



ALGORITMOS E PROGRAMAÇÃO I



01 | (0,5pt)

Implementar um programa em Java para **calcular o salário líquido** de um funcionário, a partir de seu salário base, do bônus mensal em porcentagem e do total de descontos em reais.

Entrada	Saída
1000 10 300	"Seu salário líquido é de R\$ 800.0"
1000 20 200	"Seu salário líquido é de R\$ 1000.0"

Implemente o exercício ao lado, seguindo a estrutura de funções abaixo

Função

Nomeação

```
calcular(double salario,  
         double bonus,  
         double desc) => double
```

Chamada

```
calcular(1000,10,300) = 800.0
```



02 | (0,5pt)

Implementar um programa em Java que **calcule quantas paradas para abastecimento** são necessárias para realizar uma viagem, a partir da capacidade do tanque de combustível (em litros), do consumo do veículo (km por litros) e da distância da viagem (km).

- * *Você deve partir do principio que o tanque está vazio.*
- * *Os valores de entrada devem ser entendidos como decimais.*

Entrada	Saída
50.0 10.0 400.0	"Você precisará fazer 1.0 paradas para abastecer."
50.0 10.0 700.0	"Você precisará fazer 2.0 paradas para abastecer."

Implemente o exercício ao lado, seguindo a estrutura de funções abaixo

Função

Nomeação

```
paradas(double capacidade,  
         double consumo,  
         double distancia) => double
```

Chamada

```
paradas(50.0, 10.0, 400.0) = 1
```



03 | (1,0pt)

Implemente um programa em Java que a partir da temperatura, **avale a situação** da pessoa conforme a tabela ao lado. Ao final, apresente a classificação.

Média	Situação
Maior igual a 41	Hipertermia
Maior igual a 39,6 e menor que 41	Febre Alta
Maior igual a 37,6 e menor que 39,6	Febre
Maior igual a 36 e menor que 37,6	Normal
Abaixo de 36	Hipotermia

Entrada	Saída
39.8	A situação para sua temperatura é Febre Alta
35.9	A situação para sua temperatura é Hipotermia

Implemente o exercício ao lado, seguindo a estrutura de funções abaixo

Função

Nomeação

```
situacaoFebre(double temperatura)  
=> String
```

Chamada

```
situacaoFebre(38.8) = "Febre"
```



04 | (1,0pt)

Implemente um programa em Java que informe a **situação de um orçamento familiar** baseado no total de ganhos e gastos. A situação deve ser calculada a partir da tabela abaixo:

Média	Situação
Gastos maiores que os Ganhos	Orçamento comprometido! Hora de rever seus gastos!
Gastos entre 81% e 100% dos Ganhos	Cuidado, seu orçamento pode ficar comprometido!
Gastos entre 51% e 80% dos Ganhos	Atenção, melhor conter os gastos!
Gastos entre 21% e 50% dos Ganhos	Muito bem, seus gastos não ultrapassam metade dos ganhos!
Gastos entre 0% e 20% dos Ganhos	Parabéns, está gerenciando bem seu orçamento!

Entrada	Saída
3000.0 2700.0	"Cuidado, seu orçamento pode ficar comprometido!"
3200.0 2000.0	"Atenção, melhor conter os gastos!"

Implemente o exercício ao lado, seguindo a estrutura de funções abaixo

Função

Nomeação

```
situacaoOrçamento(double ganhos,  
double gastos)  
=> String
```

Chamada

```
situacaoOrçamento(3000.0, 2700.0)  
= "Atenção, melhor conter os gastos!"
```

Referências

Tipos e Operadores



Tipos de Variáveis

Tipo	Tamanho	Limite
short	2 bytes	-32.768 a 32.767
int	4 bytes	-2.147.483.648 a 2.147.483.647
long	8 bytes	-9.223.372.036.854.775.808 a -9.223.372.036.854.775.807
float	4 bytes	$-3.40282347 \times 10^{38}$ a $1.40239846 \times 10^{-45}$
double	8 bytes	$1.76769313486231570 \times 10^{308}$ a $4.94065645841246544 \times 10^{-324}$
char	2 bytes	Todos caracteres Unicode.
boolean	1 byte	true e false



Operadores Matemáticos

Operações matemáticas envolvem números em seus operandos e em sua resposta. É possível realizar operações com valores fixos ou com variáveis. Quando a operação envolve valores de tipos diferentes, a resposta **será sempre do tipo com o maior conjunto de valores**, ou seja, do conjunto que contém o outro. Também é possível realizar expressões matemáticas contendo mais de uma operação. **A ordem da execução respeita as regras da matemática**, então se quisermos dar prioridade a uma adição ao invés de multiplicação, envolve-lá entre parênteses.

