Vetores

Definição de vetor:

Vetor também é conhecido como variável composta homogênea unidimensional. Isto quer dizer que se trata de um conjunto de variáveis de mesmo tipo, que possuem o mesmo identificador (nome) e são alocadas sequencialmente na memória. Como as variáveis têm o mesmo nome, o que as distingue é um índice que referência sua localização dentro da estrutura.

Declaração de vetor em Java:

Os vetores em Java são definidos pela existência de colchetes vazios antes ou depois do nome da variável no momento da declaração. Logo depois, deve ser feito o dimensionamento do vetor.

```
tipo_da_variavel [] nome_variavel = new tipo_da_variavel[dimensionamento];
```

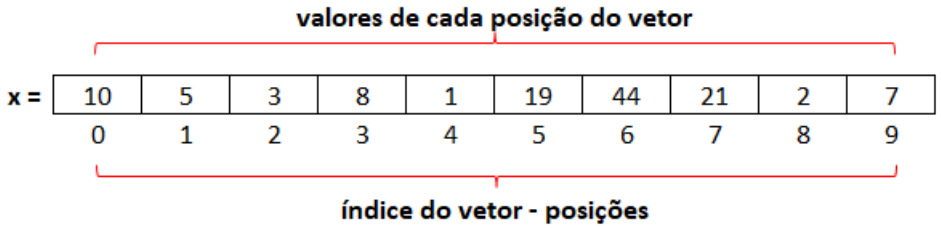
Exemplo de vetor:

Nos exemplos a seguir são utilizadas duas linhas de comando:

- a primeira declara um vetor e,
- a segunda define o seu tamanho.

Exemplo 1: Na primeira linha deste exemplo, os colchetes vazios após o nome definem que **x** será um vetor. O tipo int determina que todas as suas posições armazenarão valores inteiros. A segunda linha determina que o vetor **x** terá tamanho 10, ou seja, das posições de 0 a 9.

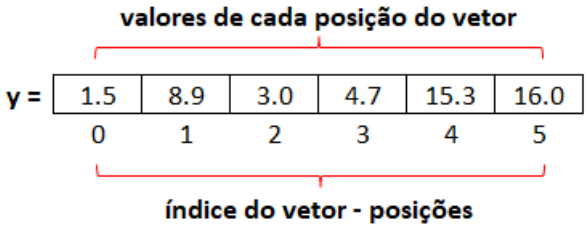
```
int x[];
x = new int[10];
```



Exemplo 2: Na primeira linha deste exemplo, os colchetes vazios antes do nome definem que **y** será um vetor. O tipo float determina o tipo do número que poderá ser armazenado em todas as suas posições. A segunda linha determina que o vetor **y** terá tamanho 6, ou seja, das posições de 0 a 5.

float [] y;

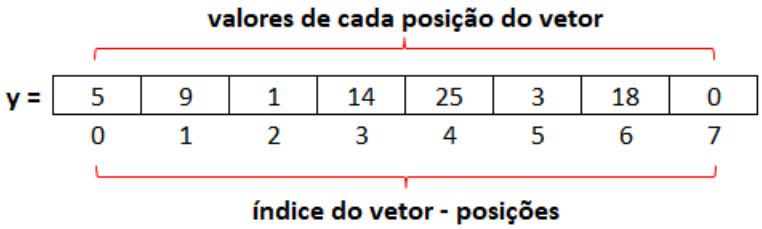
y = new float[6];



Já nos exemplos apresentados a seguir, utilizou-se a forma condensada, onde a declaração e o dimensionamento de um vetor são feitos utilizando-se uma única linha.

