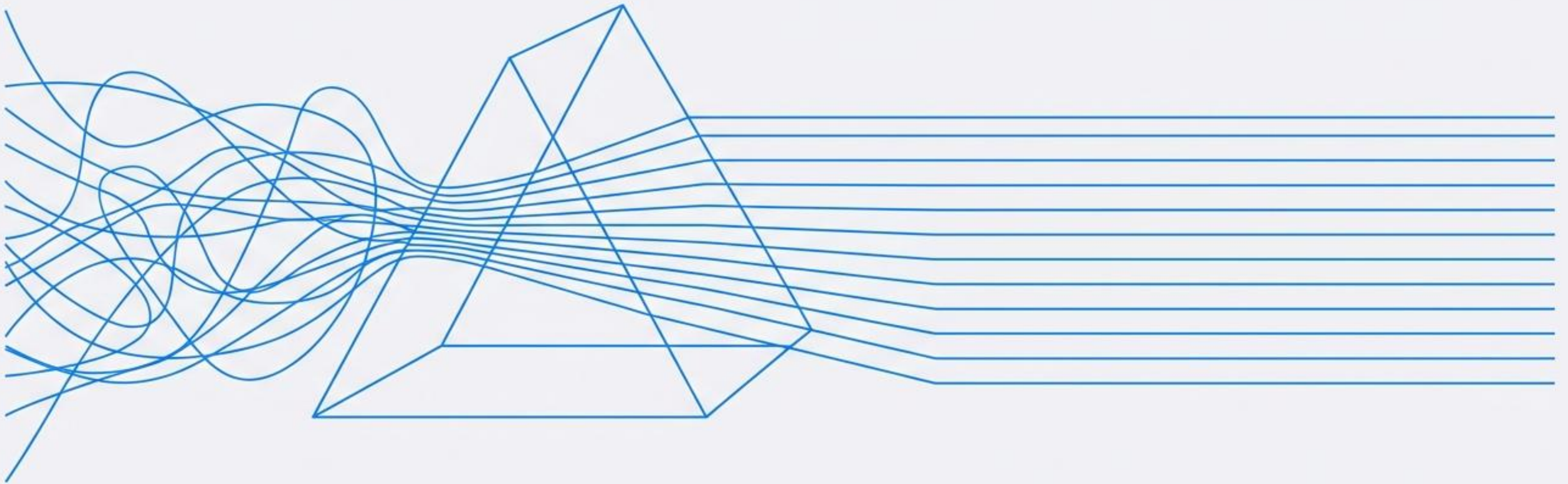





Algoritmos e POO: Aula 03

Estruturas de Controle e Lógica Booleana



Recap Aula 02: A Matéria-Prima

Inventário de Dados		
Variáveis  let (Mutável – O valor pode mudar) const (Imutável – O valor é fixo)	Tipos Primitivos  String (Texto) Number (Números inteiros ou decimais) Boolean (Verdadeiro ou Falso)	Operadores  Aritméticos: + - * / Lógicos: && (E) (OU) ! (NÃO)

Na última aula, aprendemos a armazenar dados. Hoje, vamos aprender a controlar o fluxo deles.

