Aula 03 - Linguagem C: Operadores

Prof. Me. Claudiney R. Tinoco

profclaudineytinoco@gmail.com

Faculdade de Computação (FACOM)
Bacharelado em Ciência da Computação (BCC)
Bacharelado em Sistemas de Informação (BSI)

Programação Procedimental (PP)
GBC014 - GSI002

Operadores

- Os operadores são usados para desenvolver diferentes tipos de operações. Com eles podemos:
  - Realizar operações matemáticas com suas variáveis.
  - Realizar operações de comparação entre suas variáveis.
  - Realizar operações lógicas entre suas variáveis.
  - Realizar operações em nível de bits com suas variáveis

Operadores aritméticos

- São aqueles que operam sobre números (valores, variáveis, constantes ou chamadas de funções) e/ou expressões e têm como resultados valores numéricos
  - Note que os operadores aritméticos são sempre usados em conjunto com o operador de atribuição.

Operador	Significado	Exemplo
+	Adição de dois valores	z = x + y
-	Subtração de dois valores	z = x – y
*	Multiplicação de dois valores	z = x * y
/	Quociente de dois valores	z = x / y
%	Resto de uma divisão	z = x % y

Operadores aritméticos

- Podemos devolver o resultado para uma outra variável ou para um outro comando ou função que espere receber um valor do mesmo tipo do resultado da operação, no caso, a função printf()

```
int main(){
    int x = 10, y = 20, z;

    z = x * y;
    printf("z = %d\n",z);

    z = y / 10;
    printf("z = %d\n",z);

    printf("x+y = %d\n",x+y);

    return 0;
}
```

Operadores aritméticos

- IMPORTANTE
  - As operações de multiplicação, divisão e resto são executadas antes das operações de adição e subtração. Para forçar uma operação a ser executada antes das demais, ela é colocada entre parênteses
    - **z = x * y + 10;**
    - **z = x * (y + 10);**
  - O operador de subtração também pode ser utilizado para inverter o sinal de um número
    - **x = −y;**
  - Neste caso, a variável **x** receberá o valor de **y** multiplicado por **−1**, ou seja,
    - **x = (−1) * y;**

Operadores aritméticos

- IMPORTANTE
  - Em uma operação utilizando o operador de quociente /, se o numerador e o denominador forem números inteiros, por padrão o compilador retornará apenas a parte inteira da divisão

```c
int main(){
    float x;
    x = 5/4; // x = 1.000000
    printf("x = %f\n",x);

    x = 5/4.0; // x = 1.250000
    printf("x = %f\n",x);

    return 0;
}
```

Operadores Relacionais

- São aqueles que verificam a magnitude (qual é maior ou menor) e/ou igualdade entre dois valores e/ou expressões.
  - Os operadores relacionais são operadores de comparação de valores
  - Retorna ***verdadeiro*** (*1*) ou ***falso*** (*0*)

Operador	Significado	Exemplo
>	Maior do que	X > 5
>=	Maior ou igual a	X >= Y
<	Menor do que	X < 5
<=	Menor ou igual a	X <= Z
==	Igual a	X == 0
!=	Diferente de	X != Y

IMPORTANTE

- Símbolo de atribuição = é diferente, muito diferente, do operador relacional de igualdade ==

```c
int Nota;
Nota == 60; // Nota é igual a 60?
Nota = 50; // Nota recebe 50
// Erro comum em C:
// Teste se a nota é 60
// Sempre entra na condição
if (Nota = 60) {
    printf("Você passou raspando!!");
}
// Versão Correta
if (Nota == 60) {
    printf("Você passou raspando!!");
}
```

Importante

- Símbolo de atribuição = é diferente, muito diferente, do operador relacional de igualdade ==

```
int Nota;
Nota == 60; // Nota é igual a 60?
Nota = 50; // Nota recebe 50
// Erro comum em C:
// Teste se a nota é 60
```

