



Desenvolvimento de Sistemas

Condicionais: lógica booliana, estrutura condicional simples e composta

Antes de conseguir programar diversas linhas de código, é necessário que você tenha uma base de **fundamentos teóricos** para que entender esse incrível mundo da programação. Essa base envolve **algoritmos e lógica de programação**.

Você aprendeu que a lógica de programação é o ponto-chave para saber programar e que algoritmos são uma sequência finita de passos que pode ser aplicada a qualquer coisa do mundo real. Também já sabe a importância de identificar um dado e seu tipo para saber o real valor que esse dado ocupa na memória a fim de declará-lo da forma correta. Com esses temas bem definidos, neste conteúdo você compreenderá o **fundamento** da lógica booliana.

Para praticar, monte um algoritmo no quebra-cabeça a seguir. Você pode utilizar seus conhecimentos ou tentar realizar o desafio por dedução.



Lógica booleana

Se os computadores trabalham apenas com números, então como eles conseguem executar tantas tarefas que antes eram somente executadas por humanos? Diante dessa reflexão, fica uma pergunta: o computador trabalha apenas com números? Essa afirmação não está certa nem errada.

A lógica booleana, também chamada de álgebra booleana, é uma forma de representação lógica com o uso de equações matemáticas. Antes da apresentação da lógica booleana com o uso de equações matemáticas, você verá exemplos usando situações do cotidiano.

Exemplo 1: “Está chovendo?”

Se essa pergunta for uma variável, ela pode ser verdadeira e falsa, dependendo do clima. A linguagem da máquina, para essa situação, tem duas alternativas, que são verdadeiro (representado pelo número 1) e falso (representado pelo número 0). Esses valores são chamados de valores booleanos ou valores lógicos.

Exemplo 2: “Devo Dormir?”

Agora a resposta será modificada, ainda com duas opções, uma verdadeira e uma falsa, porém mais extensa, usando os mesmos valores.

Verdadeiro = “Sim, estou cansado”

Falso = “Não, ficarei acordado”

A mensagem que o usuário verá está de acordo com a opção escolhida, mas, para a linguagem do computador, ainda segue sendo 1 para verdadeiro e 0 para falso.

Operadores relacionais

Dentro da lógica de programação, há os **operadores relacionais**. Eles podem ser utilizados para qualquer linguagem de programação. Quem, por exemplo, já utilizou algum programa de planilha eletrônica não terá dificuldade em entendê-los.

Esses operadores são utilizados para comparar **valores** e **também expressões**, e seu resultado é um valor lógico (falso ou verdadeiro).

São eles:

Operador	Símbolo Matemático	Exemplo
Igual	=	$X = Y$
Maior que	>	$X > Y$
Menor que	<	$X < Y$
Maior ou igual a	\geq	$X \geq Y$
Menor ou igual a	\leq	$X \leq Y$
Diferente de	\neq	$X \neq Y$

Você entenderá um pouco melhor cada um e verá alguns exemplos também, aplicando-os à linguagem de algoritmo Portugol.

Operador Igual (==)

Usado para **saber se algum valor ou expressão é igual a outro**. Na matemática e em algumas linguagens de programação, usa-se o símbolo =, mas em Portugol e em linguagens, como Java, C++, C# e Javascript, usa-se == para comparar igualdade porque o símbolo = já é utilizado para atribuições.

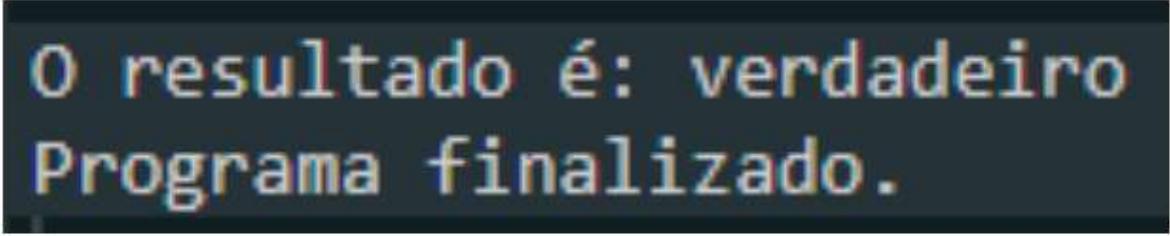
É importante notar que a igualdade será falsa quando os tipos entre os lados comparados são diferentes.

Por exemplo, ao comparar a cadeia de caracteres “3” com o número 3 (“3” == 3), resultará em falso, pois o número inteiro 3 é diferente da cadeia de caracteres “3”. No exemplo a seguir, há dois tipos de dados diferentes, inteiro e cadeia, que não podem ser comparados.

Veja:

```
programa {
    funcao inicio() {
        escreva ("O resultado é: " , 3 == 3)
    }
}
```

Esse exemplo compara se o número 3 é igual ao número 3. Então o resultado será:

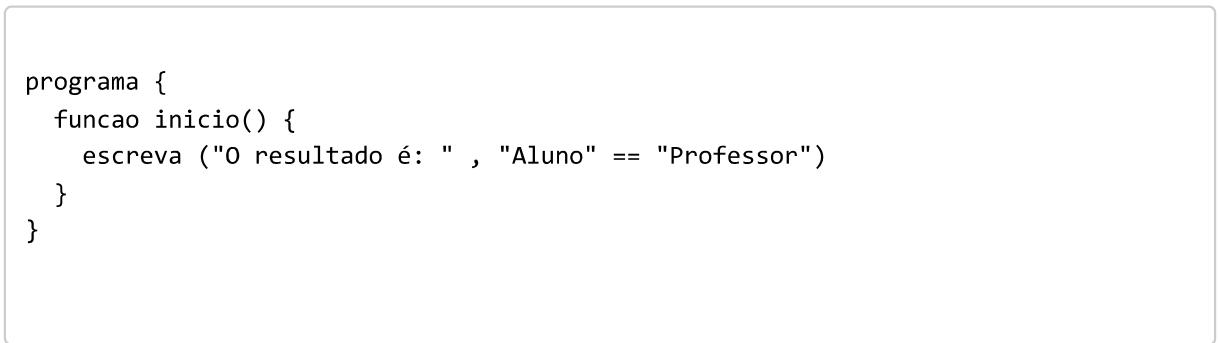


O resultado é: verdadeiro
Programa finalizado.

Figura 1 – Resultado da execução do código anterior

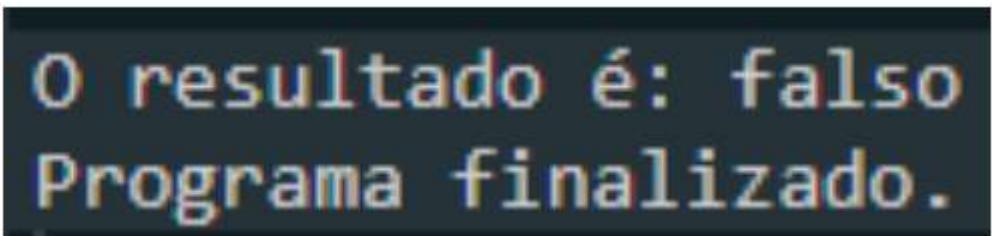
Mensagem “O resultado é verdadeiro”, que é o resultado do primeiro exemplo, que está comparando números inteiros. Abaixo há a mensagem “Programa finalizado”.

