Aula 01

Introdução à Programação

O computador e os elementos básicos da linguagem Java

Programação 1, 2015-2016

v1.1, 01-10-2015

, DETI, Universidade de Aveiro

01.1

Conteúdo

01.2

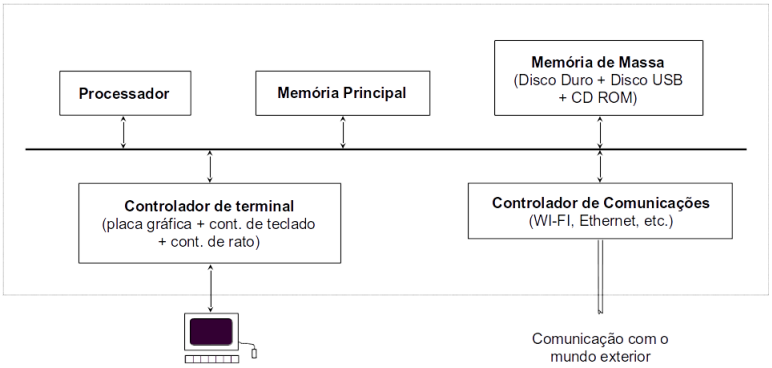
1 Organização de computadores

Computador. . .

- Máquina programável que processa informação de forma autónoma.
- Executa, com uma cadência muito rápida, sequências de operações elementares sobre informação recebida, devolvendo ao utilizador resultados.
- A sequência de operações elementares, designada habitualmente por programa, pode ser alterada ou substituída por outra, sempre que se deseje.
- Durante a execução do programa, a sequência de operações elementares e os valores temporários produzidos estão armazenados num dispositivo interno, chamado memória.

01.3

Organização de um computador



01.4

Organização de um computador

- Todo o tipo de informação pode ser representada por números (desde que deles se faça a devida interpretação).
- É isso que é feito nos computadores onde informação como imagens, som, texto, programas, documentos, etc., são armazenados e processados como números.
- Pelo facto dos seres humanos serem dotados de 10 dedos, o sistema numérico mais utilizado é o decimal.
- No entanto, qualquer base numérica, começando na binária, permite representar qualquer quantidade.

01.5

Organização de um computador

- Na sua estrutura mais fundamental, os computadores fazem uso do sistema binário, sendo um *bit* a sua quantidade mais pequena (valor 0 ou 1) e um *byte* o conjunto de 8 bits.
- A forma como a informação é localizada em computadores recorre a *endereços de memória* que mais não são do que a posição da informação na memória do computador.
- Assim, o armazenamento e acesso de informação num computador requer o conhecimento do seu endereço e do número de bytes ocupados.
- A interpretação adequada dessa informação é feita pelos programas que as operam.

01.6

2 Programação: Resolução de problemas

Homem Vs. Computador

Homem	**Computador**
a abordagem é criativa	a abordagem não é criativa
• aprende com a experiência passada; • associa conceitos distintos, conseguindo isolar elementos comuns; • usa em larga medida um raciocínio de tipo indutivo (intuição);	• não tem capacidade directa de aprendizagem; • só associa conceitos cuja afinidade foi previamente estabelecida; • usa mecanismos de raciocínio dedutivo (lógico);

01.7

Homem	**Computador**
propõe soluções	não propõe soluções
• descobre métodos de resolução;	• possibilita a validação das soluções encontradas;
comete erros	não comete erros
• as inferências produzidas são muitas vezes incorrectas; • está sujeito a lapsos de concentração provocados por distracção ou cansaço.	• salvo avaria, limita-se a executar de um modo automático a sequência de operações estabelecida.

