

## Exercícios - Ponteiros

1. O método `misterio(&i,&j)` tem um problema. Qual é? Antes da chamada do método, temos a seguinte linha de comando: `int i=6, j=10;`

```
void misterio(int *p, int *q){  
    int *temp;  
    *temp = *p;  
    *p = *q;  
    *q = *temp;  
}
```

2. O que as linhas abaixo fazem?

```
int i=99, j;  
int *p;  
p = &i;  
j = *p + 100;
```

3. O que as linhas abaixo fazem?

```
int a=5, b=12;  
int *p;  
int *q;  
p = &a;  
q = &b;  
int c = *p + *q;
```

4. O que as linhas abaixo fazem?

```
int i=7, j=3;  
int *p;  
int **r;  
p = &i;  
r = &p;  
c = **r + j;
```

5. Crie um vetor `V` com `n` inteiros, onde `n` é um valor inteiro fornecido pelo usuário. O vetor só deve ser alocado na memória depois que o usuário fornecer o valor de `n`.
6. Escreva um método que receba um vetor `V` com 10 inteiros armazenados. O método deve retornar o vetor `V` com o valor de cada posição multiplicado por 2. Os valores devem ser acessíveis fora do método.
7. Escreva um método que receba um vetor de inteiros e retorne o maior elemento e o menor elemento. Observe que o método deve retornar ao local da chamada os dois valores (não imprimir ao usuário). Portanto, você precisará usar passagem de parâmetro por referência, já que os métodos só podem retornar um único valor.
8. Escreva um método recursiva para determinar o menor elemento de um vetor não-ordenado de inteiros.
9. Dada uma lista encadeada `lista1`, escreva métodos que:
  - (a) Verifique se `lista1` está ordenada ou não (em ordem crescente);

- (b) Faça uma cópia de lista1 em um outra lista chamada lista2;
- (c) Faça uma cópia da lista1 e da lista2 em outra lista3, eliminando elementos repetidos e assumindo lista1, lista2 e lista3 ordenadas;
- (d) Inverta lista1 colocando o resultado em lista2;
- (e) Inverta lista1 colocando o resultado na própria lista1;
- (f) Intercale lista1 com lista2 gerando lista3. Assuma que lista1, lista2 e lista3 não são ordenadas.

10. Explique o que acontece nas atribuições abaixo (dica: use desenhos):

- (a) `p->item = q->item;`
- (b) `p = p->prox;`
- (c) `p = q;`
- (d) `p->prox = q;`
- (e) `q->ant = p;`
- (f) `p->prox = NULL;`
- (g) `p=p->prox->prox;`
- (h) `p->prox = q->prox;`