BCC701 – Programação de Computadores I
Universidade Federal de Ouro Preto
Departamento de Ciência da Computação

www.decom.ufop.br/bcc701

Aula Teórica 09
Funções

Material Didático Proposto.

Propósitos do Uso de Funções

- Modularizar um programa em partes menores;
- Executar uma tarefa que é frequentemente solicitada;
- Aumentar a legibilidade e manutenibilidade do programa;
- Implementar as chamadas UDF (**U**ser **D**efined **F**unctions), para complementar as necessidades do programador na execução de tarefas não suportadas pelo ambiente de programação.

Exemplo de Uso de Funções

- Ler dois valores inteiros;
- Calcular o maior valor desses dois números;
- Imprimir o maior valor.

Exemplo de Uso de Funções

```
x1 = input("Primeiro Valor = ");
x2 = input("Segundo  Valor = ");
Maior = maior2val(x1, x2);
Printf("O MAIOR VALOR É: %g", Maior);
```

Exemplo de Uso de Funções

```
clear; clc;
// Definição da função
function Retorno = maior2val(a, b)
    if a > b then
        Retorno = a;
    else
        Retorno = b;
    end
endfunction
// Programa Principal
x1 = input("Primeiro Valor = ");
x2 = input("Segundo  Valor = ");
Maior = maior2val(x1, x2);
printf("\nO MAIOR VALOR É: %g", Maior);
```

Sintaxe de Função

Parâmetro de Saída: calculado pela função

```
function Retorno = maior2val(a,b)
    if a > b then
       Retorno = a;
    else
       Retorno = b;
endfunction
```

Parâmetro de Entrada: fornecido na chamada da função

Exemplo de Uso de Funções

- Fazer a leitura de seis valores numéricos positivos, não nulos e inteiros;
- Realizar a validação da entrada;
- Encontrar o primeiro menor, e o segundo menor, valores lidos;

SOLUÇÃO:

1) Sem usar funções;

2) Usando funções.

Solução 1

```
1  clc; clear;
2  n1 =input("DIGITE UM NÚMERO POSITIVO E INTEIRO: ")
3  while (n1 <= 0) | (n1 <> int(n1))
4      printf("ERRO: O NÚMERO DEVE SER INTEIRO E > QUE 0 !");
5      n1 =input("DIGITE UM NÚMERO POSITIVO E INTEIRO: ")
6  end    // n1 válido !
7  n2 =input("DIGITE UM NÚMERO POSITIVO E INTEIRO: ")
8  while (n2 <= 0) | (n2 <> int(n2))
9      printf("ERRO: O NÚMERO DEVE SER INTEIRO E > QUE 0 !");
10     n2 =input("DIGITE UM NÚMERO POSITIVO E INTEIRO: ")
11 end    // n2 válido !
12 if n1 <= n2 then
13     menor1 = n1;
14     menor2 = n2;
15 else
16     menor1 = n2;
17     menor2 = n1;
18 end
```

Solução 1

```
19 for i = 3:6
20     n =input("DIGITE UM NÚMERO POSITIVO E INTEIRO: ")
21     while (n <= 0) | (n <> int(n))
22         printf("ERRO: O NÚMERO DEVE SER INTEIRO E > QUE 0 !");
23         n =input("DIGITE UM NÚMERO POSITIVO E INTEIRO: ")
24     end    // n válido !
25     if (n <> menor1) & (n <> menor2) then
26         if n < menor1 then
27             menor2 = menor1;
28             menor1 = n;
29         elseif n < menor2    // menor1 <= n <= menor2
30             menor2 = n;
31         end
32     end
33 end
34 printf("\n\nPRIMEIRO MENOR NÚMERO: %g", menor1);
35 printf("\nSEGUNDO  MENOR NÚMERO: %g", menor2);
```

Solução 2

```
1  clc; clear;
2  // Função que devolve um valor lido pelo teclado;
3  // valor positivo, não nulo e maior que zero.
4  //-------------------------------------------------------
5  function R = entradaValida()
6      x =input("DIGITE UM NÚMERO POSITIVO E INTEIRO: ")
7      while (x <= 0) | (x <> int(x))
8          printf("ERRO: O NÚMERO DEVE SER INTEIRO E > QUE 0 !");
9          x =input("DIGITE UM NÚMERO POSITIVO E INTEIRO: ")
10     end    // x válido !
11     R = x;
12 endfunction
13 //-------------------------------------------------------
14 n1 = entradaValida();    // n1 válido !
15 n2 = entradaValida();    // n2 válido
16 if n1 <= n2 then
17     menor1 = n1;
18     menor2 = n2;
19 else
20     menor1 = n2;
21     menor2 = n1;
22 end
```

