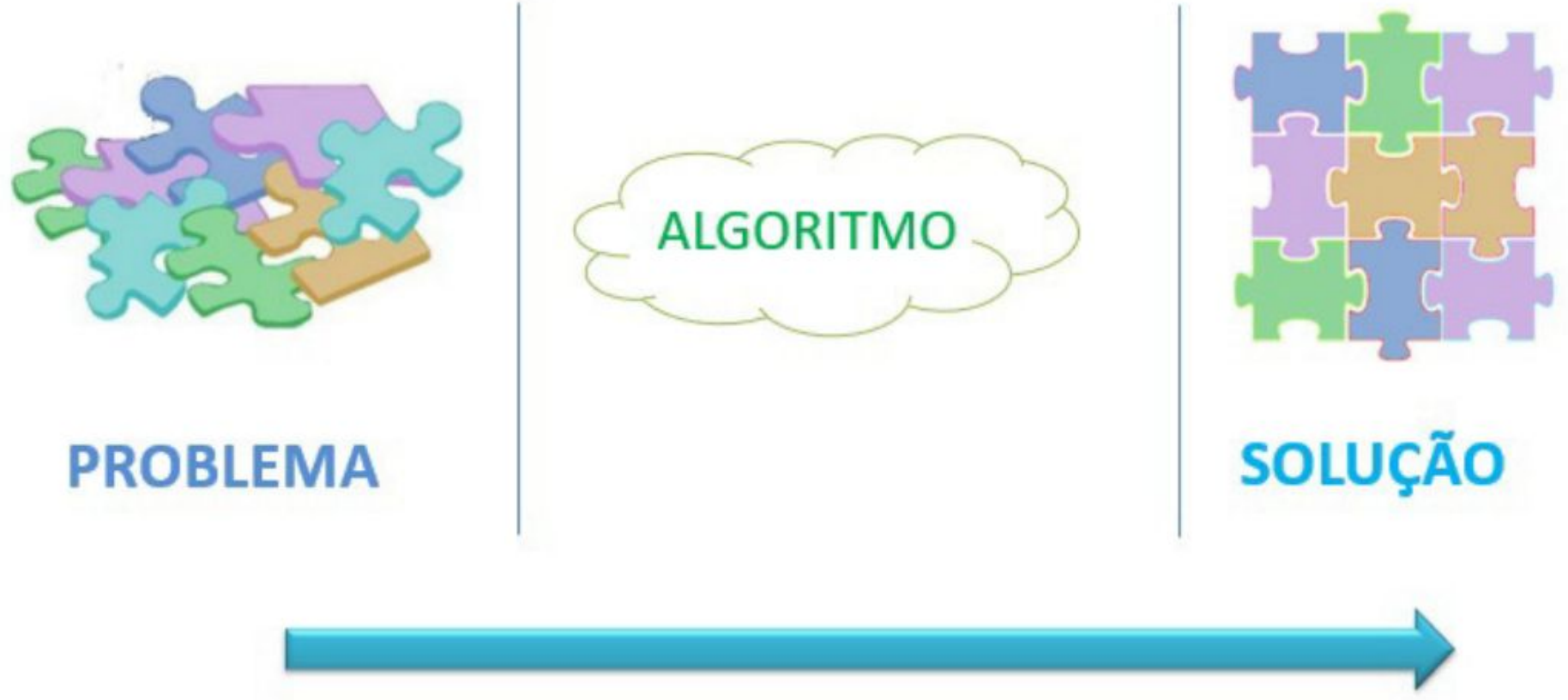
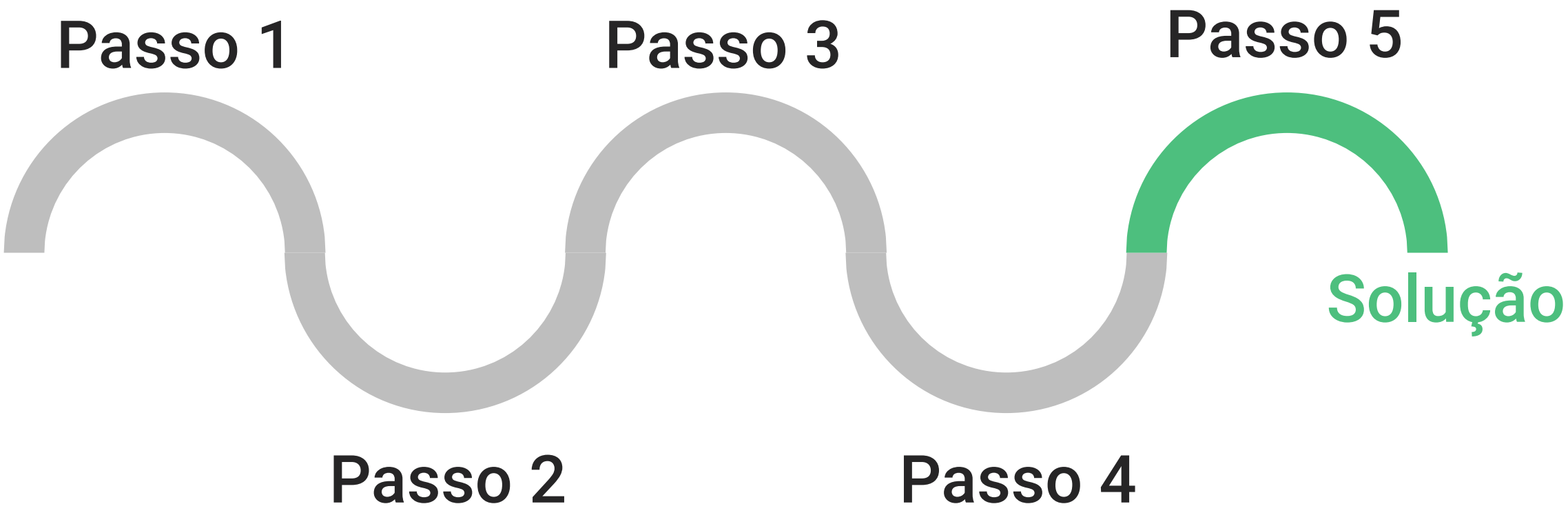
O que são Algoritmos ?

O que são Algoritmos?



Sequência de passos(finito) para resolver um problema.

