



# Estruturas Condicionais


## Introdução

Algoritmo é definido como uma sequência de instruções que possui um início e fim bem delimitados. Até este ponto você deve compreender. Mas vamos analisar a palavra sequência. Os algoritmos mais simples, como aqueles para realizar uma operação aritmética sobre dois números, por exemplo, possui a entrada (os números que vamos realizar a operação), um processamento (que é a soma) e um resultado.

Analisando de perto, temos uma sequência encadeada de instruções. Porém, e se o usuário não informar um número, ou então, ao invés de informar um número ele informar uma letra? E se o valor informado for negativo? São muitos os que podemos considerar, ou seja, muitas condições que podem ser impostas em um único código. Nesta aula, compreenderemos o conceito de fluxo de código, bem como, conheceremos as principais estruturas condicionais utilizadas neste controle de fluxo.

## Objetivos da aula

- Compreender o conceito de controle de fluxo de código.
- Reconhecer os tipos de operadores lógicos e de comparação.

- Compreender o significado de cada tipo de estrutura de controle de fluxo. 
- Diferenciar as estruturas de condição simples, composta e aninhada.

## Resumo

Antes de começarmos a trabalhar o conceito de estrutura de decisão e também sobre estrutura de repetição, vamos entender um pouco mais sobre o fluxo de um algoritmo. Considere o algoritmo da figura 1 a seguir.

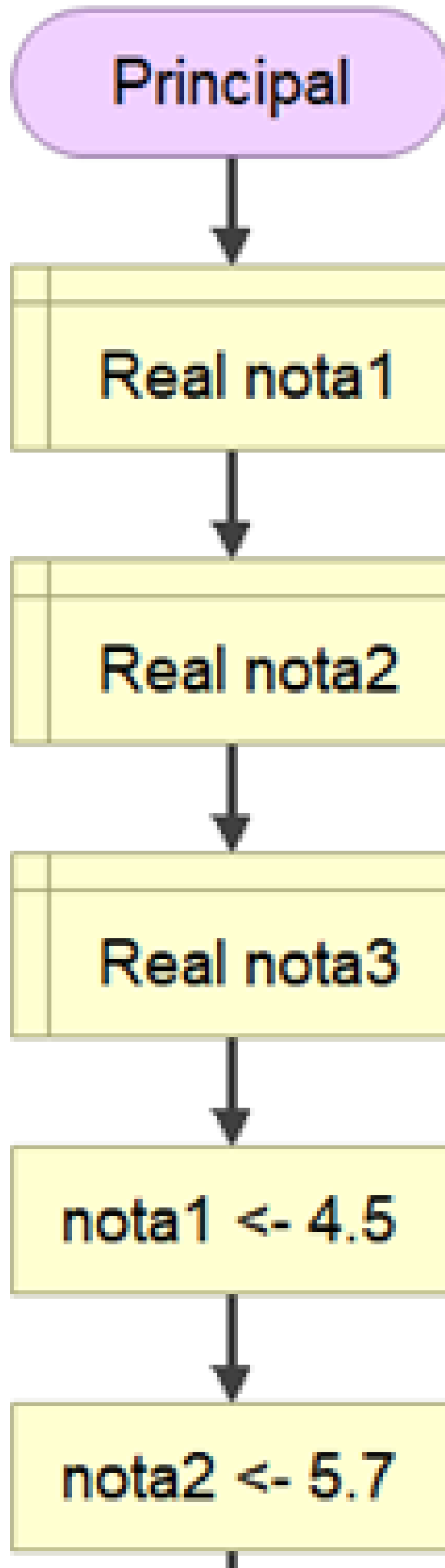





Figura 1 – Algoritmo com estrutura sequencial Fonte: Elaborado pelo autor

Na figura 1, o algoritmo é responsável por declarar e receber três notas de um aluno, em seguida é realizado o cálculo para obter a média.  Por fim, a média é apresentada na tela para o usuário. Observe então que se trata de uma sequência linear de instruções. A esta sequência chamamos de **fluxo**.

Porém, qual a razão de se calcular uma média se não for para dizer se o aluno está com desempenho satisfatório ou insatisfatório? Em outras palavras, se calcularmos a média, desejamos saber se o aluno está aprovado ou se está reprovado. Aqui nós utilizaremos um conceito de controle de fluxo, onde vamos alterar o fluxo do nosso código de acordo com a média do aluno. Por exemplo, se a nota for maior ou igual a 6, significa que ele está aprovado, caso contrário, ele estará reprovado.

