

MAC 2166 – Introdução à Computação

POLI - PRIMEIRO SEMESTRE DE 2007

Material Didático

Prof. Ronaldo Fumio Hashimoto

OPERADORES LÓGICOS

Expressões Lógicas

Vimos na aula sobre comandos básicos que existem as expressões aritméticas e expressões relacionais. Vamos fazer uma pequena revisão.

Expressões Aritméticas

Expressões aritméticas são expressões matemáticas envolvendo números inteiros, variáveis inteiras, e os operadores "+" (soma), "-" (subtração), "/" (quociente de divisão inteira), "%" (resto de uma divisão inteira) e "*" (multiplicação). O resultado de uma expressão aritmética é sempre um número inteiro.

Expressões Relacionais

Expressões relacionais são expressões que envolvem comparações simples envolvendo operadores relacionais "<" (menor), ">" (maior), "<=" (menor ou igual), ">=" (maior ou igual), "!=" (diferente), "==" (igual). Estas expressões são normalmente usadas como <condição> do comando de repetição **while** e do comando de seleção **if** ou **if-else**

Uma comparação simples só pode ser feita entre pares de expressões aritméticas da forma:

`<expr_aritmética_01> <oper_relacional> <expr_aritmética_02>`

onde <expr_aritmética_01> e <expr_aritmética_02> são expressões aritméticas e <oper_relacional> é um operador relacional.

Expressões Lógicas

Agora vamos entrar no tópico desta aula que são as expressões lógicas que são usadas para criar condições mais complexas. Como vimos, uma expressão relacional é uma comparação só pode ser feita entre pares de expressões aritméticas. Agora, poderíamos estar interessados em comparações que envolvam duas expressões relacionais tais como verificar se o conteúdo de uma variável x é positivo (ou seja, $x > 0$ - primeira expressão relacional) e ao mesmo tempo menor que 10 (ou seja, $x < 10$ - segunda expressão relacional). Para isso, vamos precisar utilizar os operadores lógicos **&&** (operador "e") e **||** (isto mesmo, duas barras verticais - operador "ou"). Note que o resultado de uma expressão lógica pode ser **verdadeiro** ou **falso**.

Observação:

Para verificar se o conteúdo de uma variável x é positivo (ou seja, $x > 0$) e ao mesmo tempo menor que 10 (ou seja, $x < 10$), **NÃO** está correto escrever esta condição como $0 < x < 10$. O correto seria escrever usando o operador lógico `&&` da seguinte forma $(x > 0) \ \&\& \ (x < 10)$.

Operador Lógico `&&`

A tabela verdade para o operador lógico `&&` é dada abaixo:

<code><expr_aritmética_01></code>	<code><expr_aritmética_02></code>	<code><expr_aritmética_01> \&\& <expr_aritmética_02></code>
verdadeiro	verdadeiro	verdadeiro
verdadeiro	falso	falso
falso	verdadeiro	falso
falso	falso	falso

Observando a tabela acima, podemos concluir que o resultado do operador `&&` é verdadeiro **APENAS** quando os dois operandos (`<expr_aritmética_01>` e `<expr_aritmética_02>`) tiverem valor verdadeiro. Se algum deles, ou ambos, tiverem valor falso, o resultado de toda expressão lógica é falso.

É verdade que basta um dos operandos (`<expr_aritmética_01>` OU `<expr_aritmética_02>`) ser falso para que toda a expressão tenha um resultado falso.

Operador Lógico `||`

A tabela verdade para o operador lógico `||` é dada abaixo:

<code><expr_aritmética_01></code>	<code><expr_aritmética_02></code>	<code><expr_aritmética_01> \ \ <expr_aritmética_02></code>
verdadeiro	verdadeiro	verdadeiro
verdadeiro	falso	verdadeiro
falso	verdadeiro	verdadeiro
falso	falso	falso

Observando a tabela acima, podemos concluir que o resultado do operador `||` é falso **APENAS** quando os dois operandos (`<expr_aritmética_01>` e `<expr_aritmética_02>`) tiverem valor falso. Se algum deles, ou ambos, tiverem valor verdadeiro, o resultado de toda expressão lógica é verdadeiro.

É verdade que basta um dos operandos (`<expr_aritmética_01>` OU `<expr_aritmética_02>`) ser verdadeiro para que toda a expressão tenha um resultado verdadeiro.

