Estruturas de Desvios (Expressões)

Parte 2

Sumário

Operadores Aritméticos
Operadores Relacionais
Operadores Lógicos
Revisão de Expressões
Exemplo de Programa
Precedência e Associatividade de Operadores
Expressões Aritméticas, Relacionais e Lógicas
Expressões Envolvendo o Operador de Atribuição

Operações Aritméticas

Operadores Aritméticos	
+	adição
-	subtração
*	multiplicação
/	divisão
%	resto

Precedência de operadores:
prioridade de aplicação de operações

1. ( )
2. - [unários]
3. * / %
4. + -

Converter temperatura em Farenheit:

```
celsius = (fahrenheit - 32) * 5.0 / 9.0;
```

Encontrar o resto da divisão:
22 % 5 terá valor 2.

Expressões e Variáveis

Expressão aritmética válida:

```
int raio = 3 * 5 + 1;

cout << "circunferência = “<< 2 * 3.14 * raio << endl;
```

Exemplos de lugares onde uma expressão aritmética **NÃO** pode ser usada:

```
int yucky + 2 = 5;

cin >> oops * 5
```

Erro de compilação!

Operadores Relacionais

Operadores Relacionais	
<	menor
>	maior
<=	menor ou igual
>=	maior ou igual
==	igual
!=	diferente

Precedência de operadores:

1. ( )
2. - [unário]
3. * / %
4. + -
5. < <= > >=
6. == !=

O valor de uma expressão relacional é:
0 (correspondendo a **falso**), ou
1 (correspondendo a **verdadeiro**).

Operadores Relacionais

```
/* Programa “carteira_motorista” */
#include <iostream>
using namespace std;
int main()
  {
    int idade;
    idade = 17;
    cout << "Pode tirar carteira de motorista? " << (idade >= 18) << endl;
    idade = 35;
    cout << "Pode tirar carteira de motorista? " << (idade >= 18) << endl;
  }
```

Pode tirar a carteira de motorista?
17 >= 18 é falso, que é **0**.

Pode tirar a carteira de motorista?
35 >= 18 é verdadeiro, que é **1**.

Operadores Lógicos

Operadores Lógicos	
&&	AND (e)
||	OR (ou)

Entre duas expressões.
Ex.
a==0 && b==0
a==0 || b==0

Operam da esquerda pra direita

- Uma expressão usando **&&** é verdadeira somente se ambos os operadores forem verdadeiros (não zero)
- Uma expressão usando **||** é falsa somente se ambos os operadores forem falsos (zero).

Expressão 1	Expressão 2	Resultado &&	Resultado ||
Verdadeira	Verdadeira	Verdadeira	Verdadeira
Verdadeira	Falsa	Falsa	Verdadeira
Falsa	Verdadeira	Falsa	Verdadeira
Falsa	Falsa	Falsa	Falsa

Operadores Lógicos

Operadores Lógicos	
&&	AND (e)
||	OR (ou)

Operadores lógicos têm precedência menor que os operadores relacionais

Precedência de operadores:

1. ( )
2. - [unário]
3. * / %
4. + -
5. < <= > >=
6. == !=
7. &&
8. ||

Exemplos:

```
x >= 3 && x <= 50
x == 1 || x == 2 || x == 3
```

Revisão de Expressões:

```
/* Programa “Expressoes” */
#include <iostream>
using namespace std;

int main ()
{
int score = 5;
   cout << 5 + 10 * 5 % 6;          // 7
   cout << 10 / 4;                  // 2
   cout << 10.0 / 4.0;              // 2.5
   cout << 'A' + 1                  // B
   cout << score + (score == 0);    // 5
 }
```

Saída

Exemplo de Programa

Programa que lê um número inteiro e imprime 0, se o número for par, e 1, se o número for ímpar:

```
/* Programa “par ou ímpar” */
#include <iostream>
using namespace std;
int main ()
{
   int numero;
   cout << "Entre com um numero inteiro: ";
   cin >> numero;
   cout << "\nPar? " << numero % 2 << endl;
}
```

Precedência e Associatividade de Operadores

- Operadores aritméticos têm precedência maior que os operadores relacionais.
- Para obter resultado de uma operação relacional em uma expressão aritmética, é necessário usar parêntesis.