Operadores Lógicos

- Certas situações não podem ser modeladas utilizando apenas os operadores aritméticos e/ou relacionais
  - Um exemplo bastante simples disso é saber se determinada variável x está dentro de uma faixa de valores.
  - Por exemplo, a expressão matemática
    - $0 < x < 10$
  - indica que o valor de x deve ser maior do que 0 (zero) e também menor do que 10

Operadores Lógicos

- Os operadores lógicos permitem representar situações lógicas unindo duas ou mais expressões relacionais simples em uma composta
  - Retorna ***verdadeiro*** (*1*) ou ***falso*** (*0*)
- Exemplo
  - A expressão ***0 < x < 10***
  - Equivale a ***(x > 0) && (x < 10)***

Operador	Significado	Exemplo
&&	Operador **E**	(x > 0) && (x < 10)
||	Operador **OU**	(a == 'F') || (b != 32)
!	Operador **NEGAÇÃO**	!(x == 10)

Operadores Lógicos

- Tabela verdade
  - Os termos ***a*** e ***b*** representam o resultado de duas expressões relacionais

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

Operadores Lógicos

- Exemplos

```c
int main(){
    int r, x = 5, y = 3;
    r = (x > 2) && (y < x);//verdadeiro (1)
    printf("Resultado: %d\n",r);
    r = (x%2==0) && (y > 0);//falso (0)
    printf("Resultado: %d\n",r);
    r = (x > 2) || (y > x);//verdadeiro (1)
    printf("Resultado: %d\n",r);
    r = (x%2==0) || (y < 0);//falso (0)
    printf("Resultado: %d\n",r);
    r = !(x > 2);// falso (0)
    printf("Resultado: %d\n",r);
    r = !(x > 7) && (x > y);// verdadeiro (1)
    printf("Resultado: %d\n",r);

    return 0;
}
```

Operadores de pré e pós-incremento/decremento

- Esses operadores podem ser utilizados sempre que for necessário somar uma unidade (incremento) ou subtrair uma unidade (decremento) a determinado valor

Operador	Significado	Exemplo	Resultado
++	incremento	++x ou x++	x = x + 1
--	decremento	--x ou x--	x = x - 1

Operadores de pré e pós-incremento/decremento

- Qual a diferença em usar antes ou depois da variável?

Operador	Significado	Resultado
++x	pré-incremento	soma +1 à variável x antes de utilizar seu valor
x++	pós-incremento	soma +1 à variável x depois de utilizar seu valor
--x	pré-decremento	subtrai −1 da variável x antes de utilizar seu valor
x--	pós-decremento	subtrai −1 da variável x depois de utilizar seu valor

tem importância se o operador for usado sozinho

- Porém, se esse operador for utilizado dentro de uma expressão aritmética, a diferença entre os dois operadores será evidente

Operadores de pré e pós-incremento/decremento

- Essa diferença de sintaxe no uso do operador não tem importância se o operador for usado sozinho
  - Porém, se utilizado dentro de uma expressão aritmética, a diferença entre os dois operadores será evidente

```
int main(){
    int x, y;
    x = 10;
    y = x++;
    printf("%d \n",x);// 11
    printf("%d \n",y);// 10

    y = ++x;
    printf("%d \n",x);// 12
    printf("%d \n",y);// 12

    return 0;
}
```

Operadores de atribuição simplificada

- Muitos operadores são sempre usados em conjunto com o operador de atribuição.
  - Para tornar essa tarefa mais simples, a linguagem C permite simplificar algumas expressões

Operador	Significado	Exemplo		
+=	Soma e atribui	x += y	igual a	x = x + y
-=	Subtrai e atribui	x -= y	igual a	x = x – y
*=	Multiplica e atribui	x *= y	igual a	x = x * y
/=	Divide e atribui o quociente	x /= y	igual a	x = x / y
%=	Divide e atribui o resto	x %= y	igual a	x = x % y

