Algoritmia e Programação

Estruturas de Controlo Condicional

Marta Martinho 2025

Organização da apresentação

- Operadores relacionais;
- Operadores lógicos;
- Estruturas de controlo condicional;
    - Decisão (se - if)
    - Seleção (seleciononar - switch)

Operadores relacionais

- Comparam valores e devolvem sempre um valor lógico (true ou false);
- Usados em condições de if, ciclos, etc.

Operador	Exemplo	Significado
>	a > b	verdadeiro se a for maior que b
<	a < b	verdadeiro se a for menor que b
>=	a >= b	verdadeiro se a for maior ou igual a b
<=	a <= b	verdadeiro se a for menor ou igual a b
.==	a == b	comparação com conversão de tipos (evitar)
.===	a === b	comparação estrita (mesmo valor e mesmo tipo)
!=	a != b	diferente, com conversão de tipos
!==	a !== b	diferente, comparação estrita

== vs === — comparação fraca vs comparação estrita

- == → converte os tipos antes de comparar (evitar!)Usados em condições de if, ciclos, etc.
- === → não converte tipos (recomendado)

```
"3" == 3; // true
false == 0; // true
```

```
"3" === 3   // false
false === 0 // false
```

Exercício

Considera:

let a = 3, b = 7, c = 4;

Calcula o resultado (verdadeiro/falso) das expressões:

1. a + b) > b
2. b >= (a + 2)
3. c === b - a
4. (b + a) <= c
5. (c + a) > b

```
let a = 3;
let b = 7;
let c = 4;

console.log("(a + b) > b =", a + b > b);
console.log("b >= (a + 2) =", b >= a + 2);
console.log("c === b - a =", c === b - a);
console.log("(b + a) <= c =", b + a <= c);
console.log("(c + a) > b =", c + a > b);
```

Depois, implementa um pequeno programa em JavaScript que calcule e mostre estes resultados na consola.

Exemplo

Algoritmo “Maior de idade”

- Lê o nome e a idade de uma pessoa;
- Considera a maioridade aos 18 anos;
- Determina se a pessoa é maior de idade;
- Escreve o resultado.

```
algoritmo "VerificarMaioridade"
// Função : Verificar maioridade
// Autor : Célio Carvalho
// Versão 1.0
var
   nome : caracter
   idade, maioridade : inteiro
   maior : logico

inicio

   // atribuir valores
   nome <- "manuel"
   idade <- 19
   maioridade <- 18

   // comparar a idade com a maioridade e atribuir o
   // resultado logico à variavel maior
   maior <- idade >= maioridade

   // escrever na consola o valor de maior
   escreval("o ", nome, " é maior de idade?: ", maior)

fimalgoritmo
```

Exemplo

Algoritmo “Maior de idade”

- Lê o nome e a idade de uma pessoa;
- Considera a maioridade aos 18 anos;
- Determina se a pessoa é maior de idade;
- Escreve o resultado.

```
const prompt = require("prompt-sync")();

// Função: verifica se uma pessoa é maior de idade
// Autor: John Doe

const maioridade = 18;
const nome = prompt("NOME: ");
const idade = Number(prompt("IDADE: "));

// comparar a idade com a maioridade e atribuir
// o resultado lógico à variável maior
maior = idade >= maioridade;

// mostrar o resultado no ecrã
console.log(`O ${nome} é maior de idade?: ${maior}`);

// simular o getchar() no browser
prompt("Prima ENTER para terminar...");
```

Exercício

Implemente um algoritmo que leia o salário mensal de um trabalhador e determine se ele tem direito a receber o apoio estatal para compensar o aumento dos preços.

Especificações:

Entrada:

- Salário (real positivo)

Processamento:

- Considera que o apoio é atribuído apenas se o salário for menor ou igual a 1000 €.
- Utilize um operador relacional (<=) para verificar essa condição.

Saída:

- Uma mensagem a indicar se o trabalhador tem direito ao apoio ou não tem direito ao apoio.

Operadores lógicos

Algoritmos	Javascript	Notas
E	**& / &&**	Exemplo: (A > B) & (C < D) Se ambos os operandos forem verdadeiros, ou seja, se A for maior que B e C menor que D, a expressão no seu todo é avaliada como verdadeira. Quando usado o operador lógico &, ambos os operandos são sempre avaliados. Se for usado o && e se o primeiro operando resultar em falso, o segundo já não é avaliado já que, qualquer que seja o seu resultado, a expressão completa será avaliada sempre como falsa.
OU	**I / II**	Exemplo: (A > B) I (C < D) Basta que um dos operandos seja verdadeiro para que a expressão no seu todo seja avaliada como verdadeira.
NÃO	**!**	~ (A > B) / ! Se A maior que B então o resultado é falso já que A > B é verdadeiro e a negação de verdadeiro é falso. Qualquer outro caso é verdadeiro .

