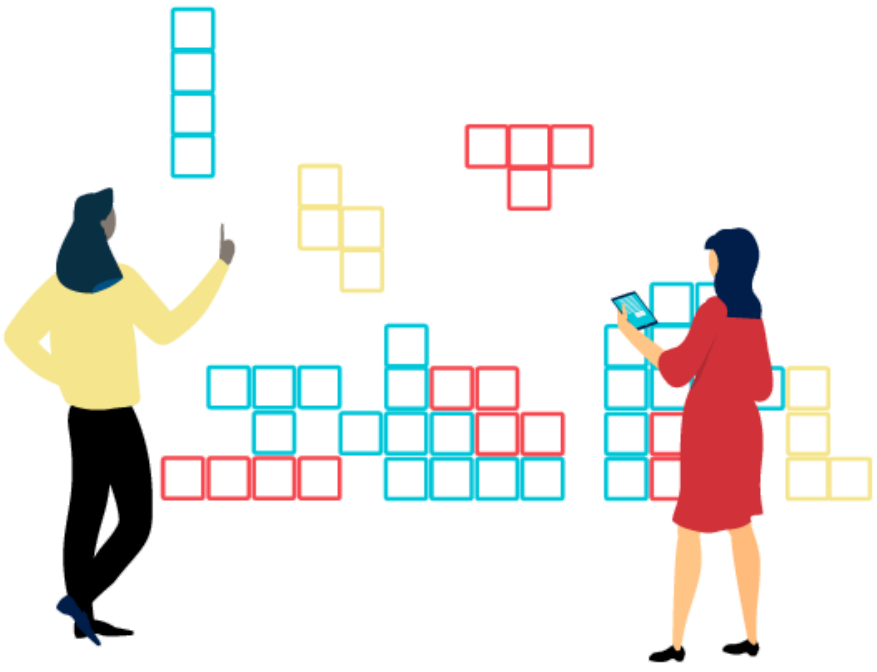


DESENVOLVENDO A LÓGICA

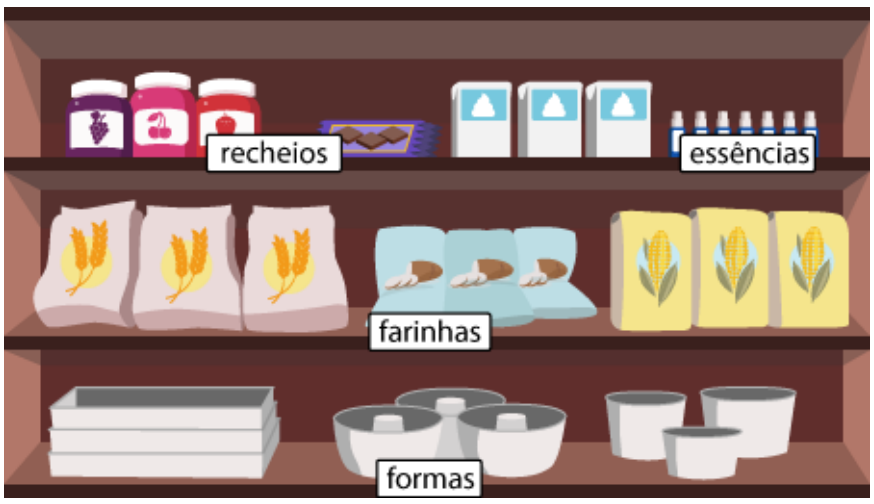
← Voltar

Este texto-base aborda os seguintes temas:

- Um olhar inicial sobre variáveis e dados
- Criando o modelo físico
- Manipulando o modelo físico

Um olhar inicial sobre variáveis e dados

Na unidade anterior, você viu que algoritmo é uma sequência de instruções ordenadas de forma lógica, como uma receita de bolo. Essa receita é escrita de uma determinada maneira, podendo ser registrada como pseudocódigo ou fluxograma. Agora, apresentamos os conceitos de **variáveis** e **dados**:



Bem, podemos entender uma **variável** como uma parte definida no armário **em que guardamos** alguns ingredientes e itens necessários à receita (**dados**). Para que qualquer pessoa consiga localizar os itens na cozinha, podemos etiquetar as áreas do armário indicando o nome do que se encontra ali. Da mesma forma, a variável precisa ser identificada (nome) e possuir um único tipo de dado (por exemplo, somente tipos de farinha).

