

Linguagem de Programação

MATRIZES

Prof. Me. Humberto Zanetti



Matrizes

- O que são matrizes?
- Declarando uma matriz
- Percorrendo uma matriz
- Exibindo e preenchendo uma matriz

Matrizes

- Matrizes, ou **vetores bidimensionais**, são estruturas de dados que podem ser acessados de maneira similar aos **vetores unidimensionais**
 - **Através de um único nome e índices diferentes**
- Conceitualmente, o Java apresenta uma solução de criar uma “**vetor de vetor**”, criando uma multi-dimensão de índices.
 - Na aula, vamos abordar apenas 2 dimensões.

Declarando uma matriz

define uma matriz

quantidade de
linhas

```
tipo nome_matriz[][] = new int [l][c]
```

quantidade de
colunas

```
int m1[][] = new int [3][3]
```

```
int m2[][] = new int [2][4]
```

m1

	0	1	2
0			
1			
2			

3 x 3

m2

	0	1	2	3
0				
1				

2 x 4

Matriz na memória

```
int matriz[][] = new int[3][3]
```

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

	0	1	2
0	matriz[0][0]	matriz[0][1]	matriz[0][2]
1	matriz[1][0]	matriz[1][1]	matriz[1][2]
2	matriz[2][0]	matriz[2][1]	matriz[2][2]

matriz[0][0]
matriz[0][1]
matriz[0][2]
matriz[1][0]
matriz[1][1]
matriz[1][2]
matriz[2][0]
matriz[2][1]
matriz[2][2]

Memória

2

4

7

8

9

1

3

6

0

...

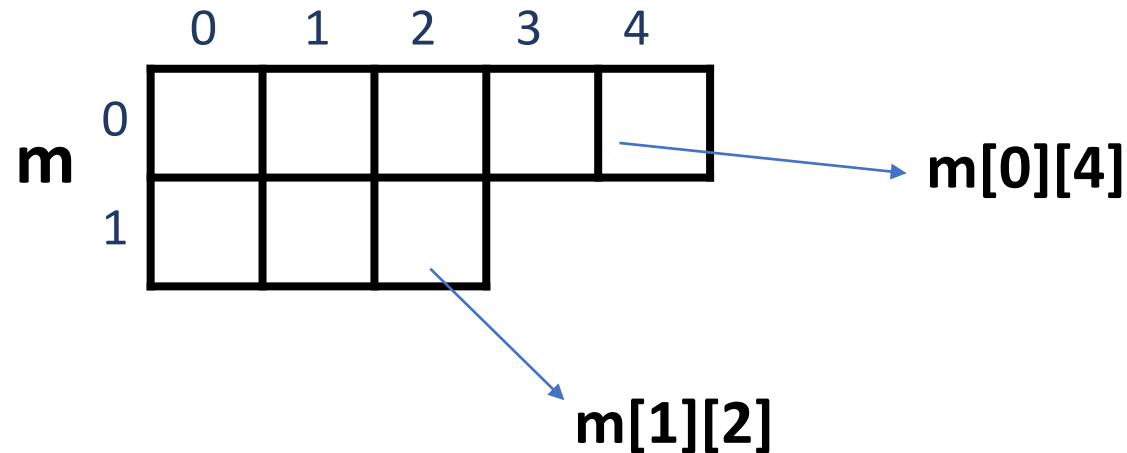
Matriz inicializada

```
int m[][] = {  
    1ª linha    2ª linha    3ª linha  
    {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
```

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

Linhas de tamanhos diferentes

```
int m[][] = new int[2][]; // cria 2 linhas  
// cria 5 colunas para a linha 0  
m[0] = new int[5];  
// cria 3 colunas para a linha 1  
m[1] = new int[3];
```