Símbolo	Nome	Exemplo
$\underline{\quad} + \underline{\quad}$	Adição	$10 + 5$
$\underline{\quad} - \underline{\quad}$	Subtração	$10 - 5$
$\underline{\quad} * \underline{\quad}$	Multiplicação	$10 * 5$
$\underline{\quad} / \underline{\quad}$	Divisão	$10 / 5$
$\underline{\quad} \% \underline{\quad}$	Módulo (Resto da divisão)	$10 \% 2$
$\underline{\quad} += \underline{\quad}$	Incrementação por Adição	$x += 5$
$\underline{\quad} -= \underline{\quad}$	Decrementação por Subtração	$x -= 5$
$\underline{\quad} *= \underline{\quad}$	Incrementação por Multiplicação	$x *= 5$
$\underline{\quad} /= \underline{\quad}$	Decrementação por Divisão	$x /= 5$
$\underline{\quad} ++ \underline{\quad}$	Incrementação Pré-Fixado	$++x$
$\underline{\quad} -- \underline{\quad}$	Decrementação Pré-Fixado	$--x$
$\underline{\quad} ++$	Incrementação Pós-Fixado	$x++$
$\underline{\quad} --$	Decrementação Pós-Fixado	$x--$



Operadores Relacionais

Operações relacionais podem envolver qualquer tipo de valores em seus operandos diferente dos operadores matemáticos. Sua especificidade se dá em sempre **retornarem um valor booleano**, ou seja, a ideia de relacionar está intimamente ligada a comparar.

Comparamos se algo é maior, menor, igual, diferente, entre outras disponíveis na linguagem. **Algumas comparações podem não ser implementadas para alguns tipos**, como por exemplo: Não é possível verificar se um texto é maior ou menor que outro.

Símbolo	Nome	Exemplo
$_ > _$	Maior que	$10 > 5$
$_ < _$	Menor que	$10 < 5$
$_ \geq _$	Maior ou igual que	$10 \geq 5$
$_ \leq _$	Menor ou igual que	$10 \leq 5$
$_ == _$	Igual a	$10 == 5$
$_ != _$	Diferente de	$10 != 5$






Operadores Lógicos

Operadores lógicos assim como os relacionais retornam um valor booleano. **Sua característica principal é receber em seus operandos apenas expressões booleanas.** Assim, operadores lógicos trabalham apenas com valores booleanos.

A operação lógica E retornará verdadeiro apenas se seus dois operandos forem verdadeiros, caso contrário retornará falso.

A operação lógica OU retornará verdadeiro se qualquer um de seus dois parâmetros for verdadeiro. Retornará falso apenas se os dois forem falsos.

Símbolo	Nome	Exemplo
	E lógico	10 > 5 && 5 > 0
	OU Lógico	10 < 5 5 > 6
	Negação	!true



Outros Operadores

Símbolo	Nome	Observação	Exemplo
$_ + _$	Concatenação	Junta um texto com outra informação	"Meu nome é: " + " Bruno "
$_ [_]$	Indexação	Acessa uma posição específica de um array	numeros[0]



Funções Matemáticas

Função	Observação	Retorno	Exemplo
<code>abs(_)</code>	Retorna o valor absolute de um número	int/double	<code>double x = Math.abs(-10);</code> // 10
<code>ceil(_)</code>	Retorna o número arredondado para cima	double	<code>double x = Math.ceil(9.1);</code> // 10
<code>floor(_)</code>	Retorna o número arredondado para baixo	double	<code>double x = Math.floor(9.9);</code> // 9
<code>pow(_,_)</code>	Retorna a potencia de um número	double	<code>double x = Math.pow(2, 4);</code> // 16
<code>log10(_)</code>	Retorna o logaritmo de um número na base 10	double	<code>double x = Math.log10(10);</code> // 1
<code>random()</code>	Retorna um valor aleatório entre 0 e 1	double	<code>double x = Math.random();</code> // ?
<code>round(_)</code>	Retorna o valor arredondado de um número	int/long	<code>long x = Math.round(5.6);</code> // 6
<code>sqrt(_)</code>	Retorna a raiz quadrado de um número	double	<code>double x = Math.sqrt(25);</code> // 5

