

# Lógica de programação





Lógica de  
programação



Estruturas de  
Controle



# Lógica de programação



# Lógica de Programação

---

Lógica de programação é a organização coesa de uma sequência de instruções voltadas à resolução de um problema, ou à criação de um software ou aplicação.

Cada linguagem tem suas próprias particularidades, como sua sintaxe, seus tipos de dados e sua orientação, mas a lógica por trás de todas é a mesma.

A lógica de programação é importante porque é ela quem nos dá as ferramentas necessárias para executar o processo mais básico no desenvolvimento de alguma aplicação: a criação de seu **algoritmo**.

# Algoritmo

---

De acordo com o dicionário, é um processo de cálculo que, por meio de **uma sequência finita** de operações, aplicada a um número finito de dados, **leva à resolução de problemas**.

Vamos tomar como exemplo o café que tomamos de manhã.

Quando perguntam como tomamos nosso café, a maioria de nós responde que, ao acordarmos, preparamos o café com auxílio de uma cafeteira elétrica, colocamos ele em uma caneca e o tomamos.

Mas, ao destrinchar este processo, somos capazes de estipular uma sequência de passos que nos levaram ao ato final de beber este café. Esta sequência pode ser:

# Algoritmo

---

Podemos dividir um algoritmo em três fases fundamentais: **entrada, processamento e saída**.

**Entrada** recebe as informações necessárias para iniciar nosso algoritmo;

**Processamento** são os passos necessários para atingir nosso objetivo;

**Saída** é o resultado esperado da fase de processamento.

# Algoritmo

---

1. Ao acordar, levanto da cama;
2. Após levantar da cama, desço as escadas;
3. Após descer as escadas, entro na cozinha;
4. Após entrar na cozinha, pego o pó de café no armário;
5. Após pegar o pó de café, o coloco dentro da cafeteira;
6. Após colocar o pó na cafeteira, jogo água no compartimento específico;
7. Após inserir todos os ingredientes na máquina, aperto o botão de ligar;
8. Quando o café está pronto, pego a garrafa;
9. Após pegar a garrafa, despejo o café dentro de uma caneca;
10. Após colocar o café na caneca, bebo o café.

# Algoritmo

---

## Exercícios:

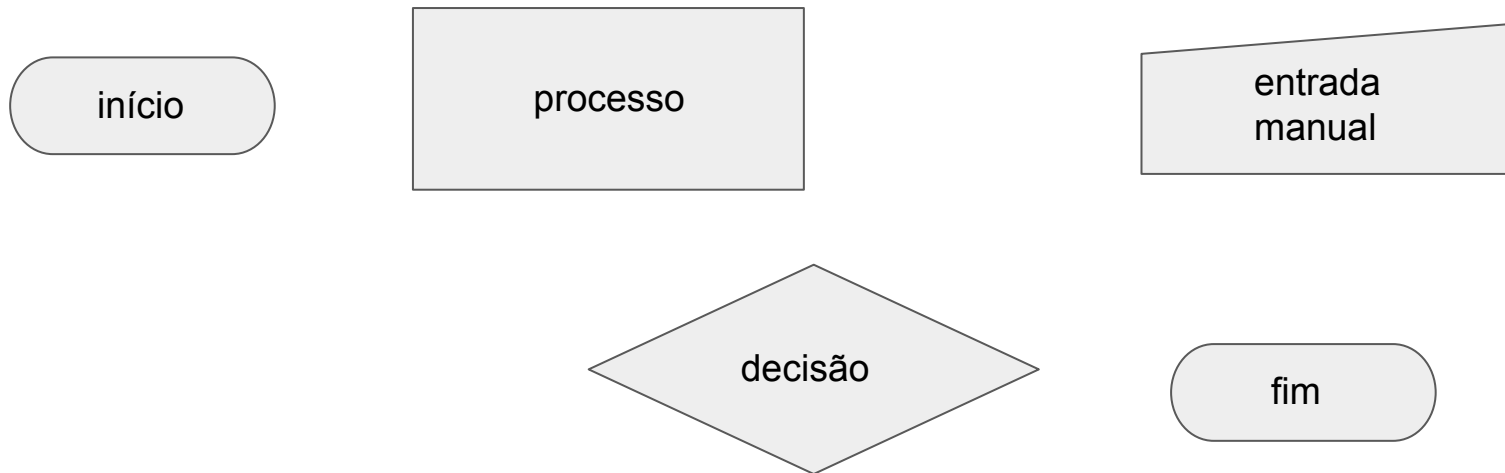
1. Faça um algoritmo que mostre o passo a passo para trocar uma de lâmpada queimada.
2. Faça um algoritmo que mostre o passo a passo para passear com seu animal de estimação.
3. Faça um algoritmo que mostre o passo a passo para acessar um computador.
4. Faça um algoritmo que mostre o passo a passo para lavar um copo
5. Faça um algoritmo que mostre o passo a passo para postar uma foto em um rede social