Também é possível comparar expressões, como neste exemplo:



```
programa {
    funcao inicio() {
        escreva ("O resultado é: " , "Aluno" == "Professor")
    }
}
```

No exemplo, a cadeia de caracteres “Aluno” é comparada com a cadeia de caracteres “Professor”. Como as palavras são diferentes, o resultado será:



O resultado é: falso
Programa finalizado.

Figura 2 – Resultado do código com a comparação “Aluno”== “Professor”

Mensagem “O resultado é falso”, que é o resultado do segundo exemplo, em que se está comparando cadeia de caracteres. Abaixo há a mensagem “Programa finalizado”.

Ao usar o comparador de igualdade em expressões, deve-se ter cuidado com espaços, pois cada espaço dentro das aspas é contado como um caractere. Veja o exemplo:

```
funcao inicio() {  
    escreva ("O resultado é: " , "Aluno" == "Aluno ")  
}
```

Aparentemente o exemplo acima seria verdadeiro, pois há duas palavras “Aluno” escritas igualmente. Porém, na segunda palavra “Aluno”, há um espaço a mais no final, e o sistema verifica somente as expressões idênticas, então, nesse caso, o resultado é:

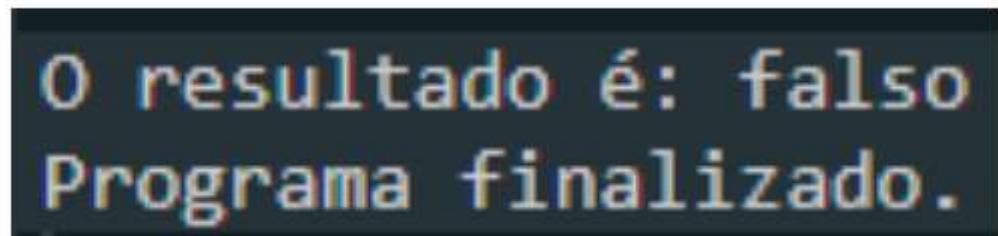


Figura 3 – Resultado do código com a comparação “Aluno” == “Aluno ”

Mensagem “O resultado é falso”, que é o resultado do terceiro exemplo, em que se está comparando cadeia de caracteres. Abaixo é exibida a mensagem “Programa finalizado”.

Ao apagar o espaço após a letra “o”, conforme a seguir, o resultado será verdadeiro.

```
escreva ("O resultado é: " , "Aluno" == "Aluno")
```

Em todo caso, observe que foi usado “==”, pois ao usar somente um “=”, é atribuído algo, por exemplo, a uma variável. Isso é especialmente importante quando variáveis são comparadas, como no exemplo a seguir:

```
funcao inicio() {  
    real nota;  
    leia(nota)  
    escreva("Nota é igual a 10? ", nota == 10)  
}
```

Nesse exemplo, se o usuário digitar o número 10, o resultado será “Nota é igual a 10? Verdadeiro”; caso o usuário digite um número que não seja 10, a frase emitida será “Nota é igual a 10? Falso”. Experimente então retirar um dos símbolos de = na expressão:

```
escreva("Nota é igual a 10? ", nota == 10)
```

Execute e note que o resultado será algo como: “Nota é igual a 10? 10.0”. Por que isso acontece?

Basicamente porque, em vez de realizar uma comparação, está sendo realizada uma atribuição – ou seja, a variável nota está recebendo o valor 10. Como valor de retorno para o texto de escreva(), a atribuição resulta no próprio valor atribuído à variável, ou seja, em vez de “verdadeiro”, imprime “10.0”.

É importante ter atenção ao uso do símbolo para não cair em pequenas armadilhas como essa.

Além de comparações entre valores constantes, é claro que você pode comparar duas variáveis, como no exemplo:

```
funcao inicio() {  
    real nota1, nota2  
    leia(nota1)  
    leia(nota2)  
    escreva("As notas são iguais? ", nota1 == nota2)  
}
```

A expressão resultará em verdadeiro apenas se o usuário digitar exatamente o mesmo número nas duas atribuições do programa.

Operador maior que (>)

Esse operador fará a **comparação se um valor ou expressão é maior que o outra**. Veja alguns exemplos:

```
funcao inicio() {  
    escreva ("O resultado é: " , 5>3)  
}
```

No exemplo acima, é verificado se o número 5 é maior que o número 3. Como, de fato, o número 5 é maior que o número 3, então o resultado é:

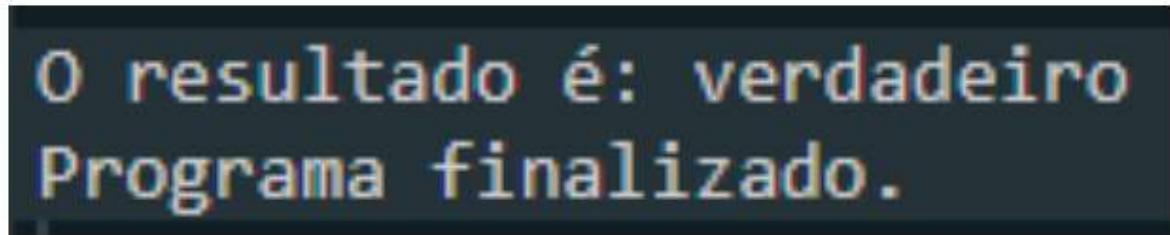


Figura 4 – Resultado do código com comparação “5>3”

Mensagem “O resultado é verdadeiro”, que é o resultado do primeiro exemplo, que está comparando números inteiros. Abaixo é exibida a mensagem “Programa finalizado”.

Também é possível armazenar números ou expressões dentro de uma variável e realizar a comparação, veja:

```
funcao inicio() {  
    inteiro numero1 = 10  
    inteiro numero2 = 15  
    escreva ("O resultado é: " , numero1 > numero2)  
}
```

Nesse exemplo, duas variáveis são declaradas como valores fixados: variável numero1 do tipo inteiro com o valor 10 e variável numero2 do tipo inteiro com o valor 15. Após realizar a comparação usando os nomes das variáveis, o resultado é:

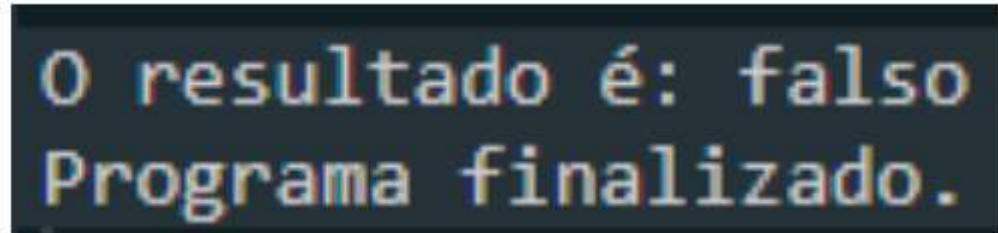


Figura 5 – Resultado do código com comparação “numero1 > numero2”

Mensagem “O resultado é falso”, que é o resultado do segundo exemplo comparando duas variáveis do tipo inteiro com seus valores definidos. Abaixo é exibida a mensagem “Programa finalizado”.

