Algoritmia e Programação

TP9

ASSUNTO – Estruturas de Dados indexadas

Objetivos Específicos:

- Mediante a apresentação dum problema, os alunos deverão ser capazes de o analisar, conceber e descrever o algoritmo estruturado em módulos e utilizar *arrays mono e bidimensionais*.
- Desenvolver métodos de manipulação de *arrays*.

Conteúdo da aula

Exercício 1

Analise o seguinte programa e diga qual a sua funcionalidade:

```
public class TP9_1 {
    static final int NUMBER_OF_DIGITS = 10;

    public static void main(String[] args) {
        int[] arr1 = {12345, 1011, 20946, 82530, 3322222};
        int[][] arr2 = new int[arr1.length][NUMBER_OF_DIGITS];

        calculateDigits(arr1, arr2);
        printDigitsOfNumbers(arr1, arr2);
    }
    public static void calculateDigits(int[] arr1, int[][] arr2) {
        for (int i = 0; i < arr1.length; i++) {
            int number = arr1[i];
            do {
                int digit = number % 10;
                number /= 10;
                arr2[i][digit]++;
            } while (number != 0);
        }
    }
    public static void printDigitsOfNumbers(int[] arr1, int[][] arr2) {
        for (int number = 0; number < arr1.length; number++) {
            System.out.printf("[%7d]", arr1[number]);
            for (int digit = 0; digit < arr2[number].length; digit++) {
                if (arr2[number][digit] > 0) {
                    System.out.printf(" %d(%dX);", digit, arr2[number][digit]);
                }
            }
            System.out.println("");
        }
    }
}
```

Output:

```
[  12345] 1(1X); 2(1X); 3(1X); 4(1X); 5(1X);
[   1011] 0(1X); 1(3X);
[  20946] 0(1X); 2(1X); 4(1X); 6(1X); 9(1X);
[  82530] 0(1X); 2(1X); 3(1X); 5(1X); 8(1X);
[3322222] 2(5X); 3(2X);
```

arr1

12345
1011
20946
82530
3322222

arr2

	0	1	2	3	4	5	6	7	8	9
12345		1	1	1	1	1				
1011	1	3								
20946	1		1		1		1			1
82530	1		1	1		1			1	
3322222			5	2						

Algoritmia e Programação

TP9

Exercício 2

Elabore um programa modular com as seguintes funcionalidades:

a) Ler os vencimentos de N funcionários (a indicar pelo utilizador) para um array;
b) Verificar se existe no array um determinado vencimento, à escolha do utilizador;
c) Determinar a sua média;
d) Criar um array bidimensional com duas colunas contendo, na primeira coluna o valor do vencimento e na segunda coluna o respetivo desvio em relação à média;
e) Mostrar os vencimentos e respetivos desvios em relação à média;
f) Mostrar os valores dos vencimentos abaixo da média.
g) Ordenar os vencimentos por ordem decrescente;

Uma proposta de resolução

```
public class TP9_2 {

  static Scanner sc = new Scanner(System.in);
  public static void main(String[] args) {

    int numberOfWorkers = sc.nextInt();
    if (numberOfWorkers > 0) {
      double[] arrSalaries = new double[numberOfWorkers];
      double[][] arrDeviations;
      //a)
      readWorkersSalaries(arrSalaries);
      //b)
      double salaryToSearch = sc.nextDouble();
      if(thisSalaryExist(salaryToSearch, arrSalaries))
        System.out.println("There are workers with salary="+salaryToSearch);
      else
        System.out.println("There are no workers with salary="+salaryToSearch);
      //c)
      System.out.println("Average salary = "+averageSalary(arrSalaries));
      //d)
      arrDeviations=getDeviationOfSalariesFromAverage(arrSalaries);
      //e)
      printDeviationOfSalaries(arrDeviations);
      //f)
      printSalariesBelowAverage(arrDeviations);
      //g)
      sortSalaries(arrSalaries);
    }
  }
  //=================================================================

  public static void readWorkersSalaries(double[] arr) {
    for (int i = 0; i < arr.length; i++) {
      arr[i] = sc.nextDouble();
    }
  }
```

TP9

```
  //===================================================================

  public static boolean thisSalaryExist(double salary, double[] arr) {
    for (int i = 0; i < arr.length; i++) {
      if (arr[i] == salary) {
        return true;
      }
    }
    return false;
  }
  //===================================================================

  public static double averageSalary(double[] arr) {
    double sum = 0.0;
    for (int i = 0; i < arr.length; i++) {
      sum += arr[i];
    }
    return (sum / arr.length);
  }
  //===================================================================

  public static double[][] getDeviationOfSalariesFromAverage(double[] arr) {
    double[][] arrResult = new double[arr.length][2];
    double average = averageSalary(arr);
    for (int i = 0; i < arr.length; i++) {
      arrResult[i][0] = arr[i];
      arrResult[i][1] = arr[i] - average;
    }
    return arrResult;
  }
  //===================================================================

  public static void printDeviationOfSalaries(double[][] arr) {
    for (int i = 0; i < arr.length; i++) {
      System.out.printf("%7.2f € (%7.2f) %n",arr[i][0],arr[i][1]);
    }
  }
  //===================================================================

  public static void printSalariesBelowAverage(double[][] arr){
    for (int i = 0; i < arr.length; i++) {
      if(arr[i][1] < 0)
        System.out.printf("%.2f € %n",arr[i][0]);
    }
  }
  //===================================================================

  public static void sortSalaries(double[] arr) {
    for (int a = 0; a < arr.length - 1; a++) {
      for (int b = a + 1; b < arr.length; b++) {
        if (arr[a] < arr[b]) {
          double aux = arr[a];
          arr[a] = arr[b];
          arr[b] = aux;
        }
      }
    }
  }
}
```