



# SCC0221 – Introdução à Ciência de Computação I

---

**Prof.: Dr. Rudinei Goularte**  
(rudinei@icmc.usp.br)

## Aula 9 – Operadores e Expressões em C

Instituto de Ciências Matemáticas e de Computação - ICMC  
Sala 4-229



# Expressões

---

- expressões são compostas por:
  - operandos:  $a$ ,  $b$ ,  $x$ ,  $\text{Meu\_dado}$ ,  $2$ , ...
  - operadores:  $+$ ,  $-$ ,  $\%$ , ...
  - pontuação:  $( )$ ,  $\{ \}$ ,  $"$ ,  $'$

■ Ex:

$x$

$14$

$x + y$

$(x + y) * z + w - v$



# *Expressões*

---

- expressões retornam um valor:

*x = 5 + 4 /\* retorna 9 \*/*

- esta expressão retorna 9 como resultado da expressão 5 + 4 e atribui 9 à x

*((x = 5 + 4) == 9) /\* retorna true \*/*

- na expressão acima, além de atribuir 9 à x, o valor retornado é utilizado em uma comparação e o resultado da comparação é um valor lógico



# *Expressões*

---

- expressões podem aparecer em diversos pontos de um programa

- comandos

- ```
/* x = y; */
```

- parâmetros de funções

- ```
/* sqrt(x + y); */
```

- condições de teste

- ```
/* if (x == y) */
```

- a ordem em que uma expressão é avaliada depende da prioridade dos operadores e da pontuação.

# Precedência

## Maior precedência

( ) [ ] <-

! ~ ++ -- . -(unário) (cast) \*(unário) &(unário) sizeof

\* / %

+ -

<< >>

<<= >>=

== !=

&

^

|

&&

||

?

= += -= \*= /=

## Menor precedência



# Operadores

---

## Unários:

|    |                            |                                |
|----|----------------------------|--------------------------------|
| +  | : mais unário ou positivo  | <code>/* + x; */</code>        |
| -  | : menos unário ou negativo | <code>/* - x; */</code>        |
| !  | : NOT ou negação lógica    | <code>/* ! x; */</code>        |
| &  | : endereço                 | <code>/* &amp;x; */</code>     |
| *  | : conteúdo (ponteiros)     | <code>/* (*x); */</code>       |
| ++ | : pré ou pós incremento    | <code>/* ++x ou x++ */</code>  |
| -- | : pré ou pós decremento    | <code>/* -- x ou x-- */</code> |



# *Operadores*

---

## Binários:

$+$ : adição de dois números

`/* x + y */`

$-$  : subtração de dois números

`/* x - y */`

$*$  : multiplicação de dois números

`/* x * y */`

$/$  : quociente de dois números

`/* x / y */`

$\%$ : resto da divisão:

`/* x % y */`



# Operadores bit a bit

---

Operandos inteiros.

## Operações bit-a-bit

<<: desloca à esquerda

```
/* x << 2 */
```

>>: desloca à direita

```
/* x >> 2 */
```

^ : ou exclusivo

```
/* x ^ 240 */
```

& : E bit-a-bit

```
/* x & 7 */
```

| : OU bit-a-bit

```
/* x | 200 */
```

~ : Complemento bit-a-bit

```
/* ~ x */
```





# *Operações bit a bit*

---

•

Ex

```
char x = 0xD5;
```

```
x = x & 0x0F
```

máscara de 0's

0x0F

x & 0x0F

|          |   |   |   |   |   |   |   |   |
|----------|---|---|---|---|---|---|---|---|
| x        | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0x0F     | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| x & 0x0F | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |



# Operações bit a bit

Ex:

$x = x \mid 0x0F;$

máscara de 1's

x

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

0x0F

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

ou

$x \mid 0x0F$

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

$x = x \wedge 0x0F;$

inversão controlada

x

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

0x0F

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

ou  
exclusivo

$x \wedge 0x0F$

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|

# Operações bit a bit

Ex

$x = \sim x;$

complemento

x

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

$\sim x$

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|

$x = x \ll 2;$

desloca 2 bits (  $x * 4$  )

x

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

$x = x \ll 2;$

$x = x \gg 2;$

desloca 2 bits (  $x / 4$  )

x

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| ? | ? | 1 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

$x = x \gg 2;$



# Atribuição

---

= : atribui

$x = y;$

+= : soma e atribui

$x += y; \Leftrightarrow x = x + y;$

-= : subtrai e atribui

$x -= y; \Leftrightarrow x = x - y;$

\*= : multiplica e atribui

$x *= y; \Leftrightarrow x = x * y;$

/= : divide e atribui quociente

$x /= y; \Leftrightarrow x = x / y;$

%= : divide e atribui resto

$x \% = y; \Leftrightarrow x = x \% y;$

&= : E bit-a-bit e atribui

$x \& = y; \Leftrightarrow x = x \& y;$

|= : OU bit-a-bit e atribui

$x |= y; \Leftrightarrow x = x | y;$

<<= : shift left e atribui

$x << = y; \Leftrightarrow x = x << y;$

...



