

Micro Apostila

Programação em Javascript

José de Figueiredo

13 de maio de 2023

Conteúdo

1	Introdução	2
2	Controle de Fluxo	3
3	Laços de Repetição	4
4	Laço While	4
4.1	Problema 4.1	5
4.2	Problema 4.2	7
5	Laço Do-While/Do-Until	14
5.1	Problema 5.1	14
5.2	Problema 5.2	17
6	Tipo de laço	18
6.1	Problema ??	18

1 Introdução

Esta micro apostila apresenta o básico da programação com a linguagem Javascript.

Os conceitos da construção de programas são apresentados, primeiramente de forma macro, utilizando diagramas de fluxo e pseudo-código. Por fim, os exemplos são apresentando em JavaScript.

Este trabalho é uma micro apostila, portanto, fica implícito que o documento apresenta apenas o básico. Em outras palavras, outras formas ou métodos (normalmente mais avançadas) de programação podem ser implementadas.

Por esse motivo, o leitor deverá fazer suas pesquisas e ampliar horizontes sobre as técnicas construção de programas usando Javascript.

2 Controle de Fluxo

Em construção....

3 Laços de Repetição

Frequentemente precisamos construir algoritmos que repitam operações. Laços, são estruturas que controlam estas repetições. Cada repetição de um laço é chamada *ITERAÇÃO*.

Os laços podem repetir operações (iterar) por uma quantidade determinada de vezes. Por exemplo:

- De 1 até 10: um laço neste caso irá repetir 9 vezes;
- De 1 até 10, inclusive: neste caso o laço repete 10 vezes, porque o 10 foi incluído na contagem;
- De 0 até 1000: neste caso o laço repete 1000 vezes, de 0 até 999.
- De -10 até 10: neste caso o laço repete 20 vezes.

Importante compreender que as repetições podem ocorrer nos mais diversos cenários e modos, não limitando-se aos exemplos da lista acima.

Para trabalhar com repetição, é preciso compreender o básico das técnicas de controle de repetição, ou laços de repetição. Nesta sessão, vamos conhecer as principais técnicas para controle de repetição presentes nas linguagens Javascript e Octave.

4 Laço While

A palavra *while* pode ser traduzida para enquanto.

-> -> *A instrução while executa um laço do tipo “enquanto-faça”* <- <-

Nesta técnica repetimos um conjunto de instruções enquanto uma determinada condição for verdadeira.

Utilizamos *while* quando queremos que um determinado bloco de código, seja executado ENQUANTO uma **condição** seja VERDADEIRA. No

momento em que a **condição** for FALSA, o bloco de código deixa de ser repetido (ou finaliza).

O fluxograma na Figura 1 a seguir demonstra graficamente este tipo de repetição.



Figura 1: Fluxograma demonstrando um laço de repetição.

Para demonstrar o uso desta estrutura de laço, apresentamos 3 exemplos simples. As soluções dos problemas serão demonstradas usando representação gráfica de fluxograma; representação em Pseudocódigo; com Javascript e com Octave.

4.1 Problema 4.1

Escrever um algoritmo/programa para *mostrar na tela os números de 1 até 10, usando repetição*.

4.1.1 Fluxograma para solução do problema 4.1

A Figura 2 mostra um algoritmo, que resolve este problema, representado em fluxograma.

Para resolver o problema criamos uma variável (i) com valor inicial 1; no losango representamos a verificação do laço, onde o algoritmo testa se

a variável i ainda é menor ou igual a 10; se i ainda é menor/igual a 10, o programa mostra o valor de i e faz com que a variável i receba o valor de i mais 1 (chamamos isso de incremento de variável). Caso o teste de i seja maior que 10, então o fluxo é desviado para o fim do programa.

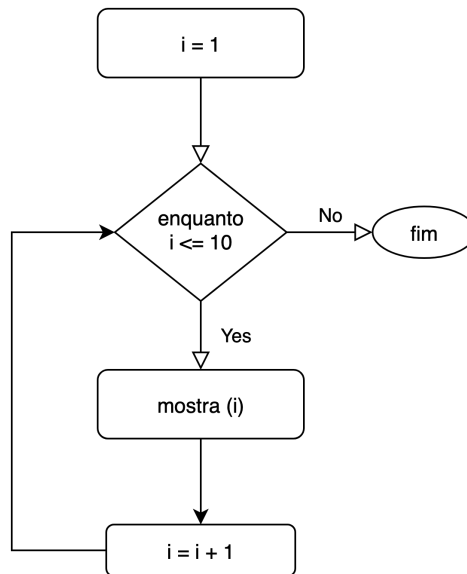


Figura 2: Fluxograma representando a solução do problema 4.1.