Operadores lógicos

- Com **&&**: se o primeiro operando for **false** — o segundo não é avaliado.
- Com **||:** se o primeiro operando for **true** — o segundo não é avaliado.
- Em JavaScript, “**true**” e “**false**” não se referem apenas a booleanos:
  - valores como 0, "", null, undefined, NaN são considerados **false**,
  - outros (strings não vazias, números diferentes de 0, objetos, arrays, etc.) são **true**.

Short-circuit — && e || têm comportamento especial

- **Operador AND (&&)**
  - Se a parte esquerda for **false** → a direita não é avaliada.

```
false && console.log("isto não acontece");
```

- **Operador OR (||)**
  - Se a parte esquerda for **true** → a direita não é avaliada.

```
"Olá" || console.log("isto não acontece");
```

- Em JavaScript, estes operadores devolvem valores, não apenas booleanos → muito usado em programação moderna.

```
let nome = prompt("NOME: ") || "Anónimo"; // valor por defeito
```

```
let opcao = prompt("sim || nao: ").trim().toLowerCase() === "sim"; // true se for sim
```

Tabelas de verdade

a	*b*	*a E b*	*a OU b*	*NAO a*	*NAO b*
V	*V*	*V*	*V*	*F*	*F*
V	*F*	*F*	*V*	*F*	*V*
F	*V*	*F*	*V*	*V*	*F*
F	*F*	*F*	*F*	*V*	*V*

Prioridade	Operador
1	Aritmético
2	Relacional
3	Não
4	E
5	OU

Exercício

Determine o resultado das seguintes expressões, considerando as variáveis do tipo lógico e as respetivas atribuições.

- a = verdadeiro; b = falso; c = falso; d = verdadeiro;
  - a OU (b E c) OU (NAO d OU NAO a);
  - NAO(NAO a OU b) E (c OU d);
  - b E NAO c OU a E d;
  - NAO a E NAO NAO c E d.

De seguida desenvolva um programa em javascript que permita a confirmação dos resultados obtidos.

Exemplo

Condições para conseguir conduzir:

- Tem carta de condução;
- Idade mínima de 22 anos;
- Altura mínima de 1,5 m (para chegar aos pedais).

```
algoritmo "ConseguirConduzir"
// Função : Quem consegue conduzir?
// Autor : Célio Carvalho
// Versão : 1.0
var
   // Declarar variáveis e atribuir valores exemplo
   nome : caracter
   idade : inteiro
   altura : real
   cartaConducao, consegueConduzir : logico

inicio

   // atribuir valores
   nome <- "antónio"
   idade <- 22
   altura <- 1.65
   cartaConducao <- falso

   // Basta que uma pessoa tenha carta de condução ou que tenha no minimo
   // 22 anos que se assume conseguir conduzir. No entanto, nunca deve ter
   // altura inferior a 1.5m senão, não chega aos pedais!
   consegueConduzir <- (altura >= 1.5) E ((idade >= 22) OU (cartaConducao = verdadeiro))

   // escrever o resultado da avaliação
   escreval("o ",nome," consegue conduzir?: ", consegueConduzir)

fimalgoritmo
```

Exemplo

Condições para conseguir conduzir:

- Tem carta de condução;
- Idade mínima de 22 anos;
- Altura mínima de 1,5 m (para chegar aos pedais).

```
const prompt = require("prompt-sync")();

// Declarar variáveis e pedir valores ao utilizador
const nome = prompt("Nome:");
const idade = Number(prompt("Idade:"));
const altura = Number(prompt("Altura (m):"));
// true se respondeu "s"
const cartaConducao = prompt("Tem carta de condução? (s/n)") === "s";

/*Condições para conseguir conduzir
  - Deve ter altura igual ou superior a 1.5m  → chega aos pedais
  - Deve ter pelo menos 22 anos  OU  ter carta de condução */
const consegueConduzir = altura >= 1.5 && (idade >= 22 || cartaConducao);

// escrever o resultado
console.log(`${nome} consegue conduzir? ${consegueConduzir}`);

prompt("Prima ENTER para terminar...");
```

Exercício

Implemente um programa que determine se é possível realizar um piquenique com base em três condições introduzidas pelo utilizador.

