Lógica, Algoritmos e Prática com JavaScript

Marcle Rodrigues

O que vamos ver hoje?

- Introdução à Algoritmos
- Formas de Representação
- Tipos de Dados
- Vetores
- Variáveis
- Expressões
- Entrada, Saída, Atribuição
- Estruturas de Controle e Repetição
- Funções

1. Introdução à Algoritmos

Características:

- **Definição**
  Bem denifidos, objetivando a clareza e evitando a ambiguidade
- **Finitude**
  Sempre tem um fim, deve ter um número finito de passos
- **Efetividade**
  Suas operações deve sem básicas o suficiente para serem executadas de maneira exata e em um tempo finito

1. Introdução à Algoritmos

Características:

- **Entradas**
  É tudo o que o algoritmo precisa para realizar a sua função. Pode ter zero ou mais.
- **Saídas**
  Resultado da operação realizada. Pode ter uma ou mais.

Formas de Representação

Dica

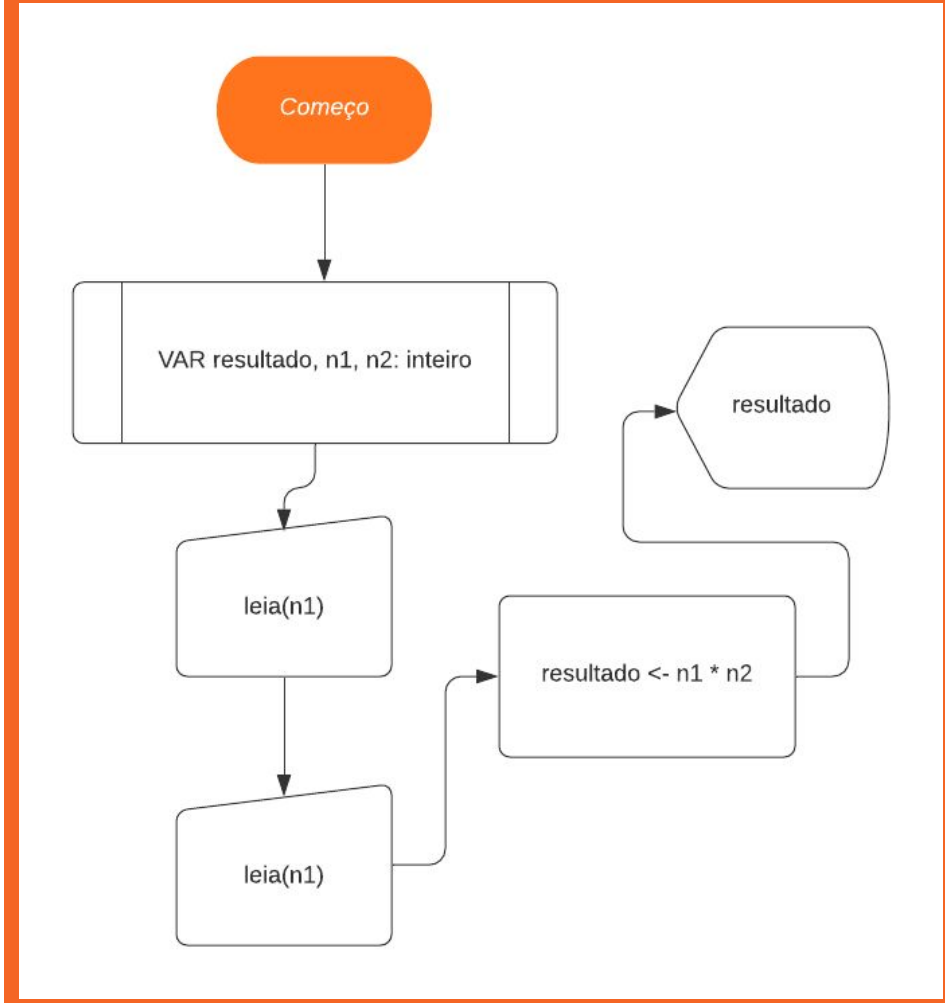
Importante: Existem 3, precisamos dominar apenas 1, porém ajuda e muito saber quais as outras formas por algumas são complementares.

Descrição Narrativa

Problema: Escreva um algoritmo que realize a soma de dois números.

