



# Pensamento Computacional e Lógica de programação



# Pensamento Computacional

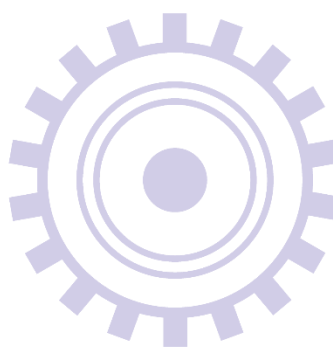
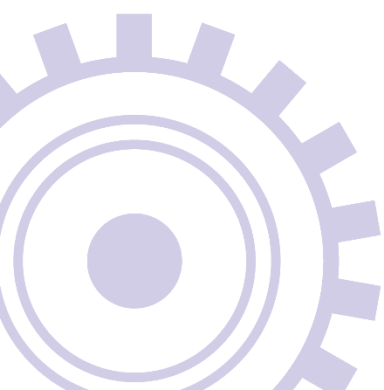



## Objetivos

Compreender o que é o pensamento computacional, suas etapas e como ele auxilia na resolução de problemas.



## Conceitos:

- Pensamento Computacional:** Habilidade de resolver problemas de forma lógica e estruturada.
  - Analogia:** Imagine um chef de cozinha preparando um jantar completo. Ele divide o preparo em etapas (entrada, prato principal, sobremesa), organiza os ingredientes necessários, segue uma receita e ajusta conforme necessário. Da mesma forma, o pensamento computacional divide problemas complexos em partes menores para facilitar sua resolução.
- 
- 
- 

# Etapas do Pensamento Computacional:

**Decomposição:** Dividir problemas grandes em partes menores.

*Analogia:* Ao planejar uma viagem, você separa as tarefas: escolher o destino, reservar o transporte, organizar hospedagem e planejar passeios.

**Reconhecimento de Padrões:** Identificar tendências e semelhanças.

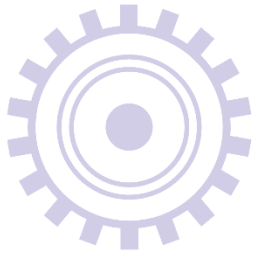
*Analogia:* Ao organizar roupas, você separa por cores ou tipos (camisetas, calças).

**Abstração:** Filtrar informações irrelevantes e focar no essencial.

*Analogia:* Ao preparar uma receita, você ignora detalhes como a cor da embalagem dos ingredientes e foca no conteúdo.

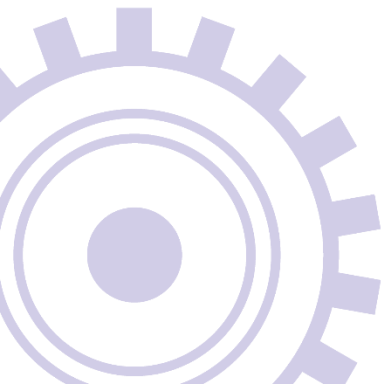
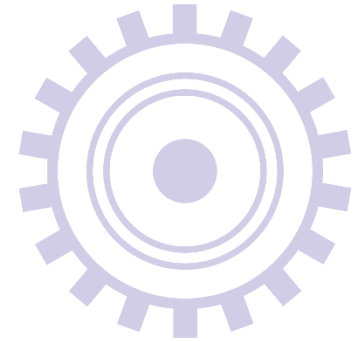
**Algoritmos:** Criar sequências de passos para resolver o problema.

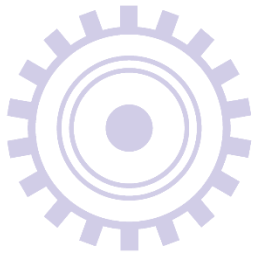
*Analogia:* Seguir uma receita culinária para preparar um bolo.



Aplicando a Decomposição:

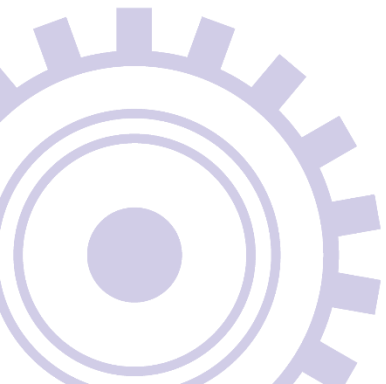
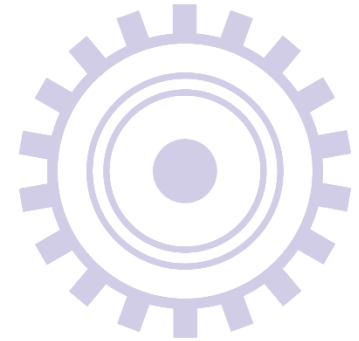
1. Definição do Problema: Calcular a média de notas de uma turma.
2. Decomposição em Etapas:
  - Etapa 1: Coleta de Dados:
    - Ler as notas de cada aluno da turma.
    - Armazenar as notas em uma variável ou estrutura de dados adequada.
  - Etapa 2: Cálculo da Soma das Notas:
    - Somar todas as notas armazenadas na variável ou estrutura de dados.
  - Etapa 3: Cálculo da Média:
    - Dividir a soma das notas pelo número total de alunos.
  - Etapa 4: Exibição da Média:
    - Mostrar a média calculada para o usuário.

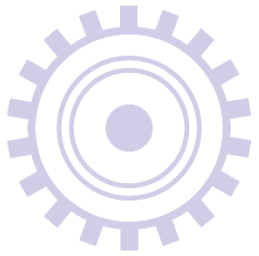




## Desvendando o Problema:

1. Definição do Problema: Converter um valor em reais para dólares americanos.
2. Decomposição em Etapas:
  - Etapa 1: Obter o Valor em Reais:
    - O valor em reais pode ser obtido através da entrada do usuário, de um arquivo ou de uma variável pré-definida.
  - Etapa 2: Obter a Taxa de Câmbio:
    - A taxa de câmbio é a cotação atual de uma moeda em relação a outra. Ela pode ser obtida de um site especializado, de uma API ou de um banco de dados atualizado.
  - Etapa 3: Realizar a Conversão:
    - Dividir o valor em reais pela taxa de câmbio para obter o valor equivalente em dólares.
  - Etapa 4: Apresentar o Resultado:
    - Mostrar o valor convertido em dólares para o usuário.

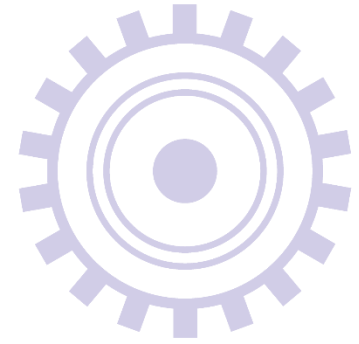


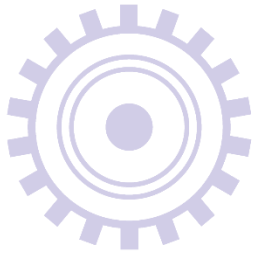


Vamos mostrar algo mais complexo dentro deste contexto de decomposição no pensamento computacional.

Exemplo 1: Desenvolvimento de um sistema de gerenciamento de tarefas.

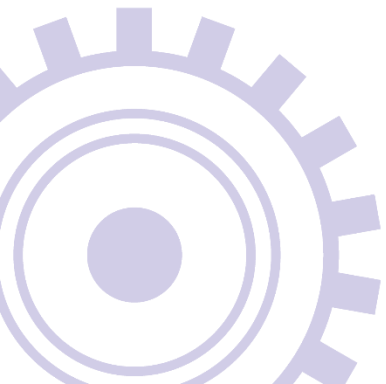
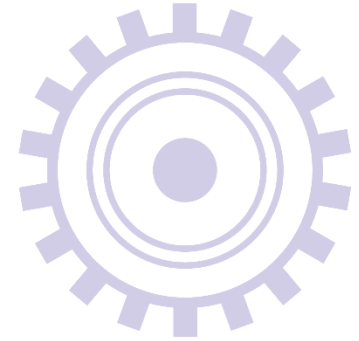
- Identificar as partes do problema: criar uma interface de usuário, implementar a funcionalidade de adicionar tarefas, definir a lógica de priorização e criar a funcionalidade de marcar tarefas como concluídas.
- Quebrar o problema em tarefas menores: para cada parte identificada, desenvolver algoritmos específicos que resolvam cada uma delas.
- Implementar as soluções: criar as interfaces de usuário, escrever o código para adicionar tarefas, definir as regras de priorização e implementar a funcionalidade de marcar tarefas como concluídas.