# Fluxograma

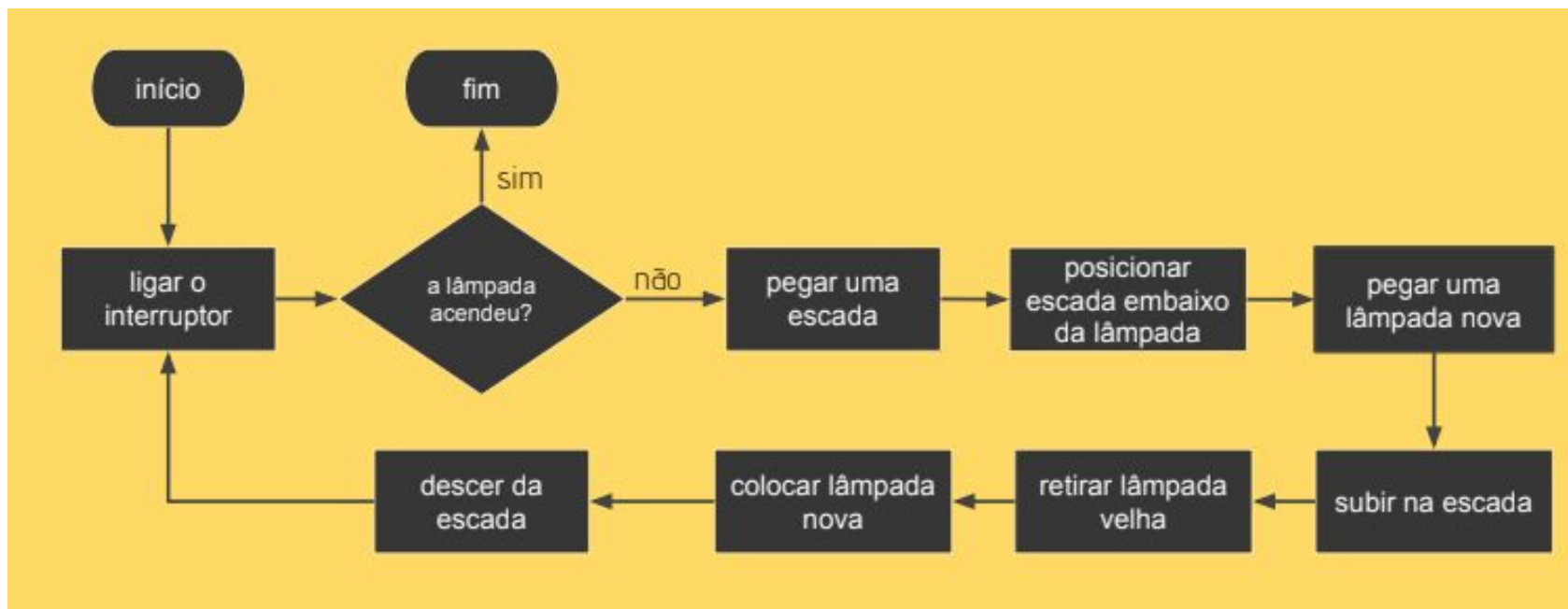
---

Um fluxograma é a representação gráfica de um procedimento, problema ou sistema, cujas etapas ou módulos são ilustrados de forma encadeada por meio de símbolos geométricos interconectados.



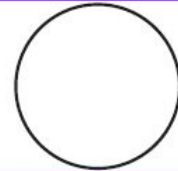
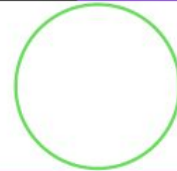
# Fluxograma

## Algoritmo troca de lâmpada



# Java

## Estruturas de controle



# Java

---

## Operadores aritméticos

+	operador de adição
-	operador subtração
*	operador de multiplicação
/	operador de divisão
%	operador de módulo (ou resto da divisão)

# Java

---

## Operadores de incremento e decremento

`numero++;`

`numero--;`

## Operadores de igualdade

igual (“**==**”) ou diferente (“**!=**”)

# Java

## Opções de operadores relacionais

>	Utilizado quando desejamos verificar se uma variável é maior que outra.
>=	Utilizado quando desejamos verificar se uma variável é maior ou igual a outra
<	Utilizado quando desejamos verificar se uma variável é menor que outra.
<=	Utilizado quando desejamos verificar se uma variável é menor ou igual a outra.

