INTRODUÇÃO À PROGRAMAÇÃO

2018/2019

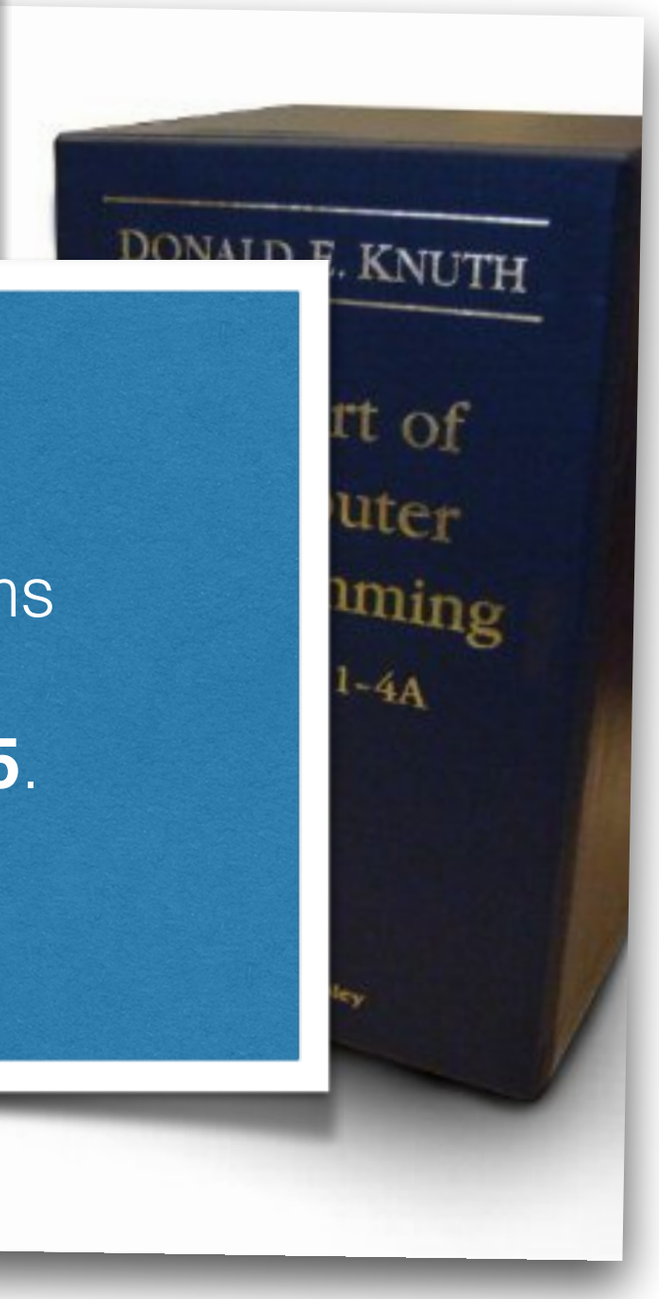
e-learning.iscte-iul.pt

O QUE É PROGRAMAR?

Knuth began the project, originally conceived as a single book with twelve chapters, in **1962**. The first three of what was then expec
publishe
instalmer
was publ

Volume 5: Syntactic Algorithms

Estimated to be ready in **2025**.



A ARTE DE RESOLVER PROBLEMAS

"when we prepare a program, it can be like composing poetry or music"

