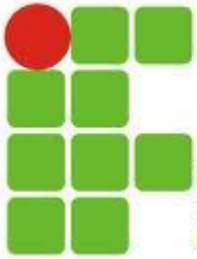


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

AULA 12

FUNÇÕES

Disciplina: Algoritmos e POO

Professora: Alba Lopes

alba.lopes@ifrn.edu.br

http://docente.ifrn.edu.br/albalopes

Funções e Procedimentos

- Também chamados de **subalgoritmos**
- São trechos de algoritmos que efetuam um ou mais cálculos determinados
- Ao invés de escrever um código grande, pode-se escrever vários algoritmos menores **(Modularização)**
  - Em conjunto, resolvem o problema proposto
- É conveniente utilizá-los quando uma tarefa é efetuada em diversos lugares no mesmo algoritmo
- Ao invés de escrever um trecho diversas vezes, escreve-se um subalgoritmo e chama-o diversas vezes

Funções e Procedimentos

- Reduzem o tamanho do algoritmo
- Facilitam a compreensão e visualização do algoritmo
- São declarados no início do algoritmo e podem ser chamados em quaisquer pontos após sua declaração
- Podem ser:
  - **Funções** que retornam algum valor
  - **Procedimento** (ou subrotina) que não retorna nada

Funções

- Uma função é um bloco de comandos que tem por objetivo retornar um valor ou uma informação
- A chamada de uma função é feita através da citação do seu nome seguido, opcionalmente, de seus argumentos iniciais entre parênteses
- As funções podem ser **predefinidas** pela linguagem ou **criadas** pelo programador

Funções

- **Funções predefinidas**
  - A linguagem do Visualg possui diversas funções predefinidas que podem ser usadas na construção de algoritmos

  - Exemplo 1: Criar um algoritmo que calcule o valor da raiz quadrada de um número.
    - O Visualg possui uma função predefinida que recebe como parâmetro uma variável (do tipo real) e retorna um valor (também do tipo real) referente à raiz quadrada do número.

Funções

- **Funções predefinidas**
  - Assinatura da função para cálculo da Raiz Quadrada

Raizq(valor : real) : real

Nome da função

Parâmetro : Tipo do parâmetro

Tipo de retorno

Funções

- **Funções predefinidas**
  - Como utilizar

```
algoritmo "Uso_de_Funcao_Predefinida"
var
   numero, raiz: real
inicio
   escreva("Digite um número: ")
   leia(numero)
   raiz <- Raizq(numero)
   escreva("A raiz quadrada do número digitado é: ", raiz)
fimalgoritmo
```

Funções

- **Funções predefinidas**
  - Como utilizar

Chamada da função passando a variável **numero** (do tipo real) como parâmetro da função **Raizq**

```
algoritmo "Uso_de_Funcao_Predefini
var
   numero, raiz: real
inicio
   escreva("Digite um número: ")
   leia(numero)
   raiz <- Raizq(numero)
   escreva("A raiz quadrada do número digitado é: ", raiz)
fimalgoritmo
```

Funções

- **Funções predefinidas**
  - Como utilizar

```
algoritmo "Uso_de_Funcao_Predefi
var
   numero, raiz: real
inicio
   escreva("Digite um numero: ")
   leia(numero)
   raiz <- Raizq(numero)
   escreva("A raiz quadrada do número digitado é: ", raiz)
fimalgoritmo
```