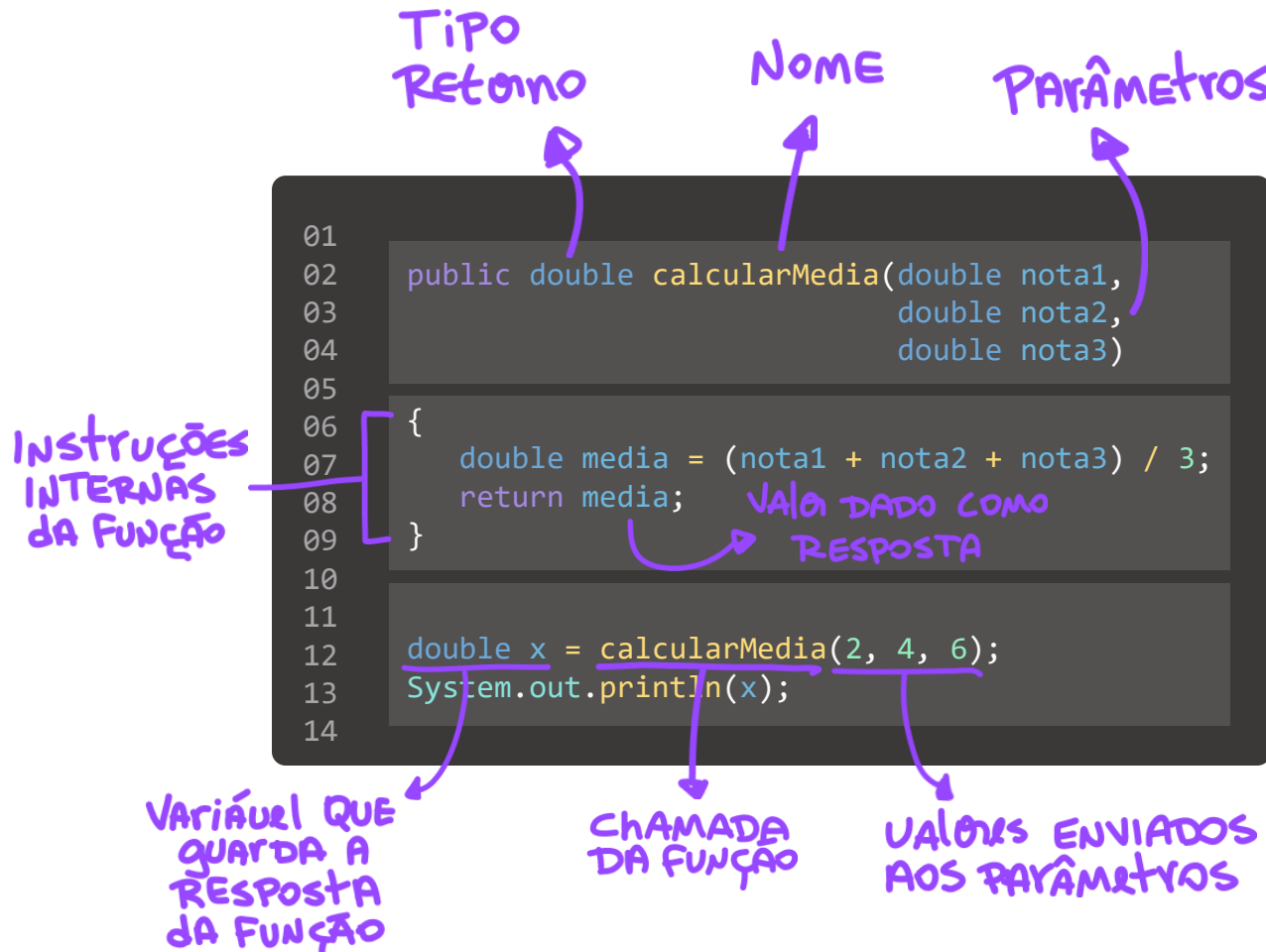


Funções de Texto

Função	Observação	Retorno	Exemplo
			<pre>string s = "Dev";</pre>
<code>charAt(_)</code>	Retorna o caractere de uma posição	char	<pre>char x = s.charAt(0); // 'D'</pre>
<code>codePointAt(_)</code>	Retorna o Código UNICODE de uma posição	int	<pre>int x = s.codePointAt(0); // 68</pre>
<code>contains(_)</code>	Verifica se um texto existe	boolean	<pre>boolean x = s.contains("v"); // true</pre>
<code>equals(_)</code>	Verifica se é igual a uma string	boolean	<pre>boolean x = s.equals("Dev"); // true</pre>
<code>indexOf(_)</code>	Retorna a posição de um texto	int	<pre>int x = s.indexOf("v"); // 2</pre>
<code>length()</code>	Retorna a quantidade de caracteres	int	<pre>int x = s.length(); // 3</pre>
<code>matches(_)</code>	Retorna se uma expressão regular é aceita	boolean	<pre>boolean x = s.matches("D.v"); // true</pre>
<code>replace(_,_)</code>	Substitui um texto por outro	String	<pre>String x = s.replace("e", "i"); // Div</pre>
<code>substring(_,_)</code>	Recorta uma string	String	<pre>String x = s.substring(1,3); // ev</pre>
<code>toLowerCase()</code>	Retorna todos caracteres em minúsculo	String	<pre>String x = s.toLowerCase(); // dev</pre>
<code>toUpperCase()</code>	Retorna todos caracteres em maiúsculo	String	<pre>String x = s.toUpperCase(); // DEV</pre>
<code>trim()</code>	Remove os espaços do começo e fim	String	<pre>String x = s.trim(); // Dev</pre>



Sintaxe de uma Função



Uma função é composta de sua **assinatura** e de seu **corpo de implementação**.

A assinatura é composta de **Nome, Parâmetros e Tipo de Retorno**. Na assinatura estão as informações para que o **programador application** utilize a função.

O corpo é composto das instruções que serão executadas quando a função for chamada. No corpo está o algoritmo criado pelo **programador implementor** quando ele criou a função.

Depois de criada, a função pode ser chamada (executada), para isso devemos **chamá-la pelo nome, enviar os valores aos parâmetros e guardar sua resposta**.



Entendendo Funções

No campo da ciência da computação, uma função (f) possui significado diferente quando comparada à matemática. Na matemática, uma função corresponde a associação dos elementos de dois conjuntos. Na ciência da computação, uma função é um elemento capaz de nomear uma sequência de operações realizadas a partir de valores de entrada com objetivo de chegar a um resultado, ou valor de saída.

