento que é adicionado à estrutura de dados da fila primeiro será removido da fila primeiro.

**Arvore** é uma estrutura de dados hierárquica, em que o elemento mais alto é chamado de raiz da árvore. Exceto o elemento raiz, cada elemento em uma árvore tem um elemento pai e zero ou mais elementos filhos. Todos os elementos da subárvore esquerda vêm antes da raiz na ordem de classificação e todos os da subárvore direita vêm depois da raiz.

**Texto branco sobre fundo preto

Descrição gerada automaticamente com confiança média**

**Dicionário** é uma estrutura de dados que mantém um conjunto de itens indexados com base em chaves. O dicionário armazena dados na forma de pares de elementos-chave.

**Grafos** é uma estrutura de dados em rede que conecta uma coleção de nós chamados vértices, por conexões, chamadas arestas. Uma aresta pode ser vista como um caminho ou link de comunicação entre dois nós e podem ser direcionadas ou não direcionadas. Se um caminho é direcionado, você pode se mover apenas em uma direção, enquanto em um caminho não direcionado, o movimento é possível em ambas as direções.**Forma

Descrição gerada automaticamente**

# Encapsulamento da representação:

Encapsulamento = A inclusão de uma coisa dentro de outra para que a incluída não seja aparente

Desencapsulamento = A remoção ou tornar aparente uma coisa previamente encapsulada.

A ideia de encapsulamento vem da necessidade de distinguir claramente entre a especificação e a implementação de uma operação;

Existem duas versões de encapsulamento:

* Visão linguagem de programação;
* Visão Banco de dados

A ideia de encapsulamento em linguagens de programação vem de tipos de dados abstratos. Nela, um objeto tem uma parte de interface e uma parte de implementação. A parte da interface é a especificação do conjunto de operações que podem ser realizadas no objeto. parte de implementação tem uma parte de dados e uma parte de procedimento.

O encapsulamento é um dos fundamentos da OOP (Programação Orientada a Objetos).

Métodos publicamente acessíveis geralmente são fornecidos na classe (chamados getters e setters) para acessar os valores. Outras classes de cliente chamam esses métodos para recuperar e modificar os valores dentro do objeto.

Tido como o princípio mais importante da orientação a objetos, o encapsulamento é a ideia de que os dados dentro do objeto só devem ser acessados por meio de uma interface pública, ou seja, os métodos do objeto.

# Atualização, pesquisa e ordenação:

Antes de tomar decisões baseadas em dados, é preciso que esses dados sejam armazenados em algum lugar e sejam acessíveis. Os dados precisam ser atualizados quando necessário ou descartados quando não forem mais necessários.

Na prática, os dados são inseridos, atualizados e excluídos das tabelas usando uma ampla variedade de métodos. Eles variam de operações manuais a aplicativos externos e tudo mais. Independentemente do método, nos bastidores, tudo se resume a consultas SQL, sendo executadas por uma pessoa ou automaticamente em resposta a um evento.]

Dentro da programação de computadores, a sigla CRUD significa criar (create), ler (read), atualizar (update) e excluir (delete), que são as quatro funções básicas do armazenamento persistente.

No CRUD, os procedimentos **create** executam a instrução **insert** para criar um registro. Os procedimentos **read** leem os registros da tabela com base na nota-chave primária dentro do parâmetro de entrada. Já os procedimentos update executam uma instrução update na tabela com base na chave primária especificada para um registro dentro da cláusula **where** da instrução. E os procedimentos delete excluem uma linha especificada na cláusula **where**.

O processo de ordenar os registros em um banco de dados é chamado de Sorting. A ordenação e a busca em conjunto constituem uma importante área de estudo em métodos computacionais, em que ambos são campos de estudo muito importantes em estrutura de dados e algoritmos.

A pesquisa é o processo de encontrar um determinado item em uma coleção de itens e, normalmente, responde se o item está presente na coleção ou não. A pesquisa requer um campo-chave como nome, ID ou código relacionado ao item de destino.

