



## Introdução ao Processamento de Dados Turma 3 (2020.1)



# Introdução a Algoritmos

**Gilson. A. O. P. Costa (IME/UERJ)**

[gilson.costa@ime.uerj.br](mailto:gilson.costa@ime.uerj.br)

# Definição de Algoritmo

Uma **sequência** de passos para realizar uma tarefa.

■ Exemplo: separar a roupa suja para lavar.

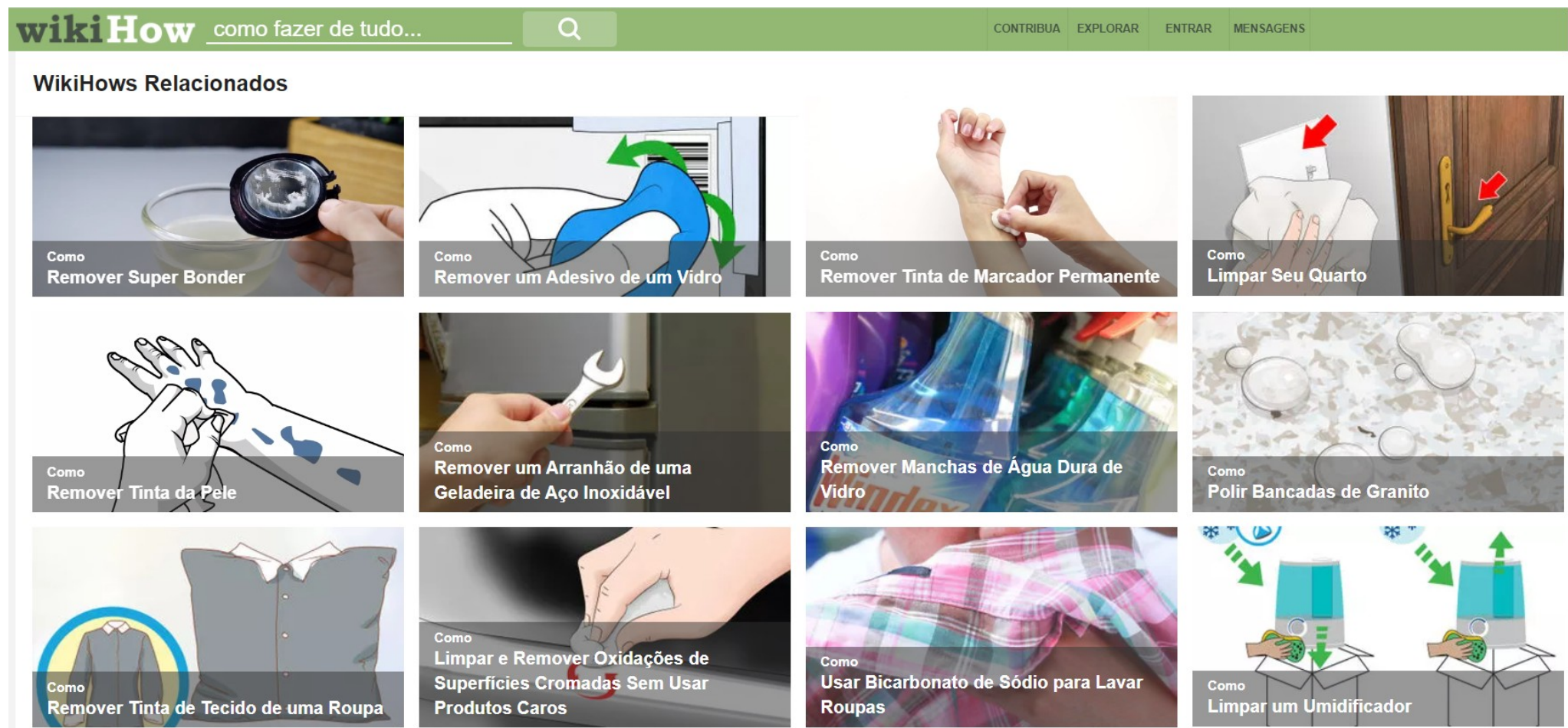
1. Pegue uma peça do cesto de roupa suja
2. Se ela for branca, coloque na cesta azul
3. Se ela tiver cor clara, coloque na cesta verde
4. Se ela for escura, coloque na cesta cinza
5. Repita a partir do passo 1 até esvaziar o cesto de roupa suja



wikiHow

# Definição de Algoritmo

Uma **sequência** de passos para realizar uma tarefa.





# Definição de Algoritmo

Uma **sequência** de passos para realizar uma tarefa.

■ Outro exemplo: cozinhar um bolo.

1. Fazer a massa
2. Untar a forma
3. Colocar a massa na forma
4. Esquentar o forno
5. Colocar a forma com a massa no forno
6. Esperar 20 minutos
7. Esperar mais 5 minutos
8. Abrir o forno e enfiar um palitinho
9. Se o palitinho sair molhado, repita a partir de (7)
10. Tirar do forno e esperar 5 minutos
11. Desenformar o bolo

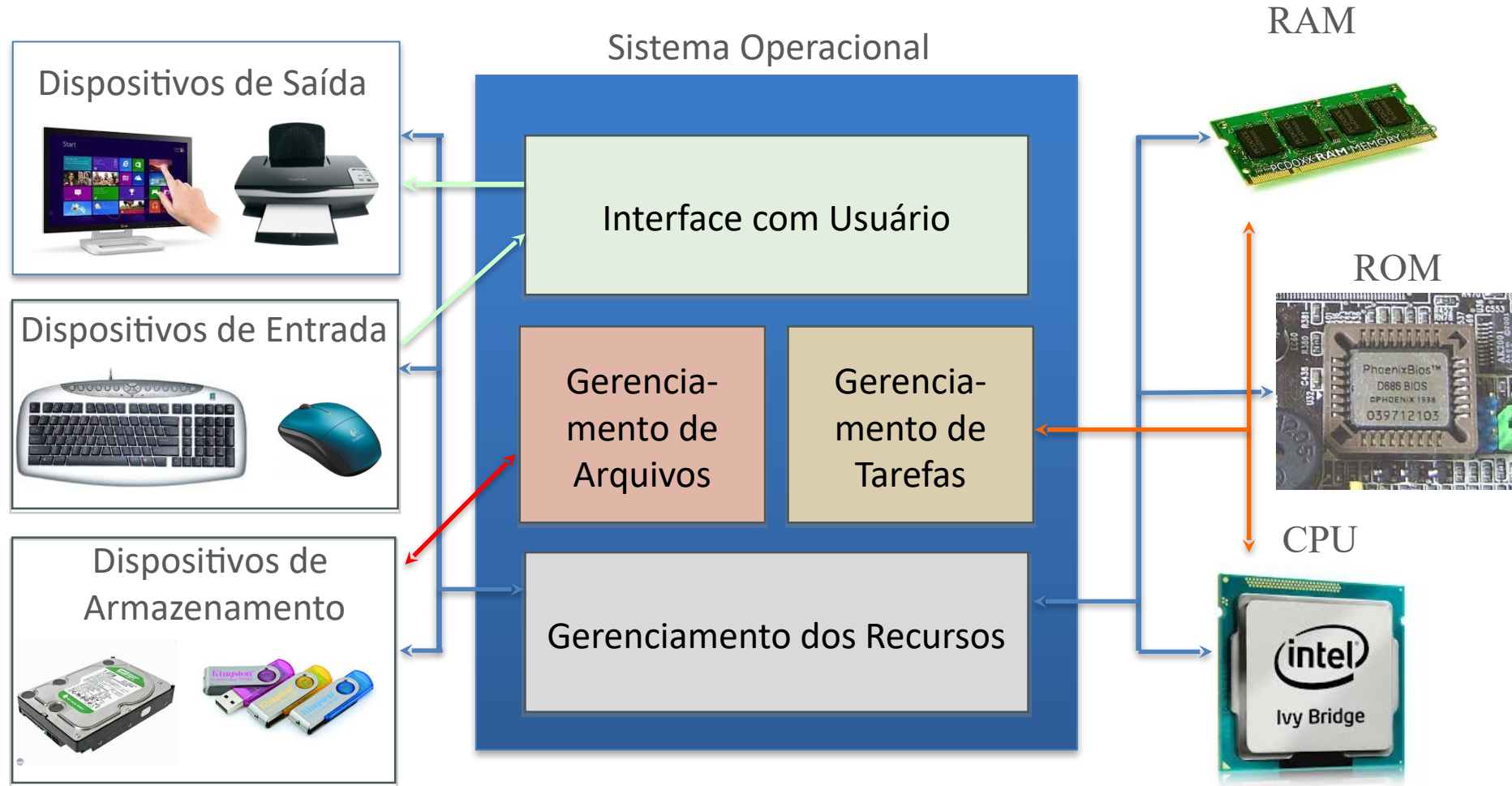


# Definição de Algoritmo

Uma **sequência** de passos para realizar uma tarefa.