Dado

Valor ou operação que uma variável pode armazenar.

Clique para virar o cartão

← Voltar

Variáveis

Durante os seus estudos, você notou que até o momento não armazenamos nenhum dado, trabalhamos apenas com procedimentos, como o caso do algoritmo “Fritar ovos”.

A partir de agora, trabalharemos com valores variáveis e fixos, por exemplo, o nome do cliente de uma loja que varia de cliente para cliente (variável) ou valores que são fixos (constantes), como é o caso de um valor que se mantém para todos os registros, como o valor de PI (3.14159) ou a quantidade de meses do ano, que sempre será 12. Nesse caso, não há alteração de conteúdo. Essa é a definição de uma constante, isto é, um local na memória do computador que armazena um dado que não se altera ao longo da execução do programa.

Se um computador trabalha em contínua interação com o usuário, por que até este momento não trabalhamos com essa interação?

A resposta é simples! Antes de iniciarmos efetivamente a interação homem-máquina, precisamos compreender os aspectos básicos de qualquer Linguagem de Programação, sendo assim, agora que você já conhece esses aspectos, estamos prontos para trabalharmos um novo conceito: **Variável**.

O computador é um objeto que não possui a capacidade de pensar por conta própria, sendo necessário sempre uma programação para ensiná-lo a trabalhar.

Quando iniciamos uma interação homem-máquina, não há como o computador saber o que o usuário fará na sequência de seus atos, por isso, devemos “ensiná-lo” a se preparar para uma interação com o usuário do computador.

Essa interação é feita por meio de uma estrutura chamada variável. Ela consiste na alocação de um pedacinho da memória RAM do computador para que ele receba uma informação vinda de um dispositivo de entrada de dados, no nosso caso, o teclado.

← Voltar

Para melhor compreender esse conceito, imagine a seguinte situação:

Em sua primeira aula de Lógica de Programação, você conheceu um colega de turma chamado Caio. Para que em um futuro próximo vocês possam trocar informações sobre as atividades propostas, você decidiu pedir a ele seu WhatsApp, ou seja, seu número de telefone.

No momento em que você pede ao Caio o número do telefone, você inconscientemente prepara um lugar para armazená-lo: seu celular, um pedaço um papel ou até mesmo na sua memória, não é mesmo?

Você saberia, de antemão, qual seria o número que Caio lhe passaria, ou seja, quais seriam os tipos de dados que receberia? Provavelmente não, mas saberia com certeza que ele lhe passaria um **número de telefone**.

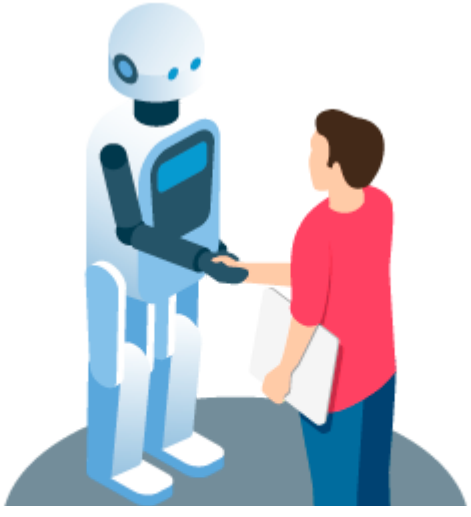
Resumindo:

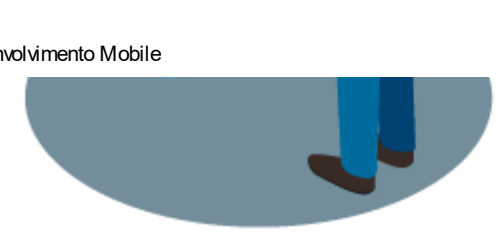
Se você fosse um computador, estaria utilizando uma variável (espaço na memória) do tipo numérica para receber o dado que seria passado pelo Caio (agente externo ao programa).

