

Departamento de Engenharia de Sistemas e Computação

Pensamento Algorítmico

Algoritmos, refinamentos sucessivos, fluxogramas e pseudocódigo

---

João Araujo

Algoritmos Computacionais I

Aula 04

Roteiro

- Linguagem de programação e algoritmos (Cap. 2.1–2.2)
- Um algoritmo computacional simples: a busca no dicionário (2.3)
- Refinamentos sucessivos
- Fluxogramas (2.4)
- Pseudocódigo (2.5)
- **Prática:** algoritmos no papel

Algoritmo: passos não ambíguos

O computador executa *exatamente* o que você escreve

Ele não "entende a intenção". Cada passo precisa ser preciso e executável.

Ambíguo

- "Misture bem"
- "Espere um pouco"
- "Ache a palavra"

Preciso

- "Misture por 30 segundos"
- "Espere 5 minutos"
- "Abra o livro na página do meio e compare"

Um algoritmo computacional: achar uma palavra no dicionário

Primeira tentativa (força bruta):

1. Abra na primeira página
2. Se a palavra estiver ali, pare
3. Senão, vire a página e volte ao passo 2

Funciona, mas é lento: um dicionário com 1000 páginas pode exigir 1000 passos.

Pergunta

Como uma pessoa *de verdade* procura no dicionário? Isso explora o fato de ele estar **ordenado**.

Refinamento: a busca binária

1. Abra o dicionário no meio
2. Se a palavra está na página, pare
3. Se a palavra vem *antes* da página atual, descarte a metade de depois
4. Se vem *depois*, descarte a metade de antes
5. Repita com a metade que sobrou
6. Se não sobrar nenhuma página, a palavra não existe

Eficiência

Cada passo divide o problema por 2: com 1000 páginas bastam $\lceil \log_2 1000 \rceil = 10$ passos.

Refinamentos sucessivos

Estratégia para problemas grandes: começar com passos de **alto nível** e detalhar cada um.

Nível 1

1. Ler as notas
2. Calcular a média
3. Informar a situação

Nível 2 (refinando o passo 3)

3.1 Se média ≥ 7: "aprovado"

3.2 Senão, se média ≥ 4: "prova final"

3.3 Senão: "reprovado"

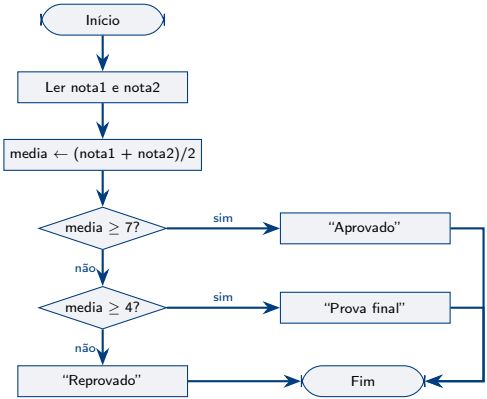
Só quando todos os passos são simples o suficiente é que escrevemos o código.

Fluxogramas: símbolos básicos



- Setas indicam o **fluxo** de execução
- Decisão tem duas saídas: *sim* e *não*
- Um fluxograma bom cabe em uma página e se lê de cima para baixo

Exemplo: situação do aluno



Pseudocódigo

Descrição **textual e estruturada** do algoritmo, sem a sintaxe rígida de uma linguagem.

Pseudocódigo

```
LEIA nota1, nota2
media ← (nota1 + nota2) / 2
SE media ≥ 7 ENTÃO
  ESCREVA “Aprovado”
SENÃO SE media ≥ 4 ENTÃO
  ESCREVA “Prova final”
SENÃO
  ESCREVA “Reprovado”
```

- Foco na lógica, não na sintaxe
- Cada comando é uma ação clara
- Indentação mostra a estrutura
- Depois é traduzido para Python quase linha a linha

Do pseudocódigo ao Python

```
nota1 = float(input("Primeira nota: "))
nota2 = float(input("Segunda nota: "))
media = (nota1 + nota2) / 2
if media >= 7:
    print("Aprovado")
elif media >= 4:
    print("Prova final")
else:
    print("Reprovado")
```

- `input` lê texto; `float(...)` converte para número
- `elif` = "senão se"

Fonte: `code/Cap3/prog.3.3_calcula_final_reprova.py`

Prática: uma máquina só de somar e subtrair

Imagine uma máquina que só sabe **somar** e **subtrair**, e comparar com zero.

Desafio (Exercícios 2.4 a 2.6)

Escreva, em pseudocódigo, um algoritmo para

1. multiplicar $a \times b$
2. calcular a^b
3. dividir $a \div b$ (quociente e resto)

Dica para a multiplicação: $a \times b = a + a + \cdots + a$ (b vezes). Use um contador e um acumulador.

Depois desenhe os fluxogramas (Exercício 2.7).

Solução esperada: multiplicação por somas

```
a = int(input("a = "))
b = int(input("b = "))
produto = 0
contador = 0
while contador < b:
    produto = produto + a
    contador = contador + 1
print(a, "x", b, "=", produto)
```

- Já usamos uma repetição (`while`); ela será estudada na Aula 8
- Teste na mão: $a = 3$, $b = 4$ gera 0, 3, 6, 9, 12

Para praticar: exercícios do livro

Site do livro – Capítulo 2

- Algoritmos do dia a dia: **2.1, 2.2**
- Busca binária e máquina de somar: **2.3 a 2.6**
- Fluxogramas: **2.7**
- Busca binária mais enxuta: **2.19**

Enunciados e soluções comentadas em
`https://www.araujo.eng.uerj.br/livro/exercicios/cap2/`

Exercícios adicionais (sem solução): `https://www.araujo.eng.uerj.br/livro/extras/`

Resumo da aula

- Algoritmo: passos finitos, precisos e sem ambiguidade
- Refinamentos sucessivos: de passos gerais a passos simples
- Fluxograma (visual) e pseudocódigo (textual) servem para pensar *antes* de programar
- Próxima aula: identificadores, dados e tipos em Python