- Em matemática e ciência da computação, um algoritmo é uma **sequência finita** de instruções, **bem definidas** e **implementáveis por um computador**.
- Para ser implementável por um programa de computador, um algoritmo deve ser escrito em uma **linguagem de programação** (programa).
- Os algoritmos são inequívocos: sempre que a sequência de passos é realizada sobre as mesmas **entradas**, as mesmas **saídas** devem ser produzidas.

# Sistema Operacional



# Entradas e Saídas

- Os **dados de entrada** de um programa são **lidos** de um dispositivo de entrada (teclado, mouse, touch pad, microfone, etc.) ou de um arquivo.
- Os **dados de saída** de um programa são **escritos** em um dispositivo de saída (monitor, impressora, auto falante, etc.) ou em um arquivo.
- Durante a execução da sequência de passos (instruções) o programa manipula (lê e escreve) **variáveis** na memória do computador.

# Variáveis

- Variáveis correspondem a **posições na memória (RAM)** do computador onde se armazenam dados manipulados pelo programa.

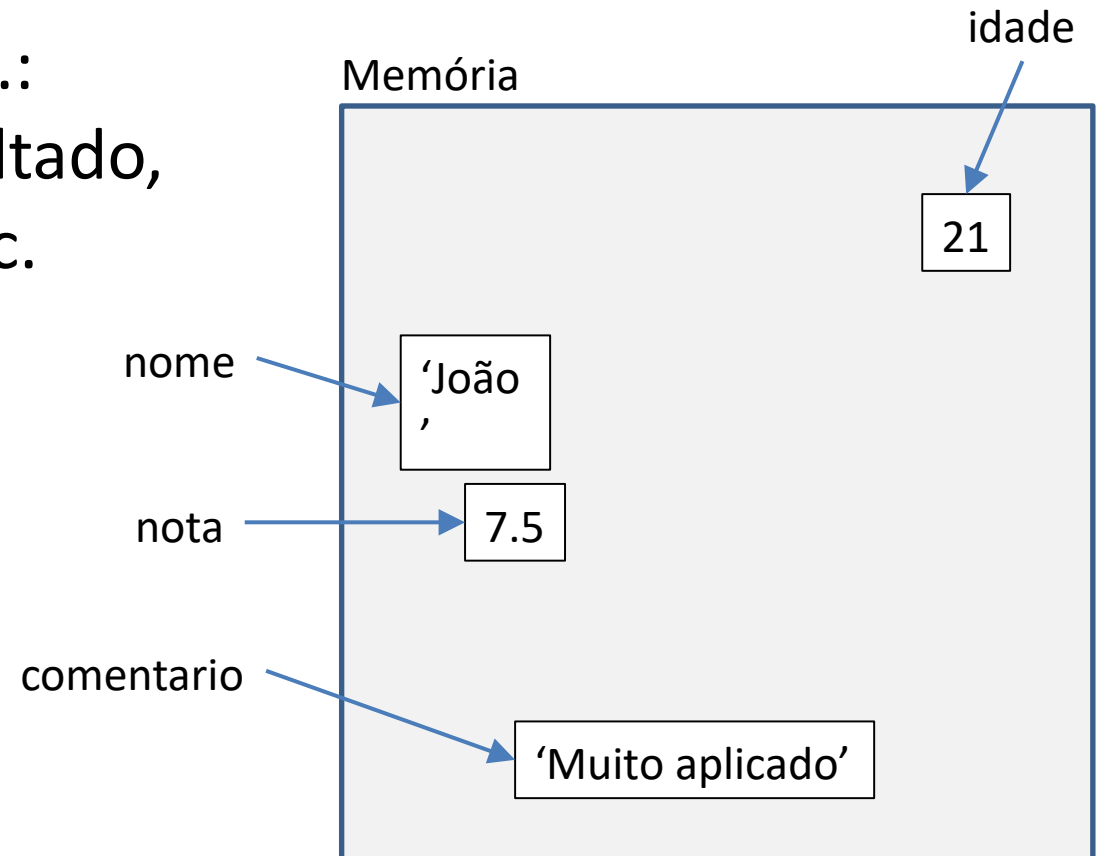
Memória

0	1	0	0	0	1	0	1	0	1	0	0	0	0	1	1	1	1	0	0
1	1	1	0	1	1	1	0	0	1	0	1	1	0	1	0	1	1	0	1
0	1	0	0	1	0	0	1	1	0	1	1	0	1	1	0	0	1	0	1
1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	1	1	1	0
0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	1	0	0	0	1	0	1	0	1	0	0	0	0	1	1	1	1	0	0
1	1	1	0	1	1	1	0	0	1	0	1	1	0	1	0	1	1	0	1
0	1	0	0	1	0	0	1	1	0	1	1	0	1	1	0	0	1	0	1
1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	1	1	1	0
0	1	1	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	1	1
0	1	0	0	0	1	0	1	0	1	0	0	0	0	1	1	1	1	0	0
1	1	1	0	1	1	1	0	0	1	0	1	1	0	1	0	1	1	0	1
0	1	0	0	1	0	0	1	1	0	1	1	0	1	1	0	0	1	0	1
1	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0	1	1	1	0
0	1	1	1	0	1	0	0	1	0	1	1	0	1	0	1	0	0	1	1
0	1	0	0	0	1	0	1	0	1	0	0	0	0	1	1	1	1	0	0
1	1	1	0	1	1	1	0	0	1	0	1	1	0	1	0	1	1	0	1
0	1	0	0	1	0	0	1	1	0	1	1	0	1	1	0	0	1	0	1
1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	1	1	1	0
0	1	1	1	0	1	0	0	1	0	1	1	0	1	0	1	0	0	1	1



# Variáveis

- Variáveis correspondem a **posições na memória** (RAM) do computador onde se armazenam dados manipulados pelo programa.
- São **identificadas por um nome**, e.g.:  
X, y, nome, idade, P1, P2, nota, resultado, zwy, Turma, banana, comentario, etc.
- Durante a execução, um programa geralmente altera os valores das variáveis.



# Variáveis

Existem **regras para se dar nome** a uma variável:

- Uma única letra ou inicia com uma letra que pode ser seguida de dígitos ou letras, em qualquer quantidade.
- Não deve possuir caracteres que tenham funções específicas, e.g., + - \* / = % “ ‘ ! ~ ? : ; , ( ) .
- Não deve possuir espaços.
- Sublinhado \_ funciona como uma letra (ok).
- Geralmente só se aceitam letras presentes na língua inglesa.
- Não pode ser um nome “reservado” numa linguagem, e.g., float, string, int, if, def, True, False, input, print, ...

# Variáveis

Quais destes são nomes de variáveis válidos?

A ✓	5B ✗	A32B ✓	x-y ✗
A:B ✗	KM/H ✗	Caixa_Preta ✓	b*d ✗
E(2) ✗	_NUM ✓	Caixa Preta ✗	A1 ✓
Endereço ✗	A1111 ✓	média ✗	ação ✗

# Tipos de Dados

Cada variável está associada a um **tipo de dado**.

