

# Programação com Octave/Matlab

Coltec - UFMG

## Capítulo 8 - Strings

Márcio Fantini Miranda

20 de junho de 2021

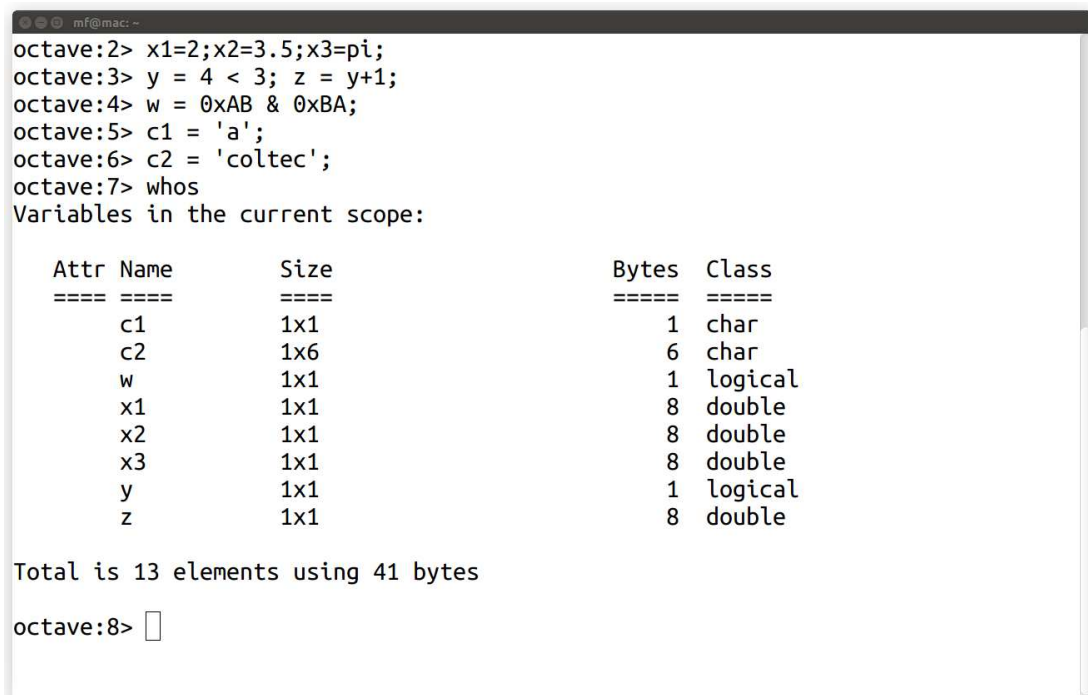
### Sumário

<b>1</b>	<b>Tipos de Dados no Octave</b>	<b>2</b>
<b>2</b>	<b>Strings no Octave</b>	<b>4</b>
2.0.1	Exercícios de Fixação . . . . .	6
2.1	Comparando Strings . . . . .	7
2.1.1	Exercícios de Fixação . . . . .	8
<b>3</b>	<b>Obtendo Caracteres de uma String</b>	<b>9</b>
3.0.1	Exercícios de Fixação . . . . .	9
<b>4</b>	<b>Tipo Célula (cell)</b>	<b>10</b>
4.0.1	Exercícios de Fixação . . . . .	12
<b>5</b>	<b>Outras Funções para Lidar com Strings</b>	<b>12</b>
5.1	Exemplos com str2num() e num2str() . . . . .	13

## 1 Tipos de Dados no Octave

O Octave possui diferentes **tipos de dados**, sendo que as duas categorias principais são os dados numéricos e os caracteres.

Na figura 1 é apresentado um exemplo com tipos de dados do Octave. Nesse exemplo foi executado o comando **whos** que mostra as variáveis que estão armazenadas na memória do Octave e mostra os tipos de cada uma.



```
mf@mac: ~  
octave:2> x1=2;x2=3.5;x3=pi;  
octave:3> y = 4 < 3; z = y+1;  
octave:4> w = 0xAB & 0xBA;  
octave:5> c1 = 'a';  
octave:6> c2 = 'coltec';  
octave:7> whos  
Variables in the current scope:  
  
  Attr Name      Size      Bytes  Class  
  ==== =====  
      c1         1x1         1  char  
      c2         1x6         6  char  
      w          1x1         1 logical  
      x1         1x1         8  double  
      x2         1x1         8  double  
      x3         1x1         8  double  
      y          1x1         1 logical  
      z          1x1         8  double  
  
Total is 13 elements using 41 bytes  
octave:8> 
```

Figura 1: Exemplos de tipos de dados básicos

Repare que temos basicamente **três tipos de dados**: números (ou tipo numérico), caracteres (ou tipo caracter) e lógicos (ou tipo

lógico).

