DEVSUPERIOR

Formação Desenvolvedor Moderno
Módulo: Lógica de Programação

Capítulo: Funções

https://devsuperior.com.br

Funções

https://devsuperior.com.br

Prof. Dr. Nelio Alves

Conceito informal de função

Uma função, também chamada de subprograma ou sub-rotina, é uma unidade de código que:

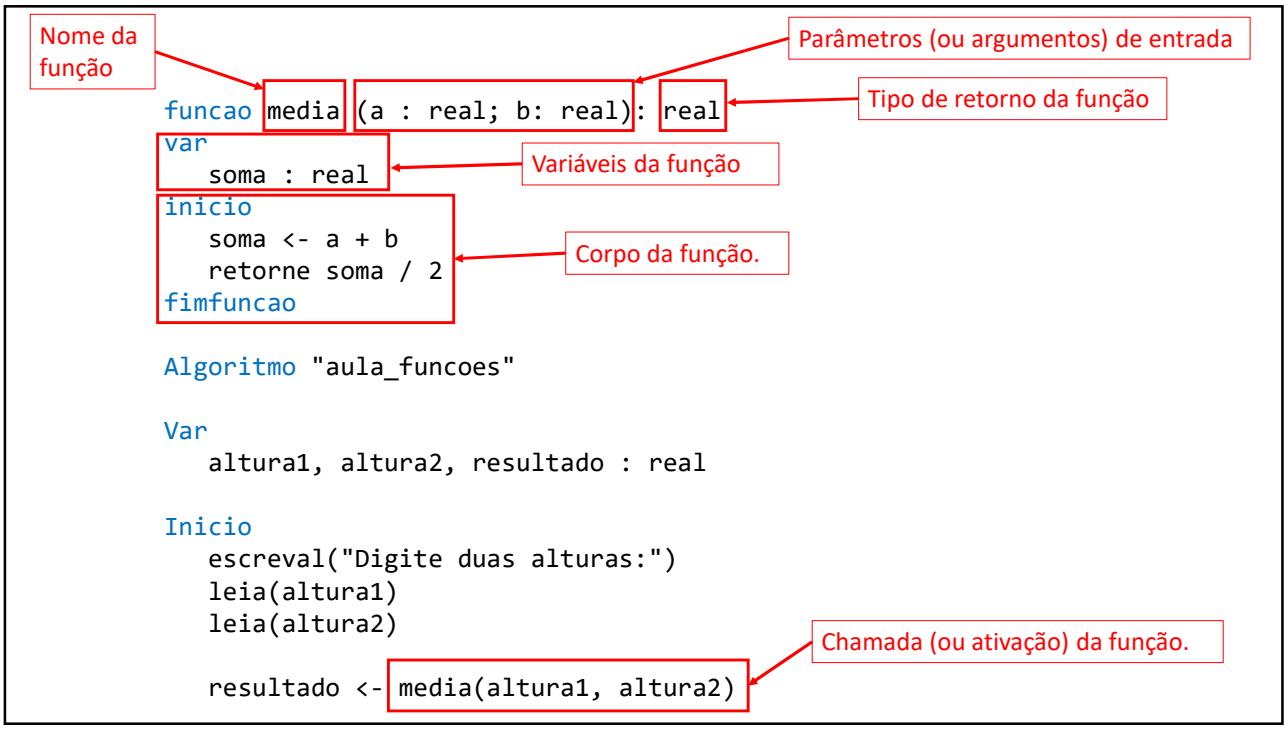
- Pode receber **parâmetros** (ou **argumentos**) de **entrada**
- Pode executar uma lógica
- Pode retornar um valor de **saída**

Qual a importância do uso de funções?