Operadores de atribuição simplificada

Sem operador

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int x = 10,y = 20;
    x = x + y - 10;
    printf("x = %d\n",x);
    x = x - 5;
    printf("x = %d\n",x);
    x = x * 10;
    printf("x = %d\n",x);
    x = x / 15;
    printf("x = %d\n",x);

    return 0;
}
```

Com operador

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int x = 10,y = 20;
    x += y - 10;
    printf("x = %d\n",x);
    x -= 5;
    printf("x = %d\n",x);
    x *= 10;
    printf("x = %d\n",x);
    x /= 15;
    printf("x = %d\n",x);

    return 0;
}
```

Operadores

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

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

OPERADORES

- Exercício
  - 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;
```

Operadores

- Exercício
  - Diga se as seguintes expressões serão verdadeiras ou falsas:

```
int x = 7;
(x > 5) || (x > 10)
(!(x == 6) && (x >= 6)
```

Conversões de Tipos na Atribuição

- Atribuição entre tipos diferentes
  - O compilador converte automaticamente o valor do lado direto para o tipo do lado esquerdo do operador de atribuição "="
    - Pode haver perda de informação

```
int x = 65;
char ch;
float f = 25.1;
//ch recebe 8 bits menos significativos de x
//converte para a tabela ASCII
ch = x;
printf("ch = %c\n",ch);// 'A'
//x recebe parte apenas a parte inteira de f
x = f;
printf("x = %d\n",x);// 25
//f recebe valor 8 bits convertido para real
f = ch;
printf("f = %f\n",f);// 65.000000
//f recebe o valor de x
f = x;
printf("f = %f\n",f);// 25.000000
```

Modeladores (Casts)

- Um modelador é aplicado a uma expressão
- Força o resultado da expressão a ser de um tipo especificado.
  - (tipo) expressão

- Exemplo

```
float x, y, f = 65.5;

x = f / 10.0;
y = (int) (f / 10.0);
printf("x = %f\n",x);//6.550000
printf("y = %f\n",y);//6.000000
```

Precedência dos Operadores

MAIOR PRECEDÊNCIA	
++ --	Pré-incremento/decremento
()	Parênteses (chamada de função)
[]	Elemento de array
.	Elemento de struct
->	Conteúdo de elemento de ponteiro para struct
++ --	Pós-incremento/decremento
+ -	Adição e subtração unária
! ~	Não lógico e complemento bit a bit
(tipo)	Conversão de tipos (*type cast*)
*	Acesso ao conteúdo de ponteiro
&	Endereço de memória do elemento
sizeof	Tamanho do elemento
* / %	Multiplicação, divisão e módulo (resto)
+ -	Adição e subtração
<< >>	Deslocamento de bits à esquerda e à direita
< <=	"Menor do que" e "menor ou igual a"
> >=	"Maior do que" e "maior ou igual a"
== !=	"Igual a" e "diferente de"
&	E bit a bit
^	OU exclusivo
|	OU bit a bit
&	E lógico
||	OU lógico
?:	Operador ternário
=	Atribuição
+= -=	Atribuição por adição ou subtração
*= /= %=	Atribuição por multiplicação, divisão ou módulo (resto)
<<= >>=	Atribuição por deslocamento de bits
&= ^= | =	Atribuição por operações lógicas
,	Operador vírgula
MENOR PRECEDÊNCIA	

Referências

✓ Básica

- BACKES, André. *"Linguagem C: completa e descomplicada"*. Elsevier Brasil, 2013.
- **DAMAS, Luís. "Linguagem C". Grupo Gen-LTC, 2016.**
- **MIZRAHI, Victorine V. "Treinamento em linguagem C", 2a. ed., São Paulo, Pearson, 2008.**

✓ Extra

- BACKES, André. *"Programação Descomplicada Linguagem C"*. Projeto de extensão que disponibiliza vídeo-aulas de C e Estruturas de Dados. Disponível em: `https://www.youtube.com/user/progdescomplicada`. Acessado em: 25/04/2022.

✓ Baseado nos materiais do professor:

- Prof. André Backes (UFU)

Dúvidas?

Prof. Me. Claudiney R. Tinoco
profclaudineytinoco@gmail.com

Faculdade de Computação (FACOM)
Universidade Federal de Uberlândia (UFU)