Aprenda o que são Estrutura de Dados e Algoritmos

# Conceitos iniciais sobre estruturas dedados, arrays e registro

## Introdução e objetivos

* O que é estrutura de Dados
* Vetores e Matrizes
* Registro

## O que é estrutura de dados?

Estrutura de Dados é uma estrutura organizada de dados na memória de um computador ou em qualquer dispositivo de armazenamento, de forma que os dados possam ser utilizados de forma correta.

Vamos imaginar que nós precisamos organizar os dados na memória, nossa memória é volátil e vamos imaginar que nós precisamos organizar esses dados na memória, como que nós organizamos esses dados? Através de estruturas. Então vamos imaginar como se fosse uma cidade, em uma cidade nós temos em organização, corredores, ruas, prédios, casas, meios de transportes, então isso são estruturas de organização de uma cidade, e para organizar os dados, para que esses dados possam ser manipulados e utilizados de forma correta nós temos as estruturas de dados, que podem ser organizadas tanto na memória de um computador ou em qualquer dispositivo de armazenamento como um pen drive, HD externo ou nuvem.

Essas estruturas encontram muitas aplicações no desenvolvimento de sistemas, sendo que algumas são altamente especializadas e utilizadas em tarefas específicas. Por exemplo estrutura de busca, que utiliza uma estrutura de busca especializada em busca recursiva por exemplo.

Usando as estruturas adequadas através de algoritmos, podemos trabalhar com uma grande quantidade de dados ou pequena quantidade de dados também, como aplicações em bancos de dados ou serviços de busca.

Como que a gente aplica essas estruturas a memória, como criamos uma situação para que essas estruturas sejam efetivas, criamos essas situações através de um algoritmo. Então vamos lá instruir.

### Algoritmo

Um Algoritmo é um conjunto de instruções estruturadas e ordenadas, seu objetivo é realizar uma tarefa ou operação específica.

Os algoritmos são utilizados para manipular dados nas estruturas de várias formas, como por exemplo: inserir, excluir, procurar e ordenar dados.

Imagine que nós temos cinco gavetas, e essas cinco gavetas estão dentro de um armário e nós estamos abrindo essas gavetas para ver se a gente acha alguma coisa, se a gente localiza alguma coisa nesse armário, nessas gavetas. É um algoritmo, por exemplo, de varrer um vetor.

Em uma estrutura de dados devemos saber como realizar um determinado conjunto de operações básicas, como por exemplo:

* Inserir Dados;
* Excluir Dados;
* Localizar um elemento;
* Percorrer todos os itens constituintes da estrutura para visualização;
* Classificar, que se resume em colocar os itens de dados em uma determinada ordem (numérica, alfabética etc.)

Imagine que nós temos um armário, com várias gavetas, esse armário vamos supor que ele tem 5 gavetas, nós podemos inserir arquivos nesse armário, tirar arquivos, localizar um arquivo, como nós podemos também percorrer todo o armário atrás de algum item para nós visualizarmos. E nós também podemos classificar ele, colocar os arquivos em ordem numérica, alfabética e assim sucessivamente. Isso é um exemplo de Vetor.

Porém, em estrutura de dados nós temos diversas estruturas a ser combinas, estudadas, utilizadas.

### Principais Estruturas de Dados

As principais estruturas de dados utilizadas em algoritmos são:

* Vetores e Matrizes
* Registro
* Lista
* Pilha
* Fila
* Árvore
* Tabela Hash
* Grafos

## Vetores e matrizes

Vetores e Matrizes ou Arrays são estruturas de dados simples que podem auxiliar quando há muitas variáveis do mesmo tipo em um algoritmo.

Vamos imaginar que nós temos várias variáveis do tipo inteiro que nós vamos utilizar, nós não precisamos ficar declarando variável por variável, nós em certas situações podemos utilizar de uma variável só fixa, como por exemplo, cinco valores inteiros, que nós podemos colocar dentro de uma variável só. Então ao invés de nós colocarmos cada variável um valor, nós vamos criar uma variável com cinco valores do tipo inteiro, ou por exemplo, você quer criar um vetor de nomes, o vetor de nomes é fixo, então por exemplo, você quer utilizar cinco nomes, você não precisa criar uma variável para cada nome, você vai criar uma variável chamada nomes e colocar dentro dessa variável os nomes.

