

FICHA 01 – REVISÕES

O objectivo central desta ficha é promover a revisão de conhecimentos que os alunos já devem ter adquiridos nas disciplinas de programação anteriores, bem como permitir um primeiro contacto com a linguagem de programação Java.

Estrutura de um programa em Java

A um nível básico, a sintaxe do Java não difere muito da sintaxe do C, já conhecido de PPP. Em particular, as declarações e instruções de atribuição, selecção e repetição têm uma sintaxe igual.

Um exemplo pode ajudar a localizar as principais diferenças ao nível de programas simples. Trata-se de um programa para somar os dígitos de um número recebido do utilizador. Uma solução, em Java, para este problema pode ser (a negrito indicam-se as diferenças para uma solução semelhante em C):

```
import java.util.*;
public class SomaDigitos {
    public static void main(String[] args) {
        int num;
        int soma = 0;
        System.out.println ("Escreva o número: ");
        Scanner sc = new Scanner(System.in);
        num = sc.nextInt();
        while (num > 0) {
            soma = soma + (num % 10);
            num = num / 10;
        }
        System.out.print ("Soma dos dígitos = "+soma);
    }
}
```

1. Os programas em Java são constituídos por classes. Para já vamos ficar por programas só com uma classe. A declaração de uma classe tem a sintaxe **public class NomeDaClasse { ... }**

É claro que **NomeDaClasse** é definido pelo programador, devendo ser um identificador que identifique o propósito do programa. Esse mesmo nome terá que ser utilizado para o ficheiro que guarda o código do programa, neste caso seria `NomeDaClasse.java`.

2. O programa tem obrigatoriamente uma (e só uma) função com o seguinte cabeçalho:

```
public static void main(String[] args)
```

Tal como em C, a função **main** é a função principal, começando nela a execução do programa.

3. A escrita na consola é conseguida através de: **System.out.print ();** ou **System.out.println ();** que diferem apenas por a segunda promover uma mudança de linha após a escrita. Entre parêntesis coloca-se o que se pretende escrever, cadeias de caracteres entre aspas e nome de variáveis sem aspas, sendo utilizado o operador **+** para concatenar os diferentes elementos a escrever. Exemplos:

```
System.out.println ("Escreva o número: ");
```

```
System.out.print ("Soma dos dígitos = "+soma);
```

4. A leitura de informação do teclado pode ser feita recorrendo à classe **java.util.Scanner** (<http://java.sun.com/j2se/1.5.0/docs/api/java/util/Scanner.html>). Para criar uma instância desta classe escreva:

```
Scanner sc = new Scanner(System.in);
```

Esta classe inclui funções de leitura para os vários tipos primitivos. Por exemplo, para ler um inteiro:

```
num = sc.nextInt();
```

Para ler um **double** usa-se **sc.nextDouble()**, para um **float** **sc.nextFloat()** e para uma **string** **sc.nextLine()**.

Com esta informação e com os conhecimentos adquiridos em PPP espera-se que os alunos sejam capazes de resolver os problemas seguintes em Java, alguns dos quais já foram resolvidos na disciplina anterior.

Exercícios

1. Combinações

Escreva um programa que calcule o número de combinações de n elementos de um conjunto, tomados k a k , de acordo com a equação seguinte.

$$C_{n,k} = \frac{n!}{k!(n-k)!}$$

2. Somar até ao limite

Escreva um programa que vá somando todos os números inteiros começando pelo 1 e que termine quando a soma ultrapasse um limite indicado pelo utilizador. Por exemplo, se o utilizador tiver indicado 5 deverá adicionar os números 1, 2 e 3, visto que a sua soma dá 6 enquanto $1 + 2$ dá apenas 3. No final deve indicar o número em que parou.

3 Conversão binário - decimal

Desenvolva um algoritmo que receba do utilizador um número binário e indique no ecrã qual o número de zeros e uns que existem nesse número, bem como o seu equivalente decimal.

Exemplo:

b = 10110

no de zeros = 2

no de uns = 3

equivalente decimal = 22

4. Múltiplos

Construa um algoritmo para calcular os 4 menores múltiplos entre 0 e 100 de um número n qualquer desse intervalo. Implemente o algoritmo em C.

Exemplo:

$n = 25$

resultado: 0, 25, 50, 75

5. Números amigos

Dois números dizem-se amigos se a soma dos divisores de qualquer deles, incluindo a unidade e excluindo o próprio número, for igual ao outro número. Desenvolva um algoritmo que permita verificar se dois números m e n são números amigos.

Exemplo:

220 e 284 são números amigos

Divisores de 220: 1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110 Soma: 284

Divisores de 284: 1, 2, 4, 71, 142 Soma: 220

6. Números perfeitos

Um número perfeito é um número cuja soma dos seus divisores é o próprio número. Escreva um programa que leia um número $n > 3$ e determine os números perfeitos de 3 até n . Os números perfeitos encontrados deverão ser apresentados da seguinte forma:

Exemplo:

Número Perfeito: 6

Factores: 1 2 3

7. Característica

O número 153 tem uma propriedade interessante. Este número é igual à soma dos cubos dos seus dígitos, ou seja, $153 = 1^3 + 5^3 + 3^3$. Existem quatro números de três dígitos que possuem esta propriedade. Escreva um programa que encontre estes quatro números.

8. Replicação de Algarismos menos significativos

Dado um valor n , determine todos os números com n algarismos que possuem a característica de se replicar nos algarismos menos significativos, quando elevados ao quadrado.

Exemplo:

$n=2$

possui: 25 ($25 \cdot 25 = 625$)

possui: 76 ($76 \cdot 76 = 5776$)

9. Números primos

Implemente a função **int ePrimo (int x)**, que verifica se o número x é primo. Deverá devolver 1 no caso de ser verdade, e 0 no caso de ser falso.

Escreva um programa que calcule e imprima os números primos compreendidos entre 1 e um limite máximo pedido ao utilizador.

10. Capicua

Construa uma função **int inverte(int i)** que inverte a ordem dos dígitos de um número inteiro. Usando essa função, pretende-se que construa um programa que verifique se um número inteiro dado é uma capicua, ou seja, se se lê da mesma forma do princípio para o fim e do fim para o princípio. Por exemplo, 1221 é capicua, 121 também.

11. Mínimo múltiplo comum

Escreva uma função que determine o mínimo múltiplo comum de dois números inteiros n e m que lhe são fornecidos como parâmetros. Utilize essa função para listar o mmc entre todos os pares de números situados num intervalo definido pelo utilizador.

12. Logaritmos binários

O logaritmo binário inteiro de um número é dado pelo número de vezes que esse número pode ser dividido por 2 até que o resultado da divisão seja inferior a 2. Por outras palavras, o logaritmo binário de x é a maior potência de 2 menor ou igual a x . Por exemplo, $\lg(7.9) = 2$ e $\lg(8) = 3$.

Escreva um programa em Java que calcule e imprima os logaritmos binários de todos os números múltiplos de 100 entre 100 e 1000. Estructure devidamente o seu programa.

13. Número contido

Escreva uma função que permita verificar se um dado inteiro x , entre 1 e 99, está contido num número inteiro n , maior do que x .

Exemplificando:

$x = 7$ e $n = 1977$ -- Sim

$x = 56$ e $n = 12345567$ -- Sim

$x = 54$ e $n = 12345567$ -- Não

Utilizando esta função crie um programa que receba do utilizador o valor de x e escreva no ecrã todos os números inteiros menores que 1000 que contêm x .