Solução 2

```
23 for i = 3:6
24     n = entradaValida();     // n válido !
25     if (n <> menor1) & (n <> menor2) then
26         if n < menor1 then
27             menor2 = menor1;
28             menor1 = n;
29         elseif n < menor2     // menor1 <= n <= menor2
30             menor2 = n;
31         end
32     end
33 end
34 printf("\n\nPRIMEIRO MENOR NÚMERO: %g", menor1);
35 printf("\nSEGUNDO  MENOR NÚMERO: %g", menor2);
```

Exemplo de Uso de Funções

- Cálculo do número de combinações de ***n*** tomados ***k*** a ***k***;
- Observe que o cálculo do fatorial é repetido três vezes.

$$\binom{n}{k} = \frac{n!}{(n-k)!\,k!}$$

Exemplo de Uso de Funções

- Para calcular o fatorial de um número inteiro ***n*** pode-se usar o seguinte trecho de programa:

```
fat = 1;
for i = 1:n
      fat = fat * i;
end
```

- Entretanto é necessário adaptar este código para obter o cálculo do número de combinações:

Exemplo de Uso de Funções

```
n = input("n="); k = input("k=");
fat_n = 1;
for i = 2:n
     fat_n = fat_n * i
end
fat_n_k = 1;
for i = 2:(n - k)
     fat_n_k = fat_n_k * i
end
fat_k = 1;
for i = 2:k
     fat_k = fat_k * i
end
nComb = fat_n / (fat_n_k * fat_k);
```

Exemplo de Uso de Funções

- Agora o programa anterior será dividido em duas partes: o programa principal e a função;
- O programa principal será codificado da seguinte forma:

```
n = input("n="); k = input("k=");
nComb = fatorial(n) / ...
        fatorial(n - k) * fatorial(k);
```

A função será codificada da seguinte forma:

```
function fat = fatorial(n)
    fat = 1;
    for i = 1:n
        fat = fat * i;
    end
endfunction
```

Observações: Funções

- Um programa é designado principal quando ele faz chamadas as funções.
- A execução de um programa com funções se inicia pelo programa principal.
- A execução de uma chamada transfere o controle de execução para a função.
- Ao término da execução da função, o controle é devolvido ao ponto de chamada, em uma operação chamada de retorno da função.

Sintaxe de Função

Parâmetro de Saída: calculado pela função

```
function fat = fatorial(n)
    fat = 1;
    for i = 1:n
        fat = fat * i;
    end
endfunction
```

Parâmetro de Entrada: fornecido na chamada da função

Sintaxe de Função: Vários Parâmetros

```
function [x1, x2] = eq2g(a, b, c)
    delta = b^2 - 4 * a * c;
    x1 = (-b + sqrt(delta)) / (2 * a);
    x2 = (-b - sqrt(delta)) / (2 * a);
endfunction
```

```
// Programa Principal;
x = 2; y = 4; z = 6;
[raiz_1, raiz_2] = eq2g(x, y, z);
```

Observações: Funções

- Uma função cria um espaço novo para as variáveis, que podem ter nomes iguais aos de variáveis já definidas no programa principal.
- As variáveis definidas por uma função são denominadas variáveis locais.
- As variáveis definidas no programa principal são denominadas variáveis globais.
- Mais sobre funções: Introdução à Organização e à Programação de Computadores – Prof. Oswaldo Carvalho.

Exemplo 1

Codifique um programa que faça a leitura de *n* valores através do teclado.

Para cada valor lido no teclado deve ser aplicada a função f(x) = x – sqrt(x). O resultado da aplicação da função deve ser acumulado em um somatório.

O cálculo de f(x) deve ser codificado em uma função definida pelo usuário.

Ao final o programa imprime o valor do somatório calculado.

Exemplo 1

```
function f = minhaF(x)
      f = x - sqrt(x);
endfunction

n = input("QUANTIDADE DE LEITURAS: ");
soma = 0;
for i = 1:n
      x = input("DIGITE UM VALOR: ");
      soma = soma + minhaF(x);
end
printf("\nSOMATÓRIO CALCULADO: %7.3f",
        soma);
```

Exemplo 2

Codifique um programa que calcule a série a seguir, onde *n* é o número de parcelas.

Cada parcela contém um numerador e um denominador. O Cálculo de ambos deve ser feito por funções definidas pelo usuário.

Ao final o programa imprime o valor da série.

$$\sum_{i=1}^{n} \frac{i - sen(i)}{i^3 - \cos(2i)}$$

Exemplo 2

```
function resposta = numerador(x)
    resposta = x - sin(x);
endfunction
// ---------------------------------------------------------------------------
function resposta = denominador(x)
    resposta = x^3 - cos(2 * x);
endfunction
// ---------------------------------------------------------------------------
n = input("QUANTIDADE DE PARCELAS: ");
soma = 0;
for i = 1:n
    soma = soma + numerador(i) / ...
                  denominador(i);
end
printf("\nSOMATÓRIO CALCULADO: %7.3f",
  soma);
```