LISTA SOBRE PONTEIROS

1. Se uma variável for declarada como um ponteiro, o que deve ser armazenado na variável?

2. Das seguintes declarações, quais criam variáveis do tipo ponteiro?
   a. long a;
   b. char b;
   c. char *c;
   d. int x;
   e. int *p;
   f. double w;
   g. float *k;
   h. float l;
   i. double *z;

3. Pelas seguintes declarações,

```
int *x_pt, *y_addr;
long *pt_addr;
double *pt_z;
int a;
long b;
double c;
```

   Determine qual das seguintes sentenças é válida:

a. y_addr = &a;	h.pt_addr = &b;	o. pt_addr = &c;
b. y_addr = &b;	i. pt_addr = &c;	p. pt_addr = a;
c. y_addr = &c;	j. pt_addr = a;	q. pt_addr = b;
d. y_addr = a;	k.pt_addr = b;	r. pt_addr = c; [...]
e. y_addr = b;	l. pt_addr = c;	s. y_addr = x_pt;
f. y_addr = c;	m. pt_z = &a;	t. y_addr = dt_addr;
g. pt_addr = &a;	n.pt_addr = &b;[ver]	u. y_addr = pt_addr;

4. Escreva a declaração das seguintes sentenças:
   a. A variável apontada por y_addr é um inteiro.
   b. A variável apontada por ch_addr é um caractere.
   c. A variável apontada por pt_yr é um inteiro longo
   d. A variável apontada por amt é uma variável de dupla precisão(double).
   e. A variável apontada por z é um inteiro.
   f. A variável apontada por qp é uma variável apontada por real.
   g. date_pt está apontado para um inteiro.
   h. yld_addr está apontado para uma variável de dupla precisão(double).
   i. amt_pt está apontado para uma variável real.
   j. pt_chr está apontado para um caracter.

5. Escreva as seguintes expressões:
    a. A variável apontada por x_addr
    b. A variável que está no endereço y_addr
    c. A algo apontado por pt_yld
    d. A variável apontada por pt_miles
    e. A variável apontada por mptr
    f. A variável que está no endereço pdate
    g. A variável apontada por dist_ptr
    h. A algo apontado por tab_pt
    i. A variável que está no endereço hours_pt

6. Escreva um programa em C que peça do usuário dois caracteres em letras minúsculas. Passe os dois caracteres digitados, por meio de ponteiros, para uma função chamada **capit**. A função **capit** deve pegar essas duas letras passadas como parâmetros e retornar seus valores também por meio de ponteiros. Por fim, a função que chamou **capit** deve exibir todas as quatro letras (as duas letras passadas como parâmetros para **capit** e as duas letras retornadas por **capit**).

7. Considerando as variáveis e ponteiros declarados abaixo, quais são as atribuições permitidas?

```
int x, *ptx, **pp;
float a, *pta, **pf;
```

a) x = 100;
b) *pta = &a;
c) ptx = &a;
d) *pf = &a;
e) pp = &pta;
f) **pf = 7.9;
g) *ptx = 20;
h) ptx = &x;
i) pp = &x;
j) pf = &pta;

8. Assumindo que queremos ler o valor de x, e o endereço de x foi atribuído a px, a instrução seguinte é correta? Por que?

```
scanf("%d", px);
```

9. Qual o resultado que será armazenado na variável **c**, após a execução do programa abaixo?

```
#include <stdio.h>
main()
{       int a, b, *p, *pp;
        a = b = 5;
        p = &a;
        pp = &b;
        c = *p + *pp;
}
```

10. Um ponteiro pode ser usado para dizer a uma função onde ela deve depositar o resultado de seus cálculos. Escreva uma função horas_minutos() que converta minutos em horas-e-minutos. A função recebe um inteiro **mnts** e os endereços de duas variáveis inteiras, digamos **h** e **m**, e atribui valores a essas variáveis de modo que **m** seja menor que 60 e que 60*h + m seja igual a **mnts**. Escreva também uma função **main** que use a função horas_minutos().