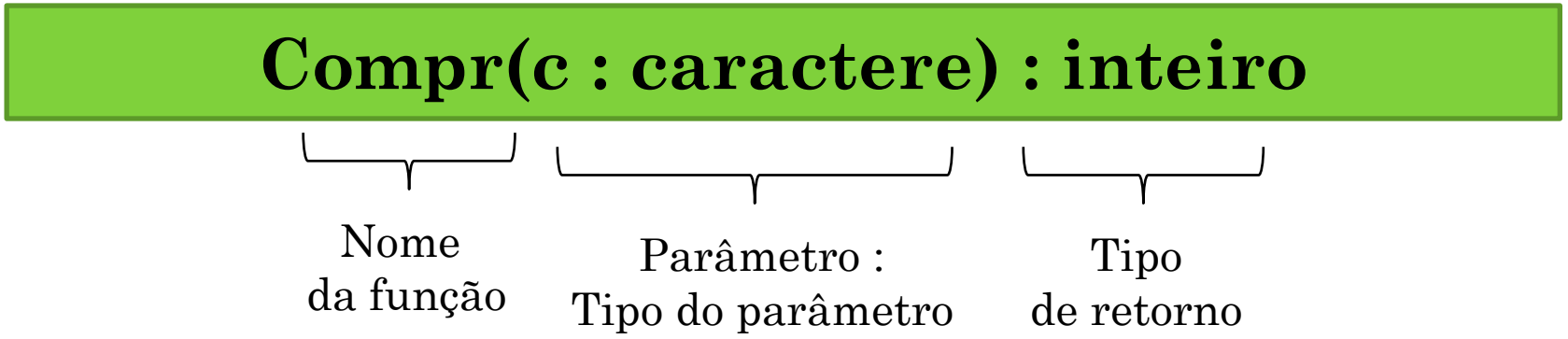
Atribuindo à variável **raiz** (do tipo real) o retorno da função **Raizq**

Funções

- **Funções predefinidas**
  - Exemplo 2: Criar um algoritmo que, dada uma palavra qualquer, informe ao usuário quantas letras essa palavra possui.
    - O Visualg possui uma função predefinida que recebe como parâmetro uma variável (tipo caractere) e retorna um valor (inteiro) correspondente à quantidade de caracteres existentes.

Funções

- **Funções predefinidas**
  - Assinatura da função para contar caracteres de uma palavra



Funções

- **Funções predefinidas**
  - Como utilizar

```
algoritmo "Uso_de_Funcao_Predefinida"
var
   palavra: caractere
   qtd: inteiro
inicio
   escreva("Digite uma palavra qualquer: ")
   leia(palavra)
   qtd <- Compr(palavra)
   escreva("A palavra que você digitou possui ", qtd, " caracteres")
fimalgoritmo
```

FUNÇÕES

- **Funções predefinidas**
  - Como utilizar

```
algoritmo "Uso_de_Funcao_Predefinida"
var
   palavra: caractere
   qtd: inteiro
inicio
   escreva("Digite uma palavra qualquer: ")
   leia(palavra)
   qtd <- Compr(palavra)
   escreva("A palavra que você digitou possui ", qtd, " caracteres")
fimalgoritmo
```

Chamada da função passando a variável **palavra** (do tipo caractere) como parâmetro da função **Compr**

Funções

- **Funções predefinidas**
  - Como utilizar

```
algoritmo "Uso_de_Funcao_Predefinida"
var
   palavra: caractere
   qtd: inteiro
inicio
   escreva("Digite uma palavra qualquer: ")
   leia(palavra)
   qtd <- Compr(palavra)
   escreva("A palavra que você digitou possui ", qtd, " caracteres")
fimalgoritmo
```

Atribuindo à variável **quantidade** (do tipo inteiro) o retorno da função **Compr**

Funções

- **Lista das Funções Predefinidas do Visualg**
  - Funções numéricas, algébricas e trigonométricas

Abs(expressão) - Retorna o valor absoluto de uma expressão do tipo inteiro ou real. Equivale a **| expressão |** na álgebra.
ArcCos(expressão) - Retorna o ângulo (em radianos) cujo co-seno é representado por expressão.
ArcSen(expressão) - Retorna o ângulo (em radianos) cujo seno é representado por expressão.
ArcTan(expressão) - Retorna o ângulo (em radianos) cuja tangente é representada por expressão.
Cos(expressão) - Retorna o co-seno do ângulo (em radianos) representado por expressão.
CoTan(expressão) - Retorna a co-tangente do ângulo (em radianos) representado por expressão.
Exp(base, expoente) - Retorna o valor de base elevado a expoente, sendo ambos expressões do tipo real.
GraupRad(expressão) - Retorna o valor em radianos correspondente ao valor em graus representado por expressão.
Int(expressão) - Retorna a parte inteira do valor representado por expressão.

