



## **Programação I**

### **Folha exercícios 10 Pesquisa e Ordenação**

António J. R. Neves  
João Rodrigues  
Osvaldo Pacheco  
Arnaldo Martins

2018/19

## Folha exercícios 10 - Pesquisa e Ordenação

### Resumo:

- pesquisa sequencial
- pesquisa binária
- ordenação sequencial
- ordenação por flutuação

Em inúmeros problemas temos a necessidade de procurar por valores em sequências. A esta tarefa designa-se pesquisa. Existem vários algoritmos em programação para a pesquisa de valores em sequências, mas nesta disciplina vamos apenas analisar dois dos mais simples: pesquisa sequencial e pesquisa binária.

A pesquisa é uma tarefa computacionalmente dispendiosa se estivermos a tratar grandes quantidades de informação. O desenvolvimento de algoritmos eficientes torna-se essencial e, como vamos ver, a complexidade dos algoritmos não é sempre a mesma.

Em outros problemas temos a necessidade de manter as sequências ordenadas. Existem vários algoritmos em programação para a ordenação de sequências, mas nesta disciplina vamos apenas analisar dois: ordenação sequencial e ordenação por flutuação.

Na ordenação sequencial vamos colocando em cada posição da sequência o valor correto, começando no primeiro. Na ordenação por flutuação vamos comparando pares de valores da sequência e trocamos se fora de ordem. Repetimos o processo enquanto houver trocas.

Nesta folha prática encontra problemas em que o principal objetivo é a manipulação de arrays, estando presentes em todos eles a necessidade de ordenar e fazer pesquisa de valores.

### 10.1 Problemas para resolver

#### Exercício 10.1

Pretende-se que altere o programa desenvolvido para o exercício 3 da Aula 6 (Arrays) de modo a que tenha um menu de operações tal como se mostra a seguir (note que foram introduzidas duas novas funcionalidades: ordenação e pesquisa). Deverá manter a dimensão máxima da sequência com 50 números, tal como descrito na versão original do problema.

Análise de uma sequência de números inteiros

- 1 - Ler a sequência via teclado
  - 2 - Escrever a sequência
  - 3 - Calcular o máximo da sequência
  - 4 - Calcular o mínimo da sequência
  - 5 - Calcular a média da sequência
  - 6 - Detetar se é uma sequência só constituída por números pares
  - 7 - Ler uma sequência de números de um ficheiro
  - 8 - Adicionar números à sequência
  - 9 - Gravar a sequência num ficheiro
  - 10 - Ordenar a sequência por ordem crescente utilizando ordenação sequencial**
  - 11 - Ordenar a sequência por ordem decrescente utilizando ordenação por flutuação**
  - 12 - Pesquisa de valor na sequência**
  - 13 - Terminar o programa
- Opção ->

### Exercício 10.2

Escreva um programa que leia do teclado uma chave de Totoloto e que a imprima na forma de matriz (ver abaixo). O Totoloto é uma forma de lotaria em que a chave premiada é obtida pela escolha de 6 bolas de um conjunto de 49 bolas numeradas de 1 a 49.

Implemente uma função que gere aleatoriamente (use a função `Math.random()`) a chave (**gerarChave**), devolvendo um array de 6 inteiros com os valores gerados. Note que os números devem ser validados para evitar números repetidos.

**Sugestões:** Faça uma função de pesquisa (**pertenceChave**) que indique se um número pertence à chave já introduzida.

Crie uma função para escrita da aposta (**mostraChave**), que deve obedecer ao formato seguinte (para o exemplo de uma chave 2, 12, 17, 27, 30, 43).

Aposta de totoloto  
Chave: 2, 12, 17, 27, 30, 43

1	X	3	4	5	6	7
8	9	10	11	X	13	14
15	16	X	18	19	20	21
22	23	24	25	26	X	28
29	X	31	32	33	34	35
36	37	38	39	40	41	42
X	44	45	46	47	48	49

**Sugestão:** Para decidir se tem de escrever X, recorra à função **pertenceChave** que criou anteriormente.

### Exercício 10.3

Pretende-se construir um programa que processe uma sequência de números reais que poderiam corresponder a valores de pH fornecidos pelo analisador de uma piscina. Os valores de pH estão armazenados num ficheiro de texto cujo nome deve ser pedido ao utilizador. O valor do pH varia entre 0 e 14, pelo que a existência de um valor no ficheiro que não pertencente ao intervalo deve ser ignorado. O programa deve começar por contar o número de amostras válidas gravadas no ficheiro de modo a criar um array com a dimensão correta.

As funções que o programa deverá disponibilizar são as seguintes:

# Analizador de pH

- ```

1 - Ler valores de pH de um ficheiro
2 - Escrever valores de pH no terminal
3 - Calcular o pH médio das amostras
4 - Calcular o número de amostras ácidas (< 7) e básicas (> 7)
5 - Calcular o número de amostras de pH superior à média
6 - Escrever valores de pH no terminal ordenados de modo
crescente
7 - Terminar o programa
Opção ->

```

### Exercício 10.4

Considere o ficheiro **alunos.tab** que se encontra no elearning. Este ficheiro tem uma lista de alunos com a seguinte estrutura: cada linha começa com um número mecanográfico e o resto da linha tem o nome do aluno. Exemplo:

65422 Joana Marques  
...

Defina uma classe Aluno com as propriedades numero e nome e faça um programa que leia os dados deste ficheiro para um array de Alunos. Depois, deve pedir ao utilizador um número e mostrar o nome correspondente, usado pesquisa binária. O programa deve permitir repetir a pesquisa até ser introduzido o número zero.

### Exercício 10.5

Escreva uma nova versão do programa do Totoloto baseado numa representação diferente da chave: um array de 50 booleanos. Se a chave contiver o número N, a posição [N] do array deve ficar **true**. Por exemplo, o array abaixo (onde T representa true) representa a chave {2,9,17,18,28,37}.

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
|   |   |   |   |   |   |   |   |   |   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |   |   |   |   |   |   |   |   |
|   |   | π |   |   |   |   |   | π |   |   |   |   |   |   |   |   | π | π |   |   |   |   |   |   |   |   | π |   |   |   |   |   |   |   |   | π |   |   |   |   |   |   |   |   |   |   |   |

Reescreva as funções **pertenceChave**, **gerarChave** e **mostraChave** para usarem esta nova representação. Qual das soluções é mais simples?