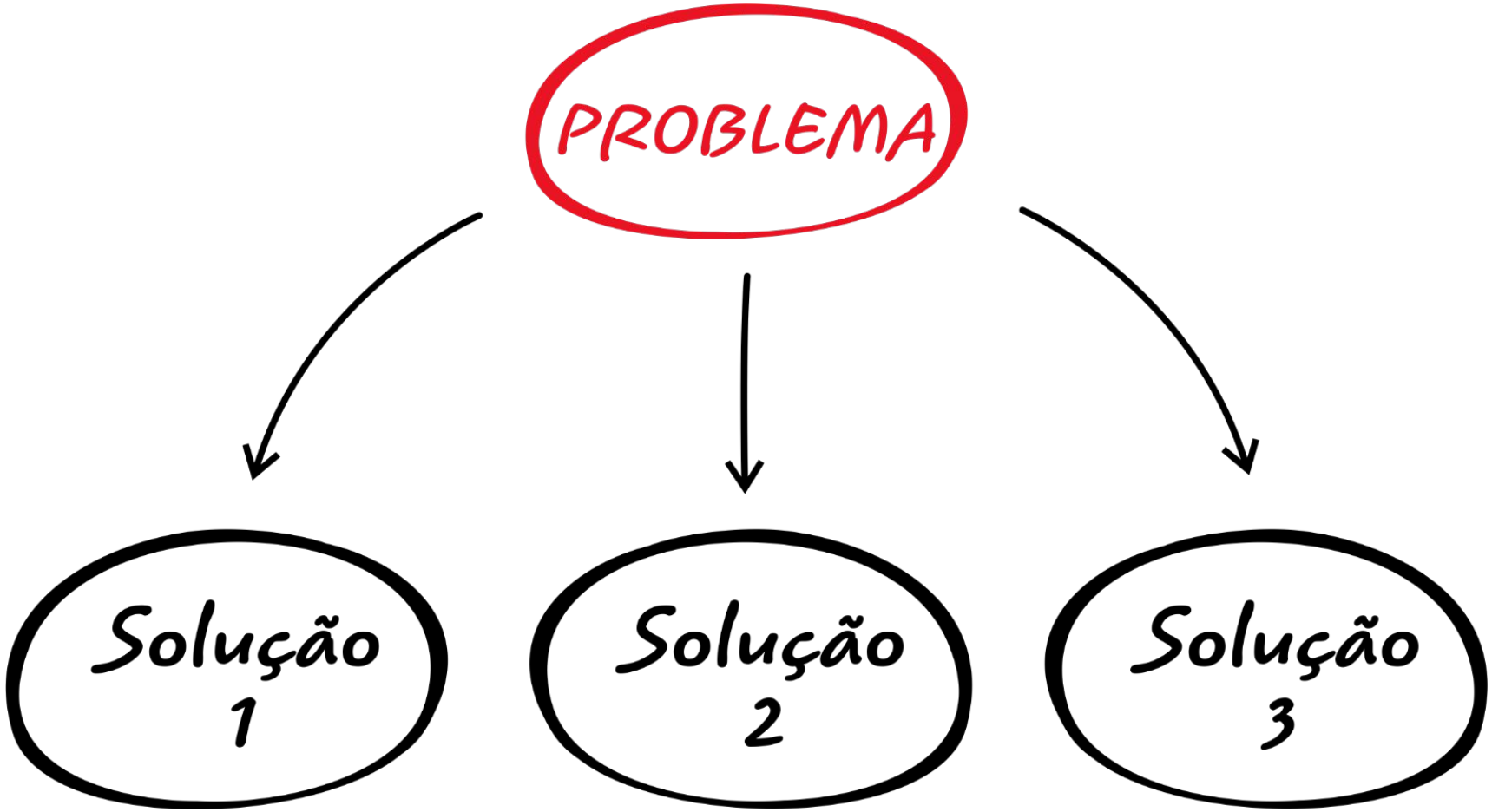
Creative Visual Design



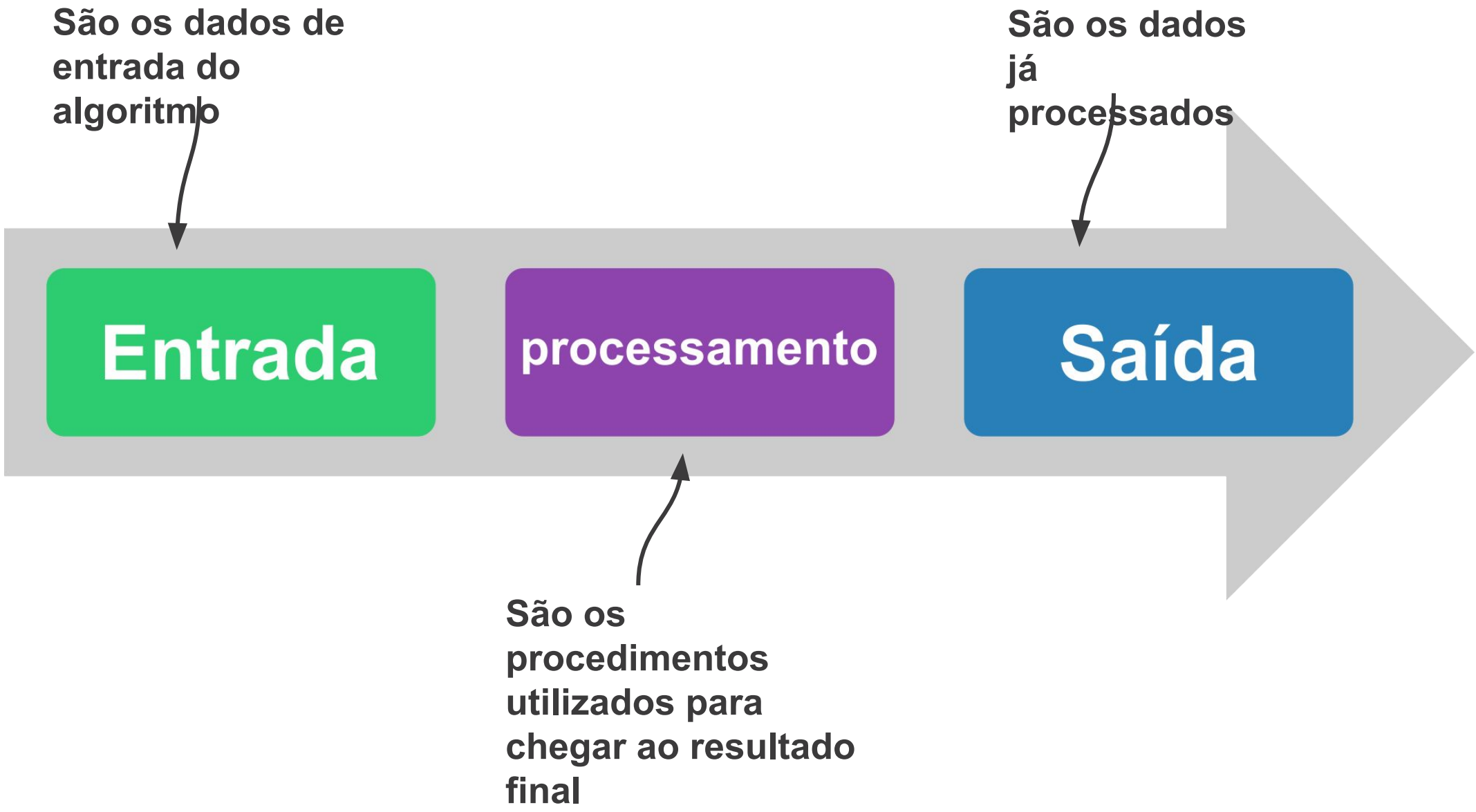
O que é lógica de programação ?

Maneira de organizar as intruções em um algoritmo.

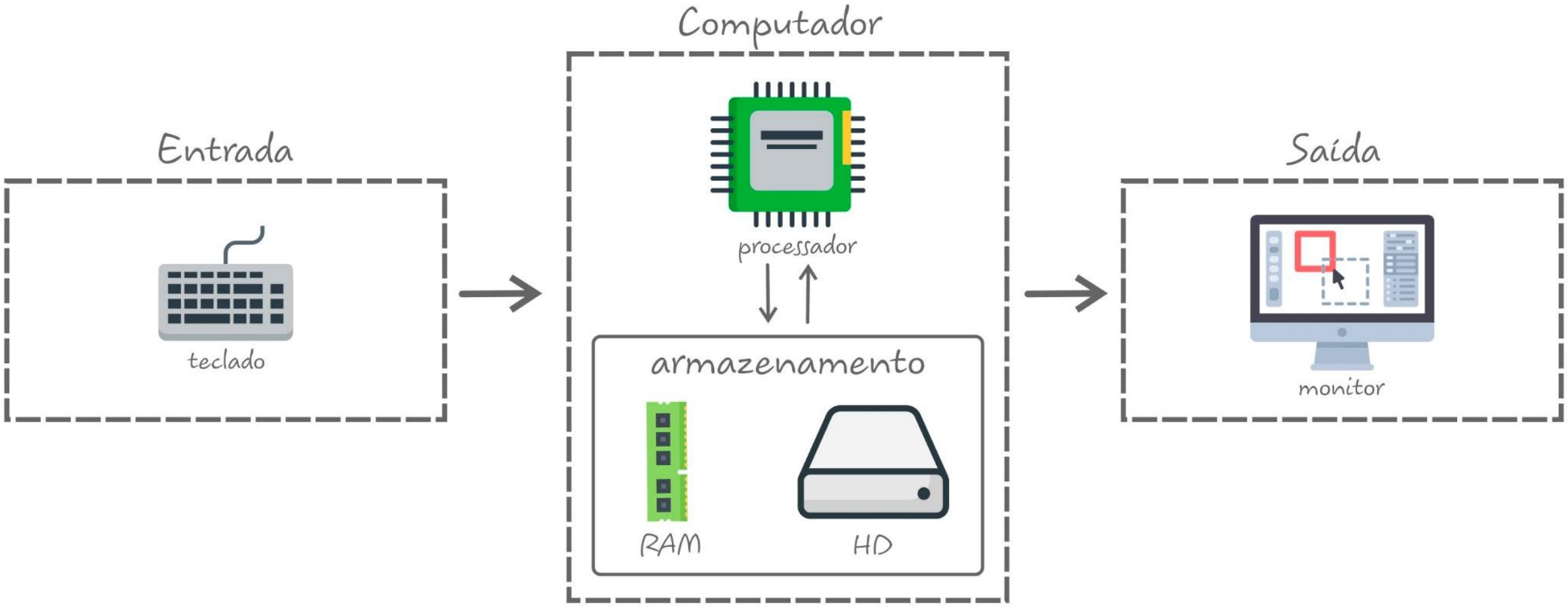
Mesmo problema, Algoritmos diferentes



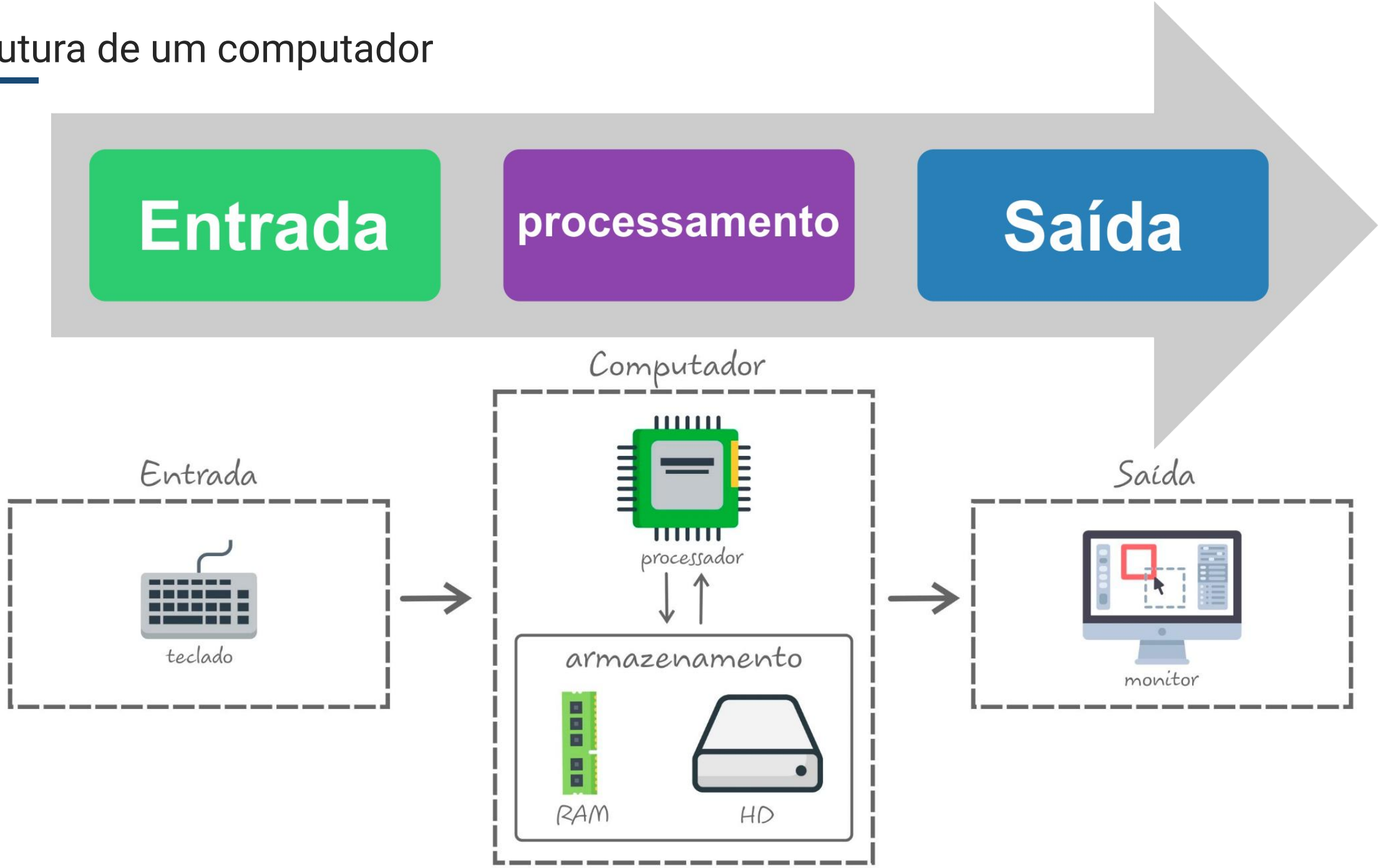
Fases do algoritmo



Estrutura de um computador



Estrutura de um computador



Javascript no HTML

Diretamente em um arquivo HTML
usando a tag **<script></script>**

- Entre as tags **<head>**
- Entre as tags **<body>**

Javascript no HTML

Em um arquivo separado

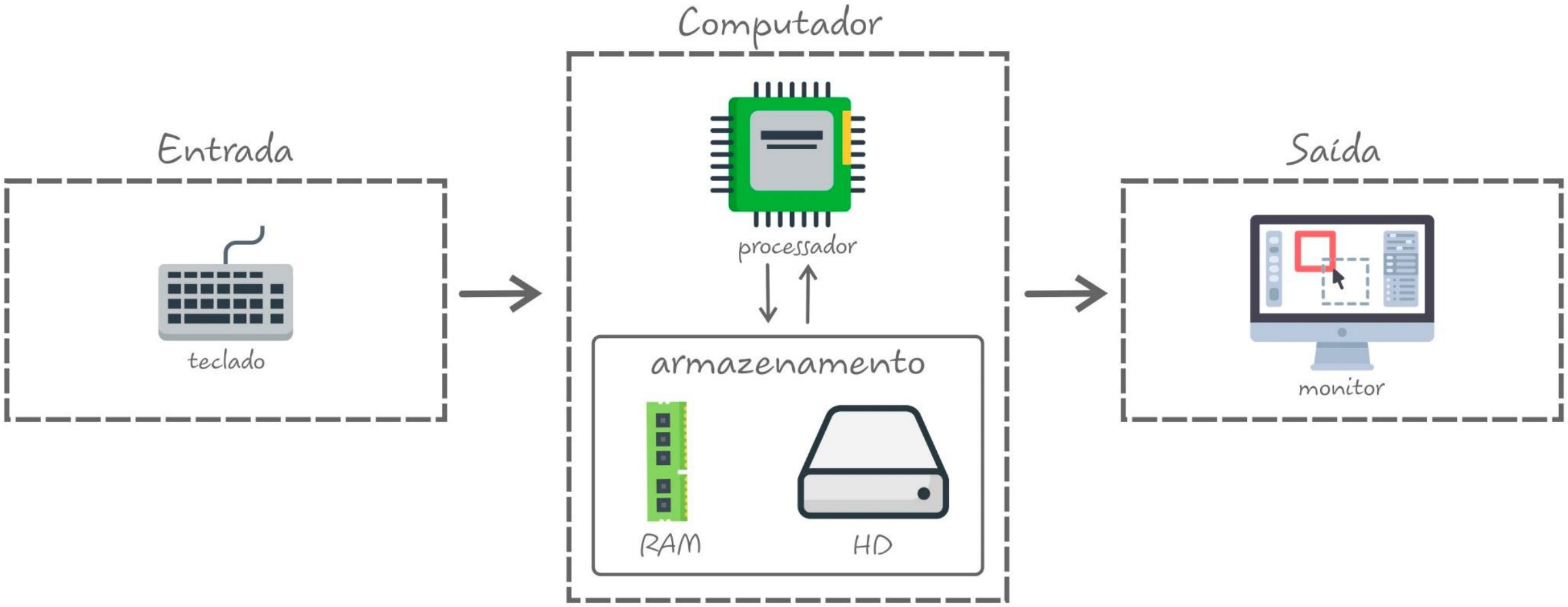
usando a tag **<script src=“ ” ></script>**

- Torna tudo muito mais sustentável e reutilizável.
- Facilita leitura e a manutenção do código;
- Diminui o tempo que as páginas carregam.