Para que nós possamos exemplificar algumas dessas estruturas como vetores, matrizes e as demais, é utilizado o <https://portugol-webstudio.cubos.io/ide>.

### Sobre Vetor

Vetor ou array unidimensional é uma variável que armazena várias variáveis do mesmo tipo.

O vetor é uma estrutura de dados indexada, ou seja, ele possuí índices do zero até aonde você disser, que pode armazenar uma determinada quantidade de valores do mesmo tipo.

Então se você coloca um vetor de cinco posições, ele vai ser a primeira posição zero e a última a quatro, tendo assim as cinco posições.

programa {

funcao inicio() {

inteiro numeros[] = {39, 45, 54, 55, 80}

escreva(numeros[0])

}

}

Outra aplicação que nós podemos realizar com vetores é imprimirmos na tela todos os elementos que o vetor possuí, para isso nós temos que percorrer o vetor, todos os índices dele para que nós possamos exibir todos os elementos dele.

programa {

funcao inicio() {

inteiro numeros[] = {39, 45, 54, 55, 80}

para (inteiro posicao = 0; posicao <=4; posicao++){

escreva(numeros[posicao], " ")

}

}

}

Outra aplicação que podemos realizar é a de procurar um número que tenha dentro do vetor.

programa {

funcao inicio() {

inteiro vetor[] = {1, 3, 5, 7, 9}

inteiro numero

logico achou = falso

escreva ("Qual número deseja procurar?")

leia(numero)

para (inteiro posicao = 0; posicao < 5; posicao++)

{

se (vetor[posicao] == numero)

{

escreva ("Encontrado número na posição: ", posicao, "\n")

achou = verdadeiro

}

}

se (nao achou)

{

escreva ("O número não está no vetor\n")

}

}

}

Temos também o cruzamento de dados entre vetores.

programa {

funcao inicio() {

cadeia nome[] = {"André", "Thiago", "Bruno", "Carlos", "Cassio"}

real altura[] = {1.71, 1.78, 1.75, 1.87, 1.71}

escreva ("----------------\n")

escreva (" TABELA \n")

escreva ("-----------------\n")

para (inteiro posicao = 0; posicao < 5; posicao++)

{

escreva (nome[posicao], "\t\t", altura[posicao], "\n")

}

}

}

### Sobre Matriz

Matriz ou array multidimensional é um vetor de vetores.

Uma matriz é um vetor que possui duas ou mais dimensões.

Foto preta e branca de relógio ao fundo

Descrição gerada automaticamente

Na imagem acima temos uma matriz com quatro colunas e três linhas que chamamos matriz quatro por três (4x3). Onde é um vetor de quatro posições e outro vetor de três posições.

programa {

inclua biblioteca Util --> u

funcao inicio() {

const inteiro TAMANHO = 5

inteiro matriz[TAMANHO][TAMANHO]

para (inteiro linha = 0; linha < TAMANHO; linha++)

{

para (inteiro coluna = 0; coluna < TAMANHO; coluna++)

{

matriz[linha][coluna] = u.sorteia(1,9)

escreva("[", matriz[linha][coluna],"]")

}

escreva("\n")

}

}

}

## O que são registros?

Um registro é uma estrutura que fornece um formato especializado para armazenar informações em memória.

Enquanto Arrays nos permitem armazenar vários dados de um único tipo de dados, o recurso de Registro nos permite armazenar mais de um tipo de dado.

Um vetor de inteiro só vai possuir valores de inteiros, uma matriz de caracteres só vai possuir caracteres já o registro não, ele permite que nós possamos armazenar dados de tipos diferentes, então um registro pode ter um CPF, um número, um texto, uma data, um decimal, caractere, pode ter tudo misturado.

Um registro é composto por campos que especificam cada uma das informações que o compõem.

Abaixo podemos ver alguns campos de exemplo que constituem o registro de um cliente:

Uma imagem contendo Retângulo

Descrição gerada automaticamente

Lembrando que, cada registro vai ter uma forma dependendo dos requisitos que aquele determinado sistema precisar, então por exemplo, um sistema que cadastra clientes de uma empresa é diferente de um sistema que cadastra clientes de outra empresa.