Tipos básicos de dados:

- Numéricos
  - Inteiros (*integer*)
  - Reais (*float*)
- Alfanuméricos
  - Caractere (*char*): um único caractere
  - Cadeia de caracteres (*string*): uma cadeia de caracteres
- Lógicos ou Booleanos
  - Verdadeiro (*True*) ou Falso (*False*)

# Programação Estruturada

Possui três tipos de comandos/instruções básicos:

- Sequência simples
  - Atribuição
  - Entrada (*ler*)
  - Saída (*escrever*)
- Decisão
- Repetição



# Atribuição

Especifica o valor que será dado a uma variável.

- Sintaxe:  $\leftarrow$

Por enquanto vamos usar este símbolo, no Python equivale ao símbolo =

- Exemplos:

- $A \leftarrow 3.2$

Significa: variável 'A' recebe o valor 3.2

- $X \leftarrow 4 + 5$

Significa: variável 'X' recebe a soma de 4 + 5

- $\text{nome} \leftarrow \text{'João'}$

Significa: variável 'nome' recebe a *string* 'João'

# Atribuição

Especifica o valor que será dado a uma variável.

- Sintaxe:  $\leftarrow$

Por enquanto vamos usar este símbolo, no Python equivale ao símbolo =

- Exemplos:

- $X \leftarrow 3 * Y$

- Significa: variável 'X' recebe o valor 3 vezes o valor da variável 'Y'

- $b \leftarrow b + 1$

- Significa: 'b' recebe o conteúdo do próprio 'b', mais 1

- Como assim?

# Atribuição

- Em várias linguagens de programação (e.g., C, C++, Pascal, Fortran), é preciso 'declarar' uma variável antes dela ser usada no programa.
- No Python a primeira atribuição a uma variável vai criá-la: **alocar espaço** na memória.
- Imagine que nosso programa tem a seguinte sequência de atribuições:

1)  $a \leftarrow 1$

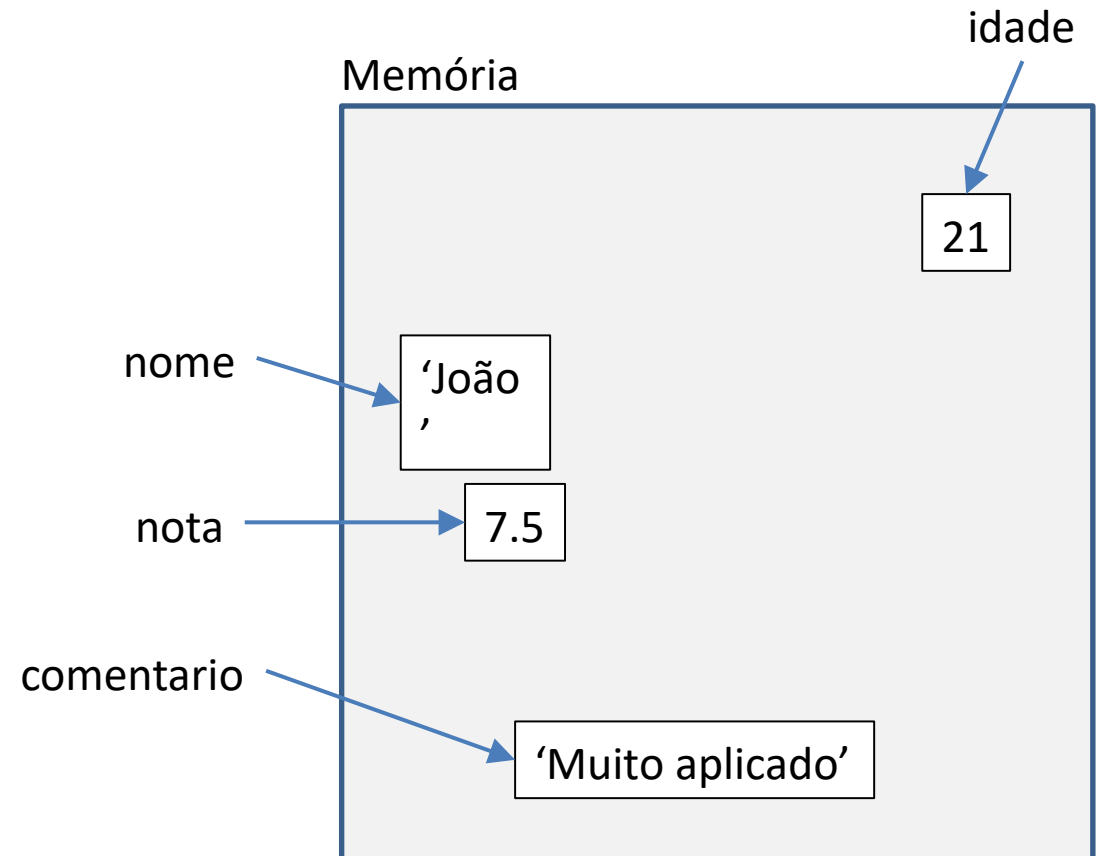
2)  $b \leftarrow a + 1$

3)  $b \leftarrow b + 1$

# Atribuição

Imagine que nosso programa tem a seguinte sequência de atribuições:

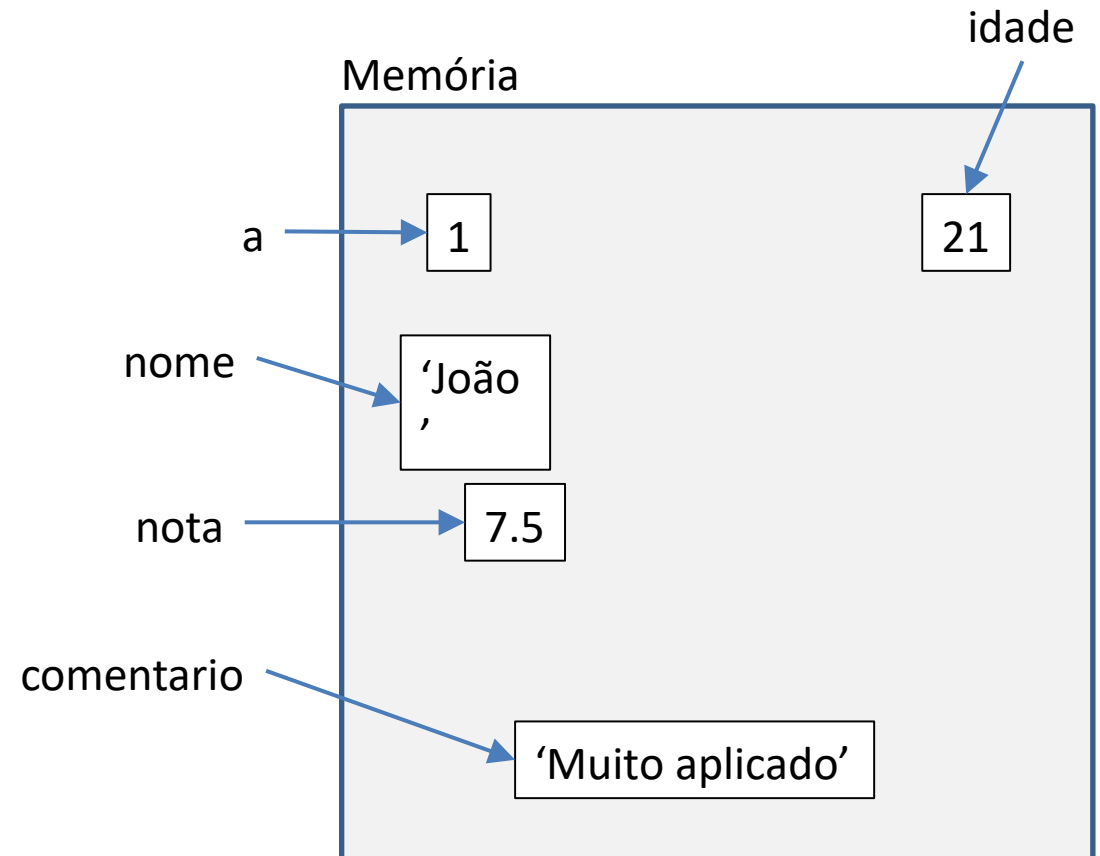
- 1)  $a \leftarrow 1$
- 2)  $b \leftarrow a + 1$
- 3)  $b \leftarrow b + 1$



