# Aula Teórica 09
# Funções

Material Didático Proposto.

# Propósitos do Uso de Funções

- Modularizar um programa em partes menores;
- Executar uma tarefa que se repete com frequência;
- Facilitar a leitura e manutenção do programa;
- Complementar as necessidades do programador na execução de tarefas que se repetem.

# Exemplo de Uso de Funções

Faça um programa para

1. Ler dois valores inteiros;
2. Calcular o maior valor desses dois números (utilizando uma função);
3. Imprimir o maior valor.

## Sem o uso de uma Função

```
clear; clc;
x1 = input("Primeiro Valor = ");
x2 = input("Segundo  Valor = ");
if a > b then
     maior = a;
else
     maior = b;
end
printf("\nO maior valor é: %g", maior);
```

## Exemplo com o uso de uma 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); //chamada da função
printf("\nO Maior valor é: %g", maior);
```

## Exemplo de Uso de Funções

```
x1 = input("Primeiro Valor = ");
x2 = input("Segundo  Valor = ");
Maior = maior2val(x1, x2);
Printf("O maior valor é: %g", Maior);
```

É parecido com as funções que já usamos:

```
x = int(2.3)
r = modulo(5,2)
n = lenght(V)                  //para V um vetor
[n_lin, n_col] = size(M) //para M uma matriz
```

Mas neste caso você define o que "deve acontecer" dentro da função **maior2val** e o que deve ser retornado por ela

# 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 uma série de valores numéricos inteiros e positivos e encontrar o maior e o menor número lido;
- A leitura deve ser interrompida quando for digitado o número 0, que não deve ser computado.
  - Realizar a validação da entrada: inteiro.
  - Processar se o valor > 0 e parar se valor = 0;
- Imprimir os valores encontrados.

Solução 1: sem função

Solução 2: com função

# Solução 1

```
clc; clear;
maior = -%inf;
menor = %inf;
n = input("Digite um inteiro qualquer ou <= zero p/ fim: ")
while (n <> int(n))
    printf("Erro: o número deve ser inteiro!");
    n = input("Digite um inteiro qualquer ou <= zero p/ fim: ")
end
while n > 0
   if n < menor then
       menor = n;
   end
   if n > maior then
       maior = n;
   end
   n = input("Digite um inteiro qualquer ou <= zero p/ fim: ")
   while (n <> int(n))
     printf("Erro: o número deve ser inteiro!");
     n = input("Digite um inteiro qualquer ou <= zero p/ fim: ")
   end
end
printf("Maior valor: %g, menor valor: %g", maior, menor);
```

# Solução 2

```
clc; clear;
// devolve um valor inteiro lido pelo teclado;
//-----------------------------------------------------------
function resp = entradaValida( )
x = input("Digite um inteiro qualquer ou <= zero p/ fim: ")
while (x <> int(x))
    printf("Erro: o número deve ser inteiro!");
    x = input("Digite um inteiro qualquer ou <= zero p/ fim: ")
end
resp = x; // retorna o valor válido em x
endfunction
//-----------------------------------------------------------
maior = -%inf;
menor = %inf;
n = entradaValida( );        // chamada da função
while n > 0
    if n < menor then
        menor = n;
    end
    if n > maior then
        maior = n;
    end
    n = entradaValida( );    // chamada da função
end
printf("Maior valor: %g, menor valor: %g", maior, menor);
```

## Observações: Funções

- Um programa é designado principal quando ele faz chamadas às 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: Vários Parâmetros

*Parâmetros de entrada*

```
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
```

*Parâmetros de saída da função*

```
// programa principal
x = 2; y = 4; z = 6;
// chamada da função
[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

Escreva um programa para ler um vetor de tamanho *n*, informado pelo usuário e posteriormente, usando uma função **soma_vet**(vetor, tamanho), some os elementos do vetor e retorne este valor.

Atenção:

1) A leitura do vetor e a impressão da soma dos elementos do vetor são realizadas no programa principal.
2) Somente a soma dos valores do vetor é feita dentro da função soma_vet (vetor, tamanho )

# Exemplo 1

```
// === definição da função  ===
function resp = soma_vet(vetor, tamanho)
  s = 0
  for i = 1: tamanho
      s = s + vetor(i)
   resp = s
endfunction
// ==== programa principal ====
n = input("Tamanho do vetor: ");
for i = 1:n
     v(i) = input(sprintf("vetor(%g) = ", i));
end
soma = soma_vet(v, n) // chamada da função
printf("A soma dos elementos do vetor é: %g", soma)
```

# Exemplo 1.1

Escreva um programa para ler um vetor em lote, informado pelo usuário e posteriormente, usando uma função **prod_vet**(vetor), calcule o produto dos elementos do vetor e retorne este valor.

Atenção:

1) A leitura do vetor e a impressão do produto dos elementos do vetor são realizadas no programa principal.
2) Somente o produto dos valores do vetor é feita dentro da função prod_vet (vetor)

# Exemplo 1.1

```
// === definição da função  ===
function p = prod_vet(vetor)
  p = 1
  for i = 1: lenght(vetor)
      p = p* + vetor(i)
endfunction
// ==== programa principal ====
vet = input("Entre com o vetor : ");

prod = prod_vet(vet) // chamada da função

printf("O produto dos eletos do vetor é: %g", prod)
```

# 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);
```

# 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:

# Sintaxe de Função

**Parâmetro de Saída: calculado pela função**

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

**Parâmetro de Entrada: fornecido na chamada da função**

## 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(x)
    fat = 1;
    for i = 1:x
        fat = fat * i;
    end
endfunction
```

## Exemplo de Uso de Funções

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

## Exemplo de Uso de Funções

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