Laboratório de Programação

Array Unidimensional (vetor)

Prof. Dr. Antonio Marcos SELMINI
selmini@fiap.com.br

Introdução

Considere o seguinte problema: escrever um programa em Java para ler e *armazenar* 10 números inteiros. O seu programa deverá calcular e imprimir a soma dos números no vídeo.

```
import java.util.Scanner;
public class Exemplo {
     public static void main(String[] args) {
          int n1, n2, n3, n4, n5;
          int n6, n7, n8, n9, n10, soma;
          Scanner entrada = new Scanner(System.in);
          System.out.println("Digite o primeiro valor");
          n1 = entrada.nextInt();
          System.out.println("Digite o segundo valor");
          n2 = entrada.nextInt();
          //repete todo o processo para as outras variáveis
          soma = n1+n2+n3+n4+n5+n6+n7+n8+n9+n10;
          System.out.println("Soma dos números = "+soma);
     }
}
```

declaração de 10 variáveis para entrada de dados!!!

muita repetição de código

inviável para uma quantidade grande de variáveis

Introdução

Considere a seguinte situação prática:

Um professor tem um total de 50 alunos. O professor gostaria de armazenar para processamento as duas notas referentes às provas aplicadas, e também calcular e armazenar a média de cada aluno

Qual o total de variáveis para resolver o problema descrito?

- Para quem pensou em 150 variáveis acertou!
- Declarar 150 variáveis parece algo muito complicado e também nada prático!!

Introdução

Qual a solução para esse problema??

Array Unidimensional (Vetor)

Array Unidimensional (Vetor)

Definição:

Um ***array unidimensional*** (ou ***vetor***) é definido formalmente como um ***conjunto de variáveis***, ou seja, um conjunto de posições de memória contínuas que são ***referenciadas pelo mesmo nome***

Os elementos armazenados em um vetor são do mesmo tipo de dado e são referenciados pelo mesmo nome

Sintaxe completa para declaração de um vetor:

```
tipo[] nome_array = new tipo[tamanho];
                ou
tipo nome_array[] = new tipo[tamanho];
```

tamanho ⇒ valor inteiro e positivo

Array Unidimensional (Vetor)

Exemplos de declaração completa:

```
int[] numero = new int[10];
//ou int numero [] = new int[10];

double[] nota = new double[50];
//ou double nota [] = new double[50];

String[] nome = new String[50];
//ou String nome [] = new String[50];
```

Array Unidimensional (Vetor)

A declaração de um vetor pode ser feita em duas etapas: `declaração` e `instanciação`. Sintaxe para declaração:

```
tipo[] nome_array;
ou
tipo nome_array[];
```

Exemplo de declaração:

```
int[] x;
ou
int x[];
```

Array Unidimensional (Vetor)

Depois de declarar um vetor, é necessário instanciá-lo. A instanciação é o processo pelo qual um endereço de memória é alocado para um objeto. A sintaxe para a declaração de um vetor é:

```
nome_array = new tipo[tamanho];
```

Exemplo de instanciação:

```
x = new int[40];
```

não usa colchetes

obrigatório definir o tamanho do array!!

Muito cuidado com essa linha de código!!!

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

Não compila!!! falta o tamanho do array!

Array Unidimensional (Vetor)

A partir do momento que um vetor foi definido, como podemos acessar os seus elementos? Como podemos adicionar novos elementos a um vetor?

Um elemento individual dentro de um vetor é acessado pelo uso de um *índice* ou *subscrito*

Um *índice* descreve a posição de um elemento dentro do vetor *Em Java, o índice do primeiro elemento é o número zero. Os índices são valores inteiros e positivos sequenciais*.

