

Educação  
Profissional  
Paulista

Técnico em  
Desenvolvimento  
de Sistemas

# Estrutura básica

## Operadores e expressões

### Aula 1

Código da aula: [SIS]ANO1C1B2S15A1



## Objetivo da Aula

- Compreender como os operadores lógicos são fundamentais para o controle de fluxo em programas, viabilizando a execução de código com base em condições booleanas.



## Competências da Unidade (Técnicas e Socioemocionais)

- Desenvolver sistemas computacionais utilizando ambiente de desenvolvimento;
- Migrar sistemas implementando rotinas e estruturas de dados mais eficazes;
- Trabalhar a criatividade e o comprometimento na resolução de problemas computacionais.



## Recursos Didáticos

- Recurso audiovisual para a exibição de vídeos e imagens;
- Folhas sulfite, canetas coloridas e lápis.



## Duração da Aula

50 minutos

# Operadores lógicos em programação

Os operadores lógicos são fundamentais para o controle de fluxo em programas, viabilizando a execução de código com base em condições booleanas. **Eles são a base para a tomada de decisões e a execução condicional de blocos de código.**

- ✓ Tipos de operadores lógicos (AND, OR, NOT).
- ✓ Uso prático em estruturas de controle (if, else, while).
- ✓ Precedência e associação de operadores.



# Tipos de operadores lógicos

Operadores lógicos são usados para combinar ou modificar valores booleanos, geralmente em condições e decisões lógicas em programação. Os mais comuns são AND, OR e NOT.

## Para que servem

**AND:** usado para garantir que múltiplas condições sejam verdadeiras simultaneamente.

**OR:** aplicado quando qualquer uma das condições é suficiente para ser verdadeira.

**NOT:** utilizado para inverter o valor booleano de uma condição.

## Exposição

# Exemplo de implementação em Python

```
A = True  
B = False
```

```
# AND  
print(A and B) # Retorna False
```

```
# OR  
print(A or B) # Retorna True
```

```
# NOT  
print(not A) # Retorna False
```

## Exemplos de implementação no mercado

**AND:** em sistemas de autenticação, em que o usuário deve fornecer um nome de usuário **E** uma senha corretos.

**OR:** em motores de busca, para encontrar itens que correspondam a um termo **OU** outro.

**NOT:** em filtros de conteúdo, para excluir itens que **NÃO** correspondam a um critério específico.

# Uso prático em estruturas de controle

Estruturas de controle são construções de programação que gerenciam o fluxo de execução com base em condições (if, else) ou repetições (while).

## Para que servem

**if/else:** tomam decisões baseadas em condições, executando diferentes blocos de código.

**while:** executa um bloco de código repetidamente enquanto uma condição for verdadeira.

## Exposição

# Exemplo de implementação em Python

```
x = 10  
y = 7
```

### # Exemplo com AND e OR

```
if (x > 5 and y < 10) or (x == 10 and y == 7):  
    print("x é maior que 5 e y é menor que 10, OU x é igual  
    a 10 e y é igual a 7")  
else:  
    print("Nenhuma das condições acima é verdadeira")
```

### # Exemplo com NOT

```
contador = 0  
while not contador >= 5:  
    print(contador)  
    contador += 1
```

## Exemplos de implementação no mercado

**if/else:** em aplicativos de finanças, para decidir se uma transação deve ser aprovada com base no saldo da conta.

**while:** em sistemas de monitoramento, para verificar continuamente o status de um dispositivo ou sensor.



# Precedência e associação de operadores

Precedência e associação de operadores determinam a ordem na qual as operações são realizadas em expressões com múltiplos operadores.

## Para que servem

**Precedência:** define qual operador tem prioridade sobre os outros.

**Associação:** determina a ordem de avaliação dos operadores de mesma precedência.

# Precedência e associação de operadores

“Associatividade e precedência de operadores é um assunto um pouco marginalizado no estudo de linguagens de programação. Em algum momento todo desenvolvedor vai se deparar com um problema relacionado a isso. [...]

**Precedência:** um operador possui maior precedência que outro se ele precisar ser analisado antes em todas as expressões sem parênteses que envolverem apenas os dois operadores. Na matemática, na expressão  $8 * 2 - 1$ , a multiplicação é sempre avaliada antes da subtração, ou seja, ela possui maior precedência. Portanto, a expressão equivalente seria:  $(8 * 2) - 1$ .

Já a associatividade especifica se os operadores de igual precedência devem ser avaliados da esquerda para a direita ou da direita para a esquerda. De volta à matemática, o operador “menos” possui associatividade à esquerda, portanto,  $10 - 9 - 8$  é equivalente a  $(10 - 9) - 8$ .”

(TEDESCO, 2020)

# Precedência e associação de operadores

“Na maioria das linguagens de programação os operadores de atribuição são definidos com associatividade à direita. Ou seja,  $a = b = c$  é o equivalente a:  $a = (b = c)$ , que alimenta as variáveis  $a$  e  $b$  com o valor armazenado em  $c$ . As linguagens de programação possuem em suas documentações uma tabela dos operadores e suas precedências, da maior pra menor, bem como a associatividade desses operadores. Portanto, a regra aqui é avaliar essa tabela na linguagem que você utiliza.”

(TEDESCO, 2020)



## Exposição

# Exemplo de implementação em Python

### # Precedência e associação

```
resultado = False and True or  
not False
```

```
# Avalia-se primeiro 'not False'  
(True), depois 'False and True'  
(False) e, por último, 'False or  
True' (True)
```

```
print(resultado) # Retorna True
```

### Exemplos de implementação no mercado

**Precedência e associação:** em sistemas de cálculo de impostos ou contabilidade, em que múltiplas operações matemáticas e lógicas precisam ser realizadas em uma ordem específica para garantir resultados corretos.