- Receber os dois números que será multiplicados
- Multiplicar os dois números
- Mostrar o resultado da multiplicação

Fluxograma


Começo
VAR resultado, n1, n2: inteiro
leia(n1)
leia(n1)
resultado <- n1 * n2
resultado

Pseudo-Código

```
INICIO ALGORITMO
  VAR resultado, n1, n2: inteiro

  LEIA(n1)
  LEIA(n2)

  resultado <- n1 * n2

  ESCREVA(resultado)
FIM ALGORITMO
```

Método Chinês

Memória Ram e Váriaveis

Identificadores

Memória Ram e Variáveis

Variáveis

Nome, Tipo de Dado e uma Informação.

Tipos de Dados

Numéricos: Inteiros

Sãos os números naturais: 1, 2, 3, 4.

Numéricos: Reais

Sãos os números fracionários: 1.4, 2.5, 3.7, 4.33.

Dados: Literais

Conhecidos como Strings ou Cadeira de Caracteres.

Exemplo: “Marcle”, “Eu, Programador”

Dados: Lógicos

Conhecidos como Booleanos, podendo ser verdadeiro ou falso.

Exemplo: true, false.

Dados: Vetores

Conhecidos como Arrays. Armazenam um conjunto de valores.

Exemplo: [1, 2, 3, 4, 5]

Expressões

Podemos entender como se fossem fórmulas.

Exemplo: media = n1 + n2

Operadores

São elementos que atuam sobre operandos.

Exemplo:

3 + 5

Operandos: 3 e 5
Operador: +

Operadores: Binários

Atuam sobre dois operandos

Exemplo: +, - , *, /, %

Operadores: Unários

Atuam sobre apenas 1 operando

Exemplo: -, !

Expressões: Aritméticas

Resultado é do tipo numérico

Exemplo:

+, -, *, /, **, %

Expressões: Lógicas

Resultado é do tipo booleano

Exemplo:

E, OU, NÃO

Operadores: Relacionais

Utilizados para realizar comparações.

Exemplo: ==, !=, <, >, <=, >=

Importante

Comparações só podem ser feitas em objetos de mesma natureza.

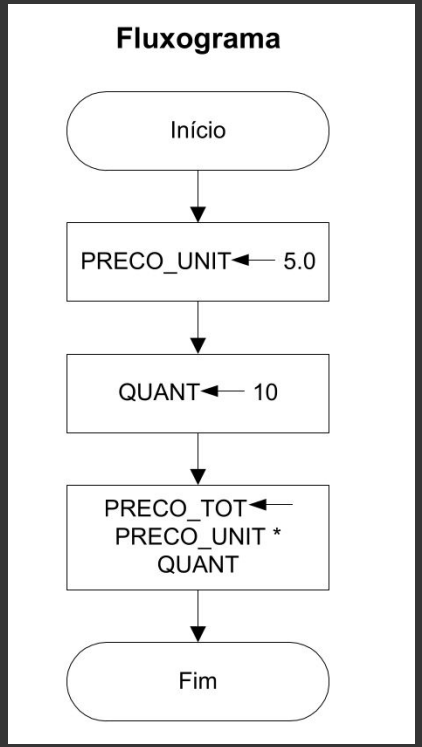
Instruções Primitivas

Comando básicos que executam tarefas essenciais.

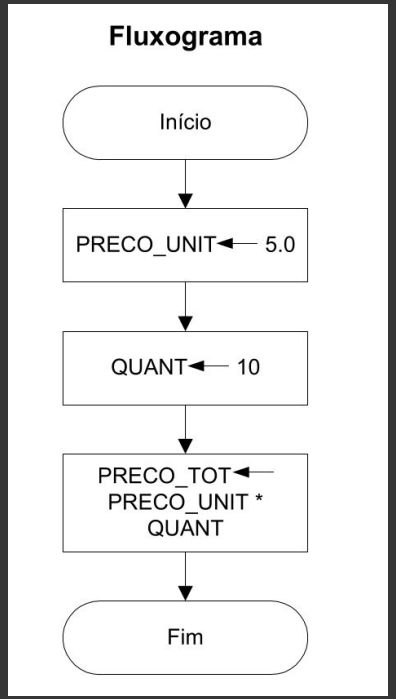
Instruções: Atribuição

Usada para armazenar valor em uma variável

[variavel] <- [expressao]



