Introdução

Programação I **2022/23**

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 / Abstrata

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 (convenção linguistica) concebida para exprimir computações

Linguagem formal

Sintaxe: *forma* (regras “gramaticais”)

Semântica: *significado*

Exemplos

Sintaxe	**Semântica**
3+4=7	soma de 3 com 4 é igual a 7
H2O	dois hidrogénios e um oxigénio

Linguagem natural vs Linguagem formal

Linguagem natural

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

Linguagem formal

- Não permite ambiguidade*
  - Significado preciso e claro

** 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 (por humanos)
  - Específica para a arquitetura do computador

Assembler

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

Raciocínio simbólico: Prolog, Haskell, (O)Caml

Scripting: Perl, Python

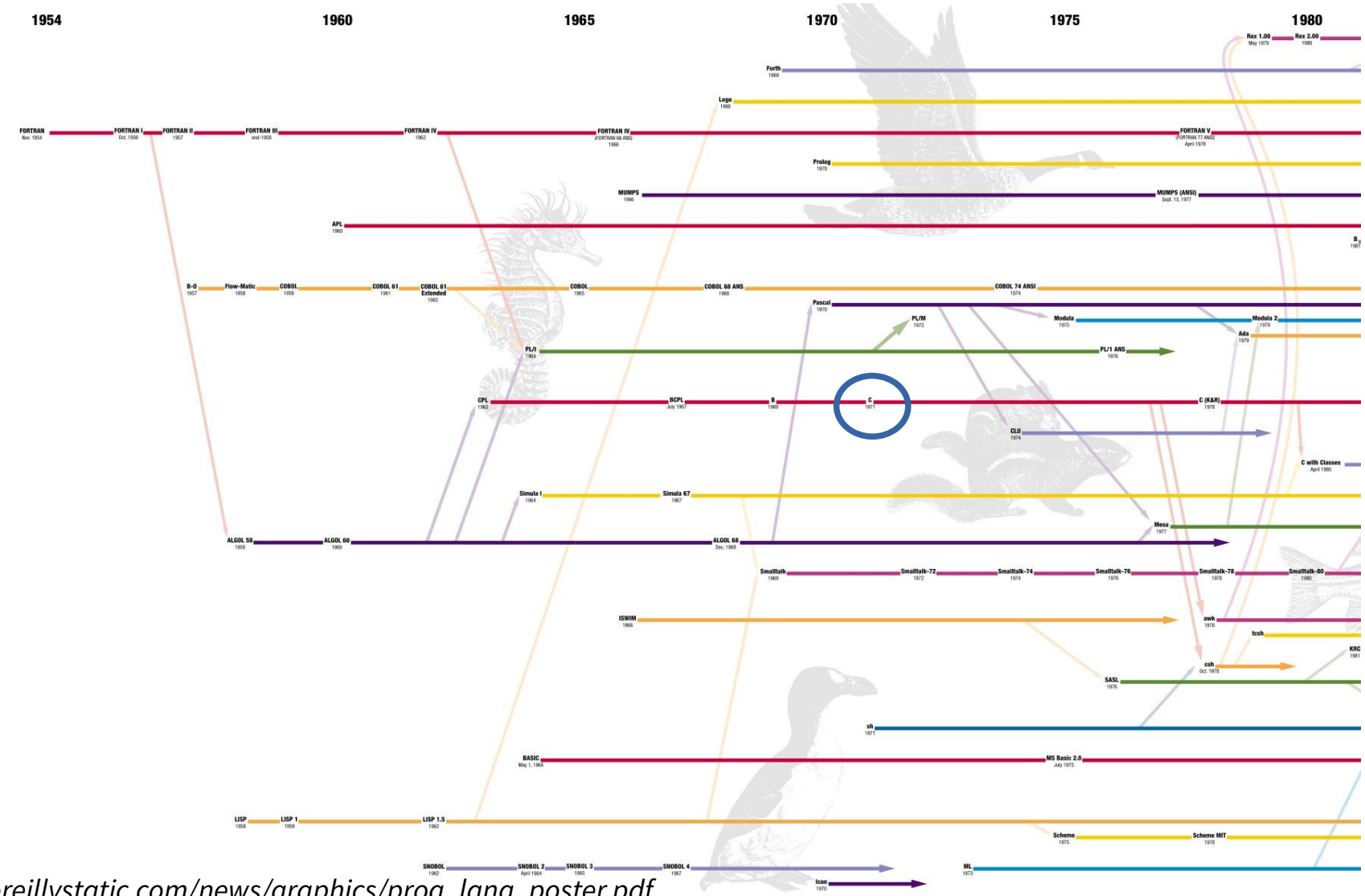
Diferentes paradigmas

Imperativo: C, Pascal, Java, C++

Funcional: Lisp, Scheme, Haskell, OCaml

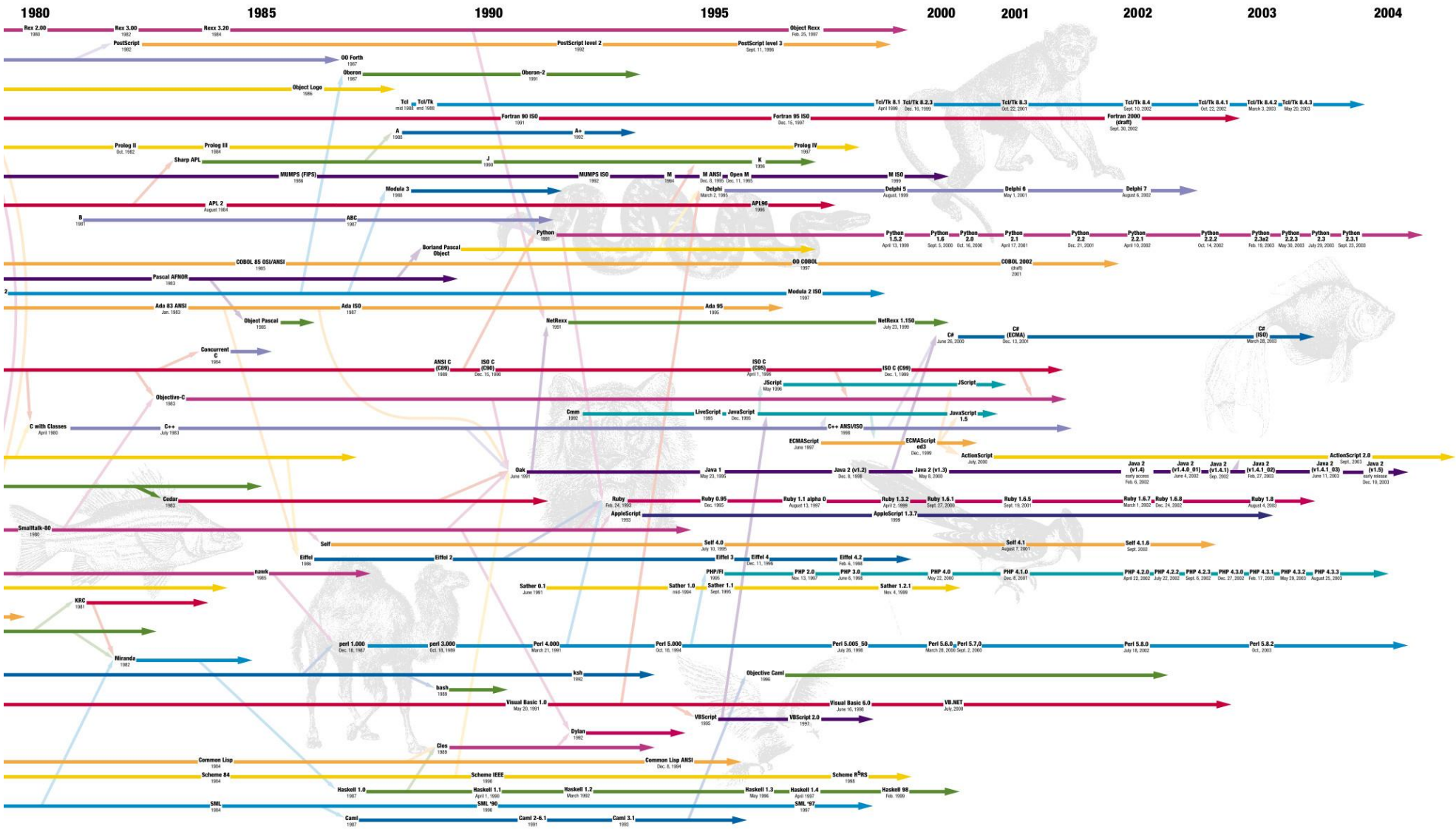
Lógico: Prolog, Picat

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

Aug 2021	Aug 2020	Change	Programming language	Ratings	Change
1	1		C	12.57%	-4.41%
2	3	↑	Python	11.86%	+2.17%
3	2	↓	Java	10.43%	-4.00%
4	4		C++	7.36%	+0.52%
5	5		C#	5.14%	+0.46%
6	6		Visual Basic	4.67%	+0.01%
7	7		JavaScript	2.95%	+0.07%
8	9	↑	PHP	2.19%	-0.05%
9	14	↑↑	Assembly language	2.03%	+0.99%
10	10		SQL	1.47%	+0.02%
11	18	↑↑	Groovy	1.36%	+0.59%
12	17	↑↑	Classic Visual Basic	1.23%	+0.41%
13	42	↑↑	Fortran	1.14%	+0.83%
14	8	↓↓	R	1.05%	-1.75%
15	15		Ruby	1.01%	-0.03%
16	12	↓↓	Swift	0.98%	-0.44%
17	16	↓	MATLAB	0.98%	+0.11%
18	11	↓↓	Go	0.90%	-0.52%
19	36	↑↑	Prolog	0.80%	+0.41%