Toda estrutura de registro tem um nome (ex:livro), e seus campos podem ser acessados por meio do uso do operador ponto (.). Por exemplo, para acessar o preço de um livro, poderíamos utilizar a seguinte declaração: livro.preco

Programa que declara registro e armazena informações de três livros:

ALGORITMO

//declaração do tipo de dado  
tipo  
 estrutura\_livro = registro  
 nome : caracter  
 preco : real  
 pagina : inteiro  
 fimregistro  
  
//declaração das variáveis  
 i inteiro  
 livro array[1...3] de estrutura\_livro  
  
 Escreva(“Entre com os nomes, preços e número de páginas de três livros”)  
 para i de 1 ate 3 faca // leitura de dados  
 Leia(livro[i].nome, livro[i].preco, livro[i].paginas)  
 fimpara  
 Escreva(“Esses foram os dados digitados: “)  
 para i de 1 ate 3 faca  
 Escreva(livro[i]nome, livro[i].preco, livro[i].paginas)  
 fimpara  
FIMALGORITMO

# Entenda o que são Listas, Pilhas e Filas

## Introdução e objetivos

* Listas
* Pilhas
* Filas

## O que são listas?

Estruturas de Dados do tipo Lista, armazena dados de um determinado tipo em uma ordem específica.

A diferença entre listas e arrays é que as listas possuem tamanho ajustável enquanto arrays possuem tamanho fixo. Diferente de quando você possuí um array de cinco valores, a lista ela pode ir preenchendo conforme você vai colocando no programa, diferente do array, do vetor ou da matriz, que você tem que inicializar ela com um valor fixo, a lista você não tem essa necessidade. Então por exemplo, você inicia uma variável do tipo lista e ela vai ali recebendo os valores e vai crescendo, é como se fosse uma lista de nomes para uma festa, ela pode ir crescendo como ela também pode ir diminuindo.

Existem dois tipos de listas:

### Ligadas

Na estrutura do tipo lista existem os nós onde cada um dos nós conhece o valor que está sendo armazenado em seu interior além de conhecer o elemento posterior a ele: por isso ela é chamada de “lista ligada”, pois os nós são amarrados com essa indicação de qual é o próximo nó.

### Duplamente Ligadas

As listas duplamente ligadas constituem uma variação das listas ligadas.

A grande diferença das listas duplamente ligadas para as listas ligadas é que elas são bidirecionais. Vimos que, naturalmente, não conseguimos “andar para trás” em listas ligadas, pois os nós de uma lista ligada sabem somente quem é o próximo elemento. Nas listas duplamente ligadas, os nós sabem quem é o próximo elemento e quem é o elemento anterior, o que permite a navegação reversa.

## O que são pilhas?

Uma pilha é uma estrutura de dados que serve como uma coleção de elementos, e permite o acesso a somente um item de dados armazenado.

Então diferente de um vetor, matriz ou mesmo de uma lista, nós temos a pilha que só permite acesso aos itens uma única vez.

Dentro das pilhas temos dois tipos de pilhas:

### LIFO OU UEPS

A estrutura do tipo PILHA LIFO (Last in First Out) ou UEPS (Último que Entra Primeiro que sai), apresenta o seguinte critério: o primeiro elemento a ser retirado é o último que tiver sido inserido.

### FIFO OU PEPS

A estrutura do tipo PILHA FIFO (First in First Out) ou PEPS (Primeiro que Entra Primeiro que sai), apresenta o seguinte critério: o primeiro elemento a ser retirado é o primeiro que tiver sido inserido.

## O que são filas?

A estrutura do tipo Fila admite remoção de elementos e inserção de novos sujeitos à seguinte regra de operação:

O elemento removido é o que está na estrutura há mais tempo, ou seja, o primeiro objeto inserido na fila é também o primeiro a ser removido seguindo o conceito FIFO.

# Estruturas de dados do tipo Árvore, Tabela Hash e Grafos

## Introdução e objetivos

* Árvore
* Tabela Hash
* Grafos

## O que são árvores?

É uma estrutura de dados que organiza seus elementos de forma hierárquica, onde existe um elemento que fica no topo da árvore, chamado de raiz e existem os elementos subordinados a ele, que são chamados de nós ou folhas.