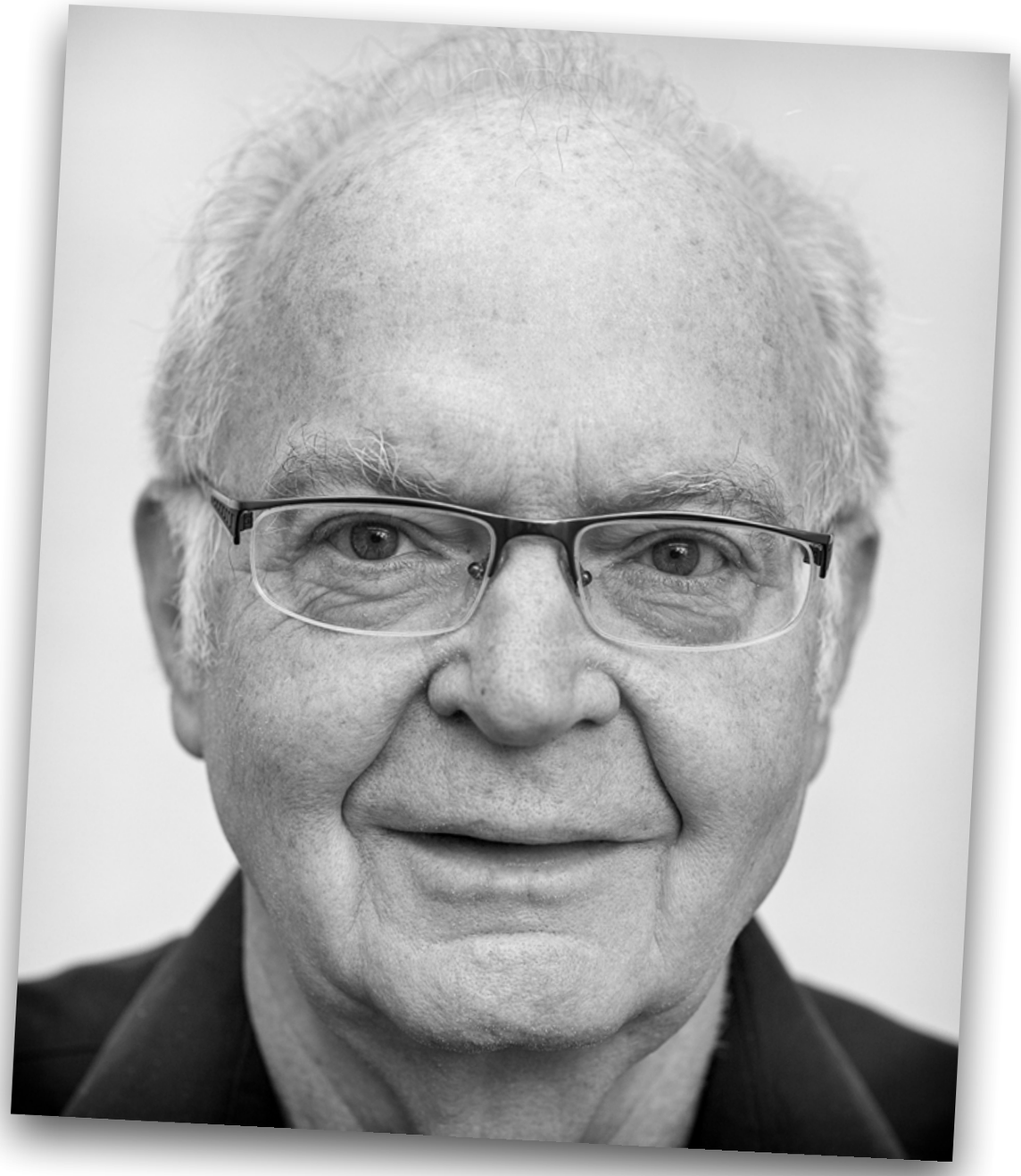
- Se o programador consegue ensinar um computador a resolver um problema, então o programador também o consegue resolver!

- Porquê arte?