Exemplos de Expressões Lógicas

Por exemplo, assumamos os seguintes valores para as variáveis inteiras $x=1$ e $y=2$. Então, veja o resultado das seguintes condições:

Expressão Lógica	Resultado
$x \geq y$	falso
$(x < y)$	verdadeiro
$x \geq (y - 2)$	verdadeiro
$(x > 0) \ \&\& \ (x < y)$	verdadeiro
$(y > 0) \ \&\& \ (x > 0) \ \&\& \ (x > y)$	falso
$(x < 0) \ \ (y > x)$	verdadeiro
$(x < 0) \ \ (y > x) \ \&\& \ (y < 0)$	falso
$x=1 \ \text{e} \ y=2$	

Precedências

Os operadores lógicos também têm precedência, como descrito na tabela abaixo:

Operador Aritmético	Associatividade
$*, /, \%$	da esquerda para a direita
$+, -$	da esquerda para a direita
$<, >, <=, >=$	da esquerda para a direita
$==, !=$	da esquerda para a direita
$\&\&$	da esquerda para a direita
$ $	da esquerda para a direita

Agora, observe a seguinte expressão lógica:

$$2 + x < y * 4 \ \&\& \ x - 3 > 6 + y / 2$$

É claro que alguém que soubesse as precedências de todos os operadores envolvidos saberia em que ordem das operações acima devem ser feitas. Agora, a legibilidade é horrível, não? Assim, uma boa dica para melhorar a legibilidade de seus programas é, mesmo conhecendo a precedência dos operadores, utilize parênteses para agrupar as expressões aritméticas como por exemplo:

$$((2 + x) < (y * 4)) \ \&\& \ ((x - 3) > (6 + y / 2))$$

que fica muito mais fácil de se entender.

Exemplos em Trechos de Programa

Agora vamos ver alguma utilidade das expressões lógicas em programação.

Considere o seguinte trecho de programa:

```

1  printf ("Entre com n > 0: ");
2  scanf ("%d", &n);
3  printf ("Entre com i > 0 e j > 0: ");
4  scanf ("%d %d", &i, &j);
5  if (n % i == 0 || n % j == 0) {
6      printf ("%d\n", n);
7  }
```

Este programa lê três números inteiros positivos n , i e j pelo teclado e verifica se o n é divisível ou por i ou por j . Em caso afirmativo, o conteúdo da variável n é impresso na tela. “Eta trecho de programa inútel!”. Mas serve para exemplificar um uso do operador lógico $||$.

Agora, vamos para um exemplo mais “legal”. Considere o seguinte trecho de programa:

```
1  i=0; par=0;
2  while (i<10 && par == 0) {
3      printf ("Entre com n > 0: ");
4      scanf ("%d", &n);
5      if (n % 2 == 0) {
6          par = 1;
7      }
8      i = i + 1;
9  }
```

Esta repetição lê uma seqüência numérica composta de no máximo 10 (dez) números inteiros positivos. Agora, se o usuário entrar somente com números ímpares, então serão lidos os 10 números ímpares e a variável `par` vai terminar com valor 0 (observe que a variável `par` foi inicializada com zero antes da repetição. Agora, se o usuário digitar algum número par, então a repetição quando retornar para verificar a condição `i<10 && par == 0` terminará (por que?) e a variável `par` terminará com valor 1 quando sair do laço.

Exercício

Dados $n > 0$ e dois números inteiros positivos i e j diferentes de 0, imprimir em ordem crescente os n primeiros naturais que são múltiplos de i ou de j e ou de ambos.

Exemplo: Para $n=6$, $i=2$ e $j=3$ a saída deverá ser : 0, 2, 3, 4, 6, 8.

Estude o trecho de código abaixo e veja se ele é uma solução para o problema acima:

```
1  printf ("Entre com n>0: ");
2  scanf ("%d", &n);
3  printf ("Entre com i>0: ");
4  scanf ("%d", &i);
5  printf ("Entre com j>0: ");
6  scanf ("%d", &j);
7  cont = 0; cont_num = 0;
8  while (cont_num < n) {
9      if (cont % i == 0 || cont % j == 0) {
10         printf ("%d\n", cont);
11         cont_num = cont_num + 1;
12     }
13     cont = cont + 1;
14 }
```

Dúvidas

Dúvidas deste material pode ser enviadas para o “Fórum para assuntos específicos da turma WEB”.