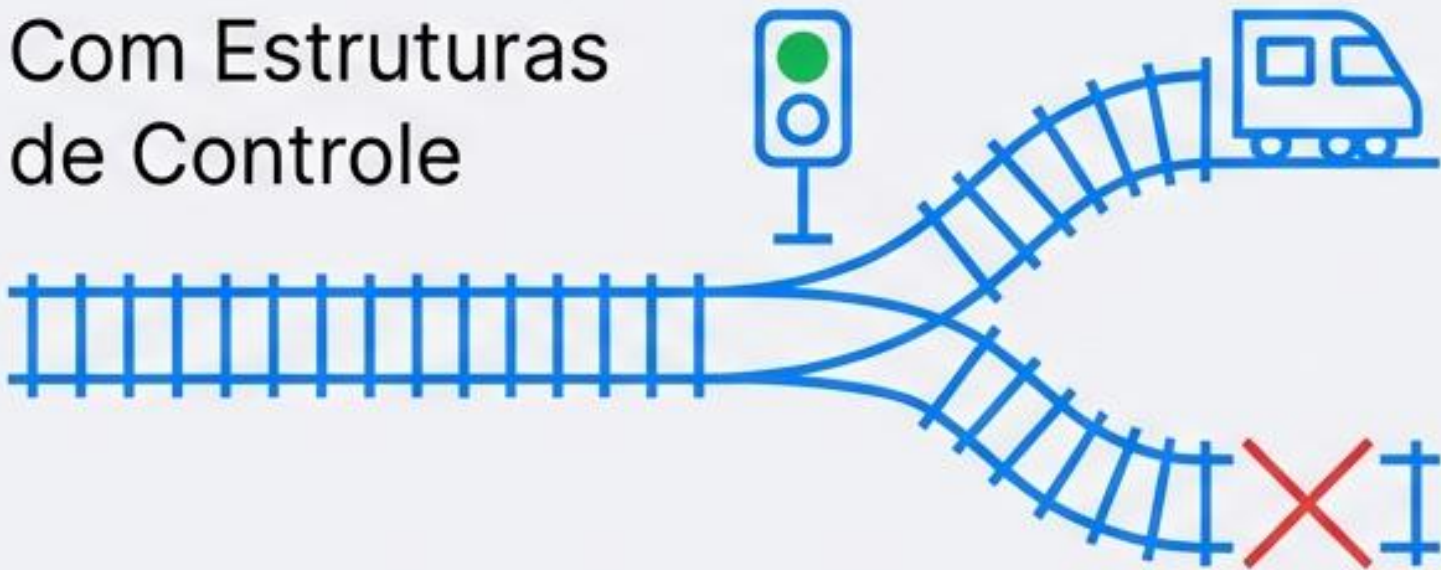
O Fluxo de Controle

Execução Padrão



Algoritmos inteligentes não são lineares. Eles tomam **decisões** e reagem a **condições**.