# Atribuição

Imagine que nosso programa tem a seguinte sequência de atribuições:

- 1)  $a \leftarrow 1$
- 2)  $b \leftarrow a + 1$
- 3)  $b \leftarrow b + 1$

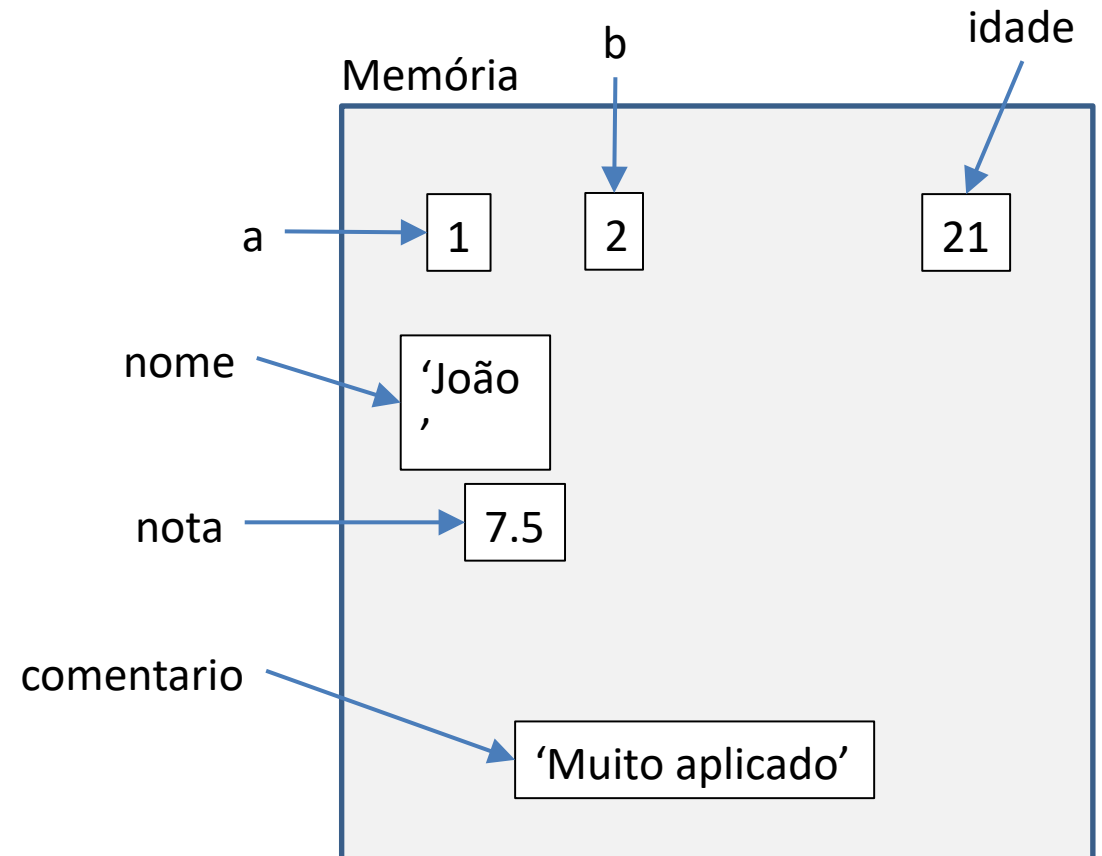




# Atribuição

Imagine que nosso programa tem a seguinte sequência de atribuições:

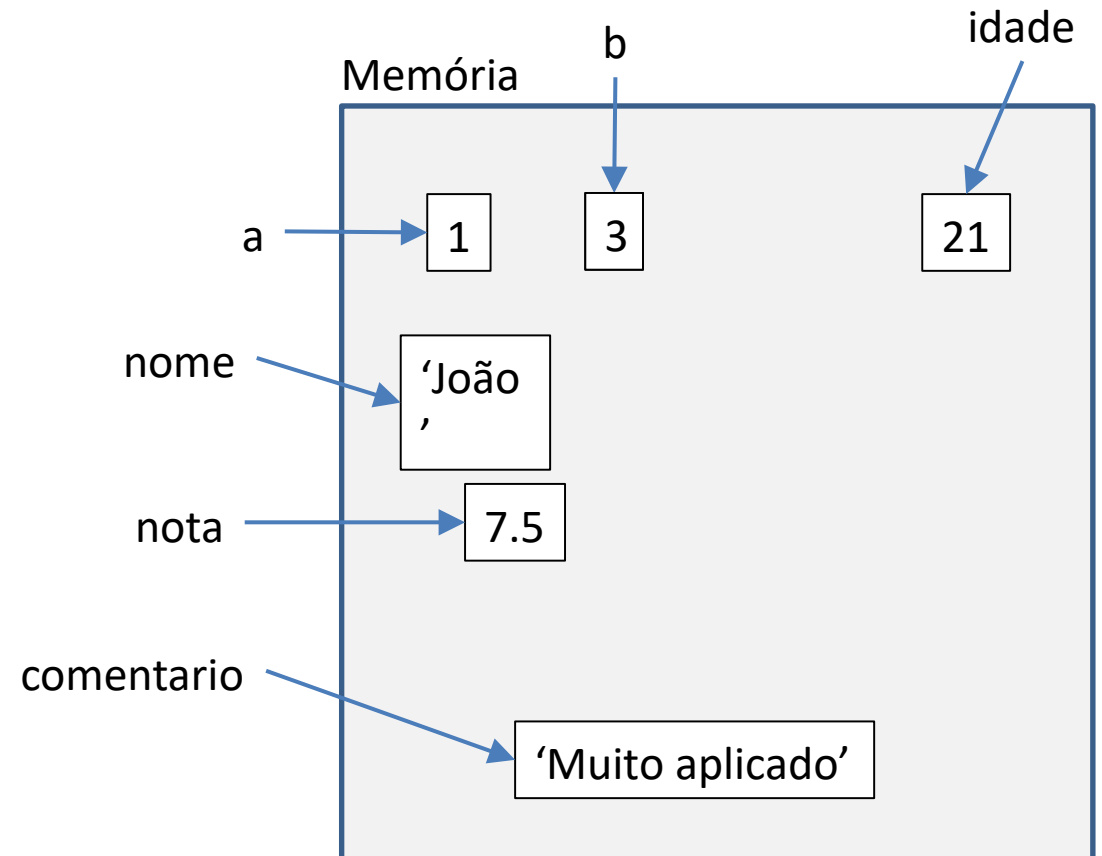
- 1)  $a \leftarrow 1$
- 2)  $b \leftarrow a + 1$
- 3)  $b \leftarrow b + 1$



# Atribuição

Imagine que nosso programa tem a seguinte sequência de atribuições:

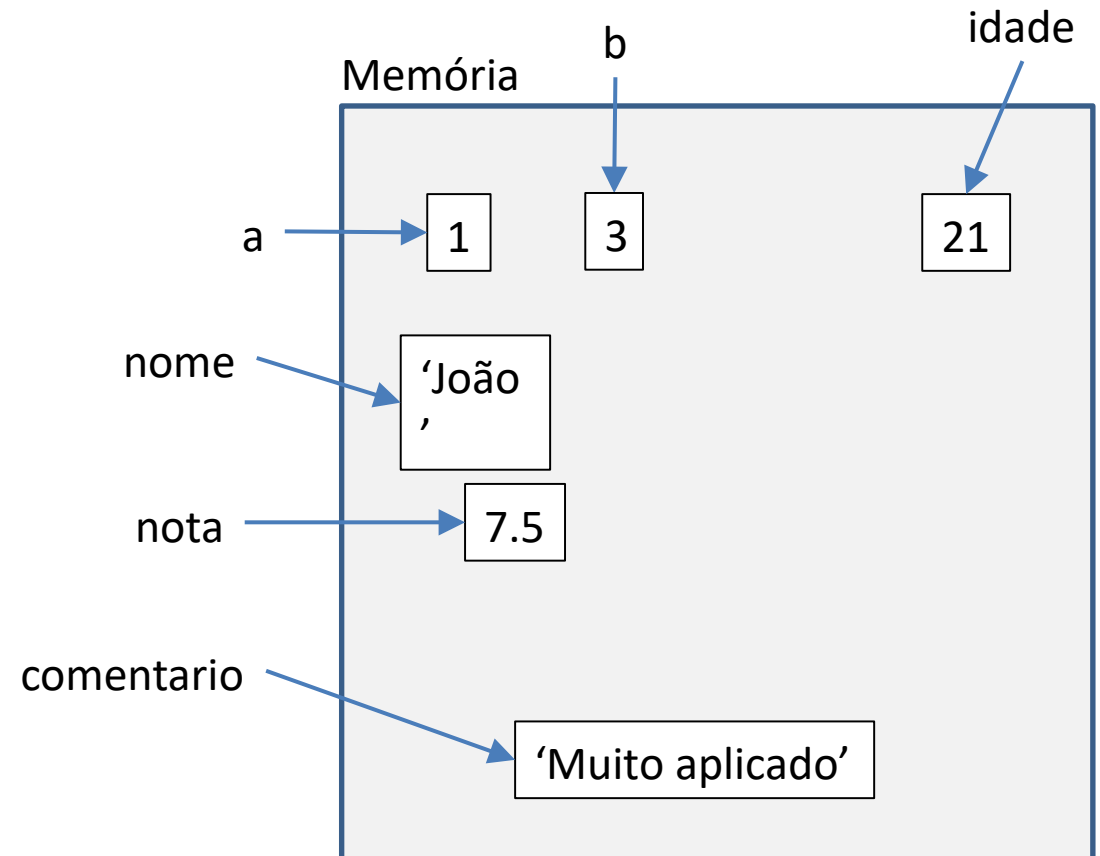
- 1)  $a \leftarrow 1$
- 2)  $b \leftarrow a + 1$
- 3)  $b \leftarrow b + 1$



# Atribuição

Perceba que uma atribuição substitui definitivamente (destrói) o valor anterior da variável!

- 1)  $a \leftarrow 1$
- 2)  $b \leftarrow a + 1$
- 3)  $b \leftarrow b + 1$



# Operadores Aritméticos

+ soma:  $5+4==9$

- subtração:  $5-4==1$

\* multiplicação:  $5*4==20$

\*\* potenciação:  $5**4==625$

/ divisão real:  $5/4==1.25$

// divisão inteira:  $5//4==1$

% resto inteiro (módulo) da divisão:  $5\%4==1$

( ) aninhamento/precedência:

$$5*(4-1) == 15$$

$$(5*4)-1 == 19$$

$$5*4-1 == 19$$

# Operadores Aritméticos

Exemplos:

$A \leftarrow 4$

$C \leftarrow A + 5$

$\text{mult} \leftarrow A * 2$

$N1 \leftarrow 9 / 2$

$N2 \leftarrow 9 // 2$

$\text{modulo} \leftarrow 9 \% 2$

$\text{divisao\_inteira} \leftarrow (9 + 4) // 2$



# Operadores Relacionais

Resultado da operação é um **valor lógico**: verdadeiro (*True*) ou falso (*False*).

= = igual

!= diferente

> maior

>= maior ou igual

< menor

<= menor ou igual

# Operadores Aritméticos

Exemplos:

$$5 > 4$$

$$A \leftarrow 5 > 4$$

$$(5 + 6) > (7 + 8)$$

$$(5 * 2) >= ((4 * 5) / 2)$$

Perceba que no segundo exemplo a variável 'A' recebe o valor lógico 'True'.

# Operadores Lógicos

Resultado da operação é um **valor lógico**: verdadeiro (*True*) ou falso (*False*).

and (*e* lógico)

or (*ou* lógico)

not (*não* lógico)

Só fazem sentido se aplicados a variáveis com valor lógico (*True* ou *False*).

# Operadores Lógicos

Seguem a **tabela verdade**:

V para verdadeiro (*True*); F para falso (*False*)

<b>a</b>	<b>b</b>	<b><i>a and b</i></b>	<b><i>a or b</i></b>	<b><i>not a</i></b>
<b>V</b>	<b>V</b>	<b>V</b>	<b>V</b>	<b>F</b>
<b>F</b>	<b>V</b>	<b>F</b>	<b>V</b>	<b>V</b>
<b>V</b>	<b>F</b>	<b>F</b>	<b>V</b>	<b>F</b>
<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>	<b>V</b>

# Operadores Lógicos

Exemplos:

$(5 > 4) \text{ and } (4 > 4)$

$A \leftarrow (5 > 4) \text{ and } (4 > 4)$

$B \leftarrow \text{not } A$

$C \leftarrow B \text{ or } A$

$\text{not } ((5 + 6) > (7 + 8))$

$((5 * 2) > ((4 * 5) / 2)) \text{ or not } A$



# Outras funções

- Existem muitas outras funções disponíveis.
- No Python estas funções estão embutidas em **módulos** como o **math**.

`sqrt(X)`: Calcula a raiz quadrada de X

`pow(X,Y)`: Eleva X ao valor de Y

`sin(X)`: Calcula o seno de X (ângulo em radianos)

`cos(X)`: Calcula o cosseno de X (ângulo em radianos)

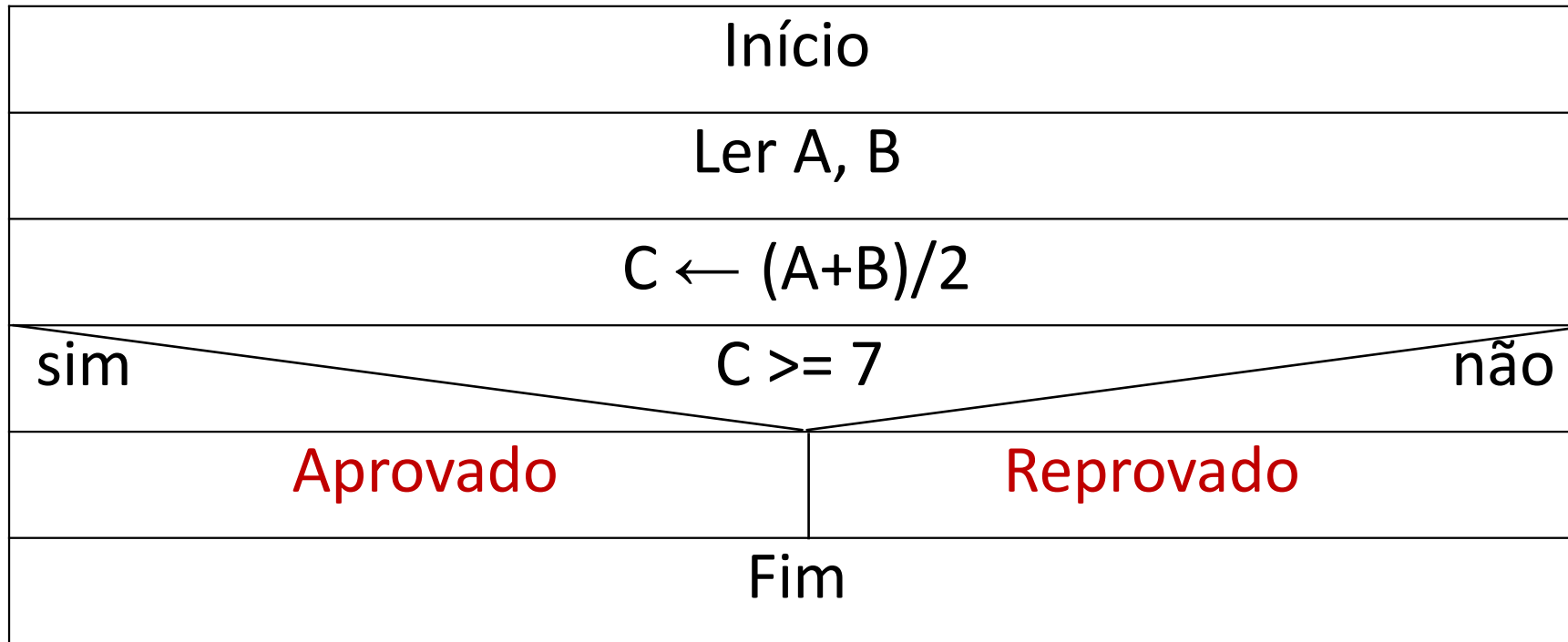
`random( )`: Produz um número aleatório entre 0 e 1