Funções

- **Lista das Funções Predefinidas do Visualg**
  - Funções numéricas, algébricas e trigonométricas

Log(expressão) - Retorna o logaritmo na base 10 do valor representado por expressão. **LogN(expressão)** - Retorna o logaritmo neperiano (base e) do valor representado por expressão.
Pi - Retorna o valor 3.141592.
Quad(expressão) - Retorna quadrado do valor representado por expressão.
RadpGrau(expressão) - Retorna o valor em graus correspondente ao valor em radianos representado por expressão.
RaizQ(expressão) - Retorna a raiz quadrada do valor representado por expressão.
Rand - Retorna um número real gerado aleatoriamente, maior ou igual a zero e menor que um.
RandI(limite) - Retorna um número inteiro gerado aleatoriamente, maior ou igual a zero e menor que limite.
Sen(expressão) - Retorna o seno do ângulo (em radianos) representado por expressão.
Tan(expressão) - Retorna a tangente do ângulo (em radianos) representado por expressão.

Funções

- **Lista das Funções Predefinidas do Visualg**
  - Funções para manipular cadeias de caracteres

Asc (s : caracter) : Retorna um inteiro com o código ASCII do primeiro caracter da expressão.
Carac (c : inteiro) : Retorna o caracter cujo código ASCII corresponde à expressão.
Caracpnum (c : caracter) : Retorna o inteiro ou real representado pela expressão. Corresponde a StrToInt() ou StrToFloat() do Delphi, Val() do Basic ou Clipper, etc.
Compr (c : caracter) : Retorna um inteiro contendo o comprimento (quantidade de caracteres) da expressão.
Copia (c : caracter ; p, n : inteiro) : Retorna um valor do tipo caracter contendo uma cópia parcial da expressão, a partir do caracter p, contendo n caracteres. Os caracteres são numerados da esquerda para a direita, começando de 1. Corresponde a Copy() do Delphi, Mid$() do Basic ou Substr() do Clipper.
Maiusc (c : caracter) : Retorna um valor caracter contendo a expressão em maiúsculas.
Minusc (c : caracter) : Retorna um valor caracter contendo a expressão em minúsculas.
Numpcarac (n : inteiro ou real) : Retorna um valor caracter contendo a representação de n como uma cadeia de caracteres. Corresponde a IntToStr() ou FloatToStr() do Delphi, Str() do Basic ou Clipper.
Pos (subc, c : caracter) : Retorna um inteiro que indica a posição em que a cadeia subc se encontra em c, ou zero se subc não estiver contida em c. Corresponde funcionalmente a Pos() do Delphi, Instr() do Basic ou At() do Clipper, embora a ordem dos parâmetros possa ser diferente em algumas destas linguagens.

Funções

- **Resumo (por ordem alfabética)**

FUNÇÃO	DESCRIÇÃO
Abs (valor : real) : real	Valor absoluto
Arccos (valor : real) : real	Arco cosseno
Arcsen (valor : real) : real	Arco seno
Arctan (valor : real) : real	Arco tangente
Asc (s : caracter) : inteiro	Retorna o código ASCII
Compr (c : caracter) : inteiro	Retorna a dimensão do caractere
Copia (c : caracter , posini, posfin : inteiro) : caracter	Copia um determinado trecho do caractere
Cos (valor : real) : real	Cosseno
Cotan (valor : real) : real	Co-tangente
Exp (<base>,<expoente>)	Potenciação
Grauprad (valor : real) : real	Converte grau para radiano
Int (valor : real) : inteiro	Converte o valor em inteiro
Log (valor : real) : real	Logaritmo de base 10
Logn (valor : real) : real	Logaritmo natural (ln)

Maiusc (c : caracter) : caracter	Converte em Maiúscula
Minusc (c : caracter) : caracter	Converte em Minúscula
Numpcarac (n : inteiro ou real) : caracter	Converte um numero inteiro ou real para caractere
Pi : real	Valor Pi
Pos (subc, c : caracter) : inteiro	Retorna a posição do caractere.
Quad (valor : real) : real	Elevado quadrado
Radpgrau (valor : real) : real	Converte Radiano para grau.
Raizq (valor : real) : real	Raiz quadrada
Rand : real	Gerador de números aleatórios entre 0 e 1
Randi (limite : inteiro) : inteiro	Gerador de números inteiros aleatórios com um limite determinado
Sen (valor : real) : real	Seno
Tan (valor : real) : real	Tangente