01.8

Tipos de problemas que o computador resolve

Problemas completamente especificados:

- as variáveis de entrada e de saída estão perfeitamente identificadas;
- se conhece uma solução; ou seja, um método que permite obter, de forma unívoca, os valores das variáveis de saída em função dos valores das variáveis de entrada;
- deve considerar-se sempre a resolução dos problemas no âmbito mais lato possível; ou seja, deve considerar-se a resolução de classes de problemas e não de problemas particulares;
- a gama de valores permitida para as variáveis de entrada deve ser claramente estabelecida;
- a solução descrita deve contemplar alternativas para toda a gama de valores das variáveis de entrada, eliminando toda e qualquer ambiguidade.

01.9

Exemplo de um problema

Conversão de distâncias (milhas para Km): Dada uma distância, expressa em milhas, que é lida do teclado, convertê-la para quilómetros e escrevê-la no ecrã do computador (terminal).

Variável de entrada:

`MILHAS` (distância expressa em milhas)
valor numérico positivo ou nulo

Variável de saída:

`KILOMETROS` (distância expressa em quilómetros)
valor numérico positivo ou nulo

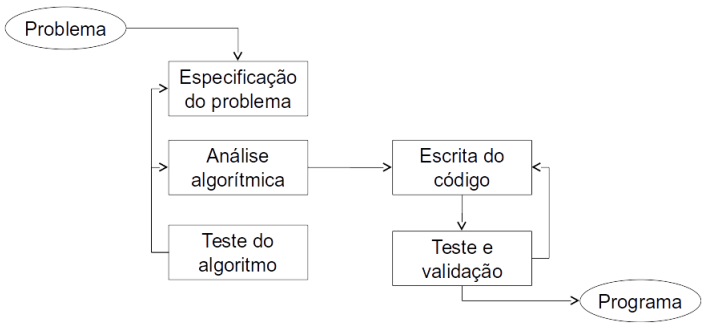
Solução:

```
KILOMETROS = 1.609 * MILHAS
```

01.10

3 Fases de desenvolvimento de um programa

- As duas etapas básicas do desenvolvimento de um programa são a *análise do problema* e a *implementação da aplicação*.



01.11

Algoritmo

- Designa-se por *algoritmo* a descrição detalhada e rigorosa da solução do problema.
- A transcrição do algoritmo para uma linguagem de programação dá origem ao *programa*.
- Supõe-se que o conjunto de operações descrito no algoritmo é realizado segundo uma ordem préestabelecida: só se inicia uma dada operação, quando a anterior estiver terminada - *execução sequencial*.
- Exemplo:

```
leitura dos valores das variáveis de entrada
processamento
escrita dos valores das variáveis de saída
```

01.12

Algoritmo

- A *programação* assenta na concepção de algoritmos que utilizando as "peças do lego" fornecidas pela linguagem de programação utilizada, tentam resolver os problemas.
- Curiosamente, são muito poucas as "peças do lego" absolutamente necessárias para resolver qualquer problema computável:
  - Registo de informação;
  - Sequência de comandos;
  - Instrução condicional;
  - Instrução repetitiva;
  - Comandos básicos de comunicação com o exterior do programa.

01.13

4 Programação em Java

Estrutura de um programa em Java

```
// inclusão de classes externas

public class Programa
{
   // declaração de constantes e variáveis globais

   public static void main (String[] args)
   {
      // declaração de constantes e variáveis locais
      // sequências de instruções
   }
}

// definição de tipos de dados (registos)
```

01.14

Exemplo de um programa

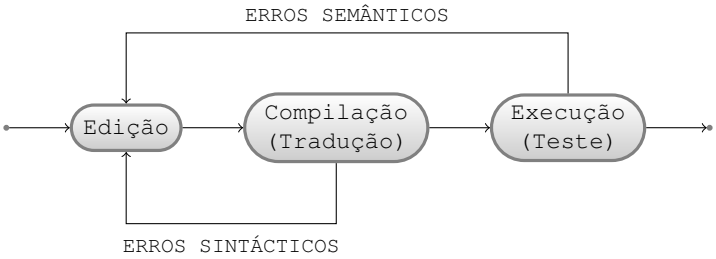
Ficheiro `MilesToKm.java`

```
import java.util.Scanner;

public class MilesToKm {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      double miles; // variável de entrada
      double km;    // variável de saída
      // ler entrada
      System.out.print("Distance (miles): ");
      miles = sc.nextDouble();
      // processamento
      km = 1.609 * miles;
      // escrever saída
      System.out.println("Distance (km): " + km);
   }
}
```

