

# **Programação I**

## **Folha de Exercícios 6**

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

2018/19/20/21

## Folha Exercícios 6

### Resumo:

- Introdução aos **arrays**
- Declaração de variáveis do tipo **array**
- Acesso aos valores de um **array**
- **Arrays** como argumentos de funções
- Arrays de 2 dimensões

Vimos anteriormente que é possível criar novos tipos de dados referência que permitem declarar variáveis onde é possível guardar mais do que um valor.

No entanto, existem aplicações informáticas que precisam de lidar com grandes volumes de dados, pelo que não é eficiente ter uma variável para cada valor a armazenar.

A linguagem Java disponibiliza outro tipo de dados referência, os arrays (podemos descrever em português como sequências, vetores ou tabelas). Nesta aula prática pretende-se introduzir este tipo de dados estruturado homogéneo. Um **array** é uma organização de memória que se caracteriza pelo facto de ser um agregado de células contíguas, capaz de armazenar um conjunto de valores do mesmo tipo e aos quais se pode aceder de forma indexada. Nesta aula iremos abordar problemas que nos permitam compreender como definir e utilizar arrays, passar arrays como argumentos de funções e arrays de 2 dimensões.

### 6.1 Problemas para resolver

#### Exercício 6.1

Escreva um programa que leia uma sequência de  $N$  números inteiros, sendo o valor  $N$  pedido ao utilizador antes do início da introdução dos números. O programa deve depois imprimir esses números pela ordem inversa com que foram inseridos.

#### Exercício 6.2

Escreva um programa que leia uma sequência de números inteiros positivos e conte o número de vezes que um determinado número, pedido ao utilizador, aparece na sequência. A leitura deve terminar após a introdução de 100 números ou com o aparecimento de um número negativo.

#### Exercício 6.3

Pretende-se escrever um programa que leia do teclado uma sequência de números inteiros positivos e que permita detetar um conjunto de características acerca da sequência. A leitura da sequência termina quando aparecer o número zero como indicador de paragem ou quando tiverem sido lidos 50 números. A interação com o programa deverá ser feita através de um menu, tal como apresentado de seguida. A cada operação do menu deverá corresponder uma função.

- 1 - Ler uma sequência de números inteiros
  - 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
  - 10 - Terminar o programa
- Opção ->

### Exercício 6.4

Escreva um programa que dada uma determinada sequência de notas (valores inteiros de 0 a 20), calcule o histograma (contagem do número de ocorrências de cada nota) e o desenhe no ecrã. O número de notas a processar deverá ser pedido ao utilizador no início do programa. O histograma deverá ser implementado como função e o resultado deverá ter o formato seguinte:

```

Histograma de notas
-----
20 | *****
19 | *
.
.
.
1  | *
0  | **
  
```

- 1) Primeiro começar por associar um "\*" a cada nota encontrada.
- 2) Numa segunda fase fazer a normalização (linear) do gráfico para que o valor máximo do histograma corresponda a 50 asteriscos.

### Exercício 6.5

Escreva um programa que leia uma sequência de  $N$  números reais, sendo o valor  $N$  pedido ao utilizador no início. O programa deverá calcular a média e o desvio padrão ( $s$ ) da sequência e imprimir no ecrã os valores superiores à média. O desvio padrão deve ser implementado como função.

$$s = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2}$$

### Exercício 6.6

Escreva um programa que leia uma frase e imprima no monitor quais as letras do alfabeto que apareceram nessa frase. Para a resolução deste problema, sugere-se a utilização de um array de valores booleanos de modo a sinalizar quais os caracteres do alfabeto que apareceram pelo menos uma vez (ou em alternativa um array de 26 inteiros com o histograma dos caracteres). Usar a função **charAt(i)** para obter o caráter  $i$  de um string.

Exemplo: "aveiro".charAt(1) → 'v'

## 6.2 Exercícios com arrays de 2 dimensões

### Exercício 6.7

Considere agora um caso genérico do problema 6.2, e escreva um programa que conte o número de vezes que cada elemento ocorre num array de números inteiros (histograma). Para gerar o array deve fazer uma função que gere valores aleatórios num determinado intervalo [inicio,fim]. Os argumentos da função são os valores do intervalo (início e fim) e o número de valores a gerar (comprimento do array). A função retorna o array com os valores gerados.

De seguida implemente a função histograma que conta o número de vezes que cada elemento ocorre num array. Deve passar o array gerado e retornar um array de 2 dimensões com o histograma.

O resultado do programa deve ser o indicado abaixo, sendo o histograma representado por um array de 2 dimensões em que a coluna [0] tem os diferentes valores da sequência e a coluna [1] tem o nº de vezes que esse valor ocorre. Sempre que aparece um valor novo se não existir na coluna [0] é acrescentado e a coluna [1] respetiva é inicializada a 1. Se o valor existir na coluna [0] a coluna [1] é incrementada:

Considere o seguinte exemplo. Supondo a sequência:

$a = \{4, 2, 5, 4, 3, 5, 2, 2, 4\}$

histograma:

4	4 ocorre 3 vezes
2	2 ocorre 3 vezes
5	5 ocorre 2 vezes
3	3 ocorre 1 vez

### Exercício 6.8

Implemente um programa para gerir uma turma de alunos, em que cada aluno tem os dados inteiros seguintes: nºmec do aluno, nota teste 1, nota teste 2. O programa deve Ler do teclado ou gerar automaticamente a id e notas do aluno e depois imprimir as notas da turma e as médias de acordo com a figura abaixo.

Deve implementar as funções seguintes:

```
1)/* Função para ler uma turma de alunos do teclado;
   * Parâmetros: t - array 2d para a turma
   * Retorna:    nº de alunos lido
   */
static int lerTurma(int[][] t) { ... }

2)/* Função para gerar uma turma automaticamente
   * Parâmetros:alunos - nº de alunos a gerar
   * Retorna:    array 2d com a turma
   */
static int[][] gerarTurma(int nalunos) { ... }

3)/* Função para listar no ecrã uma turma
   * Parâmetros: t - array 2d com a turma
   *             n - nº de alunos a listar
   */
static void imprimirTurma(int[][] t, int n) { ... }

4)/* Função para calcular a média de uma dada nota da turma
   * Parâmetros: tabela - array 2d com a turma
   *             c - nº da coluna com a nota a calcular
   *             nl - nº de alunos
   * Retorna: média da coluna c
   */
static float media (int[][] tabela, int c, int nl) { ... }
```

Figura com resultado:

TURMA: 1- Ler do teclado; 2-gerar aut.

2

ID	T1	T2	Final
10023	9	9	9.00
10025	13	1	7.00
10082	14	19	16.50
10041	18	15	16.50
10081	4	15	9.50
10076	13	2	7.50
10089	6	13	9.50
10039	11	18	14.50
10048	6	14	10.00
10083	2	3	2.50

media 9.6 10.9

### Exercício 6.9

Escrever um programa que calcule a multiplicação de duas matrizes. O programa começa por pedir as dimensões e o conteúdo das duas matrizes, e depois mostra a matriz resultado. Deve garantir que as matrizes têm dimensões compatíveis.

Em matemática, o produto de duas **matrizes** é definido como a operação que resulta numa matriz. Se  $A$  é uma matriz  $m \times n$  ( $A$  também pode ser denotada por  $A$ ) e  $B$  é uma matriz  $n \times p$  ( $B$  também pode ser denotada por  $B$ ), o elemento de cada entrada  $c_{ij}$  da