Unidade III

Nesta unidade trataremos de dois tipos de modularização: procedimentos e funções.

5 PROCEDIMENTOS

O projeto de algoritmo de problemas complexos envolve muitos blocos de códigos que possuem propósitos específicos para os quais poderiam ser dados nomes significativos.

A estrutura do algoritmo conforme apresentada até aqui constitui um módulo principal, no qual todos os comandos são executados. O módulo principal é delimitado pelas diretivas **início** e **fim**.

Lembrete

Algoritmos são processos de solução de problemas.

Para problemas grandes e complexos, a decomposição em problemas menores ajuda a reduzir a complexidade e torna o algoritmo mais organizado e compreensível.

Modularizar um algoritmo é dividi-lo em partes, ou seja, estabelecer um módulo principal que faz chamada de outros módulos específicos para resolver problemas menores. Se o algoritmo é uma rotina que estrutura uma solução, os módulos são sub-rotinas, conforme pode ser observado na figura seguinte, a qual ilustra um algoritmo para realizar as operações de uma calculadora, a saber: adição, subtração, multiplicação e divisão. Tais operações são módulos do algoritmo com codificação fora do módulo principal.

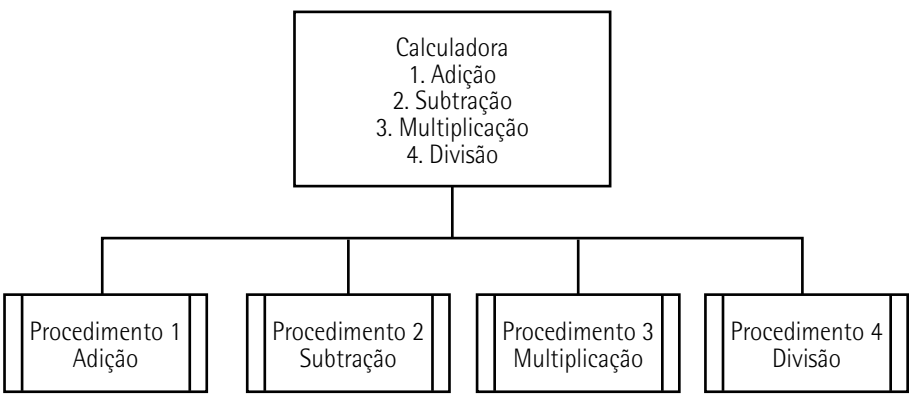


Figura 110 – Fluxograma da modularização

Em algoritmos, há dois tipos de módulos: **procedimentos** ou **funções**. Ambos são sub-rotinas que possuem uma funcionalidade com objetivo bem definido e que devem receber nomes significativos de acordo como seu propósito.

procedimento <nome>([parâmetros]) var //declaração de variáveis locais início <comando 1> <comando 2> : <comando n> fimprocedimento	Função <nome>([parâmetros]): <tipo> var //declaração de variáveis locais início <comando 1> <comando 2> : <comando n>~ retorne <valor> fimfunção

Figura 111 – Sintaxes para procedimentos e funções

Observe que na função há um valor de retorno obrigatório. Em procedimentos, não há retorno. Por esse motivo, na declaração da função, é necessário especificar o tipo do valor de retorno. Se a função retornar um valor inteiro, o valor de retorno também deverá ser do tipo inteiro.

Os módulos de procedimentos ou funções devem ser especificados fora do módulo principal e invocados dentro do módulo principal. A figura seguinte mostra a estrutura sintática de um algoritmo com a especificação de um procedimento e uma função.

A lista de parâmetros de procedimentos e funções poderá ser vazia e um programa poderá especificar um ou vários procedimentos e funções.

Um procedimento é especificado entre as linhas 6 e 10 e na linha 28 é feita a chamada do procedimento no módulo principal. Nas linhas de 13 a 22 é especificada uma função e as linhas 29 e 30 do módulo principal apresentam as formas para invocar funções.

```
1.  Algoritmo "Nome Algoritmo"
2.  Var
3.    //espaco para declaracao de variaveis
4.
5.  //modulo de procedimento
6.  procedimento <nome>([Parâmetros])
7.          //variáveis locais
8.  início
9.          //bloco de comandos
10. fimprocedimento
11.
12. //módulo de funcao
13. funcao <nome>([Parâmetros]) : <tipo>
14.         //variaveis locais
15.         <var_valor_retorno> : <tipo>
16. inicio
17.         //bloco de comandos
18.         // a variavel valor deve receber
19.         // o tipo da variavel de retorno deve ser o mesmo da          // funcao
20.         retorne < var_valor_retorno >
21. fimfuncao
22.
23. //modulo principal
24. Início
25.   :
26.   :
27.   <nomeProcedimento>([Parâmetros])
28.   <variavel> ← <nomeFuncao>([Parâmetros])
29.   escreva(<nomeFuncao>([Parâmetros]))
30.   :
31.   :
32. Fimalgoritmo
```

Figura 112 – Sintaxe para a especificação de procedimentos e funções e invocação no módulo principal

A lista de parâmetros é opcional para procedimentos e funções e a quantidade de argumentos na lista de parâmetros é indeterminada, podendo ser zero, um ou muitos argumentos. A lista de parâmetros contém valores de entrada para o módulo.

A linha da declaração do procedimento ou função é também chamada de **assinatura**. A assinatura do módulo procedimento ou função indica o seu nome, os tipos de dados e a ordem na lista de parâmetros. A chamada do procedimento ou da função deverá respeitar essa assinatura.

A partir de agora serão definidos e exemplificados os procedimentos com e sem parâmetros de entrada e, na sequência, as funções com e sem parâmetros de entrada. Por fim, daremos exemplos de algoritmos que combinam ambos os tipos de modularização.

Procedimento é um bloco de programa contendo início e fim e que será identificado por um nome, por meio do qual será referenciado em qualquer parte do programa principal ou do programa chamador da rotina (MANZANO; OLIVEIRA, 2016).

No algoritmo principal, a chamada de um procedimento é um comando que estabelece um desvio no módulo do algoritmo principal para executar um bloco de comandos externo. O algoritmo principal faz referência ao nome do procedimento como uma voz de comando, de forma imperativa, para executar o bloco de código do módulo invocado.

Usam-se procedimentos para escrever rotinas repetitivas que tornam o código principal extenso. Vamos então analisar alguns exemplos.

5.1 Procedimento sem parâmetros

Conforme apresentado na figura 111, para modularizar um bloco de código com uma funcionalidade específica, é necessário criar um **procedimento**. O nome do procedimento deve ser significativo ao propósito da funcionalidade.

No exemplo da figura a seguir, a variável opção é usada para capturar a opção do menu e verificar se o usuário deseja realizar outra operação. O menu poderia conter várias opções, e o código dentro da estrutura de repetição REPITA... ATÉ seria extenso.

Observação

Colocar o menu num módulo separado permite reutilizá-lo em outro algoritmo e define um ponto único de manutenção, caso as opções do menu sofram alterações, tais como a inclusão de novas operações ou a exclusão de alguma existente.

Exemplo 1

Escrever um algoritmo para apresentar um menu de opções de operações que o algoritmo poderá realizar e, para cada opção do menu, um procedimento para executar a operação matemática, conforme quadro a seguir.

Quadro 22

Operações	
[1]	Adição
[2]	Subtração
[3]	Multiplicação
[4]	Divisão

O algoritmo deverá primeiro perguntar ao usuário qual operação deseja efetuar e, depois, solicitar a entrada para as variáveis X e Y e exibir o resultado da operação.

Solução

```
1.  Algoritmo "Procedimento Menu e Operações"
2.  Var //variaveis globais
3.    opcao : caractere
4.    x, y : inteiro
5.
6.  procedimento escrevaMenu()
7.  inicio
8.    escreval("*********************************")
9.    escreval("*         Operações             *")
10.   escreval("*[+] Adição                      *")
11.   escreval("*[-] Subtração                   *")
12.   escreval("*[*] Multiplicação               *")
13.   escreval("*[/] Divisão                     *")
14.   escreval("*********************************")
15.   escreval("Digite a operação desejada: ")
16. fimprocedimento
17.
18. Inicio
19.   repita
20.     //entrada
21.     escrevaMenu()
22.     leia(opcao)
23.     escreva("X = ")
24.     leia(x)
25.     escreva("Y = ")
26.     leia(y)
27.
28.     //processamento e saida
29.     escolha (opcao)
30.     caso "+"
31.           escreval("Soma de ", x, " e ", y, " = ", x+y)
32.     caso "-"
33.           escreval("Diferença entre ", x, " e ", y, " = ", x-y)
34.     caso "*"
35.           escreval("Produto de ", x, " e ", y, " = ", x*y)
36.     caso "/"
37.           escreval("Divisão de ", x, " por ", y, " = ", x/y)
38.     outrocaso
39.           escreval("Operação inválida!!")
40.     fimescolha
41.
42.     escreva("Deseja efetuar outra operação? [S/N] ")
43.     leia(opcao)
44.   ate ((opcao="N") ou (opcao="n"))
45.   escreva("Fim do programa")
46. Fimalgoritmo
```

Figura 113 – Procedimento para escrever um menu

Neste exemplo, o procedimento tem uma finalidade bem específica, que é escrever o menu. O módulo principal está mais enxuto, sem muitas linhas de comandos e, com isso, mais compreensível.