Dados de entrada:

Está a chover?	Dia da semana:	Distância de deslocação (km):
(Introduza ’s’ se chove, ou ’n’ se não chove)	(Número inteiro de 1 a 7, onde 1 representa segunda-feira e 7 representa domingo)	(Número real positivo)

Considerações adicionais:	
O custo do combustível é 0,50 € por km.	O limite máximo de despesa em combustível é 5 €.

Condições para ser possível fazer o piquenique:		
Não pode estar a chover (chove == “n”)	Tem que ser sábado ou domingo	O custo da deslocação tem que ser ≤ 5 €

Exemplo de output:

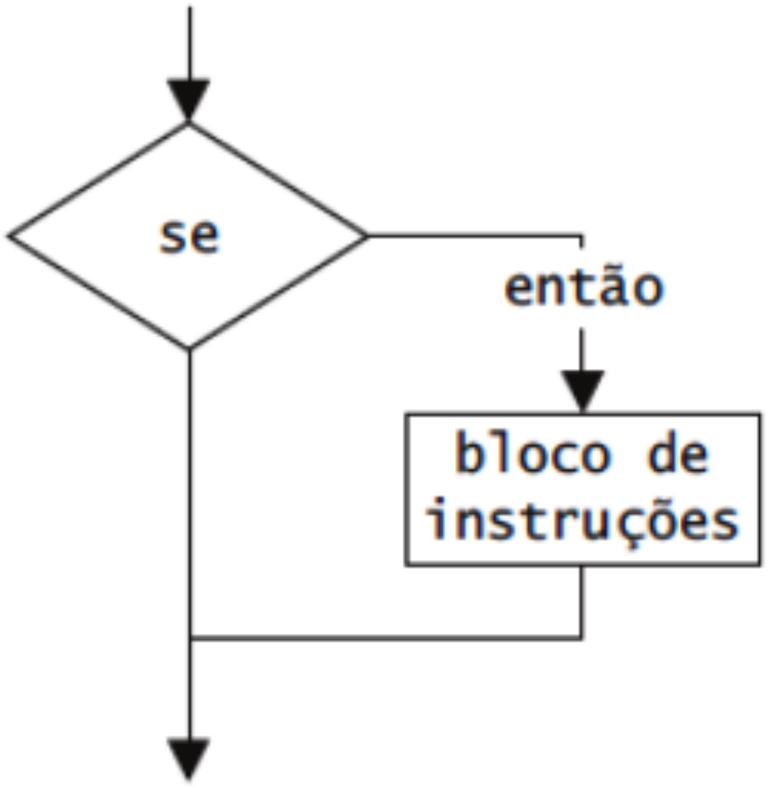
```
Está a chover (0-não, 1-sim): 0
Dia da semana (1-7): 7
Distância (km): 3
É possível fazer o piquenique?: verdadeiro
```

Operador ternário

Estruturas de controlo condicional

As estruturas de controlo condicional permitem decidir entre executar ou não determinada ação ou decidir entre duas, ou mais, alternativas possíveis, com base numa condição ou expressão **booleana**. Podemos destacar as seguintes:

- se.. então / if(){}
- Se .. então .. senão / if(){}else{}
- Se .. então .. senão se .. então / if(){}else if(){}else{}
- Se .. então .. senão se .. então / switch(){case:}




```
inicio
...
se <condição>
então <bloco-instruções>
fim-se
...
fim
```