Fácil, não?

Logo, se você estivesse criando um programa para que o Caio e outros amigos inserissem seus números de telefone para que você os consulte depois, precisaria ensinar o computador que ele deve reservar um pequeno espaço em sua memória RAM para receber esses valores.

Será que para o computador é tão simples somar 1+1 como para nós, seres humanos?





← Voltar

Vamos analisar...

Primeiramente é necessário **armazenar** o primeiro número fornecido pelo usuário na memória do computador.

> *1. Ler e armazenar o primeiro número seria a primeira tarefa.*

Por que armazenar?

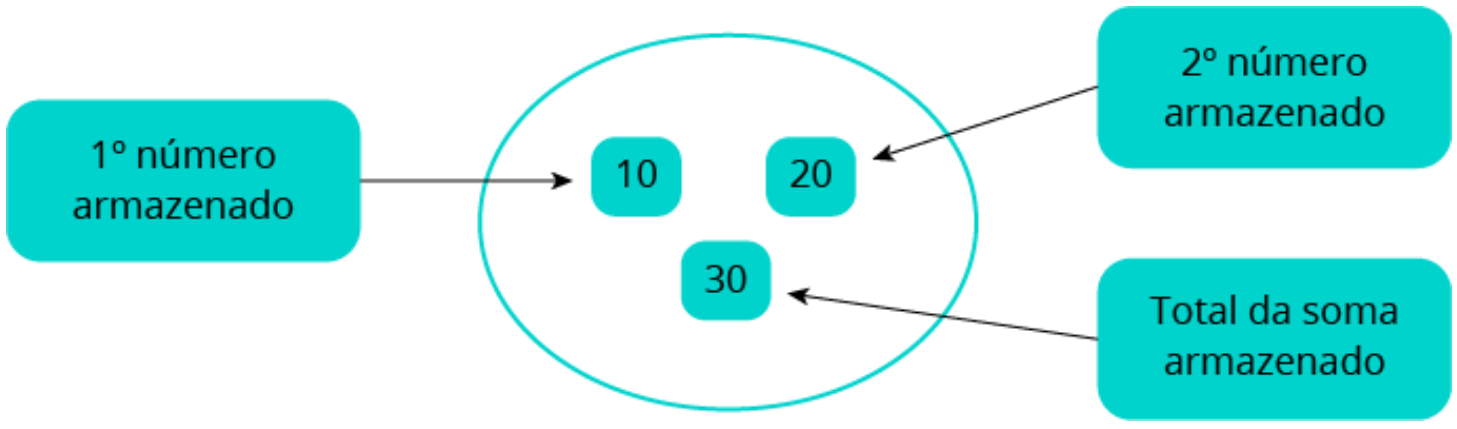
Porque o computador somará o primeiro número com o próximo número fornecido! Enquanto o usuário indica o segundo número, o computador deverá ter registrado na memória o primeiro número. Se esse passo não for feito, quando fornecermos o próximo número, o computador já terá esquecido o primeiro!

E depois?

> *1. Ler e armazenar o primeiro número seria a primeira tarefa.*
>
> *2. Ler e armazenar o segundo número, pelo mesmo motivo que o primeiro!*
>
> *3. Executar a operação de soma.*
>
> *4. Memorizar o resultado da soma para mostrar em sua tela!*

Quando falamos em **memorizar**, estamos dizendo que o dado deve ser colocado na memória do computador. Inserimos valores na memória de um computador por meio das **variáveis**.

Agora, veja esse esquema que representa a memória do computador:



Você percebeu que o computador precisará registrar três variáveis, correto? É necessário **dar nome** aos lugares onde esses valores estão armazenados e indicar **qual tipo de dado** eles podem ser. Fazemos isso por meio da **declaração de variável**, que conta com algumas regras:

Regras para a declaração de variáveis

Para a caixinha onde está armazenado o 1º número (10), poderia escolher um nome qualquer desde que:

~~1numero~~	Não seja número ou que comece com um número.