Para tanto, precisamos primeiramente utilizar os operadores relacionais, os quais recebem este nome pois relacionam um operando da esquerda com outro operando da direita. O resultado desta expressão será utilizado em uma **estrutura condicional** para indicar para qual caminho o fluxo do código seguirá.

Em qualquer linguagem de programação, as estruturas condicionais, também conhecidas como estrutura de decisão ou de controle, são utilizadas para manipular o fluxo de um código (MANZANO, 2016). Sem essas estruturas, uma linguagem de programação é basicamente inútil, afinal, não será capaz de tomar decisões inteligentes por conta própria.

Para que seja possível utilizarmos as estruturas de decisão, precisamos de ter 2 componentes essenciais:

- **operadores de comparação:**

- (maior);



- < (menor);
  - >= (maior ou igual);
  - <= (menor ou igual);
  - == (igual);
  - != (diferente).
- 
- **operadores lógicos:**
    - && (operador e em C, Java, C#, JavaScript, entre outras);
    - and (operador e em Python);
    - || (operador ou em C, Java, C#, JavaScript, entre outras);
    - or (operador ou em Python).

Os operadores de comparação, como o nome indica, compara um valor/expressão do lado esquerdo com um valor/expressão do lado direito. O retorno será true ou false.

Os operadores lógicos realizam uma conexão entre duas expressões, é o que se denomina Expressões Lógicas ou Expressões Booleanas. A tabela 1 a seguir apresenta um exemplo do uso dos operadores de comparação e operadores lógicos, bem como suas saídas em Java. Considere que a variável X possui o valor 4 e a variável Y possui o valor 8.



Instrução / Expressão	Saída
<code>X &gt; Y</code>	false
<code>X == Y</code>	false
<code>X != Y</code>	true
<code>X &lt; Y &amp;&amp; X != 4</code>	true && false □ false
<code>X &lt; Y    X != 4</code>	true    false □ true
<code>X == 5    Y == 8</code>	false    true □ true
<code>X == 4 &amp;&amp; Y == 8</code>	true && true □ true

Tabela 1 – Uso dos operadores lógicos e relacionais

Observe que o operador `&&` (operador e), para que tenha uma saída verdadeira, precisa que todas as expressões ligadas por este operador também sejam verdadeiras. Por outro lado, ao utilizar um operador `||` (operador ou), para que a saída seja verdadeira, basta que qualquer expressão tenha o valor verdadeiro.

Agora que você conhece os operadores, a montagem das instruções e as saídas destas instruções, chegou a hora de conhecermos as estruturas condicionais.

A estrutura condicional `if` é a primeira que trabalharemos. O seu papel é dividir o fluxo do código. Observe o diagrama da figura 2.