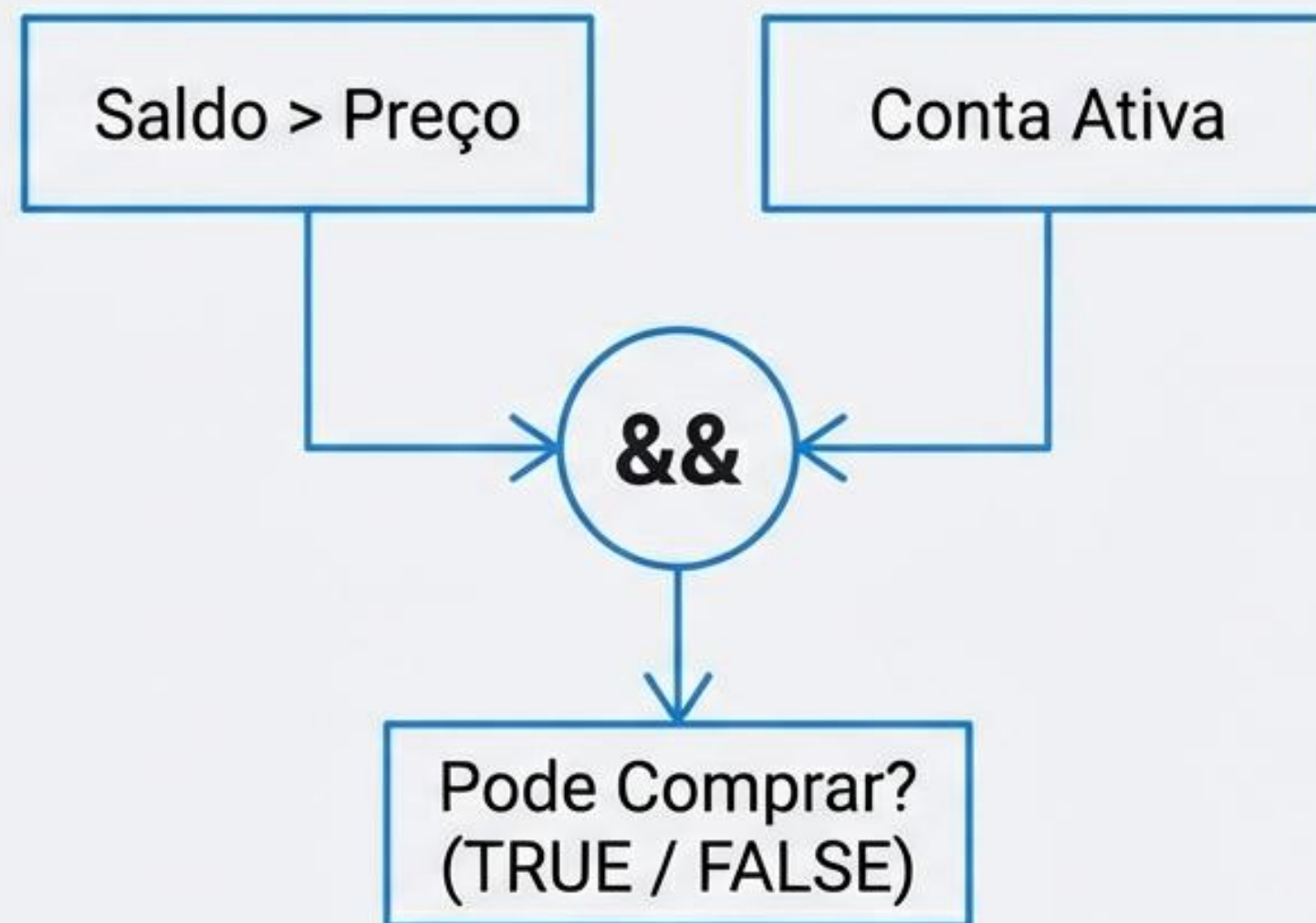
Com Estruturas de Controle



Por padrão, o código é lido de cima para baixo. As **Estruturas de Controle** nos permitem **alterar essa direção, pular etapas** ou **repetir tarefas**.