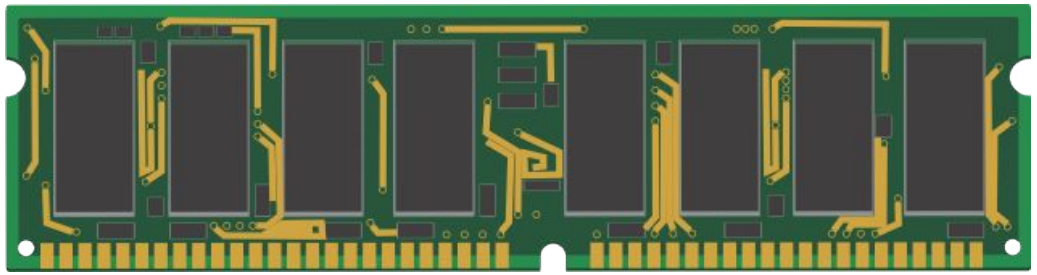
O que são variáveis ?

Posição da memória RAM onde podemos guardar algum dado

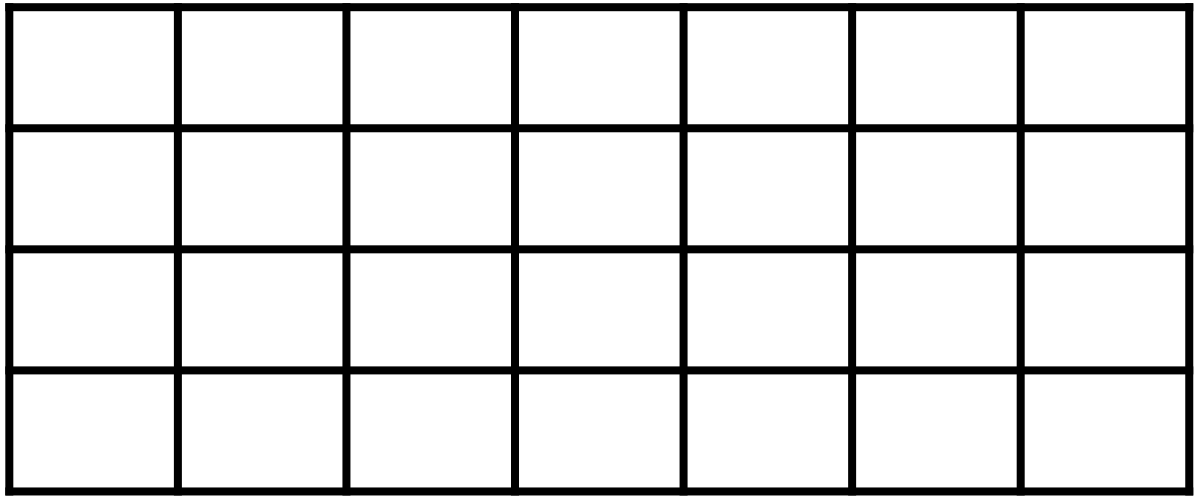
Estrutura de um computador



Variáveis – Alocação de espaço na memória

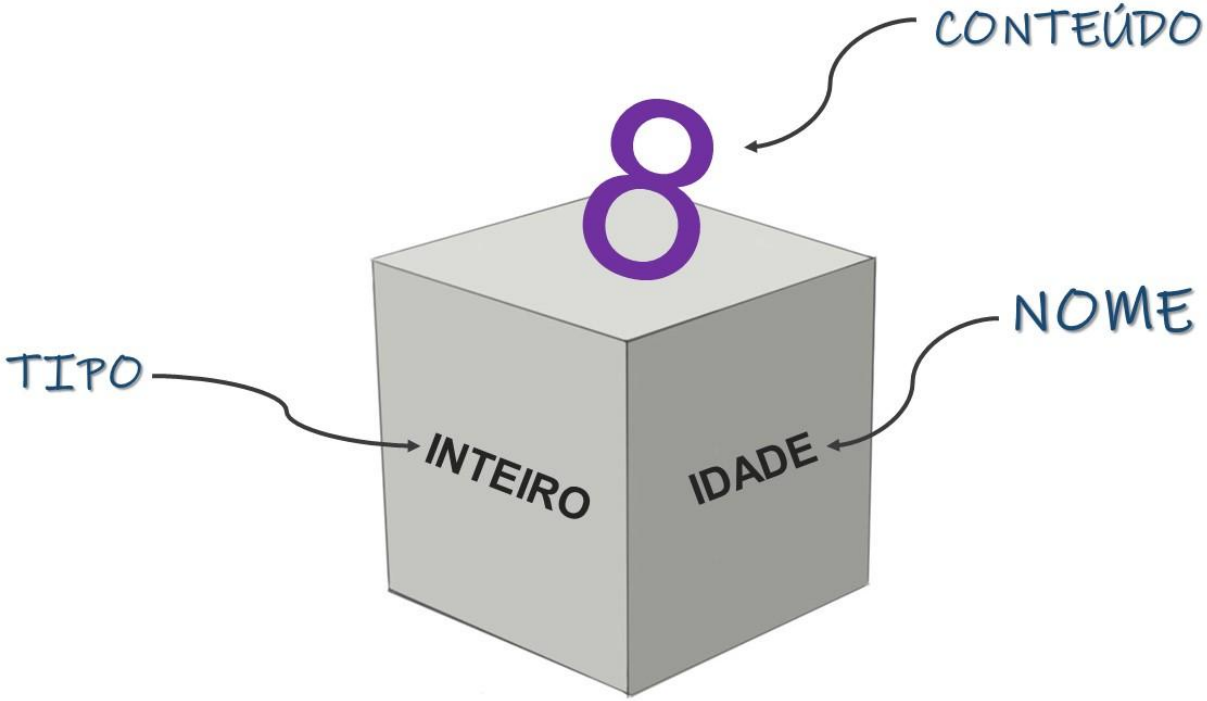


memoria



Variável

- Analogia: uma caixa, na qual você pode dar o nome que quiser, e guardar o conteúdo que desejar;
- Possui um tipo (caracter, cadeia, lógico, inteiro ou real);
- O valor dentro da “caixa” pode ser alterado de acordo com a execução do algoritmo.



Tipos de Variáveis

String

Numérica

Lógica

Ex:

```
String:
“Abc”
"3Way Networks”
‘A’
‘3’
‘@’
```

30
1000

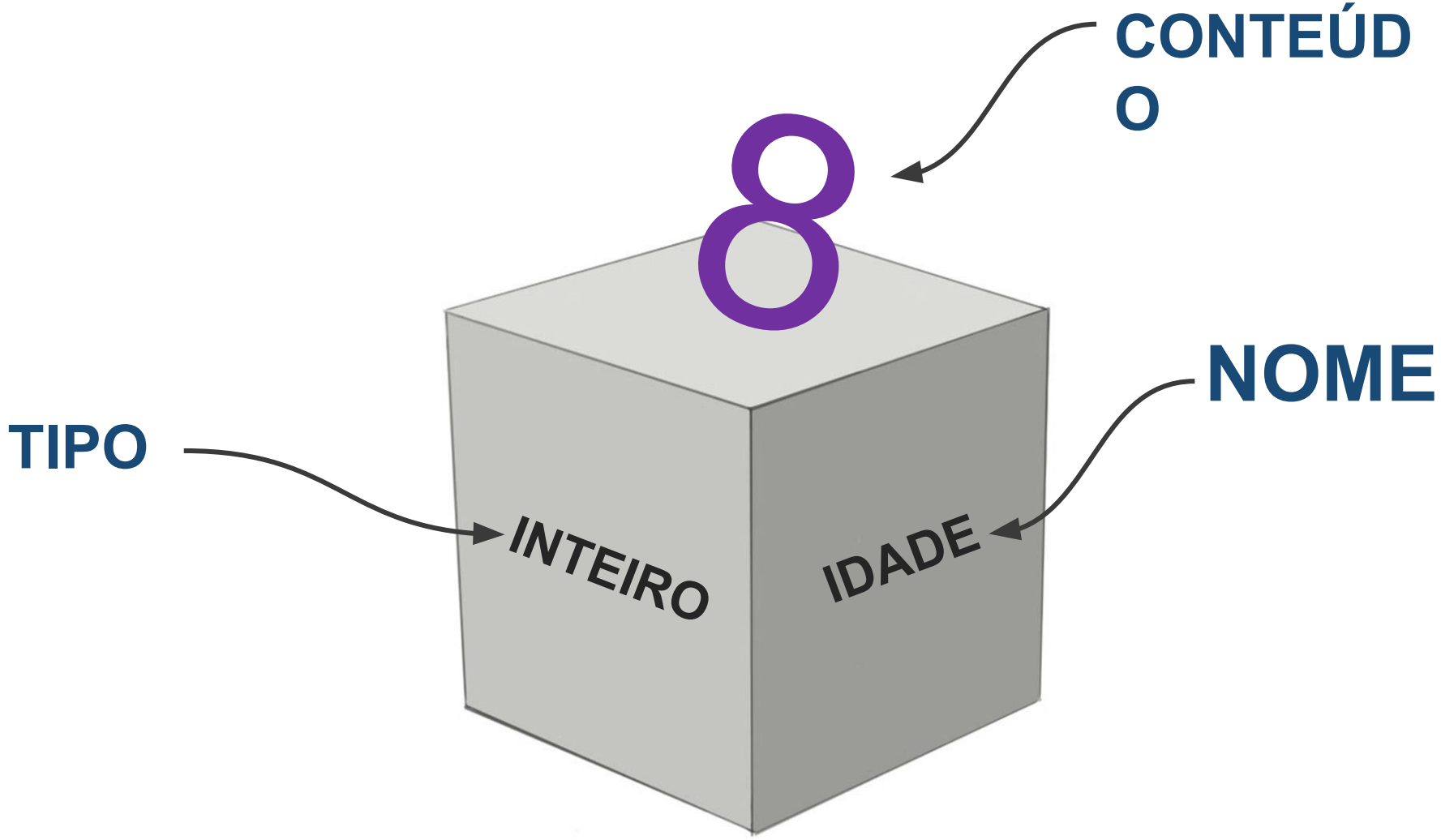
32,20
343,50...

