

# Agenda 08: Matrizes

Conceitos trabalhados:

Sumário

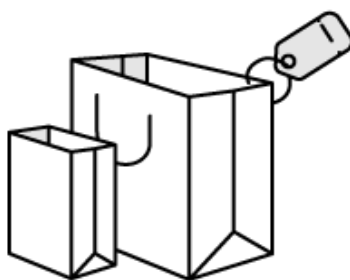
- Momento de Reflexão
- Por que Aprender?
- Para Começar o Assunto
- Mergulhando no Tema
- Ampliando Horizontes
- Resumindo o Estudo

## Momento de Reflexão

Olá, estudante ;)

Boas-vindas à oitava agenda do módulo 1! Ao longo deste percurso, você teve contato com conceitos fundamentais da lógica de programação e, agora, conhecerá mais um tipo especial de variável: as **matrizes**.

Para dar início à aula, que tal conhecer a história de Amanda?



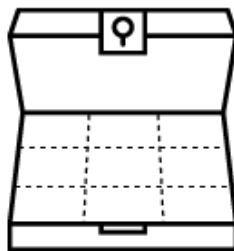
Amanda é fascinada por bijuterias! Ela vive comprando, principalmente porque nunca lembra das peças que já tem em casa...



É que Amanda não organiza seus acessórios, então quando pega o brinco azul, só encontra o anel vermelho. Parece que nada combina!



Vira e mexe, Amanda percebe que comprou peças muito parecidas, e fica frustrada.



Cansada de gastar o curto dinheiro à toa, decidiu preparar uma caixa com 3 fileiras, cada uma com 3 repartições.



Assim, Amanda conseguiu organizar suas peças por cores, sabendo exatamente como combinar as bijuterias!

Você percebeu que, para organizar as bijus de forma lógica, Amanda escolheu um tipo especial de caixa?

Desse mesmo modo acontece com os dados no mundo da programação, quando precisam ser **armazenados ou lidos de acordo com a relação entre eixos** e representados por mais de um valor dentro de uma variável. É o que chamamos de **matrizes**, variáveis que armazenam múltiplos valores **com mais de uma dimensão**, assim como a caixa de Amanda apresenta fileiras (colunas) com mais de uma repartição!

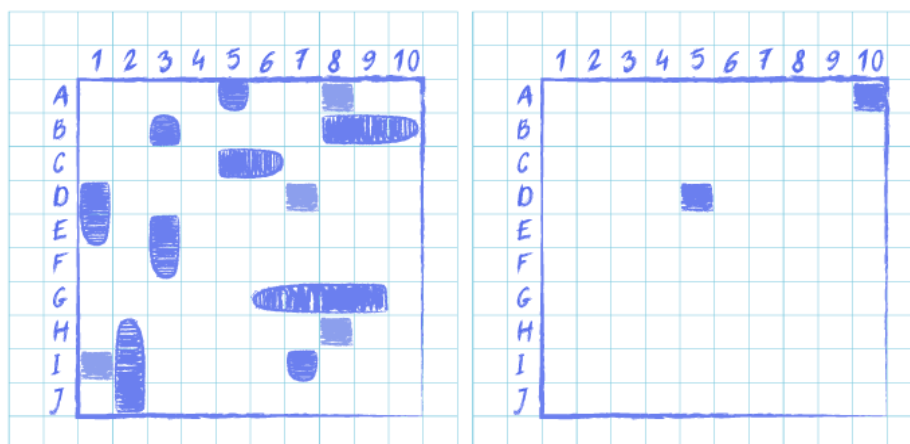


## Por que Aprender?

Na agenda anterior você estudou os vetores que, assim como as matrizes, também são variáveis especiais que armazenam mais de um dado. Porém, os vetores são variáveis indexadas com apenas **uma dimensão** (uma coluna com várias linhas). No entanto, existem situações em que é preciso trabalhar com mais de um tipo de informação, quando são necessárias mais colunas: nesses casos, há variáveis nos sentidos horizontal e vertical. Por esse motivo, é muito importante você estudar **matrizes de mais de uma dimensão**!

## Para Começar o Assunto

Você já sabe que vetores são matrizes com uma única dimensão – linha ou coluna. Um modo fácil de entender como os elementos de uma matriz são ordenados é realizar uma comparação com o jogo [Batalha Naval](#) ([Links to an external site.](#)), tradicionalmente jogado em papel. De modo breve, o jogador precisa afundar as embarcações do adversário, escolhendo em qual posição do campo de batalha irá lançar seus torpedos. Para isso, a superfície é organizada em colunas (representadas por números) e linhas (representadas por letras). Para sinalizar seu alvo, o jogador deve utilizar uma combinação de letra e número, por exemplo A10 ou D5.