Deve-se ter cuidado, por exemplo, se for dito que uma pessoa só pode dirigir se tiver 18 anos, não usar a expressão dessa maneira: $idade > 18$, pois diz que a idade deve ser maior que 18, ou seja, a partir dos 19 anos, usando o tipo de dado inteiro. Veja um exemplo:

```
funcao inicio() {  
    inteiro idade = 18  
    escreva ("Tenho 18 anos, posso dirigir? ", idade > 18)  
}
```

Foi declarada uma variável do tipo inteiro com valor de 18 e realizada a comparação. O maior sempre usará como referência o próximo número, então, nesse exemplo, para poder dirigir, a pessoa teria que ter 19 anos.

A resposta é:

```
Tenho 18 anos, posso dirigir? falso
Programa finalizado.
```

Figura 6 – Resultado do código com a comparação “idade > 18”

Mensagem “Tenho 18 anos, posso dirigir? Falso”, que é o resultado do exemplo em que se compara a variável `idade`, do tipo inteiro, com o valor 18, se ela é $>$ (maior) que 18. Abaixo é exibida a mensagem “Programa finalizado”.

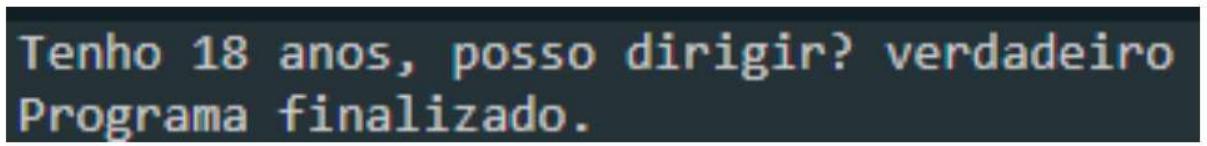
Caso o exemplo acima fosse verdadeiro, sem alterar o valor da variável, seria possível usar o operador maior ou igual que. Esse operador será estudado a seguir.

Operador maior ou igual que (\geq)

Sua função é muito **semelhante ao operador maior que (>)**, porém a **comparação iniciará a partir do número que está sendo comparado**. Exemplo: ≥ 18 compara a partir do 18. Veja o exemplo estudado anteriormente:

```
funcao inicio() {  
    inteiro idade = 18  
    escreva ("Tenho 18 anos, posso dirigir? ", idade $\geq$  18)  
}
```

A única mudança feita nesse exemplo em relação ao anterior foi que, ao final da linha do “escreva”, mudou-se a expressão para “idade ≥ 18 ”. A resposta agora será verdadeiro, conforme abaixo:



```
Tenho 18 anos, posso dirigir? verdadeiro  
Programa finalizado.
```

Figura 7 – Resultado do código com a comparação “idade ≥ 18 ”

Mensagem “Tenho 18 anos, posso dirigir? Verdadeiro”, que é o resultado do exemplo comparando a variável idade, do tipo inteiro, com valor 18, se ela é \geq (maior ou igual) que 18. Abaixo é exibida a mensagem “Programa finalizado”.

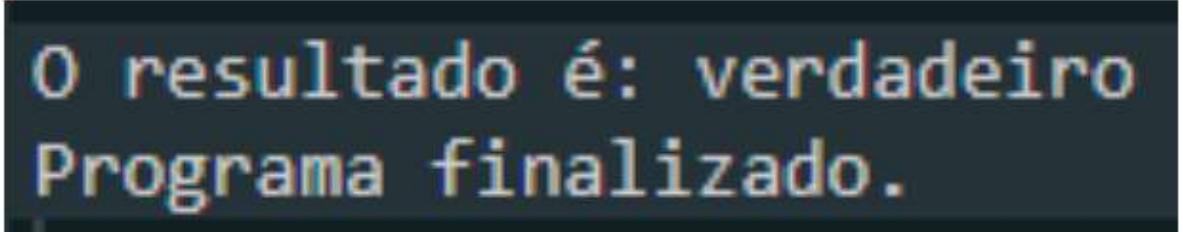
Note que, implicitamente, foi usado o operador lógico “OU”. Para a expressão ser verdadeira, então a variável deve ser maior que 18 “OU” igual a 18.

Operador menor que (<)

Este operador tem a sua função ao contrário do operador maior que. Ele fará a **comparação se um valor ou expressão é menor que o outra**. Veja alguns exemplos:

```
programa {
    funcao inicio() {
        escreva ("O resultado é: " , 10<15)
    }
}
```

O exemplo acima segue a mesma lógica dos demais, porém com o operador menor (<). Está sendo testado se o número 10 é menor que o número 15, e o resultado será:



O resultado é: verdadeiro
Programa finalizado.

Figura 8 – Resultado do código com a comparação “10 < 15”

Mensagem “O resultado é verdadeiro”, que é o resultado do primeiro exemplo comparando números inteiros. Abaixo é exibida a mensagem “Programa finalizado”.

```
funcao funcao inicio() {
    inteiro numero1 = 5
    inteiro numero2 = 2

    escreva ("O resultado é: " , numero1 < numero2)
}
```

No exemplo acima, duas variáveis do tipo inteiro são declaradas: numero1 com valor de 5 e numero2 com valor de 2. A comparação verificará se a variável numero1 é menor que a variável numero2. O resultado é:

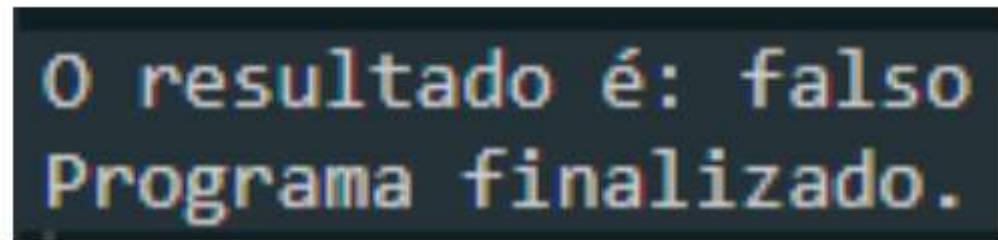


Figura 9 – Resultado do código com a comparação “numero1 < numero2”

Mensagem “O resultado é falso”, que é o resultado do segundo exemplo comparando duas variáveis do tipo inteiro com seus valores definidos. Abaixo é exibida a mensagem “Programa finalizado”.