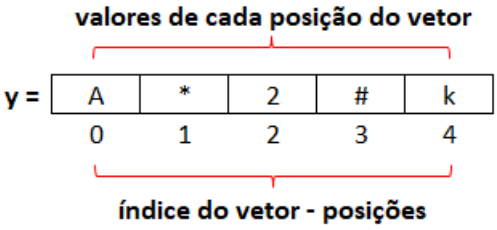
Exemplo 3: Este exemplo define e dimensiona o vetor **y** utilizando uma única linha. Assim, a parte que antecede o sinal de igual informa que **y** é um vetor e que poderá armazenar números inteiros. A parte que sucede o sinal de igual dimensiona o tamanho de **y** em 8 das posições de 0 a 7.

int y[] = new int[8];



Exemplo 4: Este exemplo define e dimensiona o vetor **y** utilizando uma única linha. Assim, a parte que antecede o sinal de igual informa que **y** é um vetor e que poderá armazenar qualquer caractere. A parte que sucede o sinal de igual dimensiona o tamanho de **y** em 5 das posições de 0 a 4.

char [] y new char[5];



Atribuindo valores ao vetor:

As atribuições em vetor exigem que seja informada em qual de suas posições o valor ficará armazenado. Deve-se lembrar sempre que a primeira posição de um vetor em Java tem índice 0.

vet[0] = 1; → atribui o valor 1 à primeira posição do vetor (lembre-se que o vetor começa na posição 0).

x[3] = 'b'; → atribui a letra b à quarta posição do vetor (lembre-se que o vetor começa na posição 0).

Preenchendo um vetor:

Preencher um vetor significa atribuir valores a todas as suas posições. Assim, deve-se implementar um mecanismo que controle o valor de índice.

```
Scanner scanner = new Scanner(System.in);
int [] vet = new int[10];
int i;

    for (i = 0; i < 10; i++) {
        vet[i] = scanner.nextInt();
    }
```

Nesse exemplo, a estrutura de repetição **for** foi utilizada para garantir que a variável **i** assuma todos os valores possíveis para o índice do vetor de 0 a 9. Assim, para cada execução da repetição, uma posição diferente do vetor será utilizada.

Mostrando os elementos do vetor:

Mostrar os valores contidos em um vetor também implica a utilização do índice.

```
for (i = 0; i < 10; i++) {
    System.out.println(vet[i]);
}
```

Nesse exemplo, a estrutura de repetição **for** foi utilizada para garantir que a variável **i** assuma todos os valores possíveis para o índice do vetor de 0 a 9. Assim, para cada execução da repetição, será utilizada uma posição diferente e, dessa forma, todos os valores do vetor serão mostrados.

```
/*
   Faça um programa que preencha um vetor com nove números, calcule e mostre
   os números impares e suas respectivas posições.

*/
```

```
package com.mycompany.vetorexemplo1;

import java.util.InputMismatchException;
import java.util.Scanner;

public class VetorExemplo1 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int [] vet = new int[9];
        int i;

        // atribuindo os valores ao vetor
        for(i = 0; i < 9; i++) {

            try {

                System.out.print("Digite o valor do vetor: ");
                vet[i] = sc.nextInt();

            } catch(InputMismatchException e) {
                System.out.println("Erro de digitação!");
                break;
            }

        }

        // apresentar os números impares do vetor e suas posições
        for (i = 0; i < 9; i++) {

            if (vet[i] % 2 != 0) {
                System.out.println("Posição: " + i + " Valor: " + vet[i]);
            }

        }
    }
}
```

Saída:

```
Digite o valor do vetor: 3
Digite o valor do vetor: 6
Digite o valor do vetor: 9
Digite o valor do vetor: 8
Digite o valor do vetor: 4
Digite o valor do vetor: 5
Digite o valor do vetor: 2
Digite o valor do vetor: 0
Digite o valor do vetor: 7
Posição: 0 Valor: 3
Posição: 2 Valor: 9
Posição: 5 Valor: 5
Posição: 8 Valor: 7
```

Matriz

Definição de matriz:

Uma matriz pode ser definida como um conjunto de variáveis de mesmo tipo e identificadas pelo mesmo nome. Essas variáveis são diferenciadas por meio da especificação de suas posições dentro dessa estrutura.

A linguagem Java permite a declaração de matrizes unidimensionais (mais conhecidas como vetores – descritos anteriormente), **bidimensionais e multidimensionais**. As matrizes mais utilizadas possuem duas dimensões. Para cada dimensão deve ser adotado um índice.