Função que dobra um número

$$f(10) = 20$$

Função que soma dois números

$$f(10, 5) = 15$$

Função que calcula a média de três notas

$$f(10, 5, 3) = 6$$



Outros cenários com Funções

Função que calcula a metade de um número

$$f(11) = 5.5$$

Função que calcula total de uma compra a partir do valor total e desconto em %

$$f(1000, 10) = 900$$

Função que calcula a área do quadrado

$$f(10) = 100$$

Função que verifica se a pessoa é de Libra a partir do mês e dia

$$f(\text{"Outubro"}, 22) = \text{true}$$

Função que calcula a área de um triângulo

$$f(10, 5) = 25$$

Função que verifica se uma cor é primária

$$f(\text{"azul"}) = \text{true}$$



Implementando de Funções

Uma função (f) pode ser vista por dois ângulos. O primeiro, vimos anteriormente, onde se envia os valores à função e ela retorna uma resposta. Nessa visão, a preocupação está apenas em **usar a função**. A segunda visão preocupa-se em **como** a função deve ser construída, ou seja, **quais operações** devem ser feitas com os **valores recebidos** para obter-se o **resultado final**.

Função que dobra um número

$$f(a) \Rightarrow a * 2$$

Função que soma dois números

$$f(a, b) \Rightarrow a + b$$

Função que calcula a média de três notas

$$f(a, b, c) \Rightarrow (a + b + c) / 3$$



Outros cenários com Funções

Função que calcula a metade de um número

$$f(a) \Rightarrow a / 2$$

Função que calcula a área de um triângulo

$$f(a, b) \Rightarrow a * b / 2$$

Função que calcula a área do quadrado

$$f(a) \Rightarrow a * a$$

Função que calcula total de uma compra a partir do valor total e desconto em %

$$f(a, b) \Rightarrow a - (a * b / 100)$$



Outros cenários com Funções

Função que verifica se a pessoa é de Libra a partir do mês e dia

```
f(a, b) => (a == "Outubro" && b <= 22) ||  
           (a == "Setembro" && b >= 23)
```

Função que verifica se uma cor é primária

```
f(a) => a == "azul" ||  
        a == "vermelho" ||  
        a == "amarelo"
```



Máquina de Função (CalcularMedia)

```
01
02  public double calcularMedia(double nota1,
03                               double nota2,
04                               double nota3)
05
06  {
07      double media = (nota1 + nota2 + nota3) / 3;
08      return media;
09  }
10
11
12
13  double x = calcularMedia(2, 4, 6);
14  System.out.println(x);
15
16
17
```

$$f(a, b, c) \Rightarrow (a + b + c) / 3$$



Boa prova!
Bruno de Oliveira