Instruções: Atribuição



Instruções: Atribuição

Pseudocódigo

```
Algoritmo EXEMPLO_6.1
Var  PRECO_UNIT, PRECO_TOT : real
     QUANT : inteiro
Início
     PRECO_UNIT ← 5.0
     QUANT ← 10
     PRECO_TOT ← PRECO_UNIT * QUANT
Fim.
```

Instruções: Saída de Dados

São utilizadas para enviar as informações que temos na memória para dispositivos de saída e para que o usuário possa visualiza-lá.

escreva([expressao])

Instruções: Entrada de Dados

São utilizadas para permitir o usuário que insira dados dentro do programa.

leia([expressao])

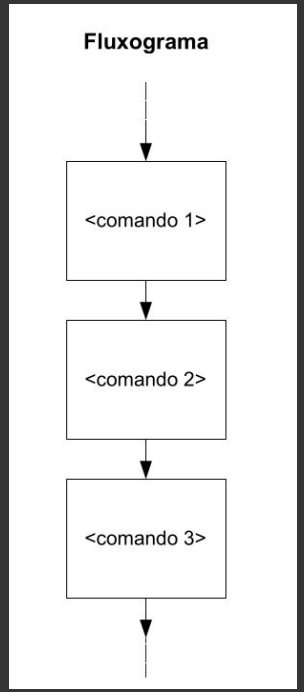
Controle do Fluxo

Como fazemos para permitir que decisões sejam tomadas e ações possam ser tomadas de maneira automática?

Estrutura Sequencial

Comandos são executados em uma ordem pré-estabelecida.

Estrutura Sequencial



Estruturas de Decisão

Alteram o fluxo da execução.

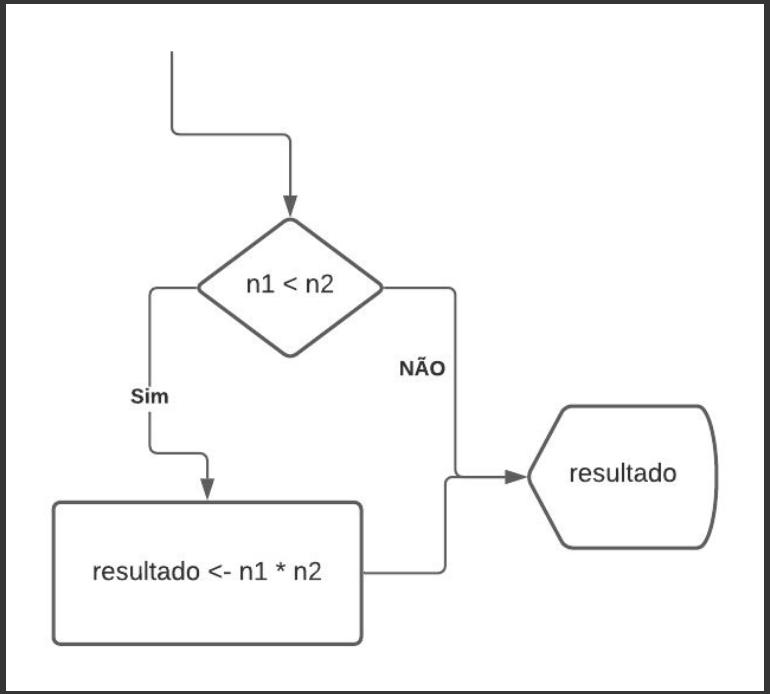
Estruturas de Decisão

Alteram o fluxo da execução.

Se... Então...

```
INICIO ALGORITMO
  VAR resultado, n1, n2: inteiro

  resultado <- 0

  LEIA(n1)
  LEIA(n2)

  SE n1 < n2 ENTAO
    resultado <- n1 * n2
  FIM SE


  ESCREVA(resultado)
FIM ALGORITMO
```

