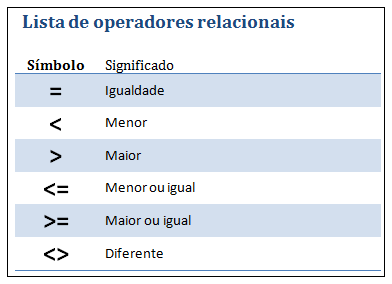
**Operadores Relacionais**

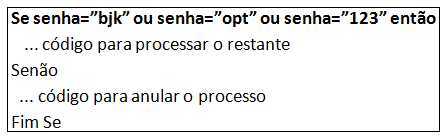
Em programação, devemos sempre combinar os operadores relacionais:



em conjunto com os **operadores lógicos (OR, AND, NOT, etc)**

**Importante**

Nos exemplos do operador lógico OR, nós sempre fizemos com apenas 2 condições. Nada impede que você construa códigos maiores, como por exemplo:

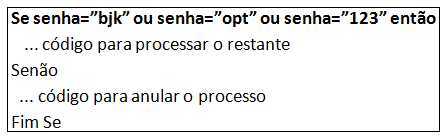


Note que esse exemplo tem 3 condições sendo testadas. A tabela verdade do operador OR vai continuar sendo igual. Se acontecer de 1 das condições serem verdadeiras, todo o resultado já é VERDADEIRO. Só não será VERDADEIRO se as 3 condições testadas resultarem em Falso.

**Operador lógico do tipo *AND***

**Importante**

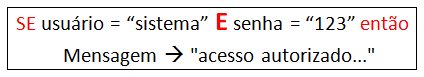
Nos exemplos do operador lógico OR, nós sempre fizemos com apenas 2 condições. Nada impede que você construa códigos maiores, como por exemplo:



# Note que esse exemplo tem 3 condições sendo testadas. A tabela verdade do operador OR vai continuar sendo igual. Se acontecer de 1 das condições serem verdadeiras, todo o resultado já é VERDADEIRO. Só não será VERDADEIRO se as 3 condições testadas resultarem em Falso.

# Explicação

A leitura da condicional fica assim:



Nesse caso, o resultado **somente será verdadeiro**, caso as 2 condições sejam atendidas.

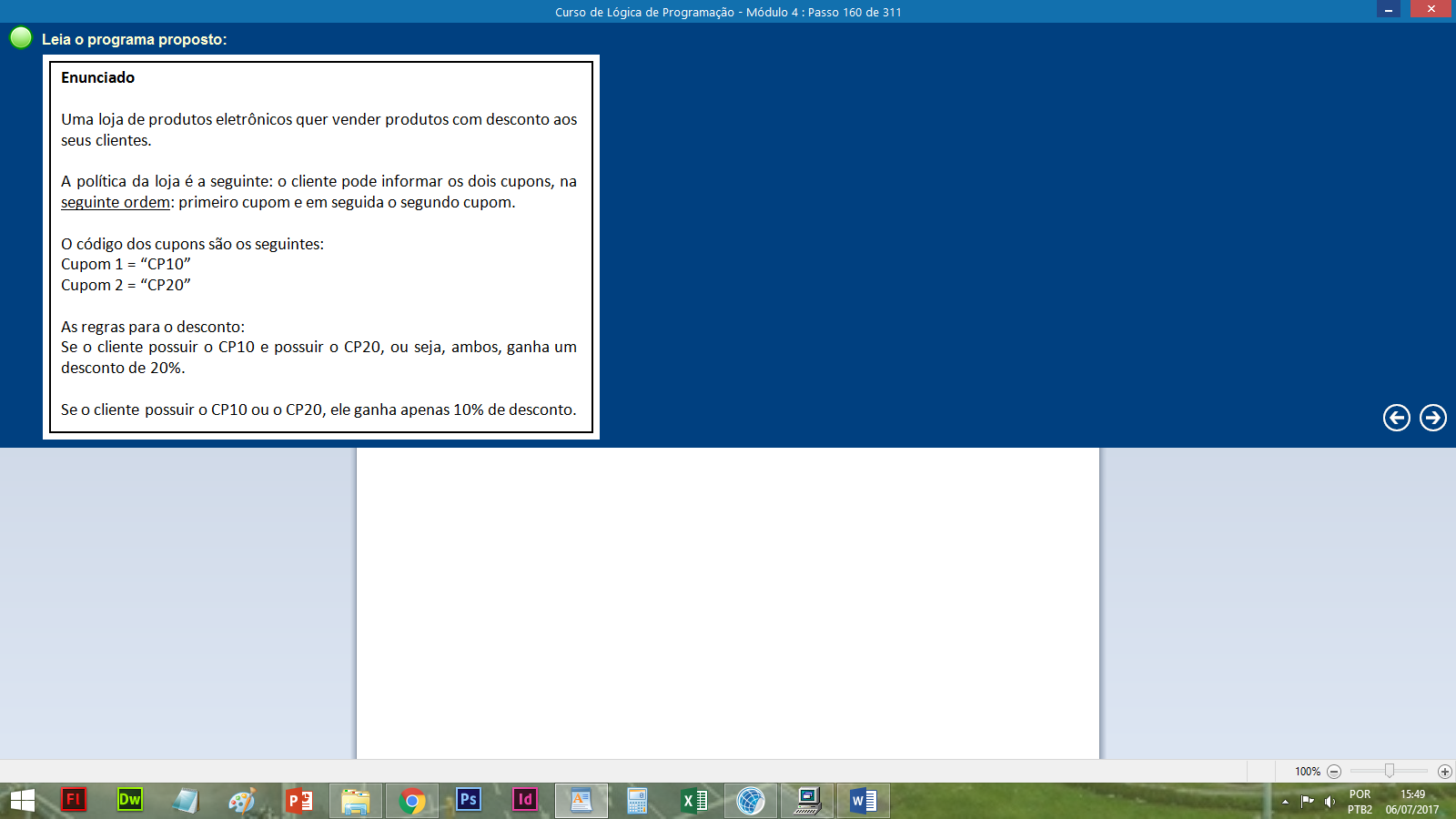
**Diferença AND e OR**

A diferença entre operadores AND e OR é o tipo de resultado que você precisa.