4.1.2 Pseudocódigo

Uma solução usando Pseudocódigo.

```
1 | Início
2 | var i = 1
3 | enquanto i <= 10, repita:
4 |     mostra i
5 |     i = i + 1
6 | fim-enquanto
7 | Fim
```

4.1.3 Javascript

Uma solução usando JavaScript.

```
1 | var i = 1;
2 | while(i <= 10) {
```

```

3 |   print(i);
4 |   i = i + 1;
5 | }

```

4.1.4 Octave

Uma solução usando Octave.

```

1 | clear
2 | i = 1
3 | while (i <= 10)
4 |     printf(" %d ", i)
5 |     i++
6 | endwhile

```

Observe que a estrutura de uso da instrução `while` é semelhante nas linguagens JavaScript e Octave. No código em Octave, a instrução `i++` é equivalente a instrução `i = i+1`.

Escolha uma das linguagens e execute o trecho observando o teste que ocorre dentro do parênteses. Observe também o resultado se tirarmos o sinal de “=” o laço repete de 1 até 9.

4.2 Problema 4.2

Escrever um algoritmo/programa que leia 25 números, calculando e mostrando a média aritmética.

Fluxograma para solução do problema 4.2

A Figura 3 apresenta uma solução, na forma de fluxograma, para o problema 4.2.

No primeiro elemento do fluxograma, são definidas 4 variáveis, que são:

- *sum*, com valor 0. Esta variável será responsável por armazenar (acumular) o valor da soma dos números digitados.

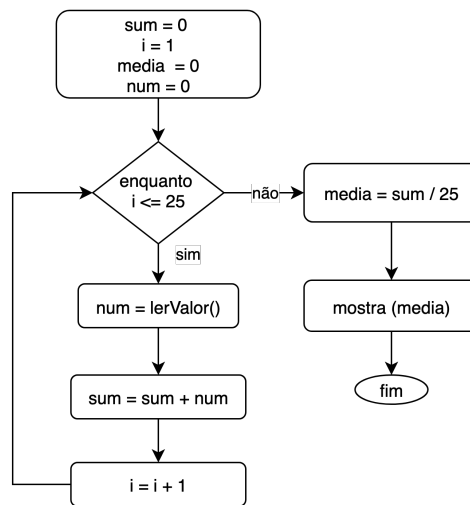


Figura 3: Fluxograma representando a solução para o 4.2.

- *i*, com valor 1. Esta variável servirá responsável pelo controle de quantas vezes o laço está sendo repetido. Lembrando que será necessário ler 25 números, portanto o laço vai precisar repetir pelo menos 25 vezes.
- *media*, com valor 0. Esta variável será usada para armazenar o cálculo da média.
- *num*, com valor 0. Esta variável será usada para armazenar cada valor lido. Importante, que cada valor lido, depois de somado pode ser descartado. Logo esta variável é reaproveitada a cada iteração do laço.

O losango representa o laço que executa enquanto *i* for menor ou igual a 25 (e que encerra quando *i* for maior que 25).

Enquanto *i* for menor que 26, serão executados as instruções:

- *num = lerValor()*; esta instrução lê um número digitado pelo teclado e armazena na variável *num*;

- $sum = sum + num$; esta instrução guarda em na variável `sum`, o valor de `sum` somado ao valor `num`;
- $i = i + 1$; esta instrução incrementa o valor de `i` para que se controle as 25 repetições dentro deste laço;

Uma outra forma de fazer analisar este algoritmo é usando uma tabela. Para isso, vamos observar a Tabela 1. Para não ficar muito longa, a tabela foi dividida ao meio montada lado a lado. Observe que a primeira coluna esta apenas dizendo se o programa ainda está dentro do laço - ele fica dentro do laço até que $i=26$. Cada linha da tabela, **enquanto o programa está dentro do laço**, é chamada de iteração. Os números na coluna `num` foram gerados aleatoriamente, simulando números que o usuário digitou.

O valor na célula `sum` é obtido conforme a equação $sum = sum + num$, ou seja, o valor nela é sempre dado pela soma de `sum` da linha anterior com o valor `num` da linha corrente.

