/codifica

Formação Desenvolvedor Full Stack Júnior

Linguagem de programação e Introdução a lógica de Programação

Como funcionam as linguagens de programação?

Compilada X Interpretada

- Convertida diretamente à linguagem de máquina
- Mais rápidas e eficientes
- Necessário *Build*
- C, C++, Rust

- Interpretada linha à linha
- Mais lenta
- Não é necessário *Build*
- *JIT (Just in Time)*
- *JavaScript, PHP, Python*

Baixo nível X Alto nível

são as linguagens mais próximas ao hardware do computador. São compostas por instruções que facilmente serão compreendidas pelo computador, mas que são difíceis para os humanos entenderem.

São linguagens voltadas para o ser humano. Em geral utilizam sintaxe estruturada tornando seu código mais legível. Necessitam de compiladores ou interpretadores para gerar instruções ao processador.

Qual Linguagem nós vamos utilizar no módulo 01?

Mas em qual ambiente ?

Introdução à lógica de programação
O que é um algoritmo?

O que é um algoritmo?

"São conjuntos de passos finitos e organizados que, quando executados, resolvem um determinado problema."
Manzano, José Augusto NG. Algoritmos lógica para desenvolvimento de programação de computadores. Saraiva Educação SA, 2010.

"Visa fixar um padrão de comportamento de forma clara e sem ambiguidade, com a melhor relação custo benefício e com o objetivo de solucionar um problema"

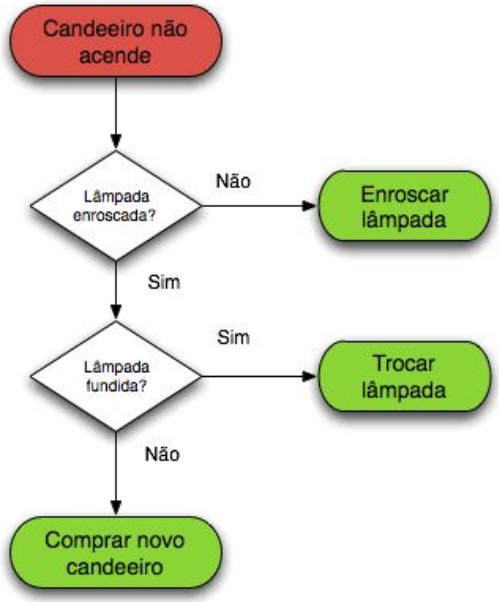
"Algoritmo é a descrição de uma sequência de passos que deve ser seguida para a realização de uma tarefa"
(Ascencio 1999 apud Ascencio 2002).

O que é a lógica?

A lógica em filosofia, a grosso modo, procura saber porque pensamos assim e não de outra forma, nos ensinando a usar corretamente as leis do pensamento, a ordenar nossos pensamentos, visto que os mesmos podem trabalhar desordenadamente

Exemplo básico: todo mamífero é um animal. Todo cavalo é mamífero. Logo, todo cavalo é animal.

Formas de se representar algoritmos




```
1   Programa Calcula_Media
2   Var
3           RESULTADO: caractere
4           N1, N2, N3, N4, SOMA, MEDIA: real
5   Início
6           Leia (N1,N2, N3, N4)
7           SOMA ←— N1+ N2 + N3 + N4
8           MEDIA ←— SOMA / 4
9           Se (MÉDIA >= 7)
10                  Então RESULTADO ←— "Aprovado"
11                  Senão RESULTADO ←— "Reprovado"
12          Fim-Se
13          Escreva ("Nota 1: ", N1)
14          Escreva ("Nota 2: ", N2)
15          Escreva ("Nota 3: ", N3)
16          Escreva ("Nota 4: ", N4)
17          Escreva ("Soma: ", SOMA)
18          Escreva ("Media:  ", MEDIA)
19          Escreva ("Resultado: ", RESULTADO)
20  Fim
```

/codifica

Métodos para construção de um bom algoritmo

- **Ler atentamente o enunciado**: é justamente o enunciado do exercício que fornece o encaminhamento necessário à resolução do problema.

- **Retirar do enunciado a relação das entradas de dados**: a entrada é o meio pelo qual o usuário pode informar dados que serão utilizados pelo programa em seu processamento.

- **Retirar do enunciado a relação das saídas de dados**: para que o usuário possa ter acesso aos resultados do processamento do programa, toda linguagem de programação fornece mecanismos de apresentação (saída) dos dados.

