Lista de Exercícios – Ponteiros

Créditos: Lista de exercícios baseados no material da Professora Rosilane Mota.

Para cada um dos exercícios a seguir, crie um arquivo .c com o main para realização dos testes. O código deve ser todo comentado com indicação das principais decisões sobre os comandos escolhidos.

- **1.** Escreva um programa em C que leia dois inteiros, armazenando-os em variáveis. O programa deve comparar os endereços das variáveis e exibir o maior deles.
- 2. Explique cada uma das expressões a seguir, indicando a diferença entre elas:

```
p++; (*p)++; *(p++);
```

Qual informação se refere a expressão *(p+10)?

3. Se o endereço de uma variável *valor* foi atribuído a um ponteiro *valorPtr*, quais alternativas são verdadeiras? Justifique sua resposta.

```
a) valor = = &valorPtr
b) valor = = *valorPtr
c) valorPtr = = &valor
d) valorPtr = = *valor
```

4. Identifique o erro no programa a seguir, de modo que seja exibido o valor 10 na tela.

```
#include <stdio.h>
int main()
{
    int x, *p, **q;
    p = &x;
    q = &p;
    x = 10;
    printf("\n%d\n", &q);
    return(0);
}
```

5. Escreva um programa em C que declare variáveis para armazenar um valor inteiro, um valor real e um caracter. Deve existir no programa ponteiros associados a cada um deles. O programa deve solicitar novos dados para as variáveis e elas devem ser modificadas usando os respectivos ponteiros. Exiba os endereços e os conteúdos de todas as variáveis e ponteiros antes e após a alteração.

6. Observe os dois programas a seguir, Código 1 e Código 2. Qual. É a diferença entre eles? Qual é o valor impresso para ptr em cada um dos códigos? Porque?

```
Código 2
Código 1
int main()
                                           int main()
                                           {
 int *ptr, i;
                                            int *ptr, i;
 ptr = (int *) malloc(sizeof(int));
                                            ptr = (int *) malloc(sizeof(int));
  *ptr = 10;
                                            *ptr = 10;
 for(i=0;i<5;i++){
                                            for(i=0;i<5;i++){
    *ptr=*ptr+1;
                                              ptr=ptr+1;
 printf("\nptr: %d", *ptr);
                                            printf("\nptr: %p", ptr);
 free(ptr);
                                            free(ptr);
 return 0;
                                            return 0;
```

7. Mostre na tabela a seguir todas as alterações dos conteúdos das variáveis (teste de mesa) e identifique qual será a saída do programa em C para os valores lidos (x = 5 e y = 6).

```
int main ()
{
    int x, y;
    int *px = &x;
    int *py = &y;
    scanf("%d",&x);
    scanf("%d",&y);
    px = py;
    *py = (*py) * (*px);
    *px = *px + 2;
    printf("x = %d, y = %d", x, y);
    return 0;
}
```

Teste de mesa			
x	y	px	py
		·	

- **8.** Crie um programa em C que:
 - a) Aloque dinamicamente 5 números inteiros
 - b) Peça para o usuário digitar os 5 números e armazene no espaço alocado
 - c) Mostre na tela os 5 números
 - d) Libere a memória alocada.
- **9.** Faça um programa em C que solicite ao usuário a quantidade de alunos de uma turma e leia a nota de cada aluno, armazenando-a em memória alocada dinamicamente. A nota é um número real de dupla precisão (double). Imprima todas as notas e também a média das notas.

- **10.** Faça um programa em C para armazenar em memoria um total de 30 valores do tipo inteiro, usando a função de alocação dinâmica de memória CALLOC:
 - a) Atribua para os primeiros 10 inteiros os valores de 1 a 10.
 - b) Atribua para os últimos 10 inteiros os valores de 10 a 1.
 - c) Exiba na tela os 30 números.
- **11.** Crie um programa em C que:
 - a) Aloque dinamicamente 5 números inteiros
 - b) Peça para o usuário digitar os 5 números e armazene no espaço alocado
 - c) Imprima os endereços onde foram armazenados os 5 números
 - d) Realoque um espaço para digitais mais 5 números inteiros
 - e) Peça para o usuário digitar novos 5 números e armazene no novo espaço alocado
 - f) Imprima os endereços onde foram armazenados os 10 números
 - g) Mostre na tela os 10 números
 - h) Libere a memória alocada.
- **12.** A alocação dinâmica de memória permite trabalhar com matrizes de formas diferentes, a saber:
 - Utilizando um ponteiro simples.
 - Utilizando um arranjo de ponteiro.
 - Utilizando um ponteiro para ponteiro.