



# ARRAY

# FUNDAMENTOS DE PROGRAMAÇÃO

# INTRODUÇÃO

**Array** é uma coleção de variáveis do mesmo tipo, referidas por um nome comum. Em Java, os arrays podem ter uma ou mais dimensões, embora o array unidimensional seja o mais comum.



# INTRODUÇÃO

**Array** são usadas para diversos propósitos porque oferecem um meio conveniente de agrupar variáveis relacionadas. Por exemplo, você pode usar um array para manter um registro da temperatura máxima diária durante um mês, uma lista de médias de preços de ações ou uma lista de sua coleção de livros de programação.



## ✓ Vantagem

A principal vantagem de um array é que ele organiza os dados de forma que possam ser facilmente manipulados.

Por exemplo, se você tiver uma matriz contendo os rendimentos de um grupo selecionado de famílias, será fácil calcular a renda média percorrendo a matriz.

Além disso, os arrays organizam os dados de forma que possam ser facilmente classificados.

## ✓ **Matrizes unidimensionais**

Uma matriz unidimensional é uma lista de variáveis relacionadas.

Essas listas são comuns em programação. Por exemplo, você pode usar uma matriz unidimensional para armazenar os números de contas dos usuários ativos em uma rede.

Outra matriz pode ser usada para armazenar as médias de rebatidas atuais de um time de beisebol.



## ✓ Declarar

Para declarar um array unidimensional, você pode usar esta forma geral:

```
type array-name[ ] = new type[size];
```

Aqui, type declara o tipo de elemento do array. (O tipo de elemento também é comumente chamado de tipo base.)

O tipo de elemento determina o tipo de dados de cada elemento contido na matriz.

O número de elementos que a matriz conterá é determinado pelo tamanho.

Como os arrays são implementados como objetos, a criação de um array é um processo de duas etapas.

Primeiro, você declara uma variável de referência de array. Segundo, você aloca memória para o array, atribuindo uma referência a essa memória para a variável do array.

Assim, os arrays em Java são alocados dinamicamente usando o operador new.

# EXEMPLO

Aqui está um exemplo. O seguinte cria um array int de 10 elementos e o vincula a uma variável de referência de array chamada sample:

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

Esta declaração funciona como uma declaração de objeto.  
A variável de amostra contém uma referência à memória alocada por new.

Esta memória é grande o suficiente para armazenar 10 elementos do tipo int.

Tal como acontece com os objetos, é possível quebrar a declaração anterior em duas. Por exemplo:

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

# EXEMPLO

Neste caso, quando a amostra é criada pela primeira vez, ela não se refere a nenhum objeto físico. Somente após a segunda instrução ser executada é que `sample` é vinculado a um array.

Um elemento individual dentro de uma matriz é acessado pelo uso de um índice.

Um índice descreve a posição de um elemento dentro de uma matriz.

Em Java, todos os arrays têm zero como índice de seu primeiro elemento.

Como `sample` possui 10 elementos, ele possui valores de índice de 0 a 9.

Para indexar uma matriz, especifique o número do elemento desejado, entre colchetes.

Assim, o primeiro elemento na amostra é `amostra[0]` e o último elemento é `amostra[9]`.

Por exemplo, o programa a seguir carrega uma amostra com os números de 0 a 9:



# EXEMPLO

```
// Demonstrate a one-dimensional array.
```

```
class ArrayDemo {  
    public static void main(String args[]) {  
        int sample[] = new int[10];  
        int i;
```

```
        for(i = 0; i < 10; i = i+1) ←  
            sample[i] = i;
```

Arrays are indexed from zero.

```
        for(i = 0; i < 10; i = i+1) ←  
            System.out.println("This is sample[" + i + "]: " +  
                               sample[i]);
```

```
    }  
}
```

# EXEMPLO

Conceitualmente, o array de amostra se parece com isto:

0	1	2	3	4	5	6	7	8	9
Sample [0]	Sample [1]	Sample [2]	Sample [3]	Sample [4]	Sample [5]	Sample [6]	Sample [7]	Sample [8]	Sample [9]

## ✓ Inicializar

Arrays podem ser inicializados quando são criados. A forma geral para inicializar um array unidimensional é mostrada aqui:

```
type array-name[ ] = { val1, val2, val3, ... , valN };
```

Aqui, os valores iniciais são especificados por val1 até valN.