A próxima figura descreve a lógica do procedimento **escrevaMenu()**. Pode-se dizer que esse procedimento contém o projeto de interface e interação com o usuário para este exemplo.

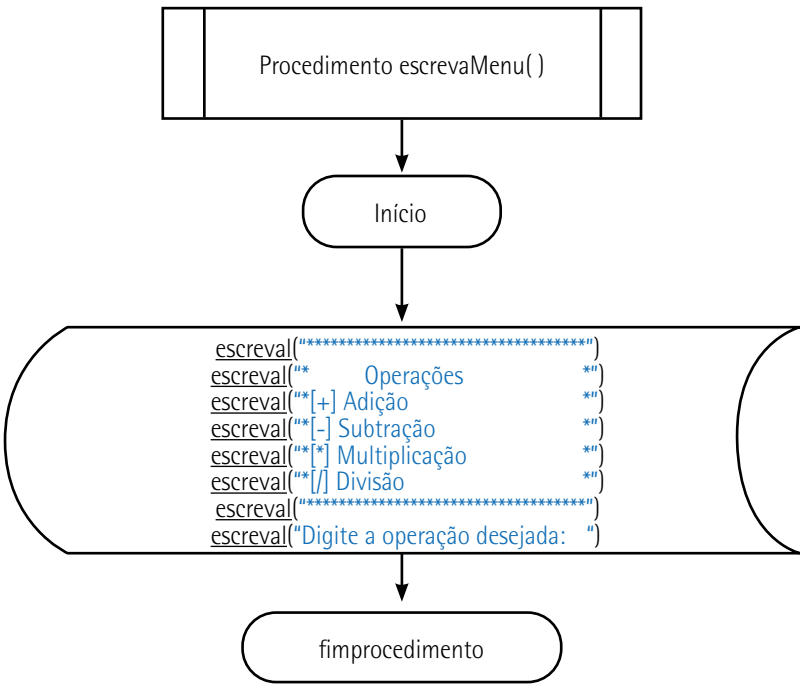


Figura 114 – Fluxograma que expressa a lógica do procedimento para escrever um menu

Por sua vez, a figura seguinte apresenta a lógica do algoritmo com uma chamada de procedimento. O procedimento **escrevaMenu()** é um subprocesso que executará vários comandos **escreva()** para desenhar o menu inicial e as mensagens de interação com o usuário.

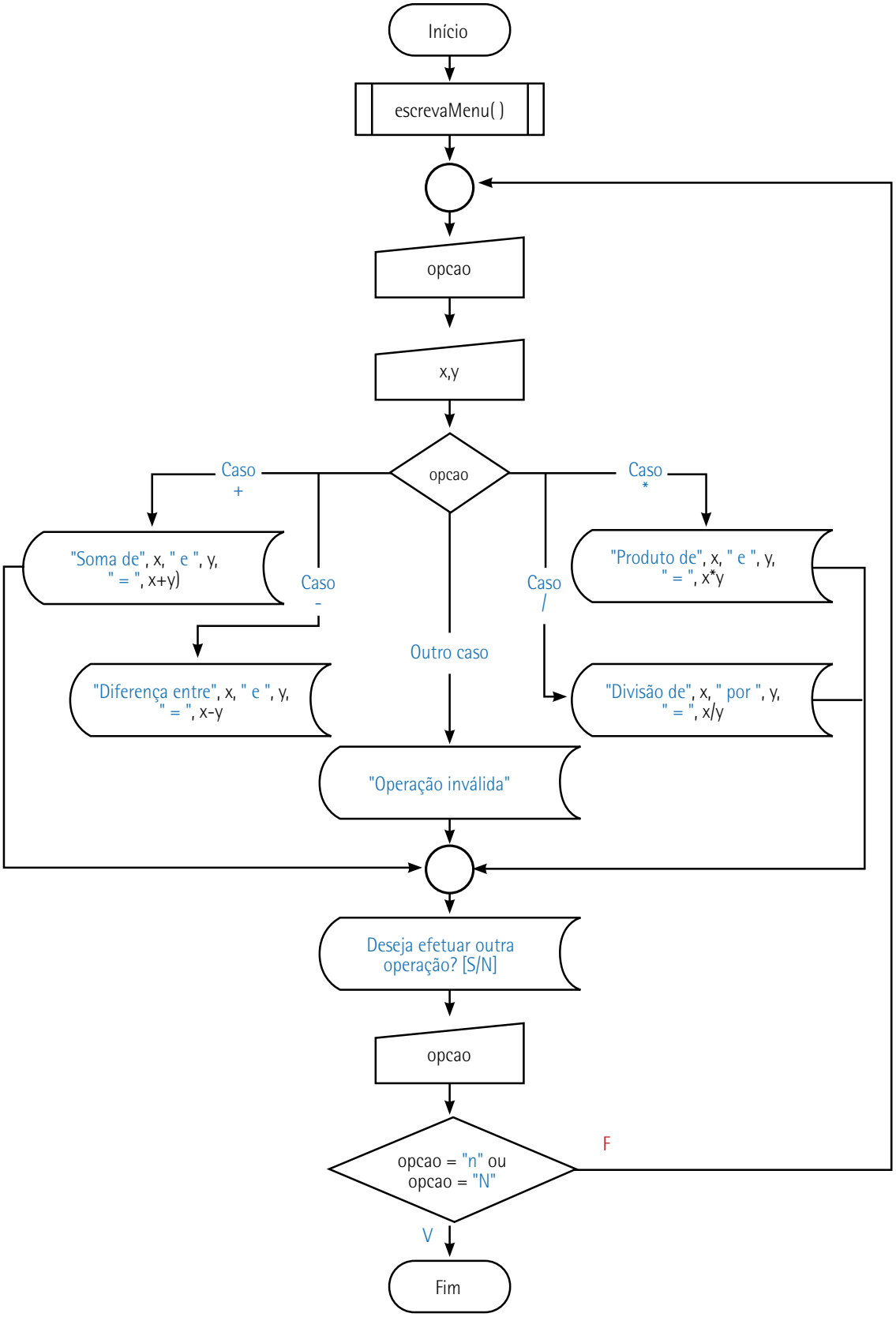


Figura 115 – Fluxograma de um programa com chamada de procedimento

Exemplo 2

Escrever um algoritmo para ler o nome, o salário e o tempo de serviço de um funcionário. Calcule e mostre a bonificação de acordo com a seguinte regra:

- menos de 10 anos: 5%;
- entre 10 e 20 anos: 10%;
- 20 anos ou mais: 15%.

Solução

```
1.  Algoritmo "Procedimento Imprimir Variaveis Globais"
2.  Var
3.  // variaveis globais
4.     nome : caractere
5.     salario, valor_bonus : real
6.     tempo_servico : inteiro
7.     bonus : inteiro
8.
9.  procedimento lerDados()
10. inicio
11.    escreva("Nome : ")
12.    leia(nome)
13.    escreva("Salario : ")
14.    leia(salario)
15.    escreva("Tempo de servico : ")
16.    leia(tempo_servico)
17. fimprocedimento
18.
19. procedimento imprimirDados()
20. inicio
21.    escreval("Nome : ", nome)
22.    escreval("Tempo de servico : ", tempo_servico)
23.    escreval("Salario : ", Salario)
24.
25.    se (tempo_servico<10) entao
26.        bonus<-10
27.    senao
28.      se (tempo_servico>=10) e (tempo_servico<20) entao
29.         bonus<-15
30.      senão
31.         bonus<-20
32.      fimse
33.    fimse
34.    valor_bonus<-salario*bonus/100
35.    escreval("Bonus de ", bonus, "%, ", valor_bonus)
36.    escreval("Salario com bonificação ", salario+valor_bonus)
37. fimprocedimento
38.
39. Inicio
40.    lerDados()
41.    imprimirDados()
42. Fimalgoritmo
```

Figura 116 – Procedimento para entrada e saída de variáveis globais

No exemplo da figura anterior, o módulo principal possui apenas dois comandos, o **lerDados()** e o **imprimirDados()**, nas linhas 40 e 41, respectivamente. Todos os comandos que fazem parte da leitura das variáveis globais estão dentro do módulo **lerDados()** e todos os comandos de saída estão no módulo **imprimirDados()**.