Com o operador AND, ambos os operandos vão ter que retornar como VERDADEIROS.

Já no operador OR, apenas um dos operandos testados é que precisam retornar VERDADEIRO.

**Praticando operadores lógicos**



**Explicação**

Nesse exemplo de algoritmo, iremos contruir 2 tipos de condicionais. Uma delas irá utilizar o operador lógico do tipo AND (e) e a outra condicional irá utilizar o operador do tipo OR (ou).

Com isso poderemos testar se: o cliente possui **ambos** os cupons de desconto, **OU** se possui apenas um deles. Também será testado se o cliente *não possui nenhum* cupom.

**Explicação**

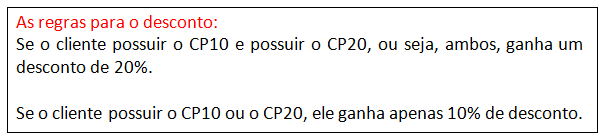
As variáveis:  
**cupom1** e **cupom2** são do tipo caractere, irão armazenar os cupons que o usuário digitar.

As variáveis:  
**preco\_produto** e **desconto** são do tipo real, pois vão envolver valores monetários."

**relembrando o sinal “<-” indica uma atribuição**

**Explicação**

**O próximo passo do algoritmo será criar as condicionais, de acordo com as regras:**



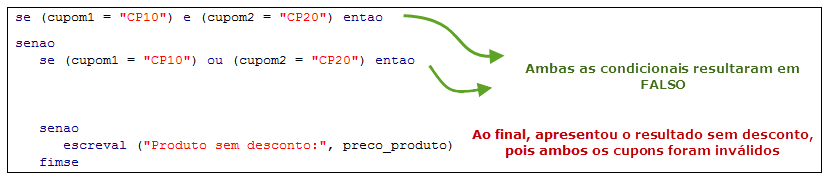
**Explicação**

Primeiro, a variável desconto passa a valer 0.20 (20 % de desconto -> ou seja, 20/100)

Em seguida, a variável desconto recebe o seguinte cálculo: **preco\_produto \* desconto** ou seja 200 \* 0,20 = **40.00**

40 será o valor do desconto do cliente

**Confira a Explicação**



**Explicação**

O programa iniciou pela condicional:

C:\Program Files\MetodoCGD\Figuras\logicadeprogramacao\log0614.png

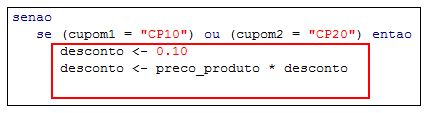
Nesse teste, o resultado da condicional foi FALSO, pois o **cupom1** tinha o valor "nada", logo, a condição que utiliza o operador do tipo "E", retornou falso.

Passou então para a seguinte condicional:

C:\Program Files\MetodoCGD\Figuras\logicadeprogramacao\log0615.png

Cupom1 é igual a "CP10"? -> resultou em Falso.  
Cupom2 é igual a "CP20"? -> resultou em **VERDADEIRO**.

Como a condicional é do tipo "OU", apenas um deles precisa ser verdadeiro, logo, foi calculado 10% de desconto:



**Explicação**

**O algoritmo está incorreto!**

Foi informado como o primeiro cupom, o código "cp20" e mesmo assim retornou "Produto sem desconto".

Leia novamente as regras do programa:

C:\Program Files\MetodoCGD\Figuras\logicadeprogramacao\log0620.png

De acordo com a regra do programa, se o cliente informar CP10 **ou** CP20, ele tem direito ao desconto.  
No exemplo foi informado CP20 para um deles, e mesmo assim, **não forneceu** o desconto.

**Explicação**

**O algoritmo está incorreto!**

Nesse teste, o usuário possui ambos os cupons de desconto -> CP10 E CP20

Leia novamente as regras do programa:

C:\Program Files\MetodoCGD\Figuras\logicadeprogramacao\log0632.png

De acordo com a regra do programa, se o cliente informar ambos os cupons, ele tem direito ao desconto máximo (20%). No exemplo, só calculou na primeira vez. Isso aconteceu pois existe um erro na condicional do tipo "E".

# 

# Explicação

Agora não importa mais ordem em que o usuário informa os cupons.

O programa está reconhecedo se:

-> O usuário possui **ambos** os cupons (operador lógico do tipo "AND" (e))  
ou  
-> O usuário **pelo menos um** dos cupons, ou um ou outro... (operador lógico do tipo "OR" (ou))  
ou  
-> Se o usuário não possui nenhum cupom válido, apresentando a mensagem de produto sem desconto.

**OPERADOR LÓGICO DO TIPO *NOT***

**Operador Lógico NOT**

Um operador lógico do tipo **NOT** é utilizado para realizar a negação de um valor. Ele não possui uma tabela verdade aplicável, pois é mais simples. Esse operador apenas faz a negação de um valor, por exemplo:

Operador A tem o valor "Verdadeiro" (true)

Se aplicar o operador NOT... ficaria assim: **Not Operador A** *= Falso* (false)

Ele sempre retorna o contrário, vejamos outro caso.

A agora tem o valor "false". Se utilizarmos: NOT A, então A passa a ter o valor "true".