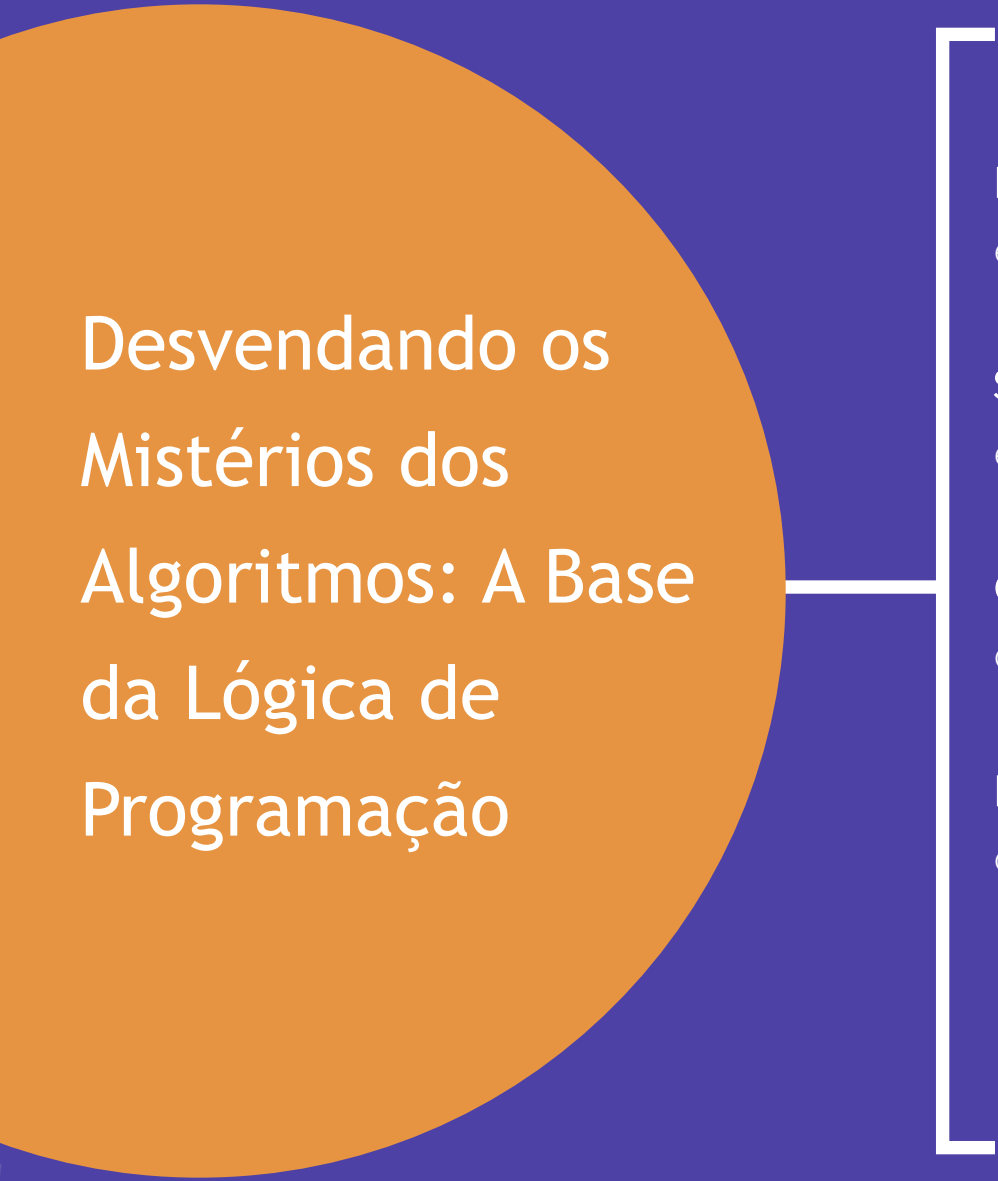


## Exemplo 2: Criação de um aplicativo de entrega de comida

- Identificar as partes do problema: criar uma interface de usuário para fazer pedidos, implementar a funcionalidade de rastreamento de pedidos, definir a lógica de cálculo de tempo de entrega e desenvolver a funcionalidade de pagamento online.
- Quebrar o problema em tarefas menores: para cada parte identificada, desenvolver algoritmos específicos que resolvam cada uma delas.
- Implementar as soluções: criar as interfaces de usuário, escrever o código para fazer pedidos, implementar a lógica de rastreamento de pedidos, desenvolver o cálculo de tempo de entrega e implementar a funcionalidade de pagamento online.








# Desvendando os Mistérios dos Algoritmos: A Base da Lógica de Programação

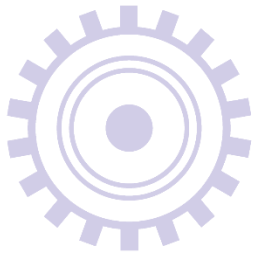
**Precisão:** As instruções devem ser claras, concisas e inequívocas para evitar interpretações errôneas pelo computador.

**Sequencialidade:** As instruções devem ser executadas em uma ordem específica para garantir o funcionamento correto do programa.

**Generalidade:** O algoritmo deve ser capaz de resolver o problema para diferentes conjuntos de dados, dentro dos limites do problema.

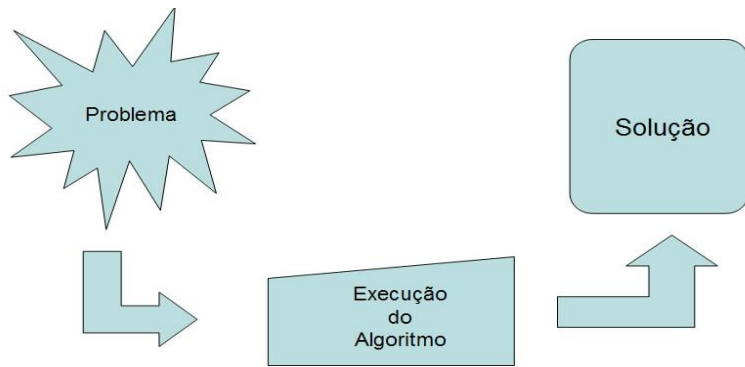
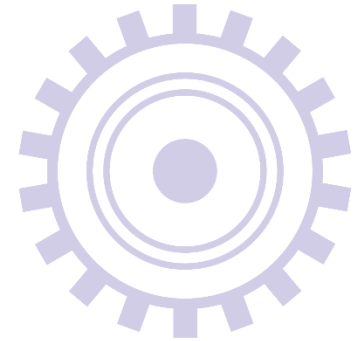
**Finitudes:** O algoritmo deve ter um número finito de instruções e chegar a um resultado em um tempo determinado.



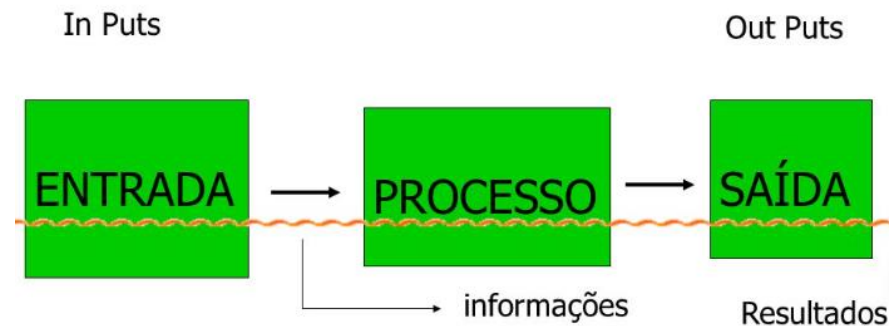


## Domine os Conceitos Essenciais DE ALGORITMOS:

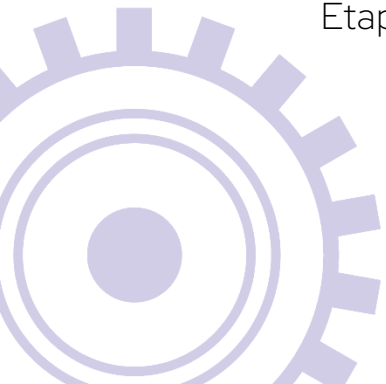
- Definição: Um algoritmo é um conjunto de instruções passo a passo, finitas e precisas, que definem como um problema deve ser solucionado. É como uma receita culinária que indica os ingredientes e os passos necessários para preparar um prato delicioso.