## Tipos Numéricos

Os dados numéricos podem se classificados de forma diferente, dependendo da forma como foram definidos. O *default*<sup>1</sup> é serem definidos como o tipo *double*, que significa que ocupam 8 bytes. O tipo *double* é definido como sendo de *dupla precisão*. Formalmente dizemos que é um tipo de *ponto flutuante de dupla precisão*. Ponto flutuante é o nome que damos para variáveis reais (que podem ter números não inteiros, ou seja, com ponto decimal. Lembre-se que não usamos vírgula para números decimais no Octave).

## Tipos Caracteres

Um caractere é um símbolo. Pode ser uma letra, um símbolo específico como por exemplo \$, #, &. Tudo que usamos em frases são caracteres. Assim na frase:

essa turma é nota 10!

temos 22 caracteres. (Pode contar!). O espaço é um caracter. Assim como o número 1, o número 0 e o ponto de exclamação.

Quando temos uma variável que armazena um único caractere, dizemos que ela é do tipo caractere. Mas quando ela armazena mais de um caractere, ela passa a ser uma **variável composta do tipo caractere**, ou seja uma string. Veja na figura 1 que a variável **c1** possui a dimensão  $1 \times 1$ , ou seja é uma **variável escala**, já a variável

---

<sup>1</sup>A palavra *default* significa "padrão". Quando dizemos que algo é default estamos dizendo que é padrão. Por exemplo, uma "configuração default" é a configuração padrão do sistema.

---

`c2` possui dimensão  $1 \times 6$ , ou seja é um vetor de dimensão 1 linha e 6 colunas. Um vetor de caracteres é uma string.

### Atribuindo caracteres e strings à variáveis

Quando atribuímos valores tipo string ou tipo caractere à uma variável, temos que usar aspas simples ou duplas, para especificar o valor.

Veja alguns exemplos

## 2 Strings no Octave

Uma string é um vetor de caracteres.

Sendo um vetor, a string é uma variável indexada (ou composta) e portanto cada elemento da variável está associada a um índice. Assim como os vetores (pois string é um vetor).

Assim sendo, quando criamos uma string a partir de uma leitura do teclado, a sequência de caracteres fica armazenada na variável que passa a ser do tipo “vetor de caracteres”. E sendo vetor, cada um dos seus elementos pode ser acessado via seu índice.

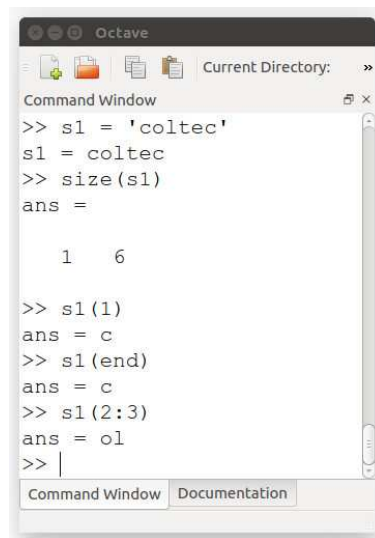


Figura 2: Exemplos de tipos de dados básicos

Estude o exemplo 2.1 dado a seguir.

### Exemplo 2.1 (Lendo uma String)

../..../cursoC\_2019/apostilaMatlab/exemplos/exemplo\_string1.m

```
1 % exemplo simples com strings e caracteres
2 clc
3 clear all
4
5 nome = input('entre com seu nome: ','s');
6 c = input('entre com um caractere qualquer: ','s');
7 x = c;
8 y = nome;
9 L1 = length(nome);
10 printf("seu nome é %s e tem %i caracteres\n",nome,L1);
11 disp("-----");
12 printf("voce digitou o caractere %c. A variavel c possui
    comprimento = %i \n",c,length(c));
```

---

## Explicação do Exemplo

- Na linha 5 é feita a leitura de uma string. Para isso usamos a função **input()** com o parâmetro 's' para indicar que o dado digitado será do tipo string.
- Na linha 6 é feita a leitura de um caractere. É pedido para o usuário digitar um caractere. Mas nada impede que ele digite uma string (mais de um caractere). O programa deve, quando for o caso, verificar se o dado inserido está de acordo com o pedido.
- Na linha 8 guardamos o valor que está na variável *c* na variável *x*. Essa e a seguinte só foram colocadas no script para mostrar que podemos atribuir variáveis strings normalmente. As variáveis *x* e *y* não são usadas no script...
- Na linha 8 obtemos o comprimento da string *nome*. A função **length()** retorna o tamanho da string (na verdade retorna o tamanho de qualquer vetor). Vetores são array de uma linha e várias colunas ou uma coluna e várias linhas.
- Nas linhas 10 e 12 imprimimos na tela a string e o caractere e seus comprimentos.