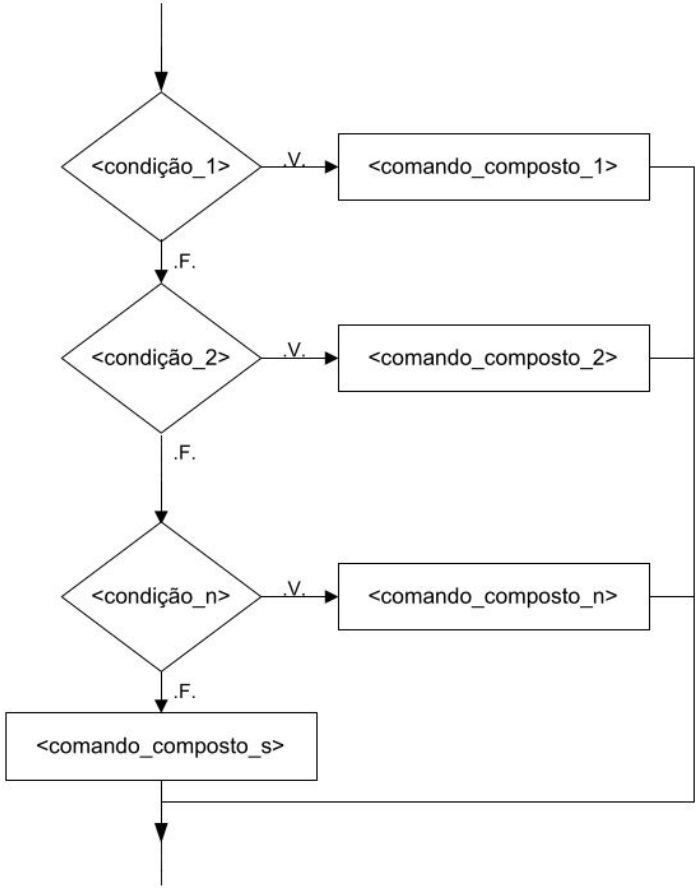
Se... Então...



Escolha... Caso... Senão...

```
INICIO ALGORITMO
  VAR resultado, n1, n2: inteiro

  resultado <- 0

  LEIA(n1)
  LEIA(n2)

  ESCOLHA
    CASO n1 > n2
      resultado <- n1 * n2
    CASO n1 === n2
      resultado <- n1 * n1
    SENÂO
      resultado <- n1
  FIM ESCOLHA

  ESCREVA(resultado)
FIM ALGORITMO
```

Fluxograma


<condição_1>
.V.
<comando_composto_1>
.F.
<condição_2>
.V.
<comando_composto_2>
.F.
<condição_n>
.V.
<comando_composto_n>
.F.
<comando_composto_s>

Estruturas de Repetição

Conhecidos como laços, executam uma operação até que uma condição seja satisfeita.

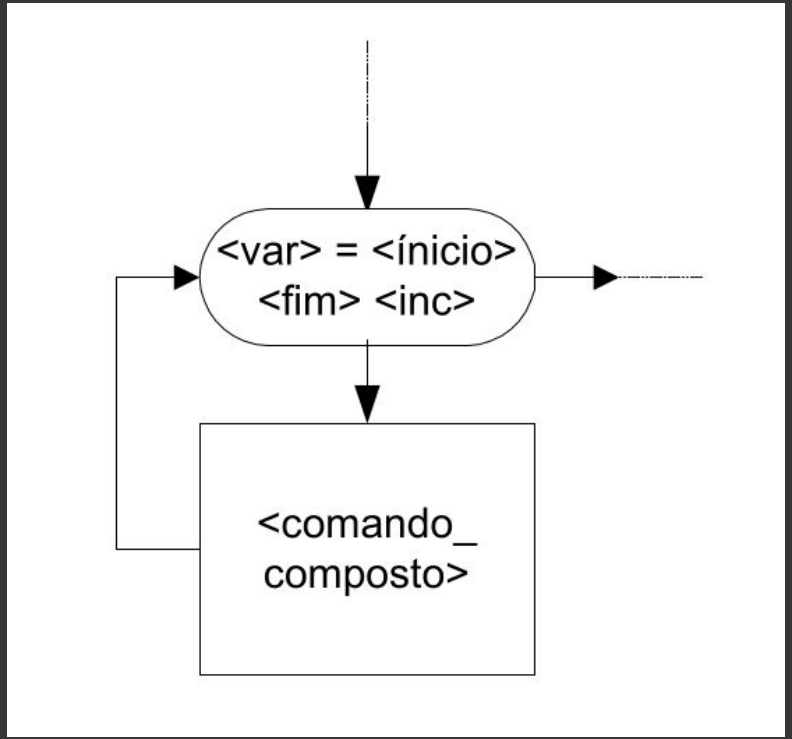
Laços Contados

Se conhece previamente o número de vezes que o comando vai ser executado.