- **Determinar o que deve ser feito para transformar as entradas determinadas nas saídas especificadas**: nesta fase é que teremos a construção do algoritmo propriamente dito, pois, a partir de alguns requisitos especificados, devemos determinar qual seqüência de ações é capaz de transformar um conjunto definido de dados nas informações de resultado.

Exercício 1

Calcular a média final dos alunos de uma disciplina. Os alunos realizaram 2 provas: P1 e P2.
Onde a média final = (P1+P2) / 2. Para montar o algoritmo proposto faremos 3 perguntas:

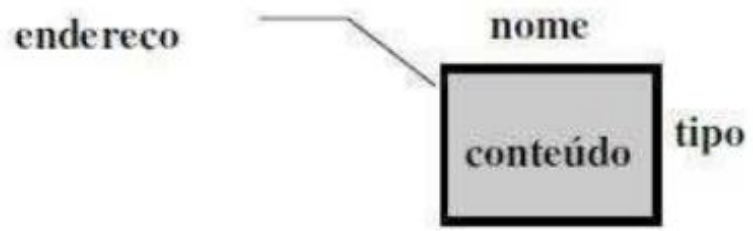
1) Quais são os dados de entrada?
2) Qual será o processamento?
3) Quais serão os dados de saída?

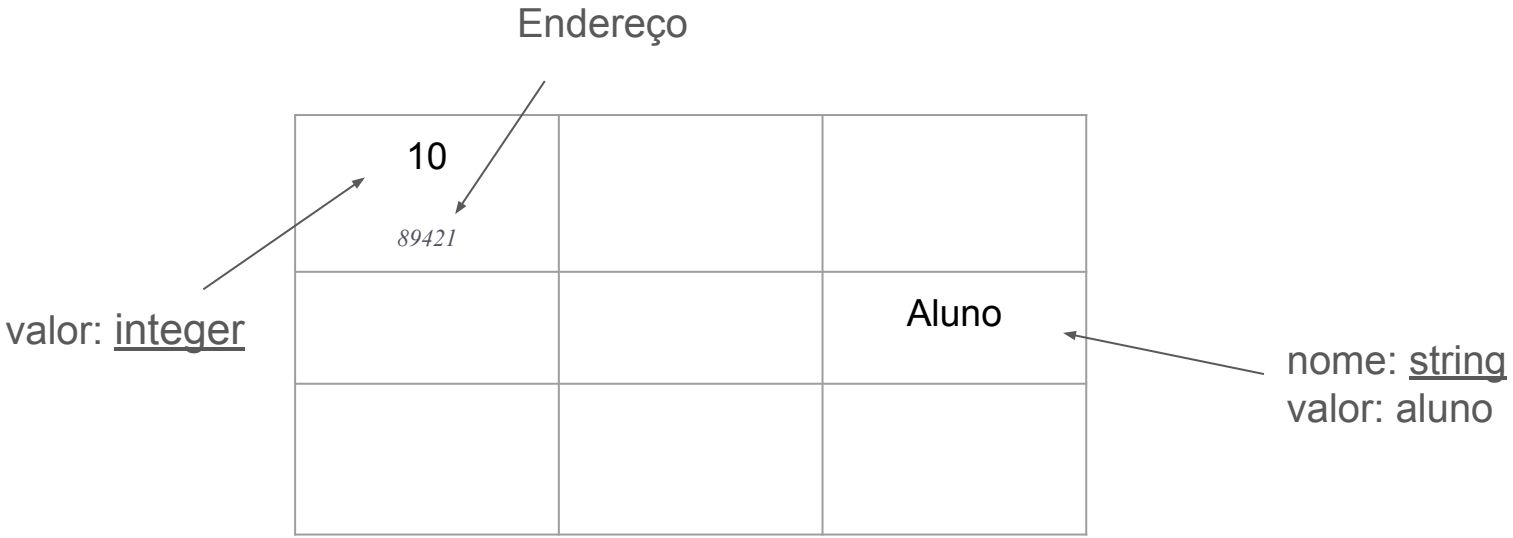
Exercício 1 - Resposta

Calcular a média final dos alunos de uma disciplina. Os alunos realizaram 2 provas: P1 e P2.
Onde a média final = (P1+P2) / 2. Para montar o algoritmo proposto faremos 3 perguntas:

1) Quais são os dados de entrada? **P1 e P2.**
2) Qual será o processamento? **Somar os dados de entrada e dividir o resultado por 2 (P1+P2)/2.**
3) Quais serão os dados de saída? **O resultado do processamento, ou seja, a média final.**