- 30
- 1000

- 32,20
- 343,50...

```
true
false
```

Variáveis – Alocação de espaço na memória



Tipo de dados no Javascript

- **Strings**: Valores de texto alfanumérico, entre aspas (simples ou duplas).
- **Numéricos**: valores numéricos, inteiros e decimais;
- **Booleanos**: valores lógicos verdadeiro/falso (true/false);
- **Arrays**: Listas de valores, entre colchetes.
- **Objetos**: Coleções de pares chave/valor, entre chaves.
- **Null**: Um valor especial que indica a ausência de qualquer valor.
- **Undefined**: Um valor especial que indica que uma variável não recebeu um valor.

Declaração de Variáveis

Reservar uma área de memória para armazenar valores.

- **Variável Local** - Vai existir somente dentro de seu escopo.
- **Variável global** – Vai existir enquanto o programa estiver em execução.

var, let, const - algumas diferenças

	var	let	const
Reatribuição	✓	✓	✗
Redeclaração	✓	✗	✗
Hoisting	✓	✓	✓
Escopo global	✓	✗	✗
Escopo de bloco	✗	✓	✓
Escopo de função	✓	✓	✓

Atribuição de Variáveis

```
nome_da_variavel = valor
```

A atribuição é uma instrução utilizada para colocar um valor em uma variável para utilizá-lo em algum momento posterior.

Instruções `prompt` e `alert`

- A instrução ***prompt*** é usada quando se deseja obter algum dado do usuário. O algoritmo aguarda o usuário entrar com algum dado desejado. É uma instrução de **entrada de dados**.

- A instrução ***alert*** é usada quando se deseja apresentar algo na tela do computador. É uma instrução de **saída de dados**.

Como escrever os nomes das variáveis

• Podem começar com letras ou com os símbolos $ ou _. Elas só podem conter letras, números, $ e _. Elas não podem começar com um número;
• Diferenciam maiúsculas de minúsculas, ou seja, "xPos" é diferente de "xpos".São case sensitive;
• Não podem ser os mesmos de variáveis já existentes.
• Devem ser claros e significativos;
• Se usar mais de uma palavra para o nome da sua variável, procure iniciar as palavras (exceto a primeira) com letras maiúsculas, usando a nomenclatura **Camelcase**.

Operadores Aritméticos

Operador	Ação
**	Potenciação
*	Multiplicação
/	Divisão
+	Adição ou concatenar literal
-	Subtração ou inversor de sinal
%	Resto da divisão

Operadores Relacionais

Operador	Nome	Exemplo	Resultado
==	Igual	a == b	Verdadeiro se a for igual a b
!=	Diferente	a != b	Verdadeiro se a não for igual a b
===	Idêntico	a === b	Verdadeiro se a for igual a b e for do mesmo tipo
!==	Não idêntico	a !== b	Verdadeiro se a não for igual a b, ou eles não são do mesmo tipo
<	Menor que	a < b	Verdadeiro se a for menor que b
>	Maior que	a > b	Verdadeiro se a for maior que b
<=	Menor ou igual	a <= b	Verdadeiro se a for menor ou igual a b.
>=	Maior ou igual	a >= b	Verdadeiro se a for maior ou igual a b.

Operadores Lógicos

Operador	Ação
&&	E, AND
||	OU, OR
!	Negação
! !	Dupla negação

Desvios Condicionais

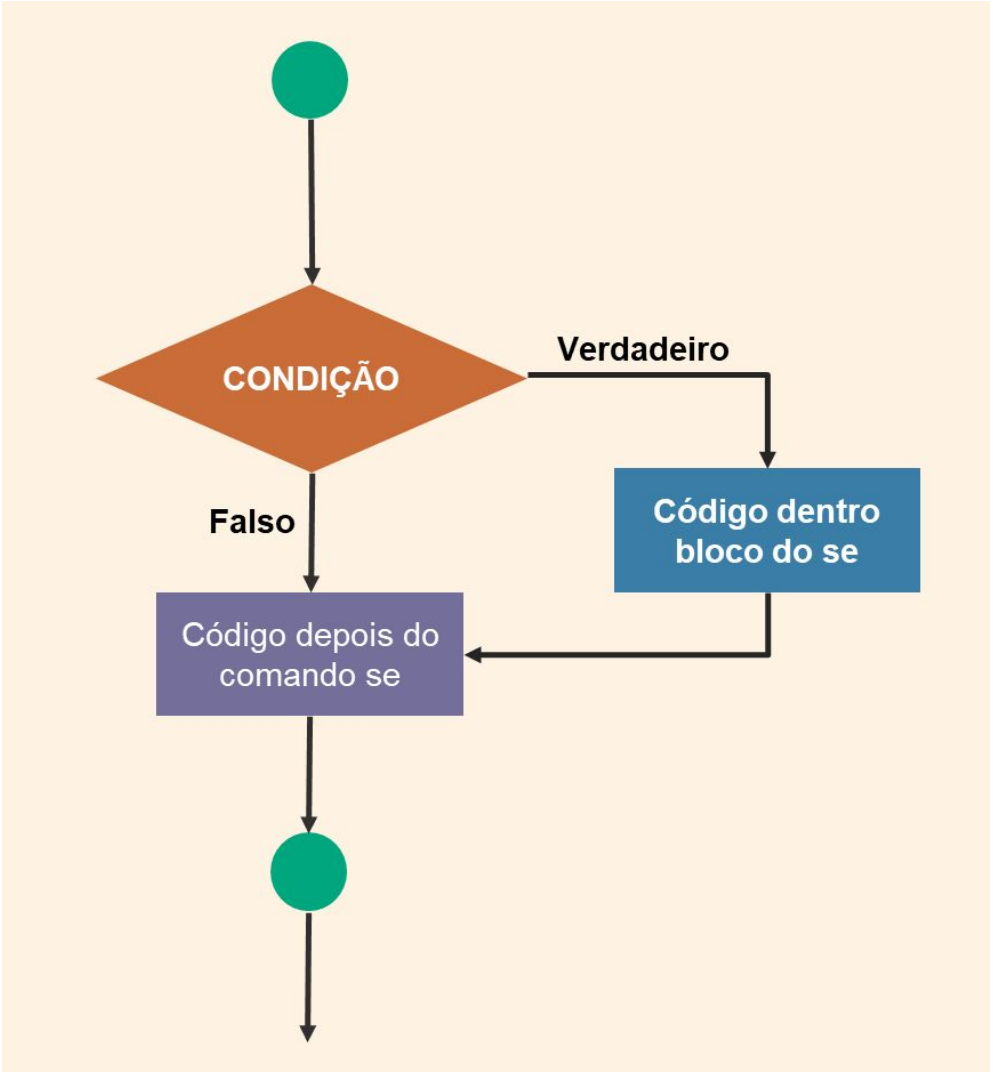
Desvios Condicionais – Comando if

```
var num = Number(prompt("Digite um
número"));

if (num === 0){

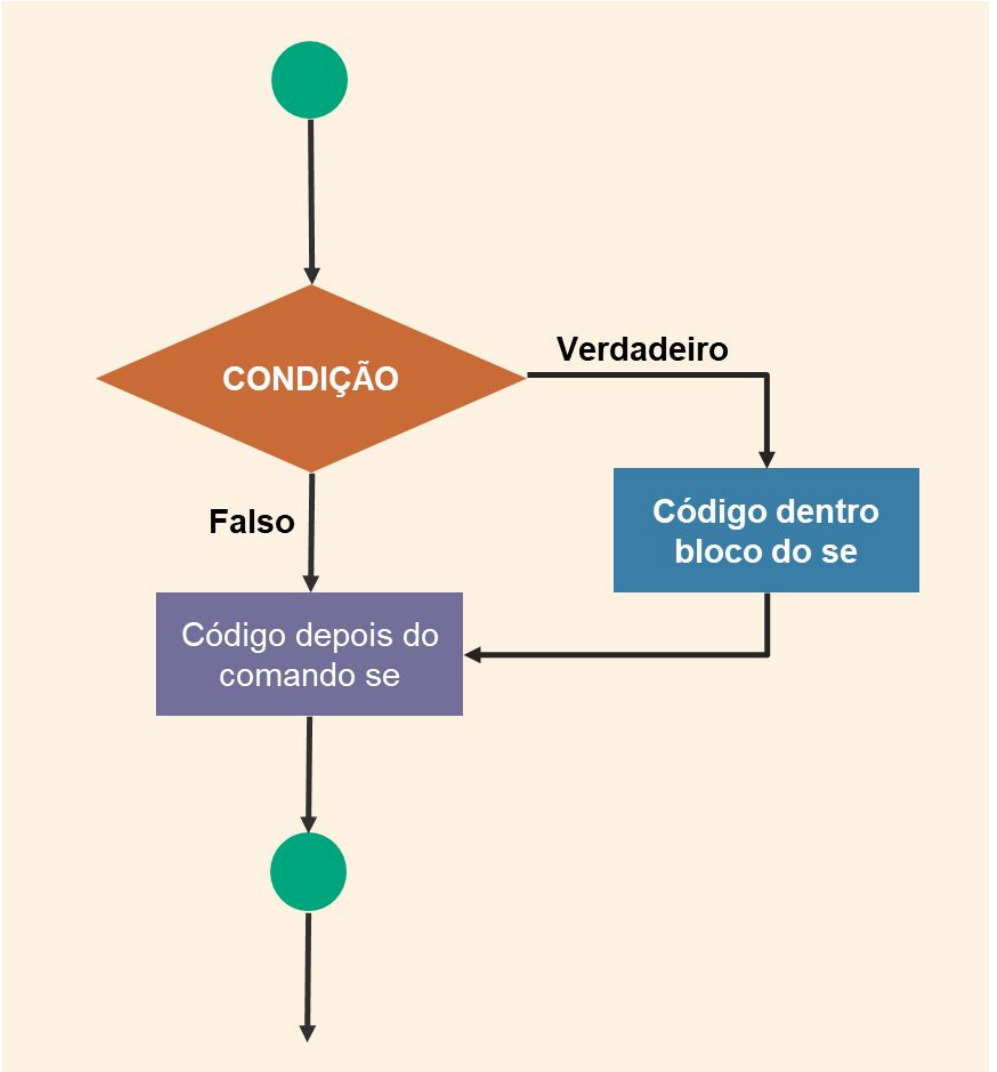
  alert("O número digitado é 0");

}
```