Dica: *Pressionando* ***CTRL+J*** *o Visualg mostra uma Lista de funções predefinidas. Basta selecionar a desejada e dar ENTER. Depois é só passar os parâmetros desejados.*

Funções Predefinidas

- Exemplo3: Criar um algoritmo que, dado o nome de uma pessoa (Nome + Sobrenome), retorna apenas o sobrenome. Ex: “Alba Lopes” , retorna somente “Lopes”

```
algoritmo "Retorna_Sobrenome"
var
   nomeCompleto, sobrenome: caractere
   qtdCaracteres, localEspaco: inteiro
inicio
   nomeCompleto <- "Alba Lopes"
   qtdCaracteres <- Compr(nomeCompleto)
   localEspaco <- Pos(" ", nomeCompleto)
   sobrenome <- Copia(nomeCompleto, localEspaco + 1, qtdCaracteres)
   escreva("O sobrenome é: ", sobrenome)
fimalgoritmo
```

Exercícios

- **Utilizando funções predefinidas da linguagem do Visualg, resolva os seguintes exercícios:**

1. Construa um algoritmo que receba dois valores do usuário (**a** e **b**) e realize a subtração entre eles. Apresente sempre o resultado positivo do cálculo. Ex: Para **a = 2** e **b = 9**, o resultado de **a – b** deverá ser 7 (e não -7).
2. Construa um algoritmo que receba o nome completo de uma pessoa e escreva esse nome todo com letras maiúsculas
3. Construa um algoritmo que, tendo como dados de entrada dois pontos quaisquer no plano, P(x1,y1) e P(x2,y2), escreva a distância entre eles. A fórmula que efetua tal cálculo é: $d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$
4. Construa um algoritmo que resolva uma equação de segundo grau. Receba os três coeficientes **a**, **b** e **c** e informe as raízes da equação.
5. Construa um algoritmo que receba o e-mail de uma pessoa e verifique se o e-mail é válido. Considere que o e-mail é válido se possuir um @ e, no mínimo, 5 caracteres.

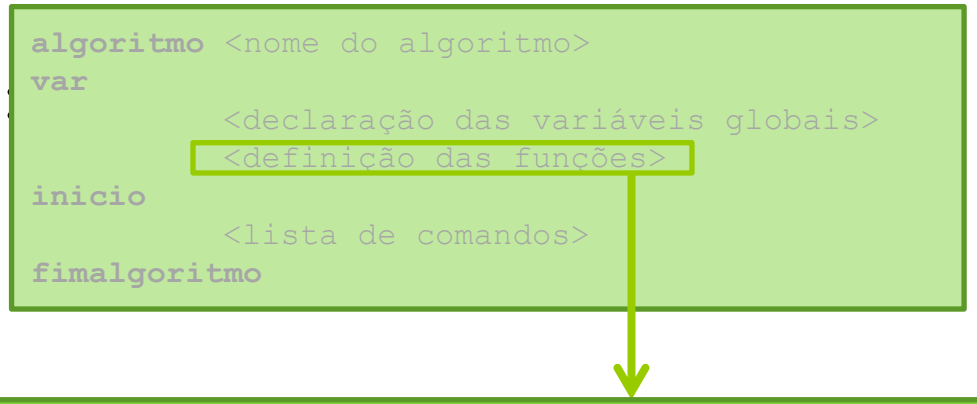
Criando Funções

- A criação de uma função deve ser realizada dentro da seção de variáveis
- Esse tipo de subalgoritmo sempre retorna apenas um valor para o algoritmo que o chamou
- As funções possuem um tipo de retorno associado
- Uma função pode possui 0, 1 ou mais parâmetros
- **Sintaxe**:

```
algoritmo <nome do algoritmo>
var
       <declaração das variáveis globais>
       <definição das funções>
inicio
       <lista de comandos>
fimalgoritmo
```