01.15

Desenvolvimento de um programa



- Edição:

```
geany MilesToKm.java
```

- Compilação

```
javac MilesToKm.java
```

- Execução

```
java MilesToKm
```

01.16

Elementos básicos da linguagem Java

- **Palavras reservadas** – símbolos que têm um significado bem definido em Java e que não podem ser usadas para outro fim (ex. `class`, `break`, `switch`, `final`, `if`, `then`, `else`, `while`, ...).
- **Identificadores** – nomes utilizados para designar todos os objectos existentes num programa. Devem começar por uma letra ou por símbolo '_' e só podem conter letras, números e o símbolo '_' (ex. `nome`, `idade`, `i`, `j`, `cont_1`, `dia_mes`, `res`, `_km`, ...).
- **Comentários** – melhoram a legibilidade de um programa (todos os caracteres na mesma linha que se seguem ao símbolos '`//`' e blocos de código delimitados por `/*` e `*/`).

01.17

Elementos básicos da linguagem Java

- **Constantes** – valor específico de um certo tipo (ex. `10`, `10`, `5.5`, `-0.8`, `'0'`, `"Aveiro"`, `true`, ...).
- **Operadores e separadores** – símbolos ou combinações de símbolos que especificam operações e usados na construção de instruções: `( ) [ ] { } < > ; . , : ? ! ' " & | = + - * / % ~ ^ # \_ $`

01.18

Registo de informação (variáveis)

- Uma variável serve para registar informação num programa (para tal ocupa uma posícão de memória no computador) e pode ser considerada como uma caixa cujo conteúdo inicialmente não está definido.
- Definição de uma variável:

```
tipo identificador nome_var_1, nome_var_2, ...;
```

- Exemplos de definição de variáveis e constantes:

```
double peso, altura, largura, erro;
int idade, dia_mes, ano;
boolean resultado;
char letra, op;
final double PI = 3.1415; // constante real
final int LIMITE = 100;   // constante inteira
```

01.19

Tipos de dados primitivos

- `byte`, `short`, `int`, `long` – números inteiros (`10`, `-10`, `0`, ...)
- `float`, `double` – números reais (`10.5`, `-10.5`, `.2`, ...)
- `boolean` – apenas dois valores possíveis (`true`, `false`)
- `char` – caracteres (`'a'`, `'1'`, `'!'`, ...)
- Para a linguagem Java cada um destes tipos de dados, implica uma representação específica dessa informação no computador.
- Por exemplo, o tipo de dados `int` implica que irão ser utilizados 4 bytes para o armazenamento destes valores inteiros (permite representar valores no intervalo: $[-2^{31}; 2^{31}-1]$).

01.20

Inicialização de variáveis

- Antes de uma variável poder ser utilizada tem de ser-lhe atribuído um valor:
  - na altura da definição:

```
double num = 10.5;
int idade = 18;
```

  - usando uma instrução de atribuição (símbolo '='):

```
double peso;
peso = 50.5;
```

  - lendo um valor do teclado ou de outro dispositivo (ex. ficheiro):

```
double milhas;
System.out.print("Valor real: ");
milhas = sc.nextDouble();
```

Nomes de variáveis: Escolher nomes para as variáveis que tornem clara qual é a informação por elas registada.