Exemplo

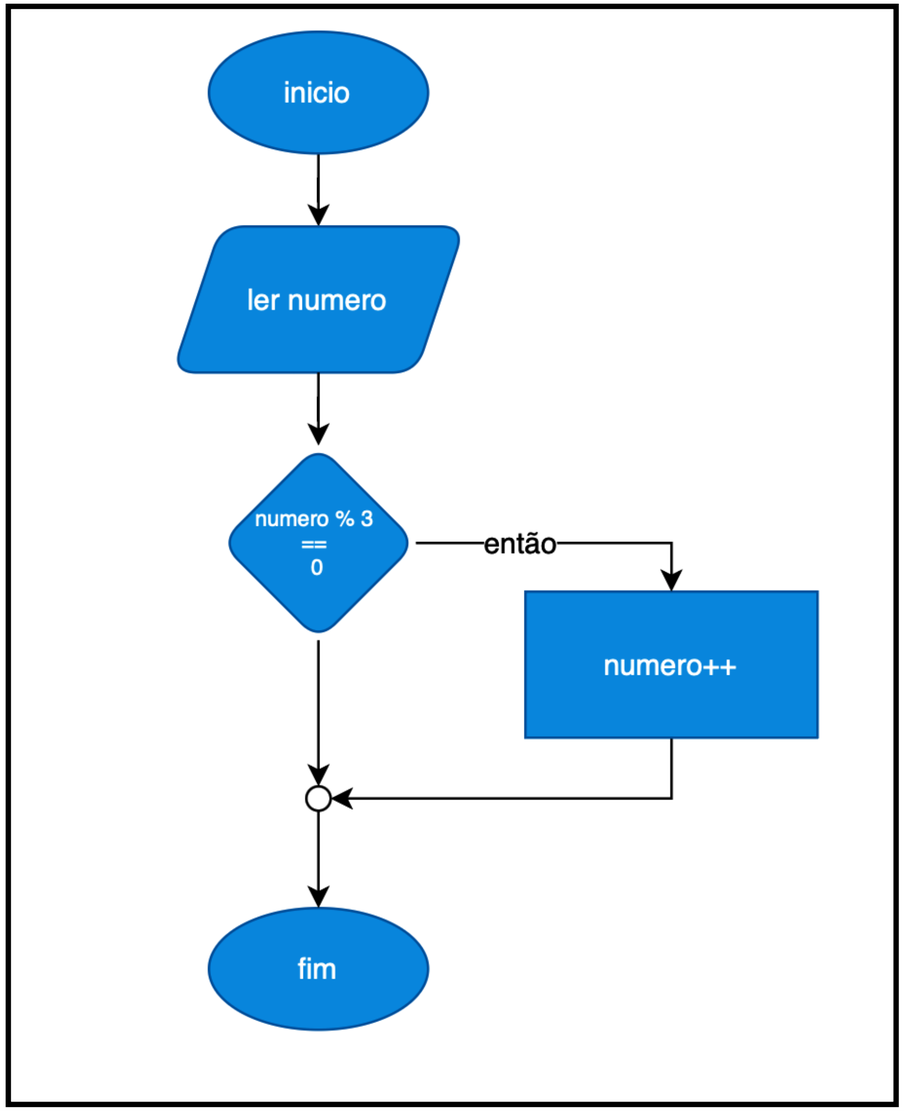
Escreva um algoritmo capaz de ler um número inteiro, que lhe incremente 1 no caso deste ser divisível por 3.

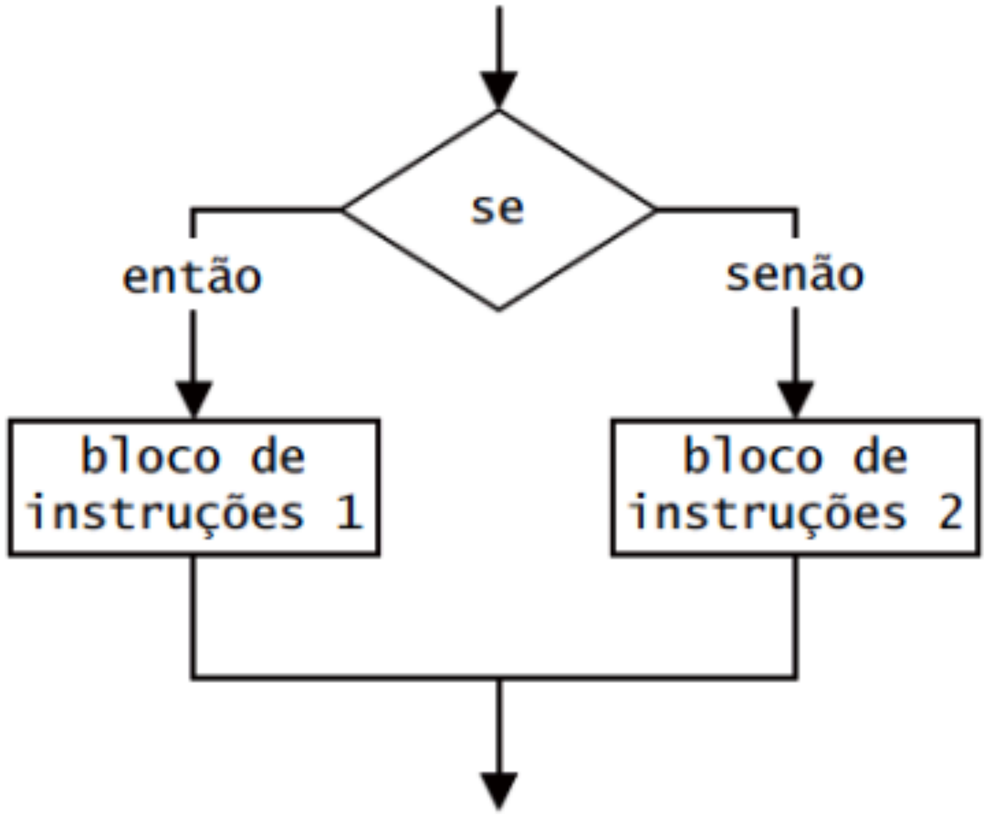
```
variáveis
    numero: inteiro
início
    ler(numero)

    se numero % 3 = 0 então
        numero ← numero + 1
    fim-se

    escrever(numero)
fim
```

```
let numero = Number(prompt("Número: "));

if (numero % 3 === 0) {
  numero++;
}

console.log("Resultado:", numero);
```