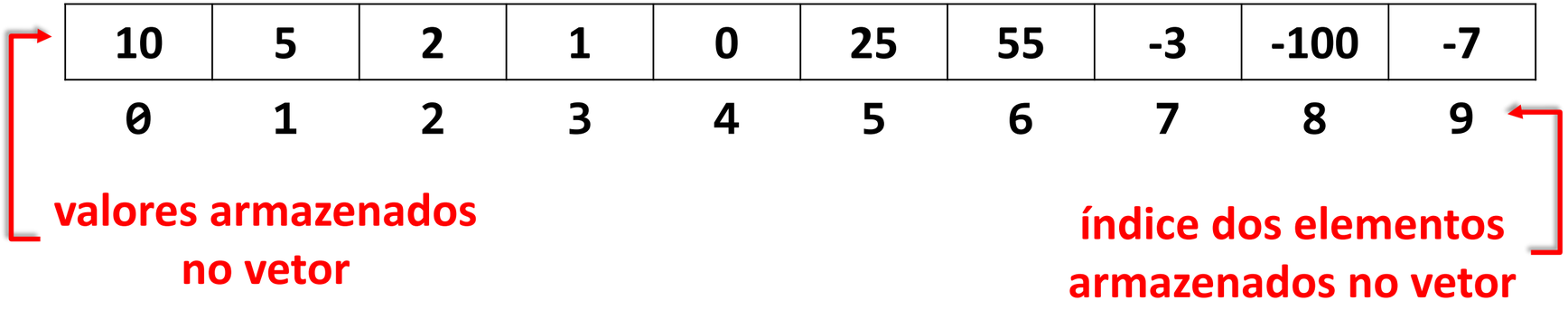
Prof. Dr. Antonio Marcos SELMINI – selmini@fiap.com.br

Array Unidimensional (Vetor)

Como exemplo, considere a seguinte declaração de um vetor:

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

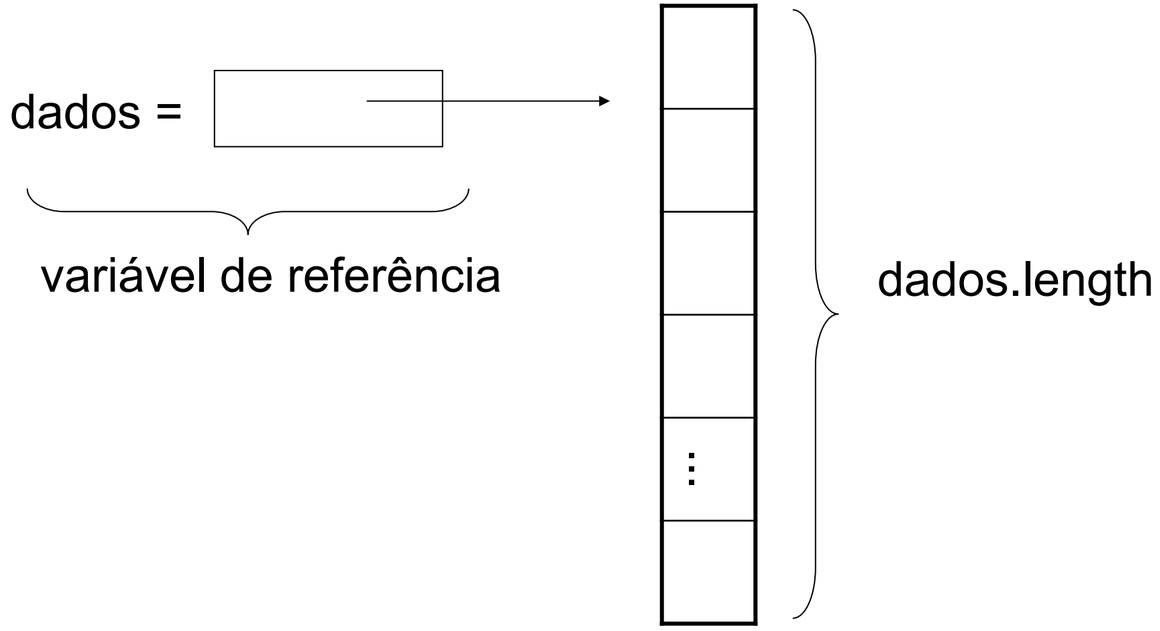
Conceitualmente podemos imaginar o vetor da seguinte forma:



Array Unidimensional (Vetor)

Exemplo:

```
double[] dados = new double[10];
```



A variável *dados* é uma *referência* a um array de números *double*

Os arrays têm comprimento fixo e não pode ser alterado

Os arrays têm elementos de um tipo específico

Observação:

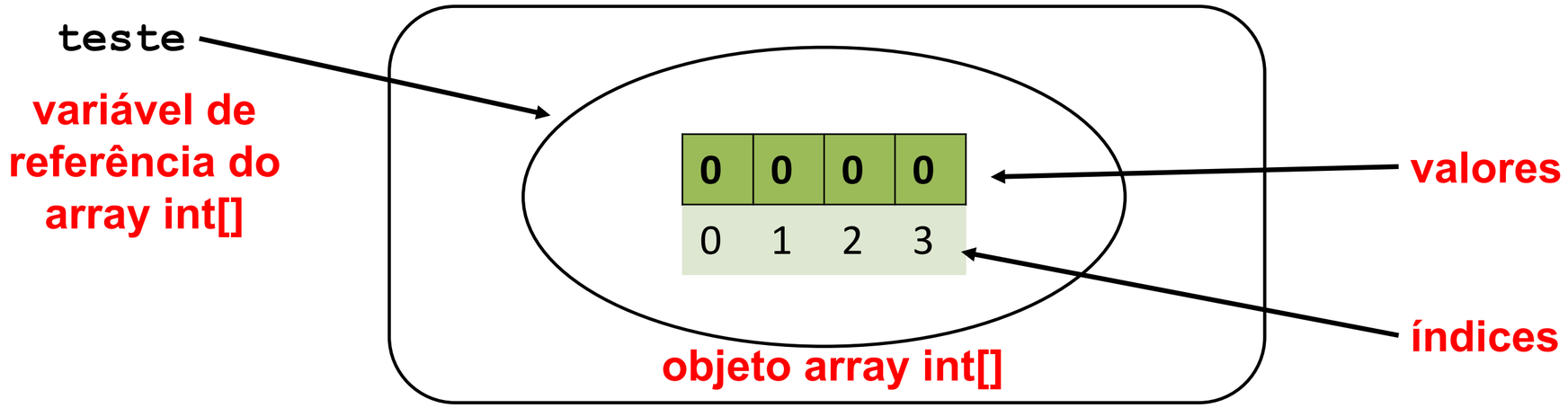
- *length não é um método (comando);*
- *length é uma propriedade de todo objeto array;*
- *length determina o tamanho do array;*

Representação gráfica na memória

```
int[] teste = new int[4];
```

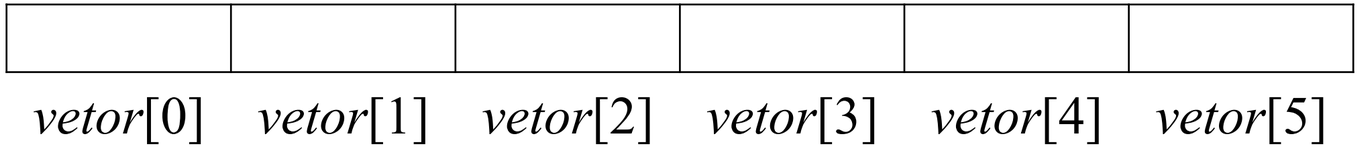
mesmo efeito!!

```
int[] teste;
teste = new int[4];
```

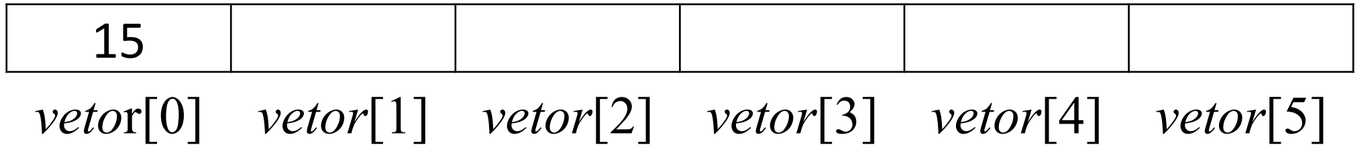


Manipulando array unidimensional

Considere a seguinte declaração:
`int[] vetor = new int[6]`. **Dessa forma temos:**



Armazenar o valor 15 na primeira posição do vetor:
`vetor[0] = 15`; **a primeira posição do vetor tem índice 0 e não 1**

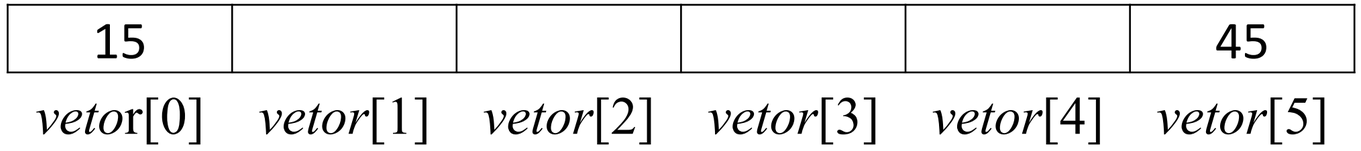


Manipulando array unidimensional

Armazenar na última posição do vetor o triplo do valor da primeira posição:

```
vetor[5] = vetor[0]*3;
```
a última posição tem índice 5

