UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Operadores e Expressões

Aula 04

Marcos Silvano Almeida
marcossilvano@professores.utfpr.edu.br
Departamento de Computação
UTFPR Campo Mourão

Roteiro

- Operadores Aritméticos
- Expressões Aritméticas
  - Divisão float vs int
- Operadores Relacionais
- Operadores Lógicos
- Precedência entre operadores

Imagem de fundo:
http://www.webdesignhot.com/free-vector-graphics/blue-waves-abstract-background/

Operadores Aritméticos

Operadores Aritméticos

- Realizam operações aritméticas, resultando em um número (int, float…).
- Considere:

```
int a = 10;
int b = 3;
```

Operador	Descrição	Exemplo	Resultado	Precedência
+	adição	a + b	13	3º
-	subtração	a - b	7	3º
*	multiplicação	a * b	30	2º
/	Divisão (interia ou real)**	a / b	3 (div int)	2º
%	resto da divisão (somente int)	a % b	1	2º
++	adição em 1 (unário)	a++	11	1º
--	subtração inteira em 1 (unário)	a--	9	1º

Operadores Aritméticos

```c
#include <stdio.h>
int main() {
 int a = 20;
 int b = 6;
 int c = 0;

 c = a + b;
 printf("a + b = %d\n", c); // 26

 c = a - b;
 printf("a - b = %d\n", c); // 14

 c = a * b;
 printf("a * b = %d\n", c); // 120

 // esta divisão tem resultado int,
 // pois ambos operandos são int
 c = a / b;
 printf("a / b = %d\n", c); // 3

 c = a % b;
 printf("a %% b = %d\n", c); // 2

 a++;
 printf("a++ = %d\n", a); // 21

 a--;
 printf("a-- = %d\n", a); // 20

 return 0;
}
```

Operadores aritméticos

- Contração: **operador + atribuição**
  - `+=, -=, *=, /=, %=`

```c
#include <stdio.h>
int main() {
  int a = 20;
  int b = 6;
  int c = 4;

  c += b;  // c = c + b
  printf("c = %d\n", c); // 10

  c *= 2;  // c = c * 2
  printf("c = %d\n", c); // 20
  return 0;
}
```

Expressões Aritméticas

Expressões

- Atribuição
  - O lado direito é realizado primeiro
  - Somente após concluir o lado direito, o valor resultante é atribuído ao lado esquerdo

```
variável = expressão (aritmética, lógica, relacional, atribuição, função…)
```

- Expressões
  - A parentização define a ordem de avaliação ⇒ dentro para fora
  - Toda expressão resulta em um único valor
  - Expressões podem ser usadas no lugar de valores literais e variáveis

```
resultado = a*b/(a-b)*c;
printf("Resultado é: %f\n", resultado);
printf("Resultado é: %f\n", a*b/(a-b)*c); // sem guardar o resultado
```

Divisão real (float)

```c
#include <stdio.h>
int main() {
 float a; // como usarei a e b para leitura, posso deixá-las sem iniciar
 float b;

 printf("Entre com A e B: ");
 scanf("%f %f", &a, &b);

 // parênteses definem a precedência de operações: primeiro a+b, depois /2
 float res = (a + b) / 2;
 printf("Resultado: %.2f\n", res);

 // usando a expressão diretamente, caso não precisemos do resultado depois
 printf("Resultado: %.2f\n", (a + b) / 2);
 return 0;
}
```

Divisão inteira/real e casting de expressões

- O resultado da divisão é int se **ambos operandos** forem int.
  - Se, ao menos, um dos operandos for float, o resultado será float.
- Se desejado, podemos forçar o resultado de expressões:

```
int a,b,resI;
float x,y,resF;
// QUANDO DESEJAMOS FORÇAR O RESULTADO PARA FLOAT, EM EXPRESSÕES DE INT
resF = a / b;           // expressão direita resulta em int
resF = a / 2;           // expressão direita resulta em int
resF = a / 2.0f;        // expressão direita resulta em float
resF = (float)a / b;    // expressão direita resulta em float
resF = a / (float)b;    // expressão direita resulta em float

// QUANDO DESEJAMOS FORÇAR O RESULTADO PARA INT, EM EXPRESSÕES DE FLOAT
resI = x / y;           // expressão resulta em float, truncado para int
resI = (int)(x / y);    // mesmo efeito da anterior
```