Desvios Condicionais – Comando se

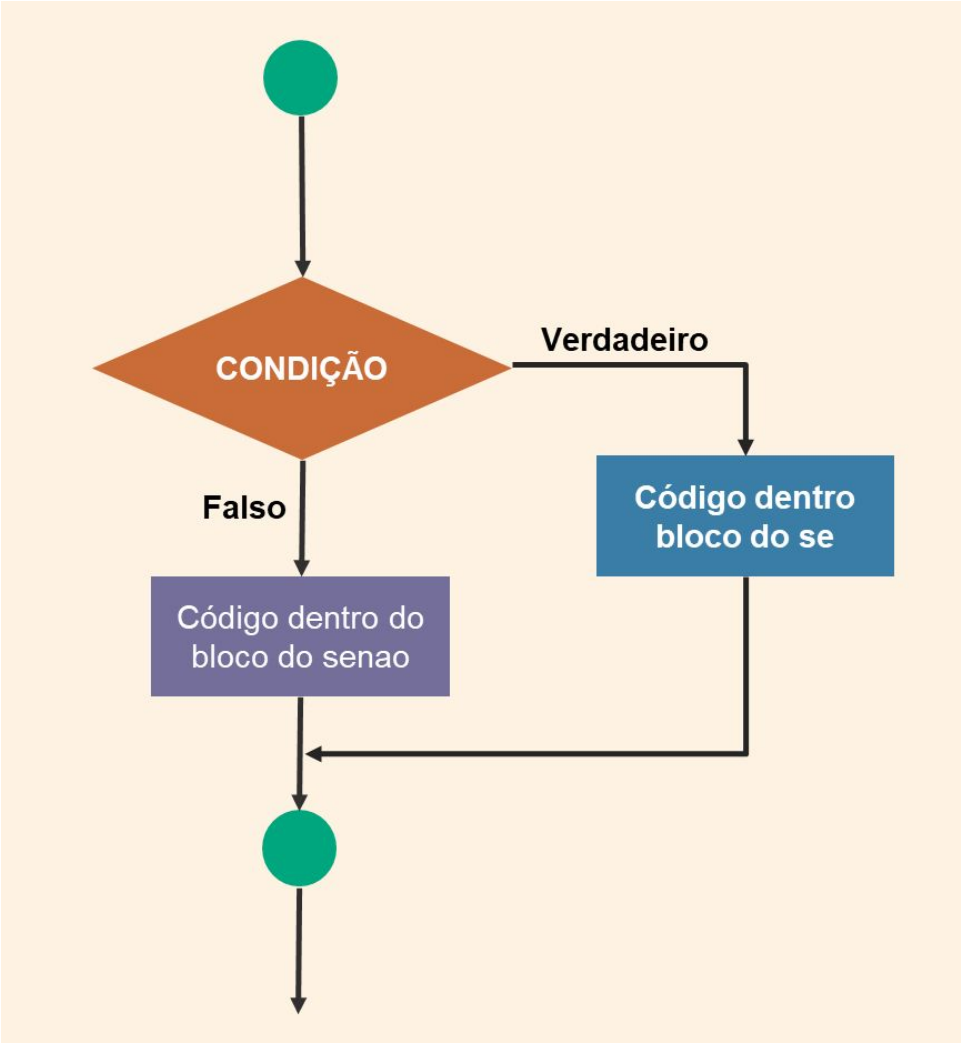
- Comando *if*
  - O comando condicional "***if***" executa um bloco de código caso uma condição seja verdade;
  - É comum a condição serem operadores relacionais.



Desvios Condicionais – Comando *if else*

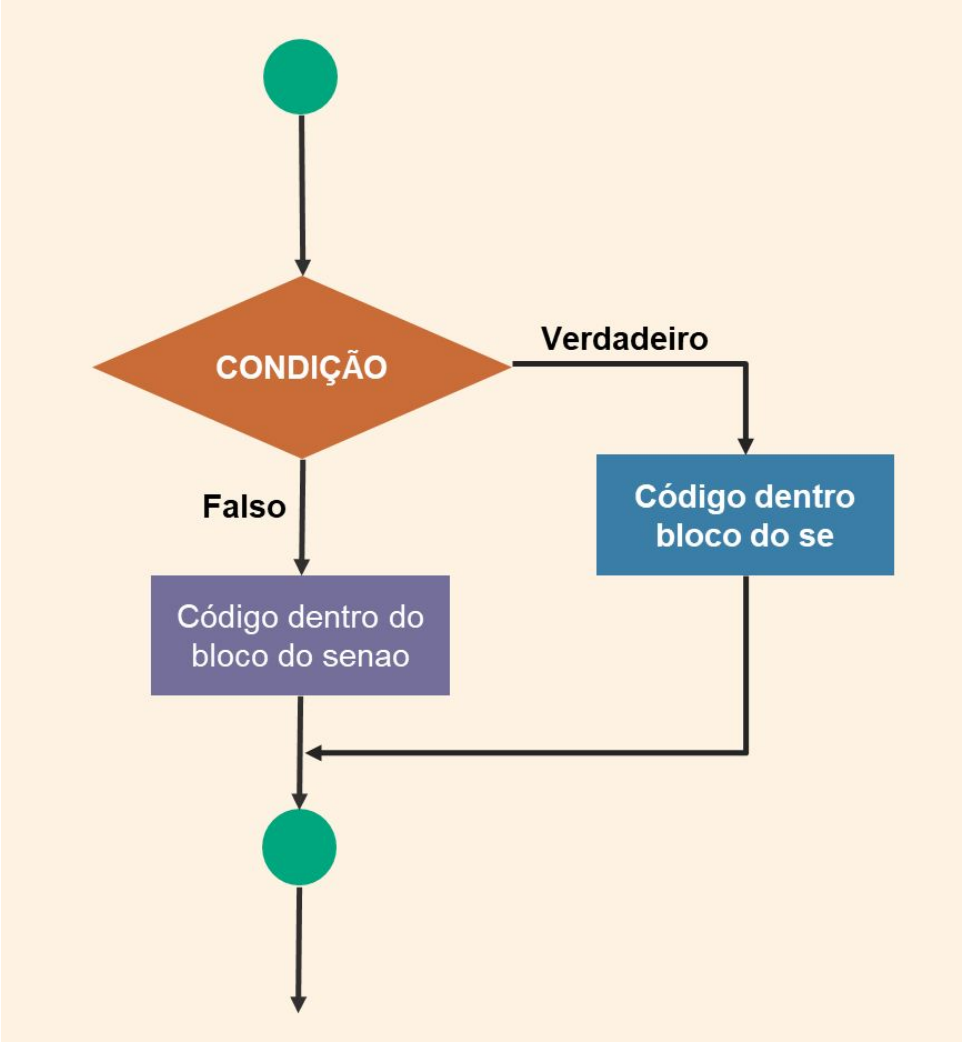
```
var hora = Number(prompt("Digite a
hora: "));

if (hora >= 6 && hora <= 18){
  alert("É dia");
} else {
  alert("É noite");
}
```



Desvios Condicionais – Comando *if else*

- Comando `else`
  - Podemos usar o comando `else` para executar um bloco de código quando a condição for falsa;
  - Cada `if` só pode ter um único `else`, e viceversa .



Desvios Condicionais – Comando *if else if*

```
if (12 < 5){

  //Instruções executadas se
condição for verdadeira

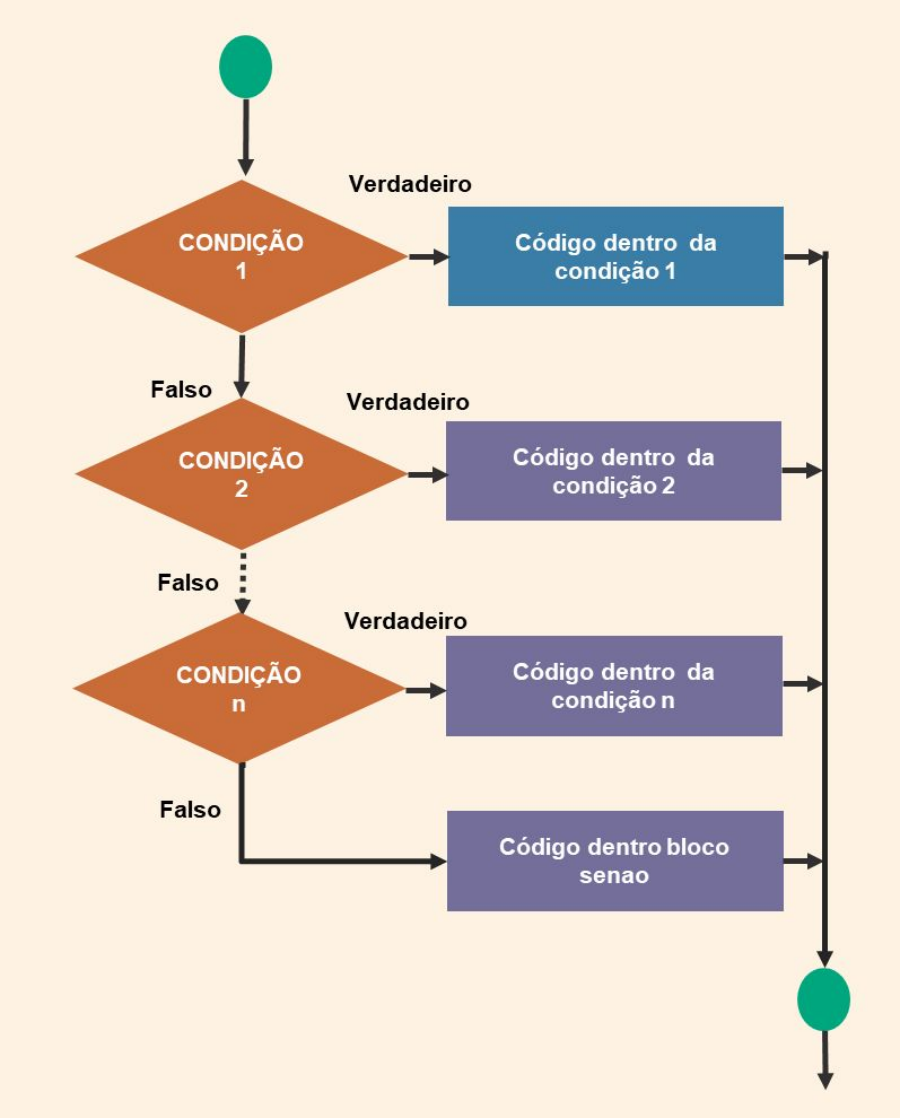
} else if ( ) {

  //Instruções executadas se
condição anterior for falsa e esta
for verdadeira

} else {

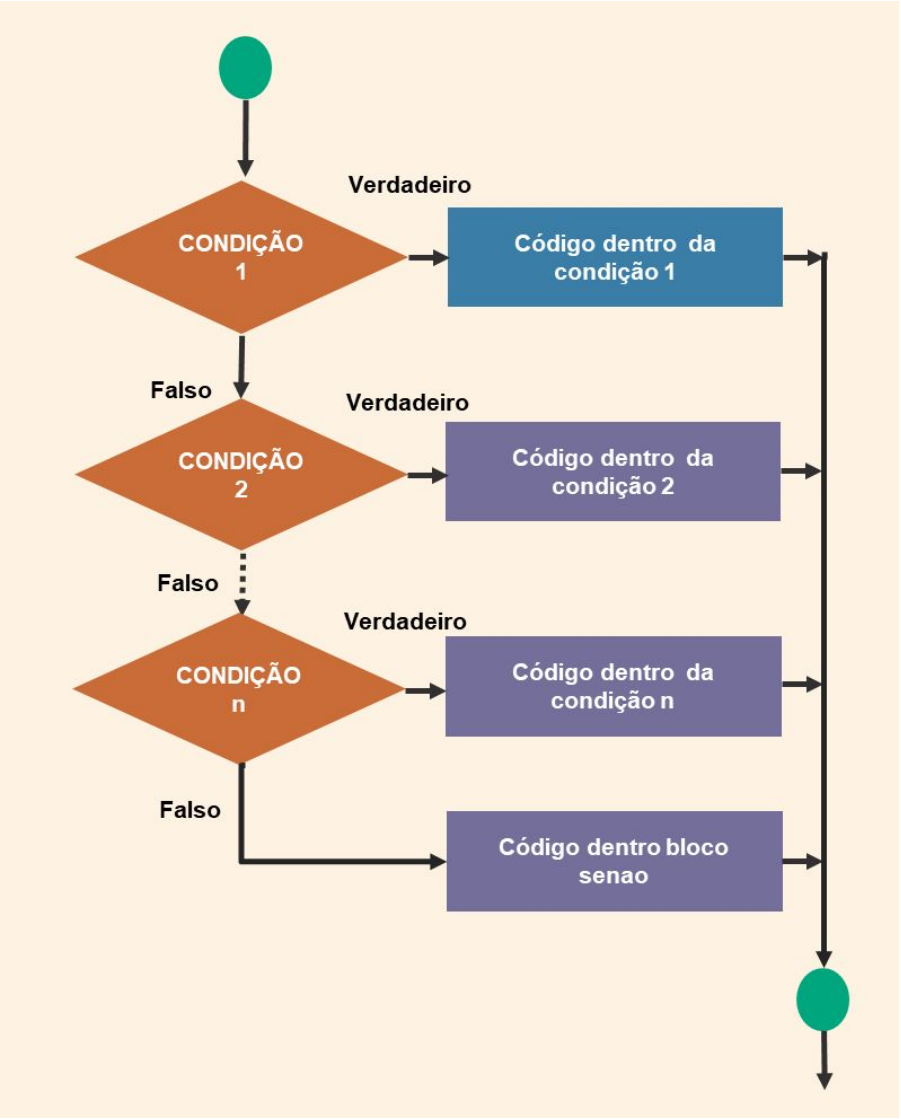
  //Instruções executadas se
condição anterior for falsa

}
```