Variáveis


endereco
nome
conteúdo
tipo


Endereço
10
89421
valor: integer
Aluno
nome: string
valor: aluno

/codifica

Variáveis e suas características

- Tem um nome, que a diferencia das demais
- Tem um tipo de dado associado, que indica o tipo de informação que poderá ser armazenada na variável
- Tem um conteúdo, que é o dado guardado na variável
- Tem um endereço, que a localiza na memória

Variáveis - Observações importantes

- Nome de uma variável é único em um algoritmo e deve seguir as regras de formação de identificadores
- Uma variável somente pode receber como conteúdo um dado do tipo que foi definido para ela
- O conteúdo de uma variável é SUBSTITUÍDO por outro conteúdo que venha a ser colocado na variável
- O nome e o tipo de dado de uma variável, uma vez definidos, não mudam por todo o algoritmo
- O uso do nome de uma variável em uma expressão significa o uso do seu conteúdo (naquele momento) dentro da expressão
- O uso de um conteúdo de variável em uma expressão não modifica o seu valor.

Regras para nomeação de identificadores

- Devem começar com um caracter alfabético;
- Podem ser seguidas por mais caracteres alfabéticos e/ou numéricos;
- Não é permitido o uso de outros caracteres especiais;
- Não poderá ser nome de uma variável, uma palavra reservada a uma instrução do programa;
- O nome de uma variável não poderá possui espaços em branco;
- Não poderão ser utilizados outros caracteres a não ser letras e números, exceto o uso do caracter especial ‘sublinha’ (), e/ou underline;

Tipos de dados em Javascript

Primitivos (em JS)

- **STRING** (Cadeia de Caracteres): representam texto e são definidas utilizando aspas simples ou duplas
- **NUMBER** (Número): representa valores numéricos, podendo esse ser inteiro ou real
- **BOOLEAN** (Booleano): representam verdadeiro ou falso e são utilizados em lógicas condicionais
- **UNDEFINED e NULL** (indefinido e nulo): undefined representa uma variável que foi declarada mas não foi inicializada e null é utilizado para representar a ausência de valor

Operadores Matemáticos

Operador	Nome	Propósito	Exemplo
`+`	Adição	Adiciona um número a outro.	`6 + 9`
`-`	Subtração	Subtrai o número da direita do número da esquerda.	`20 - 15`
`*`	Multiplicação	Multiplica um número pelo outro.	`3 * 7`
`/`	Divisão	Divide o número da esquerda pelo número da direita.	`10 / 5`
`%`	Restante *(Remainder* - as vezes chamado de modulo)	Retorna o resto da divisão em números inteiros do número da esquerda pelo número da direita.	`8 % 3` (retorna 2; como três cabe duas vezes em 8, deixando 2 como resto.)

/codifica

`+=`	Atribuição de adição	Adiciona o valor à direita ao valor da variável à esquerda e, em seguida, retorna o novo valor da variável	`x = 3; x += 4;`	`x = 3; x = x + 4;`
`-=`	Atribuição de subtração	Subtrai o valor à direita do valor da variável à esquerda e retorna o novo valor da variável	`x = 6; x -= 3;`	`x = 6; x = x - 3;`
`*=`	Atribuição de multiplicação	Multiplica o valor da variável à esquerda pelo valor à direita e retorna o novo valor da variável	`x = 2; x *= 3;`	`x = 2; x = x * 3;`
`/=`	Atribuição de divisão	Divide o valor da variável à esquerda pelo valor à direita e retorna o novo valor da variável	`x = 10; x /= 5;`	`x = 10; x = x / 5;`

/codifica

Operadores Lógicos e Relacionais

Operadores Relacionais

Operador	Função	Operador	Função
==	Igual a	!=	Diferente
>	Maior que	>=	Maior ou igual a
<	Menor que	<=	Menor ou igual a
===	Idêntico a	!==	Diferente estrito

Operadores Lógicos

Símbolo	Sintaxe em JavaScript	Função
E	&&	Conjunção
Ou	||	Disjunção
Não	!	Negação

Tabela Verdade

Operador lógico “e” ^		
p	q	p ^ q
V	V	V
V	F	F
F	V	F
F	V	F

operador lógico “ou” v		
p	q	p v q
V	V	V
V	F	V
F	V	V
F	F	F

Operador “não” ~	
p	~p
F	V
F	V

/codifica

Estruturas de Controle: Condicionais

O que são estruturas de controle condicionais

"São estruturas que irão impor condições à sequência do nosso algoritmo, as quais podem definir ou não um desvio na execução do mesmo, baseado na decisão que será tomada."