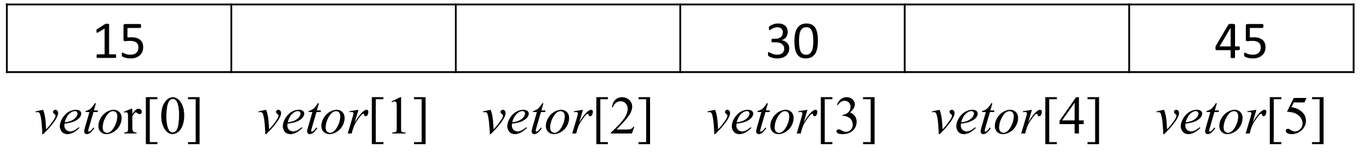
```
vetor[vetor.length-1] = vetor[0]*3;
```
tem o mesmo efeito da linha anterior



Armazenar na quarta posição do vetor a diferença entre o valor da última posição e da primeira:

```
vetor[3] = vetor[5]- vetor[0];
```
a quarta posição tem índice 3

```
vetor[3] = vetor[vetor.length-1]- vetor[0];
```
tem o mesmo efeito da linha anterior



Manipulando array unidimensional

Como obter um valor armazenado?

```
nome_do_array[indice]     // acessa o elemento em indice
```

```
int[] vetor = new int[7];
```

índice	0	1	2	3	4	5	6
vetor	10	11	87	34	46	65	16

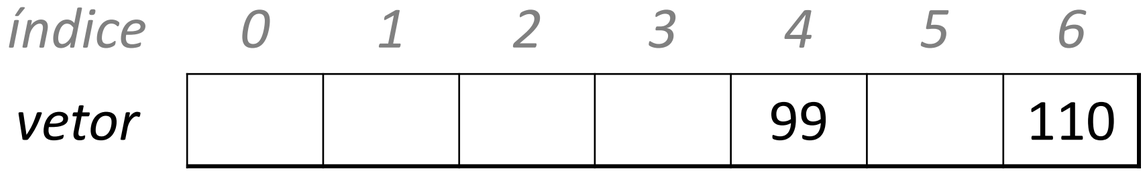
```
System.out.println(vetor[2]);
System.out.println(vetor[-1]); //erro!! Não existe índice negativo
System.out.println(vetor[7]); //erro!! O maior índice é 6 e não 7
```

Manipulando array unidimensional

Como armazenar um valor em um array?

```
nome_do_array[indice] = valor;
```

```
int[] vetor = new int[7];
```




```
vetor[-1] = 18; //erro! o índice -1 não existe
vetor[7] = 100; //erro! o índice 7 não existe
vetor[4] = 99; //não é a posição 4 e, sim, a posição de índice 4 (5ª posição)
vetor[6] = 110; //posição de índice 6 (7ª posição)
```

Manipulando array unidimensional

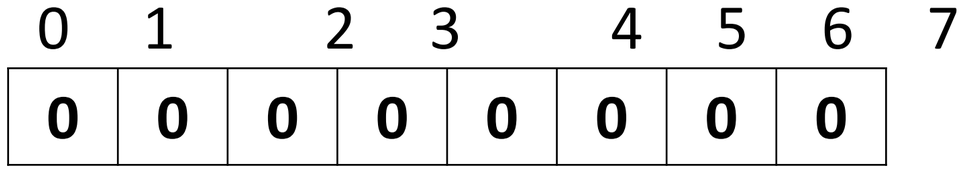
Qual o conteúdo do array após a execução do código abaixo?

```
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;

int x = numbers[1];
numbers[x] = 2;
numbers[numbers[4]] = 9;
```

Manipulando array unidimensional

Qual o conteúdo do array após a execução do código abaixo?

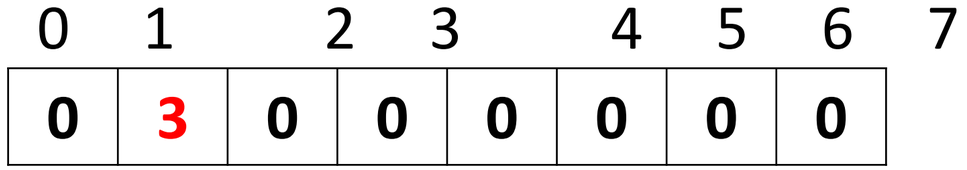
```
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;

int x = numbers[1];
numbers[x] = 2;
numbers[numbers[4]] = 9;
```



Manipulando array unidimensional

Qual o conteúdo do array após a execução do código abaixo?