Procedimentos também podem ser usados para executar comandos de entrada e saída, bem como para capturar dados necessários para cálculos matemáticos, mas irrelevantes no contexto geral do algoritmo. Dessa forma, as alocações de memória para variáveis locais são desfeitas ao término da execução do procedimento, liberando a memória.

Exemplo 3

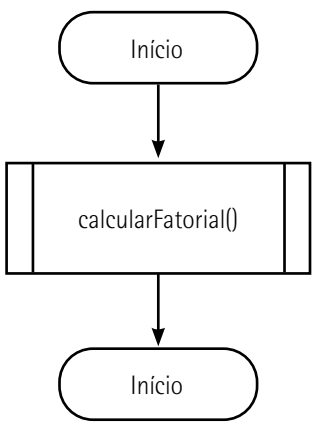
Crie um procedimento para receber um número inteiro e calcular o fatorial de um número.

Solução

```
Algoritmo "Calcular Fatorial"
Var
 //não há variáveis globais

procedimento calcularFatorial()
var //variáveis locais
  n, fatorial, cont, aux: inteiro
inicio
  //entrada
  escreva("Digite um número inteiro: ")
  leia(n)
  fatorial ← 1
  aux ← n
  para cont de 1 até aux faca
    fatorial <- fatorial * n
    n ← n-1
  fimpara
  escreva("Fatorial de ", aux, " é ", fatorial)
fimprocedimento
//módulo principal
Inicio
  calcularFatorial();
Fimalgoritmo
```

A)



B)

Figura 117 – Algoritmo com um procedimento para calcular o fatorial: A) pseudocódigo do procedimento para calcular o fatorial de um número, usando apenas variáveis locais; B) fluxograma do módulo principal

Este exemplo computa o fatorial de n, mas o valor de n não é obtido no módulo principal, mas dentro do procedimento. A lógica é apresentada na figura seguinte.

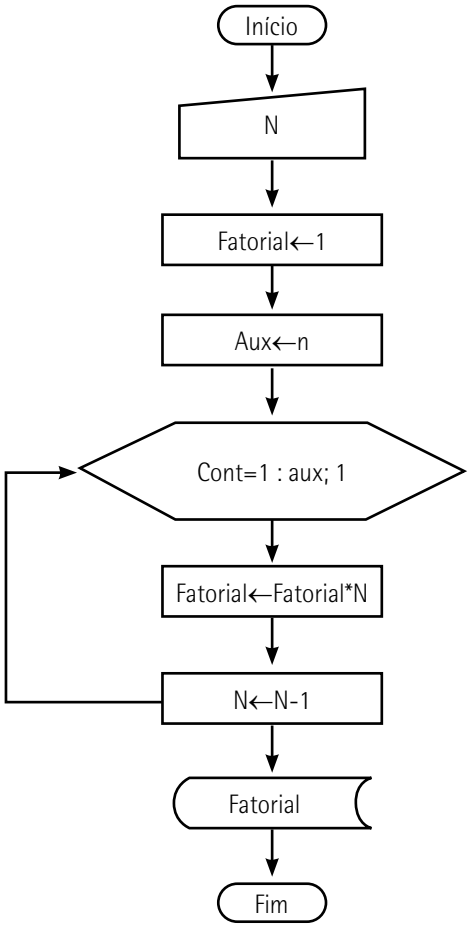


Figura 118 – Fluxograma do módulo fatorial

5.2 Procedimento com parâmetro

Um procedimento que requer parâmetros de entrada é um algoritmo que depende de um dado externo ao módulo que deverá ser processado dentro do módulo. Esse dado externo é o parâmetro de entrada, cujo valor é passado na chamada do algoritmo. O exemplo a seguir é o algoritmo de um procedimento com parâmetros de entrada.

Exemplo

Elabore um procedimento para escrever os dados nome e idade.

Solução

```
1.  Algoritmo "Procedimento com parâmetro"
2.  Var
3.  // variaveis globais
4.          nome  : caractere
5.          idade : inteiro
6.
7.  procedimento imprimirDados(texto: caractere; numero: inteiro)
8.  inicio
9.    escreval("Nome  : ", texto)
10.   escreval("Idade : ", numero)
11. fimprocedimento
12.
13. //modulo principal
14. Início
15.         nome←"Maria"
16.         idade←20
17.         imprimirDados(nome, idade)
18. Fimalgoritmo
```

Figura 119 – Algoritmo do procedimento com lista de parâmetros

A assinatura do procedimento é a linha 8 contendo o nome e a lista de parâmetros. Na linha 17, é feita a chamada do procedimento e as variáveis globais nome e idade são passadas no parâmetro; o valor guardado das variáveis nome e idade será então impresso.

A assinatura do procedimento indica o nome dele, bem como sua lista de parâmetros. A chamada do procedimento na linha 18

```
imprimirDados(nome,idade)
```

Figura 120

indica que a primeira variável deve ser do tipo caractere e a segunda do tipo inteiro, não sendo permitido inverter essa ordem, como podemos ver a seguir:

```
imprimirDados(idade,nome) //!Erro
```

Figura 121

5.3 Chamada de procedimento com ou sem parâmetro

A chamada de procedimento é a forma como os procedimentos são invocados dentro do módulo principal. O nome do procedimento sugere algo que deve ser executado, um comando, e a leitura tende a entoar de modo imperativo, como uma "voz de comando".

```
imprimirDados(nome, idade)          //é um comando, uma ordem
```

Figura 122

Para o exemplo da figura 117, a assinatura do procedimento é dada a seguir. Note: o primeiro argumento é um dado do tipo caractere e o segundo deve ser um inteiro.

```
procedimento imprimirDados(texto: caractere; numero: inteiro)
```

Figura 123

Para essa assinatura, as seguintes chamadas do procedimento são permitidas:

Quadro 23 – Chamadas de procedimento

imprimirDados(nome, idade)	Nome é uma variável do tipo caractere e idade, do tipo inteiro
imprimirDados("José", 73)	Em vez de passar a variável como argumento do procedimento, é informado um valor. Nessa chamada de procedimento, **José** é um dado do tipo caractere e **73** é um dado do tipo inteiro
imprimirDados(nome, 50)	O primeiro argumento é uma variável e o segundo é um número
imprimirDados("José", idade)	O primeiro argumento é um dado do tipo caractere e o segundo é uma variável do tipo caractere

Exemplo

A figura a seguir é um pseudocódigo para a modularização **procedimento**.

A)

```
Algoritmo "PROC_SEM PARAMETRO"
Var
  //variaveis globais
procedimento facaAlgo()
var
  //variaveis locais
inicio
        <comando 1>
        <comando 2>
        :
        <comando n>
fimprocedimento
//modulo principal
Inicio
  ...
  //chamada de procedimento
  facaAlgo()
  ...
Fimalgoritmo
```

B)

```
Algoritmo "PROC_COM PARAMETRO"
Var
  //variaveis globais
  x : real
procedimento facaAlgo(y:real)
var
  //variaveis locais
inicio
        <comando 1>
        <comando 2>
        :
        <comando n>
fimprocedimento
//modulo principal
Inicio
  ...
  // chamada de procedimento
  facaAlgo(x)
  ...
Fimalgoritmo
```

Figura 124 – A) pseudocódigo com chamada de procedimento sem parâmetro; B) pseudocódigo com chamada de procedimento com parâmetro

Os parâmetros do procedimento são dados de entrada para o algoritmo. Normalmente, quando uma variável local é passada como argumento do procedimento, tal como a variável x do algoritmo da figura anterior (B), o compilador faz uma cópia dela para realizar as operações do algoritmo. Ainda que as variáveis passadas como argumentos do procedimento sejam modificadas no escopo do procedimento, no bloco principal a variável local permanece inalterada.

Saiba mais

Amplie seus conhecimentos sobre procedimentos consultando o capítulo:

MANZANO, J. A. N. G.; OLIVEIRA, J. F. de. Aplicação prática do uso de sub-rotinas – procedimentos. *In:* MANZANO, J. A. N. G.; OLIVEIRA, J. F. de. *Algoritmos*: lógica para desenvolvimento de programação de computadores. 28. ed. São Paulo: Érica, 2016.