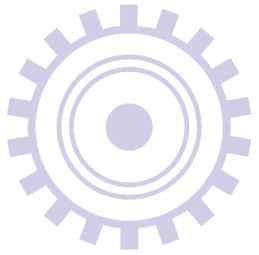


Etapas básicas de um algoritmo



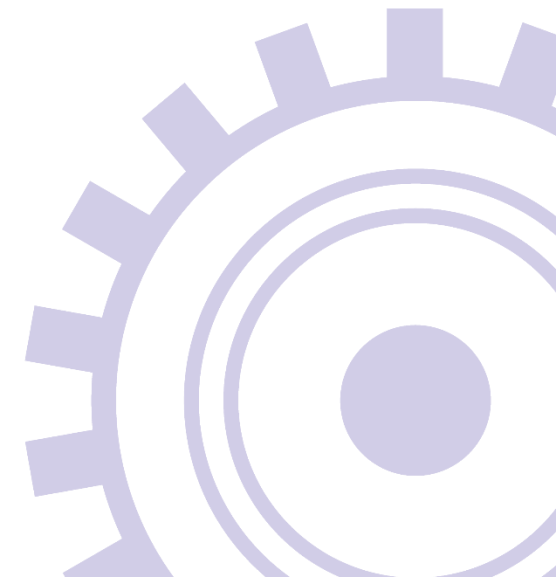
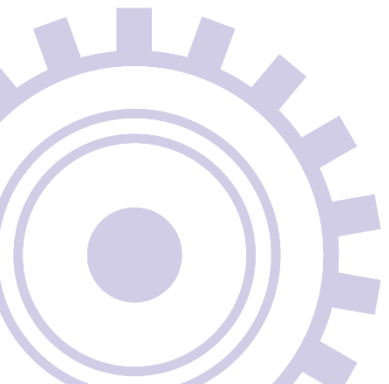
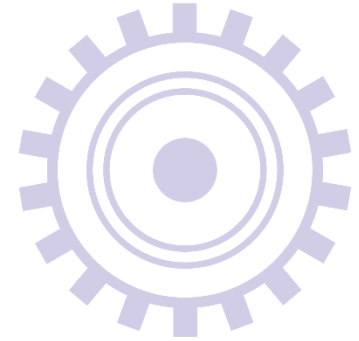
Etapas básicas em um sistema computacional

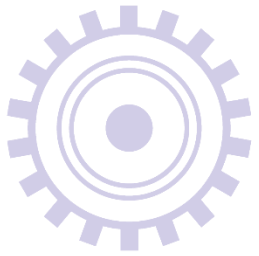




### Tipos de Algoritmos básicos:

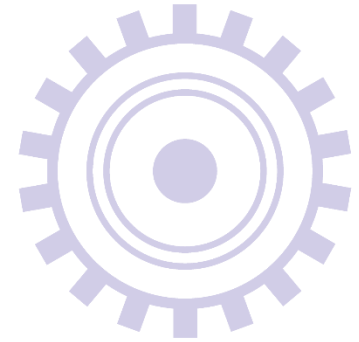
- Algoritmos Sequenciais: As instruções são executadas uma após a outra, em uma ordem predeterminada.
  - Exemplo: Algoritmo para calcular a média de notas de alunos.
- Algoritmos de Decisão: Utilizam instruções condicionais para tomar decisões com base em determinados critérios.
  - Exemplo: Algoritmo para verificar se um número é par ou ímpar.
- Algoritmos de Repetição: Repetem um bloco de instruções um número determinado de vezes ou até que uma condição seja satisfeita.
  - Exemplo: Algoritmo para imprimir os números de 1 a 100.



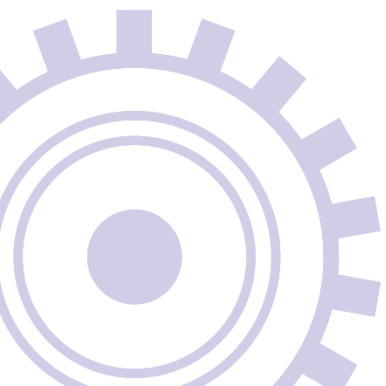


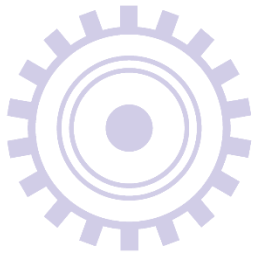
## Algoritmos Sequenciais:

Um algoritmo sequencial executa passos em uma sequência linear, sem desvios ou repetições. Aqui está um exemplo simples de um algoritmo sequencial para calcular a média de três números:



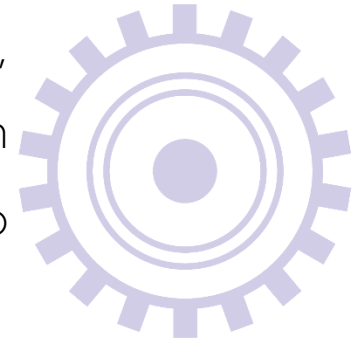
1. Ler o primeiro número (num1) do usuário.
2. Ler o segundo número (num2) do usuário.
3. Ler o terceiro número (num3) do usuário.
4. Calcular a soma dos três números:  $\text{soma} = \text{num1} + \text{num2} + \text{num3}$ .
5. Calcular a média dos três números:  $\text{media} = \text{soma} / 3$ .
6. Exibir a média calculada.





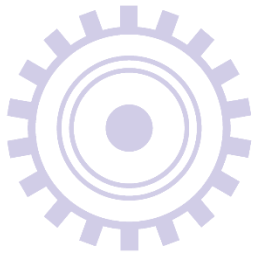
## Algoritmos de Decisão:

Algoritmos de decisão incluem estruturas de controle como if, else, e switch, que permitem que o programa tome diferentes caminhos com base em condições específicas. Aqui está um exemplo de um algoritmo de decisão que verifica se um número é positivo, negativo ou zero:



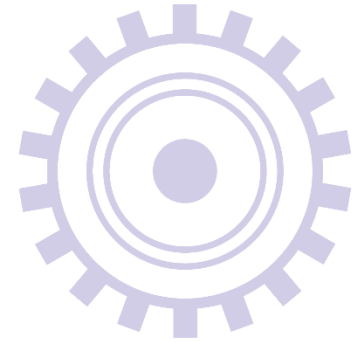
1. Ler o número (num) do usuário.
2. Se num é maior que zero, então:
  1. Exibir "O número é positivo."
3. Senão, se num é igual a zero, então:
  1. Exibir "O número é zero."
4. Senão:
  1. Exibir "O número é negativo."



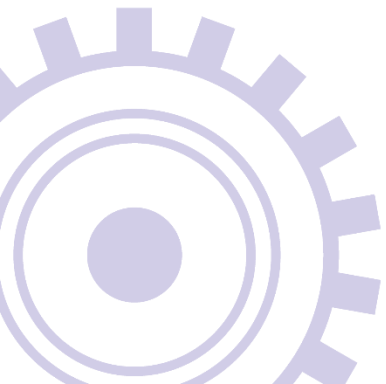


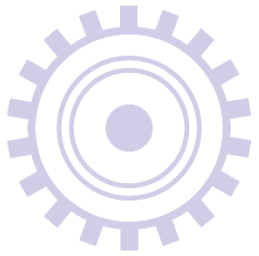
## Algoritmos de Repetição:

Algoritmos de repetição, também conhecidos como loops, repetem um bloco de código várias vezes até que uma condição seja atendida. Aqui está um exemplo de um algoritmo de repetição que imprime os números de 1 a 10:



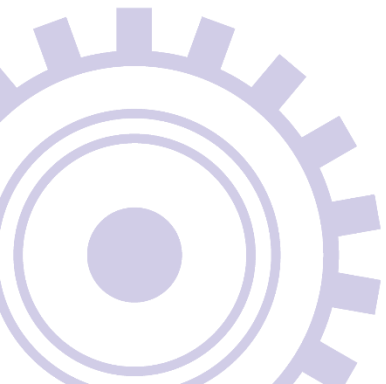
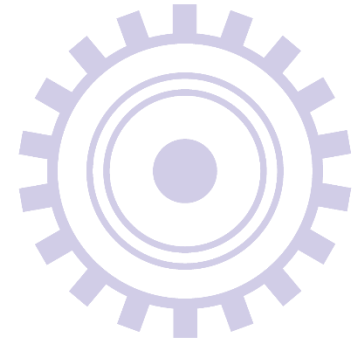
1. Inicializar uma variável 'i' com o valor 1.
2. Enquanto 'i' for menor ou igual a 10, faça:
  1. Exibir o valor de 'i'.
  2. Incrementar 'i' em 1.





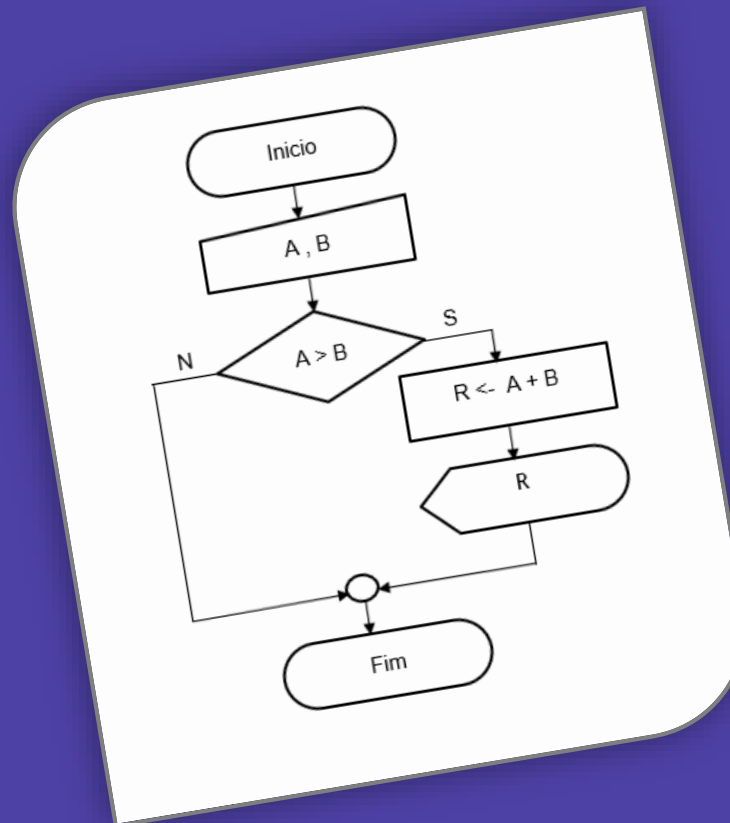
### Importância do algoritmo para o Aprendizado em Programação:

- Base para a Lógica de Programação: Os algoritmos são a base para a construção de programas lógicos e eficientes. Ao dominá-los, você desenvolve a capacidade de pensar de forma estruturada e solucionar problemas de forma organizada.
- Desenvolvimento do Raciocínio Lógico: A criação de algoritmos exige análise, decomposição de problemas e organização de ideias, aprimorando seu raciocínio lógico e habilidades de resolução de problemas.
- Aprendizagem por Prática: A prática com diferentes tipos de algoritmos permite que você explore diversas técnicas de programação e consolide seus conhecimentos na linguagem escolhida.
- Base para Habilidades Avançadas: O domínio dos algoritmos é fundamental para aprender conceitos mais complexos da programação, como estruturas de dados, algoritmos de ordenação e busca, e programação orientada a objetos.