Exemplo

Escreva um algoritmo capaz de ler um número inteiro, que lhe incremente 1 no caso deste ser divisível por 3.






```
inicio
...
se <condição>
então <bloco-instruções1>
senão <bloco-instruções2>
fim-se
...
fim
```

Exemplo

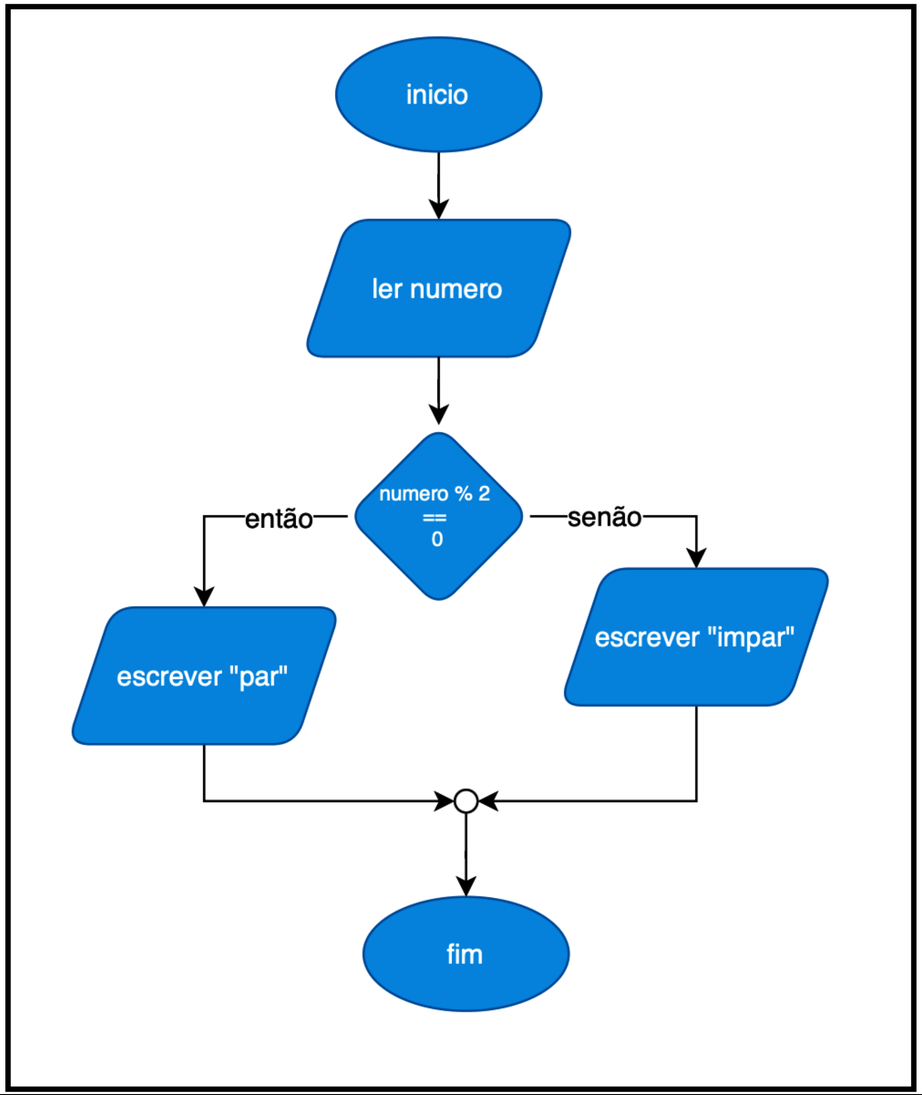
Escreva um algoritmo capaz de ler um número inteiro e verificar se este é par ou impar.