~~primeironúmero~~ Não tenha acento.

~~primeiro numero~~ Não tenha espaços em branco!

~~prim°numero~~ Não utilize caracteres especiais

Utilizando nosso exemplo, um nome de variável possível seria: primeironumero

← Voltar

Operadores

Quando utilizamos a lógica de programação, sendo ela em pseudocódigo ou em uma linguagem de programação, devemos utilizar alguns **símbolos** ou **palavras** para que o computador entenda o que queremos que ele faça, chamados **operadores**.

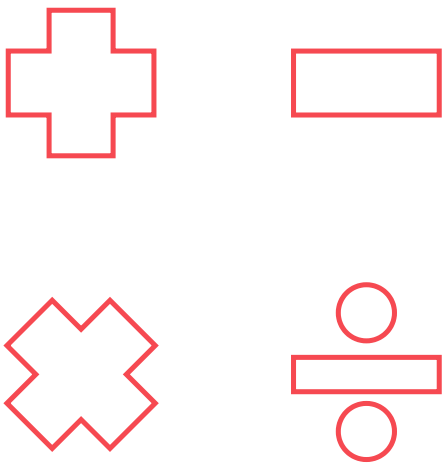
Operadores

Símbolo ou palavra específica que indica ao computador partes da instrução.

Clique para virar o cartão

Existem muitos tipos de operadores, mas neste momento estudaremos os operadores aritméticos, relacionais e lógicos.

Operadores Aritméticos



Se for necessário desenvolver uma **operação matemática** utilizando pseudocódigo ou uma linguagem de programação, você precisará dos operadores aritméticos para criar o seu programa.

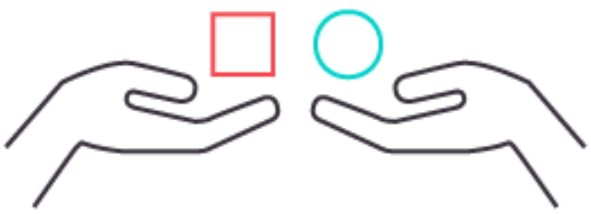
Apesar do computador utilizar a mesma regra da matemática para a resolução de cálculos, **a simbologia utilizada nem sempre é a mesma da matemática**. A seguir, você conhecerá os operadores aritméticos e sua simbologia em pseudocódigo em **Java**, linguagem que utilizaremos a partir da próxima unidade para implementar os pseudocódigos. Veja também a sua utilização e exemplos:

NOME DA OPERAÇÃO	SINTAXE EM PSEUDOCÓDIGO	SINTAXE EM JAVA	FUNÇÃO	EXEMPLO
Soma	+	+	Efetua a soma entre 2 valores numéricos.	3 + 2 = 5

NOME DA OPERAÇÃO	SINTAXE EM PSEUDOCÓDIGO	SINTAXE EM JAVA	FUNÇÃO	EXEMPLO
Subtração	-	-	Efetua a subtração entre 2 valores numéricos.	3 – 2 = 1
Multiplicação	*	*	Efetua a multiplicação entre 2 valores numéricos.	2 * 4 = 8
Divisão	/	/	Efetua a divisão entre 2 valores numéricos.	4 / 2 = 2
Potenciação	exp(b,e)(b = base e e = expoente)	^	Efetua a potenciação entre 2 valores.	Exp(3,2) = 9 (Pseudocódigo) 3 ^ 2 = 9 (Java)
Resto da divisão	mod	%	Efetua a divisão entre 2 valores, mas retorna o valor do resto da divisão.	3 mod 2 = 1 (Pseudocódigo) 3 % 2 = 1 (Java)
Concatenação	+	+	Junta 2 valores do tipo caractere.	“Lógica” + “Programação” = “LógicaProgramação”

← Voltar

Operadores Relacionais



Estes operadores são os responsáveis por **efetuar comparações entre dados**, com o objetivo de mostrar ao programa como proceder dependendo da situação apresentada. O programa de computador verá o resultado de uma comparação em duas situações lógicas, sendo verdadeiro ou falso. Assim como os operadores aritméticos, segue uma pequena tabela indicando suas finalidades:

NOME DO OPERADOR RELACIONAL	SINTAXE EM PSEUDOCÓDIGO	SINTAXE EM JAVA	FUNÇÃO	EXEMPLO	RESULTADO
Maior	>	>	Compara se o primeiro valor é maior que o segundo valor.	7 > 3	Verdadeiro
Menor	<	<	Compara se o primeiro valor é menor que o segundo valor.	(3+1) < (5*0)	Falso
Maior ou Igual	>=	>=	Compara se o primeiro valor é maior ou igual ao segundo valor.	(4 + 4) >= 8	Verdadeiro
Menor ou Igual	<=	<=	Compara se o primeiro valor é menor ou igual ao segundo valor.	4 <= 4	Verdadeiro
Igual	=	==	Compara se o primeiro valor é igual ao segundo valor.	3 = 2 (Pseudocódigo) 3 == 2 (Java)	Falso

NOME DO OPERADOR RELACIONAL	SINTAXE EM PSEUDOCÓDIGO	SINTAXE EM JAVA	FUNÇÃO	EXEMPLO	RESULTADO
Diferente	<>	!=	Compara se o primeiro valor é diferente do segundo valor.	7 <> 3 (Pseudocódigo) 7 != 3 (Java)	Verdadeiro

← Voltar

Operadores Lógicos

Estes operadores são os responsáveis por **efetuar comparações entre dados**, com o objetivo de mostrar ao programa como proceder dependendo da situação apresentada. O programa de computador verá o resultado de uma comparação em duas situações lógicas, sendo verdadeiro ou falso. Assim como os operadores aritméticos, segue uma pequena tabela indicando suas finalidades:

Paulo, um jovem de 20 anos, gostaria de saber se na próxima eleição ele será obrigado a votar ou se poderá votar de modo facultativo. Para ser obrigado a votar, o eleitor deve ter a idade maior ou igual a dezoito anos e também o eleitor deve ter menos que 70 anos.

Neste caso, para que a expressão seja corretamente resolvida, é necessário utilizar Operadores Lógicos, que serão apresentados a seguir:

NOME DO OPERADOR LÓGICO	SINTAXE EM PSEUDOCÓDIGO	SINTAXE EM JAVA	FUNÇÃO	EXEMPLO	RESULTADO
E	E	&&	Para que o resultado da comparação seja verdadeiro, os dois lados da expressão devem ser verdadeiros.	(16 >= 16) E (16 < 18) (Pseudocódigo)(16 >= 16) && (16 < 18) (Java)	Verdadeiro
OU	OU	||	Para que o resultado da comparação seja verdadeiro, apenas um dos lados da expressão deve ser verdadeiro.	((3 + 2) < 5) OU ((3*2)=6) (Pseudocódigo)((3+4) < 5) || ((3*2)==6)(Java)	Verdadeiro
NÃO	não	!	Inverte o resultado da expressão, ou seja, caso a expressão seja verdadeira, se tornará falso e vice-versa.	NAO(4 < 8) (pseudocódigo)! (4 < 8) (Java)	Falso

Observando a tabela dos Operadores Lógicos, podemos concluir que a expressão que resolveria o problema de Paulo seria:

((20 >=18) && (20 < 70))

Como a idade dele é maior ou igual a 18 e também é menor que 70, Paulo é obrigado a votar.

A Tabela Verdade (operadores lógicos)

← Voltar

Uma forma muito simples de lembrar qual é o operador correto para satisfazer uma expressão é utilizando a **Tabela Verdade**. Com ela podemos prever e entender melhor o funcionamento dos Operadores Lógicos.

A tabela consiste em **separarmos a comparação em dois blocos**, sendo o **primeiro antes** do Operador Lógico, e o **segundo logo após** o operador. Para simularmos os resultados, definimos as respostas como verdadeiras (V) ou falsas (F), facilitando a simulação e evitando o uso de tempo na resolução das expressões.