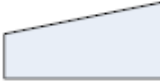

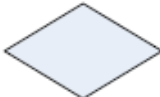


# Algoritmo em Fluxograma / Diagrama de blocos

**Fluxograma:** É uma forma universal de representação, pois se utiliza de figuras geométricas para ilustrar os passos a serem seguidos na resolução de um problema.

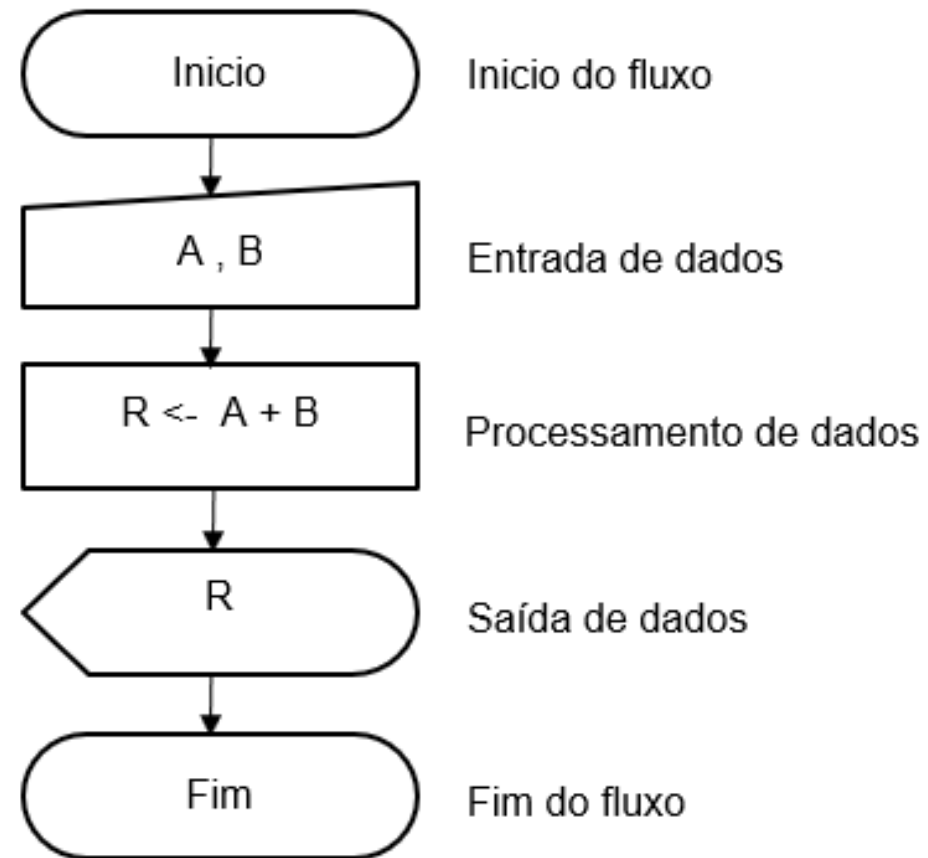




Esta forma de representação, utiliza símbolos gráficos para representar algoritmos. Existem símbolos padronizados para cada tipo de instrução, como por exemplo, início, entrada de dados, processamento (cálculos), saída de dados, fim, decisão, etc. Aqui estão alguns símbolos mais comuns, utilizados em Fluxogramas

SÍMBOLO	NOME	FUNÇÃO
	TERMINAL	Indica o início ou fim de um algoritmo.
	PROCESSAMENTO	Representa um processamento
	ENTRADA MANUAL DE DADOS	Indica entrada de dados manual, através de um teclado por exemplo.
	EXIBIR	Representa a exibição dos dados e informações, através de um monitor por exemplo.
	DECISÃO	Representa um teste lógico que escolhe qual instrução será executado.
	PREPARAÇÃO	Representa uma ação de preparação para o processamento.
	CONECTOR	Utilizado para interligar partes de um fluxograma ou para desviar o fluxo

## Fluxograma/Diagrama de blocos Sequencial



## Fluxograma/Diagrama de blocos Sequencial

É hora dos exercícios

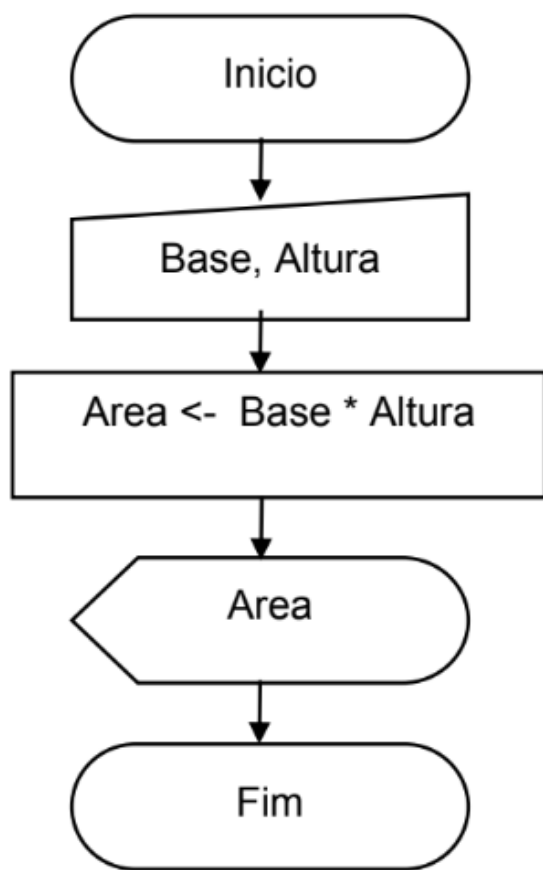
Usando os conceitos de algoritmos em sua forma de fluxograma, resolva os seguintes problemas.

1 - Escreva um algoritmo usando fluxograma para ler as dimensões de um retângulo (base e altura), calcular e escrever a área do retângulo.

2 - Faça um algoritmo usando fluxograma para ler o salário mensal atual de um funcionário e o percentual de reajuste. Calcular e escrever o valor do novo salário.