### 2.0.1 Exercícios de Fixação

1. Execute o exemplo acima e ao final, digite **whos** na linha de comando e veja as variáveis definidas e quais os tipos. Entenda o que o comando **whos** mostra.
2. Refaça o exemplo, agora verificando se o usuário digitou um

caractere (tamanho = 1) mesmo. Caso tenha entrado com mais de um caractere, o programa deve terminar.

## 2.1 Comparando Strings

Para compararmos strings usamos a função **strcmp()**. Essa função recebe duas strings e as compara. A função retorna o valor 0 quando a comparação for falsa, ou seja, quando as strings não são iguais e retorna 1 quando as strings forem iguais. É importante entender o uso da função **strcmp()**. Ela é muito útil para as aplicações com strings, como veremos mais adiante no curso.

Estude o exemplo 2.2 dado a seguir. Nesse exemplo temos um programa simples que fica dentro de um loop **while** até o usuário digitar 'sair'.

Ao rodar o exemplo, repare que as palavras digitadas pelo usuário são mostradas, enquanto o usuário não digitar 'sair'. Quando a palavra digitada for 'sair' o loop while terá sua comparação verdadeira, ou seja,

```
strcmp(s,'sair') valerá 1 e portanto  
strcmp(s,'sair') == 0 será FALSO
```

### Exemplo 2.2 (Comparando Strings)

```
../..../cursoC_2019/apostilaMatlab/exemplos/exemplo_string2.m  
1 % exemplo 2: comparando strings  
2 clc;clear all  
3 s='1';  
4 while(strcmp(s,'sair')==0)  
5     s = input('entre com uma palavra (digite sair para sair  
        do loop): ','s');
```

```
6 printf("a palavra %s tem comprimento %i\n",s,length(s));  
7 end
```

## Explicação do Exemplo

- Na linha 4 está o teste do loop. Enquanto a expressão

`strcmp(s,'sair') == 0`

for verdadeira (ou seja a string `s` for diferente de `'sair'`), o loop será mantido.

- Na linha 5 a string é lida e mostrada na linha 6.

### 2.1.1 Exercícios de Fixação

1. Execute o exemplo acima.
2. Refaça o exemplo, agora fazendo com que a palavra `'sair'` (que é usada para sair do loop) não seja considerada como uma string válida. Ou seja, se o usuário entrar com uma lista de nomes, por exemplo, ele não quer que a palavra `'sair'` seja processada como uma entrada válida. Nesse caso a palavra `'sair'` é usada para fazer o teste while ser verdadeiro, mas não deve ser impressa.
3. Refaça o exemplo usando do-until.



## 3 Obtendo Caracteres de uma String

No exemplo 3.1 um loop for é usado para percorrer a string e mostrar seus caracteres.

### Exemplo 3.1 (Percorrendo uma String)

```
../../../cursoC_2019/apostilaMatlab/exemplos/exemplo_string3.m
1 % exemplo 3: lendo caracteres de uma string
2 clc;clear all
3 do
4     s = input('entre com uma palavra (ENTER para sair):
        ', 's');
5     c1 = length(s);
6     for i = 1:c1
7         printf("caractere da posicao %i = %c\n", i, s(i));
8     end
9 until (strcmp(s, '')==1)
```

Nesse exemplo, a linha 5 contém a instrução para obtermos o comprimento da string. Esse comprimento é usado no loop for, que vai de **1** até **c1**. O loop do-until será interrompido quando se digitar ENTER.

#### 3.0.1 Exercícios de Fixação

1. Execute o exemplo acima.
2. Refaça o exemplo, agora obtendo o tamanho da string usando o comando `size()`
3. Refaça o exemplo, agora usando um loop while no lugar do loop for.

## 4 Tipo Célula (cell)

., No exemplo 4.1 a seguir, temos dois loops diferentes, um loop do-until e um loop while(1), que é um loop infinito. Esse loop infinito é interrompido no break dentro do if.

Nesse exemplo a seguir, apresentamos o tipo **cell**. O tipo de dados **cell** (ou célula) é usado com as chaves { }. Quando definimos uma variável usando as chaves e um índice, estamos criando uma célula.

A vantagem da célula é que podemos guardar variáveis de diferentes tipos numa única variável, por exemplo, a variável *x* definida a seguir conterá uma string e dois números, um inteiro e um real.

```
x{1} = 'string';  
x{2} = 1234;  
x{3} = 2.5;
```

Entenda bem o exemplo 4.1. Veja a Explicação logo após o exemplo e faça os exercícios de fixação.

