Introdução

Programação I
2018.2019

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 específica 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

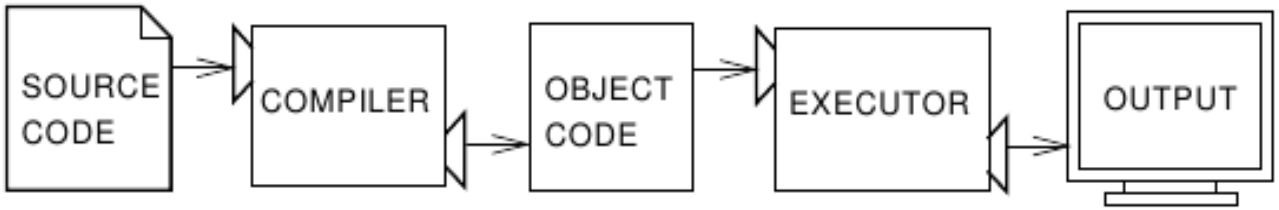
Interpretador vs compilador

Interpretador

Lê, interpreta e executa uma instrução de cada vez

Compilador

Traduz o programa para código máquina executável



Porquê tantas linguagens?

Diferentes nível de abstração

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

Baixo nível: possivelmente mais eficiente

Diferentes problemas

Cálculos numéricos: Fortran

Raciocínio: Prolog

Scripting: Perl, Python

Diferentes paradigmas

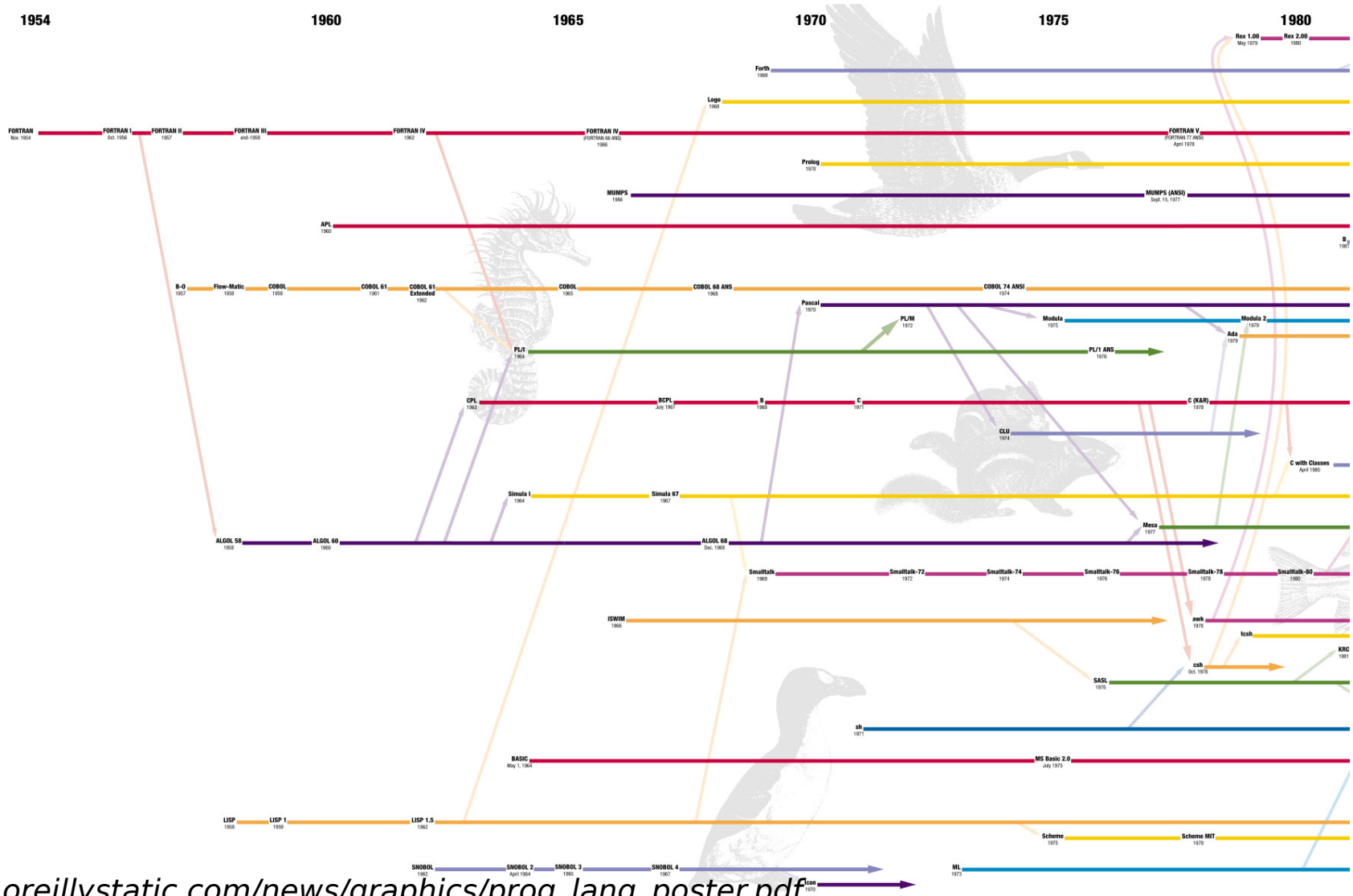
Imperativo: C, Pascal

Funcional: Haskell, Caml

Lógico: Prolog

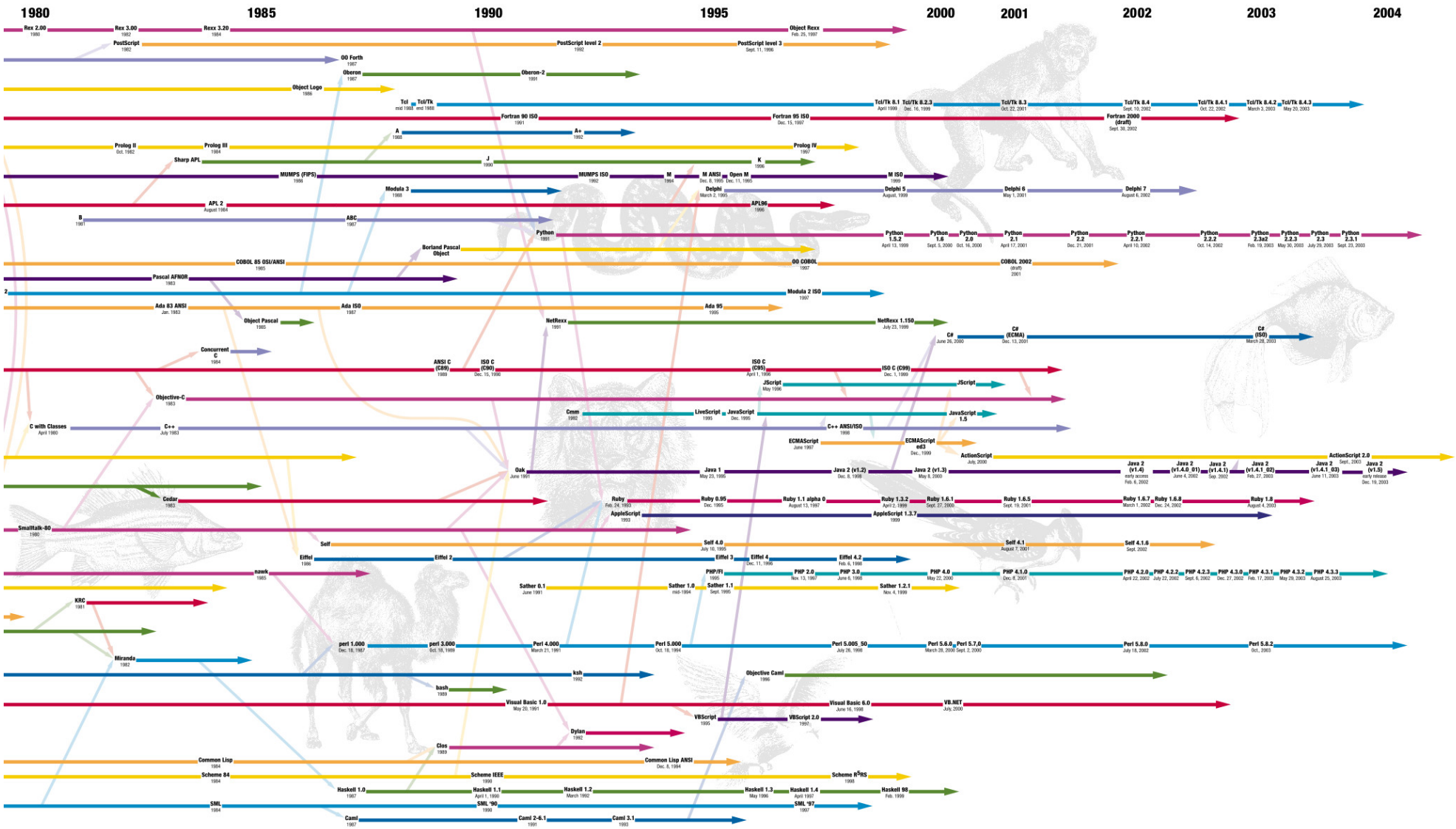
