Aula prática 9

Funções: Primeira Parte

Resumo

As atividades propostas nesta prática visam explorar as primeiras noções sobre funções definidas pelo própio programador para o desenvolvimento de suas aplicações.

Sumário

1 Funções

Você já vem utilizando, em seus programas, diversas funções pré-definidas nas bibliotecas do Scilab. Muitas vezes, gostaríamos de poder definir novas funções, para utilizá-las em nossos programas. A definição e uso de funções torna os programas mais legíveis, mais modulares (bem divididos em subproblemas menores), favorece o reuso de código e facilita, de modo geral, a depuração e manutenção de programas.

Nesta aula prática você vai aprender a definir novas funções e utilizá-las em um programa.

Vamos começar com um exemplo bem simples. O exemplo a seguir mostra um programa para calcular e imprimir o maior entre dois valores informados pelo usuário:

```
clear; clc;
x1 = input("Primeiro valor .....: ");
x2 = input("Segundo valor ......: ");
maior = maior2valores(x1, x2);
printf("O maior valor é: %g\n", maior);
```

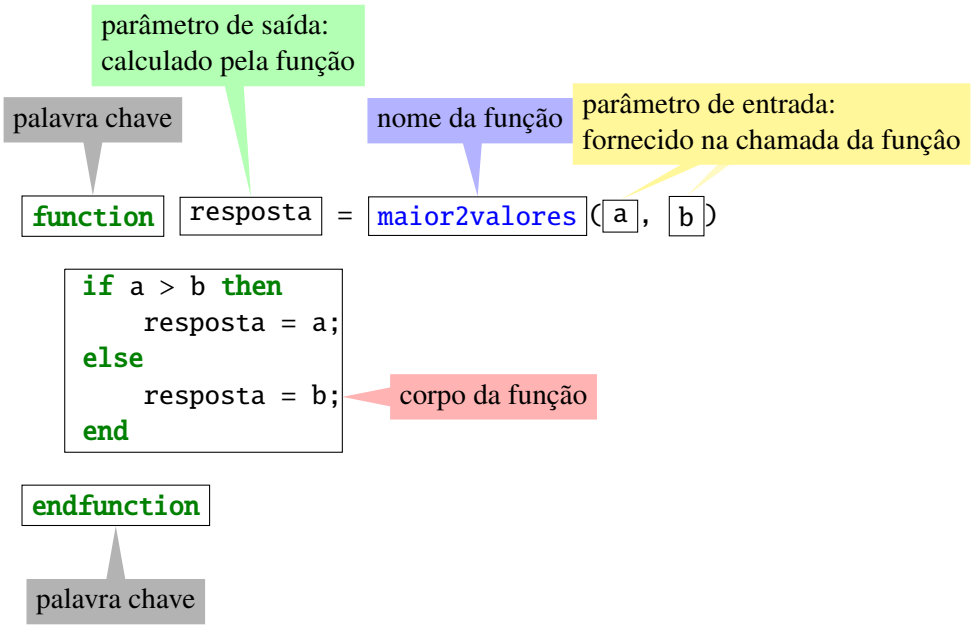
No programa acima, o cálculo do maior entre dois valores é feito por meio da função `maior2valores`. Esta função não existe nas bibliotecas do Scilab. Para que o programa acima funcione corretamente, devemos incluir, no início do programa, uma definição para a função `maior2valores` , tal como é mostrado a seguir:

```
clear; clc;

// Definição da função
function resposta = maior2valores(a, b)
    if a > b then
        resposta = a;
    else
        resposta = b;
    end
endfunction

// Programa principal
x1 = input("Primeiro valor .....: ");
x2 = input("Segundo valor ......: ");
maior = maior2valores(x1, x2);
printf("O maior valor é: %g\n", maior);
```

Copie o programa acima no Scinotes e execute o programa, certificando-se de que ele funciona corretamente.

2 Definindo funções

Uma definição e função em Scilab começa com a palavra reservada `function`. Em seguida, vem o nome de uma variável, que representa o valor a ser retornado pela função (comumente chamado de parâmetro de saída da função). Logo depois vem o nome da função, seguido da lista de parâmetros de entrada, cujos valores devem ser especificados quando a função é chamada no programa. Essas informações constituem o que se chama de cabeçalho da função. Em seguida, vem o corpo da função, que contém o código para o cálculo do valor a ser retornado pela função, o qual deve ser atribuído à variável que constitui o parâmetro de saída da função. Finalemnte, a definição é terminada pela palavra reservada `endfunction`.



Uma função pode ter qualquer número de parâmetros de entrada e pode também ter mais de um parâmtro de saída, conforme veremos mais adiante. Os parâmetros de entrada e de saída da função podem ser valores de quaisquer tipos: numéricos, lógicos, strings, vetores ou matrizes.

3 Exercícios

Tarefa 1: Força gravitacional

A lei da gravitação universal, proposta por Newton a partir das observações de Kepler sobre os movimentos dos corpos celestes, diz que:

> Dois corpos quaisquer se atraem com uma força diretamente proporcional ao produto de suas massas e inversamente proporcional ao quadrado da distância entre eles.

Essa lei é formalizada pela seguinte equação:

$$F = G\frac{m_1 m_2}{d^2}$$

onde:

- F é força de atração em Newtons (N),
- G é a constante de gravitação universal (6.67×10^{-11} N m^2/kg^2),
- m_1 e m_2 são as massas dos corpos envolvidos, em quilos (kg), e

- d é a distância entre os corpos em metros (m).