Vamos ver a iteração em que $i=4$: ao iniciar esta iteração, o valor de `sum` é 23. O usuário digitou 7 e como $sum = sum + num$, ao final da iteração `sum` recebe a soma entre o valor que tinha antes (no inicio da iteração) com o valor que o usuário acabou de digitar.

Importante: Os valores das variáveis **somente são mostrados na tela** depois que o laço acaba e a média é calculada. A Tabela 1 representa a memória usada pelo programa durante os passos da execução.

Pseudocódigo

Um algoritmo em pseudocódigo.

```

1 | Inicio
2 | var sum = 0, i = 1, num=0, media=0
3 | enquanto i <= 25, repita:
4 |     num = lerValor()
5 |     sum = sum + num
```

Tabela 1: Um teste de mesa para a solução do problema 4.2.

Laço		Variáveis				Laço					
dentro?	i	num	sum	media		dentro?	i	num	sum	media	
sim	0	0	0	0		sim	14	8	82	0	
sim	1	10	10	0		sim	15	1	83	0	
sim	2	5	15	0		sim	16	10	93	0	
sim	3	8	23	0		sim	17	2	95	0	
sim	4	7	30	0		sim	18	1	96	0	
sim	5	10	40	0		sim	19	10	106	0	
sim	6	3	43	0		sim	20	1	107	0	
sim	7	4	47	0		sim	21	9	116	0	
sim	8	8	55	0		sim	22	4	120	0	
sim	9	4	59	0		sim	23	5	125	0	
sim	10	6	65	0		sim	24	6	131	0	
sim	11	3	68	0		sim	25	3	134	0	
sim	12	2	70	0		não	26	3	137	5,48	
sim	13	4	74	0							

```

6 |         i = i + 1
7 | fim-enquanto
8 | media = sum / 25
9 | mostra(media)
10| Fim

```

JavaScript

Um programa em Javascript.

```

1 | var i = 1, sum = 0, num, media;
2 | while( i <= 25){
3 |     num = parseInt(prompt());
4 |     sum = sum + num;
5 |     i = i + 1;
6 | }
7 | media = sum / 25;
8 | print(media);

```

Octave

Um programa em Octave.

```

1 | clear
2 | i = 1, sum = 0, num = 0, media = 0
3 | while ( i <= 5)
4 |     num = input(" digite um numero " )
5 |     sum = sum + num
6 |     i++
7 | endwhile
8 | media = sum / 5
9 | printf(" %d ",media)

```

Problema 4.2

Escrever um algoritmo/programa para ler um uma quantidade variável (que muda) de valores e apresentar a média aritmética.

Este problema poderia ser enunicado de outras formas, por exemplo:

1. Ler N valores e apresentar a média aritmética na tela; ou
2. Crie um programa que leia quantos números serão informados, depois leia estes números e faça a média.

Para criar a solução para este problema é preciso compreender que cada vez que o programa executar, poderemos ler uma quantidade de valores diferentes. Por exemplo: usando o mesmo programa quero fazer a média de notas da turma X (que tem 10 alunos) e em outro momento, quero fazer a média de notas dos alunos da turma Y (que tem 25 alunos). Para isso, precisamos de um programa que execute um laço de repetição de 1 até N, onde N é um valor informado pelo usuário.

Fluxograma para solução do problema 4.2

A Figura 4 apresenta uma solução, na forma de fluxograma, para o problema 4.2.

Nesta solução iniciamos o algoritmo criando variáveis *sum* com valor 0 e *i* com valor 1. Na segunda ação, é criada a variável repetir com valor

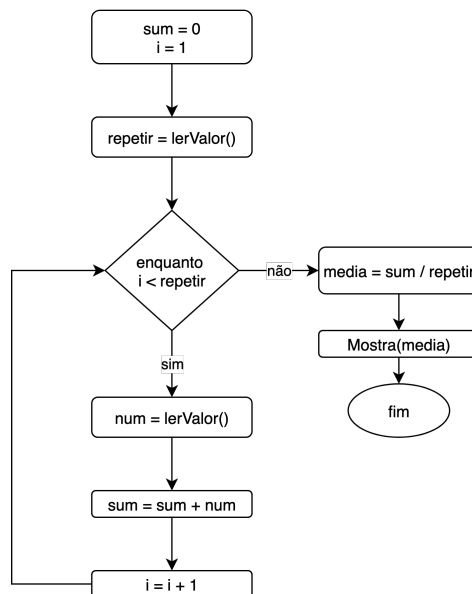


Figura 4: Fluxograma representando a solução pra o problema 4.2.

informado diretamente pelo teclado (esta variável irá determinar até quando o laço deverá repetir).