# Atribuição

---

## ■ Exemplos:

`x = 10;`

`y = 5;`

`x += y;    /* x == 15 */`

`x -= 10;   /* x == 5 */`

`x *= y;    /* x == 35 */`




# Exercícios

---

1- Diga o resultado das variáveis x, y e z depois da seguinte seqüência de operações:

```
int x, y, z;  
x=y=10;  
z=++x;  
x=-x;  
y++;  
x=x+y-(z--);
```

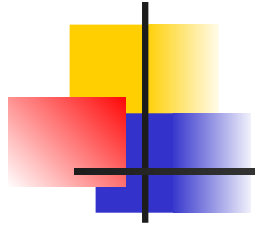
- a) x = 11, y = 11, z = 11
- b) x = -11, y = 11, z = 10
- c) x = -10, y = 11, z = 10
- d) x = -10, y = 10, z = 10
- e) Nenhuma das opções anteriores



2- Diga o resultado das variáveis x, y e z depois da seguinte sequência de operações:

```
int x,y;  
int a = 14, b = 3;  
float z;  
x = a/b;  
y = a%b;  
z = y/x;
```

- a)  $x = 4.66666$ ,  $y = 2$ ,  $z = 0.4286$
- b)  $x = 5$ ,  $y = 2$ ,  $z = 0.4$
- c)  $x = 5$ ,  $y = 2$ ,  $z = 0$ .
- d)  $x = 4$ ,  $y = 2$ ,  $z = 0.5$
- e)  $x = 4$ ,  $y = 2$ ,  $z = 0$ .
- f) Nenhuma das opções anteriores



3- Quais os valores de a, b e c após a execução do código abaixo?

```
int a = 10, b = 20, c;  
c = a+++b;
```

- a) a = 11, b = 20, c = 31
- b) a = 10 , b = 21, c = 31
- c) a = 11, b = 20, c = 30
- d) a = 10, b = 21, c = 30
- e) Nenhuma das opções anteriores





4- Qual o valor das variáveis v, x, y e z após a execução do seguinte trecho de código

```
int v = 0, x = 1, y = 2, z = 3;  
v += x+y;  
x *= y = z + 1;  
z %= v + v + v;  
v += x += y += 2;
```

- a). v=11, x=8, y=6, z=3
- b). v=0, x=1, y=2, z=3
- c). v=10, x=7, y=6, z=3
- d). v=13, x=10, y=6, z=3
- e). Nenhuma das opções anteriores



# Operadores Relacionais

---

- Aplicados a variáveis que obedecem a uma relação de ordem, retornam 1 (true) ou 0 (false)

## Operador

>

>=

<

<=

==

!=

## Relação

Maior do que

Maior ou igual a

Menor do que

Menor ou igual a

Igual a

Diferente de



# Operadores Lógicos

---

- Operam com valores lógicos e retornam um valor lógico verdadeiro (1) ou falso (0)

| Operador | Função    | Exemplo             |
|----------|-----------|---------------------|
| &&       | AND (E)   | (c >='0' && c<='9') |
|          | OR (OU)   | (a=='F'    b!=32)   |
| !        | NOT (NÃO) | (!var)              |



# Tabela Verdade

---

| a | b | !a | !b | a && b | a    b |
|---|---|----|----|--------|--------|
| 0 | 0 | 1  | 1  | 0      | 0      |
| 0 | 1 | 1  | 0  | 0      | 1      |
| 1 | 0 | 0  | 1  | 0      | 1      |
| 1 | 1 | 0  | 0  | 1      | 1      |

# Exercícios



---

5- A operação lógica

$(-5 \ || \ 0) \ \&\& (3 \ >= \ 2) \ \&\& (1 \ != \ 0) \ || (3 \ < \ 0)$  é:

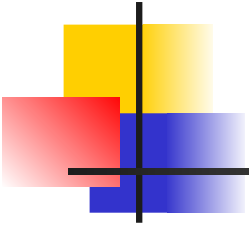
a). Verdadeira

b). Falsa

c). Inválida, pois sua sintaxe está errada.

d). Nem Verdadeira nem Falsa

e). Nenhuma das opções anteriores



FIM.