6 FUNÇÕES

Função é um bloco de programa identificado por um nome, por meio do qual será referenciada em qualquer parte do programa principal. Ela tem diversas similaridades com um procedimento.

Lembrete

> Conforme Manzano e Oliveira (2016), procedimento é um bloco de programa contendo início e fim e que será identificado por um nome, por meio do qual será referenciado em qualquer parte do programa principal ou do programa chamador da rotina.

Uma sub-rotina de função é na verdade muito parecida com uma sub-rotina de procedimento, inclusive as sintaxes são idênticas (MANZANO; OLIVEIRA, 2016).

Uma função é um módulo que tem por finalidade executar um bloco de códigos separado da estrutura principal e retornar um valor. A chamada de uma função pode ser interpretada como uma pergunta cuja resposta será o valor de retorno.

A figura 111 apresenta a sintaxe para especificação de uma função. A assinatura da função contém a palavra reservada função, o nome da função, a lista de parâmetros e o tipo de retorno da função, conforme ilustrado a seguir:

```
função <nome>([parâmetros]): <tipo>
```

Figura 125

O corpo da função possui variáveis locais e o bloco de comandos que deverá executar. A palavra reservada **retorne** indica o ponto de retorno, ou seja, o ponto em que a função será interrompida e devolverá para a variável ou para o comando que a invocou o valor de retorno. Como boa prática, sugere-se que cada função tenha apenas um ponto de retorno.

Cada algoritmo pode implementar várias funções, sem limitação. Assim como procedimentos, funções podem conter zero, um ou muitos parâmetros de entrada.

6.1 Função sem parâmetro

Uma função sem passagem de parâmetro é similar ao procedimento sem parâmetro, todavia, apesar de não serem enviados parâmetros de entrada para a função, ela deve retornar um valor. A estrutura da função sem parâmetro é:

```
função nomeFuncao() : tipo_de_retorno
        inicio
          //algoritmo
retorne valor
fimfuncao
```

Figura 126

O tipo de retorno da função poderá ser um dos tipos primitivos ou algum tipo abstrato de dados. Se o tipo de retorno da função for inteiro, o valor de retorno deverá ser inteiro também, mas, se for real, o valor de retorno também deverá ser real. Os parâmetros são os dados de entrada e o valor de retorno é a saída da função.

O exemplo da figura a seguir é um algoritmo modularizado em funções para codificar as operações de uma calculadora.

Exemplo

Nesse exemplo, o programa deve conter quatro módulos ou subalgoritmos, e cada módulo deve realizar uma das quatro operações matemáticas: soma, subtração, multiplicação ou divisão.

Cada módulo deve solicitar dois valores para efetuar o cálculo e o algoritmo principal deverá chamar cada uma das funções de acordo com a escolha do usuário, retornar o valor da operação para o algoritmo principal e escrever o resultado.

```
1.  Algoritmo "Calculadora"
2.  Var
3.              //declaracao das variaveis globais
4.              a, b, resultado: real
5.     opcao: inteiro
6.
7.  procedimento escrevaMenu()
8.          escreva("CALCULADORA")
9.          escreva("[1] Soma ")
10.         escreva("[2] Subtração ")
11.         escreva("[3] Multiplicação ")
12.         escreva("[4] Divisão ")
13.         escreva("[5] Número do PI ")
14.         escreva("Escolha uma entre as operações abaixo: ")
15. fimprocedimento
16.
17. função numeroPI() : real
18. inicio
19.    retorne 3.14159265359
20. fimfuncao
21.
22. //módulo principal
23. Inicio
24.         escrevaMenu()
25.         leia(opcao)
26.         escolha(opcao)
27.       caso 5
28.            escreva("Número do PI = ", numeroPI())
29.       outrocaso
30.          escreva("Operação não implementada neste exemplo")
31.     fimescolha
32. Fimalgoritmo
```

Figura 127 – Função sem passagem de parâmetros

A função sem parâmetro deste exemplo retorna o número do Pi dado por π = 3,141592653589793. Lembrar o número com todas as casas decimais não é fácil e a digitação é sujeita a erros. Quando o projetista do algoritmo deseja escrever ou usar o número do Pi, ele simplesmente pergunta ao programa: Qual é o número do Pi?, fazendo a chamada da função conforme apresentado na linha 28.

```
escreva("Número do PI = ", numeroPI())
```

Figura 128

Nesse comando, o valor de retorno da função é um número do tipo real, e a chamada foi feita dentro do comando **escreva()**. O comando de saída da linha 28 escreverá uma mensagem composta pela concatenação do texto com o valor retornado pela função **numeroPi()**.

Além da função com parâmetro, neste exemplo a quantidade de linhas de código foi reduzida com a extração do bloco para o procedimento **escrevaMenu()**, também sem parâmetros, dando a este bloco um significado coerente com seu propósito.

6.2 Função com parâmetro

A lista de parâmetros é formada por dados de entrada para o algoritmo da função necessários para o processamento. A passagem de parâmetro em uma função define um protocolo para a chamada, também denominado assinatura da função. Os parâmetros, na ordem em que são declarados, devem ser informados quando a função for invocada e os valores enviados puderem ser aproveitados em outras partes do algoritmo, como no exemplo a seguir:

```
1.  função soma(a, b: inteiro) : inteiro
2.  inicio
3.          retorne a + b
4.
5.  fimfuncao
6.  função maior(a, b, c: inteiro) : inteiro
7.  var
8.          m : inteiro
9.  inicio
10.         se ((a>b) e (a>c)) então
11.                 m <-a
12.         senão   se ((b>a) e (b>c)) então
13.                 m <- b
14.         senão
15.                 m <-c
16.         fimse
17.         fimse
18.         retorne m
19. fimfuncao
```

Figura 129 – Função com passagem de parâmetros

A função soma possui dois argumentos do tipo inteiro, identificados pelas letras a e b. Trata-se variáveis de entrada para a função que, ao ser invocada, retornará a soma dos dois valores. No bloco principal, a chamada da função soma poderia ser feita para diferentes valores de entrada, conforme mostrado a seguir:

```
1.  Algoritmo TesteFunçãoSoma
2.  Var
3.          x, y, z, s : inteiro
4.          r, t, w : real
5.  Inicio
6.          //entrada
7.          leia(x,y,z)
8.          leia(r, t)
9.
10.         //processamento
11.         s ← soma(x,y)
12.         //w ← soma(r, t)        !Erro
13.
14.         //saída
15.         escreva("soma dos três números é ", soma(s,z))
16. Fimalgoritmo
```

Figura 130 – Algoritmo para chamada de função com parâmetro de entrada

A chamada da função soma na linha 11 é feita em conjunto com um comando de atribuição. Assim, o valor de retorno da função soma será atribuído para a variável **s**. Já na linha 15, a variável soma é invocada dentro da função **escreva**. Será escrito o resultado da soma das variáveis **s** que, nesse ponto do algoritmo, está armazenando a soma de x e y mais o valor da variável z. Na linha 11, o valor da variável **s** é utilizado para armazenar uma soma, mas não é alterado.

A função soma recebe apenas dois parâmetros e ambos devem ser do tipo inteiro. A chamada da função na linha 12 produz um erro porque foram informadas no argumento da função **soma()** variáveis do tipo real em que são esperados valores ou variáveis do tipo inteiro, decorrente dos parâmetros da função **soma()**, os quais exigem que os números sejam do tipo inteiro, e as variáveis **r** e **t** passadas como argumentos são do tipo real.

A assinatura da função definida na linha 1 da figura 129, função **soma(a, b: inteiro) : inteiro** especifica a sintaxe do comando na chamada da função. A função soma deve receber dois valores ou variáveis inteiros.

Uma função é um algoritmo com propósito definido que pode ou não ter parâmetros de entrada, e pode possuir zero, uma ou diversas variáveis locais, a(s) qual(is) será(ão) reconhecida(s) apenas no escopo da função, a fim de guardar dados relevantes para o processamento do algoritmo. Esse é o caso

da variável local identificada como **m**. O objetivo dela é armazenar o maior valor dentre os três passados como parâmetros de entrada para a função maior. A variável **m** não é importante nem visível fora da função, pois serve apenas para guardar o maior valor durante o processamento.

```
1.  Algoritmo TesteFunçãoMaior
2.  Var
3.          x, y, z, m : inteiro
4.  Inicio
5.          //entrada
6.          leia(x,y,z)
7.
8.          //processamento
9.          m ← maior(x,y,z)
10.
11.         //saida
12.         escreva("O maior dos três números é ", m)
13. Fimalgoritmo
```

Figura 131 – Algoritmo para testar a chamada da função que retorna o maior de três números

A função **maior()** pede três argumentos de entrada. O algoritmo não será executado se na chamada da função forem passadas variáveis de um tipo diferente de inteiro ou uma quantidade diferente de parâmetros. Tais regras valem também para procedimentos com lista de parâmetros.