- Dividir um problema grande em problemas menores
- Organizar o código
  - Delegar funcionalidades
  - Legibilidade
- Reaproveitar código

```
funcao media(a : real; b: real): real
var
   soma : real
inicio
   soma <- a + b
   retorne soma / 2
fimfuncao

Algoritmo "aula_funcoes"

Var
   altura1, altura2, resultado : real

Inicio
   escreval("Digite duas alturas:")
   leia(altura1)
   leia(altura2)

   resultado <- media(altura1, altura2)

   escreval("MEDIA = ", resultado:4:2)
Fimalgoritmo
```

5



6

Passagem de parâmetros

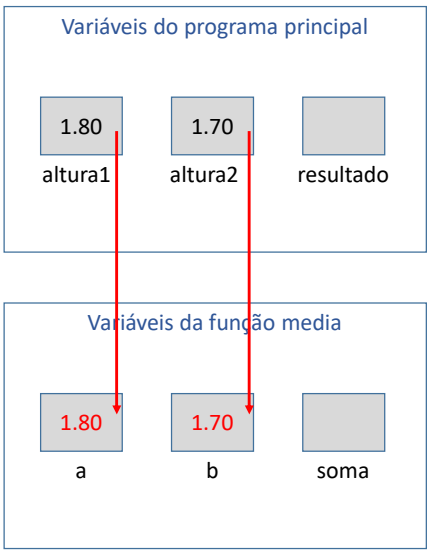
https://devsuperior.com.br

Prof. Dr. Nelio Alves

Passagem de parâmetros

Quando uma função é chamada (ou ativada), os valores passados entre parêntesis são **copiados** para as variáveis declaradas nos parâmetros de entrada da função, **na mesma ordem** em que foram passados na chamada da função.

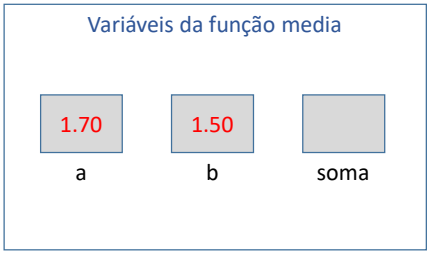
```
resultado <- media(altura1, altura2)
```



Passagem de parâmetros

Também é possível chamar a função passando valores diretamente como parâmetro

```
resultado <- media(1.7, 1.5)
```



Comando “retorne”

O comando “retorne” serve para definir qual o valor de saída da função

```
retorne soma / 2
```

Este valor de saída da função pode ser atribuído a uma variável, ou utilizado como parte de outra expressão.

```
resultado <- media(1.7, 1.5)

escreval("MEDIA = ", media(1.7, 1.5))

resultado <- 10 * media(1.7, 1.5)
```

Escopo de variáveis

https://devsuperior.com.br

Prof. Dr. Nelio Alves

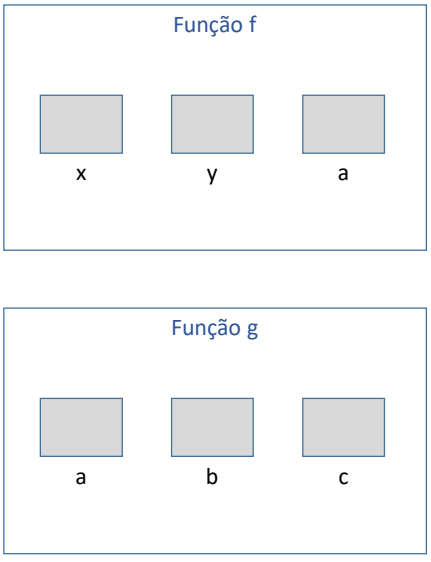
Escopo de variáveis

Cada função do programa tem suas próprias variáveis.

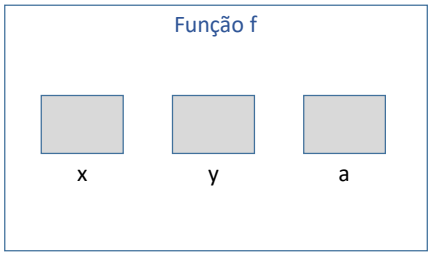
As variáveis de cada função somente "existem" no escopo de execução da sua função.

