Universidade Tecnológica Federal do Paraná
Departamento Acadêmico de Informática
IF61C – Fundamentos de Programação 1
Prof. Bogdan Tomoyuki Nassu
Profa. Leyza Baldo Dorini

1.

```
#include <stdio.h>
#include <stdlib.h>

#define N 10

int main ()
{
    int i;
    float* vetor;

    vetor = (float*) malloc (sizeof (float) * N);

    for (i = 0; i < N; i++)
        scanf ("%f", &(vetor [i]));

    for (i = 0; i < N; i++)
        printf ("%f\n", vetor [i]);

    free (vetor);

    return 0;
}
```

2.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define LIM_MAX 10

/* Lê o tamanho de um vetor, aloca, e preenche com valores aleatórios. Retorna o
  tamanho dado. Lembre-se de desalocar o vetor depois! */
int preparaVetor (int** vetor)
{
    int i, tamanho;

    printf ("Digite o numero de valores do vetor: ");
    scanf ("%d", &tamanho);
    *vetor = (int*) malloc (sizeof (int) * tamanho);

    for (i = 0; i < tamanho; i++)
        (*vetor) [i] = rand () % (LIM_MAX+1);

    return (tamanho);
}

int main ()
{
    int i, j, tamanho1, tamanho2;
    int *vetor1, *vetor2, *vetor_mul;

    tamanho1 = preparaVetor (&vetor1);
    tamanho2 = preparaVetor (&vetor2);

    /* São tamanho1 * tamanho2 elementos. */
    vetor_mul = (int*) malloc (sizeof (int) * tamanho1 * tamanho2);

    /* Para cada valor no vetor1... */
    for (i = 0; i < tamanho1; i++)
    {
        /* ... e cada valor no vetor2... */
        for (j = 0; j < tamanho2; j++)
        {
            /* A posição em vetor_mul é dada pelo número de vezes que já
               percorremos o vetor2 + a posição atual no vetor2. */
            int pos_mul = i*tamanho2 + j;

            /* Calcula e mostra. */
            vetor_mul [pos_mul] = vetor1 [i] * vetor2 [j];
            printf ("%d * %d = %d\n", vetor1[i], vetor2[j], vetor_mul[pos_mul]);
        }
    }

    free (vetor1);
    free (vetor2);
    free (vetor_mul);
    return (0);
}
```

3. O problema está em atribuir vetor_aux a vetor2, por duas razões. A primeira é que a atribuição não muda o valor de vet2 fora da função. A segunda é que vetor_aux deixa de existir após a execução da função copiaVetor. Para entender melhor, vamos acompanhar a execução do programa em um exemplo:

- vet recebe o endereço da primeira posição de um vetor de TAM (10) posições. O mesmo acontece para vet2. Suponha que vet1 = 0x0000 e vet2 = 0x0100.

- Armazenamos os valores digitados nas posições apontadas por vet1.

- Ao chamarmos a função copiaVetor, os parâmetros vetor e vetor2 funcionam exatamente como variáveis que só existem durante a execução da função. Eles são inicializados com os valores de vet e vet2, ou seja, vetor = 0x0000 e vetor2 = 0x0100.

- A função cria então um terceiro vetor, vetor_aux. Suponha que o endereço da primeira posição de vetor_aux é 0x0200.

- Copiamos o conteúdo da memória a partir do endereço 0x0000 para 0x0200, o endereço apontado por vetor_aux. Como TAM é 10 e supondo que cada int possui 4 bytes, são copiados 40 bytes

- Atribuímos a vetor2 o endereço apontado por vetor_aux – ou seja, vetor2 = 0x0200. Entretanto, vetor2 é um ponteiro que será descartado quando a função terminar, e a variável vet2, na função main, continua apontando para a posição 0x0100.

- Quando a função retorna, vetor, vetor2 e vetor_aux são descartados. Só restam as variáveis que já estavam na função main: vet e vet2 (além do contador i). Neste momento, vet = 0x0000 e vet2 = 0x0100. O conteúdo da memória apontada por vet2 permanece exatamente como estava no começo da função, nunca foi modificado – ou seja, contém “lixo”!

Para resolver o problema, em vez de usar um vetor auxiliar podemos simplesmente copiar direto o conteúdo da memória apontada por vetor (no exemplo anterior, no endereço 0x0000) para a memória apontada por vetor2 (no exemplo anterior, no endereço 0x0100). Com isso, modificamos diretamente o conteúdo da memória no endereço apontado, e não o endereço apontado em si. Ao final da função, mesmo que vetor2 não exista mais, o conteúdo no endereço 0x0100 permanece modificado – e apontado por vet2. O código é bem simples:

```
void copiaVetor (int *vetor, int *vetor2)
{
    i;
    for (i = 0; i < TAM; i++)
        vetor2 [i] = vetor [i];
}
```