O próximo passo, no losango, temos o controle do laço - veja que o laço diz: *enquanto* $i < repetir$, ou seja, enquanto a variável i (que iniciou em 1) for menor que a quantidade de valores que preciso ler (*repetir*). Enquanto i for menor que *repetir*, o programa ficará preso no laço e fará:

- $num = lerValor()$, que carrega em num um valor digitado pelo teclado;
- $sum = sum + num$, que soma na variável sum todos valores digitados;
- $i = i + 1$, que incrementa o valor de i em 1.

Este laço irá se repetir até que i não seja mais menor que *repetir*. Quando isto acontecer teremos somados todos os valores digitados pelo teclado.

Quando o laço encerrar, vamos para o fluxo da direita, onde é calculado

a média em $media = sum/repetir$ e depois mostra a media na tela em *Mostra(media)*.

Pseudocódigo

Uma solução em pseudocódigo.

```
1 | Inicio
2 | var repetir , valor , media , i =0, sum = 0 ;
3 | repetir = lerValor()
4 | enquanto i < repetir , repita:
5 |     valor = lerValor();
6 |     i = i + 1;
7 |     sum = sum + valor;
8 | fim-enquanto
9 | media = sum/repetir
10 | mostrar(media)
11 | Fim
```

JavaScript

Uma solução em JavaScript.

```
1 | var repetir , valor , media , i=1, sum = 0;
2 |
3 | repetir = parseInt(prompt("Quantos numeros vc quer ler? "));
4 | print(repetir)
5 |
6 | while(i < repetir){
7 |     valor = parseInt(prompt("Digite um valor: "));
8 |     sum = sum + valor;
9 |     i = i + 1;
10 | }
11 | media = sum / repetir;
12 | print(media);
```

Octave

Uma solução em Octave.

```
1 | clear
2 |
3 | i = 1, sum = 0
4 | repetir = input("Quantos numero quer ler? ")
5 |
6 | while (i <= repetir)
7 |     num = input("digite um numero" )
8 |     sum = sum + num
9 |     i++
```

```

10 | endwhile
11 | media = sum / repetir
12 | printf(" %f ", media)

```

5 Laço Do-While/Do-Until

->-> *O par de instruções do-while executa laços do tipo faça-enquanto*

<-<-

Nesta técnica executamos um conjunto de instrução enquanto uma determinada condição for verdadeira. O conjunto de instruções a ser repetido é executado ANTES do teste, sendo que este conjunto de instruções é executado pelo menos uma vez.

Na técnica anterior (4) o conjunto de instruções é executado DEPOIS do teste, podendo ocorrer situações em que não ocorra nenhuma execução deste bloco.

O diagrama apresentado na Figura 5, mostra graficamente este tipo de repetição.

5.1 Problema 5.1

Escreva um algoritmo/programa que apresente na tela, a mensagem “continuar”, pelo menos uma vez, até que o valor -1 seja inserido no programa.

Fluxograma de uma solução para o problema 5.1

O laço inicia mostrando a palavra *continuar*. O segundo passo a ser feito é a leitura de um valor do teclado. Por fim, o teste do laço, que verifica que foi digitado -1: Se o teste for falso, retorno para o início do laço; se o teste deu verdadeiro então o programa vai para o fim.

do..while: como funciona em fluxograma

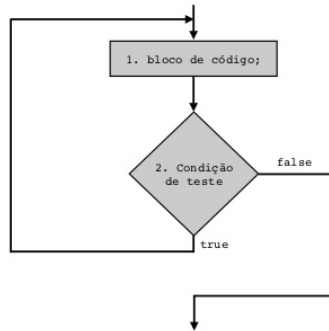


Figura 5: Fluxograma representando o funcionamento de um laço do tipo do-while.