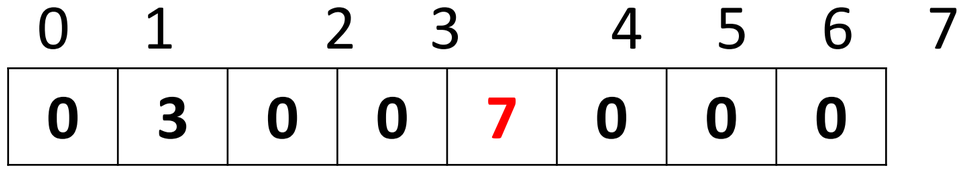
```
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;

int x = numbers[1];
numbers[x] = 2;
numbers[numbers[4]] = 9;
```



Manipulando array unidimensional

Qual o conteúdo do array após a execução do código abaixo?

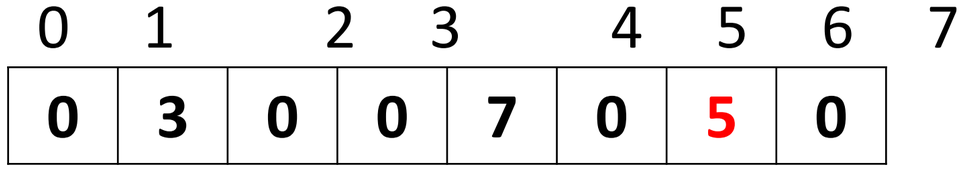
```
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;

int x = numbers[1];
numbers[x] = 2;
numbers[numbers[4]] = 9;
```



Manipulando array unidimensional

Qual o conteúdo do array após a execução do código abaixo?

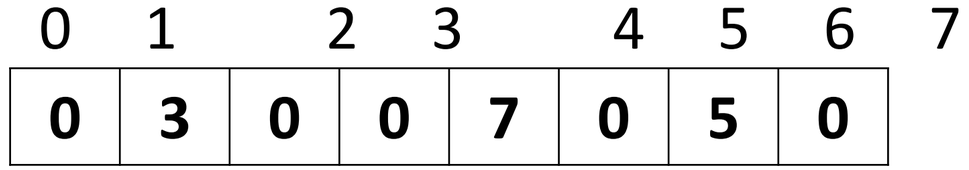
```
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;

int x = numbers[1];
numbers[x] = 2;
numbers[numbers[4]] = 9;
```



Manipulando array unidimensional

Qual o conteúdo do array após a execução do código abaixo?

```
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;

int x = numbers[1];
numbers[x] = 2;
numbers[numbers[4]] = 9;
```

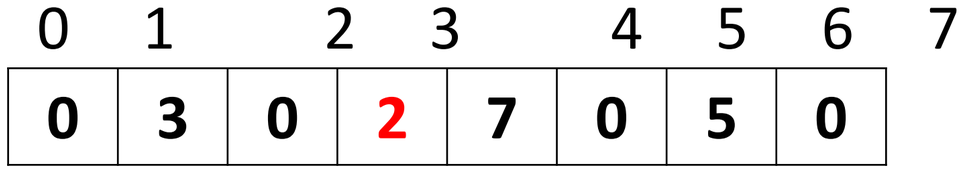


x **3**

Manipulando array unidimensional

Qual o conteúdo do array após a execução do código abaixo?

```
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;

int x = numbers[1];
numbers[x] = 2;
numbers[numbers[4]] = 9;
```

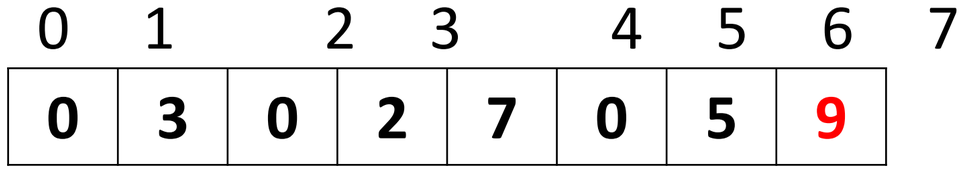


x | 3

Manipulando array unidimensional

Qual o conteúdo do array após a execução do código abaixo?

```
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;

int x = numbers[1];
numbers[x] = 2;
numbers[numbers[4]] = 9;
```



x | 3

Manipulando array unidimensional

Qual o tamanho de um array?

```
meuArray.length
```

Pontos importantes:

Qual o índice do último elemento do vetor em termos do seu tamanho *length*?