Nas matrizes, a identificação é idêntica, mas são utilizados **somente números** para a representação e, para que não haja confusão, usam-se **duas variáveis distintas**: uma para a representação das linhas e outra para a das colunas, como você verá no Mergulhando no Tema.

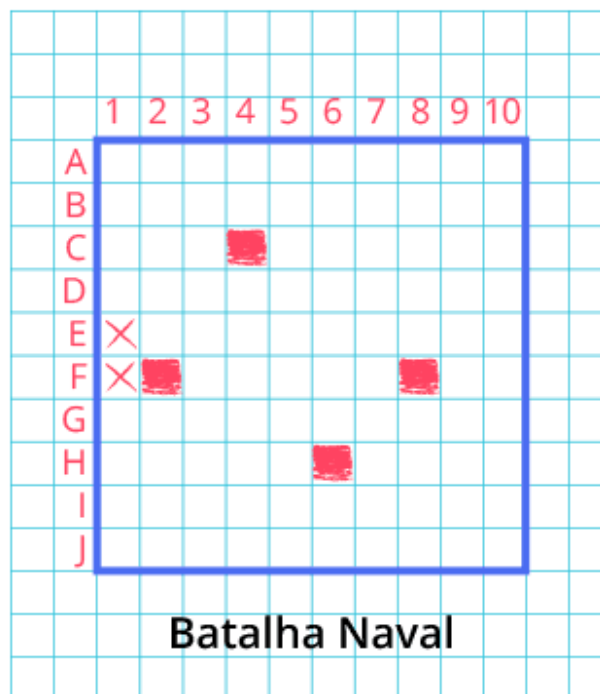
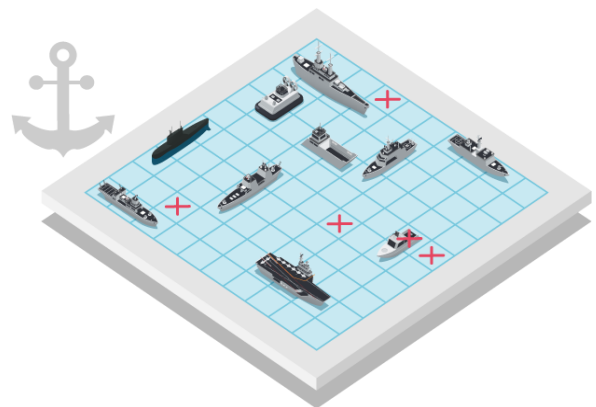
Você compreenderá melhor ao estudar os materiais a seguir.

# Mergulhando no Tema

## O que são Matrizes?

Você já sabe que, na programação, existem variáveis especiais capazes de armazenar múltiplos dados, utilizadas para organizar e facilitar a escrita do seu código, não é mesmo? **Matrizes**, conhecidas também por Array (termo originário do inglês), são variáveis desse tipo. Elas podem guardar múltiplos valores em **mais de uma dimensão**, ou seja, seus dados podem ser **armazenados ou lidos de acordo com a relação entre eixos**.

Os elementos de uma matriz são ordenados como em um jogo de Batalha Naval: em linhas e colunas identificadas. Mas, diferentemente do jogo, para indicar a localização dos elementos dentro da matriz, são utilizados apenas números, chamados de **índices**.



Então, na programação as matrizes são identificadas utilizando-se **duas variáveis distintas**: uma para a representação das **linhas** e outra para a das **colunas**. Observe a matriz representada a seguir: em vermelho, estão destacados os índices para a identificação de cada elemento da

matriz. Por exemplo, para endereçarmos o elemento de número 9, teremos: `linha[2], coluna[100]` = 9.

	1	2	...	100
1	10	7	...	
2	34	78	...	9
...	...	...	...	18
100	6	3	...	54

### Conhecendo um exemplo







As matrizes são utilizadas constantemente no cotidiano, mas não acostumamos a nos dar conta. Veja o exemplo a seguir:

O professor Rafael deseja realizar o fechamento das notas bimestrais de sua turma. Naquele bimestre, cada aluno realizou três atividades valendo nota. Para organizar esses dados, o professor preparou uma tabela.