11. Escreva uma função min_max() que receba um vetor inteiro v[0..n-1] e os endereços de duas variáveis inteiras, digamos **min** e **max**, e deposite nessas variáveis o valor de um elemento mínimo e o valor de um elemento máximo do vetor. Escreva também uma função **main** que use a função **min_max()**.

12. Suponha que **v** é um vetor. Descreva a diferença conceitual entre as expressões v[3] e v + 3.

13. Escreva um programa que armazene os seguintes números no vetor chamado **rates**: 6.25, 6.5, 6.8, 7.2, 7.35, 7.5, 7.65, 7.8, 8.2, 8.4, 8.6, 8.8, 9.0. Indique os valores no vetor mudando o endereço em um ponteiro chamado **disp_pt**. Use o comando **for** em seu programa.

14. Modifique o programa escrito no exercício anterior para uso do comando **while**.

15. Escreva um programa que armazene a seqüência de caracteres "Sistemas de informacao" em um vetor chamado string. Use a declaração string[] = "Sistemas de informacao"; assegurando-se de que o final da sequência de caracteres possua a seqüência de escape \0 incluída no vetor. Indique os caracteres no vetor, mudando o endereço em um ponteiro chamado **mess_pt**. Use o comando **for** em seu programa.

16. Modifique o programa escrito no exercício anterior usando o comando **while**.

17. Escreva um programa que armazene as seguintes letras no vetor chamado mensagem_1: “Isto e um teste”. Faça com que o programa copie os dados armazenados em mensagem_1 para um outro vetor chamado mensagem_2 e apresente então, os valores armazenados nesse vetor mensagem_2.

18. Diga (sem usar o computador) qual o conteúdo do vetor **a** depois dos seguintes comandos.

```
int a[99];
for (i = 0; i < 99; ++i) a[i] = 98 - i;
for (i = 0; i < 99; ++i) a[i] = a[a[i]];
```

19. O que há de errado com o seguinte trecho de código?

```
char *a, *b;
a = "abacate";
b = "uva";
if (a < b)
        printf ("%s vem antes de %s no dicionário", a, b);
else
        printf ("%s vem depois de %s no dicionário",a, b);
```

20. Suponha que precisamos de uma função que troque os valores de duas variáveis inteiras, digamos i e j, qual das duas soluções abaixo atende a essa solicitação? Por que?

a)
```
void troca(int i, int j)
{
        int tmp;
        tmp = i;
        i = j;
        j = tmp;
}
```

b)
```
void troca( int *p, int *q)
{
        int tmp;
        tm = *p;
        *p = *q;
        *q = tmp;
}
```

21. Elabore uma função em C que retorne o tamanho de string, usando ponteiros (idem ao comando strlen()).

22. Escreva um programa para ler N valores reais fornecidos por um usuário, e que calcule a média destes valores e apresente o resultado. O programa deve usar ponteiros.

23. Escreva um programa que leia uma matriz de 2 X 2 fornecida pelo usuário, e apresente na tela a transposta desta matriz. Use ponteiros para acessar seu conteúdo.

24. Desenvolva uma função para somar o conteúdo de duas matrizes 3x3 usando ponteiros. Construa também uma função **main** para uso da sua função.

25. Usando o operador sizeof, determine o número dos bytes usados por seu computador para armazenar o endereço de um inteiro, de um caracter e de um número de dupla precisão(double)

    Sugestão: sizeof (* int) pode ser usado para determinar o número dos bytes da memória usados de um ponteiro para um inteiro. Você esperaria o tamanho de cada endereço ser o mesmo? Por que ou por que não?

26. Substitua o comando **while** do programa abaixo pelo comando **for**.

```
#include <stdio.h>
#define NUMS 5
void main (void)
{
        int nums[NUMS] = {16,54,7,43,-5};
        int total = 0, *n_pt;
        n_pt = nums; /*armazena o endereço de nums[0] em n_pt*/
        while (n_pt < nums + NUMS)
                total+=*n_pt++;
        printf("O total dos elementos do vetor é %d", total);
}
```

27. Defina um vetor de 10 ponteiros para números de ponto flutuante. Leia então, 10 números nas posições individuais referenciadas pelos ponteiros. Agora adicione todos os números e imprima na tela esse resultado.