Operador menor ou igual que (\leq)

Sua função é muito **semelhante ao operador menor que (<)**, porém a **comparação iniciará a partir do número que está sendo comparado**. Veja alguns exemplos:

```
programa {
    funcao inicio() {
        escreva ("O resultado é: " , 8<= 8)
    }
}
```

Esse exemplo compara se o número 8 é menor ou igual a 8. Quando o operador “=” é usado, a verificação inicia a partir do número. Exemplo: `<=8` realizará a comparação do número 8 para baixo. A resposta é:

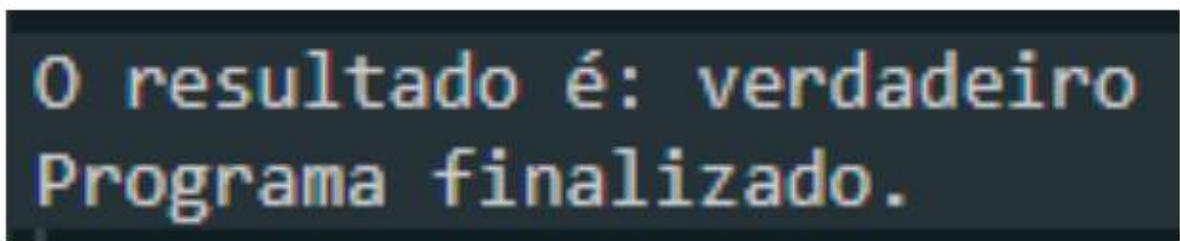


Figura 10 – Resultado do código com a comparação “8 <= 8”

Mensagem “O resultado é verdadeiro”, que é o resultado do segundo exemplo comparando se o número 8 é menor ou igual ao número 8. Abaixo é exibida a mensagem “Programa finalizado”.

Operador diferente (!=)

A função do operador diferente “!=” (símbolo matemático \neq) também é de fácil entendimento. O operador **verificará se um valor ou expressão é diferente de outro**. Veja alguns exemplos:

```
programa {
    funcao inicio() {

        escreva ("O resultado é: " , 2 != 1 )

    }
}
```

Esse exemplo testa se o número 2 é diferente do número 1. O resultado é:

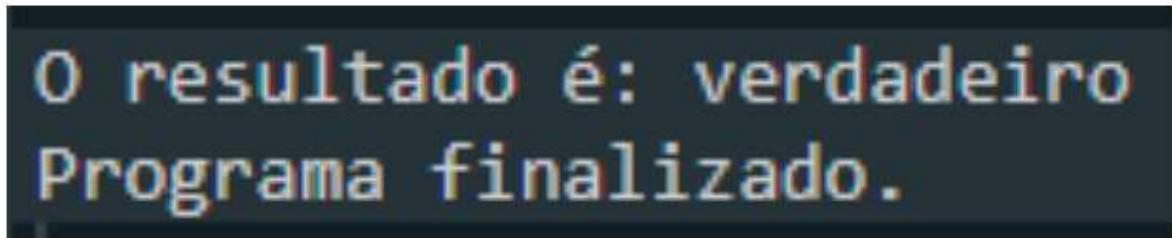


Figura 11 – Resultado do código com a comparação “2 != 1”

Mensagem “O resultado é verdadeiro”, que é o resultado do primeiro exemplo comparando se o número 2 é diferente do número 1. Abaixo é exibida a mensagem “Programa finalizado”.

```
programa {
    funcao inicio() {

        cadeiapalavra1 = "Informática"
        cadeiapalavra2 = "Tecnologia"

        escreva ("O resultado é: " , palavra1 != palavra2)

    }
}
```

Nesse exemplo há duas variáveis do tipo cadeia: palavra1 com a expressão “Infor-mática” e palavra2 com a expressão “Tecnologia”. Será realizada a comparação das duas para saber se são diferentes uma da outra. O resultado é:

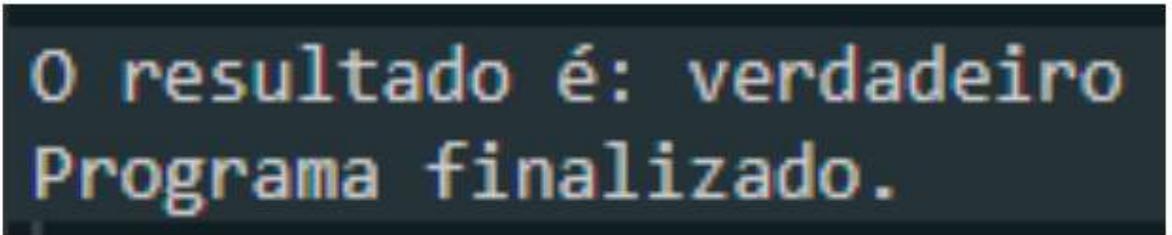


Figura 12 – Resultado do código com a comparação “palavra1 != palavra2”

Mensagem “O resultado é verdadeiro”, que é o resultado do segundo exemplo comparando se a variável palavra1, que contém a expressão “Informática”, é diferente da variável palavra2, que contém a expressão “Tecnologia”. Abaixo é exibida a mensagem “Programa finalizado”.

Equações booleanas

Para compreender melhor o que são **equações booleanas**, é possível compará-las com equações matemáticas. A diferença é que, em vez de usar operações básicas (adição, subtração, multiplicação e divisão), são usados operadores próprios para as equações booleanas.

Veja os 7 operadores lógicos, que são: E(AND), OU(OR), NÃO(NOT), NÃO-E(NAND), NÃO-OU(NOR), OU-EXCLUSIVO(XOR) e NÃO-OU-EXCLUSIVO(XNOR).

Perceba que todos os operadores lógicos estão ou podem estar presentes no seu dia a dia. Você pode colocar a lógica booleana em prática nas situações do cotidiano.

Operador E(AND)

O operador “E” ou “AND” é um dos mais usados e resultará em um VALOR VER-DADEIRO se os DOIS VALORES de entrada da operação forem VERDADEIROS. Se a alternativa for contrária, então o resultado será FALSO.

VALOR 1	VALOR 2	OPERAÇÃO E
Verdadeiro	Verdadeiro	Verdadeiro
Verdadeiro	Falso	Falso
Falso	Verdadeiro	Falso
Falso	Falso	Falso

Veja a explicação da tabela-verdade:

1. VALOR 1 = Verdadeiro E(AND) VALOR 2 = Verdadeiro, então o resultado se-rá VERDADEIRO.
2. VALOR 1 = Verdadeiro E(AND) VALOR 2 = Falso, então o resultado será FALSO.
3. VALOR 1 = Falso E(AND) VALOR 2 = Verdadeiro, então o resultado será FALSO.
4. VALOR 1 = Falso E(AND) VALOR 2 = Falso, então o resultado será FALSO.

Pode-se chegar à conclusão de que o operador E(AND) só terá uma resposta VER-DADEIRA, que será quando TODOS os valores forem VERDADEIROS.

Exemplo 1: Trabalhei ontem E(AND) amanhã vou trabalhar?

Essa pergunta só será verdadeira se a pessoa trabalhou no dia anterior e trabalhará no próximo dia.