# Java

---

## Condicionais if / else if

```
if (expressão booleana 1) {  
    // bloco de código 1  
} else if (expressão booleana 2) {  
    // bloco de código 2  
} else {  
    // bloco de código 3  
}
```

## Condicional ternária if / else

(condição) ? valor se for verdadeiro : valor se for falso;

(expressão booleana) ? código 1 : código 2;

### **Exemplo:**

```
int numeroDia = 30; //valor entre 1 e 30  
System.out.println((numeroDias <= 15) ? "Primeira quinzena" :  
"Segunda quinzena");
```

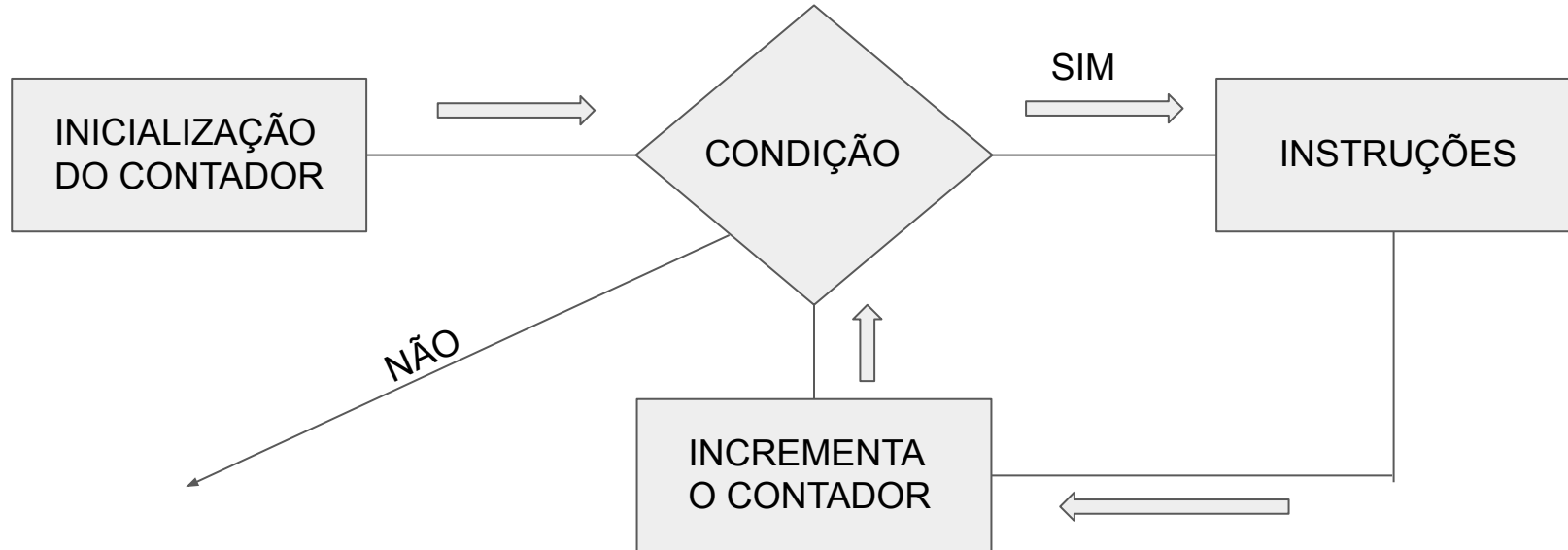
# Estrutura de repetição

A decorative graphic consisting of a white horizontal bar with a rounded right end, containing a solid purple circle. Below this bar is a black horizontal bar with a rounded left end. The background is a solid green color. In the lower right area, there are two white circles, one above the other, and a thin white horizontal line to the right of the black bar.



# Estrutura de repetição

Dentro da lógica de programação é uma **estrutura** que permite executar mais de uma vez o mesmo comando ou conjunto de comandos, de acordo com uma condição ou com um contador.



# Java

---

## While

O termo **while** pode ser traduzido para o português como “enquanto”. Este termo é utilizado para construir uma estrutura de repetição que executa, repetidamente, uma única instrução ou um bloco delas “enquanto” uma expressão booleana for verdadeira.

```
//INCREMENTADO - de 0 à 9
int i = 0;
while(i<=9){
    i = i + 1;
    System.out.println( i );
}
```

# Java

---

## Do While

A diferença desse iterador para os outros, é que o bloco de instrução será executado no mínimo uma única vez. Como podemos ver no exemplo abaixo:

```
int i = 0;
```

```
do{  
    System.out.println( i );  
    i++;  
}while(i <= 10);
```

# Java

---

## FOR

Controlando fluxo com laços

```
for (int i = 0; i < 5; i++) {  
    if( i % 2 == 0) {  
        System.out.println(i);  
    }  
}
```

## For Each

Projetado especificamente para iterar sobre matrizes e coleção de objetos.

```
String[] cars = {"Volvo", "BMW", "Ford",  
"Mazda"};
```

```
for (String i : cars) {  
    System.out.println(i);  
}
```

