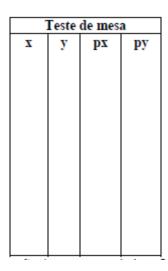


Universidade Federal de Ouro Preto Departamento de Computação e Sistemas – DECSI Programação de Computadores I Prof. Filipe Nunes Ribeiro

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 elmentos e exibir o resultado na tela. Utilize alocação dinâmica de memória. O programa deverá ter 4 funções:
- int main () \rightarrow 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).



Exercício 10 – O que fazem os seguintes programas? Escreva-os e veja a diferença entre eles. Obs. %X é igual a %p.

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

Exercício 11 - Crie uma função que recebe como parâmetro dois arrays que deverão ser somados e colocados em um terceiro array. Por exemplo, se a primeira posição do primeiro array for 10 e a primeira posição do segundo array for 5 então a primeira posição do array de destino será 15. Isso deverá ser feito para todas as posições dos arrays. O usuário deverá digitar quantos elementos o array deverá ter. Obs: não utilizar colchetes para representar os arrays. Utilize apenas aritmética de ponteiros.

Exercício 12 - Seja p um ponteiro para um tipo de dado qualquer, explique a diferença entre p++, (*p)++ e *(p++).

Exercício 13 - Escreva um programa que mostre o tamanho em byte que cada tipo de dados ocupa na memória: char, int, float, double.

Exercício 14 - Faça um programa que leia um valor inteiro N não negativo. Se o valor de N for inválido, o usuário deverá digitar outro até que ele seja válido (ou seja, positivo). Em seguida, leia um vetor V contendo N posições de inteiros, em que cada valor deverá ser maior ou igual a 1 (o programa não deve permitir a inserção de elementos menores do que 1). Esse vetor deverá ser alocado dinamicamente.

Exercício 15 - Faça uma função *aloca_vetor_float* que recebe como parâmetro a quantidade de elementos a ser alocados e retorna o ponteiro para um vetor de float alocado dinamicamente. Com o vetor retornado, solicite ao usuário que preencha cada posição do vetor e, em seguida, chame a função *exibe_vetor*, que recebe como parâmetro o vetor e quantidade de elementos e exibe os elementos na tela de maneira organizada. Utilize apenas aritmética de ponteiros para manipulação do vetor.

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.