Introdução

Programação I
2020.2021

Teresa Gonçalves
tcg@uevora.pt

Departamento de Informática, ECT-UÉ

Sumário

O que é a Programação?

Linguagem de Programação

Como Programar?

O que é a Programação?

Programação

O que é?

Conceção de métodos para resolução de problemas usando computadores

Criação de **programas** informáticos

Competências

Matemática

linguagens formais para especificar ideias

Engenharia

projectar, unir componentes para formar um sistema, avaliar prós/contras de alternativas

Ciências naturais

observar comportamento de sistemas complexos, tecer hipóteses, testar previsões

Programa

O que é?

Sequência de instruções (escritas numa linguagem de programação) para controlar o comportamento de um sistema

Objetivo

Executar uma computação

Fazer cálculos, controlar periféricos, desenhar gráficos, realizar ações

Input/Output

Input: dados necessários para executar a computação

Output: resultado da computação

Linguagem de Programação

Linguagem de programação

O que é?

Linguagem formal concebida para exprimir computações

Linguagem formal

Sintaxe: regras “gramaticais”

Semântica: associação de significados ou ações

Exemplos

- Expressões aritméticas: 3+3=6
- Estrutura molecular: H_2O

Linguagem natural vs Linguagem formal

Linguagem natural

- Utilizada pelas pessoas (Português, Inglês, …)
- Inclui ambiguidade
  - “O João viu a Maria no parque com os binóculos”
  - Propensa a erros/diferenças de interpretação

Linguagem formal

- Não permite ambiguidade*
  - Significado literal, claro e independente do contexto

** Por vezes aceita-se ambiguidade mas reduzida*

Linguagem de baixo nível

Código máquina

- Linguagem nativa dos computadores
- Exemplo: `100011 00011 01000 00000 00001 000100`
- Características
  - Única linguagem diretamente executável pelo computador
  - **Difícil** compreensão
  - Específica para a arquitetura do computador

Assembly

- Utiliza mnemónicas (texto) para representar código máquina
- Exemplo: `addi $t0, $zero, 100`
- Assemblador: programa que traduz assembly para código máquina

Linguagem de alto nível

Mais próxima da formulação matemática dos problemas

Facilita a escrita, a leitura, a resolução dos problemas

Exemplos

C, Java, Prolog, Python, ...

Características

Mais fácil de entender

Portável

Permite a execução em diferentes arquiteturas de computadores

Traduzida para código máquina por interpretadores ou compiladores

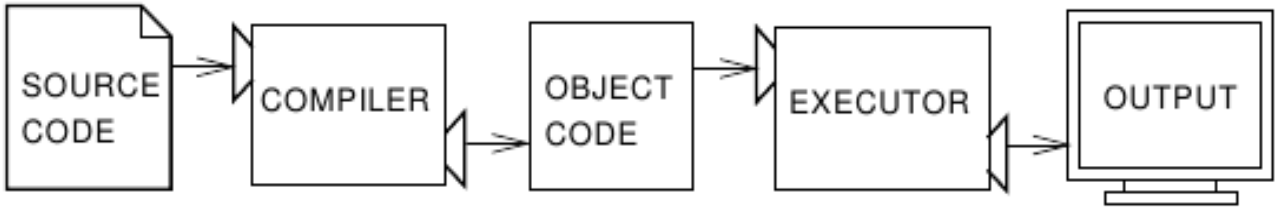
Interpretador vs compilador

Interpretador

Lê, interpreta e executa uma instrução de cada vez

Compilador

Traduz o programa para código máquina executável



Porquê tantas linguagens?