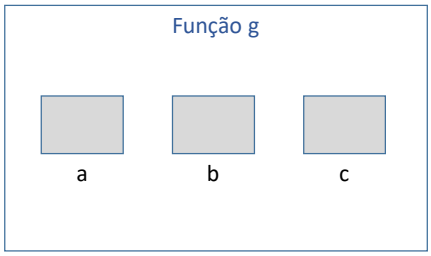
Uma função não "enxerga" as variáveis de outra função

Mesmo que funções diferentes tenham variáveis com o mesmo nome, são variáveis diferentes.



```
funcao f(x : inteiro; y: inteiro): inteiro
var a : inteiro
inicio
   a <- x + y
   retorne a * 10
fimfuncao

funcao g(a : inteiro; b: inteiro): inteiro
var c : inteiro
inicio
   c <- a * b
   retorne c
fimfuncao

Algoritmo "aula_escopo"

Var
   r1, r2 : inteiro

Inicio
   r1 <- f(3, 4)
   r2 <- g(5, 6)
   escreval(r1)
   escreval(r2)
Fimalgoritmo
```





13

Atenção: escopo global

- No VisualG, as variáveis do programa principal possuem escopo global, o que significa que funções podem acessar essas variáveis.

- Porém, dentro da função, deve-se evitar acessar variáveis fora da dela, pois:

  1) É indesejável que a função cause **efeitos colaterais**.

  2) É desejável que o resultado da função seja sempre o mesmo para os mesmos valores de entrada (**função pura**).

  https://pt.stackoverflow.com/questions/255557/o-que-%C3%A9-uma-fun%C3%A7%C3%A3o-pura

14

Exercícios resolvidos

https://devsuperior.com.br

Prof. Dr. Nelio Alves

Problema exemplo

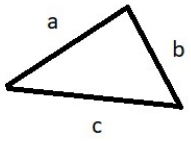
Fazer um programa para ler as medidas dos lados de dois triângulos X e Y (suponha medidas válidas). Em seguida, mostrar o valor das áreas dos dois triângulos.

A fórmula para calcular a área de um triângulo a partir das medidas de seus lados a, b e c é a seguinte (fórmula de Heron):

$$area = \sqrt{p(p-a)(p-b)(p-c)} \qquad \text{onde} \qquad p = \frac{a+b+c}{2}$$

Exemplo:

```
Entre com as medidas do triângulo X:
3.00
4.00
5.00
Entre com as medidas do triângulo Y:
7.50
4.50
4.02
Area de X = 6.0000
Area de Y = 7.5638
```



```
funcao areaTriangulo(a : real; b: real; c: real): real
var
   p : real
inicio
   p <- (a + b + c) / 2
   retorne Raizq(p * (p - a) * (p - b) * (p - c))
fimfuncao

Algoritmo "areas_triangulos"

Var
   xa, xb, xc: real
   ya, yb, yc: real
   areax, areay: real

Inicio

   escreval("Entre com as medidas do triângulo X:")
   leia(xa)
   leia(xb)
   leia(xc)
   escreval("Entre com as medidas do triângulo Y:")
   leia(ya)
   leia(yb)
   leia(yc)

   areax <- areaTriangulo(xa, xb, xc)
   areay <- areaTriangulo(ya, yb, yc)

   escreval("Area de X = ", areax:6:4)
   escreval("Area de Y = ", areay:6:4)

Fimalgoritmo
```


Problema exemplo

Fazer um programa para ler três números inteiros, e mostrar o menor dentre eles.

Exemplo:

```
Digite tres numeros inteiros:
9
5
7
Menor = 5
```

```
funcao menorDeTres(x : inteiro; y: inteiro; z: inteiro): inteiro
inicio
   se (x < y) e (x < z) entao
      retorne x
   senao
      se (y < z) entao
         retorne y
      senao
         retorne z
      fimse
   fimse
fimfuncao

Algoritmo "menor_de_tres"

Var
   n1, n2, n3, menor: inteiro

Inicio

   escreval("Digite tres números inteiros:")
   leia(n1)
   leia(n2)
   leia(n3)

   menor <- menorDeTres(n1, n2, n3)

   escreval("Menor = ", menor)

Fimalgoritmo
```


Problema exemplo

Fazer um programa para ler cinco números inteiros, e mostrar o menor dentre eles.