Exemplo 2: Maior de idade E(AND) possui carteira de habilitação?

A pergunta acima só será verdadeira se a pessoa tem mais de 18 anos e possui carteira de habilitação.

Em Portugol, é usado o operador “e” em expressões lógicas, conforme o seguinte exemplo:

```
funcao inicio() {  
    escreva ("O resultado é: " , 5 == 5 e 4 == 8 , ", devido a primeira expressão ser ", 5 == 5 , " e a segunda expressão ser ", 4 == 8)  
}
```

Nesse exemplo, o operador lógico “E” e o operador relacional “Igual” são usados na seguinte comparação: o número 5 é igual a 5 E o número 4 é igual a 8. Para que a expressão com o operador “E” seja verdadeira, todos os resultados das expressões devem ser verdadeiros. No primeiro caso ($5 == 5$), o resultado é verdadeiro, porém, no segundo caso ($4 == 8$), o resultado é falso. Então a resposta final será:

```
O resultado é: falso, devido a primeira expressão ser verdadeiro e a segunda expressão ser falso  
Programa finalizado.
```

Figura 13 – Resultado do código com a comparação com operador lógico “e”

Mensagem “O resultado é: falso, devido a primeira expressão ser verdadeiro e a segunda expressão ser falso”, em que é comparado se o número 5 é igual ao número 5 e o número 4 é igual a 8. Uma resposta é verdadeira e a outra é falsa. Abaixo é exibida a mensagem “Programa finalizado”.

Outros exemplos de expressões com “e”:

```
funcao inicio() {  
    real nota1, nota2  
    leia(nota1)  
    escreva("Nota está entre 6 e 9? ", nota1 >= 6 e nota1 <= 9)  
    escreva("\nNota está entre 0 e 6 mas é diferente de 3? ", nota1 >= 0 e nota1 <= 6 e nota1 != 3)  
    leia(nota2)  
    escreva("\nNota1 é maior que a nota2 e são diferentes? ", nota1 > nota2 e nota1 != nota2)  
}
```

Operador OU(OR)

O operador “OU” ou “OR” resultará em um valor VERDADEIRO se pelo menos UM ou MAIS VALORES de entrada da operação forem VERDADEIRO, se a alternativa for contrária, o resultado será FALSO.

Esse operador é muito usado em conjunto com o operador E(AND).

VALOR 1	VALOR 2	OPERAÇÃO E
Verdadeiro	Verdadeiro	Verdadeiro
Verdadeiro	Falso	Verdadeiro
Falso	Verdadeiro	Verdadeiro
Falso	Falso	Falso

Veja a explicação da tabela-verdade:

1. VALOR 1 = Verdadeiro OU(OR) VALOR 2 = Verdadeiro, então o resultado se-rá VERDADEIRO.
2. VALOR 1 = Verdadeiro OU(OR) VALOR 2 = Falso, então o resultado será VERDADEIRO.
3. VALOR 1 = Falso OU(OR) VALOR 2 = Verdadeiro, então o resultado será VERDADEIRO.
4. VALOR 1 = Falso OU(OR) VALOR 2 = Falso, então o resultado será FALSO.

Se, em uma expressão, você verificar se as condições são verdadeiras ou falsas com o operador OU(OR), e entre elas haver uma opção verdadeira, o resultado será VERDADEIRO.

Exemplo 1: Comprar um carro OU(OR) alugar um carro?

A afirmação acima só será falsa caso as duas opções sejam falsas. Em caso contrário, sempre será VERDADEIRO.

Exemplo 2: Maior de idade OU(OR) possui carteira de habilitação?

Usando a mesma pergunta do E(AND), porém agora com OU(OR), é possível afirmar que essa pergunta só será falsa se os dois critérios forem falsos. Em caso contrário, sempre será VERDADEIRO.

Em Portugol, usa-se a própria palavra “ou” como operador lógico. Veja o exemplo.

```
funcao inicio() {  
    escreva ("O resultado é: " , 5 == 5 ou 4==8 , ", devido a primeira expressão ser ", 5 == 5 , " e a segunda expressão ser ", 4 == 8)  
}
```

Esse exemplo tem como base o exemplo 3 do operador “E”, porém agora será verificado se o número 5 é igual a 5 OU o número 4 é igual a 8. Conforme a explicação do operador lógico “OU”, se uma expressão for verdadeira, o resultado será a palavra verdadeiro, conforme segue:

```
O resultado é: verdadeiro, devido a primeira expressão ser verdadeiro e a segunda expressão ser falso  
Programa finalizado.
```

Figura 14 – Resultado do código com a comparação com operador lógico “ou”

Mensagem “O resultado é: verdadeiro devido à primeira expressão ser verdadeiro e a segunda expressão ser falso”, em que é comparado se o número 5 é igual ao número 5 e o número 4 é igual a 8. Há uma resposta verdadeira e outra falsa. Abaixo é exibida a mensagem “Programa finalizado”.

Veja outros exemplos usando “ou”:

```
funcao inicio() {  
    inteiro idade  
    leia(idade)  
    escreva("Idade está fora da faixa entre 3 e 7 anos? ", idade < 3 ou idade > 7 )  
}
```

Operador NÃO (NOT)

O uso desse operador é simples de ser compreendido. Sua função é de inversão de valores, ou seja, se o valor de entrada for VERDADEIRO, o resultado será FALSO e vice-versa.

VALOR 1	OPERAÇÃO NÃO
Verdadeiro	Falso
Falso	Verdadeiro

Veja a explicação da tabela-verdade:

1. Se o valor for verdadeiro, o resultado é falso.
2. Se o valor for falso, o resultado é verdadeiro

Exemplo: Estou com fome?

Nesse caso, se a resposta for verdadeiro, o operador inverterá para falso e vice-versa.

Em Portugol, usa-se a palavra “nao”, sem acento, para operações de negação. Veja o exemplo a seguir.

```
funcao inicio() {  
    logico booleano  
    inteiro numero  
    leia(numero)  
    booleano = numero > 10  
    escreva("\nO resultado original é ", booleano)  
    escreva("\nO resultado inverso é ", nao(booleano))  
    escreva("\nUsando 'nao' com expressao ", nao(numero >= 20))  
}
```

Combinando operadores lógicos

Expressões mais complexas podem ser montadas usando uma combinação entre operadores lógicos. Você precisa tomar cuidado porque, assim como nos operadores matemáticos, existe precedência entre os operadores lógicos. Veja a expressão a seguir em Portugol. Imagine que um estabelecimento oferece gratuidade, em seus serviços, a crianças menores de 5 anos e idosos maiores de 70 anos desde que estejam acompanhados (em ambos os casos). Pense no seguinte trecho de código:

```
inteiro idade  
logico estaAcompanhado  
leia(idade)  
leia(estaAcompanhado)  
escreva("Resultado: ", idade < 5 ou idade > 70 e estaAcompanhado)
```

Experimente alguns valores:



- ◆ Idade = 77 e estaAcompanhado = verdadeiro: a expressão resulta verdadeiro, o que é correto (idoso acompanhado)
- ◆ Idade = 20 e estaAcompanhado = verdadeiro: a expressão resulta em falso, o que é correto (não é idoso nem criança, não tem direito à gratuidade)
- ◆ Idade = 3 e estaAcompanhado = verdadeiro resultam em verdadeiro, o que é correto (criança acompanhada)
- ◆ Idade = 3 e estaAcompanhado = falso: a expressão resulta em verdadeiro o que é **errado!** (a criança teria que estar acompanhada)

A falha que acontece nesse último caso se dá por conta da precedência entre **e** e **ou**. Em uma expressão, **e** é avaliado antes de **ou**.