Pseudocódigo

Um algoritmo em pseudocódigo, que soluciona o problema 5.1.

```
1 | Início
2 | var parar;
3 | faça
4 |     mostrar("continuar");
5 |     parar = lerValor();
6 | enquanto (parar != -1)
7 | Fim
```

JavaScript

Um algoritmo em JavaScript, que soluciona o problema 5.1.

```
1 | var parar;
2 | do{
3 |     print("continuar");
4 |     parar = prompt("-1 finalizar / outro valor para continuar");
5 | }
6 | while(parar != -1);
```

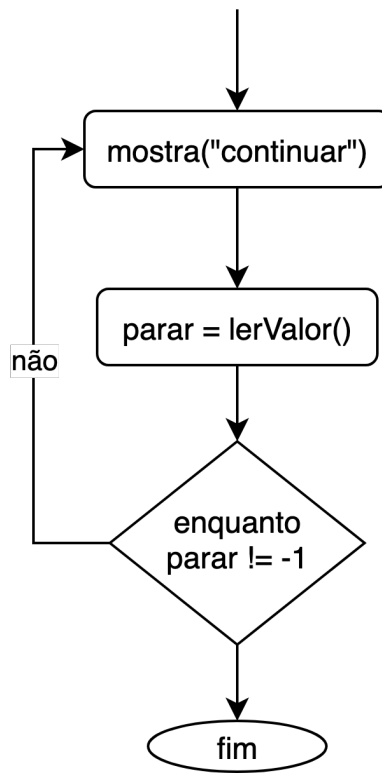


Figura 6: Fluxograma representando a solução para o problema 5.1.

Octave

A linguagem Octave não dispõe de um mecanismo faça-enquanto. Nesta linguagem utiliza-se um mecanismo chamado faça-até, executado pela instrução `do-until`. Na prática o que muda é apenas a forma com que o teste final é feito.

O laço se encerra quando a condição for verdadeira (no faça-enquanto o laço se encerra quando a condição for falsa).

Um algoritmo em Octave, que soluciona o problema 5.1 utilizando a técnica *do – until*.

```

1 | clear
2 | do
3 |     disp("Continuar...")
4 |     parar = input("-1 para finalizar: ")
  
```


5 | `until`(parar == -1)

5.2 Problema 5.2

Usando a estrutura de laço do-while (ou do-until para Octave), criar um algoritmo/programa para apresentar os números ímpares de 1 até N, sendo N um valor estipulado pelo usuário.

Fluxograma

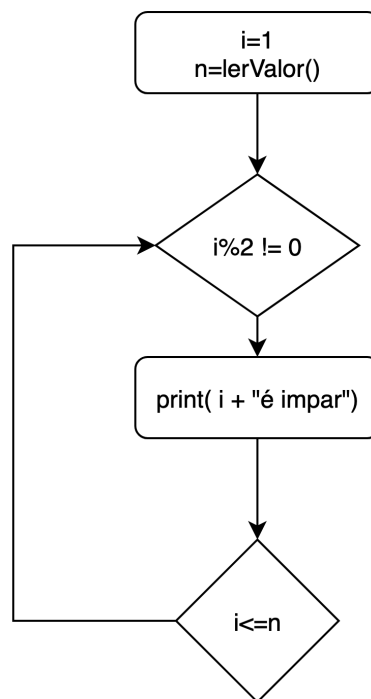


Figura 7: Fluxograma representando a solução para o problema 5.2.

Pseudocódigo

Um algoritmo em pseudocódigo, que soluciona o problema ??.

JavaScript

Um algoritmo em JavaScript, que soluciona o problema ??.

Octave

Um algoritmo em Octave, que soluciona o problema ??.

6 Tipo de laço

6.1 Problema ??

Fluxograma

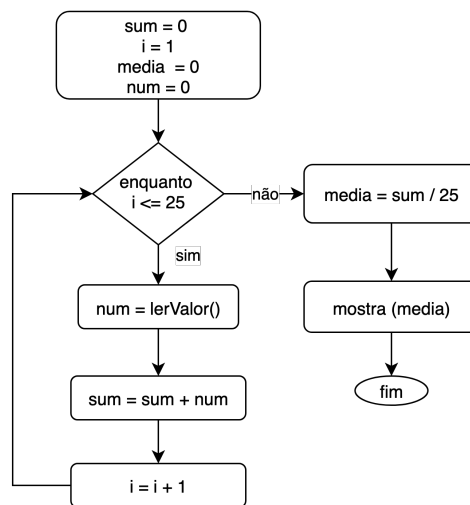


Figura 8: Fluxograma representando a solução para o problema ???.

Pseudocódigo

Um algoritmo em pseudocódigo, que soluciona o problema ??.

JavaScript

Um algoritmo em JavaScript, que soluciona o problema ??.

Octave

Um algoritmo em Octave, que soluciona o problema ??.