Criando Funções

- **Sintaxe da função:**




```
funcao <nome do funcao> (<parâmetros>) <tipo de retorno>
var
      <declaração das variáveis locais>
inicio
      <lista de comandos>
      retorne <variável de retorno>
fimfuncao
```

Criando Funções

- Variáveis Locais:
  - Declaradas dentro dos subalgoritmos (funções ou procedimentos)
  - Podem ser usadas **APENAS** dentro das funções
  - O algoritmo que chamou a função/procedimento não tem acesso à estas funções
- Variáveis Globais
  - São variáveis declaradas na seção **var** do algoritmo. Qualquer função/procedimento pode alterar o valor ou utilizá-la durante o seu processamento.

Criando Funções

- **Exemplo 1**: Criar uma função para calcular o dobro de um número passado como parâmetro

```
algoritmo "FuncoesPersonalizadas"
var
   numero, resultado: real

   funcao Dobro(valor: real) : real
   var
      total: real
   inicio
      total <- valor * 2
      retorne total
   fimfuncao

inicio

   escreva("Digite um número para calcular o dobro: ")
   leia (numero)
   resultado <- Dobro(numero)
   escreva("O dobro é: ", resultado)

fimalgoritmo
```

Criando Funções

- **Exemplo 1**: Criar uma função para calcular o dobro de um número passado como parâmetro

```
algoritmo "FuncoesPersonalizadas"
var
   numero, resultado: real

   funcao Dobro(valor: real) : real
   var
      total: real
   inicio
      total <- valor * 2
      retorne total
   fimfuncao

inicio

   escreva("Digite um número para calcular o dobro: ")
   leia (numero)
   resultado <- Dobro(numero)
   escreva("O dobro é: ", resultado)

fimalgoritmo
```

VARIÁVEIS GLOBAIS

Criando Funções

- **Exemplo 1**: Criar uma função para calcular o dobro de um número passado como parâmetro

```
algoritmo "FuncoesPersonalizadas"
var
   numero, resultado: real

   funcao Dobro(valor: real) : real
   var
      total: real
   inicio
      total <- valor * 2
      retorne total
   fimfuncao

inicio

   escreva("Digite um número para calcular o dobro: ")
   leia (numero)
   resultado <- Dobro(numero)
   escreva("O dobro é: ", resultado)

fimalgoritmo
```

DECLARAÇÃO DE FUNÇÃO

Criando Funções

- **Exemplo 1**: Criar uma função para calcular o dobro de um número passado como parâmetro

```
algoritmo "FuncoesPersonalizadas"
var
   numero, resultado: real

   funcao Dobro(valor: real) : real
   var
      total: real
   inicio
      total <- valor * 2
      retorne total
   fimfuncao

inicio

   escreva("Digite um número para calcular o dobro: ")
   leia (numero)
   resultado <- Dobro(numero)
   escreva("O dobro é: ", resultado)

fimalgoritmo
```

COMANDOS

Criando Funções

- **Exemplo 1**: Criar uma função para calcular o dobro de um número passado como parâmetro

```
algoritmo "FuncoesPersonalizadas"
var
   numero, resultado: real

   funcao Dobro(valor: real) : real
   var
      total: real
   inicio
      total <- valor * 2
      retorne total
   fimfuncao

inicio

   escreva("Digite um número para calcular o dobro: ")
   leia (numero)
   resultado <- Dobro(numero)
   escreva("O dobro é: ", resultado)

fimalgoritmo
```

Nome da função

Criando Funções

- **Exemplo 1**: Criar uma função para calcular o dobro de um número passado como parâmetro

```
algoritmo "FuncoesPersonalizadas"
var
   numero, resultado: real

   funcao Dobro(valor: real) : real
   var
      total: real
   inicio
      total <- valor * 2
      retorne total
   fimfuncao

inicio

   escreva("Digite um número para calcular o dobro: ")
   leia (numero)
   resultado <- Dobro(numero)
   escreva("O dobro é: ", resultado)

fimalgoritmo
```