Uma estrutura do tipo árvore ela facilita a busca, assim como as duas outras estruturas que vimos anteriormente chamada vetor. Então imagine que nós temos um vetor que tem sete posições como número dentro de cada posição. Imagine que eu quero fazer uma busca nesse vetor, como será realizada essa busca? Essa busca será realizada da seguinte maneira, ele vai procurar passo a passo até achar o número desejado. Essa é uma forma de busca mais lenta, onde eu tenho que passar de posição por posição até eu achar determinado valor dentro de um vetor.

Mas imagine a seguinte situação, eu tenho que procurar um número que não tenho dentro desse vetor, como que ficaria? Quando eu coloco um vetor de forma organizada, de forma crescente facilita na busca, mas não é sempre que é colocado o vetor de forma organizada, de forma crescente, as vezes os valores vão estar espalhados.

Para que isso seja facilitado foi criado a estrutura árvore, vamos imaginar que temos o mesmo exemplo acima só que é uma lista ligada, na lista ligada é impossível, porque como vamos definir sendo que cada posição é como se fosse um vetor diferente, por isso então que existe a estrutura do tipo árvore. Ele tem a raiz que é sempre o meio, então ele vai buscar da raiz para as sub-árvores e temos os nós conhecidos também como folhas.

Diagrama

Descrição gerada automaticamente

## O que são tabelas hash?

Uma tabela hash, de dispersão ou espalhamento é uma estrutura de dados especial, que associa chaves de pesquisa a valores.

Uma tabela hash é uma generalização da idéia de array, porém utiliza uma função denominada Hashing para espalhar os elementos, fazendo com que eles fiquem de forma não ordenada dentro do “array” que define a tabela.

No exemplo abaixo para que haja a pesquisa desses elementos, já que eles não estão de forma ordenada, não existe nenhuma classificação dos dados nessa estrutura, o que acontece?

A tabela hash permite a associação de “valores” a “chaves”.

Valores: é a posição ou índice onde o elemento se encontra;

Chave: parte da informação que compõe o elemento a ser manipulado.

Então agora eu não chamo mais ele só pelo índice, eu chamo a chave pelo valor ou o valor pela chave.

Diagrama

Descrição gerada automaticamente

### Por que espalhar?

Espalhar facilita a busca na estrutura de dados, pois a partir de uma chave podemos acessar de forma rápida uma posição do “array”.

Então cada valor recebe uma chave, ele recebe um código, então na hora de pesquisar esse valor ao invés de nós utilizarmos somente a posição nós utilizamos o código referente a esse valor.

Então imagine que nós temos por exemplo, dentro da posição de um vetor não só mais um valor, mas vários valores por exemplo, como dentro da posição de um vetor nós podemos ter lá um cadastro com CPF, RG é possível isso.

Imagine que nós tenhamos um jogo que em uma certa posição você tem que coletar um baú de tesouro, uma maçã ou uma moeda, então ai conforme o que você pega você ganha os pontos, então ele sabe exatamente qual é a chave daquele determinado item que você pegou para que o programa possa retornar aquele item e subir exatamente na pontuação daquele item que você pegou.

## O que são grafos?

Grafos são estruturas que permitem programar a relação entre objetos.

Os objetos são vértices ou “nós” do grafo.

Os relacionamentos são arestas.

Imagine que você está fazendo um jogo de labirinto e conforme você passa tem uma certa dificuldade, tem uma certa pontuação e conforme você passa você tem que armazenar um valor, então você faz uma estrutura na forma de um labirinto e você coloca por exemplo um vértice em cada posição daquele labirinto e conforme a pessoa vai passando ela tem uma pontuação diferenciada. Então eu consigo criar estruturas não lineares, estruturas que não são em filas, ou listas, ou vetores e agora é uma estrutura de qualquer tipo. Esse tipo de estrutura consegue ter um relacionamento em qualquer sentido.

Essas são estruturas especiais que só é utilizada em algoritmos para Inteligência Artificial, para fazer um jogo, para fazer uma busca mais avançada.