Escreva uma função para calcular a força gravitacional entre dois corpos dadas as suas massas e a distância entre eles.

Para testar sua função faça um programa principal para determinar a força sobre um satélite de $800kg$ em órbita a $38000km$ da superfície da Terra. A massa da Terra é de $5,98 \times 10^{24} kg$.

Exemplo de execução da aplicação

```
força gravitacional: 220.978 N
```

Solução:

```
clear; clc;

// função para calcular a força gravitacional
function F = forcaGravitacional(m1, m2, r)
    G = 6.67E-11;
    F = G * m1 * m2 / r^2;
endfunction

// programa principal
massaSatelite = 800;
massaTerra = 5.98E24;
distancia = 38000E3;
forca = forcaGravitacional(massaSatelite, massaTerra, distancia);
printf("força gravitacional: %g N\n", forca);
```

Tarefa 2: Total de segundos em um horário

Defina uma função que receba três números inteiros como parâmetros, representando as horas, minutos e segundos de um horário, e os converte para segundos. Por exemplo, 2h, 40min e 10s correspondem a 9,610s.

Crie um programa principal que leia as horas, os minutos e os segundos de um horário e usa esta função para convertê-lo para segundos, e exibe o resultado da conversão.

Exemplo de execução da aplicação

```
Conversão de horário
------------------------------------
horas: 3
minutos: 45
segundos: 21

total de segundos: 13521
```

Solução:

```
clc;
clear;

function segundos = horario(h, m, s)
    segundos = (h*60 + m)*60 + s;
endfunction

printf("Conversão de horário\n");
printf("------------------------------------\n");
horas = input("horas: ");
minutos = input("minutos: ");
segundos = input("segundos: ");
total_segundos = horario(horas, minutos, segundos);
printf("\ntotal de segundos: %g\n", total_segundos);
```

Tarefa 3: Soma de uma série

Defina uma função que receba como parâmetro um valor inteiro e positivo n e retorne o valor de S, obtido pelo seguinte cálculo:

$$S = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{n!}$$

Escreva um programa que solicite ao usuário que digite um número inteiro positivo e calcule e exiba o valor de S usando esta função.

Exemplo de execução da aplicação

```
Cálculo da soma da série
------------------------------------
n: 10

soma da série: 2.71828
```

Solução:

```
clc;
clear;

function soma = serie(n)
    soma = 1;
    denominador = 1;
    for i = 1:n do
        denominador = denominador * i;
        soma = soma + 1/denominador;
    end
endfunction

printf("Cálculo da soma da série\n");
printf("------------------------------------\n");
n = input("n: ");
total = serie(n);
printf("\nsoma da série: %g\n", total);
```

Tarefa 4: Triângulos

Escreva um programa que receba três valores (obrigatoriamente maiores que zero), representando as medidas dos três lados de um triângulo, verifica se estes lados formam um triângulo e, em caso afirmativo, classifica o triângulo quanto aos lados.

Para implementar o seu programa defina as funções mencionadas a seguir e utilize-as no programa

principal.

- Defina uma função para determinar se três valores dados podem ser medidas dos lados de um triângulo. Sabe-se que, para ser um triângulo, a medida de um lado qualquer deve ser inferior à soma das medidas dos outros dois lados.

  Sua função deverá ter três argumentos correspondentes às medidas dos lados do triângulo, e um resultado, que deverá ser um valor lógico (verdadeiro ou falso) indicando se os lados formam ou não um triângulo.

- Defina uma função para classificar o triângulo quanto aos lados:

  triângulo equilátero as medidas dos três lados são iguais

  triângulo isósceles as medidas de dois lados apenas são iguais

  triângulo escaleno as medidas dos três lados são diferentes

  Sua função deverá ter três argumentos (as medidas dos lados do triângulo) e o seu resultado deverá ser uma das strings `"equilátero"`, `"isósceles"` ou `"escaleno"`, de acordo com a classificação do triângulo.

O programa principal deverá ler os lados e usar as funções para determinar se os lados realmente formam um triângulo e, em caso positivo, classificar o triângulo.

Exemplo de execução da aplicação

```
Triângulos
==============================
primeiro lado: 10
segundo lado: 10
terceiro lado: 10

classificação: equilátero
```

Exemplo de execução da aplicação

```
Triângulos
==============================
primeiro lado: 10
segundo lado: 13
terceiro lado: 10

classificação: isósceles
```

Exemplo de execução da aplicação

```
Triângulos
==============================
primeiro lado: 10
segundo lado: 8
terceiro lado: 15

classificação: escaleno
```

Exemplo de execução da aplicação

```
Triângulos
==============================
primeiro lado: 3
segundo lado: 2
terceiro lado: 1

não é triângulo
```

Solução:

```
clear; clc;

// função para verificar os lados de um triângulo
function t = e_triangulo(a, b, c)
    t = a < b + c & b < a + c & c < a + b;
endfunction

// função para classificar o triângulo quanto aos lados
function c = classifica_t(a, b, c)
    if a == b & b == c then
        c = "equilátero"
    elseif a == b | a == c | b == c then
        c = "isósceles"
    else
        c = "escaleno"
    end
endfunction

// programa principal
printf("Triângulos\n");
printf("==============================\n");

lado1 = input("primeiro lado: ");
lado2 = input("segundo lado: ");
lado3 = input("terceiro lado: ");

if e_triangulo(lado1, lado2, lado3) then
    printf("\nclassificação: %s\n", classifica_t(lado1, lado2, lado3));
else
    printf("\nnão é triângulo\n");
end
```