Parâmetro : tipo do parâmetro

Criando Funções

- **Exemplo 1**: Criar uma função para calcular o dobro de um número passado como parâmetro

```
algoritmo "FuncoesPersonalizadas"
var
   numero, resultado: real

   funcao Dobro(valor: real) : real
   var
      total: real
   inicio
      total <- valor * 2
      retorne total
   fimfuncao

inicio

   escreva("Digite um número para calcular o dobro: ")
   leia (numero)
   resultado <- Dobro(numero)
   escreva("O dobro é: ", resultado)

fimalgoritmo
```

Tipo de retorno

Criando Funções

- **Exemplo 1**: Criar uma função para calcular o dobro de um número passado como parâmetro

```
algoritmo "FuncoesPersonalizadas"
var
   numero, resultado: real

   funcao Dobro(valor: real) : real
   var
      total: real
   inicio
      total <- valor * 2
      retorne total
   fimfuncao

inicio

   escreva("Digite um número para calcular o dobro: ")
   leia (numero)
   resultado <- Dobro(numero)
   escreva("O dobro é: ", resultado)

fimalgoritmo
```

Declaração de variáveis locais

Criando Funções

- **Exemplo 1**: Criar uma função para calcular o dobro de um número passado como parâmetro

```
algoritmo "FuncoesPersonalizadas"
var
   numero, resultado: real

   funcao Dobro(valor: real) : real
   var
      total: real
   inicio
      total <- valor * 2
      retorne total
   fimfuncao

inicio

   escreva("Digite um número para calcular o dobro: ")
   leia (numero)
   resultado <- Dobro(numero)
   escreva("O dobro é: ", resultado)

fimalgoritmo
```

Comandos

Criando Funções

- **Exemplo 1**: Criar uma função para calcular o dobro de um número passado como parâmetro

```
algoritmo "FuncoesPersonalizadas"
var
   numero, resultado: real

   funcao Dobro(valor: real) : real
   var
      total: real
   inicio
      total <- valor * 2
      retorne total
   fimfuncao

inicio

   escreva("Digite um número para calcular o dobro: ")
   leia (numero)
   resultado <- Dobro(numero)
   escreva("O dobro é: ", resultado)

fimalgoritmo
```

Retorno

Criando Funções

- Os algoritmos podem possuir várias funções:

```
algoritmo "FuncoesPersonalizadas"
var
   numero: real

   funcao Dobro(valor: real) : real
   var
      total: real
   inicio
      total <- valor * 2
      retorne total
   fimfuncao

   funcao Triplo(valor: real) : real
   var
      total: real
   inicio
      total <- valor * 3
      retorne total
   fimfuncao

inicio

   escreva("Digite um número: ")
   leia (numero)
   escreval("O dobro é: ", Dobro(numero) )
   escreval("O triplo é: ", Triplo(numero) )

fimalgoritmo
```

Criando Funções

- Os algoritmos podem possuir várias funções:

```
algoritmo "FuncoesPersonalizadas"
var
   numero: real

   funcao Dobro(valor: real) : real
   var
      total: real
   inicio
      total <- valor * 2
      retorne total
   fimfuncao

   funcao Triplo(valor: real) : real
   var
      total: real
   inicio
      total <- valor * 3
      retorne total
   fimfuncao

inicio

   escreva("Digite um número: ")
   leia (numero)
   escreval("O dobro é: ", Dobro(numero) )
   escreval("O triplo é: ", Triplo(numero) )

fimalgoritmo
```

Criando Funções

- As funções podem possuir mais de um parâmetro
  - Parâmetros de um mesmo tipo são separados por vírgula
  - Parâmetros de tipos diferentes são separados por ponto e vírgula

- **Exemplo 2**: Criar uma função que receba três valores reais como parâmetro e retorne a média desses valores

Criando Funções

- **Exemplo 2:**

```
algoritmo "FuncoesPersonalizadas"
var
   numero1, numero2, numero3, resultado: real

   funcao Media(n1, n2, n3: real) : real
   var
      calculoMedia: real
   inicio
      calculoMedia <- (n1 + n2 + n3)/ 3
      retorne calculoMedia
   fimfuncao

inicio

   escreval("Digite 3 numeros: ")
   leia (numero1)
   leia (numero2)
   leia (numero3)
   resultado <- Media(numero1, numero2, numero3)
   escreva("O valor da média é: ", resultado)

fimalgoritmo
```

Criando Funções

- **Exemplo 3**: Escreva uma função que recebe as 4 notas de um aluno por parâmetro e uma letra. Se a letra for A a função calcula a média aritmética das notas do aluno, se for P, a sua média ponderada (pesos: 2, 3, 4 e 6).