Considerando que todas as notas podem ser números reais, teremos tipos de dados idênticos para todos os campos, inclusive para a média das notas. Assim, **cada uma das notas individuais dos alunos da tabela representa um elemento da matriz**. E esses elementos podem ser



facilmente referenciados para a realização de cálculos ou outra operação necessária. Veja a seguir:

	N1	N2	N3	média
	5	7	6	6
	8	8	8	8
	...	...	...	...
	10	8	9	9

**Tabela de Notas**

➔

5	7	6	6
8	8	8	8
...	...	...	...
10	8	9	9

**Matriz**

Na matriz representada acima, temos as notas das 3 atividades nas três primeiras colunas e a média bimestral na última coluna. Se quisermos buscar a terceira nota do primeiro aluno, teremos: `linha[1], coluna[3] = 6`. As matrizes são um modo muito simples e eficaz de representarmos os dados em programação.

### Como utilizar uma matriz em Java?

Agora que você já compreendeu o que é uma matriz e como ela é declarada, é momento de focar na parte prática da utilização em pseudocódigo e em Java. Veja a sintaxe da declaração de uma matriz bidimensional, a seguir:

#### PSEUDOCÓDIGO

##### Declare

<nome> como **conjunto** [1..n][1..m] de <tipo>

#### JAVA

<tipo> <nome>[ ][ ] = new <tipo>[n][m];

Onde temos <nome>, você indicará o nome da matriz. Em <tipo>, deve ser indicado o tipo de variável a ser armazenada. Em n, você deverá especificar o número de linhas e, em m, o de colunas.

Veja um exemplo de declaração de matriz bidimensional:

#### PSEUDOCÓDIGO

##### Declare

mat como **conjunto** [1..10][1..10] de inteiro

#### JAVA

<tipo> <nome>int mat[ ][ ] = new int [10][10];

Tanto o pseudocódigo quanto o Java declaram uma matriz **mat** com duas dimensões (linha e coluna) com 100 posições no total (10 linhas x 10 colunas).

	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										

Pseudocódigo

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										
7										
8										
9										

Java

Aqui cabe uma observação muito importante: note que foram ilustradas duas matrizes acima. Você reparou que na matriz de **pseudocódigo**, a **contagem inicia-se pelo número 1 (um)**? Já na matriz do **Java**, a **contagem inicia-se pelo número 0 (zero)**.

**Lembre-se disso!** Quando utilizamos **Arrays em Java**, sejam eles unidimensionais (vetores) ou bidimensionais (matrizes), a **contagem sempre se inicia do zero**.

#### Dica:



Isso mesmo! Você pode utilizar esses termos: **Arrays Unidimensionais** para informar o uso de vetores (matrizes unidimensionais) e **Arrays Bidimensionais** para o uso de matrizes bidimensionais. Essa é a diferença!

Veja agora outro exemplo:

### PSEUDOCÓDIGO

nomes como **conjunto** [1..10][1..5] de caractere  
num como **conjunto** [1..4][1..4] de real

### JAVA

#### Declare

```
String nomes[ ][ ] = new String [10][5];
double num[ ][ ] = new double [4][4];
float num1[ ][ ] = new float [4][4];
```

Observe que o número de linhas e o número de colunas indicados nos [ ] (colchetes) não precisam ser iguais.

### Declarando uma Matriz Inicializada

Você já compreendeu como declarar uma matriz em seu programa, mas percebeu que não indicou nenhum valor presente nos elementos da matriz? Em outras palavras, até este momento, você sabe declarar uma matriz vazia. Agora, pense, pesquise e responda:

**Como você poderia declarar uma matriz 2x2 (duas linhas por duas colunas) com seus valores inteiros já inicializados em Java?**

Agora, analise a declaração da matriz na codificação a seguir e entenda como inicializar uma matriz com valores pré-definidos:

```
import javax.swing.JOptionPane;
    public class MatrizIncializada {
        public static void main(String[] args) {
            // Matriz 2x2 inicializada

            //declaração da matriz e inicialização
            int mat[][] = { {1,2}, {3,4} };

            //saída de valores da matriz
            for (int linha= 0; linha< 2;linha++) {
                for(int coluna = 0; coluna< 2; coluna++)
                {

                    JOptionPane.showMessageDialog(null, "Matriz[" + linha + "]" + "coluna[" +
                    coluna + "]" = " + mat[linha][coluna]);
                }//fim do segundo for
            }//fim do primeiro for
        }//fim do main
    }//fim da classe
```

Na agenda anterior, você estudou vetores e agora pode utilizar a mesma lógica para declarar um vetor com dados inicializados.