**Criar Fluxograma**

## Fluxograma/Diagrama de blocos Sequencial



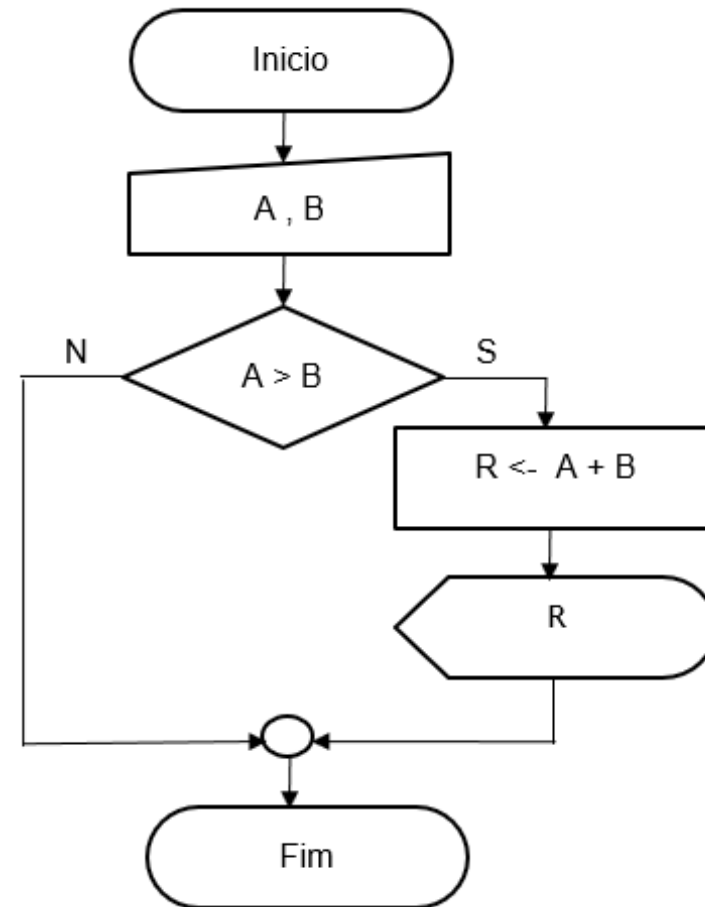
Exercício 01



Exercício 02

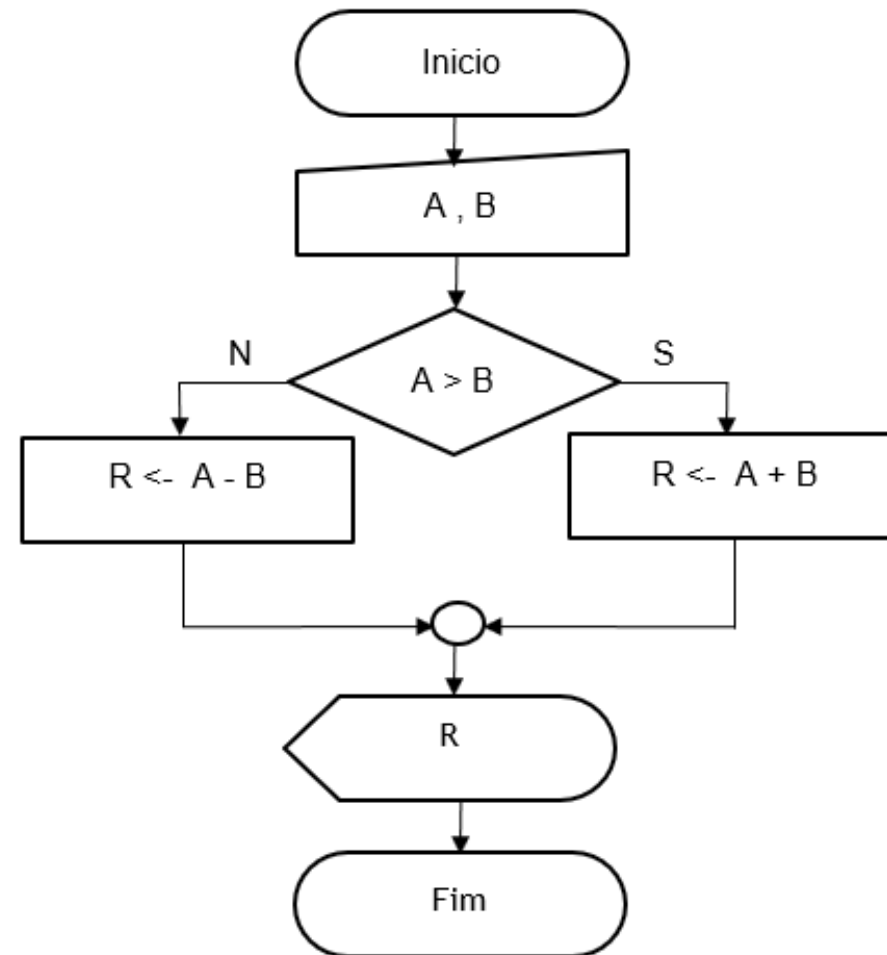
## Fluxograma/Diagrama de blocos Condicional

Nesse fluxograma, serão lidas duas variáveis A e B, o próximo passo é feito uma pergunta, SE A maior que B, se a resposta for verdadeira, então o fluxo será desviado para o lado da letra "S" e será feito a atribuição para a variável "R" que recebe a soma de  $A + B$ , logo após mostra a letra "R" e finaliza o fluxograma, caso contrário A menor que B o fluxo seguirá para o lado da letra "N" e será finalizado o fluxograma.



## Fluxograma/Diagrama de blocos Condicional

Nesse fluxograma, serão lidas duas variáveis A e B, o próximo passo é feito uma pergunta, SE A maior que B, se a resposta for verdadeira, então o fluxo será desviado para o lado da letra "S" e será feita a atribuição para a variável "R" que recebe a soma de  $A + B$ , logo após mostra a letra "R" e finaliza o fluxograma, caso contrário se A menor que B o fluxo seguirá para o lado da letra "N" e será feita a atribuição para a variável "R" que recebe a soma de  $A - B$ , logo após mostra a letra "R" e finaliza o fluxograma.



## Fluxograma/Diagrama de blocos Condicional

É hora dos exercícios

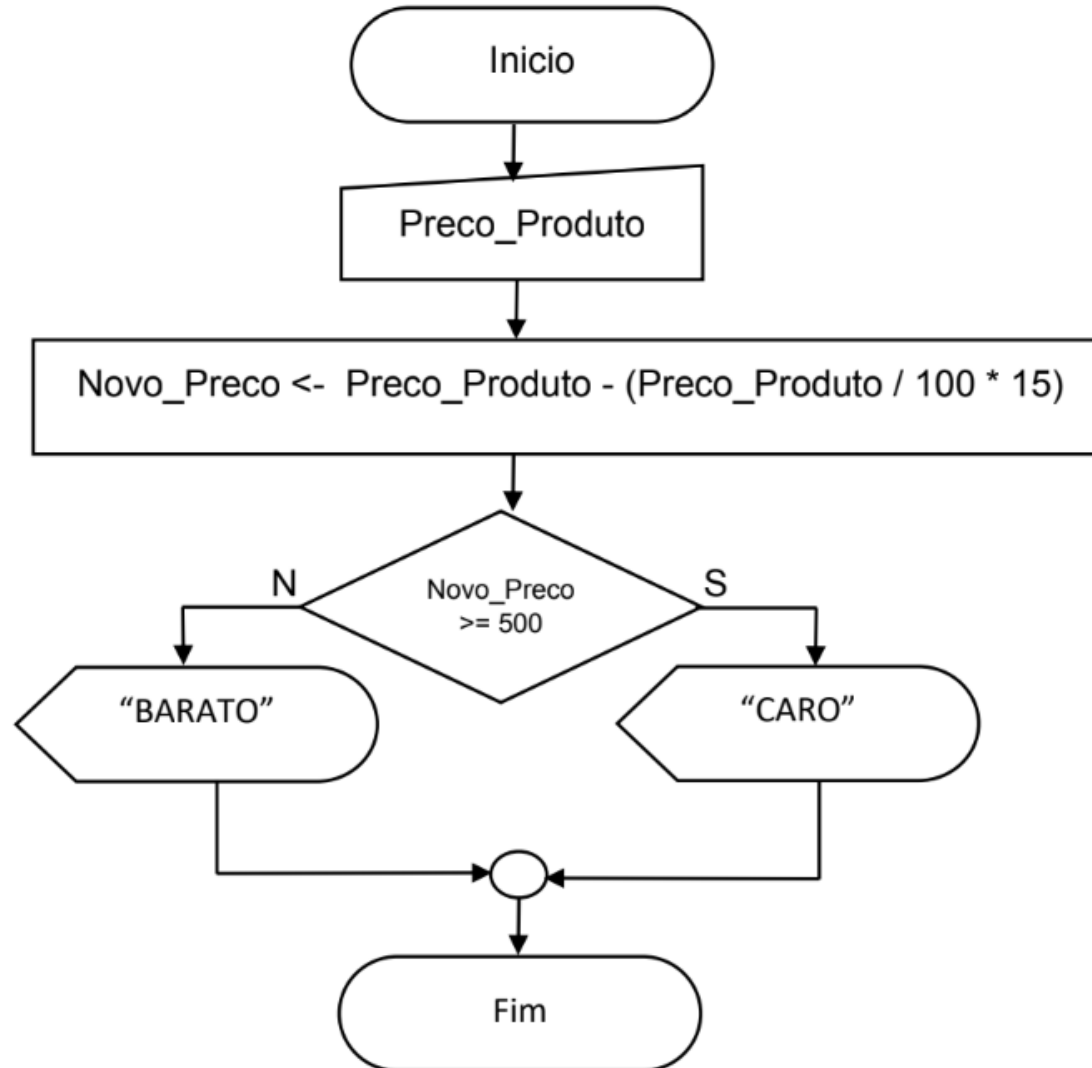
Usando os conceitos de algoritmos em sua forma de fluxograma, resolva os seguintes problemas.