Assim, na expressão de exemplo `idade < 5 ou idade > 70 e estaAcompanhado`, a primeira comparação a ser realizada é `idade > 70 e estaAcompanhado`; o resultado dessa expressão é avaliado em seguida com o ou: `idade < 5 ou resultado-da-expressao`. Se fossem usados os valores `idade = 3 e estaAcompanhado = falso`, o resultado seria: `idade > 70 e estaAcompanhado → 3 > 70 e falso → falso`

Reincorporando isso à expressão: `idade < 5 ou idade > 70 e estaAcompanhado → 3 < 5 ou falso → verdadeiro` (pois `3 < 5`).

Para forçar a expressão a usar uma determinada ordem de operações, são usados parênteses. Na expressão de exemplo, ficaria: `(idade < 5 ou idade > 70) e estaA-companhado`

Experimente adaptar o trecho anterior de código e teste com diversos valores.

No Portugol, a ordem de precedência entre os operadores é primeiramente o “`nao`”, depois o “`e`”, depois o “`ou`”.

Outros operadores

Os operadores lógicos a seguir não têm representação única dentro do Portugol, mas serão estudados para conhecimento.

NÃO-E (NAND)

É o contrário do operador E(AND), ou seja, se pelo menos um dos valores for falso, ele resultará em verdadeiro. Em resumo, esse operador é uma negação do operador E(AND).

NÃO-OU (NOR)

Para entendimento, usa-se a explicação semelhante do operador NÃO-E(NAND), porém o operador NÃO-OU (NOR) é o contrário do operador OU(OR), ou seja, o resultado é verdadeiro se os dois valores forem falsos. Pode-se dizer que o resultado dele é o operador OU(OR) seguido de um operador NÃO(NOT).

OU-EXCLUSIVO (XOR)

É verdadeiro quando apenas um valor de entrada for VERDADEIRO, ou seja, quando os valores forem diferentes um do outro.

NÃO-OU-EXCLUSIVO (XNOR)

O operador NÃO-OU-EXCLUSIVO (XNOR) é o contrário do OU-EXCLUSIVO (XOR), ou seja, o resultado é verdadeiro apenas quando as duas entradas forem iguais.

Estruturas condicionais

Agora você estudará uma parte importante, as estruturas de condicionais. Como o próprio nome já diz, são estruturas que necessitam de condições para serem testadas. São chamadas também de estruturas de decisão ou desvio condicional, pois, muitas vezes, é preciso desviar a execução do programa segundo alguma condição.

Por exemplo: devo ir trabalhar de carro ou de moto? Para responder a essa pergunta, as condições devem ser testadas com operadores lógicos e relacionais.

Estrutura condicional simples

Sua função é executar um ou vários comandos para testar se a condição é verdadeira ou falsa. Caso seja falsa, a estrutura é finalizada sem executar os comandos. Para que a execução de um algoritmo seja desviada para uma outra ação, é necessário um comando de desvio. Para iniciar essa estrutura, executa-se o comando usando as palavras reservadas “se” e “fim se”. Dentro desse bloco de código poderá haver vários comandos de atribuição, operadores lógicos e aritméticos e também novos blocos de desvio condicional (você verá a seguir as estruturas condicionais compostas).



Pseudocódigo	Fluxograma
<pre>se (condicao) { //Instruções a serem executadas se o desvio for verdadeiro }</pre> <p>Nesse caso, o bloco de código somente será executado se a condição for verdadeira.</p>	<p>Principal</p> <p>Se</p> <p>Falso Verdade</p> <p>Escrever caso a condição for verdadeiro, irá ser executada essa mensagem e se for falso, o programa será finalizado</p> <p>Fim</p> <p>(objetos/figura_15.png)</p> <p>Figura 15 – Fluxograma</p>

Agora veja alguns exemplos práticos:

Exemplo: média de três notas

Dadas 3 notas informadas pelo usuário, calcular a média e mostrar na tela se o aluno foi aprovado. A nota mínima para aprovação é 6.



Fluxograma

```

    graph TD
        Principal([Principal]) --> RealNota1[Real nota1]
        RealNota1 --> RealNota2[Real nota2]
        RealNota2 --> RealNota3[Real nota3]
        RealNota3 --> EscreverPrimeira[Escrever "Digite a primeira nota"]
        EscreverPrimeira --> LerNota1[Ler nota1]
        LerNota1 --> EscreverSegunda[Escrever "Digite a segunda nota"]
        EscreverSegunda --> LerNota2[Ler nota2]
        LerNota2 --> EscreverTerceira[Escrever "Digite a terceira nota"]
        EscreverTerceira --> LerNota3[Ler nota3]
        LerNota3 --> RealMedia[Real media]
        RealMedia --> CalculaMedia["media <- (nota1 + nota2 + nota3) / 3"]
        CalculaMedia --> EscreverMedia[Escrever "A média é " & media]
        EscreverMedia --> Decision{media >= 7}
        Decision -- Falso --> Fim([Fim])
        Decision -- Verdade --> Aprovado[Escrever "Você está APROVADO"]
        Aprovado --> Fim
    
```

Figura 16 – Fluxograma

```

programa
{
    inclua bibliotecaMatematica --> mat

    funcao inicio()
    {
        /*declaração das variáveis */
        realnota1, nota2, nota3, media

        escreva("\n")
        escreva("Digite a primeira nota: ")
        leia(nota1)
        escreva("Digite a segunda nota: ")
        leia(nota2)
        escreva("Digite a terceira nota: ")
        leia(nota3)
        /* Calcula a média final do usuário */
        media = (nota1 + nota2 + nota3) / 3
        limpa()
        se (media >= 6)
        {
            escreva("Você está APROVADO", mat.arredondar(media, 2))
        }
    }
}

```

Esse algoritmo pede ao usuário que informe três notas. Logo após, calcula a média final do usuário e exibe uma mensagem informando se ele foi aprovado. Para determinar isso, utiliza “se” para comparar o valor obtido na variável “media” com o valor 6, assumido como o valor mínimo para aprovação.

Exemplo: maior de idade.

```
funcao inicio() {
    inteiro idade
    cadeia nome

    escreva("\nDigite o seu nome: ")
    leia(nome)

    escreva("\nDigite a sua idade: ")
    leia(idade)

    limpa()

    se (idade>= 18)
    {
        escreva(nome, ", sua idade é ", idade, " anos e você é maior de idade! ")
    }
}
```

Esse algoritmo pede ao usuário seu nome e sua idade. Logo após, usa “se” para comparar a idade ao valor 18, exibindo uma mensagem caso a condição seja verdadeira (o usuário digitou de fato uma idade maior ou igual a 18).

