Fundamentos de Programação 1

Modularização

“**Funções e Procedimentos**”

Slides 12

Prof. SIMÃO

Jean Marcelo SIMÃO

Função e Procedimentos

Funções e procedimentos permitem modularizar algoritmos.

A diferença fundamental de uma função e um procedimento é que a função retorna um valor explicitamente enquanto o procedimento não.

De certa forma, até pode-se dizer que um procedimento é uma função que não retorna nada explicitamente.

Exemplo – Variável global x local.

```
algoritmo "cálculo de áreas"
var
// Variável global
area : real

procedimento area_quadrado ( )
var
   // variável local
   lado  : real

inicio
   escreval ( "Informe o valor do lado" )
   leia ( lado )
   area <- lado * lado
fimprocedimento

procedimento area_triangulo retangulo ( )
var
   // variáveis locais
   lado1, lado2   :  real

inicio
   escreval ( "Informe o valor do lado A" )
   leia ( lado1 )
   escreval ( "Informe o valor do lado B" )
   leia ( lado2 )
   area <- ( lado1 * lado2 ) /2
fimprocedimento
```

```
// parte-principal do algoritmo

// variável local
opcao : inteiro

inicio
  escreval ( "Cálculo de Áreas." )
  escreval ( "0 – Área de um quadrado." )
  escreval ( "1 – Área de um triang. retang.." )
  escreval ( "Informe sua opção: " )
  leia ( opcao )

  escolha ( opcao )
     caso 0
         area_quadrado ( )
         escreval ( " O calculo da área é ", area )

    caso 1
         area_triangulo retangulo ( )
         escreval ( "O calculo da área é ", area )

    outrocaso
          escreval ( "Opção Inválida" )
  fimescolha

fimalgoritmo
```

Exercício 0

- Refaça o exemplo anterior, mas agora validando as variáveis.

- Aplicar este mesmo exercício para cada exemplo que será apresentado desta aula...

Exercício 1

- Elabore um algoritmo, usando procedimentos e variáveis globais, que permita ao usuário escolher entre várias opções até que ele escolha a opção ‘sair’. Estas opções contemplam os seguintes cálculos:
  - Cálculo do perimetro de uma circunferência cujo valor de raio é fornecido pelo usuário.
  - Cálculo da área de um retângulo cujos valores dos lados são fornecidos pelo usuário.
  - Cálculo da área de um triângulo retângulo cujos valores dos lados são fornecidos pelo usuário.
  - Cálculo do volume de um cubo cujo valor do lado…
  - Cálculo do volume de uma esfera cujo valor do raio…

Obs. Não deixe de validar as variáveis…

Função

```
algoritmo "Cálculo de Áreas"
var
// sem variável global
// teste  : real

funcao area_quadrado(): real
var
   // variável local
   lado, ar  : real
inicio
   escreval ( "Informe o valor do lado" )
   leia ( lado )
   ar <- lado * lado
   retorne ar
fimfuncao

funcao area_trianguloretangulo() : real
var
   // variáveis locais
   lado1, lado2, ar :  real
inicio
   escreval ( " Informe o valor do lado A " )
   leia ( lado1 )
   escreval ( " Informe o valor do lado B " )
   leia ( lado2 )
   ar <- ( lado1 * lado2 ) /2
   retorne ar
fimfuncao
```

```
// parte-principal do algoritmo

// variável local
opcao : inteiro
area  : real

inicio

  escreval ( "Cálculo de Áreas." )
  escreval ( "0 – Área de um quadrado." )
  escreval ( "1 – Área de um triang. retang.." )
  escreval ( "Informe sua opção: " )
  leia ( opcao )

  escolha ( opcao )

    caso 0
        area <- area_quadrado ()
        escreval ( "O calculo da área é ", area )
    caso 1
        area <- area_trianguloretangulo ()
        escreval ( "O calculo da área é ", area )
    outrocaso
        escreval ( "Opção Inválida" )

  fimescolha

fimalgoritmo
```

Exercício 2

- Refazer o exercício 1 usando funções com retorno.

- Qual é a vantagem de função com retorno, evitando variáveis globais?