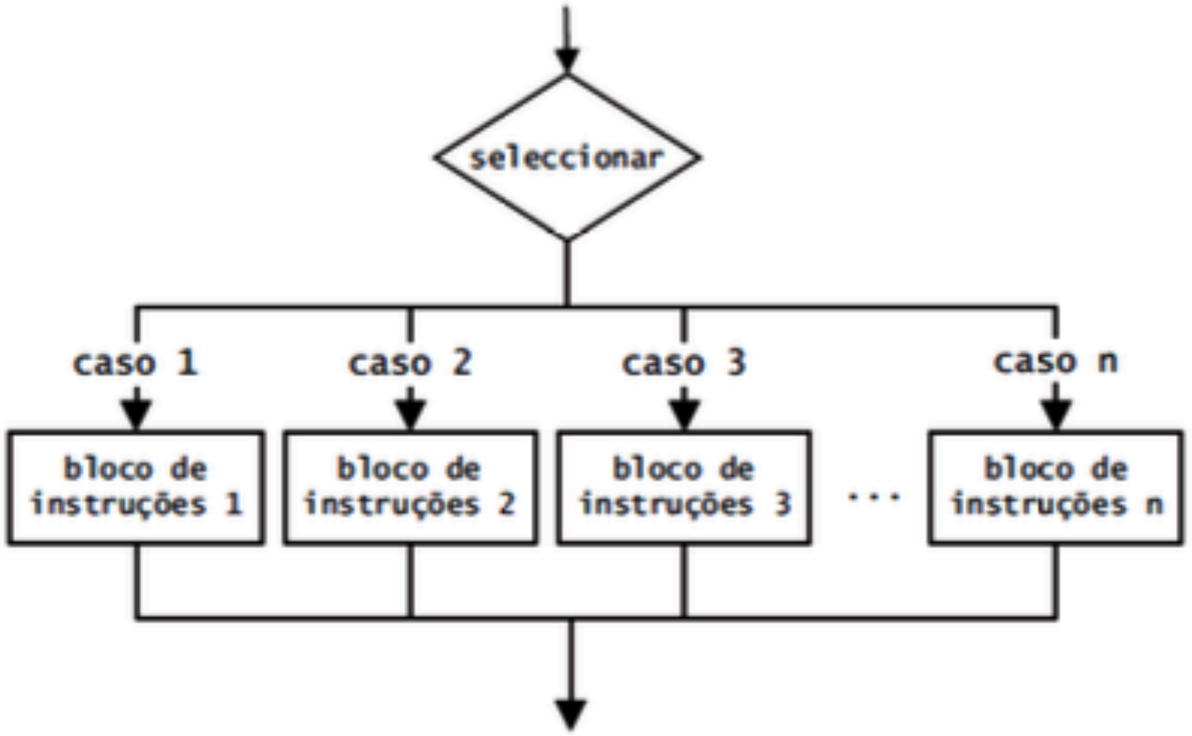
```
variáveis
    numero: inteiro
início
    ler(numero)

    se numero % 2 = 0 então
        escrever("par")
    senão
        escrever("impar")
    fim-se
fim
```

```
const numero = Number(prompt("Número: "));

if (numero % 2 === 0) {
  console.log("O número é par");
} else {
  console.log("O número é ímpar");
}
```

Exemplo

Escreva um algoritmo capaz de ler um número inteiro e verificar se este é par ou impar.






```
inicio
...
seleccionar <variável>
caso <valor1> <bloco-instruções1>
caso <valor2> <bloco-instruções2>
caso <valor3> <bloco-instruções3>
...
senão <bloco-instruçõesN>
fim-seleccionar
...
fim
```

Exemplo

Escreva um algoritmo capaz de ler um número inteiro de 1 a 7, e apresentar na consola o dia da semana correspondente.

```
variáveis
    numero: inteiro
início
    ler(numero)

    se numero = 1 então
        escrever("Domingo")
    senão se numero = 2 então
        escrever("Segunda")
    senão se numero = 3 então
        escrever("Terça")
    senão se numero = 4 então
        escrever("Quarta")
    senão se numero = 5 então
        escrever("Quinta")
    senão se numero = 6 então
        escrever("Sexta")
    senão se numero = 7 então
        escrever("Sábado")
    senão
        escrever("Número inválido")
    fim-se
fim
```