Lógica Booleana: O Cérebro da Operação

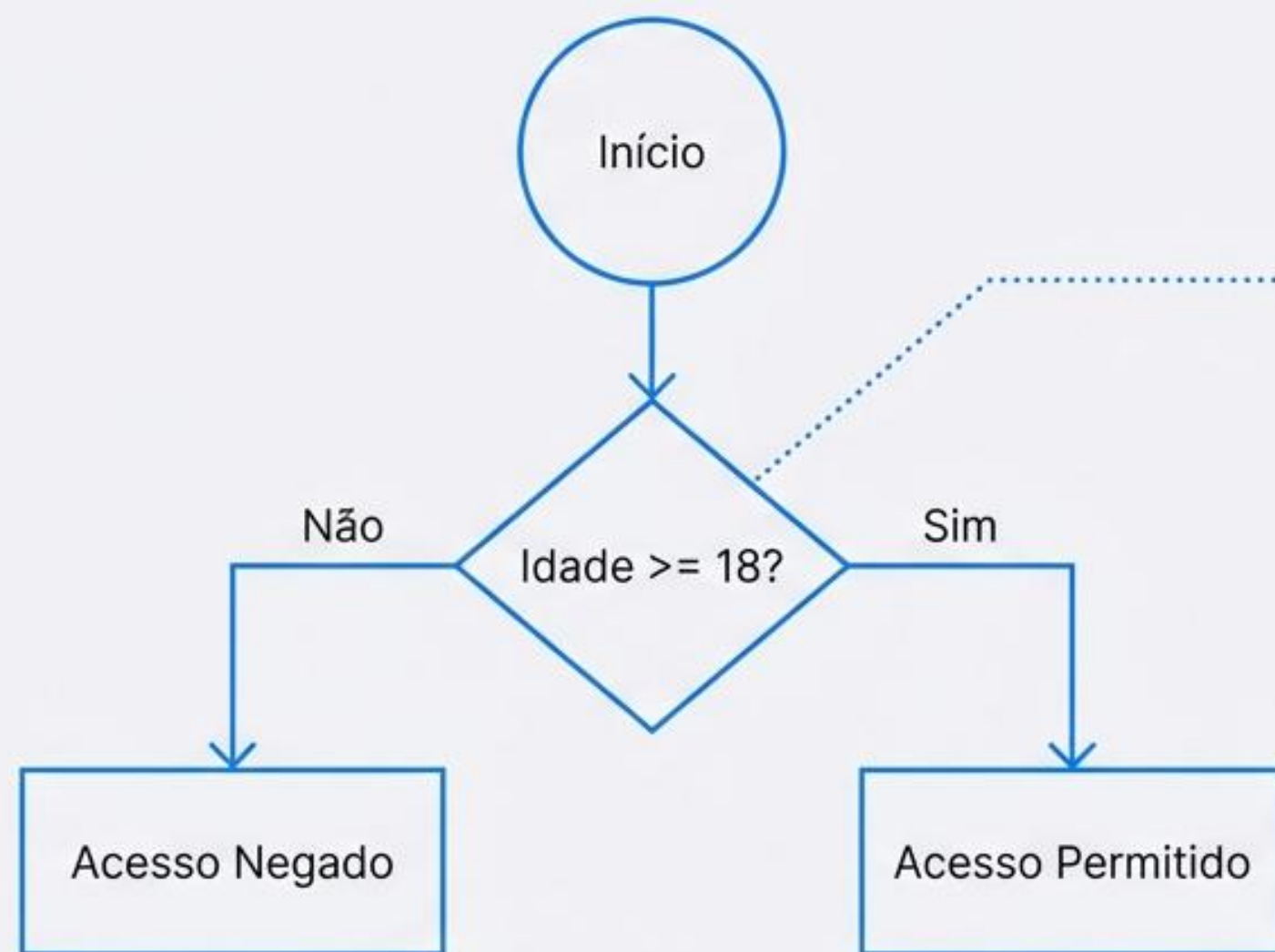
- > (Maior que)
- < (Menor que)
- === (Igualdade estrita)
- !== (Diferente de)
- >= (Maior ou igual)
- <= (Menor ou igual)