Orientado a objetos: Java, C++

História das linguagens de programação



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

História das linguagens de programação



Linguagens mais populares

TIOBE Index

Jul 2018	Jul 2017	Change	Programming language	Ratings	Change
1	1		Java	16.139 %	+2.37 %
2	2		C	14.662 %	+7.34 %
3	3		C++	7.615 %	+2.04 %
4	4		Python	6.361 %	+2.82 %
5	7	↑	Visual Basic .NET	4.247 %	+1.20 %
6	5	↓	C#	3.795 %	+0.28 %
7	6	↓	PHP	2.832 %	-0.26 %
8	8		JavaScript	2.831 %	+0.22 %
9	-	↑↑	SQL	2.334 %	+2.33 %
10	18	↑↑	Objective-C	1.453 %	-0.44 %
11	12	↑	Swift	1.412 %	-0.84 %
12	13	↑	Ruby	1.203 %	-1.05 %
13	14	↑	Assembly language	1.154 %	-1.09 %
14	15	↑	R	1.150 %	-0.95 %
15	17	↑	MATLAB	1.130 %	-0.88 %
16	9	↓↓	Delphi/Object Pascal	1.109 %	-1.38 %
17	11	↓↓	Perl	1.101 %	-1.23 %
18	10	↓↓	Go	0.969 %	-1.39 %
19	16	↓	Visual Basic	0.885 %	-1.21 %
20	20		PL/SQL	0.704 %	-0.84 %

PYPL Index (Worldwide)

Jul 2018	Change	Programming language	Share	Trends
1	↑	Python	23.59 %	+5.5 %
2	↓	Java	22.4 %	-0.5 %
3	↑↑	Javascript	8.49 %	+0.2 %
4	↓	PHP	7.93 %	-1.5 %
5	↓	C#	7.84 %	-0.5 %
6		C/C++	6.28 %	-0.8 %
7	↑	R	4.18 %	+0.0 %
8	↓	Objective-C	3.4 %	-1.0 %
9		Swift	2.65 %	-0.9 %
10		Matlab	2.25 %	-0.3 %
11		Ruby	1.59 %	-0.5 %
12	↑↑↑	TypeScript	1.58 %	+0.3 %
13	↓	VBA	1.42 %	-0.1 %
14	↓	Visual Basic	1.2 %	-0.2 %
15	↓	Scala	1.2 %	-0.1 %
16	↑↑↑	Kotlin	0.97 %	+0.5 %
17		Go	0.93 %	+0.3 %
18	↓↓	Perl	0.78 %	-0.1 %
19	↓	Lua	0.42 %	-0.1 %
20	↑↑	Rust	0.36 %	+0.0 %
21		Haskell	0.3 %	-0.1 %
22	↓↓	Delphi	0.25 %	-0.1 %