Criando Funções

- **Exemplo 3:**

```
algoritmo "FuncoesPersonalizadas"
var
   numero1, numero2, numero3, resultado: real

   funcao MediaAouP( n1, n2, n3, n4: real ; tipo: caractere ) : real
   var
      calculo_media: real
   inicio
      escolha tipo
         caso "A"
            calculo_media <- (n1 + n2 + n3 + n4)/ 4
         caso "P"
            calculo_media <- (n1*2 + n2*3 + n3*4 + n4*6)/ 15
         outrocaso
            calculo_media <- 0
       fimescolha

      retorne calculo_media
   fimfuncao

inicio

   resultado <- MediaAouP(5, 6, 7, 8, "A")
   escreval("O valor da média é: ", resultado)

fimalgoritmo
```

Criando Funções

- Funções podem ser chamadas várias vezes durante a execução de um algoritmo:

```
                    ...
        retorne calculo_media
    fimfuncao

inicio

    resultado <- MediaAouP(5, 6, 7, 8, "A")
    escreval("O valor da média é: ", resultado)

    resultado <- MediaAouP(5, 6, 7, 8, "P")
    escreval("O valor da nova média é: ", resultado)

fimalgoritmo
```

Criando Funções

- As funções podem não possuir parâmetros.

- **Exemplo 4:** Crie uma função que leia um número não determinado de valores positivos e calcule a soma desses números. A função deve parar de ler números quando um número negativo for digitado e retornar a soma dos números lidos.

Criando Funções

- **Exemplo 4**

```
algoritmo "FuncaoSomaVarios"
var
   total: inteiro

   funcao SomaVarios : inteiro
   var
      numero, soma: inteiro
   inicio
      soma <- 0
      escreva("Digite um número: ")
      leia(numero)
      enquanto (numero >= 0) faca
         soma <- soma + numero
         escreva("Digite um número: ")
         leia(numero)
      fimenquanto
      retorne soma
   fimfuncao
inicio

   total <- SomaVarios
   escreva("O valor da soma é: ", total)

fimalgoritmo
```

Exercícios Funções

1. Crie uma função que recebe por parâmetro o raio de uma esfera e calcula o seu volume (v = 4/3 * pi * raio^3).
2. Crie uma função que receba por parâmetro um número inteiro e verifique se o número é positivo ou negativo. Retorne um valor lógico (**verdadeiro** ou **falso**)
3. Crie uma função que verifique se um determinado número é par. Retorne um valor lógico (**verdadeiro** ou **falso**)
4. Crie uma função que verifique se um determinado número é par ou ímpar. Retorne um valor do tipo caractere que informe o resultado (**par** ou **ímpar**)
5. Crie uma função que recebe por parâmetro um valor inteiro e positivo e verifica se esse valor é **primo**. Retorne o valor lógico **verdadeiro** caso o valor seja primo e **falso** em caso contrário.
6. Crie uma função que recebe a idade de uma pessoa em anos, meses e dias e retorna essa idade expressa em dias.
7. Escreva uma função que receba por parâmetro um número inteiro e retorne o fatorial desse número.
8. Faça uma função que recebe, por parâmetro, a altura (alt) e o sexo de uma pessoa e retorna o seu peso ideal. Para homens, calcular o peso ideal usando a fórmula peso ideal = 72.7 * alt - 58 e ,para mulheres, peso ideal = 62.1 * alt - 44.7.

Referências

- NAPRO – Núcleo de Apoio Aprendizagem de Programação. Disponível em: http://www.guanabara.info/logica/Apostilas/VisuAlg_Ref.pdf
- http://www.inf.pucrs.br/~pinho/LaproI/Exercicios/SeqDecisao/lista1.htm
- http://www.inf.pucrs.br/flash/lapro/listafunc.html