**DESAFIO:** Faça um algoritmo que mostre o novo preço de um produto sabendo-se que este terá um desconto de 15%, mostrando também a classificação do produto segundo as seguintes informações:

Novo preço:  $\geq 500$  = "CARO" e  $< 500$  = "BARATO"

Dado 3 valores mostre o maior usando fluxograma

## Resolução





# Pseudocódigo ou Português estruturado

```
1 Algoritmo "Nome do Algoritmo"
2 // Disciplina : Técnicas e Lógica de Programação
3 // Professor : Flavio Mota da Cruz
4 // Descrição : Flavio Mota da Cruz
5 // Autor(a) : Flavio Mota da Cruz
6 // Data atual :
7
8 Var
9
10 Nota1, Nota2:real // Exemplos de variaveis
11 Media:real
12 Inicio
13
14
15 Leia(Nota1) // Comando para ler a variável Nota1
16 Leia(Nota2) // Comando para ler a variável Nota2
17
18 Media <- (Nota1 + Nota2) / 2 // Processamento de dados
19
20 Escreva(Media) // Exemplo de saída de dados
21 Escreva("A média das notas é = ", Media) // Exemplo de saída de dados
22
23
24 Fimalgoritmo
```

Utiliza linguagem estruturada e se assemelha na forma de um programa escrito em linguagem de programação, também conhecido como português estruturado ou Portugol

## Características Essenciais:

- Linguagem Natural: Utiliza linguagem comum, facilitando a compreensão e o compartilhamento entre pessoas com diferentes níveis de conhecimento em programação.
- Estrutura Clara: Define a sequência de instruções de forma organizada, utilizando blocos de código, indentação e comentários.
- Independência da Linguagem: Não está ligado a uma linguagem de programação específica, permitindo a adaptação para diferentes linguagens.

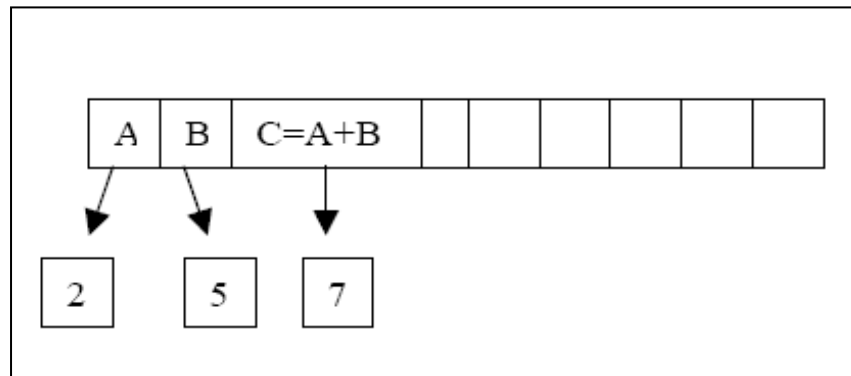
## Características Essenciais:

- Linguagem Natural: Utiliza linguagem comum, facilitando a compreensão e o compartilhamento entre pessoas com diferentes níveis de conhecimento em programação.
- Estrutura Clara: Define a sequência de instruções de forma organizada, utilizando blocos de código, indentação e comentários.
- Independência da Linguagem: Não está ligado a uma linguagem de programação específica, permitindo a adaptação para diferentes linguagens.

## Variáveis

O primeiro passo para que um programa seja executado em um computador é o carregamento desse programa para a memória. A memória é utilizada para armazenar tanto as instruções dos programas quanto os dados utilizados pelos mesmos. As variáveis são endereços de memória destinados a armazenar informações temporariamente, pois ao término da execução dos programas todos os dados utilizados serão perdidos caso não sejam salvos no computador. Um dado é classificado como variável quando tem a possibilidade de ser alterado no decorrer da execução do programa.

Variáveis de ENTRADA: armazenam informações fornecidas por um meio externo, normalmente usuários ou outro sistema. Variáveis de SAÍDA: armazenam dados processados como resultados.



Exemplo simples de como funciona uma variável na memória de um computador:

Nesse exemplo cada quadrado representa um espaço na memória do computador e cada espaço recebe apenas uma informação por vez, repare que as variáveis são representadas por letras e cada letra possui um valor  $A = 2$ ,  $B = 5$  e  $C$  é igual a soma de  $A + B$  ( $C = A + B$ ) = ( $C = 7$ ), isso quer dizer que uma variável pode receber além de valores também pode receber uma expressão matemática, seja ela básica ou mais complexa.

## Regras para nomear variáveis

Regra	Exemplo Correto	Exemplo Incorreto	Motivo
Inicie com um caractere alfabético (não com um número).	<code>nome1</code> , <code>salarioTotal</code>	<code>1nome</code> , <code>2variavel</code>	Evita erros de sintaxe e mantém a consistência com os padrões da maioria das linguagens.
Evite caracteres especiais como <code>" " ( ) / * ; + .</code>	<code>nome</code> , <code>salario_anual</code>	<code>Nome(M)</code> , <code>N*B</code>	Caracteres especiais podem confundir o compilador e dificultar a leitura do código.
Não use espaços em branco ou hífen; prefira o underline ( <code>_</code> ).	<code>salario_bruto</code>	<code>salario bruto</code> , <code>salario-bruto</code>	Espaços e hifens não são aceitos na maioria das linguagens. Underline é o padrão para separar palavras.
Escolha nomes sugestivos e descritivos para variáveis.	<code>salario_funcionarios</code>	<code>sf</code> , <code>variavel1</code>	Variáveis descritivas facilitam a compreensão do propósito delas no código.
Evite acentos e caracteres não-ASCII em nomes de variáveis.	<code>salario</code> , <code>funcionario_nome</code>	<code>salário</code> , <code>nome- funcionário</code>	Acentos e caracteres especiais podem causar problemas em ambientes de programação diversos.

## Tipos de dados

Quando declaramos as variáveis devemos atribuir para cada uma delas um tipo específico de dados, esses dados define a forma que o computador vai tratar a informação recebida ou processada.

Por exemplo: um cálculo deve ser feito por dados do tipo numérico, seja ele real ou do tipo inteiro, caso o sistema precise armazenar um nome de uma pessoa esse valor deve ser do tipo caractere, e se precisarmos guardar uma situação do tipo verdadeiro ou falso podemos usar uma variável do tipo lógica.

**Inteiro:** define variáveis numéricas do tipo inteiro, ou seja, sem casas decimais positivas ou negativas.

Ex. idade, número de filhos, números de gols.

**Real:** define variáveis numéricas do tipo real, ou seja, com casas decimais.

Ex. salário, peso, temperatura.

**Caractere:** define variáveis do tipo texto, ou seja, cadeia de caracteres.

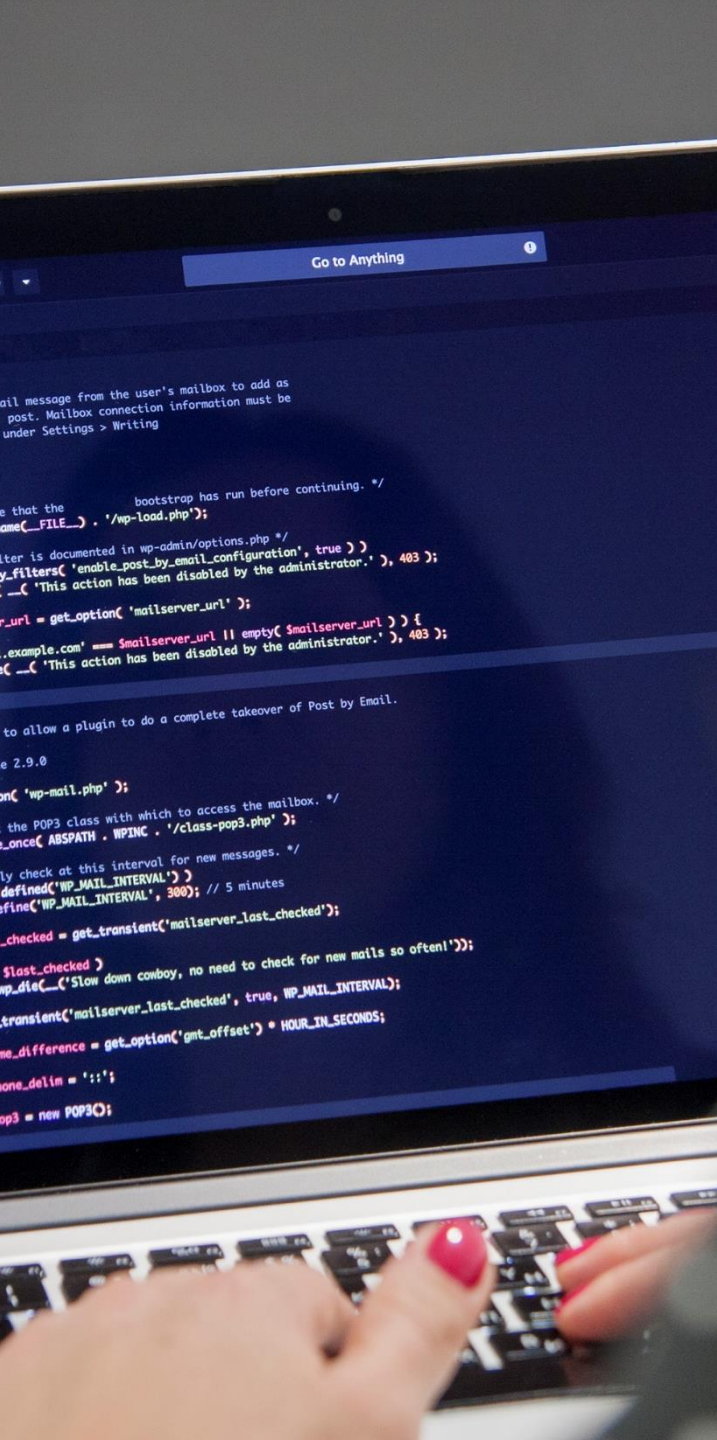
Ex. nome, endereço, frase.