# Algoritmos

- Existem várias formas de **representar um algoritmo**.
- Com frequência, antes de começar a criar um programa que implemente o algoritmo, usamos alguma destas representações:
  - Diagrama de Chapin
  - Fluxograma
  - Pseudocódigo

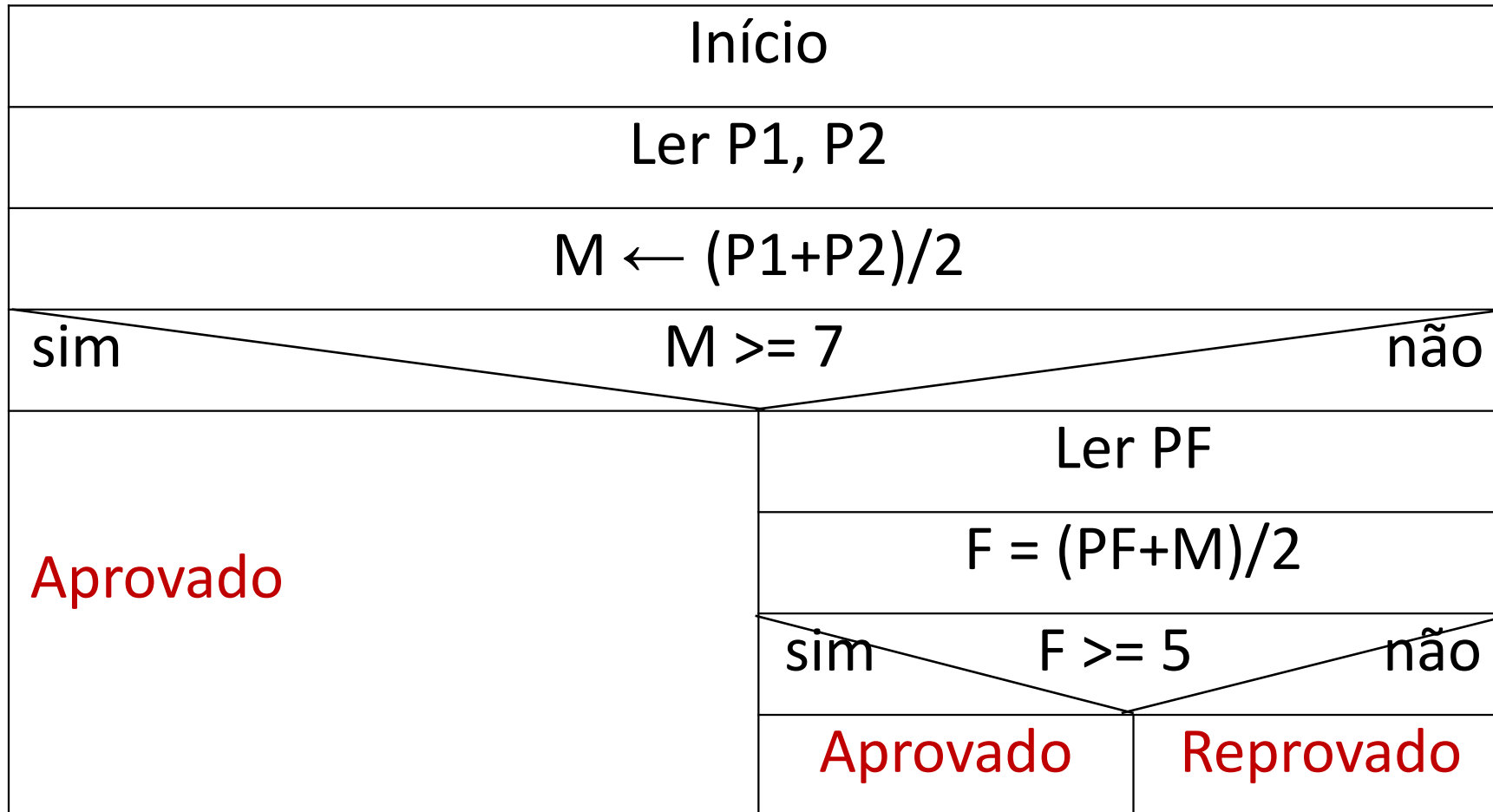
# Diagrama de Chapin

Algoritmo para ler duas notas dizer se aluno foi aprovado ou reprovado.




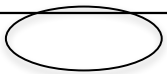
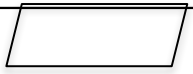
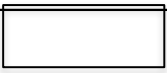
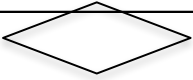
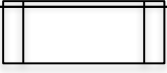
# Diagrama de Chapin

Com prova final...