### Como acessar os dados da Matriz?

O acesso aos elementos de uma matriz é feito utilizando **comandos de repetição**. Durante o curso, você viu três comandos desse tipo, sendo o **para...fim-para** (for no Java) o mais adequado para utilização com matrizes. Isso porque ele executa uma repetição por um número fixo de vezes e, como você já sabe, uma matriz possui um número fixo de linhas e colunas.

O exemplo a seguir considera uma matriz 4x4 de números inteiros, com as variáveis linha e coluna controlando o acesso à linha e à coluna das matrizes. Nela, o usuário irá incluir os valores

presentes em cada elemento da matriz, ou seja, trata-se de uma matriz não-inicializada. Veja como ficaria a sintaxe do comando em pseudocódigo e em Java:

### PSEUDOCÓDIGO

#### Programa MatrizExemplo

##### Declare

num como **conjunto** [1..4][1..4] de real  
linha, coluna como inteiro

##### Início

Escreva("inserindo os dados na Matriz")

##### Para linha = 1 Até 4 Faça

##### Para coluna = 1 Até 4 Faça

Escreva ("Entre com um número")

Escreva ("linha", linha)

Escreva ("coluna, coluna)

Leia num[linha, coluna]

##### Fim-Para

##### Fim-Para

Escreva("Mostrando os dados na Matriz")

##### Para linha = 1 Até 4 Faça

##### Para coluna = 1 Até 4 Faça

Escreva ("linha", linha)

Escreva ("coluna, coluna)

Escreva ("valor lido", num[linha, coluna])

##### Fim-Para

##### Fim-Para

##### Fim

### JAVA

```
double num [][] = new double [4][4];
int linha, coluna;
JOptionPane.showMessageDialog (null, "Inserindo os dados
na Matriz");
for (linha = 0; linha < 4; linha++) {
    for (coluna = 0; coluna < 4; coluna++) {
        num[linha][coluna] = Double.parseDouble(
JOptionPane.showInputDialog("Entre com o
número" + "\nlinha " + "" + linha + "\ncoluna " + coluna));
    }
}

JOptionPane.showMessageDialog (null, "Mostrando os dados
na Matriz");
for (linha = 0; linha < 4; linha++) {
    for (coluna = 0; coluna < 4; coluna++) {
```

```
        JOptionPane.showMessageDialog(null, "linha" +  
        linha + "\ncoluna " + coluna + "\nNúmero" + num[linha][coluna])  
    }  
}
```

O pseudocódigo inicia-se com a declaração da matriz **num 4x4** e das variáveis **linha** e **coluna** que servirão para acessarmos os elementos da matriz. Logo depois, temos **dois comandos de repetição encadeados**: um para controlarmos a **linha** (Para linha) e outro para a **coluna** (Para coluna). Dessa forma, os elementos da matriz serão inseridos na primeira linha, uma coluna por vez, e passa-se para a linha seguinte ao chegar ao final (coluna 4).

Note que no interior desses dois comandos de repetição temos uma entrada de dados (**Escreva**), para que o usuário insira o valor do elemento **num[linha][coluna]** da matriz. Então, quando a linha = 1 e a coluna = 2, teremos acesso ao elemento **num[1][2]**.

O mesmo ocorre na etapa “Mostrando os dados da matriz”, porém, ao invés de realizarmos a leitura do valor **num[linha][coluna]**, realizamos a escrita do mesmo.

### Agora, confira o programa Java.

Em Linguagem Java ocorre algo semelhante ao visto no pseudocódigo: primeiro declara-se a matriz e as variáveis de controle, para na primeira dupla do comando **for** realizar-se a entrada dos dados. Os dados inseridos são exibidos na segunda dupla.

Na linha 4, você pode ver a instrução “linha++”, dentro do comando **for**. Ela é igual à conta: linha = linha + 1.

#### Faça o teste!

Quando você executar o exemplo acima em Java, serão geradas várias janelas com a exibição dos dados. Serão pelo menos 16 janelas para a entrada e mais 16 para a saída de dados. Então, **você tem um desafio**: tente modificar o programa de modo que ele fique com as 16 janelas de entrada e somente 1 janela de saída.