Para... de... até... faça...

Conhecido como laço “for”, executa um comando até que o número de execuções seja atingido.

```
INICIO ALGORITMO
  VAR contador: inteiro

  PARA contador DE 0 até 10 FACA
    ESCREVA("Programação é TOP")
  FIM PARA
FIM ALGORITMO
```


<var> = <ínicio>
<fim> <inc>
<comando_
composto>

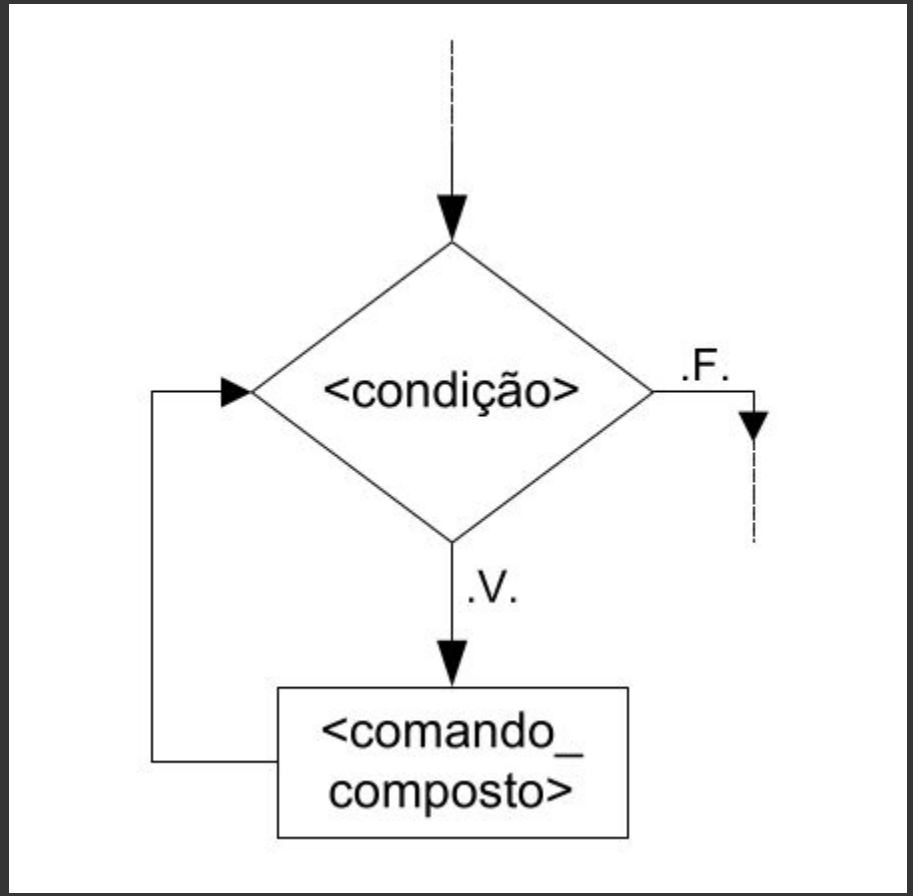
Laços Condicionais

Não se conhece o número de vezes, a quantidade varia de acordo com a lógica e as condições do algoritmo.

Enquanto

Conhecido como laço “while”, executa um comando até que a condição seja satisfeita.

```
INICIO ALGORITMO
  VAR executar: lógico
  VAR contador: inteiro

  contador <- 0

  ENQUANTO executar FACA
    contador <- contador + 1

    SE contador > 3 ENTAO
      executar <- false
    END
  FIM ENQUANTO
FIM ALGORITMO
```


<condição>
.F.
.V.
<comando_
composto>

Funções

Funcionam como subprogramas, são utilizadas para evitar repetição de código.

```
INICIO ALGORITMO
  FUNCAO media(n1, n2: inteiro): inteiro
  INICIO
    VAR media: inteiro

    media <- (n1 + n2) / 2

    RETORN media
  FIM FUNCAO

  VAR media: inteiro

  media <- media(10, 8)

  ESCREVA(media)
FIM ALGORITMO
```

Exercícios

Problemas

Obrigado