Exemplo

Escreva um algoritmo para apresentar o menu do quadro a seguir e efetuar as operações aritméticas conforme escolha do usuário.

Quadro 24

Calculadora	
[1]	Soma
[2]	Subtração
[3]	Multiplicação
[4]	Divisão
[5]	Número do Pi

Solução

```
1.  Algoritmo "Calculadora"
2.  Var
3.    //declaracao das variaveis globais
4.    a, b, resultado, x, y: real
5.    opcao: inteiro
6.
7.  funcao soma(a, b:real) :real
8.  inicio
9.     resultado <- a + b
10.    retorne resultado
11. fimfuncao
12.
13. funcao subtracao(a, b:real): real
14. inicio
15.          resultado <- a-b
16.     retorne resultado
17. fimfuncao
18.
19. funcao multiplicacao(a, b:real) :real
20. inicio
21.    resultado <- a * b
22.    retorne resultado
23. fimfuncao
24.
25. funcao divisao(a, b: real) : real
26. // retorna -1 sempre que o dividendo for 0.
27. inicio
28.          escreva("OPERAÇÃO DIVISÃO")
29.    leiaDados()
30.    se ((a=0) ou (b=0)) então
31.       resultado = -1
32.    senao
33.      se (a<b) entao
34.         resultado<- b/a
35.      senão
36.         resultado<- a/b
37.      fimse
38.    fimse
39.    retorne resultado
40. fimfuncao
41.
42. procedimento escrevaMenu()
43.          escreva("CALCULADORA")
44.          escreva("[1] Soma ")
45.          escreva("[2] Subtração ")
46.          escreva("[3] Multiplicação ")
```

```
47.             escreva("[4] Divisão ")
48.             escreva("[5] Número do PI ")
49.             escreva("Escolha uma entre as operações abaixo: ")
50.     fimprocedimento
51.
52.     //modulo principal
53.     Inicio
54.       //entrada
55.       escreva(menu)
56.       leia(opcao)
57.
58.       escreva(Informe os valores para executar a operação. ")
59.       escreva("Informe o valor de A: ")
60.       leia(x)
61.       escreva("Informe o valor de B: ")
62.       leia(y)
63.
64.       escolha(opcao)
65.         caso 1
66.           escreva("O resultado da operação é: ",soma(x,y))
67.         caso 2
68.           escreva("O resultado da operação é: ",subtracao(x,y))
69.         caso 3
70.           escreva("O resultado da operação é: ",multiplicacao(x,y))
71.         caso (4)
72.           escreva("O resultado da operação é: ",divisao(x,y))
73.       fimescolha
74.     Fimalgoritmo
```

Figura 132 – Pseudocódigo com mais de uma função com passagem de parâmetros

No exemplo da figura precedente, as funções **soma()**, **subtracao()**, **multiplicacao()** e **divisao()** estão sendo invocadas dentro do comando **escreva()**, isso significa que o valor de retorno de cada função não será armazenado, mas apenas exibido na tela. Essa é uma das formas de fazer chamada de função.

6.3 Chamada de funções com e sem parâmetro

A chamada de uma função deve ser realizada dentro de um comando de atribuição e, nesse caso, o valor de retorno da função será atribuído conforme o exemplo a seguir:

```
x <- soma(12.5, 1.5)
```

Figura 133

A variável x deve ter sido declarada anteriormente como do tipo real, uma vez que a função soma retorna um valor do tipo real.

Outra forma de invocar a função é usar uma expressão aritmética ou relacional utilizada como condição das estruturas de decisão ou repetição da seguinte forma:

```
se (soma(x,y)>70.9) então
                //faça alguma coisa
fimse
```

Figura 134

Nesse exemplo, se o valor de retorno da função soma for maior ou igual 70.9, então a condição resultará verdadeira e o bloco será executado. Com a mesma lógica, a função poderá ser invocada num laço de repetição da forma

```
enquanto (soma(a, b)<=12.5) faça
    leia(a)
    leia(b)
fimenquanto
```

Figura 135

O algoritmo ficará lendo ininterruptamente as variáveis x e y enquanto a soma delas for menor ou igual a 12.5. Quando for maior, o laço será interrompido. A figura seguinte mostra a chamada de uma função com e sem passagem de parâmetros.

Existem situações na quais as funções são restritas a outras funções, não fazendo sentido invocá-las no bloco de comando principal. Repare na figura a seguir:

```
1.  Algoritmo "Funções com e sem parâmetros"
2.  Var
3.    //declaracao das variaveis globais
4.    a, b, r: inteiro
5.
6.  //funcao para calcular o delta com parametros de entrada
7.  funcao delta(a, b, c: inteiro) : inteiro
8.  inicio
9.    retorne b*b-4*a*c
10. fimfuncao
11.
12. //procedimento para escrever as raizes
13. Procedimento escrevaRaizes(a, b, c: inteiro)
14. var
15.         x1, x2 : real
16. inicio
17.         se (delta(a,b,c)>=0) então
18.           x1 <- (-1*b+raizq(delta))/(2*a)
19.           x2 <- (-1*b-raizq(delta))/(2*a)
20.           escreva("raiz 1 é ", x1)
21.           escreva("raiz 2 é ", x2)
22.         senao
23.           escreva("Não existem raízes reais")
24.         fimse
25. fimfuncao
26.
27. Inicio
28.   escreva(Informe os valores para executar a operação. ")
29.   escreva("Informe o valor de A: ")
30.   leia(a)
31.   escreva("Informe o valor de B: ")
32.   leia(b)
33.   escreva("Informe o valor de C: ")
34.   leia(c)
35.   escrevaRaizes(a, b, c)
36. Fimalgoritmo
```

Figura 136 – Função delta invocada dentro do procedimento escrevaRaizes()

Repare: a função **delta()** é invocada dentro do procedimento **escrevaRaizes()**.

Saiba mais

Leia mais sobre funções no capítulo:

ASCENCIO, A. F. G.; CAMPOS, E. A. V. Funções de tratamento de caracteres. *In:* ASCENCIO, A. F. G.; CAMPOS, E. A. V. *Fundamentos da programação de computadores*. São Paulo: Pearson, 2012.

6.4 Escopo da variável

O escopo de uma variável é um conceito que define a visibilidade da variável, ou seja, onde ela será reconhecida dentro do algoritmo. O escopo da variável poderá ser **global** ou **local**.

Variável global é declarada no início do algoritmo e pode ser acessada tanto no módulo principal quanto em qualquer outro módulo especificado como procedimento ou função dentro do algoritmo. Variáveis globais devem ser declaradas uma única vez e não pode haver duas ou mais variáveis com o mesmo nome, e elas são conhecidas dentro do módulo principal e dos módulos de procedimentos e funções. Uma variável global aloca um endereço de memória que é identificado pelo nome da variável e é reconhecido por todos os módulos do algoritmo.

Já a variável local é declarada dentro do módulo de procedimento ou função e é visível e reconhecida apenas dentro do módulo no qual foi declarada. Variáveis locais existem apenas dentro do módulo e é possível declarar variáveis locais com nomes iguais dentro de módulos diferentes, pois cada uma delas alocará um endereço de memória específico.

A declaração de variáveis globais não constitui uma boa prática porque restringe a possibilidade de reúso dos módulos, por isso deve ser evitada sempre que possível.

Saiba mais

Leia mais sobre procedimentos e funções na obra:

FORBELLONE, A.; EBERSPACHER, H. *Lógica de programação*: a construção de algoritmos e estruturas de dados. 3. ed. São Paulo: Makron Books, 2005.

6.5 Exemplos de algoritmos com procedimentos e funções

Algoritmos podem ser longos e complexos e a modularização ajuda a organizá-los de modo a facilitar o entendimento, a dar manutenção e a evitar a repetição de blocos de código. A partir de agora abordaremos exemplos de problemas que aplicam o uso de PROCEDIMENTO e FUNÇÃO no processo de solução.

Exemplo 1

Escreva um algoritmo que permita efetuar cálculos das disciplinas de Matemática e Física. Para Matemática, calcular a área do quadrado e do retângulo, equação de primeiro e de segundo grau; para Física, calcular o volume do paralelepípedo e de um cilindro e a velocidade média. O algoritmo deve ser organizado com blocos de código criando um bloco para Matemática e outro para Física.