Declaração de matriz:

Matrizes em Java são definidas pela existência de colchetes vazios antes ou depois do nome da variável no momento da declaração. Logo depois, deve ser feita a definição do tamanho de cada dimensão da matriz.

Para utilizar uma matriz em Java, é necessário seguir dois passos:

1º Passo: DECLARAR UMA VARIÁVEL QUE FARÁ REFERÊNCIA AOS ELEMENTOS.

tipo_de_dados nome_variável [][][] ... []; → Os colchetes vazios após o nome da variável definem que a variável será uma estrutura multidimensional.

2º Passo: DEFINIR O TAMANHO DAS DIMENSÕES DA MATRIZ

nome_variavel = new tipo_de_dados[dimensão1][dimensão2][dimensão3] ... [dimensão];

onde:

- ➔ tipo_de_dados é o tipo de dados que poderá ser armazenado na sequência das variáveis que formam a matriz;
- ➔ nome_variavel é o nome da do à variável do tipo matriz;
- ➔ [dimensão1] representa o tamanho da primeira dimensão da matriz;
- ➔ [dimensão2] representa o tamanho da segunda dimensão da matriz;
- ➔ [dimensão3] representa o tamanho da n-ésima dimensão da matriz.

Exemplo de matriz:

Da mesma maneira como ocorre com os vetores, os índices de uma matriz começam sempre em 0 (zero) e podem variar até o tamanho da dimensão menos uma unidade.

É importante ressaltar que, em Java, os pares de colchetes podem aparecer todos antes do nome da variável ou todos depois do nome da variável ou, ainda, alguns antes e outros depois. Assim, todos os exemplos a seguir são validos.

Exemplo 1:

float x[][];

x = new float[2][6];

A declaração anterior criou uma variável chamada **x** contendo duas linhas de 0 a 1 com seis colunas cada de 0 a 5, capazes de armazenar números reais, como pode ser observado a seguir:

variável x

índice linhas da matriz							
	0	1.5	3	0.78	0.3	1.2	9
	1	1	0.9	4.1	4	2.5	32.2
		0	1	2	3	4	5

índice ou posições da matriz

Exemplo 2:

char [][] mat;

mat = new char[4][3];

A declaração anterior criou uma variável chamada **mat** contendo quatro linhas de 0 a 3 com três colunas cada de 0 a 2, capazes de armazenar símbolos, como pode ser observado a seguir.

variável x

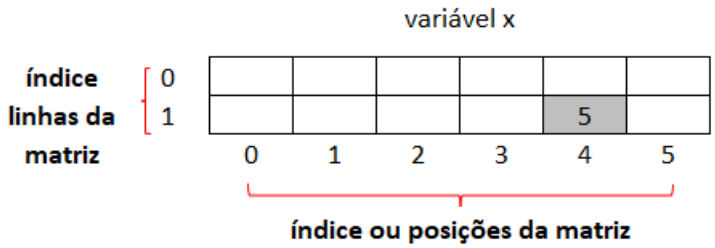
índice linhas da matriz							
	0	A	3	1	0.3	1.2	9
	1	*	k	l	v	j	3
	2	&	s	n	4	t	4
	3	$	u	)	(	[	]
		0	1	2	3	4	5

índice ou posições da matriz

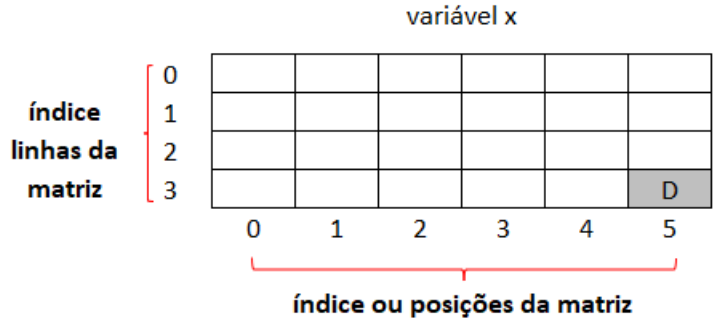
Atribuindo valores a uma matriz:

Atribuir valor a uma matriz significa armazenar uma informação em um de seus elementos, identificando de forma única por meio de seus índices.

x[1][4] = 5; → atribui o valor 5 à posição identificada pelos índices 1 (2º linha) e 4 (5º coluna).



x[3][2] = 'D' → Atribui a letra D à posição identificada pelos índices 3 (4º linha) e 2(3º coluna).