Exemplo: dado um número real, determinar se ele é maior que zero (positivo).

```
funcao inicio() {  
    real numero  
  
    escreva("\nDigite um número positivo ou negativo: ")  
    leia(numero)  
  
    limpa()  
  
    se (numero>0)  
    {  
        escreva("Número digitado = " , numero , " é maior que zero!")  
    }  
}
```

Esse algoritmo pede ao usuário que informe um número, que pode ser positivo ou negativo. Logo após, ele exibe uma mensagem caso o número digitado seja positivo, mostrando qual número foi digitado e que era maior que zero.

Observe que o { e } foi usado no Portugol para delimitar as instruções que serão executadas de acordo com a condição informada. Note ainda que { e } é usado para a “funcao inicio()” e para “programa”. As chaves no Portugol (e em várias outras linguagens de programação) definem **escopos de código**. Escopo é uma região de código em que podem ser declaradas variáveis que estarão disponíveis dentro do escopo em que foram declaradas, mas nunca fora.

Veja o exemplo a seguir:

```
funcao inicio() {  
    inteiro idade  
    escreva("digite uma idade\n")  
    leia(idade)  
    se(idade >= 18)  
    {  
        cadeia nome  
        escreva("digite um nome\n")  
        leia(nome)  
        escreva(nome, " já é maior de idade\n")  
    }  
}
```

A variável “nome” foi definida dentro da estrutura “se” e pode ser usada tranquila-mente entre as chaves { e } que delimitam o condicional. Usar a variável fora da es-trutura condicional ocasionará erro:

```
funcao inicio() {  
    inteiro idade  
    escreva("digite uma idade\n")  
    leia(idade)  
    se(idade >= 18)  
    {  
        cadeia nome  
        escreva("digite um nome\n")  
        leia(nome)  
        escreva(nome, " já é maior de idade\n")  
    }  
    escreva(nome)  
}
```

A mensagem no Portugol Studio será algo como:

```
ERRO: A variável "nome" não foi declarada neste escopo..
```

Isso indica que a variável não existe fora do condicional. Usou-se a variável “nome” em um escopo externo (funcao inicio()), que é maior e contém o escopo menor do “se”. Como a variável foi definida em um escopo menor, fora dele, ela não é visível. Por outro lado, caso a declaração de “cadeia nome” ficasse fora do “se”, não haveria problema e a variável poderia inclusive ser usada dentro do escopo menor de “se”.

```
funcao inicio() {  
    inteiro idade  
    escreva("digite uma idade\n")  
    leia(idade)  
  
    cadeia nome = ""  
    se(idade >= 18)  
    {  
        escreva("digite um nome\n")  
        leia(nome)  
        escreva(nome, " já é maior de idade\n")  
    }  
    escreva(nome)  
}
```

Note apenas que é necessário inicializar a variável “nome”, pois haveria o risco de não entrar no “se”, em que ela recebe um valor, e acabar indo direto para o comando “escreva(nome)”, em que, sem valor, levaria à falha.

É hora de praticar!

Resolva alguns desafios usando a estrutura condicional simples.

Desenvolva um algoritmo em que é solicitado ao usuário digitar um número e mostre a mensagem caso esse número esteja no intervalo entre 100 e 200.

Desenvolva um algoritmo em que é solicitado ao usuário digitar em qual país ele nasceu. Caso o usuário digite “Brasil” ou “brasil”, exibirá na tela a seguinte mensagem: “Você é brasileiro”.

O algoritmo a seguir contém erros. A função dele seria informar, por meio da idade e da resposta do usuário, se ele está apto ou não para dirigir. Exemplo: usuário tem 18 anos e digitou “Sim” na pergunta se estava habilitado, e o programa mostrou a mensagem: “Usuário, você está apto para dirigir” (lê-se o nome que foi digitado em vez de “Usuário”). Porém, um amigo fez algumas alterações e agora o algoritmo está dando erro. Você poderia ajudar?

```
/* Descrição:  
 *  
 * A função desse exemplo é dizer se o usuário está apto para dirigir.  
 * Estar apto para dirigir significa que o mesmo deve ter idade maior ou igual a 18 e ter habilitação.  
 * Caso nenhum desses critérios forem aceitos, o programa irá se finalizar  
 */  
  
programa {  
    funcao inicio() {  
  
        cadeia nome, habilitação  
        inteiro idade  
  
        escreva ("Digite seu nome: \n")  
        leia (nomeUsuario)  
        escreva ("Olá, seu nome é ", nome)  
  
        escreva ("\nQual é a sua idade: \n")  
        leia (idade)  
  
        escreva ("Você tem habilitação? Digite Sim ou Não \n")  
        leia (habilitacao)  
  
        se (idade>18ouhabilitação = "Sim")  
        {  
            escreva (nome, ", você está apto para dirigir!")  
        }  
  
    }  
}
```

Estrutura condicional composta

Após você ter realizado os desafios da estrutura condicional simples, será abordada agora a estrutura condicional composta, que segue o mesmo princípio da estrutura condicional simples, porém com a diferença de que, caso a condição não

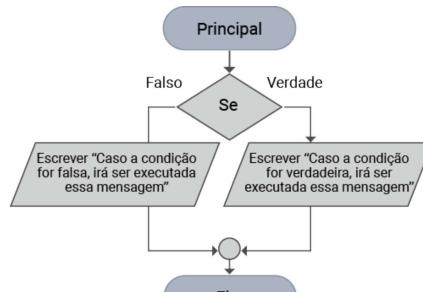
seja satisfeita, o algoritmo segue sua execução para outro comando. A estrutura de comandos é representada pelas palavras SE e SENÃO (a palavra SENÃO é usada sem acento no decorrer do código).

Pseudocódigo

```
se (condicao)
{
    //Instruções a serem executadas se o desvio for verdadeiro
}
senao
{
    //Instruções a serem executadas se o desvio for falso
}
```

Usando a estrutura condicional composta, seguindo esse exemplo, o algoritmo testará a condição e então, dependendo da interação do usuário, executará o comando A ou B. Você pode usar a estrutura condicional composta com várias condições, mas tenha cuidado para que o algoritmo fique claro.