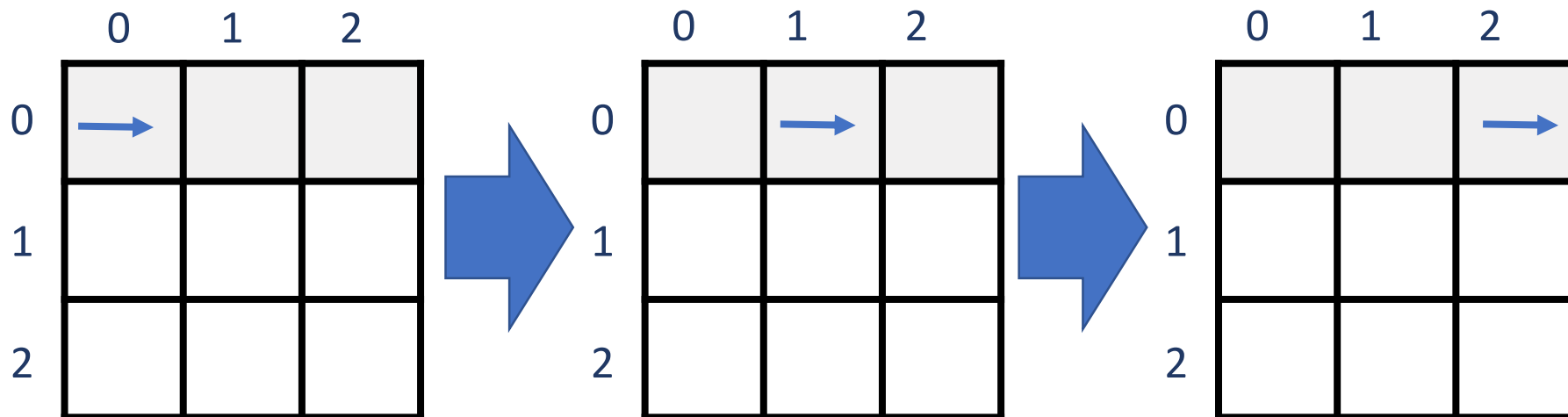


Percorrendo uma matriz

- Similaridade com os vetores unidimensionais:
 - Percorreremos todos os elementos através dos índices;
 - Temos que identificar quais os limites das dimensões, para não acessar espaços de memórias não alocados;
 - Geralmente utilizamos laços de repetição.
- Problemas:
 - As dimensões entre linhas e colunas podem variar:
 - 2×3 ; 4×7 ; 5×11 ...
 - Podemos ter variações até mesmo entre as colunas...

