

# Funções em Scilab



Prof. Jeferson Souza, MSc. (thejefecomp)  
[thejefecomp@neartword.com](mailto:thejefecomp@neartword.com)



# Introdução

## O que é uma Função?

Uma função é uma construção da linguagem do Scilab que permite a especificação de um conjunto de instruções executadas dentro de um mesmo contexto [escopo].

# Funções em Scilab

## Sintaxe

A sintaxe simples de uma função é a seguinte:

```
function <[<variáveis_retorno>]=> nome_da_função (<argumentos>)  
    <corpo_da_função>  
endfunction
```

As variáveis de retorno, e os argumentos pertencentes à assinatura da função, são opcionais.

# Funções em Scilab

## Exemplo 1

```
function resultado=soma(operando1, operando2)
```

```
    resultado = operando1 + operando2
```

```
endfunction
```

A função pode ser invocada como nos exemplos abaixo:

```
soma(2,3)
```

```
resultado=soma(2,3)
```

# Funções em Scilab

## Exemplo 2

```
function somaSemRetorno(operando1, operando2)
    resultado = operando1 + operando2
    mprintf("%f", resultado)
endfunction
```

A função pode ser invocada como no exemplo abaixo:

```
somaSemRetorno(2,3)
```

Funções declaradas sem retorno não podem ser atribuídas à variáveis.

# Funções em Scilab

## Exemplo 3

```
function [resultado,operando1,operando2]=somaRetornoMultiplo(operando1,  
operando2)
```

```
    resultado = operando1 + operando2
```

```
endfunction
```

A função pode ser invocada como no exemplo abaixo:

```
[resultado,operando1,operando2]=somaRetornoMultiplo(2,3)
```

Variáveis de retorno de nome idêntico a argumentos de entrada têm sua primeira atribuição realizada automaticamente pelo Scilab.

# Funções em Scilab com Argumentos Variáveis

## Sintaxe com argumentos variáveis

A sintaxe de uma função com argumentos variáveis pode conter argumentos de entrada e saída variáveis. Os argumentos de entrada variáveis são especificados com o uso da palavra reservada **varargin**; os argumentos de saída variáveis [i.e. variáveis de retorno definidas em tempo de execução] são especificados com o uso da palavra reservada **varargout**.

# Funções em Scilab com Argumentos Variáveis

## Sintaxe com argumentos de entrada variáveis

A sintaxe de uma função com argumentos de entrada variáveis é a seguinte:

```
function <[<variáveis_retorno>]=> nome_da_função  
(<argumentos> <,varargin>)  
    <corpo_da_função>  
endfunction
```



# Funções em Scilab com Argumentos Variáveis

## Exemplo 1

```
function resultado=soma(varargin)
    numeroEntradas = length(varargin)
    resultado = 0
    if numeroEntradas > 0 then
        for indice = 1:numeroEntradas
            resultado = resultado + varargin(indice)
        end
    end
endfunction
```

A função pode ser invocada como nos exemplos abaixo:

```
resultado=soma()
```

```
resultado=soma(2)
```

```
resultado=soma(2,3)
```

# Funções em Scilab com Argumentos Variáveis

## Exemplo 2

```
function [resultado,erro]=soma(varargin)
    numeroEntradas = length(varargin)
    resultado = 0
    erro = 0
    if numeroEntradas > 0 then
        for indice = 1:numeroEntradas
            resultado = resultado + varargin(indice)
        end
    else
        erro = 1
        resultado = -1
    end
endfunction
```

A função pode ser invocada como no exemplo abaixo:

```
[resultado,erro]=soma(2,3)
```

# Funções em Scilab com Argumentos Variáveis

## Sintaxe com argumentos de saída variáveis

A sintaxe de uma função com argumentos de saída variáveis é a seguinte:

```
function <[<variáveis_retorno><,varargout>]=> nome_da_função  
(<argumentos>)
```

```
<corpo_da_função>
```

```
endfunction
```

# Funções em Scilab com Argumentos Variáveis

## Exemplo 1

```
function varargout=soma(operando1,operando2)
    varargout(1) = operando1 + operando2
endfunction
```

A função pode ser invocada como nos exemplos abaixo:

```
soma(2,3)
```

```
resultado=soma(5,10)
```

# Funções em Scilab com Argumentos Variáveis

## Exemplo 2

```
function varargout=soma(operando1,operando2)
```

```
    resultado = operando1 + operando2
```

```
    varargout = list(resultado)
```

```
endfunction
```

A função pode ser invocada como nos exemplos abaixo:

```
soma(2,3)
```

```
resultado=soma(5,10)
```

# Funções em Scilab com Argumentos Variáveis

## Exemplo 3

```
function [resultado,varargout]=soma(operando1,operando2)
    resultado = operando1 + operando2
    varargout = list("soma",operando1,operando2)
endfunction
```

A função pode ser invocada como nos exemplos abaixo:

```
resultado=soma(2,3)
```

```
[resultado,operacao,operando1,operando2]=soma(2,3)
```

# Funções em Scilab com Argumentos Variáveis

## Exemplo com ambos **varargin** e **varargout**

```
function varargout=soma(varargin)
    numeroEntradas = length(varargin)
    resultado = 0
    if numeroEntradas > 0 then
        for indice = 1:numeroEntradas
            resultado = resultado + varargin(indice)
        end
    end
    varargout = list(resultado)
endfunction
```

A função pode ser invocada como nos exemplos abaixo:

```
resultado=soma(2,3)
```

```
resultado=soma(2,3,10)
```

# Funções InLine

## O que é uma Função inline?

Uma função *inline* é uma função cuja definição é realizada por meio textual [conjunto de caracteres]. A definição é dividida em duas partes distintas: a assinatura e o corpo da função. A assinatura contém os elementos necessários à identificação da função de maneira singular [única]. O corpo da função especifica todos os comandos que serão executados consoante sua invocação. A definição de funções *inline* somente pode ser feita por meio do uso da função **deff**.