**Essas explicações mostram como os operadores lógicos e estruturas de controle são fundamentais para a programação, possibilitando a criação de lógicas complexas e decisões automáticas em softwares e sistemas.** A compreensão desses conceitos é essencial para desenvolvedores e tem ampla aplicabilidade em diversas áreas do mercado.



Vamos  
fazer um  
**quiz**



# Teste de conhecimento

Em qual situação você usaria o operador NOT?

Se eu desejasse verificar se duas condições são verdadeiras simultaneamente.

Se eu quisesse verificar se ao menos uma de várias condições é verdadeira.

Se eu precisasse inverter o resultado booleano de uma expressão.

Se todas as condições fornecidas forem falsas.



Vamos  
fazer um  
**quiz**

## Resposta da atividade



**Se eu desejasse verificar se duas condições são verdadeiras simultaneamente.**

**RESPOSTA ERRADA**, pois o operador AND é usado para verificar se duas condições são verdadeiras simultaneamente.



**Se eu quisesse verificar se ao menos uma de várias condições é verdadeira.**

**RESPOSTA ERRADA**, porque o operador OR é utilizado quando qualquer uma das condições é suficiente para ser verdadeira.



**Se eu precisasse inverter o resultado booleano de uma expressão.**

**RESPOSTA CORRETA**, pois o operador NOT é usado para inverter o valor booleano de uma expressão.



**Se todas as condições fornecidas forem falsas.**

**RESPOSTA ERRADA**, pois não existe um operador específico para verificar se todas as condições são falsas; isso seria uma combinação de operadores lógicos.

Vamos  
fazer um  
**quiz**



## Teste de conhecimento

Qual estrutura de controle é mais apropriada para executar um bloco de código repetidas vezes, baseado em uma condição?

if

else

while

AND



Vamos  
fazer um  
**quiz**

## Resposta da atividade



**if**

**RESPOSTA ERRADA**, pois o if é usado para executar um bloco de código baseado em uma condição, mas apenas uma vez.



**else**

**RESPOSTA ERRADA**, porque o else é um complemento do if e também executa um bloco de código apenas uma vez.



**while**

**RESPOSTA CORRETA**, pois o while é usado para executar um bloco de código repetidamente enquanto uma condição for verdadeira.



**AND**

**RESPOSTA ERRADA**, pois AND é um operador lógico, e não uma estrutura de controle.



Vamos  
fazer um  
**quiz**



## Teste de conhecimento

Em uma expressão com operadores lógicos AND, OR e NOT, qual operador tem a maior precedência?

AND

OR

NOT

Todos têm a mesma  
precedência



Vamos  
fazer um  
**quiz**

## Resposta da atividade



**AND**

**RESPOSTA ERRADA**, pois o operador AND tem precedência menor que o operador NOT.



**OR**

**RESPOSTA ERRADA**, porque o operador OR tem a menor precedência entre os três.



**NOT**

**RESPOSTA CORRETA**, pois o operador NOT tem a maior precedência entre os operadores lógicos.



**Todos têm a mesma  
precedência**

**RESPOSTA ERRADA**, pois os operadores lógicos têm diferentes níveis de precedência, sendo NOT o mais alto, seguido por AND e, por fim, OR.

# Hoje desenvolvemos:

- 1** O conhecimento sobre **quais são os principais tipos de operadores lógicos (AND, OR, NOT);**
- 2** O aprendizado sobre o **uso prático dos operadores em estruturas de controle (if, else, while);**
- 3** A compreensão sobre os **conceitos de precedência e associação de operadores.**

O que nós  
**aprendemos  
hoje?**

© Getty Images





# Saiba mais

Falando sobre o conceito de operadores lógicos, que tal **aprofundarmos um pouco mais na sua função dentro do universo de desenvolvimento de software?**

CURSO EM VÍDEO. *Operadores lógicos e relacionais - curso de algoritmos #04 - Gustavo Guanabara*. Disponível em: <https://www.youtube.com/watch?v=Ig4QZNpVZYs&t=81s>. Acesso em: 10 abr. 2024.

```
void _decode_(char cbuff **buff)
{
    if (step == AES_LOC_PASS) {
        src = cbuff->load();
        dest = getattr(&ptr, &mod,
            if (mod != NULL) as dest)
            dest += buffer->table[mod];
            if (dest < 0) return _error_;
            return dest;
    }
}
```



# Referências da aula

CURSO EM VÍDEO. *Operadores lógicos e relacionais – curso de algoritmos #04 – Gustavo Guanabara*. Disponível em: <https://www.youtube.com/watch?v=lg4QZNpVZYs&t=81s>. Acesso em: 10 abr. 2024.

DELGADO, A. L. N. *6 Operadores Lógicos*. CI067 – Oficina de Computação, 21 out. 2013. Disponível em: [https://www.inf.ufpr.br/cursos/ci067/Docs/NotasAula/notas-6\\_Operadores\\_Logicos.html](https://www.inf.ufpr.br/cursos/ci067/Docs/NotasAula/notas-6_Operadores_Logicos.html). Acesso em: 10 abr. 2024.

TEDESCO, K. *Associatividade e precedência de operadores*. Treinaweb, 2020. Disponível em: <https://www.treinaweb.com.br/blog/associatividade-e-precedencia-de-operadores>. Acesso em: 10 abr. 2024.

Identidade visual: imagens © Getty Images.

E d u c a ç ã o  
P r o f i s s i o n a l  
P a u l i s t a

Técnico em  
Desenvolvimento  
de Sistemas