Exemplo:

```
Digite cinco numeros inteiros:
9
5
3
12
7
Menor = 3
```

```
funcao menorDeTres(x : inteiro; y: inteiro; z: inteiro): inteiro
inicio
   se (x < y) e (x < z) entao
      retorne x
   senao
      se (y < z) entao
         retorne y
      senao
         retorne z
      fimse
   fimse
fimfuncao

Algoritmo "menor_de_cinco"

Var
   n1, n2, n3, n4, n5, aux, menor: inteiro

Inicio

   escreval("Digite cinco números inteiros:")
   leia(n1)
   leia(n2)
   leia(n3)
   leia(n4)
   leia(n5)

   aux <- menorDeTres(n1, n2, n3)
   menor <- menorDeTres(aux, n4, n5)

   escreval("Menor = ", menor)

Fimalgoritmo
```


Funções podem chamar outras funções

https://devsuperior.com.br

Prof. Dr. Nelio Alves

Funções podem chamar outras funções

Fazer um programa para ler cinco números inteiros, e mostrar o menor dentre eles.

Exemplo:

```
Digite cinco numeros inteiros:
9
5
3
12
7
Menor = 3
```



```
funcao menorDeTres(x : inteiro; y: inteiro; z: inteiro): inteiro
inicio
   se (x < y) e (x < z) entao
      retorne x
   senao
      se (y < z) entao
         retorne y
      senao
         retorne z
      fimse
   fimse
fimfuncao

funcao menorDeCinco(x : inteiro; y: inteiro; z: inteiro; w: inteiro; u: inteiro): inteiro
var
   aux: inteiro
inicio
   aux <- menorDeTres(x, y, z)
   retorne menorDeTres(aux, w, u)
fimfuncao

Algoritmo "menor_de_cinco"

Var
   n1, n2, n3, n4, n5, menor: inteiro

Inicio
   escreval("Digite cinco numeros inteiros:")
   leia(n1)
   leia(n2)
   leia(n3)
   leia(n4)
   leia(n5)

   menor <- menorDeCinco(n1, n2, n3, n4, n5)

   escreval("Menor = ", menor)
Fimalgoritmo
```

Documentação de funções

https://devsuperior.com.br

Prof. Dr. Nelio Alves

Documentação de uma função

A documentação de uma função geralmente é composta de três informações:

- A **assinatura** (primeira linha) da função
- A descrição dos parâmetros de entrada
- O efeito da função

Documentação de uma função

Exemplo:

```
// Parametros:
//   a: primeiro numero
//   b: segundo numero
// Efeito: retorna a media aritmetica entre a e b
funcao media(a : real; b: real): real
```


Documentação de uma função

Exemplo:

```
// Parametros:
//   a: primeiro lado do triangulo
//   b: segundo lado do triangulo
//   c: terceiro lado do triangulo
// Efeito: retorna a area do triangulo
funcao areaTriangulo(a : real; b: real; c: real): real
```

Funções que não retornam valores (procedimentos)

https://devsuperior.com.br

Prof. Dr. Nelio Alves

Funções que não retornam valor

- Também são chamadas de procedimentos
- A chamada (ou ativação) da função é feita de forma “solta”, sem atribuí-la para alguma variável ou incluí-la em uma expressão.
- Em várias linguagens modernas, funções que não retornam valor são especificadas com o tipo de retorno “void”

Problema exemplo

Ler os dados de um contrato de financiamento (nome do cliente, valor total financiado e valor de entrada), depois mostrar um relatório conforme exemplo.

Exemplo:

```
Nome: Maria
Valor total: 200000.00
Valor de entrada: 30000.00

RELATORIO
NOME: Maria
VALOR TOTAL: 200000.00
VALOR DE ENTRADA: 30000.00
VALOR PARCELADO: 170000.00
```



```
procedimento mostrarRelatorio(nome: caractere; total: real; entrada: real)
var
   restante: real
inicio
   restante <- total - entrada
   escreval("RELATORIO")
   escreval("NOME: ", nome)
   escreval("VALOR TOTAL: ", total:4:2)
   escreval("VALOR DE ENTRADA: ", entrada:4:2)
   escreval("VALOR PARCELADO: ", restante:4:2)
fimprocedimento

Algoritmo "aula_procedimento"

Var
   nomeCliente: caractere
   total: real
   entrada: real

Inicio

   escreva("Nome: ")
   leia(nomeCliente)
   escreva("Valor total: ")
   leia(total)
   escreva("Valor de entrada: ")
   leia(entrada)

   mostrarRelatorio(nomeCliente, total, entrada)

Fimalgoritmo
```