Solução

```
1.  Algoritmo "CALCULADORA Matemática/Física - Procedimento"
2.  Var
3.    //Variaveis para Escolha-Caso
4.    opcao, opmat, opfis, opfisvol : Inteiro
5.    //Variaveis para calcular área do quadrado
6.    quadrado, aresta : real
7.    //Variaveis para calcular área do retângulo
8.    retangulo, base, altura : real
9.    //Variaveis para calcular Equação de 1ºGrau
10.   eq1, a1, b1, i1 : real
11.   //Variaveis para Calcular Equação de 2ºGrau
12.   a, b, c, delta, x1, x2 : real
13.   //Variaveis para calcular volume do paralelepípedo
14.   C1, L1, A2 : real
15.   //Variaveis para calcular volume do cilindro
16.   raio, ALTURA1, volume_cilindro : real
17.   //Variaveis para calcular a velocidade média
18.   vmedia, dist, tmp : real
19.
20.  //Bloco Calculos de Matematica
21.  procedimento calcularMatematica()
22.  escreva("-----------------------------------------------------")
23.  escreva("|                                                   |")
24.  escreva("|       Calculadora - Matemática                    |")
25.  escreva("|                                                   |")
26.  escreva("-----------------------------------------------------")
27.  escreva("|                                                   |")
28.  escreva("| Qual tipo de cálculo deseja realizar?             |")
29.  escreva("| [1] Área do quadrado                              |")
30.  escreva("| [2] Área do retângulo                             |")
31.  escreva("| [3] Equação de 1º Grau                            |")
32.  escreva("| [4] Equação de 2º Grau                            |")
33.  escreva("|                                                   |")
34.  escreva("-----------------------------------------------------")
35.  escreva("")
36.  escreva(" Opção escolhida: ")
37.  leia(opmat)
38.  escolha(opmat)
39.    caso 1
40.      escreva("Digite o valor da aresta: ")
41.      leia(aresta)
42.      quadrado <- aresta * aresta
43.      escreva("A área do quadrado é: ", quadrado)
44.    caso 2
```

```
45.     escreva("Digite o valor da base: ")
46.     leia(base)
47.     escreva("Digite o valor da altura: ")
48.     leia(altura)
49.     retangulo <- base * altura
50.     escreva("A área do retângulo é: ", retangulo)
51.   caso 3
52.     escreva("Para calcular a equação de 1ºGrau, considere: y=a*i+b")
53.     escreva("Digite o valor de A..: ")
54.     leia(a1)
55.     escreva("Digite o valor de B..: ")
56.     leia(b1)
57.     escreva("Digite o valor de I..: ")
58.     leia(i1)
59.     eq1 <- a1*i1+b1
60.     escreva("O valor de y é:", eq1)
61.   caso 4
62.     escreva("Programa para calcular f(x)=ax2+bx+c")
63.     escreva("a: ")
64.     leia(a)
65.     escreva("b: ")
66.     leia(b)
67.     escreva("c: ")
68.     leia(c)
69.     delta <- b*b-4*a*c
70.     se (delta>0) então
71.       x1 <- (-1*b+raiz(delta))/(2*a)
72.       x2 <- (-1*b-raiz(delta))/(2*a)
73.       escreva("A raiz 1 é ",x1,"e a raiz 2 é ",x2)
74.     senão
75.       escreva("Não existem raízes reais")
76.     fimse
77.   outrocaso
78.     escreva("------------------------------------------------ ")
79.     escreva("|                                              |")
80.     escreva("|    A opção escolhida não existe!             |")
81.     escreva("|                                              |")
82.     escreva("------------------------------------------------ ")
83. fimescolha
84. Fimprocedimento
85.
86. //Bloco Calculos de Fisica
87. procedimento calcularFisica()
88. escreva("-------------------------------------------------")
89. escreva("|                                               |")
90. escreva("|          Calculadora - Física                 |")
91. escreva("|                                               |")
92. escreva("-------------------------------------------------")
93. escreva("|                                               |")
94. escreva("| Qual tipo de cálculo deseja realizar?         |")
95. escreva("| [1] Volume                                    |")
96. escreva("| [2] Velocidade média                          |")
97. escreva("|                                               |")
98. escreva("-------------------------------------------------")
99. escreva("")
```

```
escreva(" Opção escolhida: ")
leia(opfis)
escolha(opfis)
 caso 1
  escreva("-----------------------------------------------------")
  escreva("|                                                   |")
  escreva("|    Calculadora - Física - VOLUME                  |")
  escreva("|                                                   |")
  escreva("-----------------------------------------------------")
  escreva("|                                                   |")
  escreva("| Deseja calcular VOLUME de qual objeto?            |")
  escreva("| [1] Volume de um Paralelepípedo                   |")
  escreva("| [2] Volume de um Cilindro                         |")
  escreva("|                                                   |")
  escreva("-----------------------------------------------------")
  escreva("")
  escreva(" Opção escolhida: ")
  leia(opfisvol)
  escolha(opfisvol)
   caso 1
    escreva("Comprimento do paralelepípedo (C): ")
    leia(C1)
    escreva("Largura do paralelepípedo.... (L): ")
    leia(L1)
    escreva("Altura do paralelepípedo..... (A): ")
    leia(A2)
    escreva("O Volume do Paralelepípedo é: ", C1*L1*A2)
   caso 2
    escreva("Informe o valor do RAIO: ")
    leia(raio)
    escreva("Informe o valor da ALTURA: ")
    leia(ALTURA1)
    volcil <- 3.14 * (raio * raio) * ALTURA1
    escreva("O Volume do Paralelepípedo é: ", volcil)
   outrocaso
    escreva("-----------------------------------------------------")
    escreva("|                                                   |")
    escreva("|  A opção escolhida não existe                     |")
    escreva("|                                                   |")
    escreva("-----------------------------------------------------")
  fimescolha

 caso 2
  escreva("Informe a distância percorrida (em KM): ")
  leia(dist)
  escreva("Informe o tempo gasto (em Horas): ")
  leia(tmp)
  vmedia <- dist/tmp
  escreva("A velocidade média é de: ", vmedia," Km/h")
 outrocaso
  escreva("-----------------------------------------------------")
  escreva("|                                                   |")
  escreva("|    A opção escolhida não existe!                  |")
  escreva("|                                                   |")
  escreva("-----------------------------------------------------")
```

```
155.  fimescolha
156.  Fimprocedimento
157.
158.  procedimento escrevaMenu()
159.    escreva("-----------------------------------------------------------")
160.    escreva("|                                                         |")
161.    escreva("|              Calculadora                                |")
162.    escreva("|                                                         |")
163.    escreva("-----------------------------------------------------------")
164.    escreva("|                                                         |")
165.    escreva("| Qual tipo de cálculo deseja realizar?                   |")
166.    escreva("| [1] Para Matemática                                     |")
167.    escreva("| [2] Para Física                                         |")
168.    escreva("|                                                         |")
169.    escreva("-----------------------------------------------------------")
170.    escreva("")
171.  fimprocedimento
172.
173.  //Bloco principal
174.  Inicio
175.    escrevaMenu()
176.    escreva(" Opção escolhida: ")
177.    leia(opcao)
178.    escreva("")
179.    escolha(opcao)
180.      caso 1
181.        calcularMatematica()
182.      caso 2
183.        calcularFisica()
184.      outrocaso
185.        escreva("------------------------------------------------- ")
186.        escreva("|                                                |")
187.        escreva("|    A opção escolhida não existe!               |")
188.        escreva("|                                                |")
189.        escreva("------------------------------------------------- ")
190.    fimescolha
191.  Fimalgoritmo
```

Figura 137 – Algoritmo calculadora matemática/física com procedimento

Nesse exemplo, foram reunidos alguns dos exemplos já estudados, organizados em dois grandes blocos de procedimentos, um com o algoritmo para as operações de Matemática e outro com as operações de Física, além do bloco de código principal, que fará as chamadas dos procedimentos.

O algoritmo é organizado utilizando **procedimentos** que serão executados quando necessário. Como os **procedimentos** não possuem valor de retorno, não seria possível criar um agrupamento que pudesse ser utilizado em vários blocos de código.

Outro ponto importante sobre os módulos é que eles devem ser declarados antes do bloco principal. Quando os procedimentos forem invocados no bloco principal, já devem ter sido interpretados pelo compilador para que o nome do procedimento seja conhecido do programa.

Nesse algoritmo, foram declaradas variáveis entre as linhas 3 e 18, o **procedimento** calcularMatematica() é o comando que executa o bloco de códigos referente às operações matemáticas, o **procedimento** calcularFísica() é o comando que executa o bloco de códigos referente às operações de física e no final o bloco principal, que executa o início do algoritmo e de acordo com a opção escolhida pelo usuário, dá voz de comando para iniciar um **procedimento**.