Função com parâmetros

```
algoritmo "Cálculo de Áreas"
var
// sem variável global
// teste  : Real

funcao area_quadrado ( lado : real ): real
var
   // variável Local
   ar  : real

inicio

   ar <- lado * lado
   retorne ar

fimfuncao

funcao area_trianguloretang ( lado1 :real; lado2 :real ): real
var
   // variáveis locais
   ar :  real

inicio

   ar <- ( lado1 * lado2 ) / 2
   retorne ar

fimfuncao
```

```
// parte-principal do algoritmo
// variável local
   opcao : inteiro
   area, lado1, lado2  : real
inicio
  escreval ( "Cálculo de Áreas.")
  escreval ( "0 – Área de um quadrado.")
  escreval ( "1 – Área de um triang. retang..")
  escreval ( "Informe sua opção: ")
  leia ( opcao )

  escolha ( opcao )
    caso 0
        escreval ( "Informe o valor do lado" )
        leia ( lado1 )
        area <- area_quadrado( lado1 )
        escreval ( "O calculo da área é ", area )

    caso 1
        escreval ( "Informe o valor do lado A" )
        leia ( lado1 )
        escreval ( "Informe o valor do lado B" )
        leia ( lado2 )
        area <- area_trianguloretang ( lado1, lado2 )
        escreval ( "O calculo da área é ", area )

    outrocaso
        escreval ( "Opção Inválida" )

  fimescolha
fimalgoritmo
```

Exercício 3

- Refazer o exercício 2 usando funções com parâmetro(s).

- Qual é a a diferença dessa versão do algoritmo para com a anterior?

Função com parâmetros por valor

Na verdade, as funções utilizadas no exemplo anterior são classificadas como funções com parâmetros por valor. Eis mais um exemplo de função com parâmetro por valor.

```
algoritmo "Cálculo de Áreas"
var   // teste  : real

funcao area_quadrado ( lado : real ) : real
inicio

   lado  <- lado * lado

   retorne lado

fimfuncao
```

```
// parte-principal do algoritmo
var
   area, lado   : real

inicio

   escreval ( "Informe o valor do lado" )
   leia ( lado )

   escreval ( "O valor lido é: ", lado )

   area <- area_quadrado ( lado )

   escreval ( "O calculo da área é: ", area )

   escreval ( "O valor lido inicialmente foi: ", lado )

fimalgoritmo
```

Se *lado* for lido com o valor quatro (4), por exemplo, então:
- o primeiro *escreval* apresentará este valor quatro;
- o segundo *escreval* apresentará o valor dezesseis calculado na função; e
- o terceiro *escreval* apresentará ainda o valor quatro.

Isto porque um parâmetro por valor modificado na função NÃO é considerado onde a função foi chamada.

```
algoritmo "cálculo de áreas"
var    // teste  : Real

funcao area_quadrado ( ld : real ): real
Inicio

    ld  <- ld * ld

   retorne ld

fimfuncao
```

```
//Parte-Principal do Algoritmo
var
    area, lado    : real

inicio

   escreval ( "Informe o valor do lado" )
   leia ( lado )

   escreval ( "O valor lido é: ", lado )

   area ← area_quadrado ( lado )

   escreval ( "O calculo da área é: ", area )

   escreval ( "O valor lido inicialmente foi: ", lado )

fimalgoritmo
```

Função com parâmetros por referência

A função utilizada no exemplo precedente pode ser modificada de forma a ser uma função com parâmetros por referência. Eis a modificação:

```
algoritmo "Cálculo de Áreas"
var   // teste  : real

funcao area_quadrado ( var lado : real ): real

inicio
   lado <- lado * lado
   retorne lado
fimfuncao
```

A palavra *var* define que o parâmetro é por referência.

```
// parte-principal do algoritmo
var
   area, lado   : real

inicio
   escreval ( "Informe o valor do lado" )
   leia ( lado )

   escreval ( "O valor lido é: ", lado )

   area ← area_quadrado ( lado )

   escreval ( "O calculo da área é: ", area )

   escreval ( "O valor do 'lado' agora é: ", lado )

fimalgoritmo
```

Neste segundo exemplo, se *lado* for lido com o valor quatro (4), por exemplo, então:
- o primeiro *escreval* apresentará este valor quatro;
- o segundo *escreval* apresentará o valor dezesseis calculado na função; e
- o terceiro *escreval* apresentará agora o valor dezesseis.