Você encontrará a solução para esse desafio mais adiante, no tópico **Praticando o uso de uma Matriz**

Perceba que, para resolver o problema, basta mover o **JOptionPane** para fora do laço **for**

```
double num [] [] = new double [4][4];
int linha, coluna;
JOptionPane.showMessageDialog (null, "Inserindo os dados na
Matriz");
for (linha = 0; linha < 4; linha++) {
    for (coluna = 0; coluna < 4; coluna++) {
        num[linha][coluna] = Double.parseDouble(
            JOptionPane.showInputDialog("Entre com
o número" + "\nlinha " + "" + linha +
"\ncoluna " + coluna));
    }
}
JOptionPane.showMessageDialog (null, "Mostrando os dados na
Matriz");
for (linha = 0; linha < 4; linha++) {
    for (coluna = 0; coluna < 4; coluna++) {
        JOptionPane.showMessageDialog(null,"linha "
+ linha + "\ncoluna " + coluna + "\nNúmero
" + num[linha][coluna])
    }
}
```

Agora que você já compreendeu o conceito de Matrizes e como aplicá-las, assista à videoaula a seguir, gravada pelo professor Rogério Silva, que sintetiza os conteúdos apresentados.

### DS - Lógica - Matriz em JAVA



**Retornando ao exemplo**

No começo deste texto, você viu que o professor Rafael desejava calcular a média bimestral de seus estimados alunos e, para isso, organizou os dados em uma matriz. Considerando que a turma tem 10 alunos, cada um realizou 3 atividades e uma coluna deve ser reservada para a média, sabemos que essa matriz tem 10 linhas e 4 colunas. Como poderíamos codificar um programa que realizasse esse cálculo?



Veja o pseudocódigo a seguir:

## PSEUDOCÓDIGO

### Programa MatrizExemplo

#### Declare

```
notas como conjunto [1..10][1..4] de real
linha, coluna como inteiro
media como real
```

#### Início

```
Escreva("inserindo os dados na Matriz")
```

```
Para linha = 1 Até 10 Faça
```

```
    Para coluna = 1 Até 3 Faça
```

```
        Escreva ("Entre com um número")
```

```
        Escreva ("linha", linha)
```

```
        Escreva ("coluna, coluna)
```

```
        Leia notas[linha, coluna]
```

```
    Fim-Para
```

```
Fim-Para
```

```
Escreva("calculando...")
```

```
Media = 0
```

```
Para linha = 1 Até 10 Faça
```

```
    Para coluna = 1 Até 3 Faça
```

```
        media = media + notas[linha, coluna]
```

```
    Fim-Para
```

```
    notas[linha,4] = media/3
```

```
    media = 0
```

```
Fim-Para
```

```
Escreva("Mostrando os dados na Matriz")
```

```
Para linha = 1 Até 10 Faça
```

```
    Para coluna = 1 Até 3 Faça
```

```
        Escreva ("linha", linha)
```

```
        Escreva ("coluna, coluna)
```

```
        Escreva ("valor lido", notas[linha, coluna])
```

```
    Fim-Para
```

```
    Escreva ("Média" ,notas[linha,4])
```

```
Fim-Para
```

```
Fim
```

Agora, o programa em Java:

```
import javax.swing.JOptionPane;

public class MatrizExemploMedia {

    public static void main(String[] args) {
        //declaração de variáveis
        double notas [] [] = new double [10][4];
```

```

        int linha, coluna;
        double media = 0;

        //entrada de dados
        JOptionPane.showMessageDialog (null,
        "Inserindo os dados na Matriz");
        for (linha = 0; linha < 10; linha++) {
            for (coluna = 0; coluna < 3;
            coluna++) {
                notas[linha][coluna] =
                Double.parseDouble(JOptionPane.showInputDialog("Entre com o número"
                + "\nlinha " + linha + " \ncoluna " + coluna));
            } //fim do for
        } //fim do for

        //cálculo da média
        for (linha = 0; linha < 10; linha++) {
            for (coluna = 0; coluna < 3;
            coluna++) {
                media = media +
                notas[linha][coluna];
            } //fim do for
            notas[linha][3] = media/3;
        } //fim do for

        //saída de dados
        JOptionPane.showMessageDialog (null, "Mostrando os dados
        na Matriz");
        for (linha = 0; linha < 4; linha++) {
            for (coluna = 0; coluna <
            4; coluna++) {
                JOptionPane.showMessageDialog(null, "linha " +
                linha + " \ncoluna " + coluna + " \nNúmero " + notas[linha][coluna]);
            } //fim do for
            //exibe média de aluno
            JOptionPane.showMessageDialog(null, "Média: " +
            notas[linha][3]);
        } //fim do for
    } //fim do método main
} //fim da classe

```

### Praticando o uso de uma Matriz

Agora que você já compreendeu o conceito de Matriz e viu o exemplo do professor Rafael, procure solucionar o desafio de Paulo:



Como estudante do Ensino Médio, Paulo deseja fazer um software que realize a soma de duas matrizes 4x4. Ele sabe que, para realizar a soma de duas matrizes, segundo a matemática, cada elemento da matriz A deve ser somado ao seu elemento correspondente da matriz B, gerando o resultado em uma terceira matriz C. Elabore o pseudocódigo e a codificação em linguagem Java de um software que resolva o desafio de Paulo.

