Next Up Previous

Next: 21 Mais sobre tipos: **Up:** 2 Tópicos Avançados **Previous:** 2 Tópicos Avançados

Subsecções

---

20 Operadores e Expressões Especiais

20.1 Operação de Atribuição Aritmética

É freqüente em programas C expressões do tipo:

```
tudo  = tudo + parte;

tamanho = tamanho * 2.5;

x = x * (y + 1);

j = j - 1;
```

C fornece operadores adicionais que podem ser usados para tornar estes tipos de atribuições mais curtos.

Há um operador de atribuição para cada operação aritmética listada anteriormente:

+=
operação de atribuição de adição
-=
operação de atribuição de subtração
***=**
operação de atribuição de multiplicação
/=
operação de atribuição de divisão
%=
operação de atribuição de resto

Cada uma dessas operações podem ser usadas para tornar as expressões anteriores mais curtas:

```
tudo += parte;

tamanho *= 2.5;

x *= y + 1;

j -= 1;
```

20.2 Operadores de Incremento e Decremento

Há alguns operadores em C que são equivalentes as seguintes expressões (que são bastante comuns em programas):

```
k = k + 1;

j = j - 1;
```

Estes operadores adicionais, que são ++ and - -, podem ser usados para encurtar as operações acima:

```
k++;

j--;
```

Estes operadores também podem ser colocados depois do nome da variável:

```
++k;

--j;
```

O fato do operador de incremento ser colocado antes ou depois da variável não altera o efeito da operação - o valor da variável é incrementada ou decrementada de um. A diferença entre os dois casos é QUANDO a variável é incrementada. Na expressão k++, o valor de k é primeiro usado e então é incrementada - isto é chamado *pós-incremento*. Na expressão ++k, k é incrementado primeiro, e então o valor (o novo valor) de k é usado - isso é chamado *pré-incremento*.

A diferença é ilustrada nos seguintes exemplos:

```
int main()
{
   int k = 5;

   printf("k = %d\n", k);
   printf("k = %d\n", k++);
   printf("k = %d\n", k);
}
```

O programa acima (que usa pós-incremento) imprimirá o seguinte:

```
k = 5

k = 5

k = 6
```

A segunda linha impressa com o valor de k é 5 porque o valor de k++ era 5, e k é 6 depois da impressão.

Para o programa:

```
int main()
{
   int k = 5;

   printf("k = %d\n", k);
```

```
        printf("k = %d\n", ++k);
        printf("k = %d\n", k);
    }
```

O programa, que usa pré-incremento, terá a seguinte saída:

```
k = 5

k = 6

k = 6
```

A segunda linha impressa é `6` porque o valor de `++k` é `6`.

Os operadores de atribuição não podem ser usados com expressões aritméticas. Por exemplo, as expressões

```
(ack + 2)++;

(nope + 3) += 5;
```

resultarão em erros de compilação.

Finalmente, quando usar o operador de incremento em um `printf()`, tome cuidado para não fazer o seguinte:

```
printf("%d %d\n", ++uhoh, uhoh * 2);
```

Embora isso seja perfeitamente legal em C , os resultados não são garantidados que sejam consistentes. A razão para isso é que não há garantia que os argumentos do `printf()` sejam avaliados em uma determinada ordem. O resultado do `printf()` será diferente dependendo se `++uhoh` é avaliado primeiro ou depois de `uhoh * 2`.

A solução para este problema é escrever o seguinte:

```
++uhoh;
printf("%d  %d\n", uhoh, uhoh * 2);
```

20.2.1 Expressões como Valor com Operadores de incremento e decremento

Já que incremento e decremento são formas de atribuição, o operando deve ser um *lvalue*. O valor de uma expressão de incremento ou decremento depende se o operador é usado na notação PRé ou PóS fixada (`x++, ++x, x--, --x`). Se for pré-fixada, o valor da expressão é o novo valor após o incremento ou decremento. Se for pós-fixada, o valor da expressão é o valor antigo (antes do incremento ou decremento). Por exemplo no caso de incremento, a expressão:

`x++` tem o valor de `x`

`++x` tem o valor de `x + 1`

Note que não importando a notação usada, o valor de `x` (o conteúdo do endereço de memória associada a `x`) será `x + 1`. A diferença está no valor das expressões `x++` e `++x`, não no valor de `x` (em ambos os casos o valor de `x` será incrementada de um).

20.2.2 Ambiguidade em certas expressões

Às vezes, problemas podem acontecer devido o fato que C não especifica a ordem de avaliação dos operadores em uma operação binária. Em outras palavras, em expressões como `a + b` ou `a < b`, não há maneira de saber se o valor de `a` será avaliado antes ou depois de `b` (pense em `a` e `b` como sendo qualquer expressão, não somente variáveis.) Qual deles será avaliado primeiro é particular de cada compilador, e diferentes compiladores em máquinas diferentes podem ter resultados diferentes. Portanto, se a avaliação de um dos operadores pode alterar o valor do outro, o resultado pode ser diferente dependendo da ordem de avaliação. Portanto, em expressões do tipo `x + x++`, o valor pode diferir dependendo do compilador utilizado. Isto porque não sabemos quando exatamente o incremento de `x` ocorre. Outros maus exemplos: `y = x + x--` e `x = x++`. De forma geral, para evitar este problema, não utilize senteças como estas.

Next Up Previous

Next: 21 Mais sobre tipos: **Up:** 2 Tópicos Avançados **Previous:** 2 Tópicos Avançados

Armando Luiz Nicolini Delgado
2011-02-03