Antes de tomar uma decisão (If), o computador precisa avaliar se uma condição é **Verdadeira** ou **Falsa**.

If / Else: A Bifurcação

The Logic

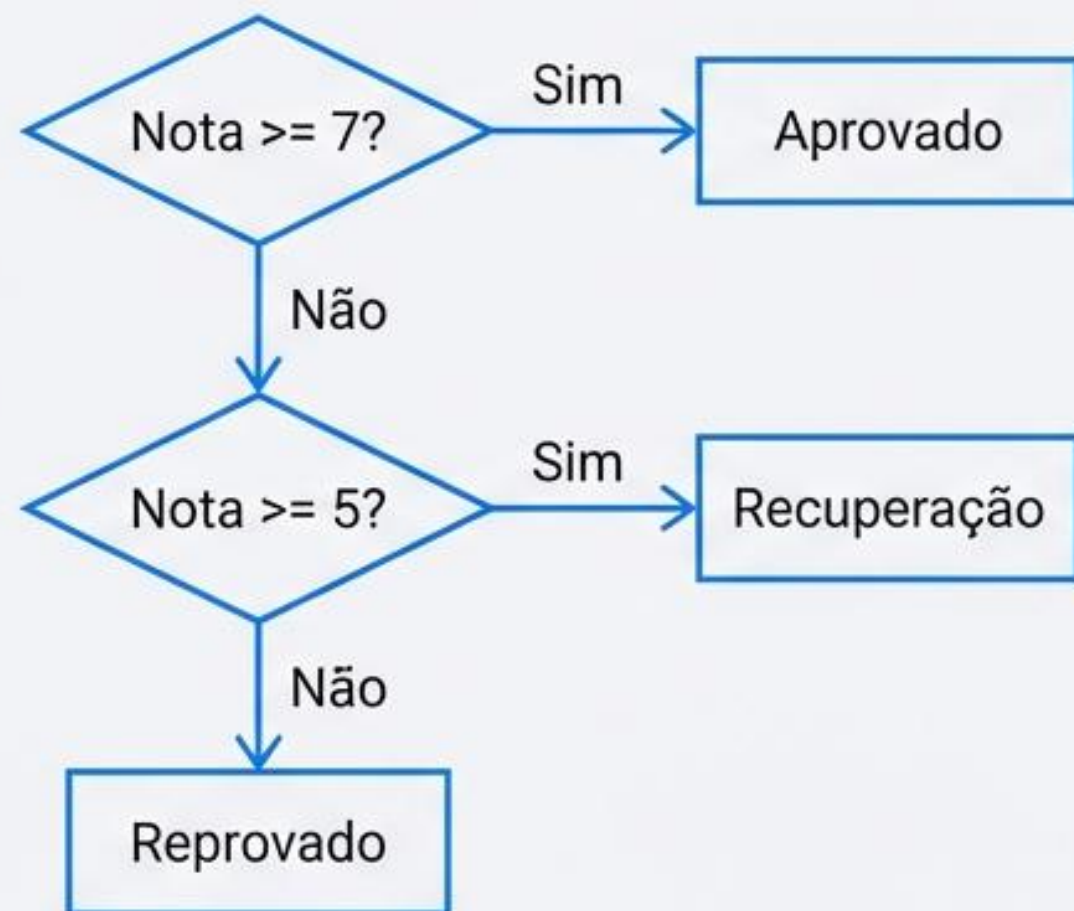


The Code

```
if (idade >= 18) {  
    console.log('Acesso Permitido');  
} else {  
    console.log('Acesso Negado');  
}
```

If / Else If: Múltiplos Caminhos

The Logic



The Code

```
if (nota >= 7) {  
    console.log('Aprovado');  
} else if (nota >= 5) {  
    console.log('Recuperação');  
} else {  
    console.log('Reprovado');  
}
```

O código para na primeira condição **verdadeira** encontrada.

Switch Case: O Menu de Opções

Use o Switch quando você compara a mesma variável contra diferentes valores específicos.



```
switch (tipoUsuario) {  
  case 'Admin':  
    console.log('Acesso Total');  
  
  case 'Visitante':  
    console.log('Acesso Leitura');  
    break;  
  default:  
    console.log('Permissão desconhecida');  
}
```

Estruturas de Repetição: Automatizando o Tédio

```
console.log(1);  
console.log(2);  
console.log(3);  
console.log(4);  
console.log(5);  
console.log(6);  
console.log(7);  
console.log(8);  
console.log(9);  
console.log(10);
```



```
for (int i = 1; i <= 10; i++) {  
    console.log(i);  
}
```

DRY Principle (Don't Repeat Yourself).

Se você precisa fazer a mesma tarefa mais de uma vez, use um **loop**.
Repetições manuais geram erros e **código sujo**.

O Laço 'For' (Para)

Quando você sabe exatamente quantas vezes quer repetir.



O Laço "While" (Enquanto)

Quando você não sabe quantas vezes vai repetir, mas sabe a condição de parada.



Enquanto (arquivo não baixou)
→ mostre "**Carregando...**"

