

Lista de Exercícios 7 – Ponteiros e Alocação Dinâmica

Exercício 1 – Escreva um programa que contenha duas variáveis inteiras declaradas e inicializadas manualmente dentro do código. Compare seus endereços e exiba o maior endereço.

Exercício 2 – Escreva um programa que contenha duas variáveis inteiras. Leia essas variáveis do teclado. Em seguida, compare seus endereços e exiba o conteúdo de maior endereço.

Exercício 3 – Crie um programa que contenha um array de float contendo 10 elementos e solicite ao usuário que digite um número de ponto flutuante para cada posição. Imprima o endereço de cada posição deste array.

Exercício 4 – Faça o mesmo que no exercício 3 porém com um array de double. O que você pode perceber de diferença?

Exercício 5 – Crie um programa que contenha um array de inteiros contendo 5 elementos. Utilizando apenas aritmética de ponteiros, leia esse array do teclado e imprima o dobro de cada valor lido.

Exercício 6 – Crie uma função que receba como parâmetro um vetor e o imprima. Não utilize índices para percorrer o vetor, apenas aritmética de ponteiros.

Exercício 7 – Crie um programa para preencher um array de inteiros, exibi-lo na tela, calcular a multiplicação entre todos os elementos e exibir o resultado na tela. Utilize alocação dinâmica de memória. O programa deverá ter 4 funções:

- int main () → na qual será inicializado um array, o usuário será questionado quanto à quantidade de inteiros ele deseja inserir e em que será controlado o fluxo de chamadas das demais funções.

- void le_array(int *vet, int qtd) → função que receberá como parâmetro o vetor com inteiros e a quantidade de elementos que devem ser cadastrados e deverá solicitar ao usuário que digite cada um dos elementos.

- void exibe_array(int *vet, int qtd) → função que receberá como parâmetro o vetor com inteiros e a quantidade de elementos cadastrados e deverá exibir os elementos do array no seguinte formato: [4, 1, 2, 8, 10].

- long int multiplica_array(int *vet, int qtd) → função que receberá como parâmetro o vetor com inteiros e a quantidade de elementos cadastrados e deverá multiplicar todos os elementos do array. Para o array de exemplo acima o resultado seria 640. O retorno é um long pois um inteiro pode não ser suficiente para armazenar o resultado.

Exercício 8 - Um array é sempre passado como parâmetro por referência e não como parâmetro por valor. Quais são as implicações desta afirmação?

Exercício 9 – Mostre na tabela abaixo todos os passos (teste de mesa) e identifique qual será a saída do programa em C, para os valores lidos (x=3 e y =4).

[illegible]

```
#include <stdio.h>
void main(){
    int vet[] = {4,9,13};
    int i;
    for(i=0;i<3;i++){
        printf("%d ",*(vet+i));
    }
}

#include <stdio.h>
void main(){
    int vet[] = {4,9,13};
    int i;
    for(i=0;i<3;i++){
        printf("%X ",vet+i);
    }
}

#include <stdio.h>
void main(){
    int vet[] = {4,9,13};
    int *o = vet;
    int i;
    for(i=0;i<3;i++){
        printf("%X ", o++);
    }
}
```

Exercício 16 - Escreva uma função 'int* multiplica_vetores(int *a, int *b, int qtd)' que receba como parâmetro dois vetores de inteiro já preenchidos previamente e a quantidade de elementos nos vetores (a quantidade deve ser a mesma para os dois vetores e eles devem ter sido preenchidos previamente). A função deve retornar o ponteiro

para um vetor c de tamanho n alocado dinamicamente, em que $c[i] = a[i] * b[i]$. Desenvolva também o restante do código para testar a função criada.