Desvios Condicionais – Comando *if else if*

- Comando ***if else if***
  - Para verificar se uma condição é verdadeira, e se não for, verificar se outras condições são verdadeiras;
  - pode-se colocar o comando ***else*** no final do ultimo ***else if***.



Desvios Condicionais – Comando switch

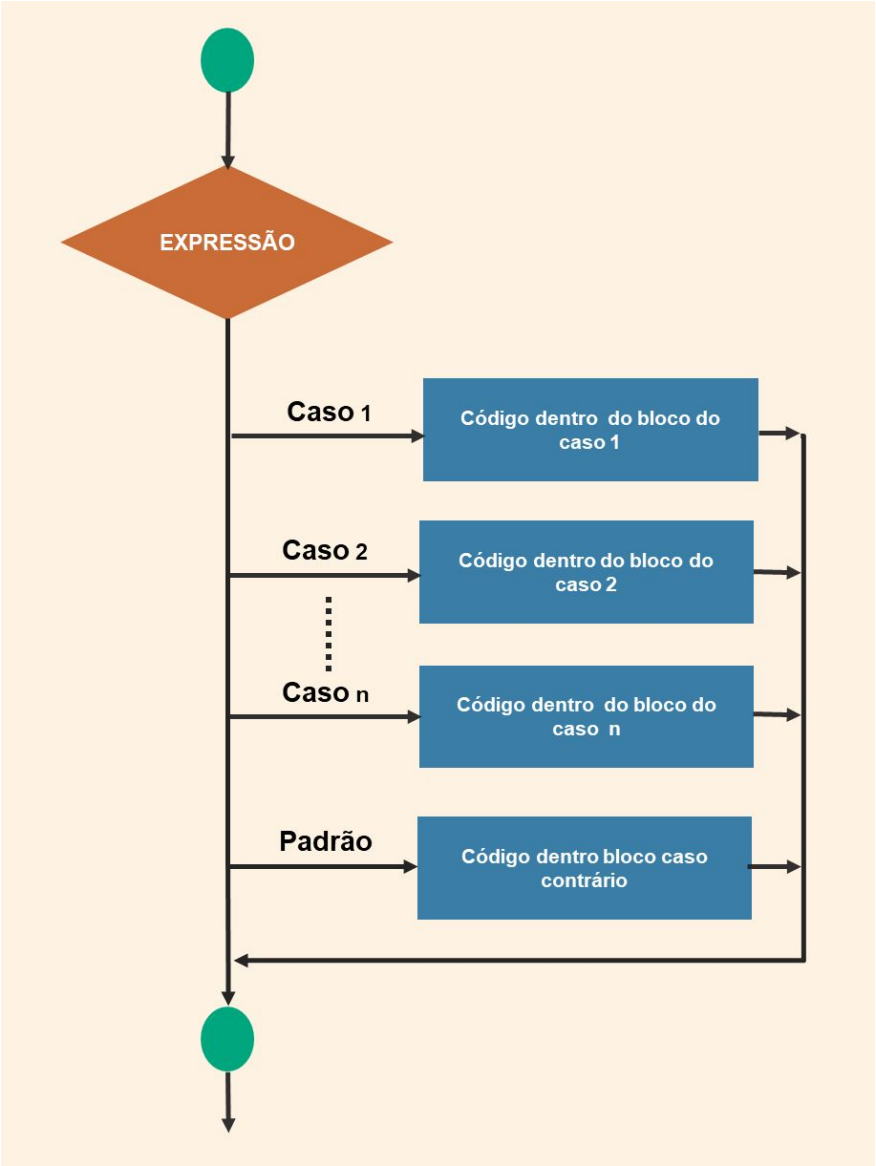
```
var valor = 1;

switch (valor){
  case 0:
    alert("O valor é igual a zero");
  break;

  case 1:
    alert("O valor é igual a um");
  break;

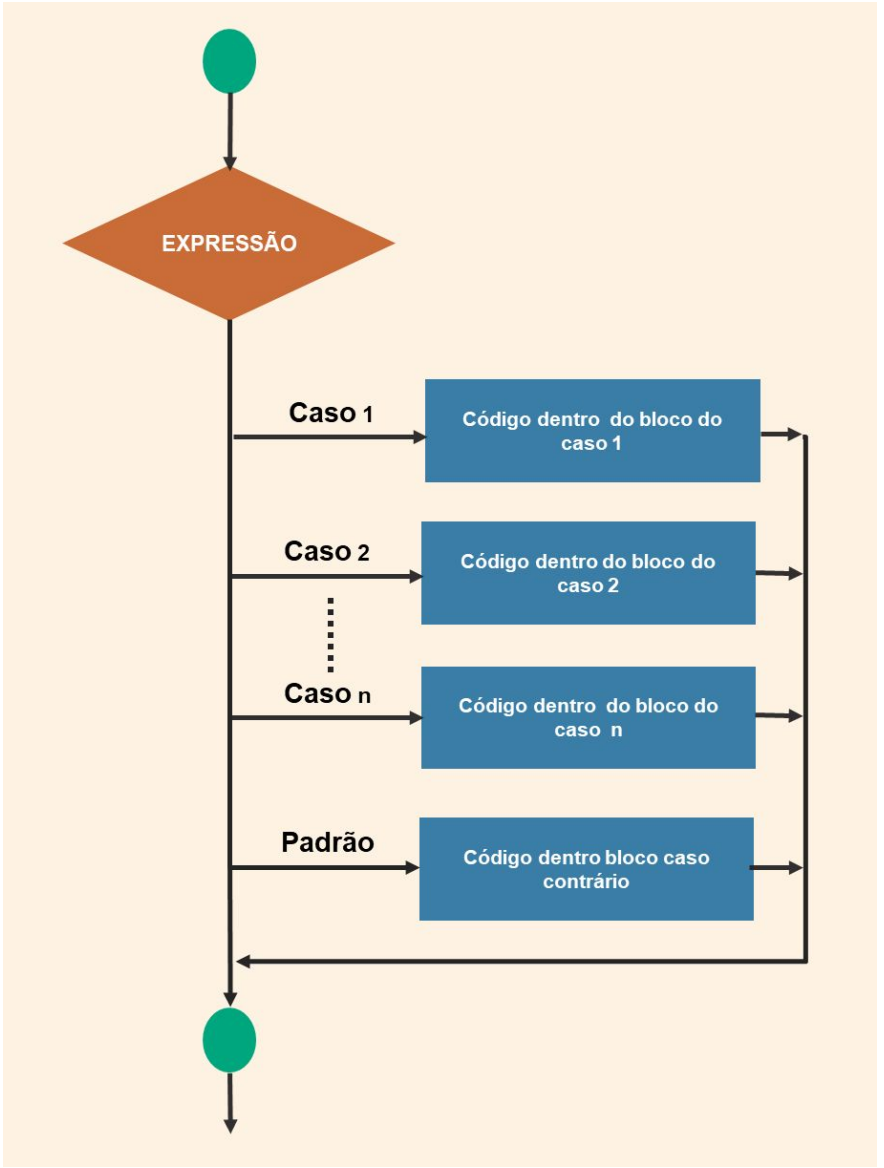
  case 2:
    alert("O valor é igual a dois");
  break;

  default:
    alert("O valor não é igual a 0, 1 ou 2");
}
```



Desvios Condicionais – Comando escolha caso

- Comando ***switch***
  - Permite ir direto no bloco de código desejado, dependendo do valor da variável de verificação;
  - Não é possível o uso de operadores lógicos. Trabalha com valores definidos, ou o valor é igual ou diferente.



Comandos de Repetição

Comandos de Repetição

- Usada quando se deseja que um trecho do algoritmo seja repetido várias vezes;

- A quantidade de repetições pode ser fixa ou depender de uma condição.

Comandos de Repetição

- As estruturas de repetição são classificadas em 3 tipos:
  - Repetição com teste no início do laço;
  - Repetição com teste no final do laço;
  - Repetição com variável de controle.

- São também chamadas de laços de repetição!

