



3a. Lista de Exercícios

Ponteiros

Prof. John Lenon C. Gardenghi

29 de agosto de 2019

1. Seja `v` um vetor com endereço inicial 1000. Considere o seguinte código.

```
int v = {1, 2, 3, 4, 5};
int *ptr;
ptr = v;
```

Qual o resultado de cada operação a seguir? Justifique.

- `ptr++;`
- `(*ptr)++;`
- `*(ptr++)`;
- `*(ptr+10)`;

2. Seja `vet` um vetor de 4 elementos: `TIPO vet[4]`. Suponha que, depois da declaração, `vet` esteja armazenado no endereço de memória 3000. Suponha ainda que na máquina usada uma variável do tipo `char` ocupa 1 *byte*, do tipo `int` ocupa 2 *bytes*, do tipo `float` ocupa 4 *bytes* e do tipo `double` ocupa 8 *bytes*.

Qual o valor de `vet+1`, `vet+2` e `vet+3` se:

- `vet` for declarado como `char`?
- `vet` for declarado como `int`?
- `vet` for declarado como `float`?
- `vet` for declarado como `double`?

3. Seja

```
int nums[2][3] = { {16, 18, 20}, {25, 26, 27} };
```

uma matriz com endereço inicial 1000. Qual o resultado de cada operação a seguir? Justifique.

- `nums+1`;
- `*(*(nums + 1))`
- `*(*(nums+1))`;
- `*(*(nums+1)+1)`

4. Considere as declarações:

```
int vetor[10];  
int *ponteiro;
```

Diga quais expressões abaixo são válidas ou não, e justifique sua resposta.

- `vetor = vetor + 2;`
- `vetor++;`
- `vetor = ponteiro;`
- `ponteiro = vetor;`
- `ponteiro = vetor + 2;`

5. O que faz a seguinte função?

```
void imprime (char *v, int n) {  
    char *c;  
    for (c = v; c < v + n; c++)  
        printf ("%c", *c);  
}
```

6. O que faz o seguinte código? Tente descobrir primeiro, depois teste no computador, para ter certeza.

```
int main () {  
    int y, *p, x;  
    y = 0;  
    p = &y;  
    x = *p;  
    x = 4;  
    (*p)++;  
    x--;  
    (*p) += x;  
    printf ("y = %d\n", y);  
    return 0;  
}
```

7. Qual a saída do código a seguir?

```
int main() {  
    int arr[] = { 9, 8, 98, 88, 87, 1, 2, 4, 101, 102, 103, 105 };  
    int *x = arr+4;  
    int *ptr = &arr[7];  
  
    arr[*ptr]++;  
    printf( "Valor 1: %d\n", *ptr );  
    printf( "Valor 2: %d\n", *x );  
    *x = 7;  
    printf( "Valor 3: %d\n\n", arr[4] + *ptr );  
  
    return 0;  
}
```

8. Qual a saída do código a seguir?

```
int main() {
    int b[5] = { 1, 2, 3, 4, 5 };
    int *bPtr;
    int i;
    bPtr = b;
    *(bPtr+2) += 10;
    bPtr = bPtr+2;
    for ( i = 0; i < 5; i++ )
        printf( "b[%d] = %d\n", i, b[i] );
    printf("\n");
    for ( i = 0; i < 5; i++ )
        printf( "bPtr[%d] = %d\n", i, bPtr[i] );
    return 0;
}
```

9. Como podemos arrumar o código a seguir para imprimir o valor 10?

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

10. O código a seguir funciona? Se sim, qual será a saída? Implemente e verifique. Explique o que aconteceu.

```
int main() {
    int var;
    char *ptr;
    ptr = &var;
    ptr[0] = 's';
    ptr[1] = 'o';
    ptr[2] = 'l';
    ptr[3] = '\0';
    printf( "%s ... var = %d\n\n", (char *) ptr, var);
    var = var - 3584;
    printf( "%s ... var = %d\n\n", (char *) ptr, var);
    return 0;
}
```

11. O programa abaixo produziu a seguinte resposta, que achei surpreendente:

```
x: 111
v[0]: 999
```

Os valores de x e v[0] não deveriam ser iguais?

```

void func1 (int x) {
    x = 9 * x;
}
void func2 (int v[]) {
    v[0] = 9 * v[0];
}
int main () {
    int x, v[2];
    x = 111;
    func1 (x); printf ("x: %d\n", x);
    v[0] = 111;
    func2 (v); printf ("v[0]: %d\n", v[0]);
    return EXIT_SUCCESS;
}

```

12. O código a seguir possui duas funções: a `troca_int`, para trocar o valor de duas variáveis inteiras, e a `troca_str`, para trocar o valor de duas *strings*. O código funciona? Se não, por quê? Como arrumar?

```

#include <stdio.h>
void troca_int (int *x, int *y) {
    int tmp;
    tmp = *x;
    *x = *y;
    *y = tmp;
}
void troca_str (char *x, char *y) {
    char *tmp;
    tmp = x;
    x = y;
    y = tmp;
}
int main() {
    int a, b;
    char *s1, *s2;
    a = 3;
    b = 4;
    troca_int (&a, &b);
    printf("a is %d\n", a);
    printf("b is %d\n", b);
    s1 = "Eu deveria aparecer depois";
    s2 = "Eu deveria aparecer primeiro";
    troca_str (s1, s2);
    printf("s1 is %s\n", s1);
    printf("s2 is %s\n", s2);
    return 0;
}

```

13. Escreva uma função `hm` que converta minutos em horas-e-minutos. A função deve receber um inteiro `min` **devolver** (não imprimir apenas!) a hora e o minuto convertidos. Escreva também uma função `main` que use a função `hm`.

14. Implemente uma função `concat()` que concatena 2 (dois) strings recebidos como argumentos. A função deve retornar um ponteiro para o string resultante da concatenação. Use ponteiros e alocação dinâmica o máximo possível.
15. Faça um programa que receba dois números inteiros a e b e um código de operação `op`. Se `op` for
- 0, você deve retornar $a + b$;
 - 1, você deve retornar $a - b$;
 - 2, você deve retornar $a * b$;
 - 3, você deve retornar a/b (certifique-se que $b \neq 0$).

Para resolver esse exercício, você **não** pode usar `if` nem tampouco alguma estrutura condicional, exceto para certificar-se se $b = 0$.