Manipulando array unidimensional

Qual o tamanho de um array?

***meuArray*.length**

Pontos importantes:

Qual o índice do último elemento do vetor em termos do seu tamanho *length*?

***meuArray*.length-1**

Qual o índice do elemento do meio do vetor em termos do seu tamanho *length*?

Manipulando array unidimensional

Qual o tamanho de um array?

***meuArray*.length**

Pontos importantes:

Qual o índice do último elemento do vetor em termos do seu tamanho *length*?

***meuArray*.length-1**

Qual o índice do elemento do meio do vetor em termos do seu tamanho *length*?

(*meuArray*.length-1)/2

Arrays ❤ laços de repetição

```
int[] numbers = new int[8];
for(int i = 0; i < numbers.length; i++) {
  numbers[i] = 2 * i;
}
```

índice	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
valor	0	2	4	6	8	10	12	14

Inicialização de arrays unidimensioanais

Declara e depois inicializa:

```
int[] primos = new int[5];
primos[0] = 2;
primos[1] = 3;
primos[2] = 5;
primos[3] = 7;
primos[4] = 11;
```

Inicialização no momento da declaração:

```
int[] primos = {2, 3, 5, 7, 11};
```

O tamanho do vetor será igual a quantidade de elementos informada

Exemplo 1

Programa em Java para preencher e imprimir um vetor

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

        //armazena os valores no vetor
        for(i = 0; i < 10; i++) {
            vetor[i] = i;
        }
        //imprime os valores do vetor
        for(i = 0; i < vetor.length; i++) {
            System.out.println(vetor[i]);
        }
    }
}
```

Exemplo 2

Programa em Java para armazenar o nome de 5 alunos

```
import java.util.Scanner;
public class Exemplo {
    public static void main(String[] args) {
        String[] nomes = new String[5];
        int i;
        Scanner entrada = new Scanner(System.in);

        //ler os nomes via teclado
        for(i = 0; i < nomes.length; i++) {
            System.out.println("Digite o nome");
            nomes[i] = entrada.nextLine();
        }
    }
}
```

Exercícios de programação

1. Escreva um programa em Java que preencha um vetor de 10 posições com valores fornecidos pelo usuário. Imprima no vídeo o maior e o menor valor armazenado. **Observação: maior valor inteiro é Integer.MAX_VALUE e o menor valor inteiro é Integer.MIN_VALUE**

2. Escreva um programa em Java que preencha um vetor de 10 posições com valores fornecidos pelo usuário. Imprima no vídeo a quantidade de números pares e ímpares digitados.

3. Escreva um programa que peça as quatro notas de 10 alunos, calcule e armazene num vetor a média de cada aluno, imprima o número de alunos com média maior ou igual a 6.0.

Exercícios de programação

4. Escreva um programa que receba a temperatura média de cada mês do ano passado e armazene-as em uma lista (vetor). Em seguida, calcule a média anual das temperaturas e mostre todas as temperaturas acima da média anual, e em que mês elas ocorreram (mostrar o mês por extenso: 1 – Janeiro, 2 – Fevereiro, . . .).

5. Escreva um programa que leia e armazene 10 números inteiros em um vetor. Em seguida, inverta os elementos inicialmente armazenados. Imprima no terminal os elementos antes e após a inversão. Por exemplo:

Array original	15	22	17	5	-1

Array invertido	-1	5	17	22	15

Exercícios de programação

6. Um array é considerado especial se cada par de seus elementos adjacentes contém dois números com paridades diferentes. Você deverá escrever um programa que leia uma quantidade de elementos e armazene em um array. Em seguida imprima uma mensagem no terminal informando se o array é ou não é especial. Por exemplo: [1] é um array especial; [2, 1, 4] tem dois pares (2, 1) e (1, 4) e é um array especial; [4, 3, 1, 6] não é um array especial porque dois números ímpares são adjacentes.

7. Dado um array não vazio de números inteiros não negativos, o grau deste array é definido como a frequência máxima de qualquer um de seus elementos. Escreva um programa que preencha um array de números inteiros e, em seguida, imprima o seu grau.

Geração de números aleatórios

Um ***número aleatório*** é um número que pertence a uma série numérica e não pode ser previsto a partir dos membros anteriores da série.

O conceito de número aleatório é um conceito relativo à série numérica a que o número pertence.

Números verdadeiramente aleatórios são muito difíceis de obter, por isso, devemos nos contentar com números ***pseudo*** aleatórios, gerados por algoritmos.

Geração de números aleatórios

Os números aleatórios são utilizados de diversas formas em programas de computador.

Eles são importantes no desenvolvimento de jogos, na área de segurança de informações (exemplo: para gerar senhas ou textos) e em programas de mineração de dados e análise estatística, apenas para citar alguns exemplos.

A linguagem Java disponibiliza funcionalidades para a geração de números aleatórios em uma classe denominada ***Random*** do pacote ***java.util***

Geração de números aleatórios

Inicialmente define-se uma variável do tipo *Random* usando a seguinte sintaxe:

```
Random nome_da_variavel = new Random();
```

Após a definição da variável do tipo ***Random***, métodos são utilizados para a geração de valores. Os principais métodos são:

- ***nextInt()***: gera números inteiros (positivos ou negativos) dentro da faixa de valores inteiros do Java;
- ***nextInt(n)***: gera um número inteiro entre 0 e *n*-1;
- ***nextDouble():*** gera números entre 0 e 1;
- ***nextFloat():*** gera números entre 0 e 1;

Geração de números aleatórios – exemplo 2

```
import java.util.Random;
    public class Aleatorio {
        public static void main(String[] args) {

            //definição da variável Random
            Random gerador = new Random();

            //imprime sequência de 10 números inteiros aleatórios
            //entre 0 e 25
            for (int i = 0; i < 10; i++) {
                System.out.println(gerador.nextInt(26));
            }
        }
}
```

Geração de números aleatórios – exemplo 1

```
import java.util.Random;
    public class Aleatorio {
        public static void main(String[] args) {

            //definição da variável Random
            Random gerador = new Random();

            //imprime sequência de 10 números inteiros aleatórios
            for (int i = 0; i < 10; i++) {
                System.out.println(gerador.nextInt());
            }
    }
}
```

Geração de números aleatórios

Suponha que você precise gerar um número aleatório entre 1 e 6. Se você simplesmente usar o método ***nextInt(6)***, obterá números entre 0 e 5, algo que você não deseja. Resolver a situação é simples, bastando aplicar o seguinte recurso:

```
int valor = gerador.nextInt(6) + 1;
```

Imagine que você precise gerar um número real qualquer entre 0 e 90, por exemplo. Os métodos ***nextDouble()*** e ***nextFloat()*** retornam sempre números entre 0 e 1. Então como podemos resolver a situação? A resposta é simples, basta multiplicar o número gerado por 90!

```
double valor = gerador.nextDouble() * 90;
```

Geração de números aleatórios – exemplo 3

```
import java.util.Random;
    public class Aleatorio {
        public static void main(String[] args) {

            //definição da variável Random
            Random r = new Random();