**Dica:**

Para fazer a soma dessas duas matrizes, deve-se ler as matrizes A e B, cada uma de duas dimensões, com 4 linhas e 4 colunas. Construir a matriz C, de mesma dimensão, ou seja, formada pela soma dos elementos da matriz A com os elementos da matriz B e apresentar os elementos da matriz C. Soma de matrizes:  $A[1,1] + B[1,1] = C[1,1]$ .

Compare sua resposta com a solução a seguir:

## PSEUDOCÓDIGO

### Programa MatrizEx1

#### Declare

```
a como conjunto [1..4][1..4] de inteiro
b como conjunto [1..4][1..4] de inteiro
c como conjunto [1..4][1..4] de inteiro
linha, coluna como inteiro
```

#### Início

```
Escreva("inserindo os dados na Matriz A")
```

```
Para linha = 1 Até 4 Faça
```

```
    Para coluna = 1 Até 4 Faça
```

```
        Escreva ("Entre com um número")
```

```
        Escreva ("linha", linha)
```

```
        Escreva ("coluna, coluna)
```

```
        Leia a[linha,coluna]
```

```
    Fim-Para
```

```
Fim-Para
```

```
Escreva("inserindo os dados na Matriz B")
```

```
Para linha = 1 Até 4 Faça
```

```
    Para coluna = 1 Até 4 Faça
```

```
        Escreva ("Entre com um número")
```

```
        Escreva ("linha", linha)
```

```
        Escreva ("coluna, coluna)
```

```
        Leia b[linha,coluna]
```

```
    Fim-Para
```

```
Fim-Para
```

```
Escreva("calculando matriz c")
```

```
Para linha = 1 Até 4 Faça
```

```
    Para coluna = 1 Até 4 Faça
```

```
        c[linha,coluna] = a[linha,coluna] + b[linha,coluna]
```

```
    Fim-Para
```

```
Fim-Para
```

```
Escreva("Mostrando os resultado da matriz C")
```

```
Para linha = 1 Até 10 Faça
```

```
    Para coluna = 1 Até 3 Faça
```

```
        Escreva ("linha", linha)
```

```
        Escreva ("coluna, coluna)
```

```
        Escreva ("valor calculado", c[linha,coluna])
```

```
    Fim-Para
```

```
Fim-Para
```

Programa em Java:

```
import javax.swing.JOptionPane;
public class MatrizEx1 {

    public static void main(String[] args) {
        // exercício 1
        //declaração de variáveis
        int a[][] = new int [4][4];
        int b[][] = new int [4][4];
        int c[][] = new int [4][4];
        int linha, coluna;

        //entrada de dados
```

```

//matriz A
for (linha = 0; linha < 4; linha++) {
    for (coluna = 0; coluna < 4; coluna++) {
        a[linha][coluna]=
Integer.parseInt(JOptionPane.showInputDialog("Entre com o elemento
[" + linha + "][" + coluna + "] da matriz A"));
    }//fim do for
}//fim do for

//matriz B
for (linha = 0; linha < 4; linha++) {
    for (coluna = 0; coluna < 4; coluna++) {
        b[linha][coluna]=
Integer.parseInt(JOptionPane.showInputDialog ("Entre com o elemento
[" + linha + "][" + coluna + "] da matriz B"));
    }//fim do for
}//fim do for

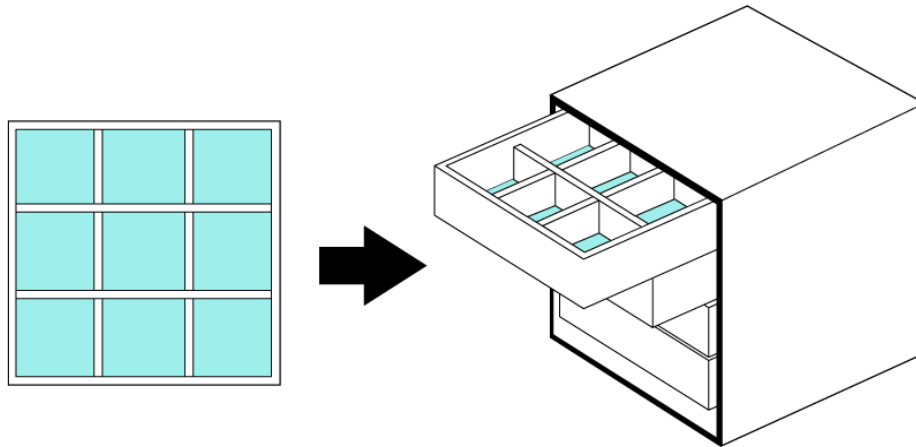
//cálculo
for (linha = 0; linha < 4; linha++) {
    for (coluna = 0; coluna < 4; coluna++) {
        c[linha][coluna]= a[linha][coluna] +
b[linha][coluna];
    }//fim do for
}//fim do for

//saída de dados
for (linha = 0; linha < 4; linha++) {
    for (coluna = 0; coluna < 4; coluna++) {
        JOptionPane.showMessageDialog(null,
"C[" + linha + "][" + coluna + "] = " + c[linha][coluna]);
    }//fim do for
}//fim do for
    }
}

