# Memória e passagem de parâmetros

Marco A L Barbosa malbarbo.pro.br

Em todos os exercícios, exceto o primeiro, o arranjo de entrada deve ser passado por referência e modificado dentro da função.

**Exercícios**

1. Para cada exercício da lista "Repetição, arranjos e conjuntos", analise as funções projetadas e altere os parâmetros para referência constante caso isso seja adequado. Veja a seção "Quando devemos usar referências constantes?".

2. Projete uma função que some um valor `n` a cada elemento de um arranjo de números.

3. Dado um arranjo ordenado em ordem não decrescente e um valor `v`, projete uma função que modifique o arranjo inserindo o valor `v` de maneira que o arranjo continue em ordem.

4. Projete uma função que receba como parâmetro um arranjo de inteiros e modifique-o trocando de posições o primeiro elemento com o último elemento, o segundo elemento com o penúltimo elemento e assim por diante.

5. Projete uma função que receba como parâmetros um arranjo e um índice `i` e modifique o arranjo removendo o elemento do índice `i`.

   ```
   vector<int> v = {7, 1, 8, 9};
   remove_indice(v, 2);
   check_expect(v, {vector<int> {7, 1, 9}};
   // Escreva mais exemplos!
   ```

   Dica: mova o elemento do índice `i` até o final e depois use `pop_back` para removê-lo. O `pop_back` funciona da seguinte forma

   ```
   vector<int> v = {3, 9, 1, 2};
   v.pop_back();
   check_expect(v, (vector<int> {3, 9, 1}));
   ```

6. Projete uma função que receba como parâmetros um arranjo, um índice `i` e um valor `v`, e modifique o arranjo inserindo o valor `v` no índice `i`. Dica: veja o exemplo `insere_ordenado`.

   ```
   vector<int> v = {5, 10, 2};
   insere_indice(v, 1, 8);
   check_expect(v, (vector<int> {5, 8, 10, 2}));
   // Escreva mais exemplos!
   ```

7. Projete uma função que remova todas as strings vazias de uma lista de strings. Use a função do exercício 5 como auxiliar.

8. Ordenação por seleção é outro algoritmo para ordenar um arranjo de valores. A ideia do algoritmo é selecionar um valor mínimo do arranjo a partir da posição 0 e colocá-lo na posição 0, depois encontrar um valor mínimo do arranjo a partir da posição 1 e colocá-lo na posição 1, depois encontrar um valor mínimo do arranjo a partir da posição 2 … e assim por diante. Por exemplo, vamos considerar o arranjo `{8, 5, 4, 1, 2}`.

   O valor mínimo a partir da posição 0 é 1 (que está no índice 3), colocando 1 na posição 0, obtemos `{1, 5, 4, 8, 2}`.

O valor mínimo a partir da posição 1 é 2 (que está no índice 4), colocando 2 na posição 1, obtemos `{1, 2, 4, 8, 5}`.

O valor mínimo a partir da posição 2 é 4 (que está no índice 2), colocando 4 na posição 2, obtemos `{1, 2, 4, 8, 5}`.

O valor mínimo a partir da posição 3 é 5 (que está no índice 4), colocando 5 na posição 3, obtemos `{1, 2, 4, 5, 8}`.

Baseado nesta descrição, projete uma função que faça a ordenação dos valores de um arranjo.