Exemplo

Escreva um algoritmo capaz de ler um número inteiro de 1 a 7, e apresentar na consola o dia da semana correspondente.

```
const diaNumero = Number(prompt("Número (1-7): "));
let diaNome;

if (diaNumero === 1) {
  diaNome = "Segunda-feira";
} else if (diaNumero === 2) {
  diaNome = "Terça-feira";
} else if (diaNumero === 3) {
  diaNome = "Quarta-feira";
} else if (diaNumero === 4) {
  diaNome = "Quinta-feira";
} else if (diaNumero === 5) {
  diaNome = "Sexta-feira";
} else if (diaNumero === 6) {
  diaNome = "Sábado";
} else if (diaNumero === 7) {
  diaNome = "Domingo";
} else {
  diaNome = "Dia inválido";
}

console.log(diaNome);
```

Exemplo

Escreva um algoritmo capaz de ler um número inteiro de 1 a 7, e apresentar na consola o dia da semana correspondente.

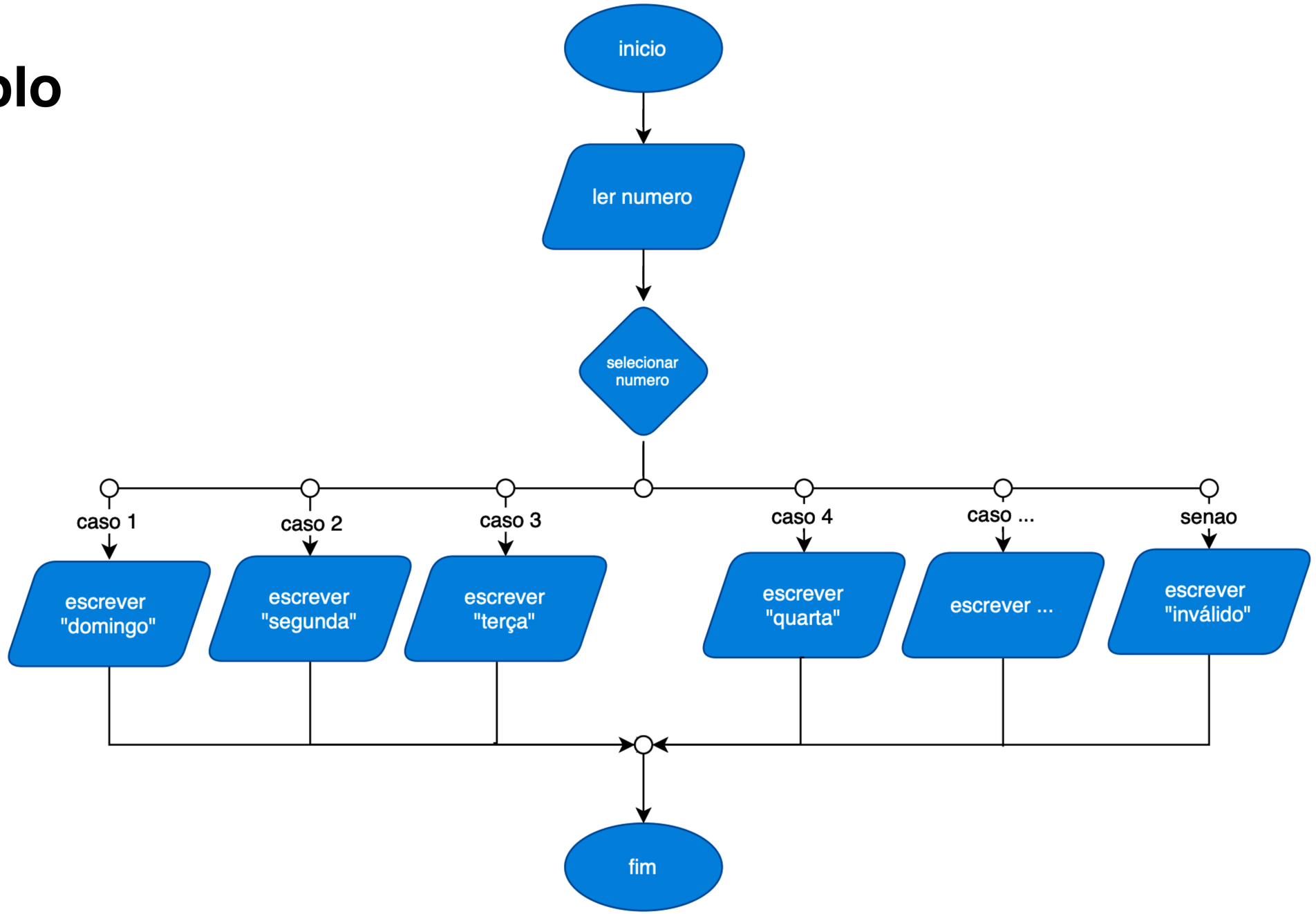
```
variáveis
    numero: inteiro
início
    ler(numero)

    selecionar numero
        caso 1:
            escrever("Domingo")
        caso 2:
            escrever("Segunda")
        caso 3:
            escrever("Terça")
        caso 4:
            escrever("Quarta")
        caso 5:
            escrever("Quinta")
        caso 6:
            escrever("Sexta")
        caso 7:
            escrever("Sábado")
        senao:
            escrever("Número inválido")
    fim-selecionar
fim
```