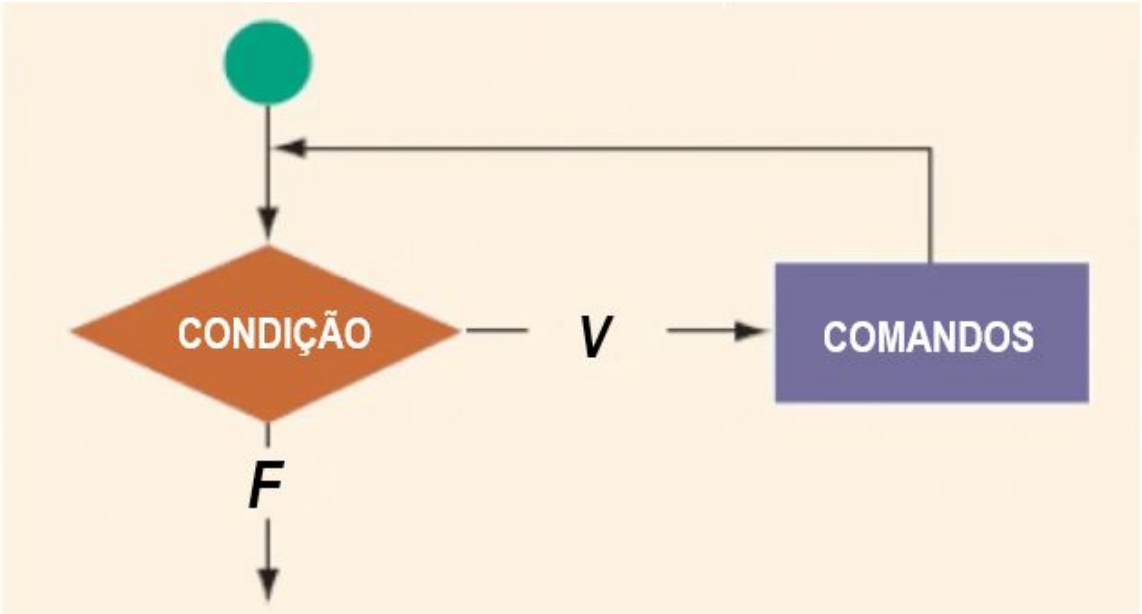
Comandos de Repetição – Repetição com Teste no Início

```
var parar = 'N';

while (parar !== 'S'){

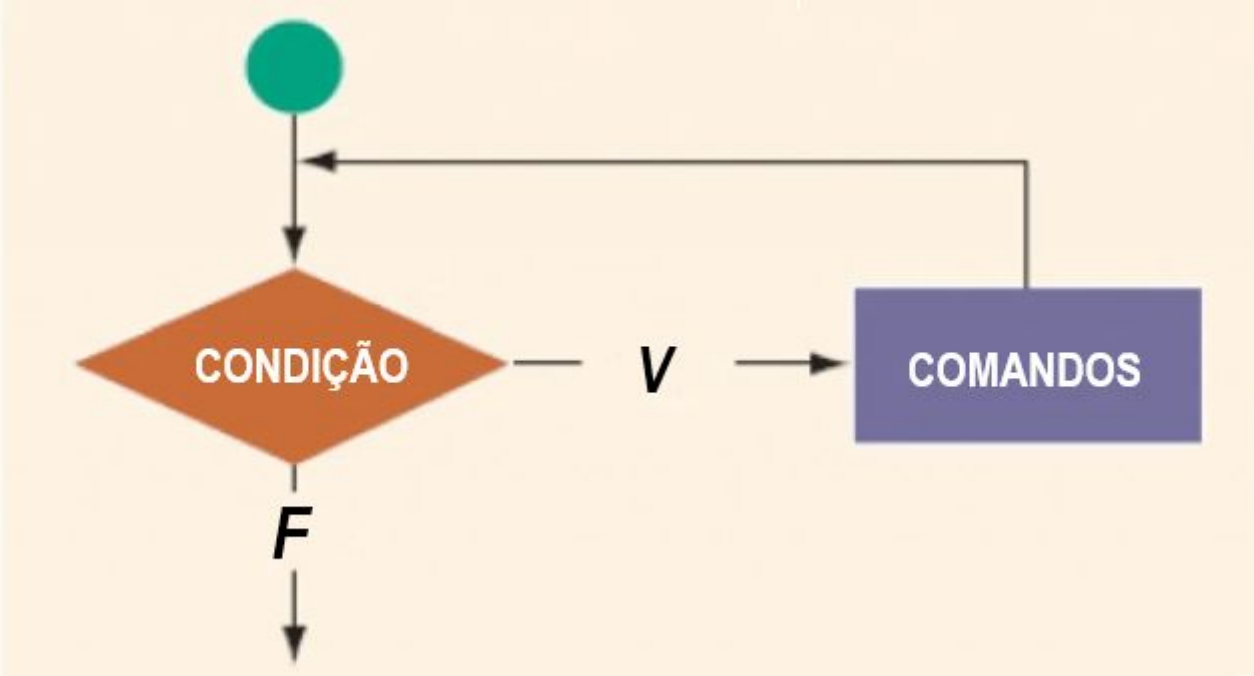
  parar = prompt("Deseja para o laço? (S/N)");

}
```



Comandos de Repetição – Repetição com Teste no Início

- Comando ***while***
  - Essa estrutura repete uma sequencia de comandos enquanto uma determinada condição for verdadeira;
  - Essa condição é determinada por uma expressão lógica;



Comandos de Repetição – Repetição com Teste no Final

```
var aresta, area;

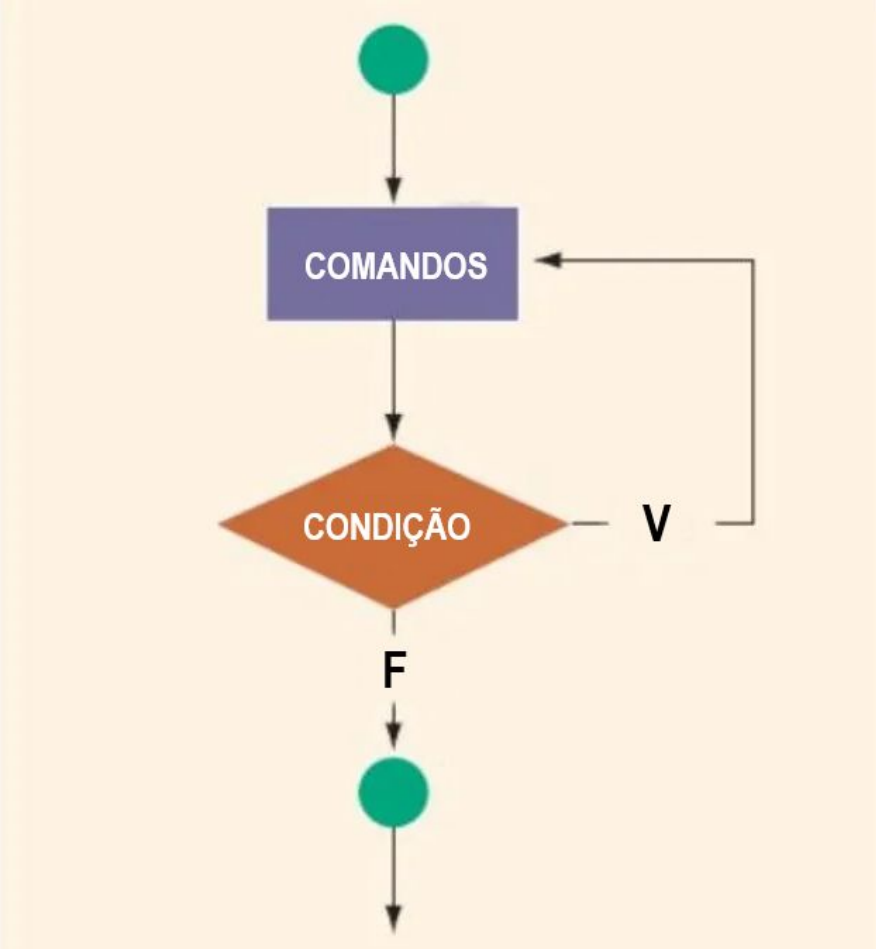
do {

  aresta = prompt("Escreva o valor da aresta");

} while (aresta <=0)

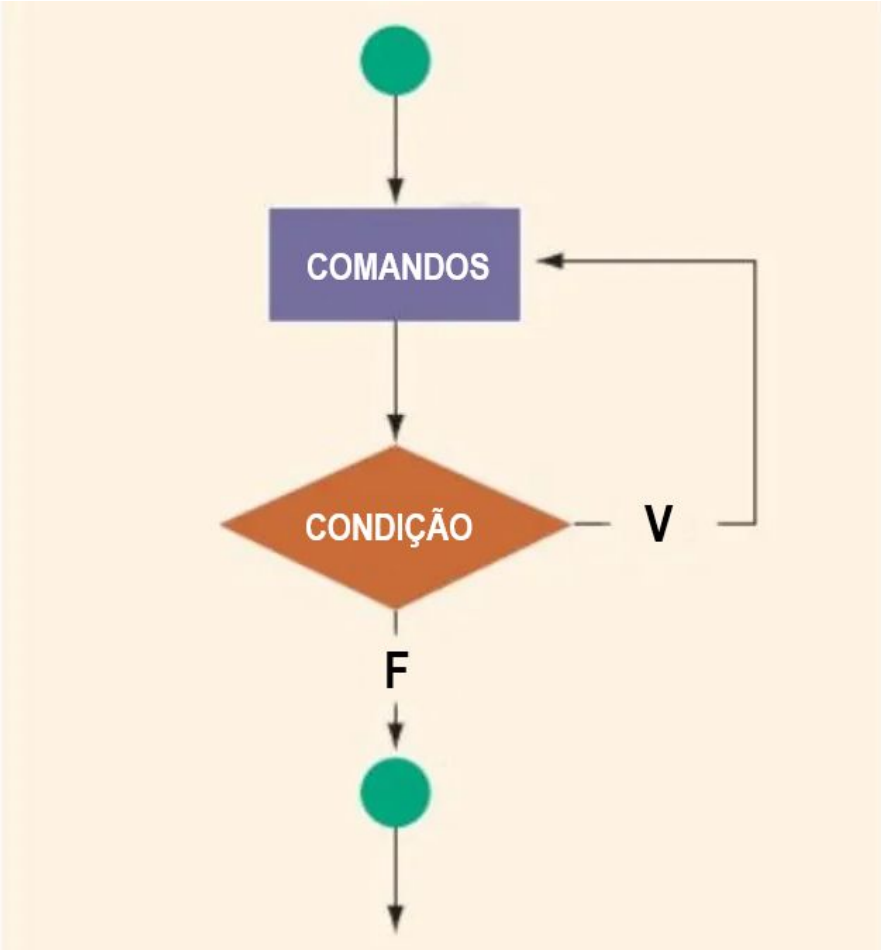
area = aresta * aresta;

alert ("A área da aresta é : " + area);
```



Comandos de Repetição – Repetição com Teste no Final

- Comando ***do-while***
  - É semelhante ao ***while***, só que a condição de teste fica no final;
  - A sequência de comandos é realizada no mínimo uma vez.



Comandos de Repetição – Repetição com Teste no Final

```
let continuar;

do {

//coloque aqui o código do programa

continuar = prompt("Deseja continuar S/N ?").toUpperCase();

} while ( continuar === "S")
```