Isto porque um parâmetro por referência modificado na função **é** considerado onde a função foi chamada.

Procedimento com parâmetros por referência.

```
algoritmo "cálculo deÁreas"
var // teste  : real

procedimento area_trianguloretangulo ( lado1 :real; lado2 : real; var ar : real )
inicio
   ar <- ( lado1 * lado2 ) / 2
fimprocedimento

procedimento area_quadrado ( lado : real; var ar : real )
inicio
   ar <- lado * lado
fimprocedimento

// parte-principal do algoritmo
// variável local
opcao : inteiro
area, lado1, lado2  : real
```

```
inicio
  area <- 0
  escreval ( " Cálculo de Áreas.")
  escreval ( " 0 – Área de um quadrado." )
  escreval ( " 1 – Área de um triang. retang.. " )
  escreval ( " Informe sua opção: " )
  leia ( opcao )
  escolha ( opcao )
    caso 0
        escreval ( " Informe o valor do lado " )
        leia ( lado1 )
        area_quadrado ( lado1, area )
        escreval ( " O calculo da área é ", area)
    caso 1
        escreval ( " Informe o valor do lado A" )
        leia ( lado1 )
        escreval ( " Informe o valor do lado B " )
        leia ( lado2 )
        area_trianguloretangulo ( lado1, lado2, area )
        escreval ( "O calculo da área é ", area )
    outrocaso
        escreval ( "Opção Inválida" )
  fimescolha
fimalgoritmo
```

Exercício 4

- Refazer o exercício 3 usando funções com parâmetro(s) por referência.

- Qual é a a diferença dessa versão do algoritmo para com a anterior?

Outros Exercícios

sobre modularidade.

Exercícios A

a) Elabore um algoritmo que:

- leia as coordenadas de um retângulo (x1,y1), (x2,y2), (x3, y3), (x4, y4)
- calcule o perímetro do retângulo formado pelas coordenadas lidas
- imprima o perímetro do retângulo

Obs.: Cada tarefa deve ser realizada por uma função. Utilize apenas variáveis globais.

b) Faça um algoritmo que:

- leia 3 valores fornecidos pelo usuário.
- verifique se estes valores formam um triângulo e o classifique como eqüilátero, isósceles ou escaleno.

Obs.: Cada tarefa deve ser realizada por uma função. Utilize apenas variáveis globais.

c) Escreva uma função que receba como parâmetro os comprimentos dos lados de um triângulo (a, b, c) e retorne os seguintes valores:

- 1 se o triângulo for retângulo.
- 2 se o triângulo for obtusângulo.
- 3 se o triângulo for acutângulo.
- 0 se nenhum triângulo é formado.

Exercícios B

a) Elabore um algoritmos que leia um valor e imprima:
   - se o valor é par ou ímpar
   - se é divisível por 5
   - seu valor absoluto (módulo)
   - seu fatorial.
   - sua tabuada.

   Obs.: Utilize uma função para executar cada uma das tarefas anteriores e utilize somente variáveis locais e parâmetros (não use variáveis globais).

b) Elabore um algoritmo que calcule o valor de π, em uma função, através da série:

$$S = 1 - \frac{1}{3^3} + \frac{1}{5^3} + \frac{1}{7^3} + ...$$

   Deverá ser fornecido à função o número de termos da série para o cálculo de π via uma parâmetro por 'valor' e o valor calculado de π deverá ser retornado via um parâmetro por 'referência'.

c) Escreva uma outra função que forneça sucessivamente a função anterior (por parâmetro) os seguintes termos: 1, 2, 3, 4, 5, ..., n. Para cada termo, a função deverá imprimir o termo e o valor correpondente calculado para π. O valor de n deverá ser fornecido pelo usuário e passado para a função via parametro por valor.