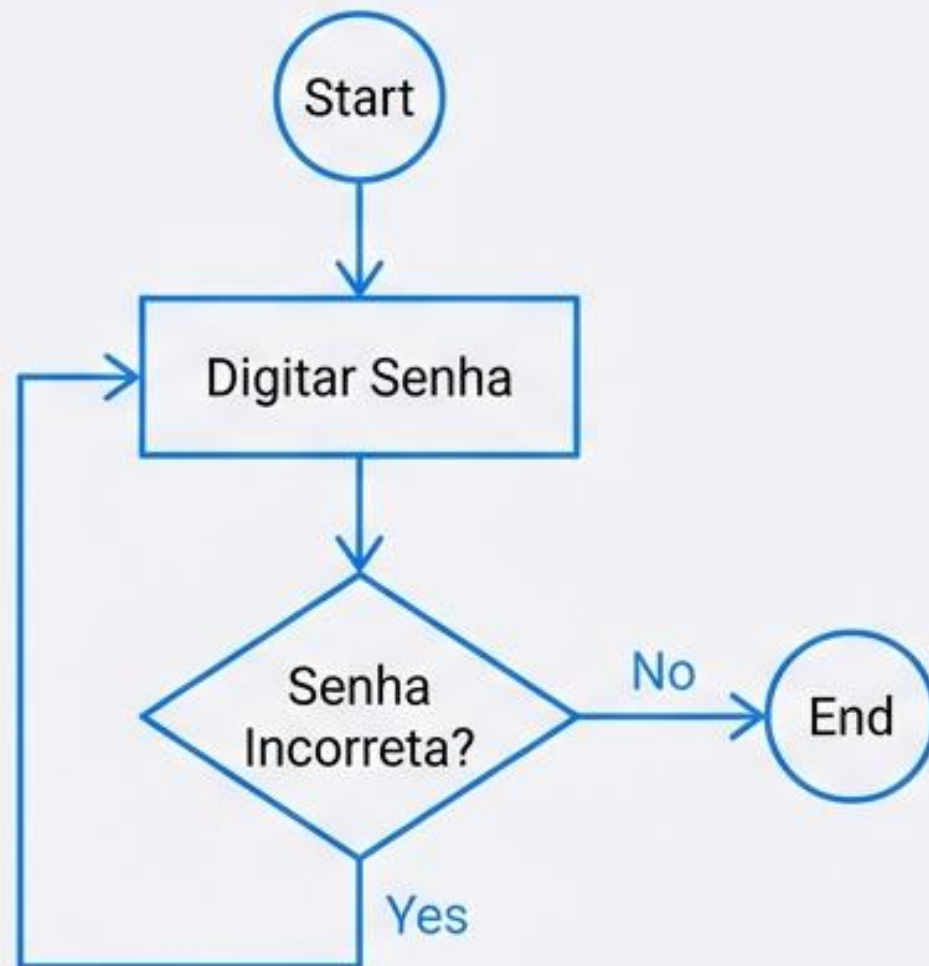
```
while (carregando == true) {  
    exibirSpinner();  
    verificarDownload();  
}
```



Cuidado com Loops Infinitos! Se a condição nunca mudar para 'falso', o programa trava.

Do / While (Fazer... Enquanto)

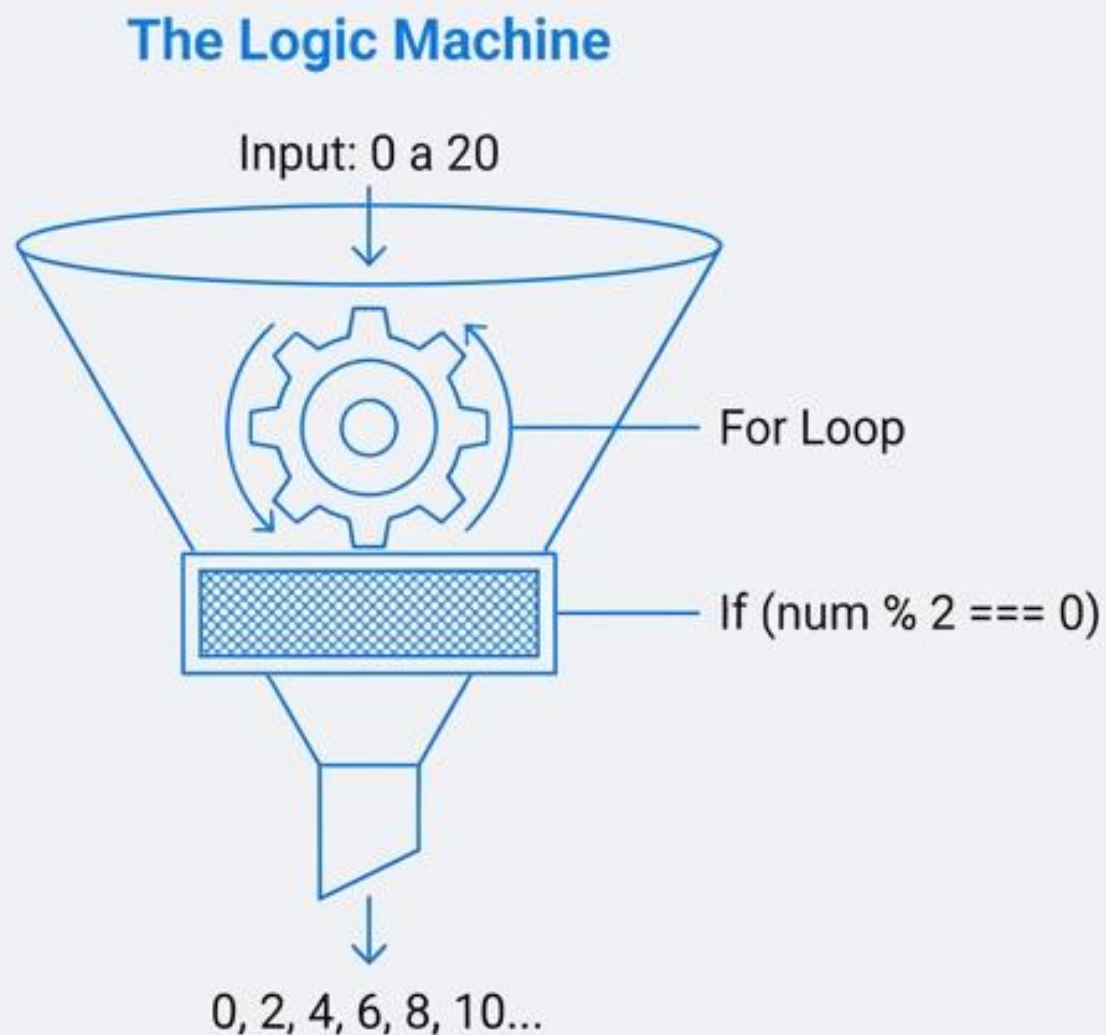
Garante que o bloco de código execute **pelo menos uma vez** antes de verificar a condição.



```
do {  
    senha = pedirSenha();  
} while (senha != '1234');
```