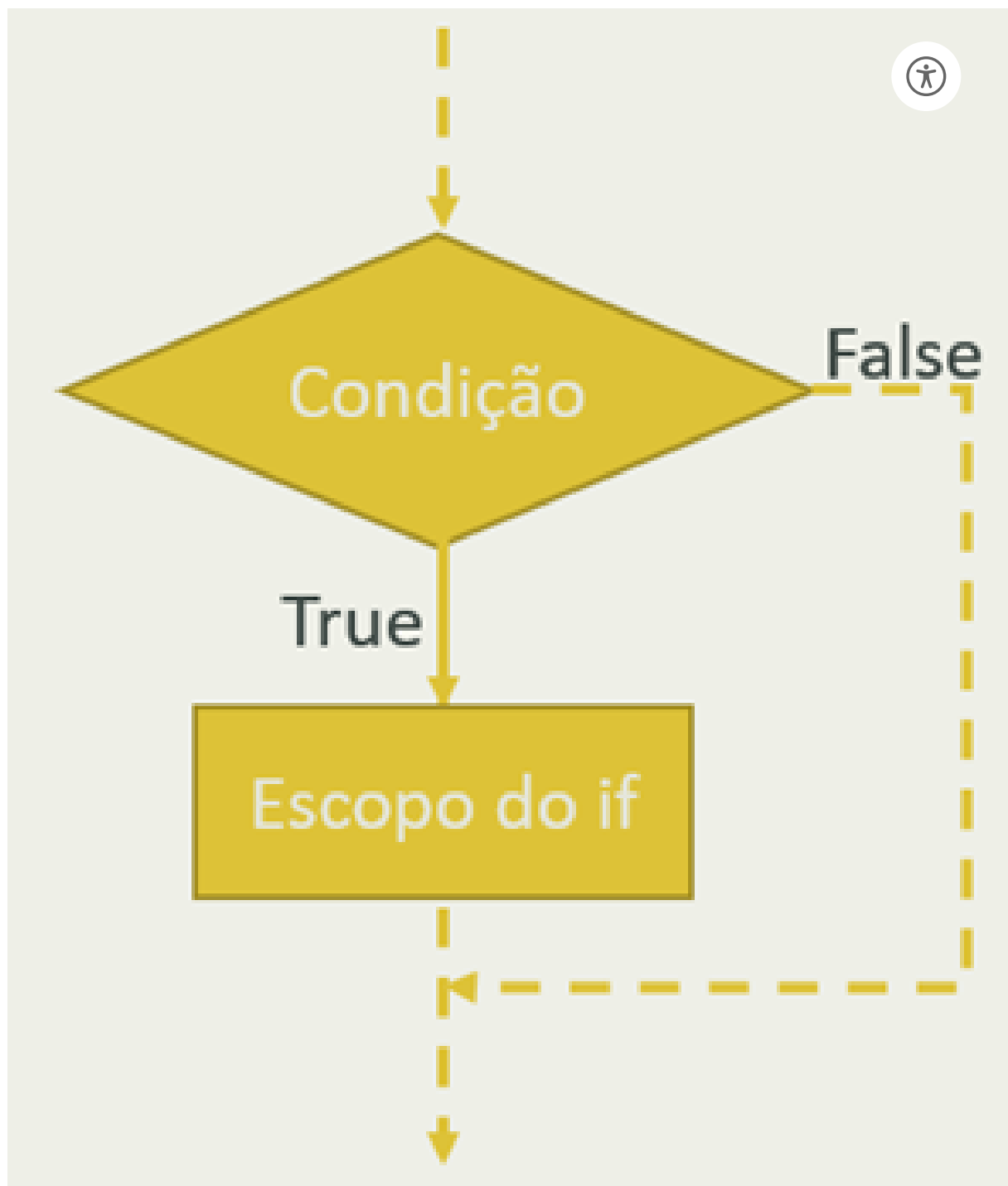



Figura 2 - Estrutura condicional if Fonte: Elaborado pelo autor

Observe na figura 2 o fluxo sequencial do algoritmo representado pela seta superior descendente. Em um dado momento, o fluxo de execução encontra uma estrutura de decisão (representado pelo losango). Nesta estrutura de decisão teremos uma expressão lógica ou uma comparação entre dois valores. Caso a condição seja **falsa**, o fluxo de execução continuará. Porém, caso a condição seja **verdadeira**, o fluxo do código



será alterado, executando uma outra sequência de instruções (FORBELLONE, 2005). O código 1 a seguir ilustra o uso da estrutura .

1	<code>int x = 4, y=8;</code>
2	<code>if(x&gt;y){</code>
3	<code>System.out.println("O valor de x é maior que o valor de y");</code>
4	<code>}</code>

Código 1 – Estrutura condicional if

Fonte: Elaborado pelo autor

Observe que uma estrutura condicional é representada pela palavra-chave `if`, seguida de uma expressão lógica. O conteúdo da linha 3 só será apresentado na tela quando **se** o valor de `x` for maior que o valor da variável `y`. Porém, neste caso, não será apresentado nada, afinal, o valor de `x` não é maior que o valor de `y`. Neste caso, temos a opção de utilizar uma **estrutura condicional composta** como ilustra a figura 3 a seguir.