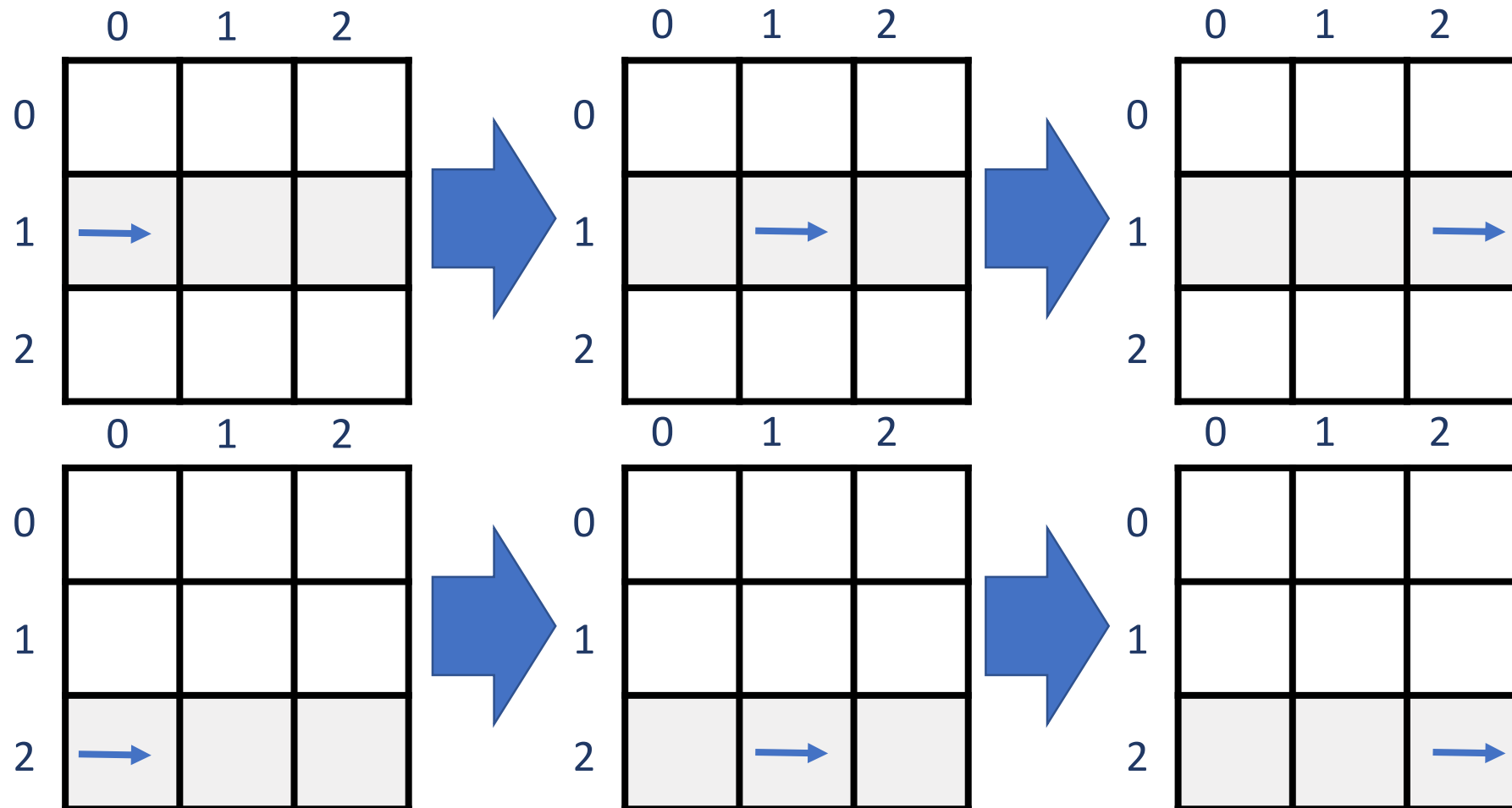
Percorrendo uma matriz

- Estratégia:
 - Lembrar que matriz é “vetor de vetor”
 - Temos que identificar a quantidade de linhas...
 - ...e depois a quantidade de colunas por linha
- Dinâmica básica: a cada linha, eu percorro as colunas!
 - Começando pela 1ª linha (“primeiro vetor”)



Percorrendo uma matriz

- Depois vá para a próxima linha (próximo vetor)



Percorrendo uma matriz

- E como fazemos isso, andar em cada linha e em cada coluna!?
 - Laços de repetição aninhados!

```
int i, j;  
    para as linhas      para as colunas  
  
for (i = 0; i < qtd_linhas; i++) {  
    ...  
    for (j = 0; j < qtd_colunas; j++) {  
        ....  
    }  
}
```

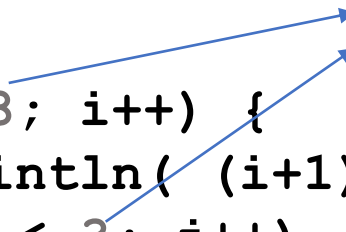
percorre as linhas

percorre as colunas

Exibindo uma matriz

```
int m[][] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };  
  
int i, j;  
  
for (i = 0; i < 3; i++) {  
    System.out.println( (i+1) + "a. linha: " );  
    for (j = 0; j < 3; j++) {  
        System.out.println(m[i][j]);  
    }  
}
```

Podemos simplesmente colocar
as dimensões das linhas e
colunas

The text box is a blue-bordered rectangle containing the text "Podemos simplesmente colocar as dimensões das linhas e colunas". Two blue arrows originate from the text box: one points to the number '3' in the condition 'i < 3' of the first for loop, and the other points to the number '3' in the condition 'j < 3' of the second for loop.

Mas essa técnica funciona com matrizes que tenham quantidade de colunas diferentes por linha?