# Funções InLine

## Sintaxe da função **deff**

A definição de uma função *inline* por meio da função **deff** tem a seguinte sintaxe:

```
deff (<assinatura_da_função>, <corpo_da_função>)
```

Onde:

<assinatura\_da\_função> é um conjunto de caracteres que especifica a assinatura da função a ser definida;

<corpo\_da\_função> é uma matriz de caracteres que especifica os comandos a serem executados aquando da invocação da função a ser definida.

# Funções InLine

## Exemplo 1

```
deff('resultado=soma(operando1, operando2)', 'resultado = operando1  
+ operando2')
```

A função pode ser invocada como nos exemplos abaixo:

```
soma(2,3)
```

```
resultado=soma(2,3)
```

# Funções InLine

## Exemplo 2

```
deff('resultado=soma(varargin)', ['numeroEntradas =  
length(varargin)'; 'resultado = 0'; 'if numeroEntradas > 0 then'; 'for indice =  
1:numeroEntradas'; 'resultado = resultado + varargin(indice)'; 'end'; 'end'])
```

A função pode ser invocada como nos exemplos abaixo:

```
resultado=soma()
```

```
resultado=soma(2)
```

```
resultado=soma(2,3)
```

# Proteção de Funções em Scilab

## Pela honra de Grayskull: Prazer, funcprot()

É possível proteger as funções definidas em Scilab pelo uso de uma função especial denominada *funcprot()*. A função *funcprot()* permite a definição de diferentes modos de segurança para alertar e evitar que funções sejam redefinidas durante uma sessão do Scilab.

# Proteção de Funções em Scilab

## Sintaxe de uso da função `funcprot()`

A função `funcprot()` possui as seguintes sintaxes de utilização:

`funcprot(modoprotecao)`

`modoProtecaoAnterior = funcprot(modoprotecao)`

`modoProtecao = funcprot()`

Onde:

**modoProtecao** é o modo de proteção aplicado pela função `funcprot()`. Pode assumir os valores de 0, 1, ou 2;

**modoProtecaoAnterior** é o modo de proteção vigente antes da aplicação de um novo modo de proteção. Pode assumir os mesmos valores de `modoProtecao`.

# Proteção de Funções em Scilab

## Modos de proteção da função `funcprot()`

Existem 3 modos de proteção que podem ser utilizados em conjunto com a função `funcprot()`. São eles:

- 0 → desativa o modo de proteção;
- 1 → ativa o modo de alerta, a avisar o usuário da redefinição de funções durante uma sessão do Scilab (modo padrão);
- 2 → ativa o modo de erro, a implicar em um erro de execução no momento que uma redefinição de função é detectada.

# Proteção de Funções em Scilab

## Exemplo 1 - Desativa o modo de proteção

```
modoProtecaoAnterior = funcprot(0)
```

```
function resultado=soma(operando1, operando2)
```

```
    resultado = operando1 + operando2
```

```
endfunction
```

A função pode ser invocada como nos exemplos abaixo:

```
soma(2,3)
```

```
resultado=soma(2,3)
```

```
funcprot(modoProtecaoAnterior) //Retorna ao modo de proteção  
definido antes da desativação.
```

# Proteção de Funções em Scilab

## Exemplo 2 - Ativa o modo de erro

```
modoProtecaoAnterior = funcprot(2)
```

```
function resultado=soma(operando1, operando2)
```

```
    resultado = operando1 + operando2
```

```
endfunction
```

A função pode ser invocada como nos exemplos abaixo:

```
soma(2,3)
```

```
resultado=soma(2,3)
```

```
funcprot(modoProtecaoAnterior) //Retorna ao modo de proteção  
definido antes da ativação do modo de erro.
```



# Bibliografia



Scilab Enterprises. **“Scilab Online Help”**. version 6.0.2, 2019. Disponível em: [https://help.scilab.org/docs/6.0.2/en\\_US/index.html](https://help.scilab.org/docs/6.0.2/en_US/index.html). Acesso em: 24 Jun. 2021.

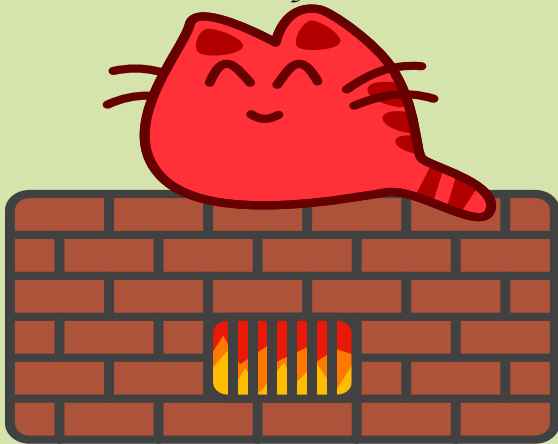


Gomez, C. and Scilab Enterprises. **“Scilab for very beginners”**, 2013.



Rietsch, E. **“An Introduction to Scilab from a Matlab User’s Point of View”**. version 2.6-1.0., 2001-2002.

*That's it folks!*



*Thank you for your attention!*