

INTRODUÇÃO À PROGRAMAÇÃO

2015/2016

RICARDO RIBEIRO

ricardo.ribeiro@iscte.pt, D6.21

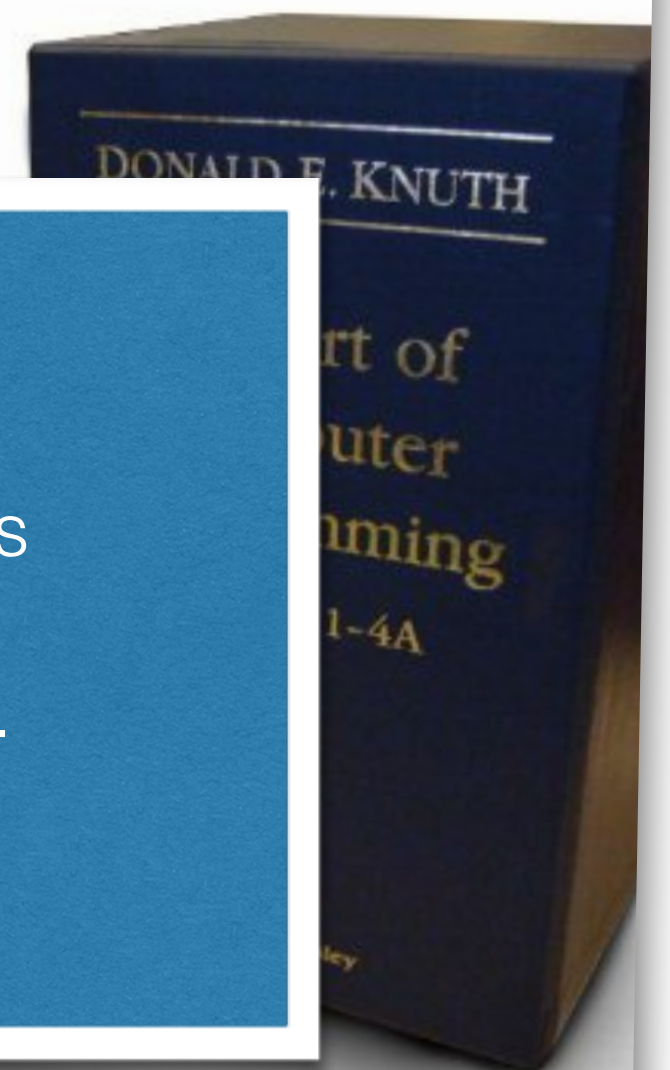
e-learning.iscte-iul.pt

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 expected to be a single volume were published in instalments. The first instalment was published in 1968.

Volume 5: Syntactic Algorithms

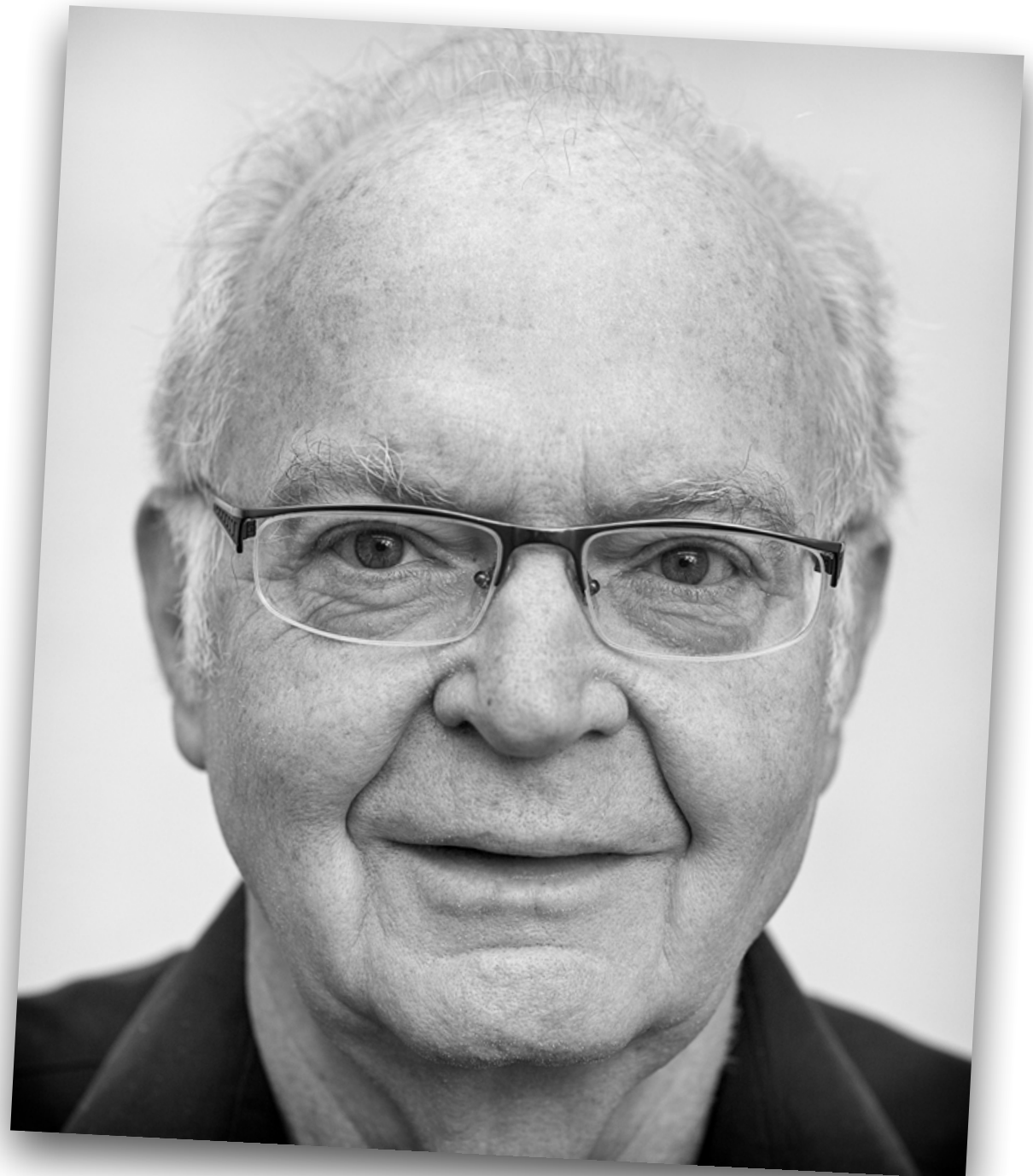
Estimated to be ready in **2020**.