Fluxograma



(objetos/figura_17.png)

Figura 17 – Fluxograma

Veja alguns exemplos práticos:



Exemplo: faixa etária (jovem e adulto)

Fluxograma

```

graph TD
    Principal([Principal]) --> Idade[Inteiro idade]
    Idade --> Print1[/Escrever "Digite a sua idade"]
    Print1 --> LerIdade[Ler idade]
    LerIdade --> Decision{idade <= 18}
    Decision -- Falso --> Adulto[/Escrever "Você é adulto!"]
    Adulto --> Decision
    Decision -- Verdade --> Jovem[/Escrever "Você é jovem!"]
    Jovem --> Decision
    Decision --> Fim([Fim])
  
```

Figura 18 – Fluxograma

```

programa
{
  funcao inicio ()
  {
    inteiro idade

    escreva ("Digite a sua idade: ")
    leia (idade)

    se (idade <= 18)
    {
      escreva ("Você é jovem!\n")
    }

    senao
    {
      escreva ("Você é adulto!\n")
    }
  }
}
  
```

No algoritmo anterior, o usuário digitará a sua idade, e o algoritmo poderá mostrar duas possíveis mensagens: se o usuário digitou uma idade menor ou igual a 18 anos, mostrará “Você é jovem”; caso contrário, se a idade digitada foi maior que 18 (ou seja “idade ≤ 18 ” resulta em falso), mostrará na tela “Você é adulto”.

Exemplo: autenticação de usuário por mês de aniversário ou idade como senha. Algoritmo que realiza uma autenticação de um usuário, permitindo acesso caso a senha seja o mês de aniversário ou sua idade.

autenticação de usuário por mês de aniversário ou idade como senha. Algoritmo que realiza uma autenticação de um usuário, permitindo acesso caso a senha seja o mês de aniversário ou sua idade.

```
funcao inicio() {
    inteiro idade, senha, mes
    cadeia nome

    escreva ("----SOLICITANDO DADOS DO USUÁRIO----\n")
    escreva ("Digite o seu nome: ")
    leia (nome)

    escreva ("Digite a sua idade: ")
    leia (idade)

    escreva ("Digite o número correspondente ao mês que você nasceu: ")
    leia (mes)
    limpa()

    escreva ("--- Acesso ao sistema ---\n")
    escreva ("Digite a sua senha: \n")
    leia (senha)

    limpa()

    se ( senha == idade ou senha == mes )
    {
        escreva ("Olá ", nome, ", acesso permitido!\n")
    }
    senao
    {
        escreva ("Acesso negado!\n")
    }
}
```

Esse algoritmo solicitará ao usuário seu nome, sua idade e o número correspondente ao mês de nascimento. Após o algoritmo pedirá ao usuário que digite sua senha, que pode ser a sua idade ou o número correspondente ao seu mês de nascimento. Caso o usuário digite sua idade ou o número do mês, o sistema mostrará uma mensagem de “Acesso permitido”, caso contrário (se senha for diferente da idade e também diferente do mês), mostrará uma mensagem de “Acesso negado”.

Exemplo: menu de opções para o usuário digitar a opção escolhida.
Algoritmo que mostra um menu de opções ao usuário, que digitará a opção escolhida.

```
funcao inicio() {
    caractere op

    escreva ("-- Menu de opções --\n")
    escreva ("1 - Acessar sistema \n")
    escreva ("2 - Testar conexão com a internet \n")
    escreva ("3 - Sair do menu \n")

    escreva ("Escolha sua opção: ")
    leia (op)

    limpa()

    se (op == '1')
    {
        escreva ("Bem-vindo, aluno!")
    }
    senao se ( op == '2' )
    {
        escreva ("Conexão com a internet está OK!")
    }
    senao se ( op == '3' )
    {
        escreva ("Saindo....")
    }
}
```

Esse algoritmo representa um menu com três opções: acessar o sistema, testar conexão com a internet e sair do menu. Cada opção mostrará uma mensagem ao usuário. Caso o usuário digite o número 1, verá a mensagem “Bem-vindo, aluno”. Caso o usuário digite o número 2, verá a mensagem “Conexão com a internet está OK”. Caso o usuário digite o número 3, verá a mensagem “Saindo...”.

Note que, no algoritmo anterior, os comandos “se” foram aninhados, ou seja, uma cláusula “se” foi incluída dentro de outra. Isso é completamente válido. Veja outro exemplo:

```
inteiro idade
leia(idade)

se(idade > 0)
{
    se(idade >= 18)
    {
        escreva("Maior de idade")
    }
    senao
    {
        escreva("Menor de idade")
    }
}
senao
{
    escreva("idade inválida")
}
```

É hora de praticar!

Resolva novos desafios, pois, na programação, praticar nunca é demais! Use a estrutura condicional composta para resolver os desafios.

Elabore um algoritmo que solicite ao usuário digitar dois números e após mostre na tela o maior número digitado.

Elabore um algoritmo que solicite ao usuário digitar uma letra e após mostre na tela se é uma vogal ou consoante.

Elabore um algoritmo que solicite ao usuário digitar seu nome e três notas. Realize o cálculo da média ($(\text{nota1} + \text{nota2} + \text{nota3}) / 3$). Caso sua nota seja maior ou igual a 7, ele estará aprovado. Caso sua nota esteja entre 6 e 6.9, ele estará em recuperação. Caso sua média seja menor que 6, ele estará reprovado.

Elabore uma minicalculadora em que o usuário possa digitar dois números e após escolha a operação matemática que deseja.

Estrutura condicional escolha caso

Agora que você já aprendeu as estruturas condicionais simples e compostas, aprenderá uma estrutura condicional bem semelhante aos comandos de SE e SENO, porém com uma reduzida complexidade.

Apesar da similaridade com os comandos SE e SENO, nessa estrutura não é possível usar operadores lógicos, pois ela trabalha apenas com valores definidos, ou seja, ou o valor é igual ou é diferente. Outra particularidade dessa estrutura é que, além das palavras reservadas ESCOLHA e CASO, a instrução PARE deve ser usada no fim de cada um dos testes. Caso não seja colocada, o comando executará todos os casos existentes.

Um bom uso da estrutura ESCOLHA é para a criação de uma calculadora simples. Essa tarefa poderia ser feita usando SE e SENO, porém seria mais trabalhoso. Em seguida, será elaborada uma calculadora simples usando ESCOLHA.

Para isso, é necessário entender a sintaxe dessa estrutura. Veja alguns exemplos:

Chegou a hora de praticar!

Nada é melhor para aprender do que praticar a estrutura estudada!

Elabore um algoritmo que receba o número do mês e mostre seu mês correspondente. Valide mês inválido.



Elabore um algoritmo em que o usuário digite o número correspondente a um time de futebol e retorne de qual estado ele é. Exemplo: usuário digita o número “1”, que é representado pelo “Internacional”, e o algoritmo retorna: “É um time do Sul” ou “É um time Gaúcho”. Inclua pelo menos cinco times e valide a opção para times de fora da lista.

Assista ao vídeo a seguir e conheça exemplos de algoritmos com condicionais.



Encerramento

Você aprendeu os operadores relacionais e os operadores lógicos, que estarão presentes em qualquer linguagem de programação. Esses operadores ajudam na tomada de decisão, juntamente com as estruturas condicionais simples, composta e escolha-caso. O importante não é decorar os comandos, mas sim saber que eles existem e quando devem ser usados. É normal, às vezes, você se esquecer da sintaxe da função, mas se souber sua aplicação, é sinal de que está aprendendo a lógica de programação.