            System.out.println(r.nextDouble());
            System.out.println(r.nextFloat());
        }
}
```

Exercícios de programação

8. Escreva um programa em Java que preencha um vetor de 10 posições com números aleatórios inteiros não duplicados.

9. Escreva um programa em Java que leia e armazene o nome e o salário dos funcionários de uma empresa. Imprima no vídeo o total que a empresa gasta com a folha de pagamento (soma do salário de todos os funcionários) e a média salarial paga aos funcionários.

10. Escrever um programa em Java que preencha um vetor de 10 posições com valores aleatórios. Em seguida, colocar os valores do vetor em ordem crescente. Imprima no vídeo os elementos do vetor antes e após a ordenação.

Exercícios de programação

11. Escreva um programa que leia e armazene os nomes dos arquivos e os respectivos tamanhos em bytes. Imprima uma listagem contendo os nomes dos arquivos e seus respectivos tamanhos em megabyte e gigabyte.

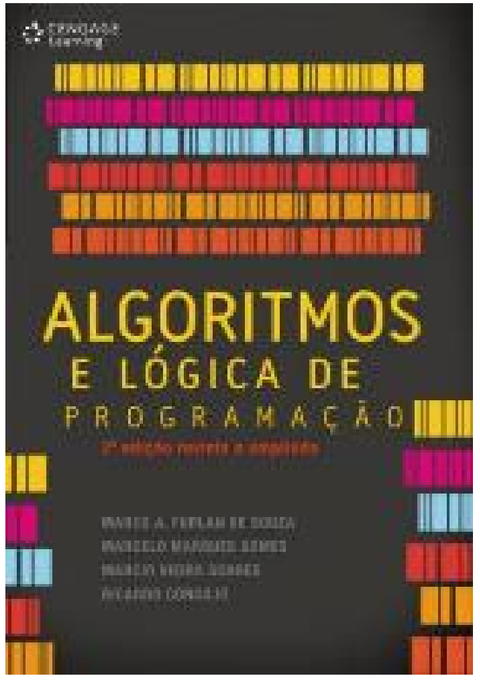
    1 Mebagyte (1MB) = 1.048.576 (2^{20} bytes) = 1024 * 1024

    1 Gigabyte (1GB) = 1.073.741.824 (2^{30} bytes) = 1024 * 1024 * 1024

Bibliografia

XAVIER, G. F. C. **Lógica de Programação**. 12ª ed. Editora Senac São Paulo, 2011.

SOUZA, M. A. F.; SOARES, M. V.; GOMES, M. M.; CONCILIO, R. **Algoritmos e Lógica de Programação**. 1ª e 2ª ed. São Paulo: Cengage Learning, 2011.

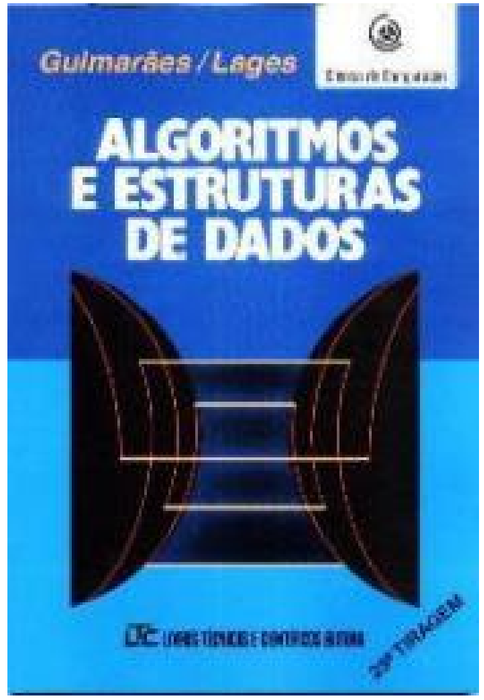


Bibliografia

PUGA, S., RISSETTI, G. **Lógica de programação e estrutura de dados com aplicações em Java**. 2ª ed. São Paulo: Editora Pearson, 2013.

FORBELLONE, A. L., EBERSPACHER, H. F. **Lógica de programação – A construção de algoritmos e estruturas de dados**. 3ª ed. São Paulo: Editora Pearson, 2005.

Bibliografia



GUIMARÃES, A. de M. **Algoritmos e estruturas de dados.** Rio de Janeiro: Editora LTC, 1994.

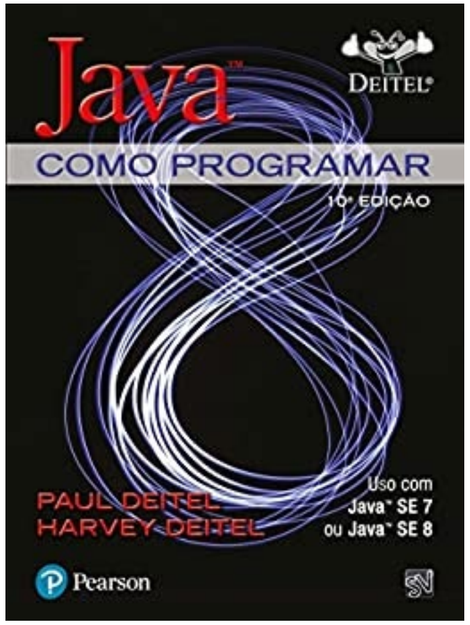
ASCENCIO, A. F. G.; CAMPOS, E. A. V. **Fundamentos da Programação de Computadores – Algoritmos em Pascal, C++ e Java.** 3ª ed. Editora Pearson, 2013.

Bibliografia

ARNOLD, K., GOSLING, J., HOLMES, D. **Java programming language**. 4th Edition, Editora Addison-Wesley, 2005.

Bibliografia

JANDL JUNIOR, P. **Introdução ao Java**. São Paulo: Editora Berkeley, 2002.



DEITEL, H. M., DEITEL, P. J. **JAVA como programar**. 10ª edição. São Paulo: Prentice-Hall, 2010.