Exibindo uma matriz

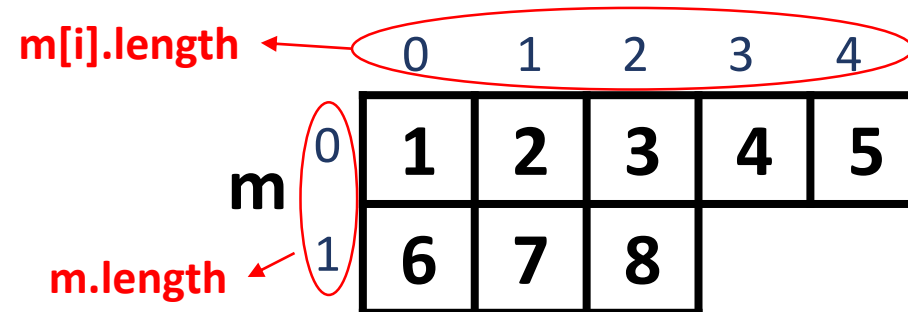
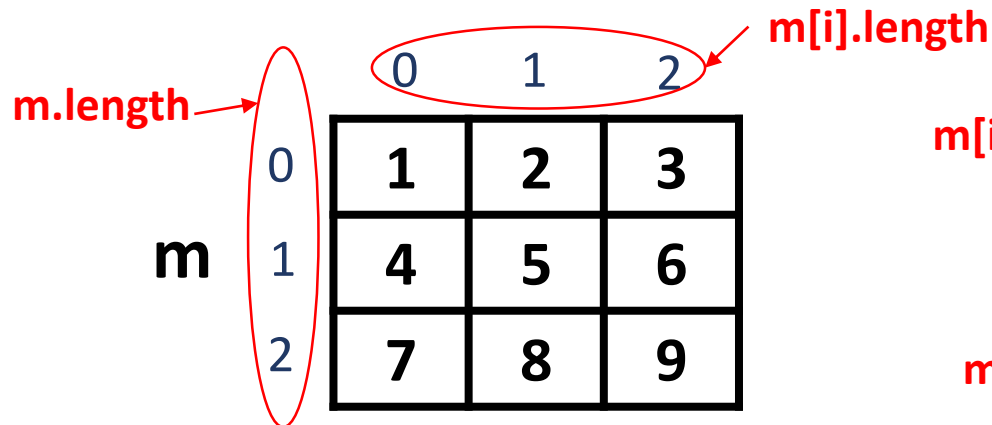
```
int m[][] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };  
\\ ou int m[][] = { {1, 2, 3, 4, 5}, {6, 7, 8} };
```

```
int i, j;
```

m.length retorna a quantidade de linhas

```
for (i = 0; i < m.length; i++) {  
    System.out.println( (i+1) + "a. linha: " );  
    for (j = 0; j < m[i].length; j++) {  
        System.out.println(m[i][j]);  
    }  
}
```

m[i].length determina o número de colunas da i-ésima linha



Resumindo...

- Se a matriz for irregular (pode haver linhas com números diferentes de colunas), nossa única opção é utilizar o **atributo length**.
 - Única maneira de andar em toda a matriz de maneira uniforme
- Se a matriz for regular (todas as linhas possuem o mesmo número de colunas) podemos explorar as dimensões com o **atributo length** ou **explicitamente**.

Preenchendo uma matriz

```
Scanner entrada = new Scanner(System.in);

int m[][] = new int[2][4];
int i, j;

for (i=0; i < 2; i++) {
    System.out.println("Informe a " + (i+1) + "a. linha:");
    for (j=0; j < 4; j++) {
        System.out.println("m[" + i + "][" + j + "] = ");
        m[i][j] = entrada.nextInt();
    }
}
```

Exercícios de fixação

1. Faça um programa que preencha uma matriz 3 x 5 de valores inteiros. Depois de preenchida a matriz, mostrar a soma de todos os valores e as quantidades de números pares e ímpares.
2. Faça um programa preencha uma matriz 8 x 4 e mostre apenas as linhas que possuem os índices das linhas ímpares.

...

Exercício (desafio)

3. Faça um programa que preencha uma matriz 4 x 4 de valores inteiros. Após o preenchimento imprima na tela os valores que formam a diagonal principal.

	0	1	2	3
0				
1				
2				
3				

- Há alguma maneira de fazer isso **usando laços de repetição aninhados**?

Dúvidas!?

humberto.zanetti@fatec.sp.gov.br