Exemplo

Escreva um algoritmo capaz de ler um número inteiro de 1 a 7, e apresentar na consola o dia da semana correspondente.

```
const diaNumero = Number(prompt("Número (1-7): "));
let diaNome;

switch (diaNumero) {
  case 1:
    diaNome = "Segunda-feira";
    break;
  case 2:
    diaNome = "Terça-feira";
    break;
  case 3:
    diaNome = "Quarta-feira";
    break;
  case 4:
    diaNome = "Quinta-feira";
    break;
  case 5:
    diaNome = "Sexta-feira";
    break;
  case 6:
    diaNome = "Sábado";
    break;
  case 7:
    diaNome = "Domingo";
    break;
  default:
    diaNome = "Dia inválido";
}

console.log(diaNome);
```

Exemplo


inicio
ler numero
selecionar numero
caso 1
caso 2
caso 3
caso 4
caso ...
senao
escrever "domingo"
escrever "segunda"
escrever "terça"
escrever "quarta"
escrever ...
escrever "inválido"
fim

Truthy e Falsy — valores “verdadeiros” e “falsos” que não são booleanos

Valores falsy

São tratados como false numa condição:

- 0
- "" (string vazia)
- null
- undefined
- NaN
- false

```
if ("") {
  console.log("não imprime");
}
```

Valores truthy

Tudo o resto:

- strings não vazias ("Olá")
- números diferentes de 0
- arrays ([])
- objetos ({})
- true

```
if ("Olá") {
  console.log("Isto imprime");
}
```

NaN — comparação especial

- NaN significa Not-a-Number.
- Representa o resultado de uma operação numérica inválida.
- O seu tipo é number (peculiaridade histórica da linguagem).
- NaN não é igual a si próprio:

```
NaN === NaN     // false
```

- Qualquer operação com NaN gera NaN:

```
10 + NaN        // NaN
NaN * 5         // NaN
```

```
Number("abc")       // NaN
0 / 0               // NaN
Math.sqrt(-1)       // NaN
```

- testar corretamente se um valor é NaN

```
Number.isNaN(Number("abc"));
```

undefined em JavaScript

- **undefined** significa: a variável foi declarada mas ainda não tem valor.
- É um tipo próprio: **undefined**.
- É um valor **false**.

- **Como testar undefined?**

```
x === undefined   // forma recomendada
```

```
let x;
console.log(x);         // undefined  (variável declarada sem valor)

function f() {}
console.log(f());       // undefined  (função sem return)

let pessoa = { nome: "Ana" };
console.log(pessoa.idade);   // undefined  (propriedade não existe)

function ola(nome) {
  console.log(nome);
}
ola();                  // undefined (parâmetro não enviado)
```

null em JavaScript

- **null** significa ausência intencional de valor.
- Tipo reportado é “**object**"..
- Também é um valor **false**.

```
let user = null;       // valor vazio intencionalmente

let resposta = null; // à espera de ser preenchida depois

let aluno = {
  nome: "Rita",
  tutor: null, // tutor ainda não definido
};
```

```
null == undefined   // true  (iguais em comparação fraca)
null === undefined  // false (tipos diferentes)
```

Resumo

Valor	Significado	Tipo	False?
undefined	variável existe mas não tem valor	undefined	✓
null	valor vazio intencional	object	✓
NaN	erro em operação numérica	number	✓

Algoritmia e Programação

Estruturas de Controlo Condicional

Marta Martinho 2025