Estrutura de controle condicional simples

COMANDO CONDICIONAL IF

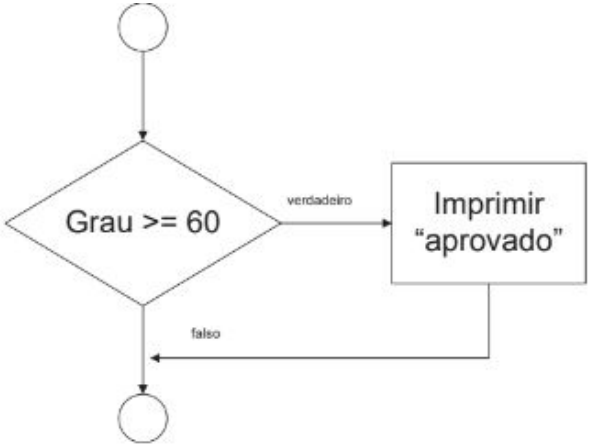
If (<condição>)
<instruções para condição verdadeira> ;
else <instruções para condição falsa> ;

<instruções para condição falsa ou após condição ser verdadeira>

Blocos de Comando

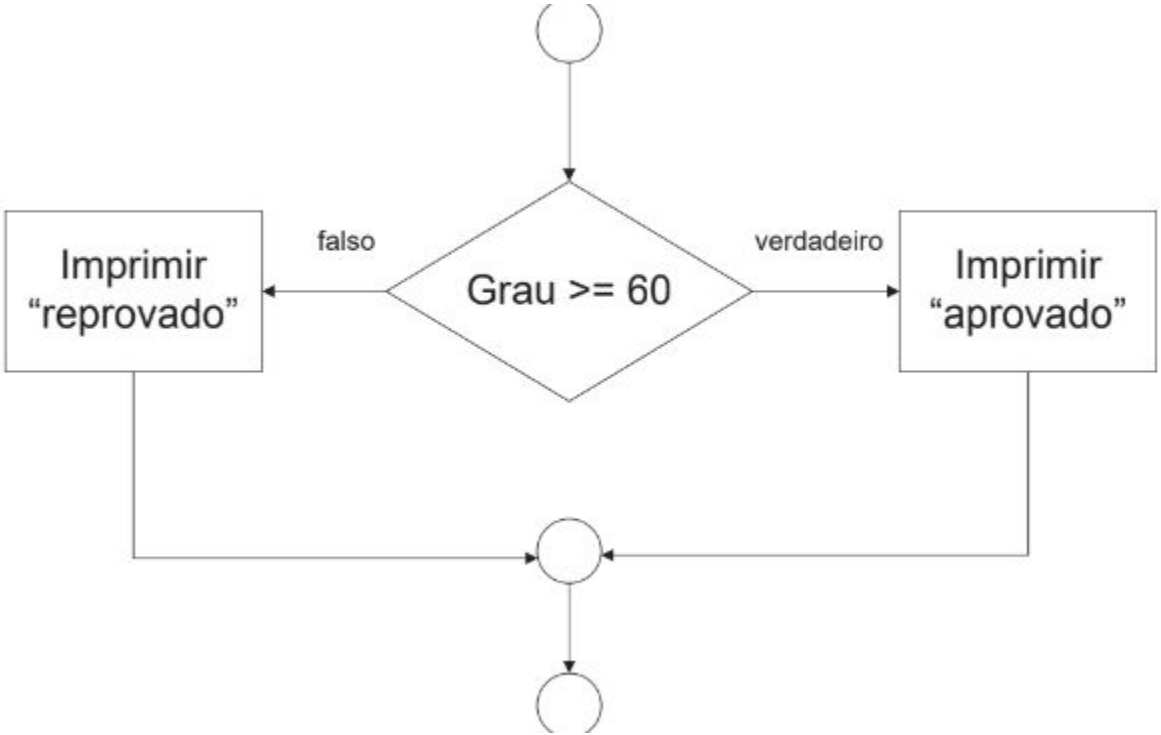
if (<condição>)
{ // Inicio
<instrução1 para condição verdadeira>;
<instrução2 para condição verdadeira>;
} // Fim

<instruções para condição falsa ou após condição ser verdadeira>



/codifica

Estrutura de controle condicional composta



Estrutura de seleção switch-case

O comando switch é um comando de seleção que permite selecionar um comando entre vários outros comandos. Isto é feito através da comparação de uma variável a um conjunto de constantes. Cada um dos comandos está ligado a uma constante.

```
switch (variável)
{
        case constante_1 :    seqüência de comandos;
                              break;
        case constante_2 :    seqüência de comandos;
                              break;
        .
        .
        .
        default: seqüência de comandos;
}
```

/codifica