Google olabi

<ESTRUTURADEDADOS>

{ O que são estruturas de dados?

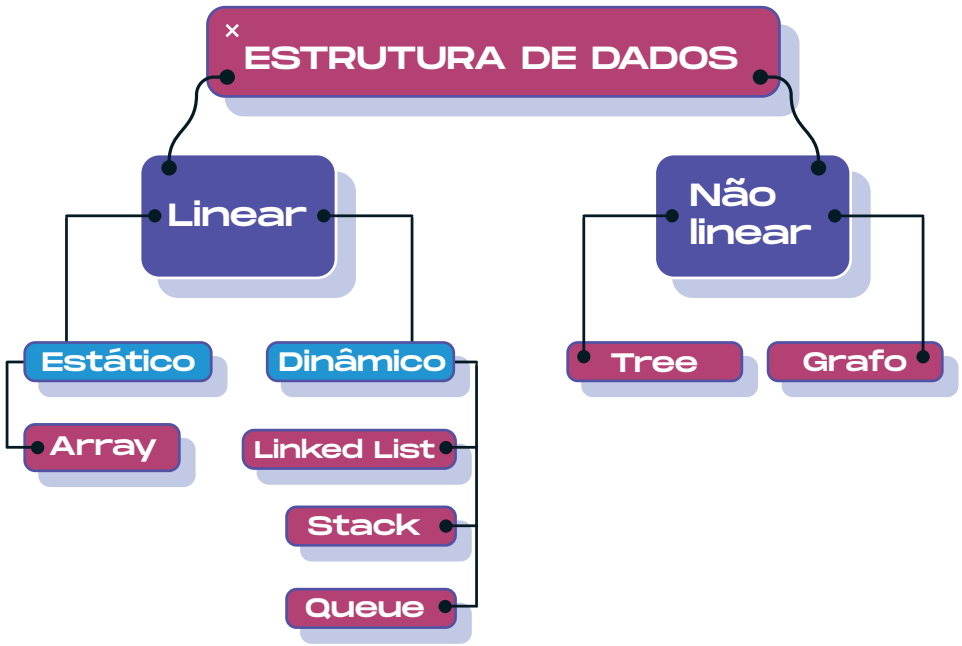
Estruturas de dados compreendem uma coleção de dados com operações e comportamento (ou propriedades) bem definida, que permite armazenar ou organizar dados na memória do computador de maneira eficaz. A estrutura armazena e gerencia dados relevantes permitindo que eles sejam pesquisados instantaneamente.

{ Características de uma estrutura de dados

• **Eficiência:** melhoram o desempenho de um sistema, organizam dados, demandam menos espaço e aumentam a velocidade de processamento;

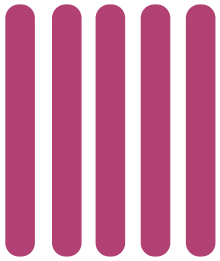
• **Reutilização:** possibilitam a reutilização de dados de forma que, após implementação de uma determinada estrutura de dados, seja possível utilizá-la outras vezes de qualquer lugar;

• **Abstração:** em Java, o ADT (*Abstract Data Type* em inglês) é usado para especificar uma estrutura de dados fornecendo certo nível de abstração. O programa-cliente faz uso da estrutura de dados apenas com a interface, sem depender do conhecimento dos detalhes de implementação;

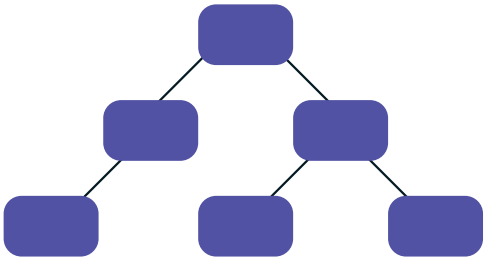


• **linearidade:** todos os elementos são organizados em ordem linear (ou sequencial), o que faz com que a estrutura seja de nível único;

• **não linearidade:** nesse caso, a estrutura não organiza os dados de maneira sequencial como nas estruturas lineares, o que a torna multinível.



Estrutura Linear



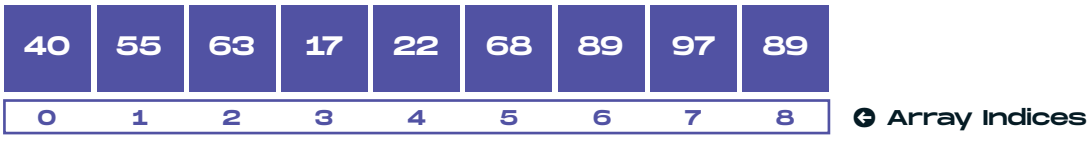
Estrutura Não linear

{ Arrays

Arrays (ou "matrizes") são estruturas de dados usadas para armazenar múltiplos valores. Cada item dentro de uma *array* é chamado de "elemento", e pode ser acessado pelo próprio número (ou "index"). O início ocorre sempre no index 0. Após a criação, o comprimento de uma *array* se torna fixo. Em *strings*, a mesma fórmula é utilizada para descobrir o tamanho de uma *array* de *length*.

Arrays armazenam um número fixo de dados do mesmo tipo de maneira ordenada. Todas as *arrays* em Java têm um campo de comprimento que armazena o espaço alocado para seus elementos. É um valor constante, usado para descobrir a capacidade máxima da *array*.

Utilizaremos a propriedade *length* para verificar o tamanho da *array*. No entanto, lembre-se que está propriedade não nos dará o número de elementos presentes na *array*, mas sim o número máximo de elementos que podem ser armazenados (independentemente de estarem presentes ou não).