Preenchendo uma matriz:

Preencher uma matriz significa percorrer todos os seus elementos, atribuindo-lhes um valor. Este valor pode ser recebido do usuário, por meio do teclado, ou pode ser gerado pelo programa.

No exemplo a seguir, todos os elementos de uma matriz bidimensional são percorridos, atribuindo-lhes valores inteiros digitados pelo usuário e capturados pelo método **nextInt()** da classe Scanner.

```
package com.mycompany.matrizexemplo1;

import java.util.InputMismatchException;
import java.util.Scanner;

public class MatrizExemplo1 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[][] mtz = new int[7][3];
        int i, j;

        try {

            for (i = 0; i < 7; i++) {
                for (j = 0; j < 3; j++) {
                    System.out.print("Digite o valor para a matriz: ");
                    mtz[i][j] = sc.nextInt();
                }
            }

        } catch(InputMismatchException e) {
            System.out.println("Erro de digitação!");
        }

    }
}
```

Como a matriz possui sete linhas e três colunas, o **for** externo deve variar de 0 a 6 (percorrendo, assim, as sete linhas da matriz) e o **for** interno deve variar de 0 a 2 (percorrendo, assim, as três colunas da matriz).

Mostrando os elementos de uma matriz:

Pode-se, também, percorrer todos os elementos da matriz, acessando o seu conteúdo. No exemplo que se segue, são mostrados todos os elementos de uma matriz contendo dez linhas e seis colunas. Observe que são usados dois índices, **i** e **j**. Estes índices estão atrelados a estruturas de repetição que mantêm a variação de ambos dentro dos intervalos permitidos, ou seja, o índice **i**, que representa as linhas, varia de 0 a 9 e o índice **j**, que representa as colunas, varia de 0 a 5.

```
for (i = 0; i < 10; i++) {
    for (j = 0; j < 6; j++) {
        System.out.print(mtz[i][j]);
    }
}
```

```
/*
    Faça um programa que preencha uma matriz M(2 x 2), calcule e mostre a matriz
    R, resultante da multiplicação dos elementos de M pelo seu maior elemento.
*/
```

```
package com.mycompany.matriz;

import java.util.InputMismatchException;
import java.util.Scanner;

public class Matriz {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[][] m = new int[2][2];
        int[][] r = new int[2][2];
        int i, j, maior;

        try {

            // preenchendo a matriz com a digitação pelo usuario
            for (i = 0; i < 2; i++) {
                for (j = 0; j < 2; j++) {
                    System.out.print("Digite o valor da matriz: ");
                    m[i][j] = sc.nextInt();
                }
            }

            // buscando o maior valor da matriz digitada pelo usuario
            maior = 0;
            for (i = 0; i < 2; i++) {
                for (j = 0; j < 2; j++) {
                    if (m[i][j] > maior)
                        maior = m[i][j];
                }
            }

            // multiplicando a matriz m pelo maior valor
            // e colocado na matriz R
            for (i = 0; i < 2; i++) {
                for (j = 0; j < 2; j++) {
                    r[i][j] = m[i][j] * maior;
                }
            }


            // mostrando a matriz r, matriz resultante
            for (i = 0; i < 2; i++) {
                for (j = 0; j < 2; j++) {
                    System.out.print(r[i][j] + " ");
                }
                System.out.println(" ");
            }

        } catch(InputMismatchException e) {

            System.out.println("Erro de digitação!");

        }

    }
}
```

Saida:

```
Digite o valor da matriz: 1
Digite o valor da matriz: 5
Digite o valor da matriz: 9
Digite o valor da matriz: 7
9 45
81 63
```