### Exemplo 4.1 (Armazenando strings numa Célula (cell))

../..../cursoC\_2019/apostilaMatlab/exemplos/exemplo\_string4.m

```
1 % exemplo 4: armazenando strings  
2 clc; clear all ;  
3 s="1"; i=1;  
4 while(strcmp(s, '')==0)  
5     s2 = s;  
6     s = input('entre com uma palavra (ENTER para sair):  
7         ','s');  
7     strings{i} = s;  
8     i=i+1;
```

```
9   end
10  printf("a ultima palavra digitada foi %s\n",s2);
11  disp('digite para continuar')
12  pause
13  clc
14  disp('Fazendo a mesma coisa, agora usando um loop
      while(1)')
15  i=1;
16  while(1)
17      s2 = s;
18      s = input('entre com uma palavra (ENTER para sair):
                ','s');
19      strings2{i} = s;
20      i=i+1;
21      if (strcmp(s,'')==1)
22          break
23      end
24  end
25  printf("a ultima palavra digitada foi %s\n",s2);
```

## Explicação do Exemplo

- Na linha 4 iniciamos o while com a comparação usando strcmp(), como já foi explicado acima.
- Na linha 7 guardamos a string lida na variável do tipo **cell**. Na linha 8, incrementamos o índice *i*.
- Da linha 15 até a linha 23 fazemos o loop while, que repete o que é feito entre as linhas 4 e 9.
- O loop while(1) que começa na linha 15 é interrompido quando o if() da linha 20 for verdadeiro. Quando isso ocorre o comando

`break` interrompe o loop.

- Repare que nos dois loops as strings são armazenadas em variáveis do tipo `cell`.

#### 4.0.1 Exercícios de Fixação

1. Execute o exemplo acima.
2. Após executar, digite o comando **`whos`**. Repare os tipos de dados que estão no workspace.
3. Digite **`disp(strings)`** e entenda o resultado.
4. Refaça o exemplo, agora executando apenas o primeiro loop (delete ou comente o loop `while(1)`). Veja que o ENTER que o usuário digita para sair do loop é armazenado na variável `strings`. Refaça o exemplo para que isso não ocorra mais.

## 5 Outras Funções para Lidar com Strings

Existem algumas funções do Octave que são muitos úteis para trabalharmos com strings e com números. Elas são usadas para convertermos números para strings e vice-versa.

Algumas delas:

1. `int2str`: *converte um número (ou um array) para uma string (array de caracteres),*
2. `num2str`, *converte um número (ou um array) para uma string (array de caracteres)*

3. `mat2str`, *converte uma matriz de números reais ou complexos em uma string*
4. `str2double`, *converte uma string para um número real ou complexo*
5. `str2num`, *converte uma string para um número*

### 5.1 Exemplos com `str2num()` e `num2str()`

A função `str2num()` (leia "str to num", ou seja "str PARA num") transforma um dado que é string ou character para um número. Por exemplo, a string "123" é transformada no número 123.

A função `num2str()` (leia "num to str", ou seja "num PARA str") transforma um dado numérico para um string/caractere. Por exemplo o número 456 pode ser transformado para "456".

Isso é útil quando queremos apresentar uma frase com um dado numérico ou quando lemos um dado como texto e queremos torná-lo número para fazermos uma operação aritmética.

**Exemplo 5.1 (Usando str2num e num2str)**

../../../../cursoC\_2019/apostilaMatlab/exemplos/testa\_Str2NUm\_e\_Num2Str.m

```
1 % exemplo com str2num e num2str
2
3
4 x = input('entre com um numero: ' , ' s' ); % numero lido
   como string
5
6 aux1 = length(x);
7 aux2 = isdigit(x);
8 aux3 = sum(aux2);
9
10 if (aux1 != aux3)
11     disp('voce digitou algum caracter que nao eh
        numero...bye!' )
12     break;
13 else
14     y = str2num(x);
15 endif
16
17 disp(['x = (string) ',x]);
18 disp(['y = (numero) ',num2str(y)]);
19
20 disp('repare os tipos de dados de x e y:');
21 whos
```

## 6 Strings no Octave e no Matlab

O Octave e o Matlab possuem várias funções específicas para trabalhar com strings (algumas apresentadas nesse capítulo). Para saber mais, visite os sites oficiais de ambos, apresentandos abaixo:

- <https://octave.org/doc/v4.2.1/Manipulating-Strings.html>
- <https://www.mathworks.com/help/matlab/characters-and-strings.html>