Array Length = 9
First Index = 0
Last Index = 8

Percorrendo *arrays*

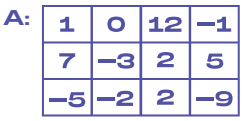
Para percorrer uma *array*, é recomendado o uso do *for* aprimorado. Em caso de necessidade de alteração de valores, o ideal é que o *for* seja controlado por contador. Nesse caso:

1. Usaremos um parâmetro como identificador;
2. Utilizaremos o nome da *array* na qual faremos as iterações.

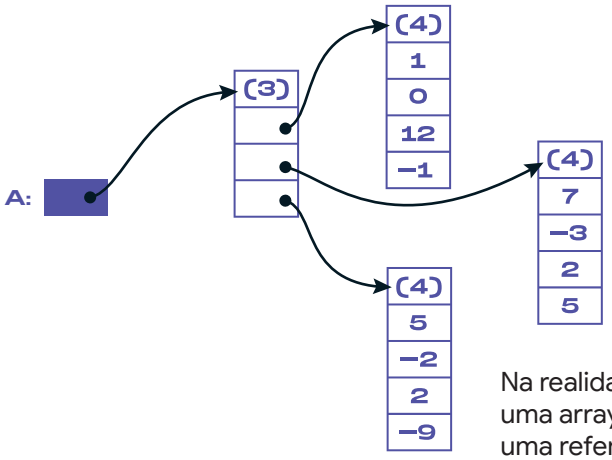
A *For* é uma instrução responsável pela criação de *loop* consistindo de três expressões opcionais: dentro de parênteses; separadas por ponto e vírgula; ou seguidas de uma ou mais declarações executadas em sequência. Recomenda-se utilizar a *for* aprimorada para percorrer uma *array*. Caso precise alterar seus valores, o ideal é optar pela for controlada por contador.

Arrays multidimensionais

Arrays contém duas dimensões, e usualmente são utilizadas para representar tabelas de valores, linhas e colunas. *Arrays* bidimensionais precisam de dois índices para identificar um elemento.



Se você criar uma array A = new int[3][4], você deve pensar nisso como uma “matriz” com 3 linhas e 4 colunas



Na realidade, A contém a referência de uma array de 3 itens, onde cada item é uma referência para uma array de 4 ints

<Sites para estudos>

https://www.devmedia.com.br/trabalhando-com-arrays-em-java/25530
https://docs.oracle.com/javase/tutorial/

Referências:

Fonte oficial: https://docs.oracle.com/javase/tutorial/
Arrays: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html
For: https://www.w3schools.com/java/java_for_loop.asp
Propriedade length:
https://www.delftstack.com/pt/howto/java/size-vs-length-in-java/

Google olabi

<ESTRUTURADEDADOS>

/exercícios/

{ Exercício coletivo I

Busque o índice de cada elemento na *array* e exiba-o.

{ Exercício coletivo II:

Crie uma *array* de acordo com os dados abaixo. Em seguida, implemente a lógica de remoção de elementos duplicados:

String[ ] products = { “PC124X”, “TP123X”, “PC122X”, “PD024X”, “PC124X”, “NO124X”}

{ Exercício coletivo III:

Considerando uma *array* Arr[] de N inteiros, encontre a *subarray* contígua contendo pelo menos um número por meio de soma máxima e retorne-a.

Exemplo 1:

Entrada
N = 5
Arr[] = {1,2,3,-2,5}

Saída
9

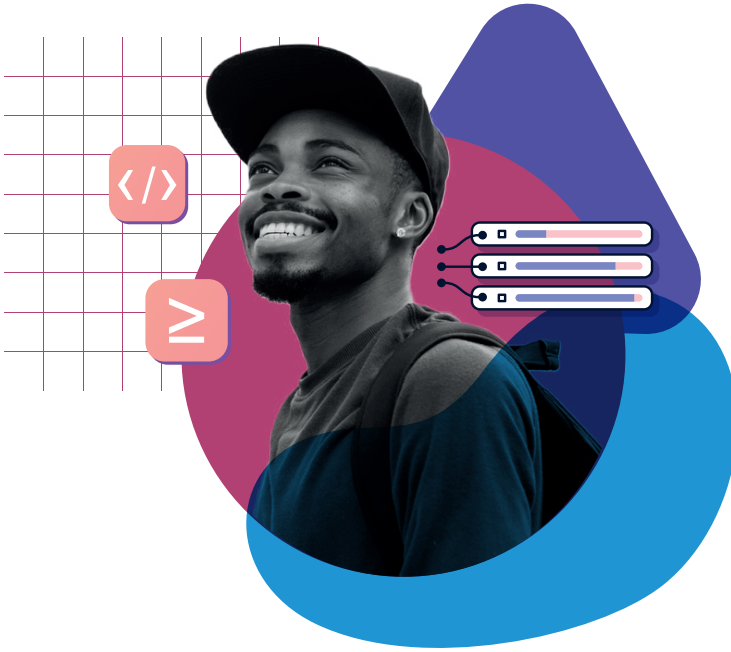
A soma máxima da *subarray* é 9 de elementos (1, 2, 3, -2, 5), ou seja, uma *subarray* contígua.

Exemplo 2:

Entrada
N = 4
Arr[] = {-1,-2,-3,-4}

Saída
-1

Aqui, a soma máxima da *subarray* é -1, do elemento (-1)



{ Consultas para solução do exercício:

https://pt.wikipedia.org/wiki/Sublista_cont%C3%ADgua_de_soma_m%C3%A1xima

https://www.youtube.com/watch?v=ylFB4coxLjQ

Referência para o exercício III:

https://practice.geeksforgeeks.org/