Operador	Associatividade
( )	esquerda para direita
- [unários]	direita para esquerda
* / %	esquerda para direita
+ -	esquerda para direita
< <= > >=	esquerda para direita
== !=	esquerda para direita
&&	esquerda para direita
||	esquerda para direita

Expressões como valores

Expressões aritméticas, relacionais e lógicas

3 + 5 * 4 % (2 + 8)	tem valor 3
3 < 5	tem valor 1
x + 1	tem valor igual ao valor da variável x mais um
(x < 1) || (x > 4)	tem valor 1 quando o valor da variável x é fora do intervalo [1,4], e 0 quando x está dentro do intervalo

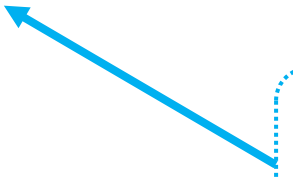
OBS: O valor resultante de expressões usando operadores aritméticos, relacionais e lógicos é um número. Este número pode ser 0 (falso) ou 1 (verdadeiro)

Expressões com operador de atribuição (=)

- **lvalue** (valor a esquerda) é um valor que se refere a um endereço na memória do computador
- O tipo do objeto do ***lvalue*** determina como o valor da ***expressão***
- é armazenada na memória
- O valor de uma expressão de atribuição é dado pelo valor da
- expressão do lado direito do =. Por exemplo:

x = 3 tem valor 3;

x = y+1 tem o valor da expressão y+1.



A expressão do lado direito é avaliada. O valor é copiado para o endereço da memória associada ao "*lvalue*"

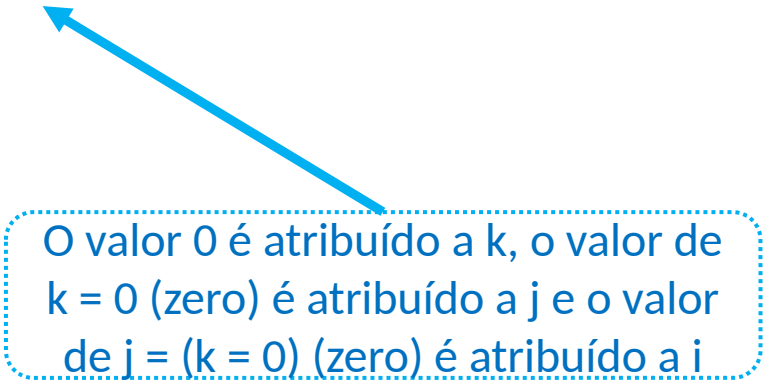
Expressões com operador de atribuição (=)

- Atribuições são expressões associadas da direita para esquerda.

$$i = j = k = 0$$

- Usando parênteses, é equivalente a:

$$i = (j = (k = 0))$$



Expressões com operador de atribuição (=)

- Expressões de atribuição podem ser usados em qualquer lugar que um valor pode ser usado
- Usá-la dentro de outros comandos alterar o valor da variável na
- memória. Exemplo:

```
int quadrado, n = 2;
cout << "Quadrado de " << n << " eh menor que 50? " << ((quadrado = n * n) < 50) << endl;
```

A avaliação da expressão relacional dentro do **cout** faz com que o número 4 seja copiado para o endereço de memória da variável "*quadrado*"

Necessário usar parênteses:
= tem menor precedência que o **<**.

Expressões com operador de atribuição (=)

– Agora compare o exemplo anterior com o próximo, no qual o valor 4

é impresso, mas sem nenhum efeito colateral:

```
int quadrado, n = 2;
  cout << "Quadrado de " << n << " eh menor que 50? " << (n * n < 50) << endl;
```

Parênteses necessários, senão compilador confunde '<' com '<<'

'*' tem maior precedência que o operador relacional '<'

Créditos: O conteúdo original deste documento é de autoria da Profª Carmem Satie Hara (DINF/ET), e foi adaptado pela Profa. Michele Nogueira (DINF/ET) e Lígia F. Borges (Doutoranda PPGInf/UFPR) para uso na disciplina *Programação de Computadores* (CI208, CI180, CI183)