Eles são atribuídos em sequência, da esquerda para a direita, na ordem do índice.

Java aloca automaticamente um array grande o suficiente para conter os inicializadores especificados.

Não há necessidade de usar explicitamente o operador new. Por exemplo, aqui está uma maneira melhor de escrever o programa MinMax:

# ARRAY

```
// Use array initializers.
class MinMax2 {
    public static void main(String args[]) {
        int nums[] = { 99, -10, 100123, 18, -978,
                       5623, 463, -9, 287, 49 };
        int min, max;

        min = max = nums[0];
        for(int i=1; i < 10; i++) {
            if(nums[i] < min) min = nums[i];
            if(nums[i] > max) max = nums[i];
        }
        System.out.println("Min and max: " + min + " " + max);
    }
}
```

← Array initializers

# ARRAY

Os limites do array são estritamente aplicados em Java; é um erro de tempo de execução ultrapassar ou ultrapassar o final de uma matriz. Se você quiser confirmar isso por si mesmo, tente o seguinte programa que ultrapassa propositalmente um array:

```
// Demonstrate an array overrun.
class ArrayErr {
    public static void main(String args[]) {
        int sample[] = new int[10];
        int i;

        // generate an array overrun
        for(i = 0; i < 100; i = i+1)
            sample[i] = i;
    }
}
```

Assim que chegar a 10, um `ArrayIndexOutOfBoundsException` será gerado e o programa será encerrado.

# MATRIZ MULTIDIMENSIONAL

Embora o array unidimensional seja frequentemente o array mais comumente usado em programação, os arrays multidimensionais (matrizes de duas ou mais dimensões) certamente não são raros. Em Java, um array multidimensional é um array de arrays.

## ✓ Matrizes bidimensionais

A forma mais simples de matriz multidimensional é a matriz bidimensional. Uma matriz bidimensional é, em essência, uma lista de matrizes unidimensionais.

Para declarar uma tabela de array inteiro bidimensional de tamanho 10, 20 você escreveria

```
int table[] [] = new int[10][20];
```



# MATRIZ MULTIDIMENSIONAL

## ✓ **Matrizes bidimensionais**

Preste muita atenção à declaração.

Ao contrário de algumas outras linguagens de computador, que usam vírgulas para separar as dimensões do array, Java coloca cada dimensão em seu próprio conjunto de colchetes.

Da mesma forma, para acessar o ponto 3, 5 da tabela de array, você usaria `table[3][5]`.

# MATRIZ MULTIDIMENSIONAL

## ✓ Matrizes bidimensionais

No próximo exemplo, uma matriz bidimensional é carregada com os números de 1 a 12.

```
// Demonstrate a two-dimensional array.
class TwoD {
    public static void main(String args[]) {
        int t, i;
        int table[][] = new int[3][4];

        for(t=0; t < 3; ++t) {
            for(i=0; i < 4; ++i) {
                table[t][i] = (t*4)+i+1;
                System.out.print(table[t][i] + " ");
            }
            System.out.println();
        }
    }
}
```

# MATRIZ MULTIDIMENSIONAL

## ✓ Matrizes bidimensionais

Neste exemplo, `table[0][0]` terá o valor 1, `table[0][1]` o valor 2, `table[0][2]` o valor 3 e assim por diante. O valor de `table[2][3]` será 12. Conceitualmente, o array será semelhante ao mostrado na Figura a seguir:

	0	1	2	3 ← right index
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

left index ↑

↑ `table[1][2]`

# THANK YOU!

## CONTACT US AT:



## Nelson Note



**nnote@ucm.ac.mz**



828848766