  - *Porque na programação aplicamos conhecimento acumulado sobre o mundo*

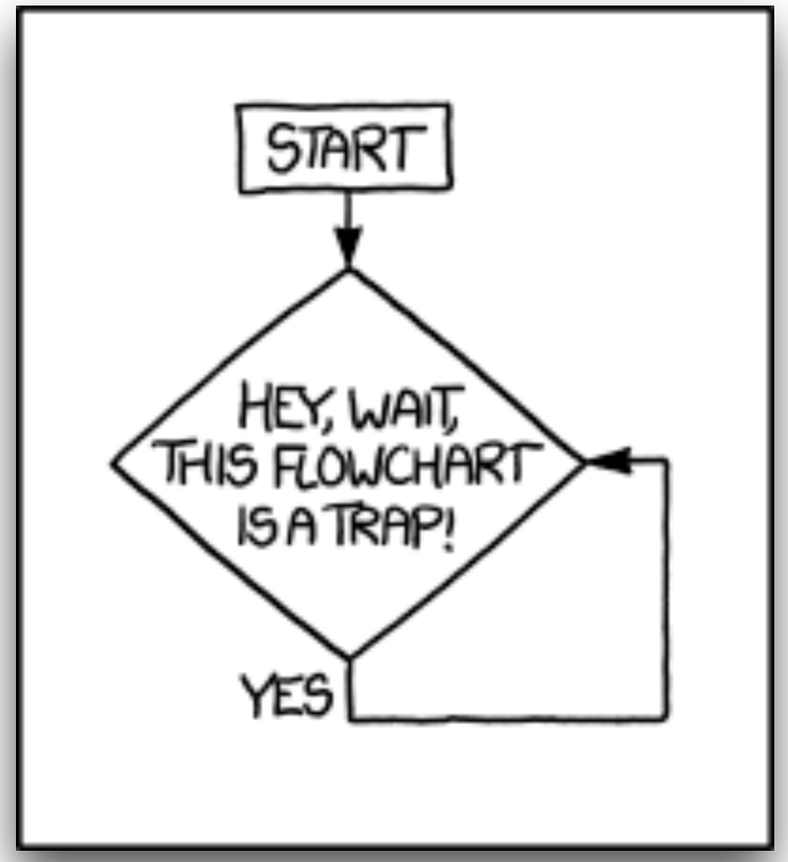
  - *Porque requer habilidade, destreza, engenho, inteligência, e criatividade*