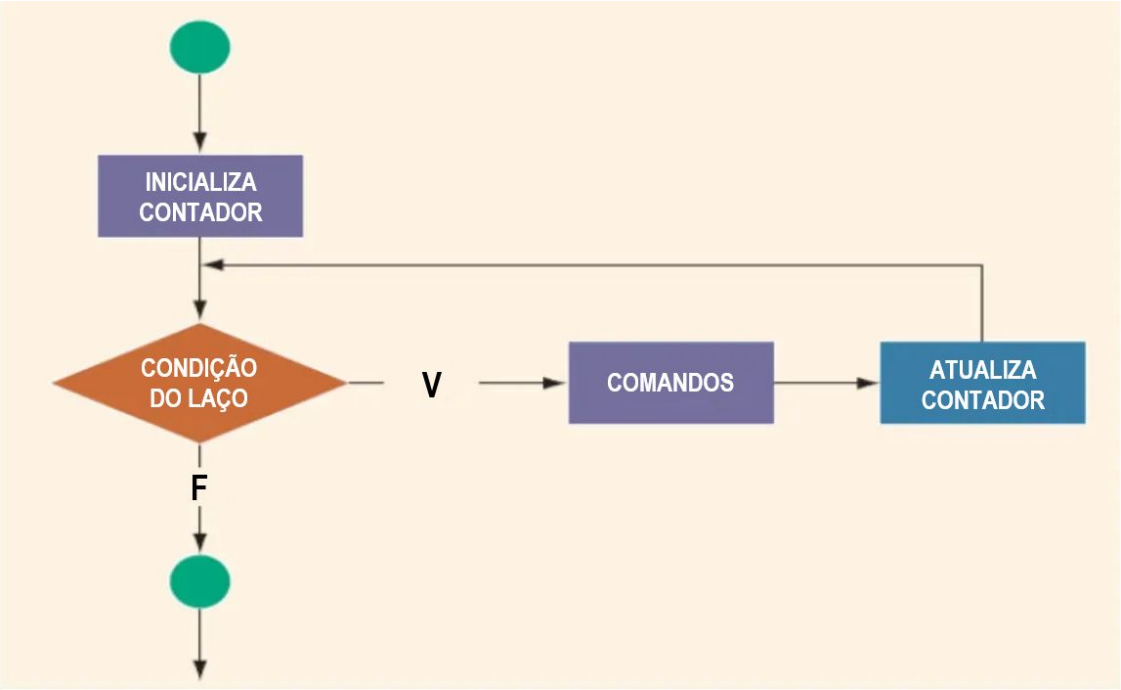
Comandos de Repetição – Repetição com Variável de Controle

```
var tab;

for (let c = 0; c <= 10; c++){

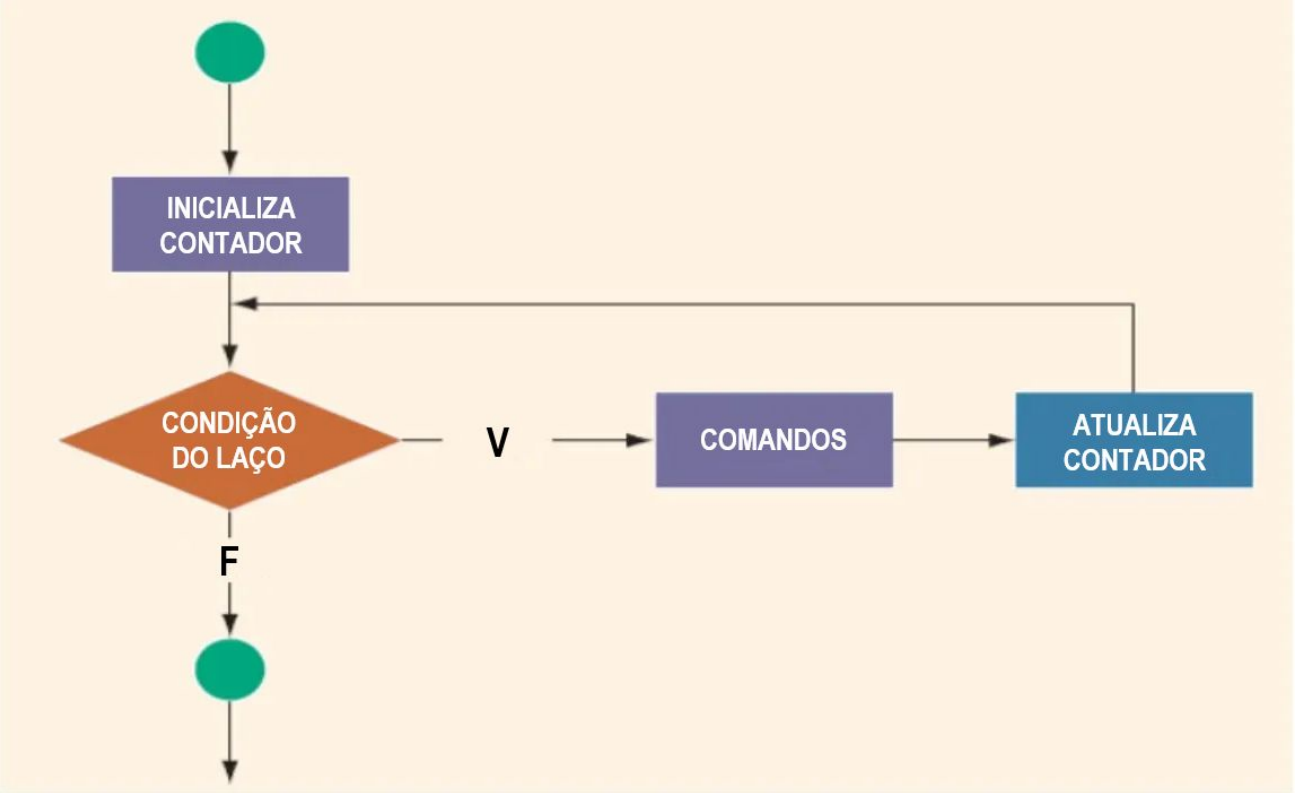
  tab = c * 3;
  alert("3 x " + c + " = " + tab);

}
```



Comandos de Repetição – Repetição com Variável de Controle

- Comando ***for***
  - Quando se conhece o número de vezes que um trecho do algoritmo deve ser repetido, deve-se usar uma estrutura com variável de controle;
  - Descreve a repetição de um número definido de vezes, fixando limites iniciais e finais para a variável de controle.

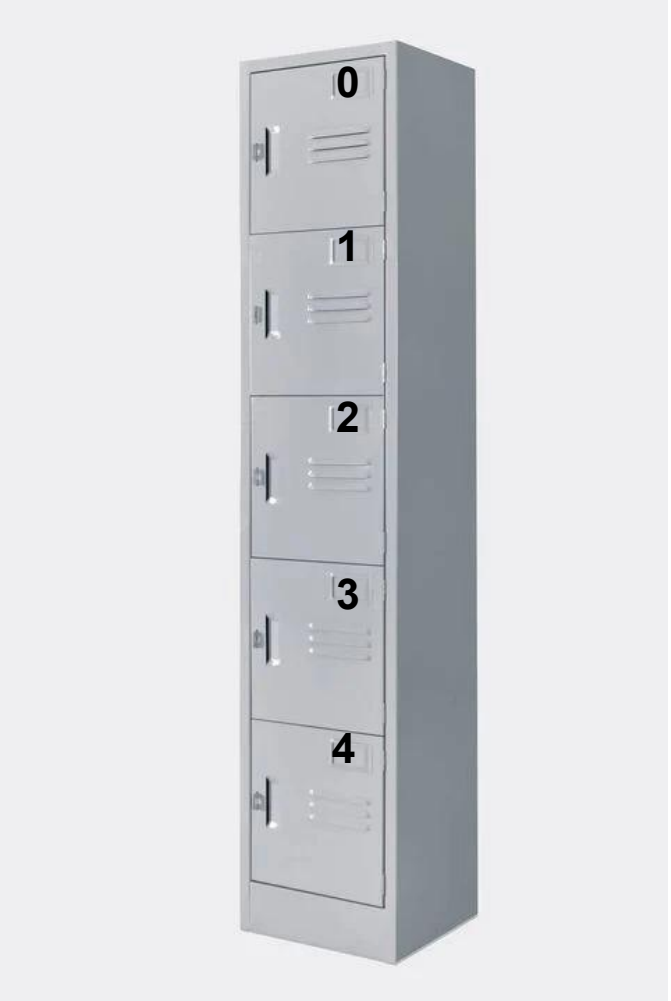


Arrays

Vetores (Arrays)

- O problema começa quando se precisa declarar várias variáveis para atender a um fim.

Vetores (Arrays)



Array

- Em casos como esse que é útil a utilização da estrutura de dados conhecida como array;

- Uma array é uma espécie de caixa com várias divisórias para armazenar coisas (dados).
  - É uma variável que pode armazenar vários valores.

Array

- Como criar uma array vazia:
  - `const arr = [ ];`

- Como criar uma array com valores:
  - `const numeros = [0, 3.14, 9.81, 37, 98.6, 100];`
  - `const frutas = ['banana', 'laranja', 'manga', 'limão'];`

- Imprimindo uma array e seu cumprimento:
  - `console.log('Números:', numeros);`
  - `console.log('Número de números', numeros.length);`

Array

- Uma array pode ter diferentes tipos:
  - ```
    const arr = [
        '3way Academy',
        250,
        true,
        { country: 'Finland', city: 'Helsinki' },
        { skills: ['HTML', 'CSS', 'JS', 'React',
    'Python'] }
    ]
    ```
    ```

Funções

Funções


palavra reservada
nome da função
parenteses
chaves
function name() {}

Funções

```
//declarando uma função sem parâmetro

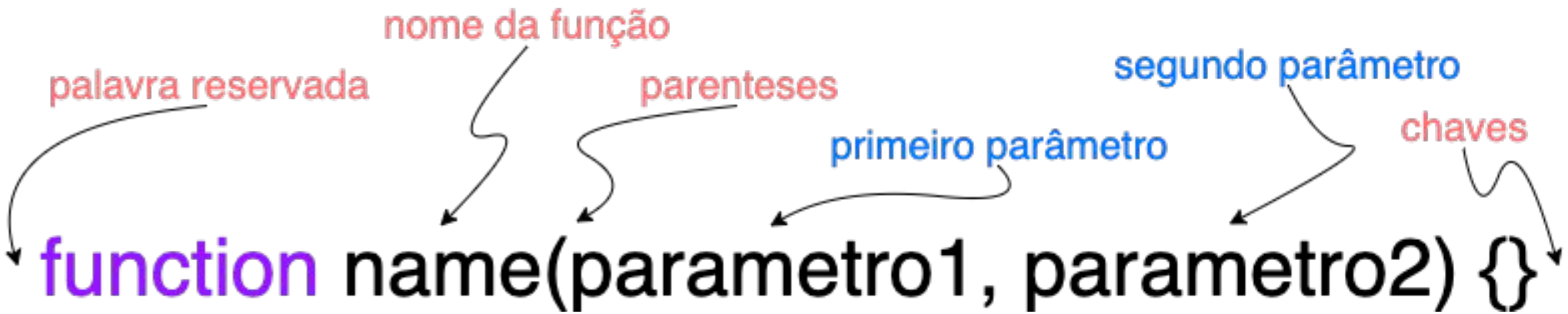
function functionName() {

  // o código vai aqui

}

functionName() // chamando uma função pelo seu nome com parêntese
```

Funções – Parâmetros



Funções – Parâmetros

```
//declarando uma função com parâmetros

function functionName(param1, param2, ...) {

  // o código vai aqui

}

functionName(arg1, arg2, ...) // chamando uma função pelo
seu nome com parêntese e argumentos
```

Funções – Retorno

```
//retorno em uma função

function functionName(param1, param2, ...) {

  // o código vai aqui

  return retorno

}

functionName(arg1, arg2, ...)
```

Funções – Função de expressão

```
// Função de expressão

const square = function(n) {

  return n * n

}

console.log(square(2))
```

Funções

- São trechos de algoritmos que efetuam um ou mais cálculos determinados;
- Ao invés de escrever um código grande, pode-se escrever vários algoritmos menores (Modularização);
- Em conjunto, resolvem o problema proposto;
- É conveniente utilizá-los quando uma tarefa é efetuada em diversos lugares no mesmo algoritmo;
- Ao invés de escrever um trecho diversas vezes, escreve-se uma função e que é chamada diversas vezes.

Funções

- Reduzem o tamanho do algoritmo;
- Facilitam a compreensão e visualização do algoritmo;
- Podem :
  - Retornar algum valor;
  - Não retornar nada.
- Uma função pode possui 0, 1 ou mais parâmetros

Criando Funções

- **Variáveis Locais**
  - Declaradas dentro da função;
  - Podem ser usadas APENAS dentro das funções;
  - O algoritmo que chamou a função não tem acesso a estas variáveis.
- **Variáveis Globais**
  - São variáveis declaradas no escopo do programa;
  - Qualquer função pode alterar o valor ou utilizá-la durante o seu processamento.