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

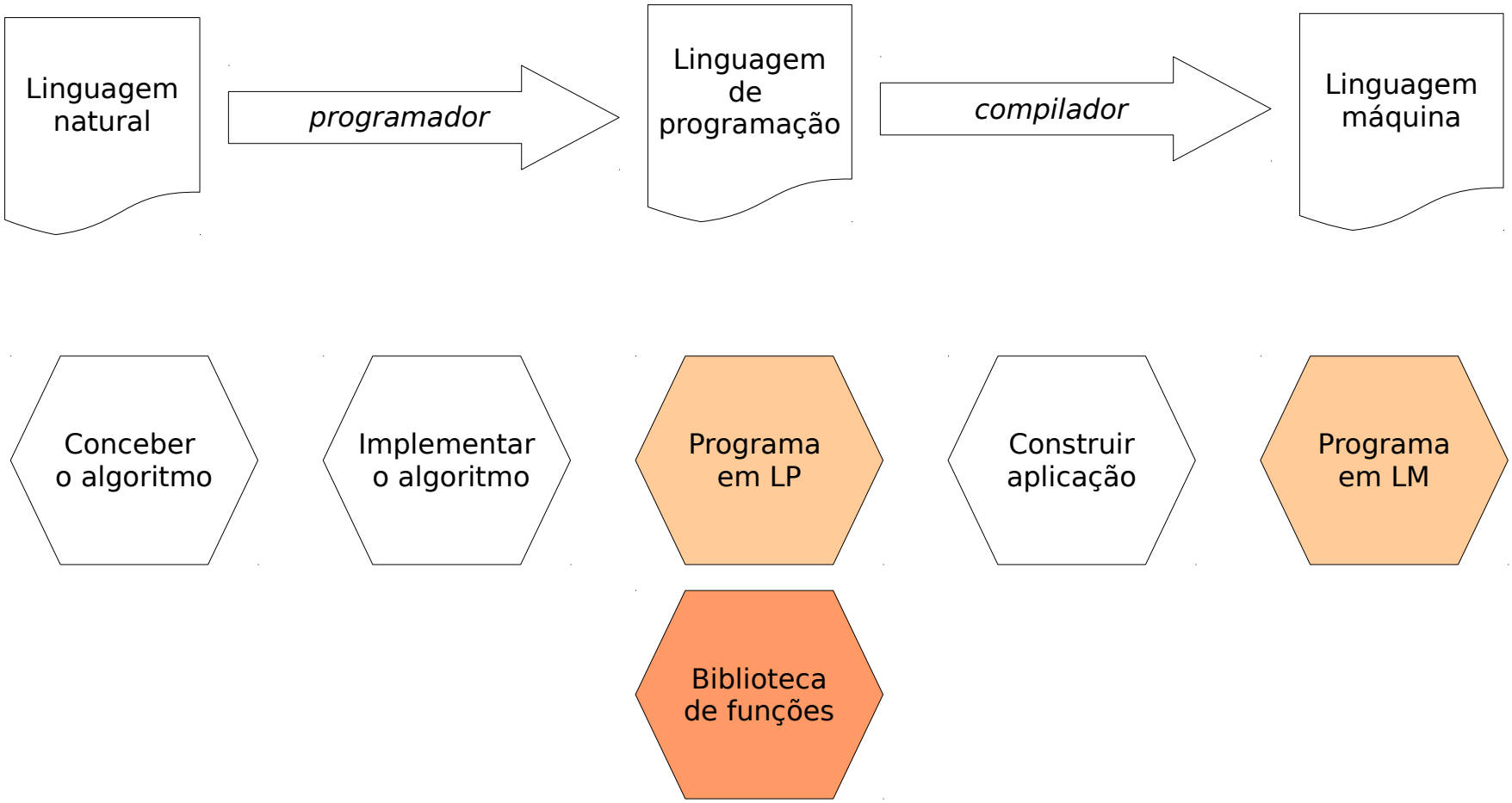
Linguagem de programação

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 obtenho 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 programa em sub-programas
  - 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 de algo errado; altera o programa; faz um teste 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!