Desafio Prático: O Filtro

Identifique e imprima apenas os números PARES entre 0 e 20.



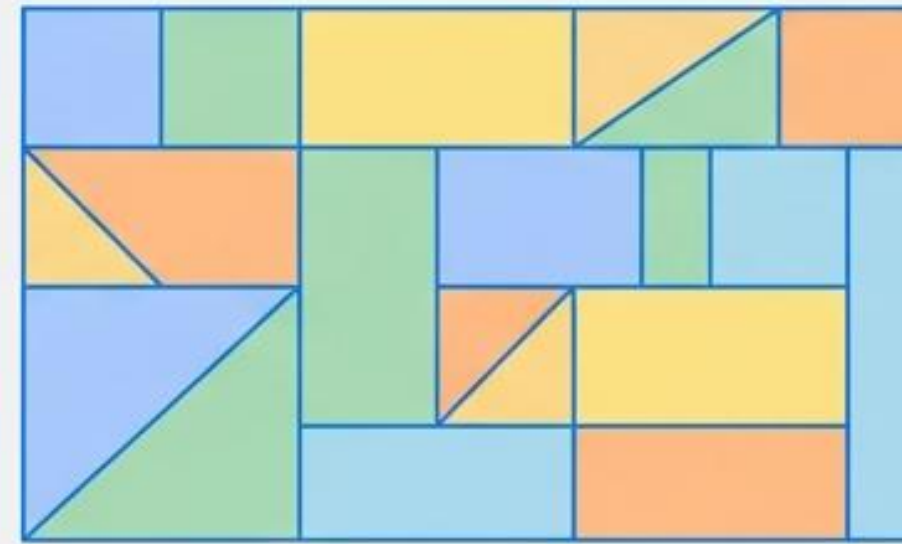
Solução: Combinação de **For** (para percorrer) com **If** (para filtrar).

Preview Aula 04: Funções

O Próximo Nível: Modularização



Antes (Código Solto)



Depois (Modularização)

- Hoje vimos como controlar o fluxo.
- Na próxima aula, vamos aprender a empacotar esse fluxo em ferramentas reutilizáveis.
- Palavras-chave: **function**, **arrow function**, **escopo**.

Dúvidas & Código Fonte

Pratique a lógica. O código é apenas a consequência do pensamento.



Repositório da Aula

github.com/gabriel-colares



Dúvidas?

gabriel.colares.dev@gmail.com