Observação

Em algumas linguagens de programação, tais como Cobol e Python, a indentação faz parte da sintaxe do comando, impedindo o programa de ser compilado quando os deslocamentos inexistem. No caso da linguagem Cobol, a quantidade de espaços é considerada pelo compilador.

Exemplo 2

Elabore um algoritmo para calcular a área do quadrado e do retângulo, equação de primeiro grau, volume do paralelepípedo e do cilindro e obter a velocidade média. Considere as boas práticas na elaboração desse algoritmo.

Solução

```
1.  Algoritmo "Calculadora Matemática/Física"
2.  Var
3.     //Variaveis para Escolha-Caso
4.     opcao, opmat, opfis, opfisvol : Inteiro
5.     //Variaveis de entrada para efetuar calculos
6.     aresta, base, altura, a1, b1, c1 : real
7.
8.  procedimento menuPrincipal()
9.  Inicio
10.   escreva("-------------------------------------------------")
11.   escreva("|                                               |")
12.   escreva("|              Calculadora                      |")
13.   escreva("|                                               |")
14.   escreva("-------------------------------------------------")
15.   escreva("|                                               |")
16.   escreva("| Qual tipo de cálculo deseja realizar?         |")
17.   escreva("| [1] Para Matemática                           |")
18.   escreva("| [2] Para Física                               |")
19.   escreva("|                                               |")
20.   escreva("-------------------------------------------------")
21.   escreva("")
22.   escreva(" Opção escolhida: ")
23.   fimprocedimento
24.
25.   procedimento menuMatematica()
26.   Inicio
27.   escreva("-------------------------------------------------")
28.   escreva("|                                               |")
29.   escreva("|       Calculadora - Matemática                |")
30.   escreva("|                                               |")
31.   escreva("-------------------------------------------------")
```

```
32.  escreva("|                                                    |")
33.  escreva("| Qual tipo de cálculo deseja realizar?              |")
34.  escreva("| [1] Área do quadrado                               |")
35.  escreva("| [2] Área do retângulo                              |")
36.  escreva("| [3] Equação de 1º Grau                             |")
37.  escreva("| [4] Equação de 2º Grau                             |")
38.  escreva("|                                                    |")
39.  escreva("----------------------------------------------------")
40.  escreva("")
41.  escreva(" Opção escolhida: ")
42. fimprocedimento
43.
44. procedimento menuFisica()
45.  Inicio
46.  escreva("----------------------------------------------------")
47.  escreva("|                                                    |")
48.  escreva("|          Calculadora - Física                      |")
49.  escreva("|                                                    |")
50.  escreva("----------------------------------------------------")
51.  escreva("|                                                    |")
52.  escreva("| Qual tipo de cálculo deseja realizar?              |")
53.  escreva("| [1] Volume                                         |")
54.  escreva("| [2] Velocidade média                               |")
55.  escreva("|                                                    |")
56.  escreva("----------------------------------------------------")
57.  escreva("")
58.  escreva(" Opção escolhida: ")
59.  fimprocedimento
60.
61.  procedimento menuOpcaoVolume()
62.  Inicio
63.   escreva("----------------------------------------------- ")
64.   escreva("|                                                |")
65.   escreva("|     Calculadora - Física - VOLUME              |")
66.   escreva("|                                                |")
67.   escreva("----------------------------------------------- ")
68.   escreva("|                                                |")
69.   escreva("| Deseja calcular VOLUME de qual objeto?         |")
70.   escreva("| [1] Volume de um Paralelepípedo                |")
71.   escreva("| [2] Volume de um Cilindro                      |")
72.   escreva("|                                                |")
73.   escreva("----------------------------------------------- ")
74.   escreva("")
75.   escreva(" Opção escolhida: ")
76.  fimprocedimento
77.
78.  procedimento msgErrpadrao()
79.   escreva("----------------------------------------------- ")
80.   escreva("|                                                |")
81.   escreva("|    A opção escolhida não existe!               |")
82.   escreva("|                                                |")
83.   escreva("----------------------------------------------- ")
84.  fimprocedimento
85.
86.  funcao areaQuadrado (lado:real) :real
87.   Var
88.   quadrado : real
89.   Inicio
90.     quadrado <- lado * lado
91.     retorne quadrado
```

```
92.  fimfuncao
93.
94.  funcao areaRetangulo (bas,alt: real) :real
95.   Var
96.   retangulo : real
97.   Inicio
98.     retangulo <- bas * alt
99.     retorne retangulo
100.   fimfuncao
101.
102.   funcao equacao1grau (m,n,p: real) :real
103.    Var
104.    equacao1grau : real
105.    Inicio
106.      equacao1grau <- m*n+p
107.      retorne equacao1grau
108. fimfuncao
109.
110. procedimento volumeParalelepipedo()
111.  Var
112.   C1, L1, A2 : real
113. Inicio
114.   escreva("Comprimento do paralelepípedo... (C): ")
115.   leia(C1)
116.   escreva("Largura do paralelepípedo..... (L): ")
117.   leia(L1)
118.   escreva("Altura do paralelepípedo..... (A): ")
119.   leia(A2)
120.   escreva("O Volume do Paralelepípedo é: ", C1*L1*A2)
121. fimprocedimento
122.
123. procedimento volumeCilindro()
124. Var
125.   raio, ALTURA1, volcil : real
126. Inicio
127.   escreva("Informe o valor do RAIO: ")
128.   leia(raio)
129.   escreva("Informe o valor da ALTURA: ")
130.   leia(ALTURA1)
131.   volcil <- 3.14 * (raio * raio) * ALTURA1
132.   escreva("O Volume do Paralelepípedo é: ", volcil)
133. fimprocedimento
134.
135. procedimento velocidadeMedia()
136. Var
137.   vmedia, dist, tmp : real
138. Inicio
139.   escreva("Informe a distância percorrida (em KM): ")
140.   leia(dist)
141.   escreva("Informe o tempo gasto (em Horas): ")
142.   leia(tmp)
143.   vmedia <- dist/tmp
144.   escreva("A velocidade média é de: ", vmedia,"Km/h")
145. fimprocedimento
146.
147.  //Aqui inicia o BLOCO PRINCIPAL
148.  Inicio
149.   repita
150.    menuPrincipal()
151.    leia(opcao)
```

```
escreva("")
escolha(opcao)
  caso 1
    menuMatematica()
    leia(opmat)
    escolha(opmat)
      caso 1
        escreva("Digite o valor da aresta: ")
        leia(aresta)
        escreva("A área do quadrado é: ", areaQuadrado(aresta))
      caso 2
        escreva("Digite o valor da base: ")
        leia(base)
        escreva("Digite o valor da altura: ")
        leia(altura)
        escreva("A área do retângulo é: ",areaRetangulo(base,altura))
      caso 3
        escreva("Para equação de 1ºGrau, considere: y=a*i+b")
        escreva("Digite o valor de A...: ")
        leia(a1)
        escreva("Digite o valor de B...: ")
        leia(b1)
        escreva("Digite o valor de I...: ")
        leia(c1)
        escreva("O valor de y é:", equacao1grau(a1,b1,c1))
      outrocaso
        msgErrpadrao()
    fimescolha
  caso 2
    menuFisica()
    leia(opfis)
    escolha(opfis)
      caso 1
        menuOpcaoVolume()
        leia(opfisvol)
        escolha(opfisvol)
          caso 1
            volumeParalelepipedo()
          caso 2
            volumeCilindro()
          outrocaso
            msgErrpadrao()
        fimescolha
      caso 2
        velocidadeMedia()
      outrocaso
        msgErrpadrao()
    fimescolha
  outrocaso
    msgErrpadrao()
fimescolha
ate resposta = "N"
Fimalgoritmo
```

Figura 138 – Algoritmo calculadora matemática/física – blocos independentes

No exemplo da figura anterior, o algoritmo foi organizado em 11 módulos do tipo procedimentos e funções, todos invocados dentro do bloco principal. A estrutura do algoritmo não diminui a quantidade

de linhas, se compararmos a figura 118, exceto pelo bloco principal, o qual ficou menos poluído de comandos, organizado com apenas o esqueleto principal de código.

Para decidir quando usar **procedimento** ou **função**, devemos considerar se o algoritmo deverá ou não retornar algum valor como saída para ser usado dentro do módulo que invocou. Neste exemplo, os cálculos de matemática foram estruturados em funções e o de física em procedimentos a fim de mostrar as diferenças entre as duas práticas.