20	13	↓↓	Perl	0.78%	-0.33%

PYPL Index (Worldwide)

Aug 2021	Change	Programming language	Share	Trends
1		Python	29.93 %	-2.2 %
2		Java	17.78 %	+1.2 %
3		JavaScript	8.79 %	+0.6 %
4		C#	6.73 %	+0.2 %
5	↑	C/C++	6.45 %	+0.7 %
6	↓	PHP	5.76 %	-0.0 %
7		R	3.92 %	-0.1 %
8		Objective-C	2.26 %	-0.3 %
9	↑	TypeScript	2.11 %	+0.2 %
10	↓	Swift	1.96 %	-0.3 %
11	↑	Kotlin	1.81 %	+0.3 %
12	↓	Matlab	1.48 %	-0.4 %
13		Go	1.29 %	-0.2 %
14	↑↑	Rust	1.21 %	+0.2 %
15	↓	VBA	1.16 %	-0.1 %
16	↓	Ruby	1.02 %	-0.1 %
17		Scala	0.79 %	-0.1 %
18	↑	Ada	0.77 %	+0.2 %
19	↓	Visual Basic	0.75 %	+0.0 %
20		Dart	0.68 %	+0.2 %

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

TIOBE ratings are calculated by counting hits of the most popular search engines
PYPL is created by analyzing how often language tutorials are searched on Google

Como programar?