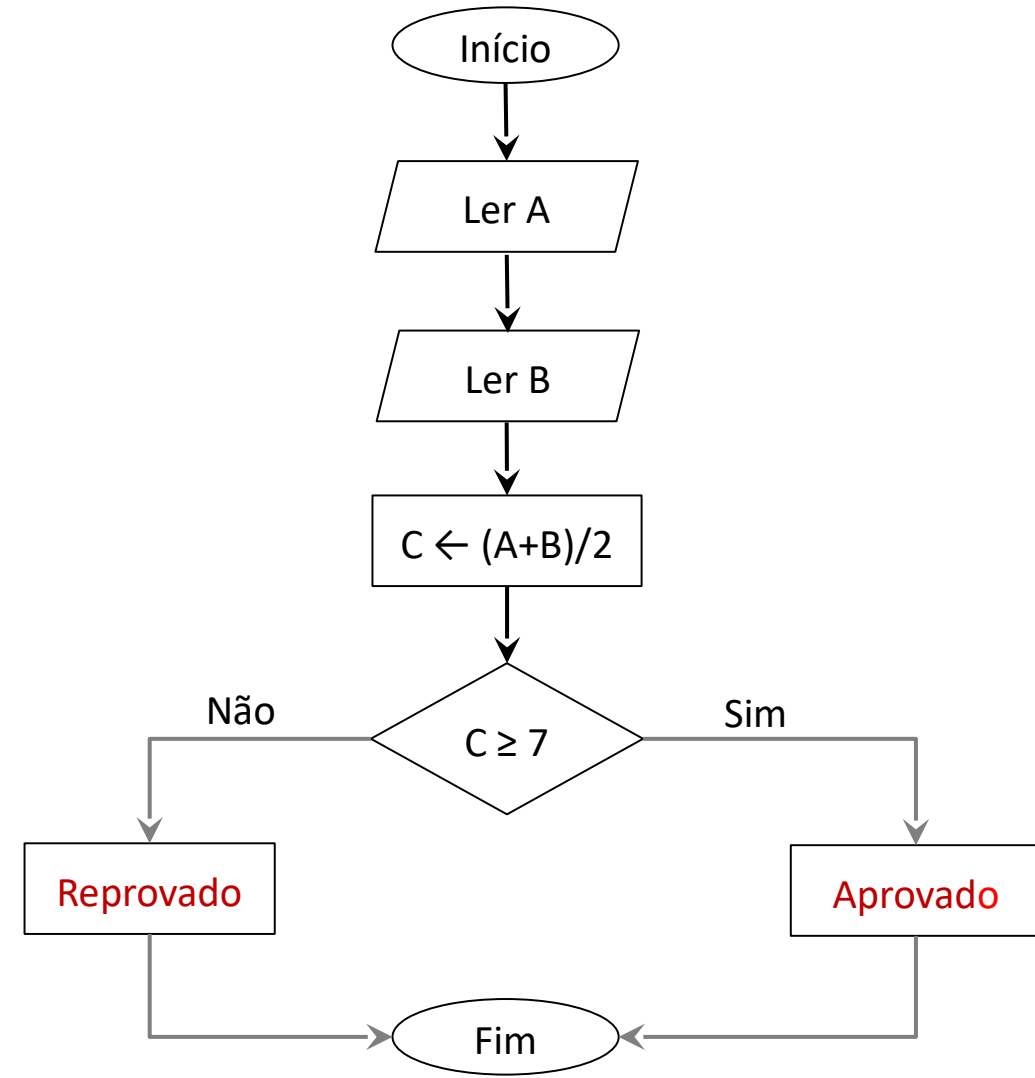
# Fluxograma

- Representação esquemática de um algoritmo.
- **Grafo dirigido** composto por formas que representam diferentes ações (formas básicas):

Forma	Nome	Descrição
	Seta	Indica que o controle passa para a forma apontada
	Terminal	Representa o começo ou término do algoritmo
	Entrada/Saída	Representa entrada ou saída de dados
	Processo	Representa uma ação/cálculo/processo
	Decisão	Representa uma decisão
	Proc. Predefinido	Representa um outro fluxograma (aninhado)

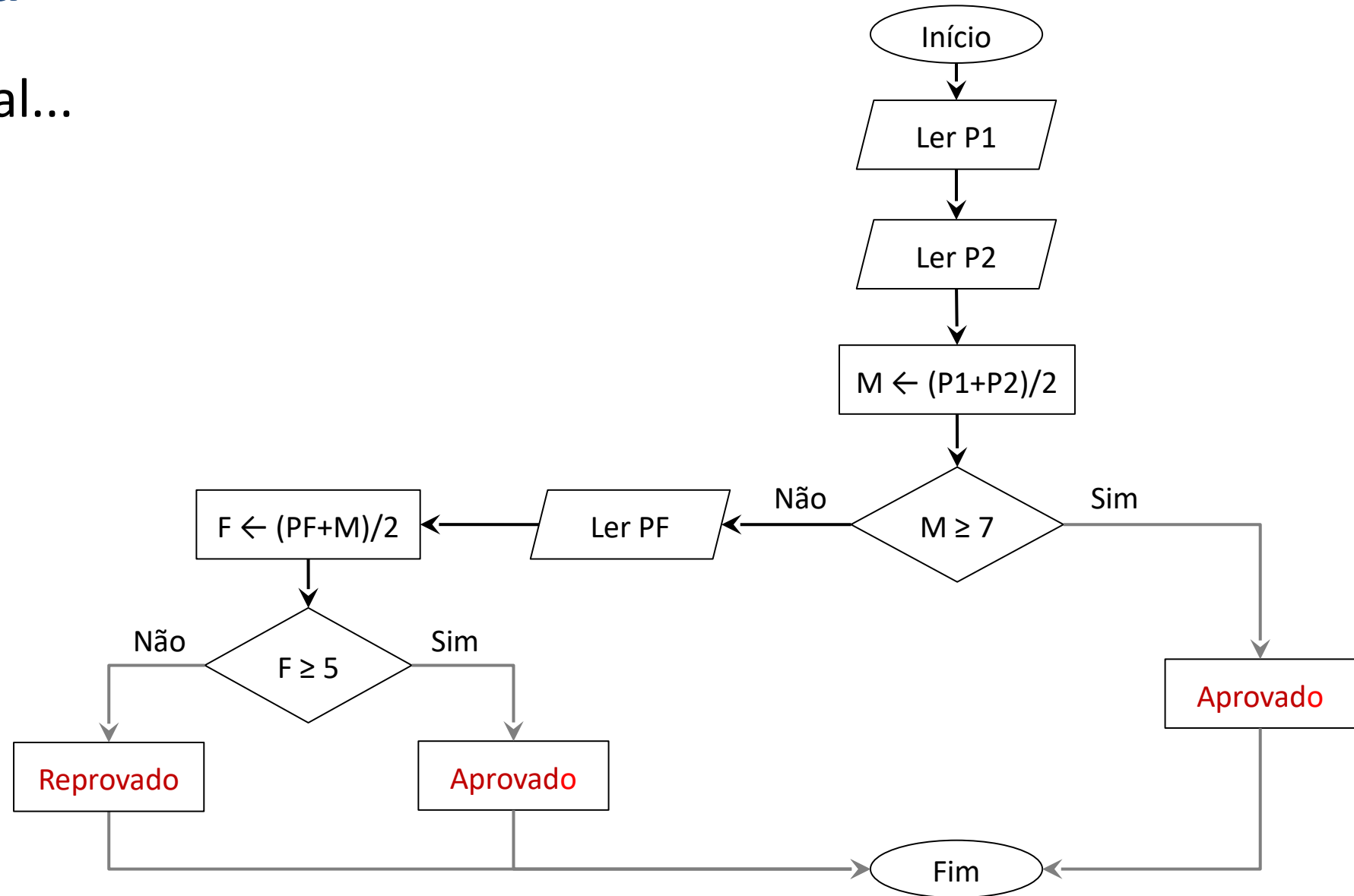
# Fluxograma

Algoritmo para ler duas notas dizer se aluno foi aprovado ou reprovado.



# Fluxograma

Com prova final...



# Pseudocódigo

- Forma genérica de escrever um algoritmo, utilizando uma **linguagem simples**, próxima da natural.
- Não há necessidade de conhecer a sintaxe de nenhuma linguagem de programação.
- Não pode ser executado num sistema real (computador), de outra forma deixaria de ser *pseudo*.



# Pseudocódigo

- Há algumas propostas mais rígidas (em termos de sintaxe) de português estruturado: Portugol, G-Portugol, Portugol Viana.
- Estes formalismos não se justificam neste curso: nos concentraremos posteriormente na sintaxe da linguagem Python.

# Algoritmo 1

Ler o valor de dois números e escrever a soma:

```
algoritmo soma
inicio
    ler a
    ler b
    resultado  $\leftarrow$  a + b
    escrever 'soma:'
    escrever resultado
fim
```

## Algoritmo 2

Ler o valor de três números e escrever a média:

```
algoritmo media_aritmetica
inicio
    ler n1, n2, n3
    media  $\leftarrow$  (n1 + n2 + n3)/3
    escrever 'media:', media
fim
```

## Algoritmo 3

Ler uma temperatura em Celsius e escreve-la em Farenheit:

```
algoritmo conversao_temperatura
inicio
    ler c
     $f \leftarrow (c/5)*9 + 32$ 
    escrever 'temp. em Farenheit:', f
fim
```

## Algoritmo 4

Ler dois valores inteiros e trocar o conteúdo desses valores (escrever os valores antes e depois da troca)

```
algoritmo troca
inicio
    ler a, b
    escrever 'a:' a, ' b:', b
    aux ← a
    a ← b
    b ← aux
    escrever 'a:' a, ' b:', b
fim
```