```

### Utilizando uma Matriz Tridimensional

Até o momento você viu como utilizar matrizes bidimensionais, mas saiba que também é possível utilizar **Arrays Tridimensionais**. Se podemos relacionar as matrizes bidimensionais às caixas organizadoras com diversos compartimentos, podemos entender que matrizes tridimensionais seriam como caixas especiais, com diversas gavetas bidimensionais.



Para utilizá-las, basta tomar alguns cuidados:

- Para declarar a matriz: utilizar uma variável a mais. Pensando em uma matriz de 2 x 2 x 2 elementos, o código Java ficará assim:  
`double matriz [ ][ ][ ] = new double [2][2][2];`
- Para acessar os elementos: acrescentar mais um **for** em seu código

Na videoaula a seguir, o professor Sandro Valérius retoma os conceitos vistos até aqui e apresenta mais um exemplo de uso de matrizes, confira!



**Tratando erros em Matrizes**

Você já imaginou como é possível tratar erros utilizando o comando Try-Catch com matrizes? Veja o programa a seguir, preparado com base no exemplo do professor Rafael.

```
import javax.swing.JOptionPane;

public class MatrizExemploMedia {

    public static void main(String[] args) {
        double notas [] [] = new double [10][4];
        int linha, coluna;
        double media = 0;
        try {
            JOptionPane.showMessageDialog (null, "Inserindo os
dados na Matriz");
            for (linha = 0; linha < 10; linha++) {
                for (coluna = 0; coluna < 3; coluna++) {
                    notas[linha][coluna] =
Double.parseDouble(JOptionPane.showInputDialog("Entre com o número"
+ "\nlinha " + linha + "\ncoluna " + coluna));
                }//fim do for
            }//fim do for

            for (linha = 0; linha < 10; linha++) {
                for (coluna = 0; coluna < 3; coluna++) {
                    media = media + notas[linha][coluna];
                }//fim do for
                notas[linha][3] = media/3;
            }//fim do for

            JOptionPane.showMessageDialog (null, "Mostrando os
dados na Matriz");
            for (linha = 0; linha < 4; linha++) {
                for (coluna = 0; coluna < 4; coluna++) {
                    JOptionPane.showMessageDialog(null,"linha
" + linha + "\ncoluna " + coluna + "\nNúmero " + notas[linha]
[coluna]);
                }//fim do for
                JOptionPane.showMessageDialog(null, "Média: "
+ notas[linha][3]);
            }//fim do for
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(null, "Entre somente
com números\nEncerrando","ERRO",JOptionPane.ERROR_MESSAGE);
        } //fim do try-catch
    } //fim do método main
} //fim da classe
```



