4. Operadores Lógicos 2

4.1 Conceito de TRUE, FALSE e NOT no algoritmo

Já conhecemos a estrutura condicional e como podemos executar trechos de códigos se uma condição for VERDADEIRA (TRUE) ou outro trecho de código se a mesma condição for FALSA (FALSE). Vamos relembrar a estrutura condicional:

SE **<condição>** ENTÃO ***<código se for verdadeiro>*** SENÃO ***<código se for falso>*** FIM SE	IF **<condição>** THEN ***<código se for verdadeiro>*** ELSE ***<código se for falso>*** END IF

Isto significa que o trecho **<condição>**, quando verificado, retorna sempre um valor: ou VERDADEIRO ou FALSO, executando seu respectivo bloco ENTÃO ou SENÃO.

Estes operadores TRUE e FALSE são chamados de ***operadores booleanos*** e podem ***funcionar como valores***, como se fossem números. Vamos rever aquele nosso algoritmo de imposto sobre o salário que fizemos e realizar uma alteração nele, incluindo estes operadores booleanos:

Algoritmo original	Algoritmo com operadores booleanos
[1] Salário = 1000 [2] Quantidade_Filhos = 0 [3] SE **Quantidade_Filhos = 0** [4] ENTÃO [5] Imposto = 100 [6] SENÃO [7] Imposto = 50 [8] FIM SE [9] Salário_Final = Salário – Imposto	[1] Salário = 1000 [2] Tem_Filhos = FALSE [3] SE **Tem_Filhos = TRUE** [4] ENTÃO [5] Imposto = 50 [6] SENÃO [7] Imposto = 100 [8] FIM SE [9] Salário_Final = Salário – Imposto

Vamos prestar bastante atenção nessa modificação que fizemos nos algoritmos de imposto:

- O algoritmo original contém na linha 2 o valor da quantidade de filhos em forma de número, onde poderia ser 0 (zero), ou seja, o funcionário não tem filhos, como foi neste caso.

- A estrutura condicional original compara na linha 3 se ***Quantidade_Filhos*** com zero, caso verdadeiro, o funcionário não tem filho e seu imposto era maior (100). Funcionários com filhos tem imposto menor (50).

- Na estrutura com operadores booleanos, trocamos o nome do valor ***Quantidade_Filhos*** para ***Tem_Filhos***, na linha 2. Isto significa que antes este valor era numérico (0, 1, 2, 3...), agora este possui uma resposta para "Tem_Filhos", que é um ***valor lógico*** (VERDADEIRO ou FALSO).

- Na linha 3 da estrutura com operadores booleanos, como a operador condicional SE compara se o valor de "Tem_Filhos" é VERDADEIRO. Como ele contém o valor FALSO, este funcionário terá o valor de imposto de 100, pois realmente não tem filho.

- Por fim e mais importante, reparem que os valores de imposto estão invertidos, pois antes verificámos a condição de ele ***não ter filhos*** (Quantidade_Filhos = 0) e agora verificamos a condição de ele ***ter filhos*** (Tem_Filhos = TRUE), pela lógica, quem tem filho paga menos imposto.

Vamos ver um conceito bem interessante agora, peço que você ***preste bastante atenção***. O valor em Tem_Filhos da linha 2 é um valor lógico (TRUE ou FALSE) e a estrutura condicional SE verifica se uma condição é VERDADEIRA ou FALSA (TRUE ou FALSE).

Isto significa que ambos valores estão na mesma forma: valores booleanos (TRUE e FALSE). Então, podemos simplificar a verificação se um valor é VERDADEIRO, da seguinte forma:

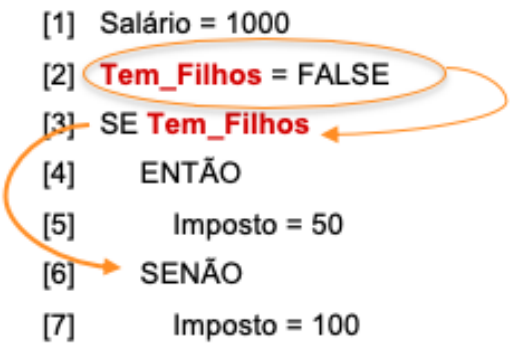
Algoritmo com verificação ***completa***	Algoritmo com verificação ***simplificada***
[1] Salário = 1000 [2] Tem_Filhos = FALSE [3] SE **Tem_Filhos = TRUE** [4] ENTÃO [5] Imposto = 50 [6] SENÃO [7] Imposto = 100 [8] FIM SE [9] Salário_Final = Salário – Imposto	[1] Salário = 1000 [2] Tem_Filhos = FALSE [3] SE **Tem_Filhos** [4] ENTÃO [5] Imposto = 50 [6] SENÃO [7] Imposto = 100 [8] FIM SE [9] Salário_Final = Salário – Imposto

Para entender esta simplificação, vamos rever a estrutura da lógica condicional do IF:

SE <condição> ENTÃO *<código se for verdadeiro>* SENÃO *<código se for falso>* FIM SE	IF <condição> THEN *<código se for verdadeiro>* ELSE *<código se for falso>* END IF

Isto significa que o IF ***sempre busca*** um ***valor verdadeiro*** para executar seu trecho de código contido em THEN, não sendo verdadeiro (***valor falso***) ele passa para o trecho contido em ELSE.

Então, como "Tem_Filhos" é do mesmo tipo (TRUE e FALSE), não precisamos compará-lo com TRUE, basta colocar o nome dele. Como seu conteúdo é um valor FALSO, ele passa direto para o SENÃO.



Com isso entendido, temos então o ***operador NOT*** (NÃO). O funcionamento do operador NOT é bem simples: ele NEGA o conteúdo de um valor booleano, ou seja, se o valor for TRUE, o NOT o transforma em FALSE e vice-versa. Vamos ver como funciona sua ***tabela verdade***:

Valor	NOT Valor
TRUE	FALSE
FALSE	TRUE

Assim, podemos alterar nosso algoritmo para incluir o operador NOT e inverter o fluxo da estrutura IF, vejamos:

Algoritmo simplificado	Algoritmo simplificado com NOT
1] Salário = 1000 [2] Tem_Filhos = FALSE [3] SE Tem_Filhos [4] ENTÃO [5] Imposto = 50 [6] SENÃO [7] Imposto = 100 [8] FIM SE [9] Salário_Final = Salário – Imposto	[1] Salário = 1000 [2] Tem_Filhos = FALSE [3] SE NOT Tem_Filhos [4] ENTÃO [5] Imposto = 50 [6] SENÃO [7] Imposto = 100 [8] FIM SE [9] Salário_Final = Salário – Imposto

Neste exemplo, como o valor de “Tem_Filhos” é FALSE, ele então vai para o SENÃO. Quando usamos o NOT antes de “Tem_Filhos”, invertemos seu valor que passa a ser TRUE. O Valor TRUE é reconhecido pelo IF, que passa o fluxo para o ENTÃO.

Bacana, né? Vamos estudar isso na prática agora.