  - *Porque produz objetos de beleza*



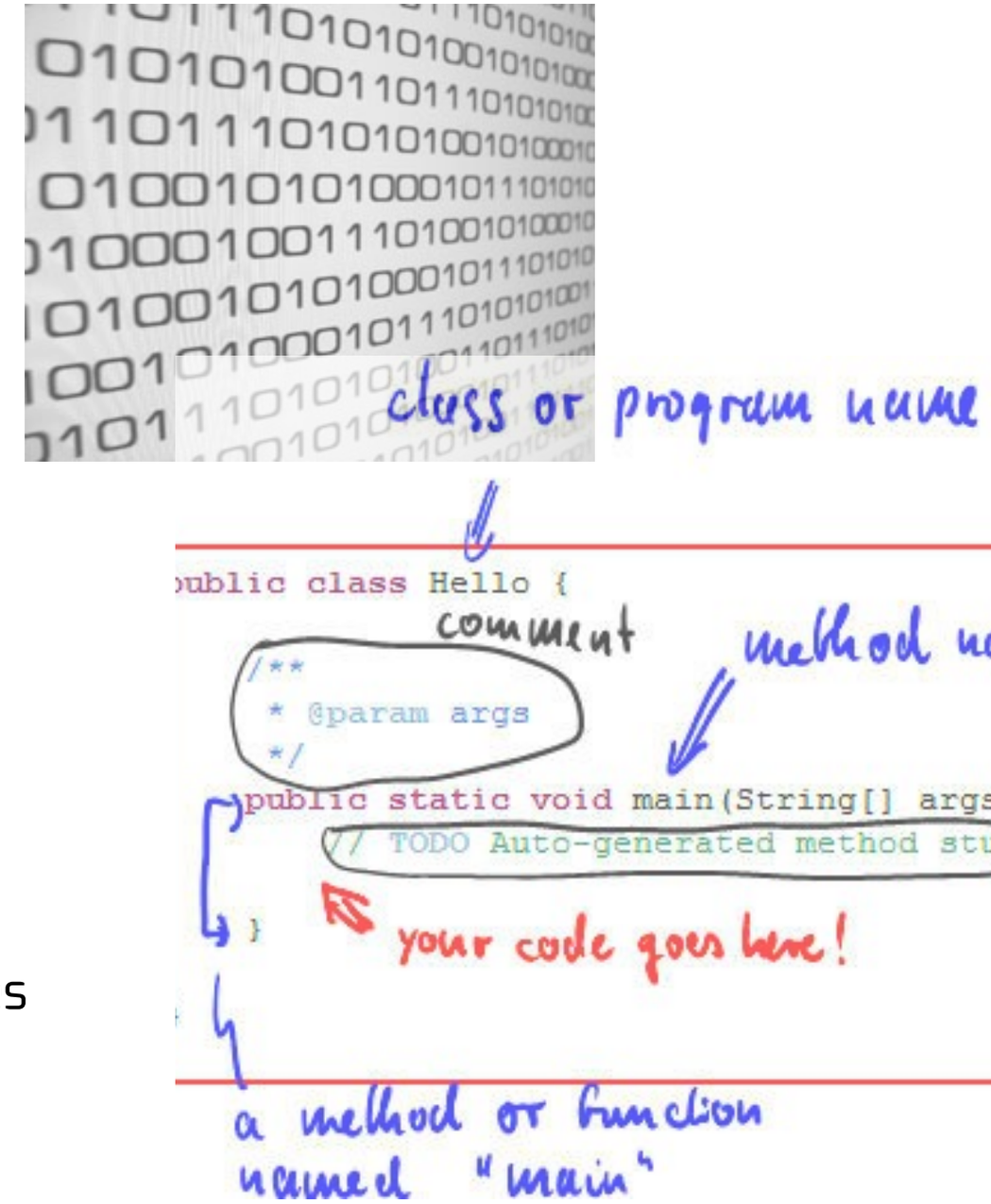
ALGORITMO

- Método de resolução de problema
- Características:
  - **Finitude**: tem de terminar
  - **Definitude**: cada passo bem definido
  - **Entradas**: zero ou mais, de conjunto bem definido
  - **Saídas**: uma ou mais, dependem das entradas
  - **Eficácia**: operações todas executáveis



COMO *ENSINAR* O COMPUTADOR?

- Linguagem natural
  - Português, Inglês, Francês, …
  - Ambígua e imprecisa
- Linguagem máquina
  - Muito básica: usada pelos computadores
- Linguagem de programação de alto nível
  - FORTRAN, C, Java, C++, Python, …
  - Sem ambiguidades nem imprecisões
  - Não tão penosa como linguagem máquina



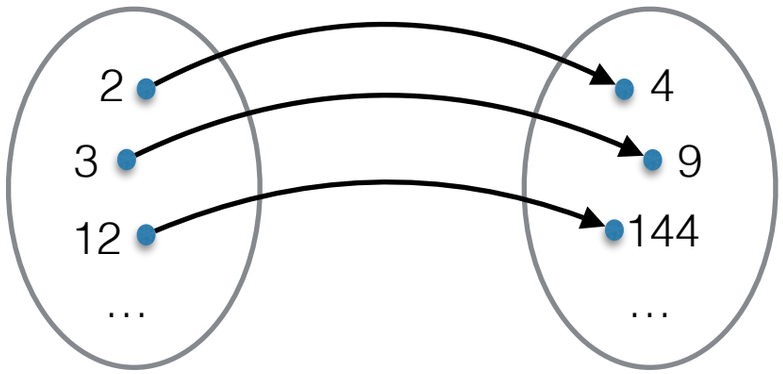
FUNÇÕES

FUNÇÕES

- Em **Matemática**, uma função *f()* associa a um argumento *x*, um valor *y*. Ou seja, $y = f(x)$.

  Se $f(x) = x^2$

  então $f(3) = 9$



- Em **Programação**, o conceito de função é similar

```
static int square(int x) {
    return x*x;
}
```

- As funções são **implementações** de métodos de resolução de problemas.

FUNÇÕES: ESPECIFICAÇÃO

- Para que seja possível utilizar uma função, tal como na Matemática, esta tem de ter um nome. O nome da função deve indicar aquilo que a função calcula.

