Ponteiros

Ponteiros e Vetores

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

Vejamos o seguinte trecho de código:

```
1       int v[3] = {1, 2, 3};
2       int *pv = v; /* lembre-se que também poderia ter escrito int *pv = &v[0] */
3       printf("%i", *pv++);
4       printf("%i", *pv);
```

A execução da linha 1 faz o seguinte:

Endereço	RAM
1000	1
1004	2
1008	3

Supondo que **v** foi alocada no endereço 1000

A execução da linha 2 faz o seguinte:

Endereço	RAM
1000	1
1004	2
1008	3
1012	1000

Supondo que **pv** foi alocada no endereço 1012

Na linha 3, como os operadores "* e "++" têm igual precedência (ver tabela), deve-se executá-los com base na regra de associatividade existente entre eles, ou seja, da direita para a esquerda. Logo, deve-se executar primeiro o "++". Conclusão, **pv** receberá o valor 1004.

Observação 1
Com uma ressalva, uma vez que o operador "++" se encontra depois de um operando, vai significar que a operação deve ser realizada DEPOIS da execução da linha onde ele se encontra, ou seja, **pv** receberá o valor 1004, somente após o término da execução da linha 3 e, imediatamente, antes da execução da linha 4.

Observação 2
Note outro detalhe importante sobre o operador "++", o seu resultado é um valor numérico 1000, puro. E não existe mais qualquer ligação entre esse valor numérico 1000 e alguma variável, por isso, a escrita de uma linha tipo "&pv++" daria um erro de compilação no gcc, uma vez que seria o mesmo que fazer "&1000", ou seja, tenta-se obter o endereço do endereço de memória 1000, mas isso não faz sentido.

Na seqüência, deve ser executado o operador "*". E, conforme foi explicado, o operador "*" é aplicado sobre o valor 1000!

Como o operador "*" significa que se deve retornar o valor que estiver na posição de memória 1000, o valor retornado será o valor 1. Com isso, esse o valor impresso na tela na linha 3.

Já na linha 4, o operador "*" também vai retornar o valor que estiver na posição de memória armazenada na variável **pv**. Como vimos, **pv** conterá o valor 1004, logo, "*pv" retornará o valor 2, que será impresso na tela.

Exercícios:

1. Para cada uma das linhas de código seguintes, diga se a mesma é válida ou não (nestes casos deve dizer o porquê de não ser válida) e o que será impresso na tela quando a mesma for válida (utilize a tabela de precedência e associatividade existente no final deste documento.):
    a. printf("%p", v);
    b. printf("%i", *v);
    c. printf("%p", v++);
    d. printf("%p", pv++);
    e. printf("%p", *v++);
    f. printf("%p", *pv++);
    g. printf("%p", *++pv);
    h. printf("%i", &v++);
    i. printf("%i", &pv++);
    j. Compare os resultados e/ou erros das letras 'h' e 'i' anteriores.
    k. printf("%i", v[i++]);
    l. printf("%i", *(pv + ++i));
    m. printf("%i", *(pv + sizeof --pv)); **Obs.: considere uma arquitetura de 32 bits.**
    n. printf("%i", --*(--pv + sizeof v[i + --**&pv]++)); **Obs.: considere uma arquitetura de 32 bits.**

    Observação. Assuma para os itens deste exercício a existência do seguinte trecho de código inicial e alocação de memória:

```
int v[4] = {1, 2, 3, 4};
int *pv = v;
int i = 0;
```

Endereço	RAM
1000	1
1004	2
1008	3
1012	4
1016	1000

v foi alocada no endereço 1000 e **pv** foi alocada no endereço 1016

Tabela importante de precedência

Grupos de operadores	Associatividade	Precedência
(), [], **->**, **.**	Esquerda para a direita	Mais alta
(tipo), **sizeof**, **&**, *, **++**, **--**, **!**, **+**,**-** *Todos neste grupo são unários*	Direita para a esquerda	
* [multiplicação], **/**, **%**	Esquerda para a direita	
+, **-** [binários]	Esquerda para a direita	
<<, **>>**	Esquerda para a direita	
<, **<=**, **>**, **>=**	Esquerda para a direita	
==, **!=**	Esquerda para a direita	
& [conjunção entre bits]	Esquerda para a direita	
^	Esquerda para a direita	
|	Esquerda para a direita	
&&	Esquerda para a direita	
||	Esquerda para a direita	
?:	Direita para a esquerda	
=, **+=**, **-=**, ***=**, **/=**, **%=**, **<<=**, **>>=**, **&=**, **|=**, **^=**	Direita para a esquerda	
, [vírgula]	Esquerda para a direita	Mais baixa