

# L1: Algoritmos de ordenação

Murilo Dantas

## Exercícios

1. Faça um programa que cadastre o nome e o salário de 5 funcionários. Usando um método de ordenação diferente para cada item a seguir, liste todos os dados dos funcionários das seguintes formas:
  - a. Em ordem crescente de salário;
  - b. Em ordem decrescente de salário;
  - c. Em ordem alfabética.
2. Faça um programa que cadastre 10 números, ordene-os pelo *bubble sort* e em seguida encontre e mostre:
  - a. O menor número e quantas vezes ele aparece no vetor.
  - b. O maior número e quantas vezes ele aparece no vetor.
3. Faça um programa que cadastre 8 alunos. Para cada aluno devem ser cadastrados: nome, nota1 e nota2. Primeiro, liste todos os alunos cadastrados, ordenando-os pela média ponderada das notas, tendo a primeira nota peso 2 e a segunda, peso 3. Em seguida, ordene os alunos, de forma crescente, pela nota1, e liste-os. Finalmente, considerando que, para ser aprovado, o aluno deve ter no mínimo média 7, liste, em ordem alfabética, os alunos reprovados. Em cada ordenação use um algoritmo diferente.
4. Desenvolva uma aplicação que, dados dois vetores de inteiros quaisquer com tamanho de 20 elementos, gere um terceiro com os elementos de ambos em ordem crescente, usando o *merge sort*. Apresente o resultado final.
5. Crie uma aplicação que implemente uma matriz quadrada com 16 números inteiros, os quais devem ser fornecidos aleatoriamente pelo usuário. Implemente um menu com duas opções:
  - a. Colocar os elementos em ordem crescente (use o *insertion sort*).
  - b. Colocar os elementos em ordem decrescente (use o *selection sort*).
6. Elabore um programa que implemente um vetor de estruturas com os seguintes dados: nome, idade e sexo. O programa deve apresentar os dados em:
  - a. Ordem crescente alfabética de nome (use o *quick sort*).
  - b. Ordem decrescente de idade (use o *bubble sort*).
7. Crie um vetor que armazene dados de 10 funcionários de uma empresa. Deverão ser considerados os campos: código funcional, nome, salário e data de admissão. Elabore um programa que: preencha o vetor com os dados fornecidos pelo usuário e ordene de forma crescente os elementos pelo campo de código funcional, usando o *quick sort*.

8. Crie um vetor que armazene dados de 15 casas numa imobiliária. Deverão ser considerados os campos: código, bairro, tamanho em m<sup>2</sup>, valor de venda e valor de aluguel. Elabore um programa que: preencha o vetor com os dados fornecidos pelo usuário e ordene de forma decrescente os elementos pelo campo de venda, usando o *bubble sort*.
9. Crie uma aplicação que permita inserir cerca de 10 mil números inteiros aleatórios de 1 a 10 mil num vetor de inteiros. Registre o tempo de início e término da operação de ordenação e compare essas diferenças entre os algoritmos *bubble sort*, *insertion sort* e *quick sort*. Comente as diferenças e considere testar com números diferentes de elementos. Dica: quando tiver rodando os algoritmos, evite executar outros programas na máquina.
10. Crie uma aplicação que permita inserir cerca de 8 mil números inteiros aleatórios de 1 a 8 mil num vetor de inteiros. Faça um comparativo considerando o número de trocas realizadas entre os algoritmos *selection sort*, *merge sort* e *quick sort*. Comente as diferenças e considere testar com números diferentes de elementos. Dica: quando tiver rodando os algoritmos, evite executar outros programas na máquina.