A ARTE DE RESOLVER PROBLEMAS

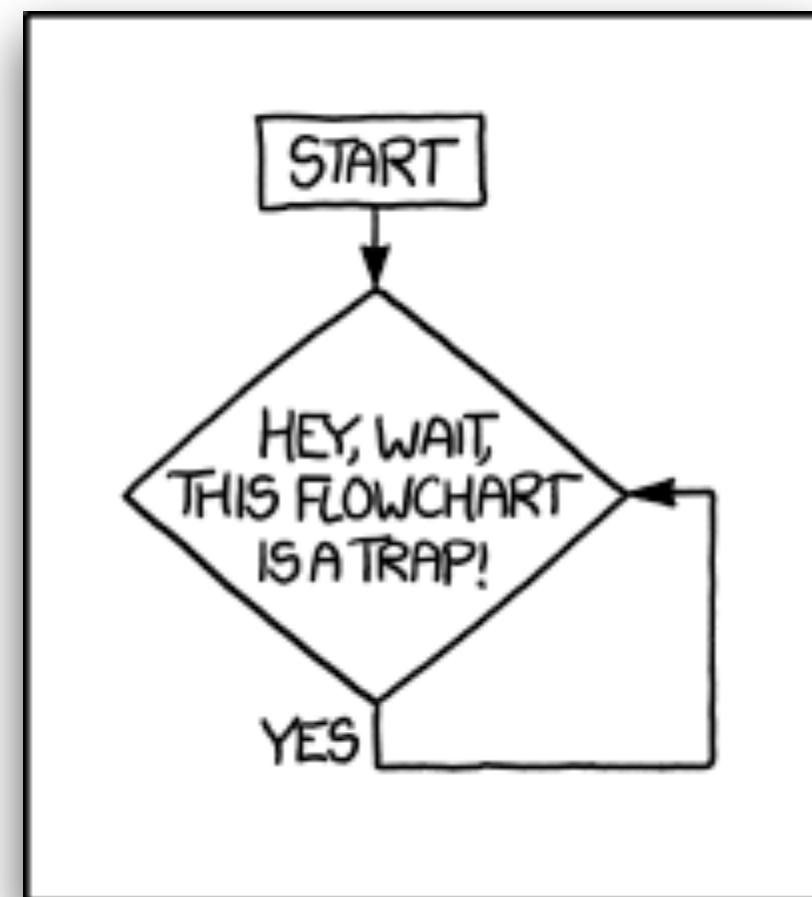
"when we prepare a program, it can be like composing poetry or music"

- 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



class or program name

```
public class Hello {  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
    }  
}
```

comment

method name

your code goes here!

a method or function named "main"

FUNÇÕES E PARÂMETROS

FUNÇÕES

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

$$\text{Se } f(x) = x^2$$

$$\text{então } f(3) = 9$$

- 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: ASSINATURA

- 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 assinatura, 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) {  
    ...  
}
```


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 assinatura 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 assinatura 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 assinatura 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	<i>inteiros</i>	1, 2, -1, 0, 70000
double	<i>decimais</i>	3.14, 0.0, -1.2
char	<i>caracteres</i>	'a', '8', '?', '-'
boolean	<i>booleanos</i>	true, false
...		

CONVERSÕES ENTRE TIPOS NUMÉRICOS

- `int` para `double`

`(double)8` → `8.0`

- `double` para `int`

`(int)7.3` → `7`

`(int)7.8` → `7`

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

OPERADORES ARITMÉTICOS

+	<i>adição</i>
-	<i>subtração</i>
*	<i>multiplicação</i>
/	<i>divisão</i>
%	<i>resto da divisão inteira</i>

O operador da divisão, ao ser utilizado com dois valores inteiros, efectua uma divisão inteira ($7/2 \rightarrow 3$).

OPERADORES RELACIONAIS

<code>==</code>	<i>igual</i>
<code>!=</code>	<i>diferente</i>
<code><</code>	<i>menor</i>
<code><=</code>	<i>menor ou igual</i>
<code>></code>	<i>maior</i>
<code>>=</code>	<i>maior ou igual</i>

OPERADORES LÓGICOS

&&	<i>conjunção (e)</i>
	<i>disjunção (ou)</i>
!	<i>negação</i>

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 Functions {  
  
    static int square(int x) {  
        return x*x;  
    }  
  
    static int 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
 - Assinatura
 - Nome
 - Tipo de devolução
 - Parâmetros (e argumentos)
 - Corpo (implementação)
- Tipos primitivos
- Operadores
- Classes estáticas