# Algoritmo 5

Ler o valor do tempo em segundos e imprimir e hora, minuto e segundos, e.g, 4000s = 1h 6min 40s

```
algoritmo converte_segundos
inicio
    ler tempo
    h ← tempo // 3600
    resto ← tempo % 3600
    min ← resto // 60
    seg ← resto % 60
    escrever h, 'h ', min, 'min ', seg, 's'
fim
```

## Algoritmo 6

Ler os valores de A, B e C e imprimir as raízes da equação do segundo grau correspondente.

```
algoritmo bhaskara
inicio
    ler A, B, C
    delta ← sqrt((B**2)-4*A*C)
    R1 ← (delta-B)/(2*A)
    R2 ← (delta+B)/(-2*A)
    escrever 'R1:', R1, ' R2:', R2
fim
```

## Algoritmo 6

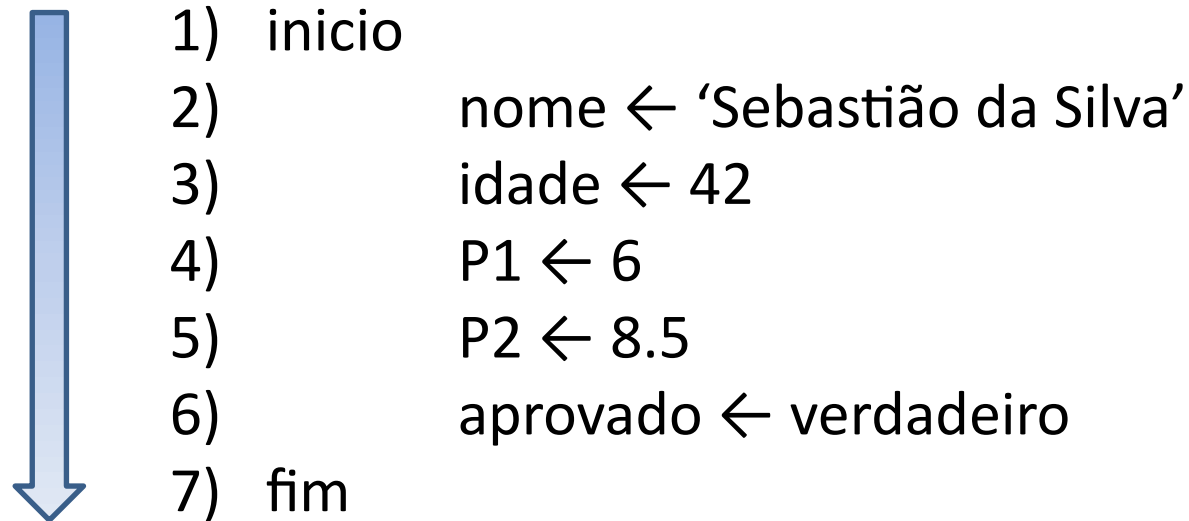
Ler valores em hora, minuto e segundo e transformar tudo para segundos.

```
algoritmo converte_segundos2
inicio
    ler h, min, seg
    total_seg ← h*3600+min*60+seg
    escrever 'tempo: ', total_seg, ' seg'
fim
```



# Fluxo de Processamento

Ordem em que as instruções (linhas) do algoritmo/programa são executados.

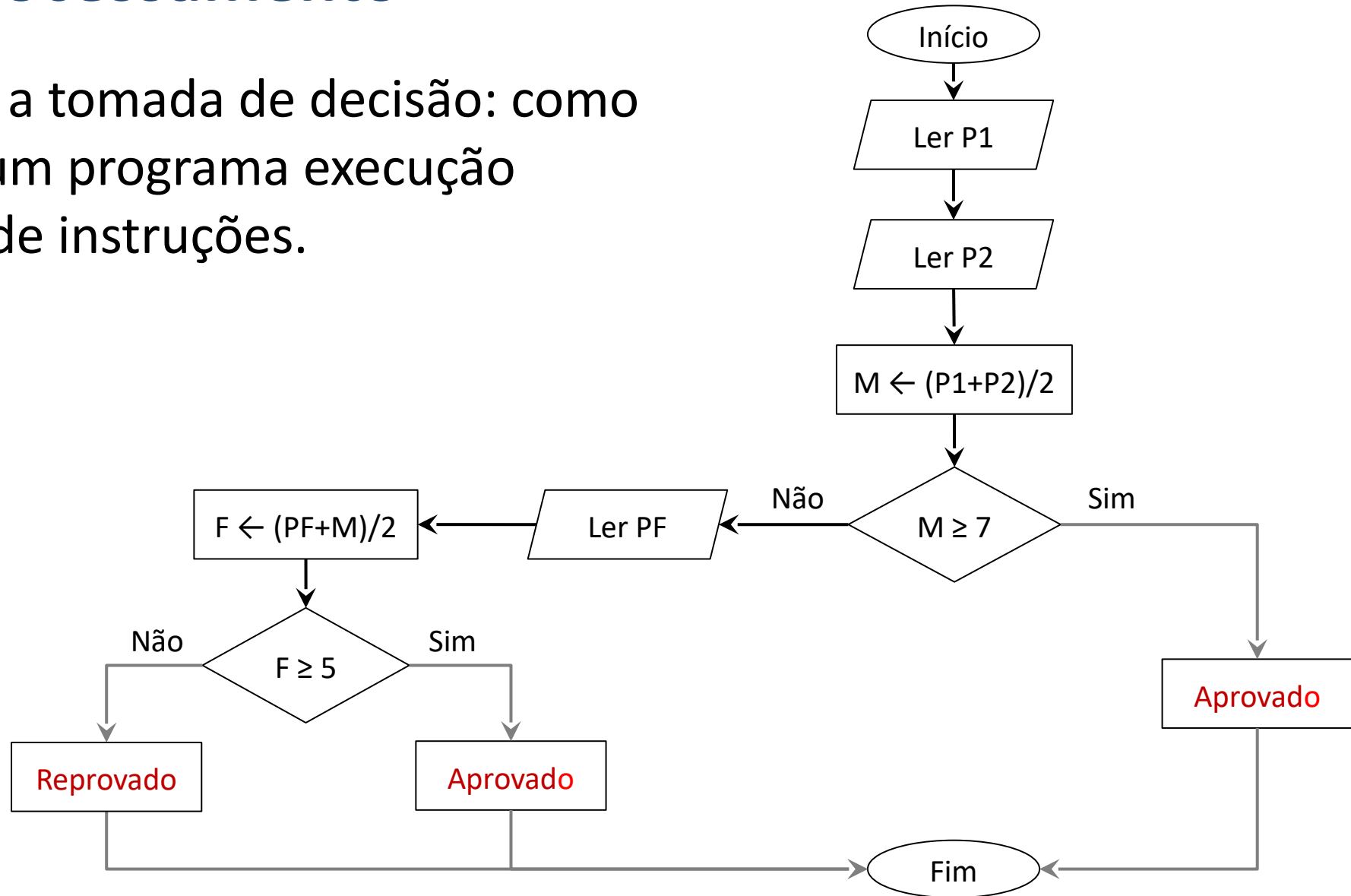


# Fluxo de Processamento

- Até agora vimos algoritmos (em pseudocódigo) que seguem um fluxo de processamento **sequencial**.
- Uma instrução é sempre executada depois da outra.
- A execução de certas instruções pode ser **condicionada**: só é executada se passar por um teste (decisão).
- Estrutura para a tomada de decisão: como representar num programa execução condicionada de instruções.

# Fluxo de Processamento

Estrutura para a tomada de decisão: como representar num programa execução condicionada de instruções.



# Fluxo de Processamento

- Além disso, em muitos casos vamos querer **repetir** de forma controlada um conjunto de instruções.
- A repetição (*loop*) deve acontecer **enquanto alguma condição é atendida**.
- Estes são os assuntos da próxima aula.



## Introdução ao Processamento de Dados Turma 3 (2020.1)



# Introdução a Algoritmos

**Gilson. A. O. P. Costa (IME/UERJ)**

[gilson.costa@ime.uerj.br](mailto:gilson.costa@ime.uerj.br)