Diferentes níveis de abstração

Alto nível: facilita a programação e a deteção e correção de erros

Baixo nível: possivelmente mais eficiente

Diferentes tipos de problemas

Cálculos numéricos: Fortran

Raciocínio: Prolog

Scripting: Perl, Python

Diferentes paradigmas

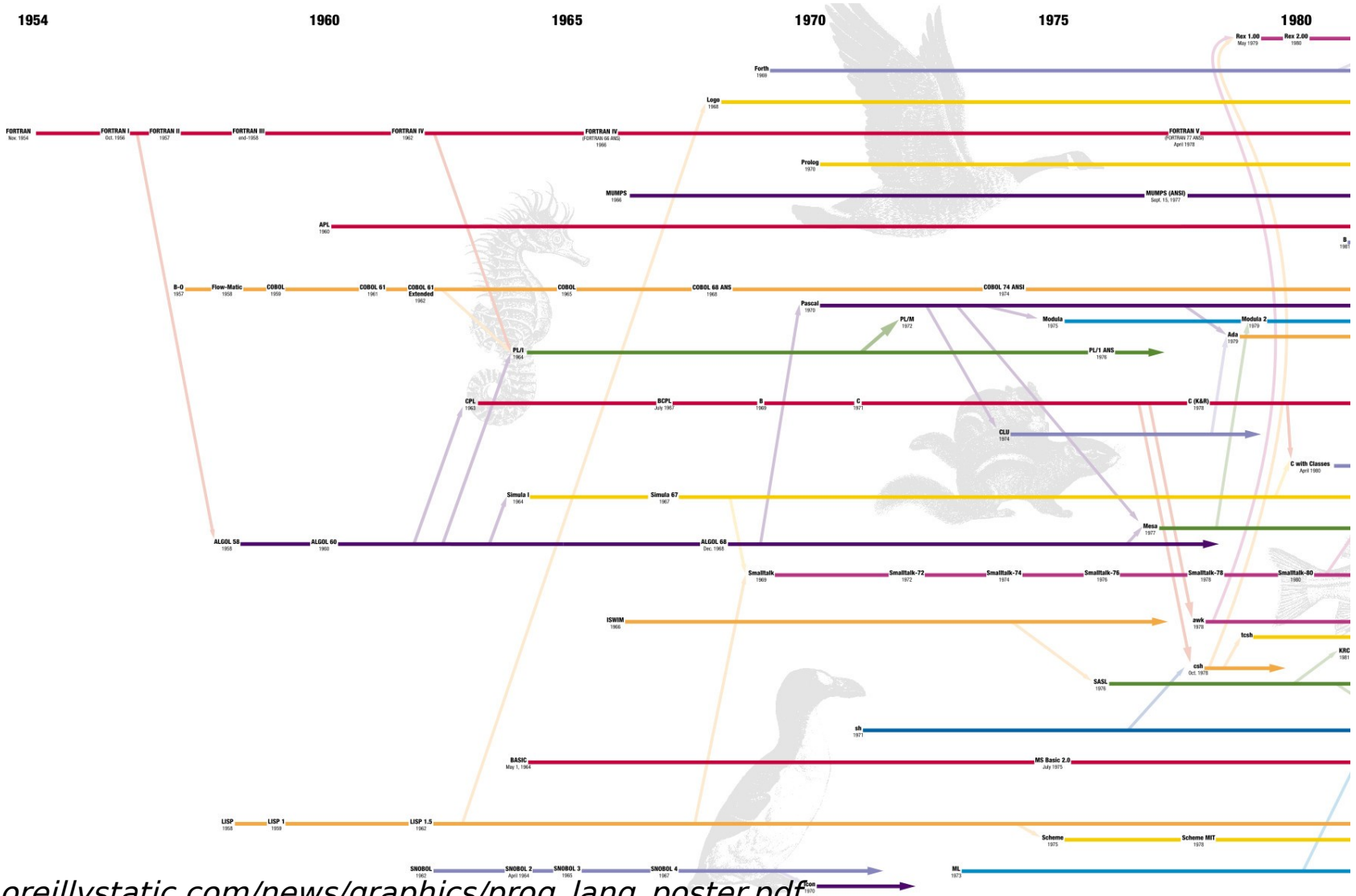
Imperativo: C, Pascal

Funcional: Haskell, Caml

Lógico: Prolog

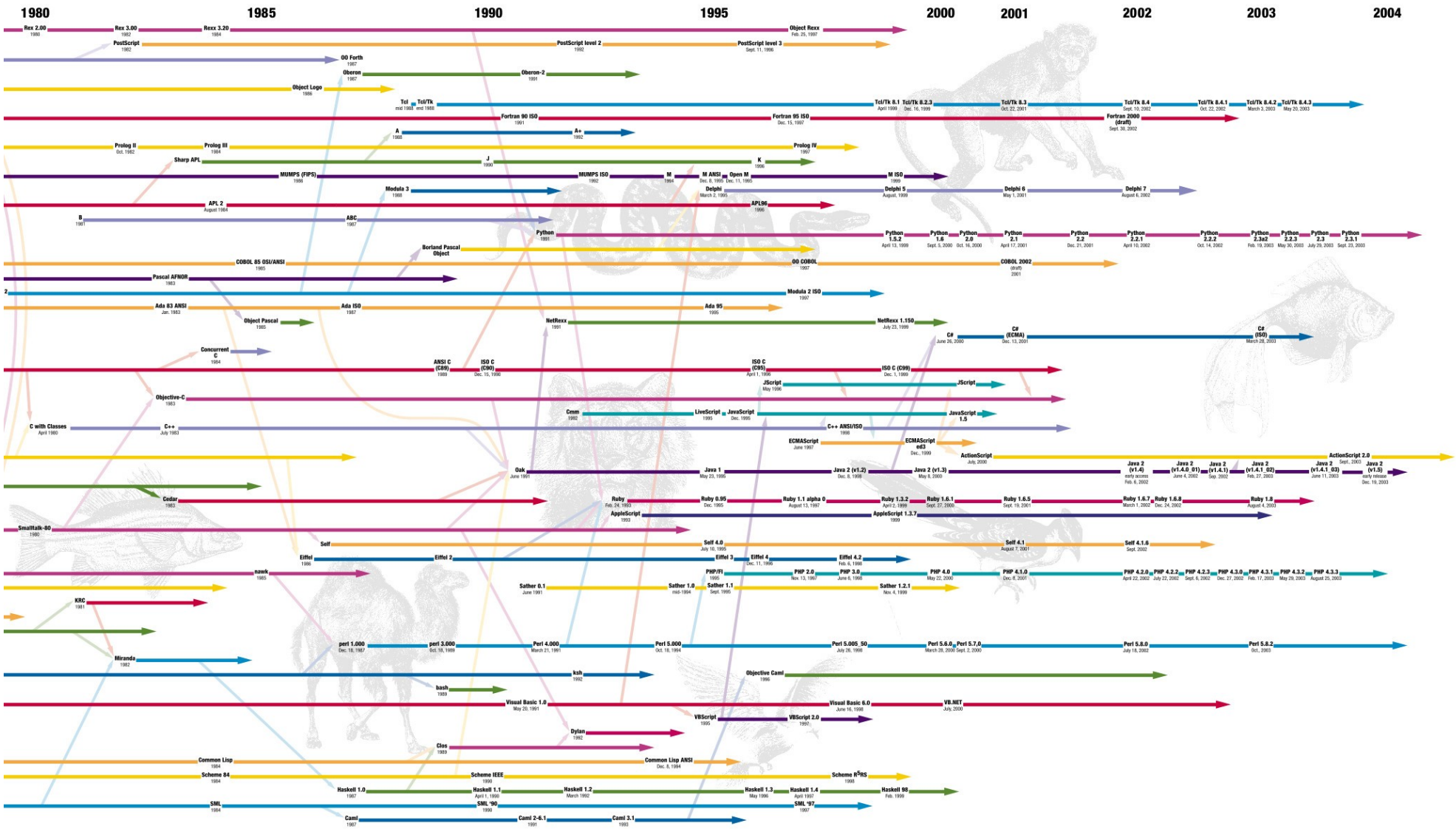
Orientado a objetos: Java, C++

História das linguagens de programação



http://cdn.oreillystatic.com/news/graphics/prog_lang_poster.pdf

História das linguagens de programação



Linguagens mais populares

Oct 2020	Oct 2019	Change	Programming Language	Ratings	Change
1	2	↑	C	16.95%	+0.77%
2	1	↓	Java	12.56%	-4.32%
3	3		Python	11.28%	+2.19%
4	4		C++	6.94%	+0.71%
5	5		C#	4.16%	+0.30%
6	6		Visual Basic	3.97%	+0.23%
7	7		JavaScript	2.14%	+0.06%
8	9	↑	PHP	2.09%	+0.18%
9	15	⇈	R	1.99%	+0.73%
10	8	↓	SQL	1.57%	-0.37%
11	19	⇈	Perl	1.43%	+0.40%
12	11	↓	Groovy	1.23%	-0.16%
13	13		Ruby	1.16%	-0.16%
14	17	↑	Go	1.16%	+0.06%
15	20	⇈	MATLAB	1.12%	+0.19%

http://statisticstimes.com/tech/top-computer-languages.php

Como programar?