Passos da programação

1. Compreender o problema

2. Conceber o algoritmo

Linguagem natural / gráfica

3. Implementar o algoritmo

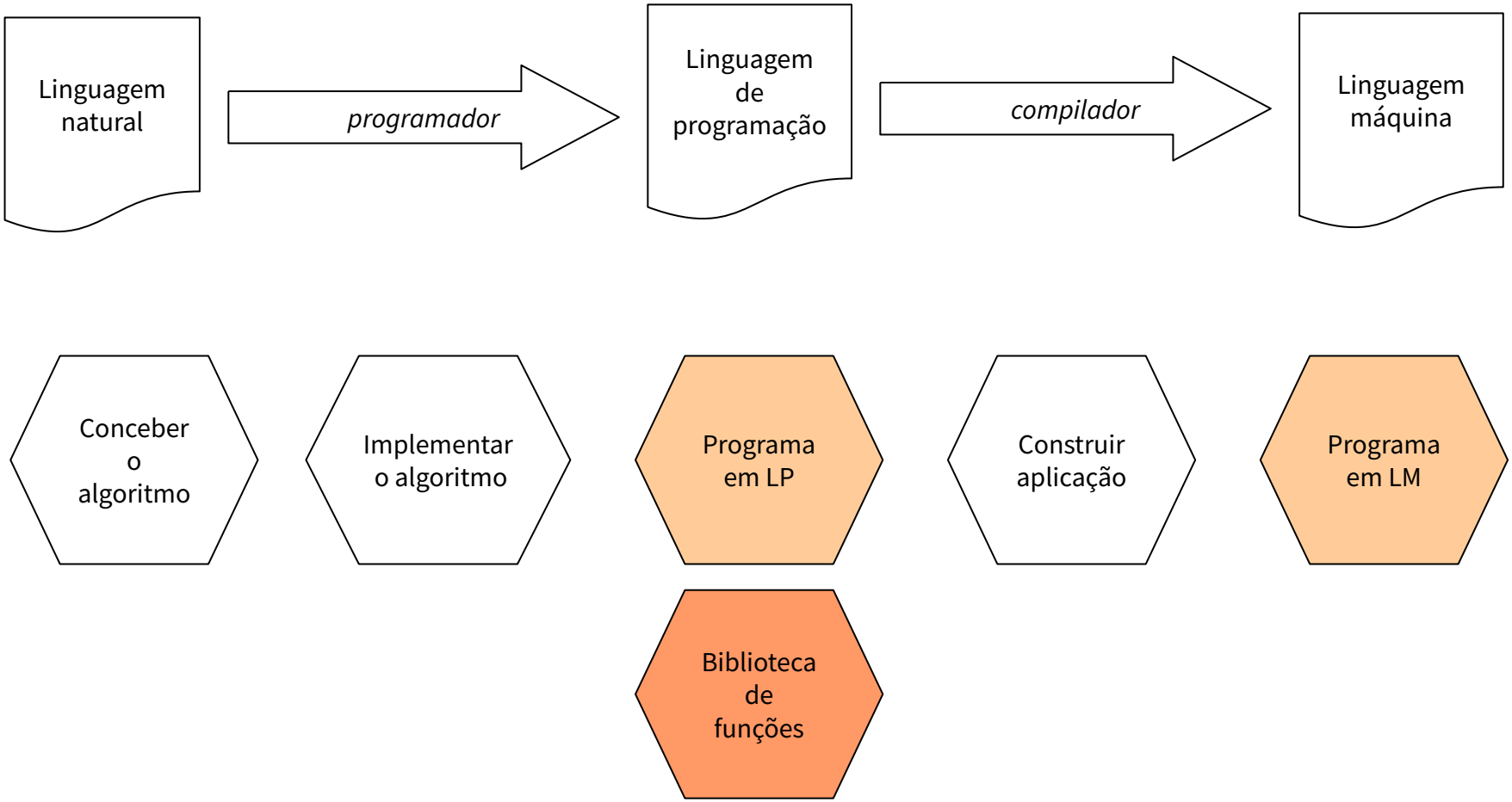
Linguagem de programação

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

Exemplo

2. Conceber o algoritmo

1. aceder ao texto do comprimento, interpretar como um número e guardar o valor em **comp**
2. aceder ao texto da largura, interpretar como um número e guardar o valor em **larg**
3. calcular comp x larg e guardar o resultado em **area**
4. escrever o texto correspondente ao valor de **area**

Exemplo

3. Implementar o algoritmo

```
01:05:03$ cat 02-area.c
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
    int comp = atoi (argv[1]);
    int larg = atoi (argv[2]);
    printf ("A area da sala e %d\n", comp * larg);
}
01:05:05$
```

Exemplo

4. Construir a aplicação

i.e. compilar o código fonte para produzir o código máquina

```
01:05:13$ gcc -o 02-area 02-area.c
```

Exemplo

5. Testar a aplicação

```
01:05:16$ ./02-area 12 8
A area da sala e 96
01:05:28$
01:08:55$ ./02-area 5 7
A area da sala e 35
01:08:59$
01:09:00$ ./02-area 8 8
A area da sala e 64
01:09:04$
```

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 as regras gramaticais da linguagem

```
a = 1)+2[
```

Semântico

Aparentemente executa bem mas não produz os resultados corretos!

Mais difícil de encontrar onde está o erro...

Runtime (exceção)

Manifestam-se apenas durante a execução e sob circunstâncias especiais

Indicam que algo excecional (e normalmente mau) aconteceu!