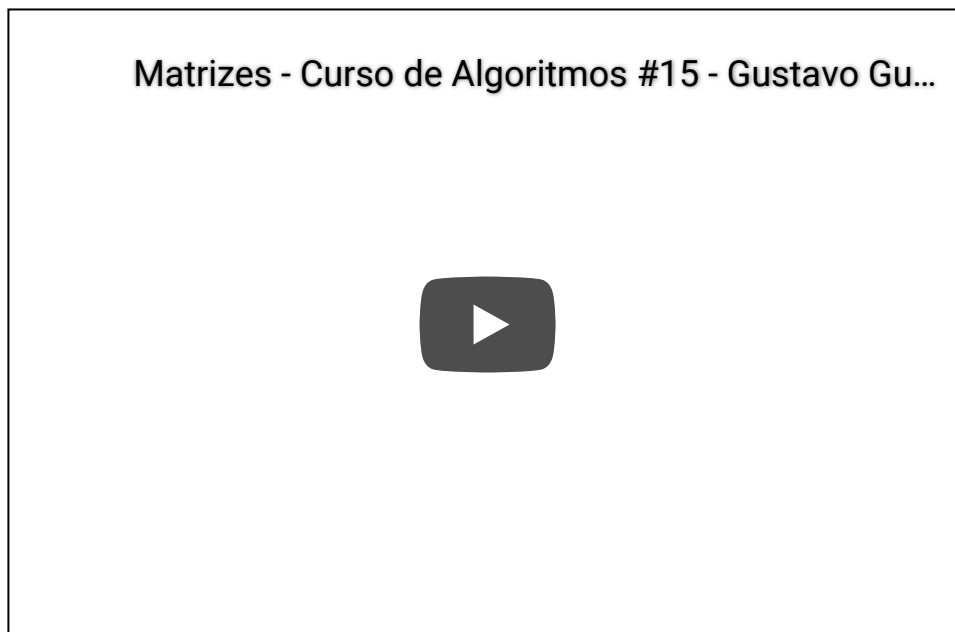
# Ampliando Horizontes

## Onde encontro mais informações sobre Matrizes?

Para aprofundar o tema desta agenda, você pode consultar os materiais indicados a seguir. Essas dicas são muito importantes para você!

### Matrizes - Curso de algoritmos #15

Neste vídeo, Gustavo Guanabara explica o conceito de Matrizes.



**Lógica de programação: a construção de algoritmos e estrutura de dados.** FORBELLONE, André L. V; ELBERSPACHER, Henri Frederico. Editora Pearson, 2000.

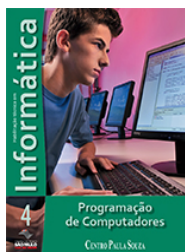
**Algoritmos: lógica para desenvolvimento de programação.** MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo. Editora Érica, 2007.



**Lógica de programação e estruturas de dados com aplicações em Java.** PUGA, Sandra; RISSETTI, Gerson. Editora Pearson, 2009.



**Java para iniciantes.** SCHILDT, Hebert. Editora Bookman, 2015.



**[Informática, programação de computadores | Vol. 4 \(Links to an external site.\).](#)**

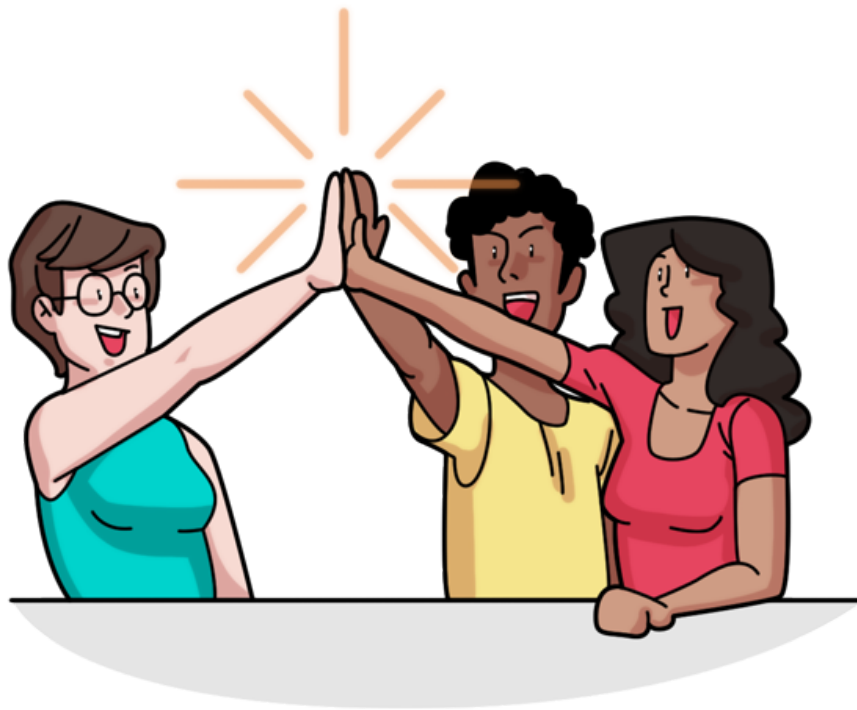
Centro Paula Souza. (2010). São Paulo: Fundação Padre Anchieta.



## Resumindo o Estudo

Até agora você estudou conteúdos fundamentais da **Lógica de Programação** e agora está pronto para avançar para a segunda etapa deste módulo, onde aprenderá sobre **Banco de Dados**!

Na próxima agenda onde estudará os **conceitos básicos dos bancos de dados**.



**Vamos lá!**