- Uma função tem uma especificação, onde se define o **nome**, o **tipo do valor devolvido** (o que sai), e os **parâmetros** da função (o que entra)

```
static int square(int x) {
    …
}
```

Tipo de devolução

Nome

Parâmetros

FUNÇÕES: CORPO

- O corpo de uma função é a **explicação ao computador do método de resolução de um problema**. Esta definição aparece a seguir à especificação da função, entre chavetas.

```
static int square(int x) {
    return x*x;
}
```

FUNÇÕES: PARÂMETROS E ARGUMENTOS

- Os **parâmetros** indicam a que informação é que a função se aplica
  - **x** é um parâmetro do tipo **`int`**
  - Em termos matemáticos, diríamos que o domínio da função é o conjunto dos inteiros
- Os **argumentos** são os valores que se dão aos parâmetros
  - 0, 2, 3, -1 são possíveis argumentos

FUNÇÕES: PARÂMETROS E ARGUMENTOS

- Uma função pode ter **vários parâmetros**, por exemplo:

```
static int sum(int a, int b) {
    …
}
```

- (2, 3), (-1, 5), (10, 20) são pares de possíveis argumentos

FUNÇÕES: DEVOLUÇÃO

- A especificação da função define que o tipo de devolução é um inteiro (`int`)

```
static int square(int x) {
    return x*x;
}
```

- Em termos matemáticos, diríamos que o contradomínio da função é o conjunto dos inteiros

- A instrução **`return`** indica qual o valor devolvido pela função

  - O valor devolvido tem de ser compatível com o que é indicado na especificação da função (neste caso, um número inteiro)

  - A expressão **`x*x`** calcula o quadrado de **`x`** e a instrução **`return`** devolve o valor calculado

TIPOS PRIMITIVOS

`int`	*inteiros*	`1, 2, -1, 0, 70000`
`double`	*decimais*	`3.14, 0.0, -1.2`
`char`	*caracteres*	`'a', '8', '?', '-'`
`boolean`	*booleanos*	`true, false`
...		

CONVERSÕES ENTRE TIPOS NUMÉRICOS

- `double` para `int`

```
(int)7.3    →    7
(int)7.8    →    7
```

  - Esta conversão não corresponde a um arredondamento. O número decimal é ***truncado***!

- `int` para `double`

```
(double)8    →    8.0
```

OPERADORES ARITMÉTICOS

+	*adição*
−	*subtração*
*	*multiplicação*
/	*divisão*
%	*resto da divisão inteira*

O operador da divisão, ao ser utilizado com dois valores inteiros, efectua uma divisão inteira (7/2 → 3 e 7%2 → 1).

O tipo do resultado de qualquer operação é o mesmo dos tipos dos operandos; quando estes são diferentes (mas compatíveis), o tipo do resultado é o tipo mais expressivo.

OPERADORES RELACIONAIS

==	*igual*
!=	*diferente*
<	*menor*
<=	*menor ou igual*
>	*maior*
>=	*maior ou igual*

OPERADORES LÓGICOS

&&	*conjunção (e)*
||	*disjunção (ou)*
!	*negação*

FUNÇÕES: EXEMPLO

```
static boolean isEven(int n) {
    return n%2 == 0;
}
```

- Esta função verifica se um dado número **n** é par.
- O tipo de devolução é booleano (`boolean`).
  - A função devolve verdadeiro (**`true`**) se **`n`** é par e falso (**`false`**), caso contrário.
- O resultado da expressão **`n%2`** irá ser `0` ou `1`. Caso seja igual a zero (`== 0`), então a expressão **`n%2 == 0`** é verdadeira, caso contrário é falsa. Esta avaliação determinará o valor a devolver.

CLASSES

- As funções em Java são obrigatoriamente definidas dentro de módulos designados por **classes**

```
class SimpleMath {
    static int square(int x) {
        return x*x;
    }

    static boolean isEven(int n) {
        return n%2 == 0;
    }

    …
}
```

CLASSES ESTÁTICAS

- Em Java, o código das classes tem obrigatoriamente de estar definido num ficheiro com o nome da classe e extensão **.java**

- Em Java, existem outros módulos designados por classes mas com propósitos diferentes

- Designaremos por **classe estática** uma classe que apenas contém funções estáticas

A RETER

- Funções
  - Especificação
    - Nome
    - Tipo de devolução
    - Parâmetros (e argumentos)
  - Corpo (implementação)
- Tipos primitivos
- Operadores
- Classes estáticas