Os algoritmos de busca mais comuns são:

* Pesquisa linear;
* Pesquisa binária;
* Pesquisa de interpolação;
* Tabela de hash.

Na **Busca Linear** feita uma busca sequencial em todos os itens para buscar o item alvo. Cada item é verificado em sequência até que a correspondência seja encontrada.

Na **Busca binária** temos que ordenar a coleção de dados em ordem crescente primeiro e depois procurar o item de destino comparando o item mais central da coleção.

**Interpolação** é uma melhoria da pesquisa binária, funcionando em listas de dados classificadas e distribuídas igualmente.

A **Tabela hash** é uma estrutura de dados que armazena dados em formato de matriz. Aqui, cada dado tem seu próprio índice exclusivo. Se você conhece o valor do índice dos dados necessários, a pesquisa é muito fácil e rápida. A tabela de hash usa um array como meio de armazenamento e usa a técnica de hash para gerar um índice de onde um elemento deve ser inserido ou localizado.

O **SORTING** é o processo de colocar elementos de uma coleção em algum tipo de ordem. Por exemplo, uma lista de palavras pode ser classificada em ordem alfabética ou por comprimento. A classificação eficiente é importante para otimizar o uso de outros algoritmos que exigem que as listas classificadas funcionem corretamente. Os algoritmos de sorting mais comuns são:

* Tipo bolha (bubble sort);
* Ordenação por inserção (insertion sort);
* Ordenação de seleção (selection sort);
* Ordenação rápida (quick sort);
* Classificação mesclada (merge sort);
* Classificação shell (shell sort).

**Tipo Bolha** - Baseia-se na comparação em que cada par adjacente de elemento é comparado e trocado se não estiverem em ordem. Ele funciona percorrendo repetidamente a lista a ser classificada, comparando dois itens por vez e trocando-os se estiverem na ordem errada.

**Seleção** - Selecionamos repetidamente o menor elemento restante e o movemos para o final de uma lista ordenada crescente. A ordenação por seleção é conhecida por sua simplicidade e tem vantagens de desempenho sobre algoritmos mais complicados em determinadas situações.

**Inserção** - Sub lista ordenada é mantida inserindo um elemento de cada vez, que deve encontrar sua localização apropriada e, em seguida, deve ser inserido nela.

**Classificação rápida** - A classificação rápida particiona uma matriz e, em seguida, chama a si mesma recursivamente duas vezes para classificar as duas submatrizes resultantes. Esse algoritmo é bastante eficiente para grandes conjuntos de dados.

**Mesclada** - Primeiro dividimos a matriz em metades e depois as combinamos de maneira ordenada. Esse algoritmo de ordenação sofre de complexidade de espaço, mas é um algoritmo estável porque preserva a ordem de entrada de elementos iguais na saída ordenada.

**Shell** - Ela melhora a classificação por inserção comparando elementos separados por um intervalo de várias posições.

# Estruturas Lineares:

Uma estrutura de dados linear é aquela em que os itens de dados são ordenados sequencialmente. A estrutura de dados linear tem as seguintes características:

* Armazena e gerencia dados em uma ordem linear;
* Os elementos de dados da sequência são vinculados uns aos outros em ordem sequencial;
* A implementação da estrutura linear na memória de um computador é simples, pois os dados são organizados sequencialmente;
* Como os elementos de dados são armazenados em um único nível, é possível percorrê-los em uma única execução;
* Se uma estrutura de armazenamento de dados linear for implementada, a memória do computador será subutilizada;
* A complexidade de tempo de uma estrutura de dados aumenta com o aumento do tamanho da estrutura de dados.
* Arrays, listas, pilhas e filas são os diferentes tipos de estruturas de dados lineares.

As operações de inserção e exclusão dentro da pilha e da fila levam tempo. Em estruturas de dados lineares, a memória não é utilizada de forma eficiente em comparação com as estruturas de dados não lineares, o que é considerado uma desvantagem para as estruturas lineares.