Passos da programação

1. Compreender o problema

2. Conceber o algoritmo

Linguagem natural / gráfica

3. Implementar o algoritmo

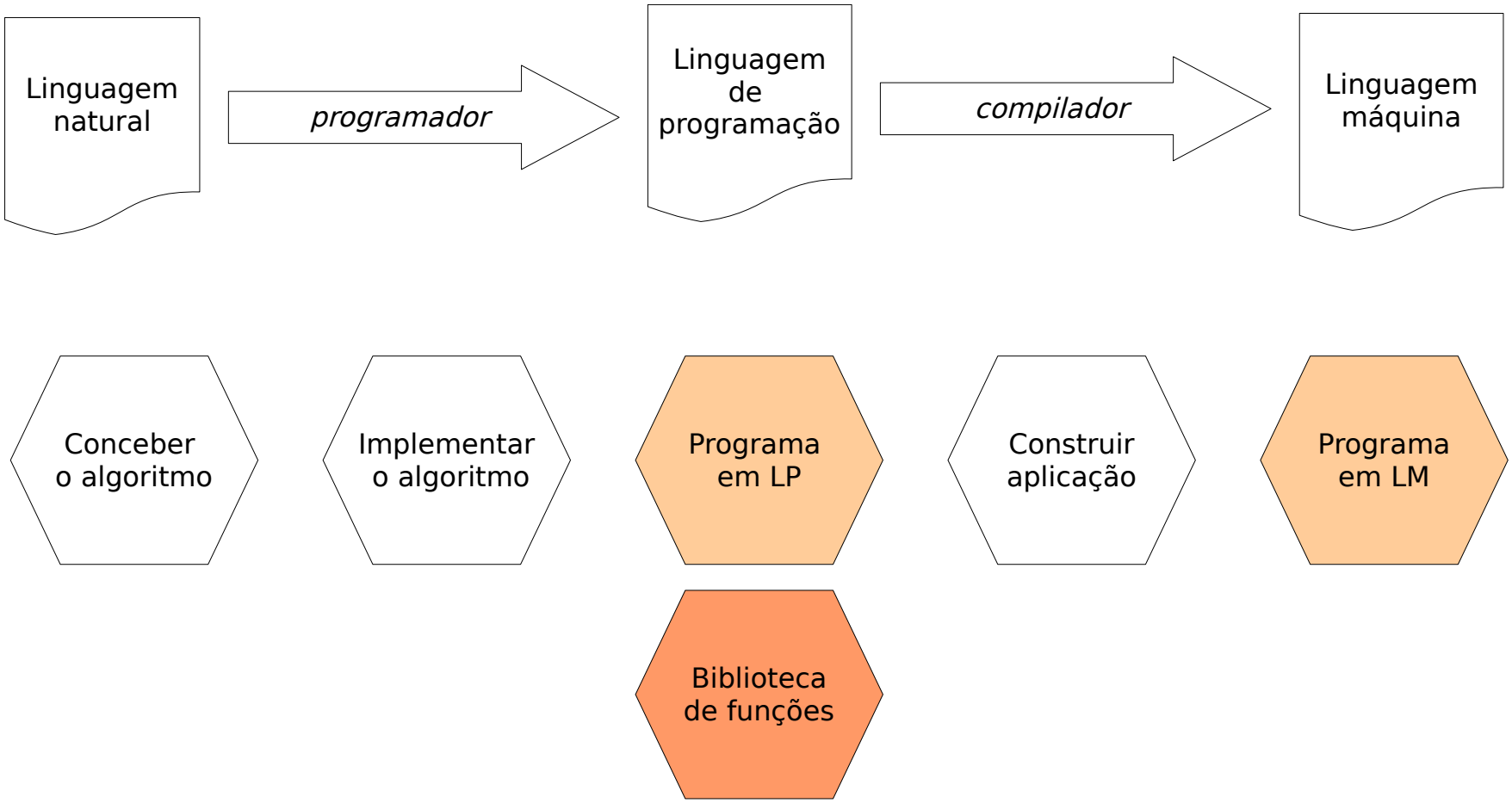
Linguagem de programação

4. Construir a aplicação

Linguagem máquina

5. Testar

Programar



Baseado num slide de P1, FEUP

Exemplo

Problema

Calcular a área duma sala

1. Compreender o problema

Que dados são necessários?

Como obter o resultado a partir dos dados

2. Conceber o algoritmo

1. perguntar o comprimento e guardar o valor em `comp`
2. perguntar a largura e guardar o valor em `larg`
3. calcular comp x larg e guardar o resultado em `area`
4. escrever o valor de `area`

Exemplo

3. Implementar o algoritmo

```
#include <stdio.h>

int main()
{
        int comp, larg;
        printf( "Qual o comprimento da sala? " );
        scanf( "%d", &comp );
        printf( "Qual a largura da sala? " );
        scanf( "%d", &larg );
        printf( "A área da sala é %d\n", comp*larg );

}
```

Exemplo

4. Construir a aplicação

```
gcc -o area area.c
```

5. Testar a aplicação

```
tcg@pitanga:~/UE-dinf/1819/P1$ ./area
Qual o comprimento da sala? 5
Qual a largura da sala? 7
A área da sala é 35
tcg@pitanga:~/UE-dinf/1819/P1$ ./area
Qual o comprimento da sala? 8
Qual a largura da sala? 8
A área da sala é 64
```

Como aprender?

Estudar, estudar, …

Praticar, praticar, …

Cometer erros, cometer erros, …

Aprender com os erros, …

Princípios a utilizar na programação

Programação estruturada

- Decompor um problema em sub-problemas
  - Reutilização
  - Teste independente
  - Facilidade de modificação

Legibilidade

- Programas devem ser escritos para serem lidos por humanos!
  - Comentários, estrutura, nomes das “coisas”, ...

Correção – simplicidade – eficiência

Erros de programação

Bug

Erro de programação

Durante a programação surgem muitos erros!!!

Debugging (depuração)

Processo de encontrar erros

Semelhante ao trabalho de um detetive

Suspeita-se de algo errado; altera-se o programa; faz-se testes para confirmar a resolução

Tipos de erros (bugs)

Sintático

O código fonte não respeita a sintaxe da linguagem

```
A = 1+2)
```

Semântico

Aparentemente executa bem mas não produz os resultados corretos!

Mais difícil de encontrar onde está o erro...

Runtime

Manifestam-se apenas durante a execução e sob circunstâncias especiais

Indicam que algo excecional (e normalmente mau) aconteceu!