Ponteiros

Precedência de operadores "*" e "&": igual precedência com associatividade da direita para a esquerda

Até o momento vimos os dois operadores unários que permitem que criemos e trabalhemos com variáveis do tipo ponteiro: o operador "*" e o operador "&".

Normalmente, trabalhamos com elas separadamente, usando, ou somente o operador "*", ou somente o operador "&". Mas e nas situações em que elas forem usadas em conjunto, como fica? Por exemplo, suponha as seguintes linhas de código:

```
int x = 2;
printf("%p", &*&*&*&*&x);
```

Desse trecho de código, duas perguntas surgem automaticamente: qual valor que será impresso? Além disso, essa expressão é correta?

Vamos aprender a responder a essas perguntas que envolvam conjuntos de operadores "*" e "&".

Aparte

Um aluno mais arguto veria o "%p" na *string* de controle da função *printf* e já responderia a primeira pergunta dizendo que será impresso um ponteiro. Sim, de fato, isso é verdade. Mas, ao responder isso, o aluno se esquece de alguns pontos importantes.

Primeiro ponto, quem vai ter que indicar o comando de formatação da *string* de controle (no caso, o "%p") a ser usado na função *printf* terá que ser ele próprio, logo em uma situação real, ele não teria essa *string* pronta e digitada.

Segundo ponto, mesmo estando o "%p" escrito, não se pode assumir que ele esteja correto, pois deve ser sempre lembrado que o compilador C assume que o programador é experiente o suficiente para saber o que está escrevendo, ou seja, mesmo que o resultado não seja de fato um ponteiro (como por exemplo, se o resultado da expressão resultar no valor 2 armazenado na variável x), o compilador vai permitir que seja impresso esse valor no formato ponteiro.

Mas um ainda mais esperto aluno de C diria: "posso tirar essa dúvida usando a opção '-Wall' do gcc que vai indicar se o comando de formatação '%p' está compatível com o resultado da expressão". Sim, isso mesmo. Mas neste caso esse aluno não saberia de fato o porquê dessa compatibilidade. Simplesmente vai saber que é compatível. Como o objetivo é sabermos exatamente o porquê de tudo o que escrevemos em C, devemos ser capazes de "desvendar" o resultado da expressão escrita na função *printf* anterior.

Primeiro aspecto: C permite a utilização de quantos operadores "*" e "&" se quiser, e em qualquer ordem, desde que a sua aplicação tenha sentido.

Então, o fato de eu usar diversos "*" e "&" na linha não está incorreto. Só estará incorreto se a sua aplicação não tiver sentido. Vamos ver então se tem sentido a sua aplicação.

Para isso precisamos saber que ambos os operadores têm igual precedência. Em outras palavras, nenhum deles deve ter prioridade de execução sobre o outro.

Fica fácil entender e memorizar isso se compararmos eles aos operadores já conhecidos "+" e "-". Quando estes operadores estão em uma mesma expressão, eles devem ser executados na ordem em que aparecem da esquerda para a direita na expressão, lembram?
Ex.: 3 + 2 – 1 = 5 – 1 = 4

Mas no caso do "*" e "&", ao contrário do "+" e "-", eles têm associatividade à direita, ou seja, devem ser executados na ordem que aparecem **da direita para a esquerda**. Por que isso? Pelo fato de serem operadores unários, eles estão associados a variável que vai aparecer à direita deles, logo, nada mais lógico que seja executado primeiro, o operador "mais próximo da variável".

Exemplo. Suponha a existência de uma variável x inteira, logo a ordem de execução de uma expressão tipo "*&x", deveria ser equivalente a se escrever *(&x). Ou seja, primeiro executa o operador "mais próximo" da variável (associatividade da direita para a esquerda)

Com base nessa associatividade, a expressão anterior "&*&*&*&*&x" está correta, pois ela seria executada do seguinte modo:

= &*&*&*&*(&x)
= &*&*&*&(*endereço)
= &*&*&*(& valor no endereço)
= &*&*&(*endereço)
= &*&*(& valor no endereço)
= &*&(*endereço)
= &*(& valor no endereço)
= &(*endereço)
=(& valor no endereço)
= endereço

Conseqüentemente, já se pode ver que o resultado será de fato um endereço e, portanto, o uso do comando de formatação "%p" está coerente com o resultado da expressão! (resposta a primeira pergunta feita)

Fica como pergunta, se está correta a execução das seguintes expressões e o porquê (assuma a declaração "int x = 2"):

a) printf("%p", *&*&*&*&x);
b) printf("%p", &*&*&*&*x);
c) printf("%p", **x);
d) printf("%p", &&x);