Divisão real vs divisão inteira

```c
int main() {
 float a, b, res;
 printf("Entre com A e B: ");
 scanf("%f %f", &a, &b);

 res = (a + b) / 2;
 printf("Divisão real: %.2f\n", res);

 res = (int)(a + b) / 2; // casting => permite forçar o tipo (apenas números)
 printf("Divisão inteira: %.2f\n", res);

 res = (a + b) % 2;
 printf("Resto (inteiro): %.2f\n", res);
 return 0;
}
```

Expressões aritméticas

```c
#include <stdio.h>
int main() {
 int x, y;
 printf("Entre com X e Y: ");
 scanf("%d %d", &x, &y);

 // precedência de operadores:  * /   >>   + -
 // parênteses permitem forçar a ordem de avaliação da expressão
 float res = (x + y) * 2 + 15.0 / x;

 // printf: 010: 10 dígitos, alinhado à direita, preenchidos com zeros
 //          .2: arredondamento para 2 dígitos após a vírgula
 printf("Resultado: %010.2f\n", res);

 return 0;
}
```

Precedência de Operadores Aritméticos

```c
int main() {
    int a = 20, b = 10, c = 15, d = 5, res;

    res = a + b * c / d;
    printf("(a + b) * c / d = %d\n", res); // 50

    res = (a + b) * c / d;
    printf("((a + b) * c) / d = %d\n", res); // 90

    res = (a + b) * (c - d);
    printf("(a + b) * (c - d) = %d\n", res); // 300

    res = a + b * c - d;
    printf("a + b * c - d = %d\n", res); // 165
    return 0;
}
```

Operadores Relacionais

Operadores Relacionais

- Comparam dois valores numéricos, resultado em booleano:
  - 0 (false) ou 1 (true)
- Considere:

```
int a = 10;
int b = 3;
```

Operador	**Descrição** (dir ⇒ esq)	**Exemplo**	**Resultado**
==	igual	a == b	0 (false)
>	maior	a > b	1 (true)
<	menor	a < b	0 (false)
>=	maior ou igual	a >= b	1 (true)
<=	menor ou igual	a <= b	0 (false)
!=	diferente (não igual)	a != b	1 (true)

Operadores Relacionais

```c
#include <stdio.h>
int main() {
   int a = 10;
   int b = 3;
   int c;

   printf("a == b? %d\n", a == b);
   printf("a > b? %d\n", a > b);
   printf("a < b? %d\n", a < b);
   printf("a >= b? %d\n", a >= b);
   printf("a <= b? %d\n", a <= b);
   printf("a != b? %d\n", a != b);

   printf("a == b? %d\n", a-5 == b+2); // precedência: aritméticos >> lógico
   return 0;
}
```

Operadores Lógicos

- Operam sobre true/false (lógica booleana)
- Considere:

```
int a = 1; // true
int b = 0; // false
```

Operador	**Descrição** (dir ⇒ esq)	Exemplo	Resultado
&&	AND (resulta em 1 se ambos forem 1)	`a && b`	`0 (false)`
||	OR (resulta em 1 se algum for 1)	`a || b`	`1 (true)`
!	NOT (inverte)	`!(a && b)`	`1 (true)`

Operadores Lógicos

```c
#include <stdio.h>
int main() {
    int a = 1; // true
    int b = 0; // false

    printf("a && b = %d\n", a && b); // 0
    printf("a || b = %d\n", a || b); // 1

    a = 5; // true
    b = 10;// true

    printf("a && b = %d\n", a && b); // 1
    printf("a || b = %d\n", a || b); // 1
    printf("!(a || b) = %d\n", !(a || b));// 0
    return 0;
}
```

TABELA VERDADE (AND, OR, NOT)				
a	**b**	**a && b**	**a || b**	**!a**
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Precedência de operadores

- Ordem de precedência (prioridade) dos operadores
- Pode ser manualmente definida por **parênteses**

Categoria	Operador	Exemplo
Aritmético (unário)	`++ --`	`a++`
Aritmético (unário)	`+ - ++ --`	`++a, -a`
Aritmético	`* / %`	`a / 2`
Aritmético	`+ -`	`a + 5`
Relacional	`< <= > >=`	`a > 5`
Relacional	`== !=`	`a == 5`
Lógico	`&&`	`a && b`
Lógico	`||`	`a || c`
Atribuição	`= += -= *= /= %=`	`a += b`

Precedência de operadores

```c
#include <stdio.h>

int main() {
   int a = 2;
   int b = 5;
   int c = 'm';

   printf("(a == 2 || b > 10) && c == 'm': %d\n",
           (a == 2 || b > 10) && c == 'm');

   c = 'k';
   printf("(a == 2 && b > 10) || c == 'm': %d\n",
           (a == 2 && b > 10) || c == 'm');
   return 0;
}
```