**Lógico:** define variáveis do tipo lógica, ou seja, com valor VERDADEIRO ou FALSO.



Exemplo de algoritmo (pseudocódigo), declaração de variáveis usando VISUALG

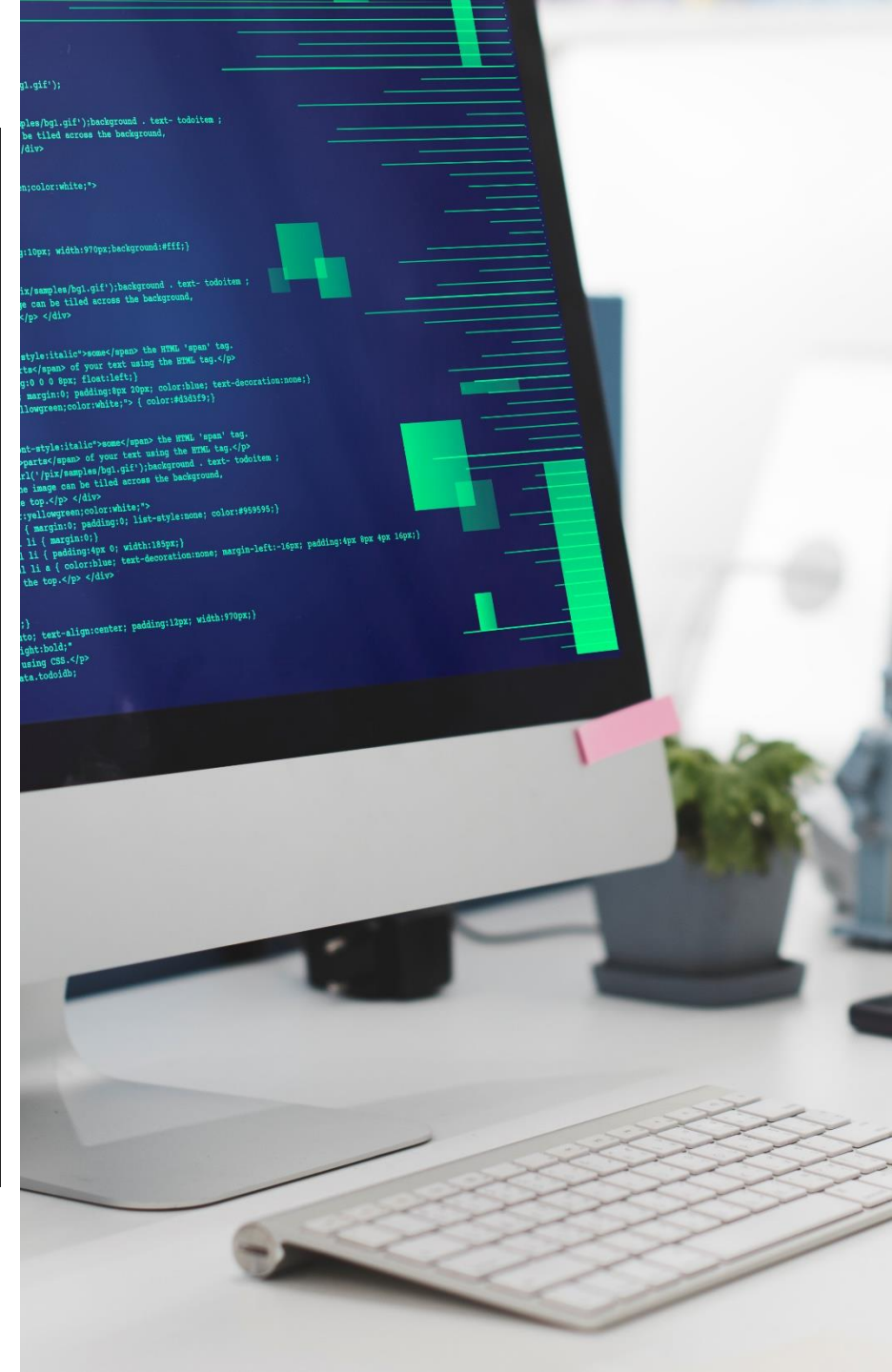
```
1 Algoritmo "Nome do Algoritmo"  
2 // Disciplina   : Técnicas e Lógica de Programação  
3 // Professor    : Flavio Mota da Cruz  
4 // Descrição    :  
5 // Autor(a)     : Flavio Mota da Cruz  
6 // Data atual   :  
7  
8 const  
9  
10 Var  
11 ValorDolar, Nota1, Nota2: real    // Exemplos de variaveis  
12 Fgts: real  
13 Inicio  
14  
15 Fgts <- 8    // Exemplo de constante  
16  
17 Fimalgoritmo
```

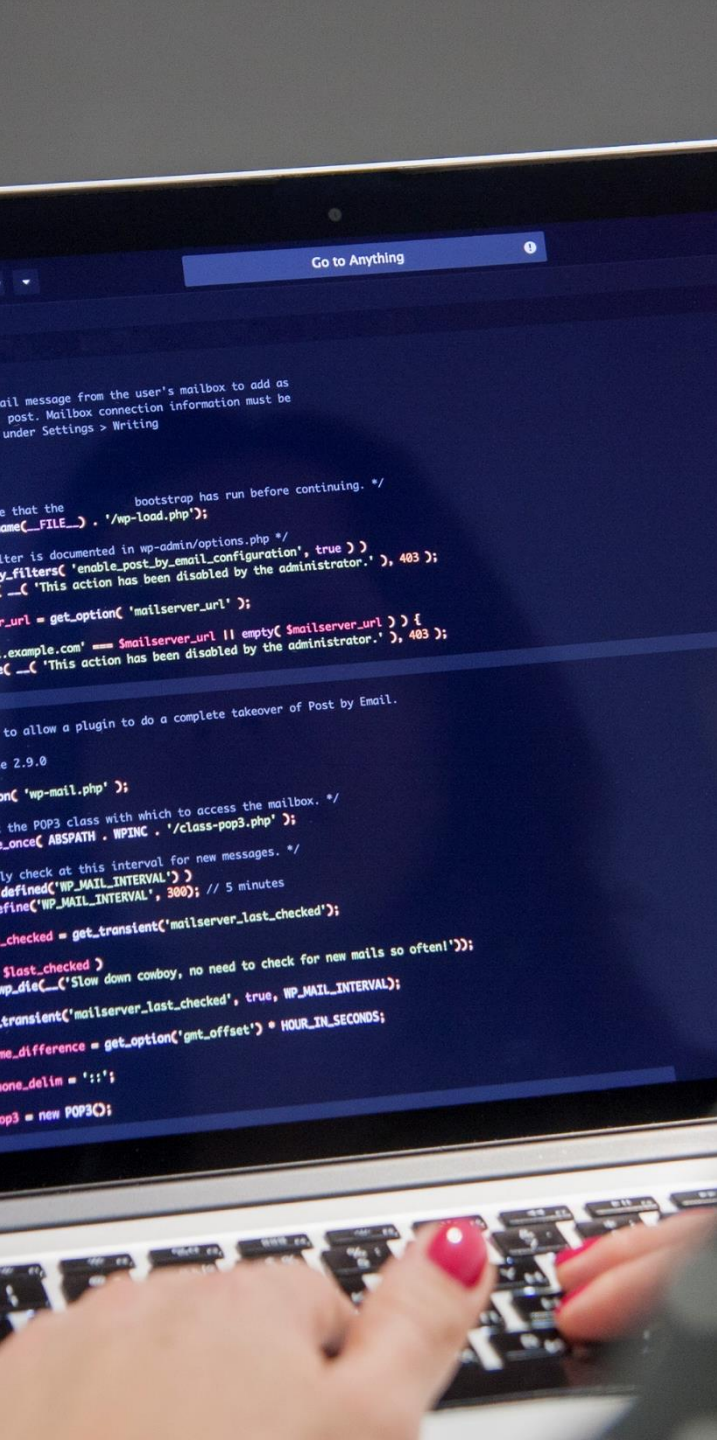


## Tabela de Operadores de Atribuição

Operador	Descrição	Exemplo em Pseudocódigo	Exemplo em Python	Exemplo em JavaScript	Resultado
=	Atribuição simples	A = 10	A = 10	A = 10	A recebe o valor 10
+=	Adição e atribuição	A = A + 5	A += 5	A += 5	Soma 5 ao valor atual de A
-=	Subtração e atribuição	A = A - 3	A -= 3	A -= 3	Subtrai 3 do valor atual de A
*=	Multiplicação e atribuição	A = A * 2	A *= 2	A *= 2	Multiplica o valor atual de A por 2
/=	Divisão e atribuição	A = A / 4	A /= 4	A /= 4	Divide o valor atual de A por 4
%=	Módulo e atribuição	A = A MOD 3	A %= 3	A %= 3	Calcula o resto da divisão de A por 3 e atribui a A
**=	Exponenciação e atribuição	A = A ^ 2	A **= 2	A **= 2	Eleva o valor atual de A ao quadrado
//=	Divisão inteira e atribuição	A = A DIV 3	A //= 3	Não aplicável	Realiza divisão inteira de A por 3 e atribui a A