Usando esse tipo de tabela verificamos todas as possibilidades de resultados: ambas as comparações podem ser verdadeiras, ambas falsas ou apenas uma delas verdadeira. Analise as tabelas a seguir:

Tabela verdade do operador E

TABELA VERDADE DO OPERADOR E		
Entrada 1	**Entrada 2**	**Saída**
V	V	V
V	F	F
F	V	F
F	F	F

No operador E, a saída será verdadeira somente se todas as entradas forem verdadeiras, caso contrário a saída será falsa

A entrada 1 é o primeiro bloco. A entrada 2, o segundo. A saída é o resultado das entradas de dados.

Aplicando a tabela verdade em Pseudocódigo e Java temos:

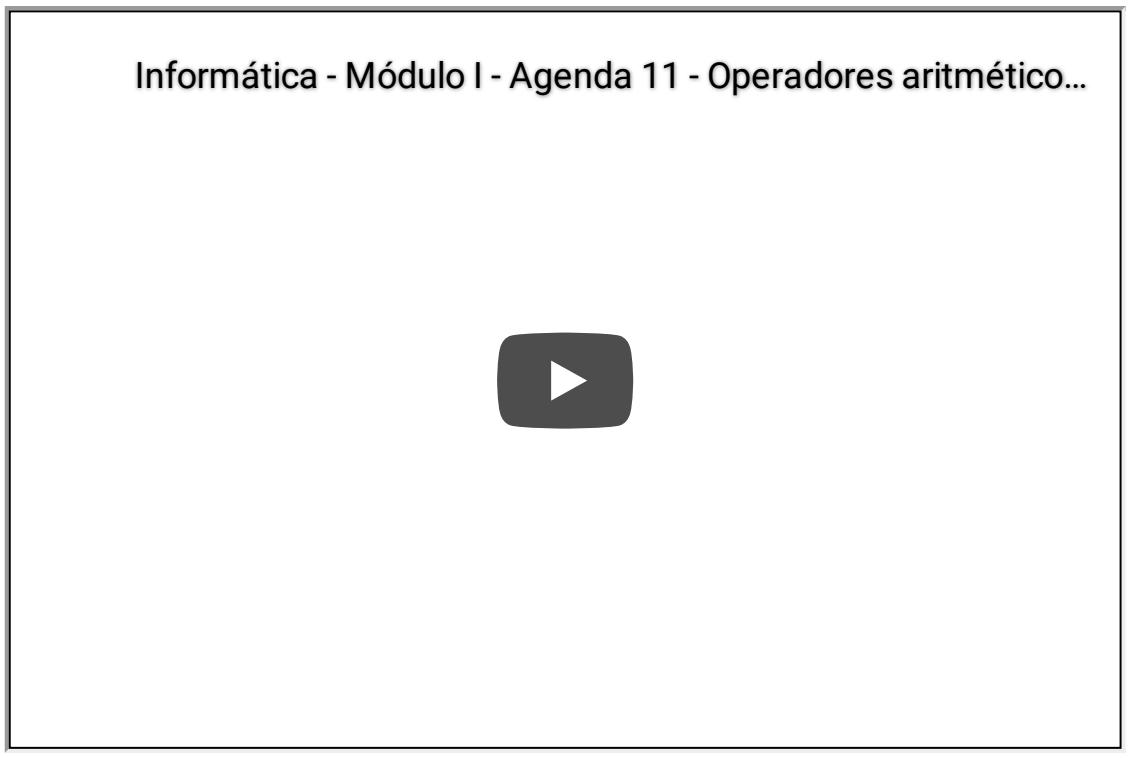
OPERADOR E (& &)		
Expressão em Pseudocódigo	**Expressão em Java**	**Resultado**
VEV	V&&V	V
VEF	V&&F	F
FEV	F&&V	F
FEF	F&&F	F

Note que, assim como na tabela superior, o resultado da expressão foi verdadeiro apenas quando ambos os lados da expressão são verdadeiros.

VOLTAR PRÓXIMO

Para entender mais sobre os Operadores Aritméticos, Relacionais e Lógicos, assista à videoaula do Prof. Sandro Valérius:

← Voltar



Avaliar este recurso | Reportar erro