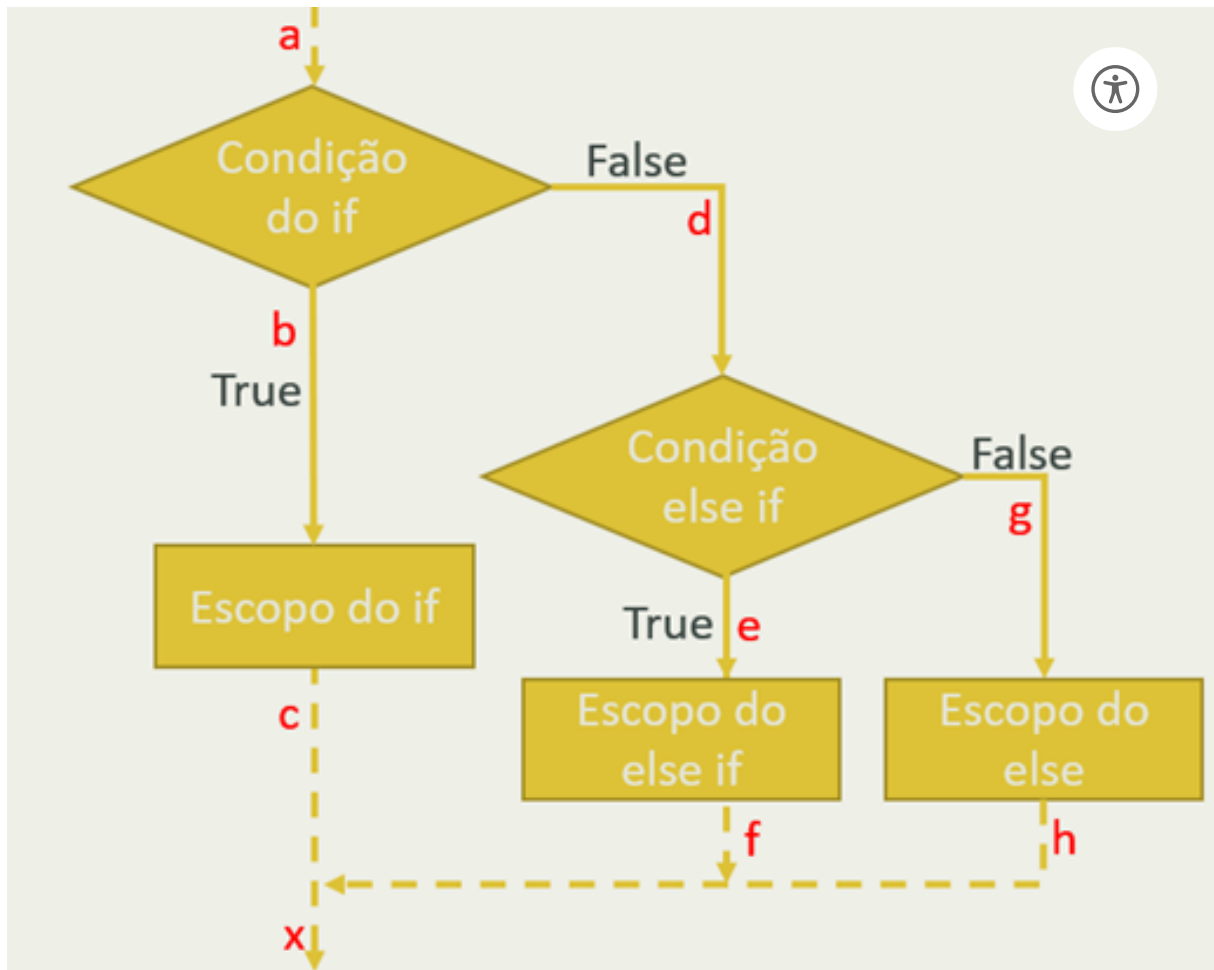



Figura 3 - Estrutura condicional aninhada e composta Fonte: Elaborado pelo autor

Observe pela figura 3 que o fluxo do código estava sequencial em (a). Entretanto, em um primeiro momento ele encontrou uma condição if. Caso a expressão seja verdadeira, o fluxo do código seguirá em (b) e, assim, executará as instruções internas ao if, seguindo o fluxo (c) até concluir em (x). Podemos então atualizar o código 1 para o código 2 a seguir.

1	<code>int x = 4, y=8;</code>
2	<code>if(x&gt;y){</code>
3	<code>System.out.println("O valor de x é maior que o valor de y");</code>

4	}	
5	else if(x<y){	
6	System.out.println("O valor de x é menor que o valor de y");	
7	}	
8	else {	
9	System.out.println("O valor de x é igual a y");	
10	}	

Código 2 – Estrutura condicional if composta

Observe agora pelo código 2 que temos 3 condições: se x for maior que y, se x for menor que y e uma saída para o caso onde x não é maior e nem menor, ou seja, para quando x for igual a y. O conteúdo das linhas 3, 6 ou 9 só serão apresentados na tela **se** as condições forem verdadeiras.

Com isso é possível concluir que as estruturas condicionais são responsáveis por controlar o fluxo do código. Considere, por exemplo, uma aplicação como das redes sociais Instagram ou Facebook. Ao acessá-las, você precisa colocar um nome de usuário e senha. Se o nome de usuário e senha estiverem corretos, você terá acesso a aplicação. Por outro lado, se não estiverem corretos, você não terá acesso. Note então que a condicional é essencial em qualquer linguagem de programação e para todo tipo de projeto de software.

## Tópicos avançados

Observando as tabelas verdade, vemos que `TRUE` e/ou `TRUE` retorna `TRUE`. As tabelas verdade completas, fornecem exemplos das expressões básicas usadas na lógica e suas conclusões. Essas tabelas matemáticas comuns são úteis para memorizar ou manter em mente ao construir algoritmos (instruções) na programação de computadores.

Estes algoritmos lógicos podem ser aplicados em diferentes cenários e um dos mais promissores é a inteligência artificial ou IA.