01.21

Conversões de tipos

- Sempre que uma expressão tenha operandos aritméticos de tipos diferentes, os operandos com menor capacidade de armazenamento são automaticamente convertidos para o tipo com maior capacidade:

  byte → **short** (ou **char**) → **int** → **long** → **float** → **double**

- A conversão inversa não é admitida e gera um erro de compilação.
- Podemos sempre forçar uma conversão através de um operador de conversão (*cast* em inglês):

```
double x;
int y;
y =(int)x; // estamos a forçar a conversão para int
```

01.22

Operadores e expressões

- Operadores:
  - Aritméticos: `*`, `/`, `+`, `-`, `%`
  - Relacionais: `<`, `<=`, `>`, `>=`, `==`, `!=`
  - Lógicos: `!`, `||`, `&&`
  - Manipulação de bits: `&`, `~`, `|`, `^`, `>>`, `<<`
- Expressões:

```
int x, z;
double y;
x = 10 + 20; // o valor 30 é armazenado em x
y = 8.4 / 4.2;// o valor 2.0 é armazenado em y
```

  - As expressões são calculadas da esquerda para a direita.
  - Atenção às prioridades dos operadores e aos parênteses.

01.23

Operadores aritméticos unários

- simétrico: `- (-x)`
- incremento de 1: `++ (++x, x++)`
- decremento de 1: `-- (--x, x--)`
- Os operadores unários de incremento e decremento só podem ser utilizados com variáveis e atualizam o seu valor de uma unidade.
- Colocados antes são pré-incremento e pré-decremento. Neste caso a variável é primeiro alterada antes de ser usada.
- Colocados depois são pós-incremento e pósdecremento e neste caso a variável é primeiro usada na expressão onde está inserida e depois atualizada.

01.24

Alguns módulos (classes) da linguagem Java

- A linguagem java disponibiliza um vasto conjunto de classes que permitem manipular dados e realizar diversas operações. Serão apresentadas conforme forem sendo necessárias. Ficam três exemplos:
- Classe `Math`:

```
double Math.cos(double);
double Math.acos(double);
double Math.sin(double);
double Math.asin(double);
double Math.sqrt(double);
double Math.pow(double, double);
double Math.toRadians(double);
```

- Classe `Integer` e `Double`:

```
Integer.MAX_VALUE
Integer.MIN_VALUE
```

```
Double.MIN_VALUE
Double.MAX_VALUE
```

01.25

Leitura de dados

- Leitura de informação digitada no teclado (`System.in`);
- Interpretação e conversão feita pela classe `Scanner` (criada para ler do teclado);
- É necessário indicar a localização desta classe:

```
import java.util.Scanner;

ou:

import java.util.*;
```

- Utiliza-se a operação (método) com a conversão desejada: `nextInt()`, `nextDouble()`, `nextLine()`, ...
- Exemplo:

```
Scanner sc = new Scanner(System.in);
int x;
x = sc.nextInt();
```

01.26

Escrita de dados

- Escrita de informação no terminal (`System.out`);
- Interpretação e conversão feita pela classe `PrintStream` (não é necessário indicar a localização desta classe);
- Utilizam-se os seguintes métodos: `print()`, `println()`, `printf()`, ...
- Exemplos:

```
   // não muda de linha:
System.out.print("The value of x is " + x);
   // muda de linha:
System.out.println("The value of x is " + x);
   // formatada:
System.out.printf("The value of x is %3d\n", x);
```

Informação ao utilizador: Quando se pede informação ao utilizador (leitura) é uma boa prática informá-lo sobre o que está a ser pedido.

01.27

Escrita formatada

- A função printf permite escrever informação formatada:

```
System.out.printf("formato de escrita", lista de variáveis);
```

- O formato de escrita é uma sequência de caracteres, que pode conter especificadores de conversão.
- O especificador de conversão é composto pelo símbolo `%` seguido de um caracter que indica qual o tipo de dados que queremos escrever: `%d`, `%f`, `%c`, `%s`, ...
- Este caracter pode ser precedido de um número com o qual se controla o formato: `%3d`, `%5.1f`, `%3c`, `%10s`, ...
- Exemplo:

```
System.out.printf("Int.: %6d", 15);      // Int.: ____15
System.out.printf("Real: %6.2f", 14.2); // Real: _14.20
```

01.28