Ponteiros

c) Qual o valor de y no final do programa? Escreva um /* comentário */ em cada comando de atribuição explicando o que ele faz e o valor da variável à esquerda do '=' após sua execução. Explique se os parênteses são realmente necessários.

```
#include <stdio.h>
main()
{
    int y, *p, x;
    y = 0;
    p = &y;
    x = *p;
    x = 4;
    ++(*p);
    x--;
    (*p) += x++;
    printf ("y = %d\n", y);
}
```

3. Ponteiros e Vetores

- Vetores como ponteiros

Para que possamos entender esta similaridade, devemos primeiro entender como o C trata vetores. Quando declaramos uma matriz da seguinte forma:

tipo_da_variável nome_da_variável [tam1][tam2] ... [tamN];

- Vetores como ponteiros (continuação)

O compilador C calcula o tamanho, em bytes, necessário para armazenar esta matriz. Este tamanho é:

tam1 x tam2 x tam3 x ... x tamN x tamanho_do_tipo

O compilador então aloca este número de bytes em um espaço livre de memória. O *nome_da_variável* que você declarou é na verdade um ponteiro para o *tipo_da_variável* da matriz.

- Vetores como ponteiros (continuação)

Mas aí surge a pergunta: então como é que podemos usar a seguinte notação?

nome_da_variável[índice]

Isto pode ser facilmente explicado desde que você entenda que a notação acima é *absolutamente equivalente* a se fazer:

**(nome_da_variável+índice)*

- Vetores como ponteiros (continuação)

Dessa forma um ponteiro pode ser utilizado, por exemplo, para fazer uma varredura seqüencial de uma matriz. Pois, quando temos que varrer todos os elementos de uma matriz de uma forma seqüencial, podemos usar um ponteiro, o qual vamos incrementando.

Qual seria a vantagem em se fazer uma varredura seqüencial?

- Vetores como ponteiros (continuação)

Considere o seguinte programa para zerar uma matriz:

```
main ()
{
float matrx [50][50];
int i,j;
for (i=0;i<50;i++)
        for (j=0;j<50;j++)
                matrx[i][j]=0.0;
}
```

- Vetores como ponteiros (continuação)

Podemos reescrevê-lo usando ponteiros:

```
main ()
{
        float matrx [50][50], *p;
        int count;
        for (count=0, p=matrx[0];count<2500;count++)
              *(p++)=0.0;
 }
```

- Vetores como ponteiros (continuação)

Há uma **diferença** entre o nome de um vetor e um ponteiro que deve ser **destacada**: um ponteiro é uma variável, mas o nome de um vetor não é uma variável. Isto significa, que não se consegue alterar o endereço que é apontado pelo "nome do vetor". Logo:

```
int vetor[10], *ponteiro, i;
ponteiro = &i;
/* as operações a seguir são invalidas */
vetor = vetor + 2;
vetor++;
vetor = ponteiro;
```

- Vetores como ponteiros (continuação)

Se testarmos as operações anteriores em nossos compiladores. Eles darão uma mensagem de erro.

- Lvalue;
- incompatible types in assignment.

- Vetores como ponteiros (continuação)

Exercício:

Construa um programa que declare uma matriz [3,4] de inteiros e a inicializa da seguinte forma:

$$\begin{bmatrix} 01 & 02 & 03 & 04 \\ 05 & 06 & 07 & 08 \\ 09 & 10 & 11 & 12 \end{bmatrix}$$

e depois a imprima na saída padrão. A manipulação da matriz deve ser feita utilizando ponteiros. Retorne a matriz resultante na saída padrão, com o layout apropriado.