Operador	Descrição	Exemplo em Pseudocódigo	Exemplo em Python	Exemplo em JavaScript	Resultado
+	Adição	A + B	A + B	A + B	Soma de A e B
-	Subtração	A - B	A - B	A - B	Subtração de A por B
*	Multiplicação	A * B	A * B	A * B	Multiplicação de A por B
/	Divisão	A / B	A / B	A / B	Divisão de A por B (retorna float)
MOD	Módulo (resto da divisão)	A MOD B	A % B	A % B	Resto da divisão de A por B
^	Exponenciação	A ^ B	A ** B	A ** B	A elevado à potência B
DIV	Divisão Inteira	A DIV B	A // B	Math.floor(A / B)	Quociente inteiro da divisão de A por B





## Tabela de Operadores Relacionais

Operador	Descrição	Exemplo em Pseudocódigo	Exemplo em Python	Exemplo em JavaScript	Resultado
=	Igual a	A = B	A == B	A === B	Verdadeiro se A é igual a B
≠	Diferente de	A ≠ B	A != B	A !== B	Verdadeiro se A é diferente de B
>	Maior que	A > B	A > B	A > B	Verdadeiro se A é maior que B
<	Menor que	A < B	A < B	A < B	Verdadeiro se A é menor que B
>=	Maior ou igual a	A >= B	A >= B	A >= B	Verdadeiro se A é maior ou igual a B
<=	Menor ou igual a	A <= B	A <= B	A <= B	Verdadeiro se A é menor ou igual a B



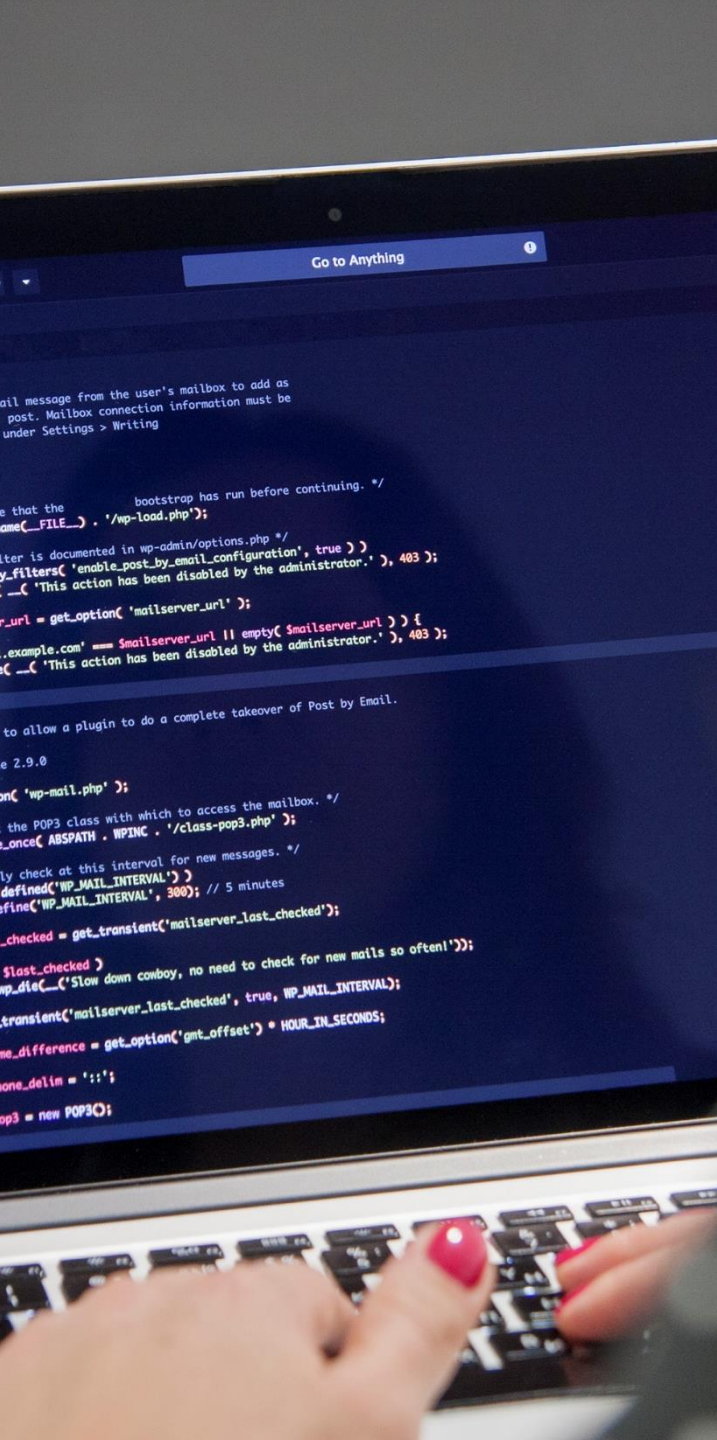
## Tabela de Operadores Lógicos

Operador	Descrição	Exemplo em Pseudocódigo	Exemplo em Python	Exemplo em JavaScript	Resultado
AND	E lógico	A AND B	A and B	A && B	verdadeiro se ambos A e B são verdadeiros
OR	OU lógico	A OR B	A or B	A    B	
NOT	Negação lógica	NOT A	not A	!A	Inverte o valor lógico de A
XOR	OU exclusivo	A XOR B	A ^ B (Bitwise)*	A ^ B (Bitwise)*	verdadeiro se apenas um dos valores A ou B é verdadeiro

## Tabela Verdade para Operadores Lógicos em Pseudocódigo

A	B	A AND B	A OR B	NOT A
Verdadeiro	Verdadeiro	Verdadeiro	Verdadeiro	Falso
Verdadeiro	Falso	Falso	Verdadeiro	Falso
Falso	Verdadeiro	Falso	Verdadeiro	Verdadeiro
Falso	Falso	Falso	Falso	Verdadeiro





# Prioridade entre os operadores

OPERADOR ARITMÉTICO	PRIORIDADE
Exponenciação	3 (maior)
Multiplicação	2
Divisão	2
Adição	1
Subtração	1 (menor)

OPERADOR LÓGICO	PRIORIDADE
e	3
ou	2
nao	1

OPERADOR	PRIORIDADE
Operadores aritméticos	3
Operadores relacionais	2
Operadores lógicos	1

# Pseudocódigo Entrada e Saída de Dados

```
1 Algoritmo "Nome do Algoritmo"
2 // Disciplina : Técnicas e Lógica de Programação
3 // Professor  : Flavio Mota da Cruz
4 // Descrição  : Flavio Mota da Cruz
5 // Autor(a)   : Flavio Mota da Cruz
6 // Data atual :
7
8 Var
9
10 Not1, Not2:real
11 Media:real // Exemplos de variaveis
12 Inicio
13
14
15 Leia(Not1) // Comando para ler a variável Not1
16 Leia(Not2) // Comando para ler a variável Not2
17
18 Media <- (Not1 + Not2) / 2 // Processamento de dados
19
20 Escreva(Media) // Exemplo de saída de dados
21 Escreva("A média das notas é = ", Media) // Exemplo de saída de dados
22
23 Fimalgoritmo
```

Em pseudocódigo os comandos que representam a entrada e saída de dados são: Leia() e Escreva()

Comandos de ENTRADA e SAIDA

LEIA(Variavel) ou Leia(Variavel), vamos usar um exemplo no VisualG

```
1 Algoritmo "Nome do Algoritmo"
2 // Disciplina : Técnicas e Lógica de Programação
3 // Professor : Flavio Mota da Cruz
4 // Descrição :
5 // Autor(a) : Flavio Mota da Cruz
6 // Data atual :
7
8 Var
9
10 Nota1, Nota2: real // Exemplos de variaveis
11
12 Inicio
13
14
15 Leia(Nota1) // Comando para ler a variável Nota1
16 Leia(Nota2) // Comando para ler a variável Nota2
17
18 Fimalgoritmo
```



No VisualG 3.0 usamos o comando **ESCREVA** conforme o exemplo abaixo.

```
1 Algoritmo "Nome do Algoritmo"
2 // Disciplina : Técnicas e Lógica de Programação
3 // Professor : Flavio Mota da Cruz
4 // Descrição :
5 // Autor(a) : Flavio Mota da Cruz
6 // Data atual :
7
8 Var
9
10 Notal, Nota2: real // Exemplos de variaveis
11 Media: real
12 Inicio
13
14
15 Leia(Notal) // Comando para ler a variável Notal
16 Leia(Nota2) // Comando para ler a variável Nota2
17
18 Media <- (Notal + Nota2) / 2 // Processamento de dados
19
20 Escreval(Media) // Exemplo de saída de dados
21 Escreva("A média das notas é = ", Media) // Exemplo de saída de dados
22
23
24 Fimalgoritmo
```

# RECODE

Institucional



/rederecode



/recoderede