Para mensagens dos menus, foram implementados procedimentos, uma vez que eles executam os comandos sequenciais e não retornam valores. No exemplo, para as funções de cálculo, as entradas foram inseridas no bloco de códigos principal, isso permite a passagem de parâmetros diferentes e o reaproveitamento da fórmula utilizada. Nos cálculos com uso de procedimentos, as entradas e saídas foram postas diretamente no procedimento, uma vez que não é possível enviar argumentos ou retornar valores.

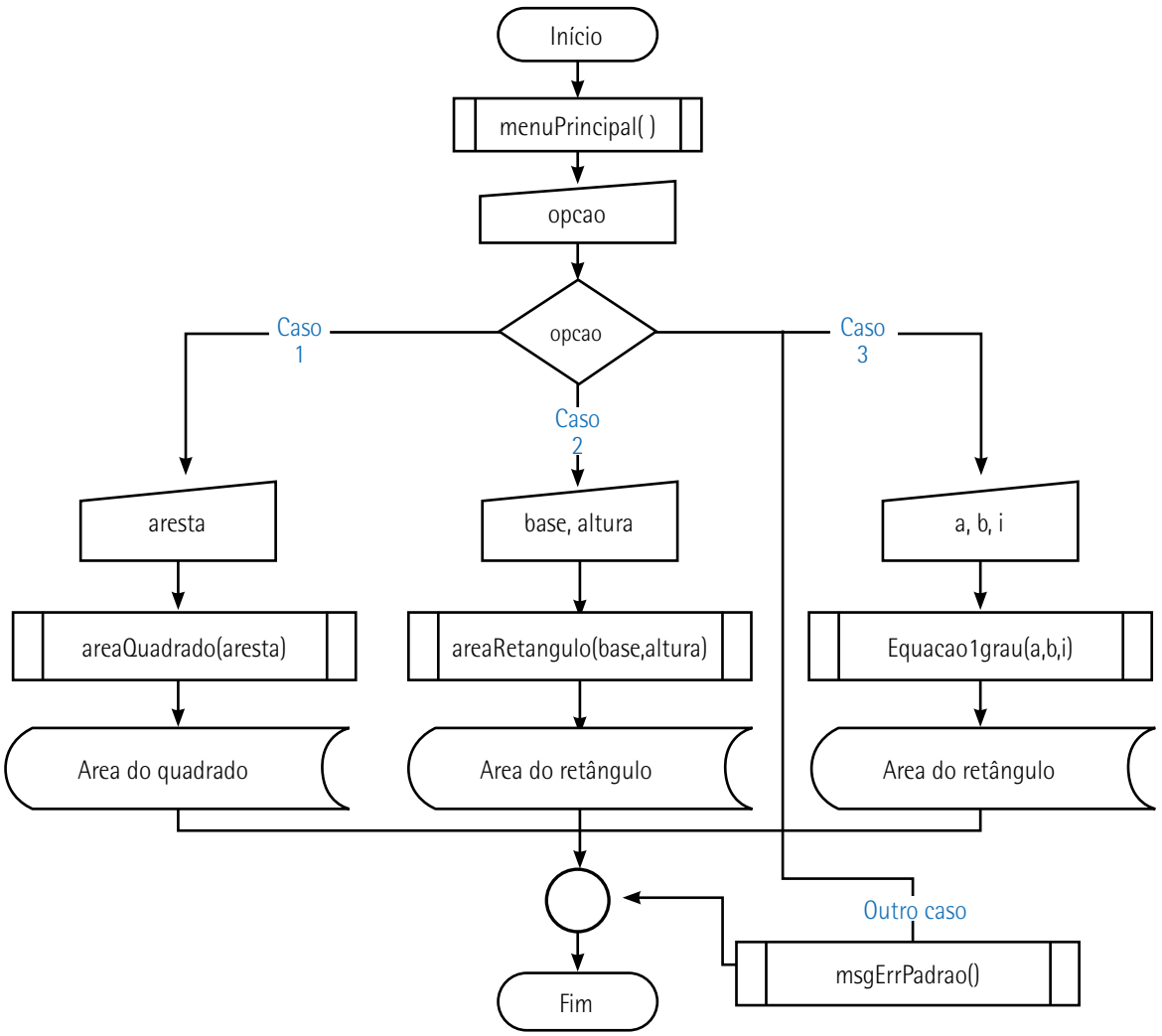


Figura 139 – Fluxograma parcial do algoritmo da figura 138

Resumo

Modularização é a divisão do algoritmo em módulos menores para que o problema seja dividido em subproblemas, reduzindo, assim, sua complexidade, facilitando a compreensão e permitindo o reúso de códigos comuns a outros algoritmos. Ao resolver os problemas menores, o problema mais complexo estará resolvido. Há duas formas de modularizar um algoritmo: procedimentos ou funções.

O procedimento executa sequencialmente um conjunto de comandos e para a chamada do procedimento no bloco principal, basta escrever uma linha de comando da forma **facaalgumacoisa()**.

A função também executa uma sequência de comandos, mas retornará um valor. O tipo de retorno da função deve ser o mesmo tipo do valor de retorno, por isso a função deve ser invocada dentro na atribuição de variáveis, ou como argumento de outra função ou expressão.

var_int ← funcao()

Procedimentos e funções são úteis para separar a lógica aplicada em algoritmos extensos, mas com trechos de código independentes. Ambos podem ser encadeados quando no corpo de um módulo é invocado outro procedimento ou função, conforme pode ser verificado na figura seguinte:

procedimento f() g() escreva(10) : fimprocedimento	procedimento g() h() escreva(20) : fimprocedimento	procedimento h() i() escreva(30) : fimprocedimento	procedimento i() escreva(40) : fimprocedimento

Figura 140 – Representação de procedimentos encadeados

A decisão de usar um algoritmo com procedimentos pode variar de acordo com o contexto. A modularização facilita a manutenção, compreensão e reúso do bloco modularizado. No exemplo da figura anterior, o procedimento **f()** está encadeado porque ele invoca no corpo do módulo o procedimento **g()**. Este, por sua vez, está encadeado porque faz a chamada do procedimento **h()**. Por fim, mas não menos importante, o procedimento **i()** é o único que interrompe as chamadas de função e apenas escreve uma mensagem. Nesse encadeamento, ao chamar a função **f()** serão escritos os números 40, 30, 20 e 10, nessa ordem.

Uma função organiza a estrutura do algoritmo de modo que facilita sua edição e sua manutenção. Uma função provê uma funcionalidade específica e se diferencia do procedimento porque nela sempre é esperada uma resposta, ou seja, o valor de retorno da função.

Os módulos são restritos a um problema menor e bem específico. Por isso são mais simples de serem escritos e compreendidos. Adicionalmente, por terem objetivos claramente definidos, o corpo do módulo é o processamento da solução, e nem sempre são necessários dados de entrada ou saída para o processamento.

Ter um algoritmo organizado facilita a compreensão e a manutenção. Quando novas funcionalidades são necessárias, a inclusão de módulos é mais simples e os blocos de código podem estar separados do código principal, reduzindo a complexidade.

O bloco de código invocado pelo procedimento é escrito fora do algoritmo principal, caracterizando a independência e a oportunidade de reúso. Os procedimentos podem ou não ter parâmetros de entrada, mas nunca retornam qualquer valor para o módulo que o chamou.

Procedimentos que não possuem parâmetros de entrada são úteis para executar tarefas repetitivas a fim de separar blocos extensos do algoritmo principal, dentre outras funcionalidades.

Exercícios

Questão 1. Em uma universidade, um professor solicitou aos alunos que escrevessem um programa para determinar se um número natural maior do que zero é ou não um número primo. Esse número deve ser inserido pelo usuário, e devemos nos certificar de que o número seja maior do que 1 antes de fazer qualquer operação. A saída do programa deve dizer se o número é ou não é primo. Para ajudar os alunos a resolverem o problema, o professor recordou que, se um número natural maior do que zero for também um número primo, ele deverá ter apenas dois divisores naturais: o número 1 e o próprio número.

Com base nesses conhecimentos, um aluno escreveu o programa apresentado a seguir no VisuAlg.

```
Algoritmo "DetectaPrimo"
Var
achou, num, cont: inteiro
Inicio
  escreval("Entre com um numero inteiro maior do que um:")
  leia(num)
  se num > 1 entao
    achou<-0
    cont<-num-1
    enquanto (achou <> 1) e (cont>1) faca
      se num % cont = 0 entao
        achou<-1
      fimse
      cont<-cont-1
    fimenquanto
    se achou=0 entao
      escreval("O numero:",num," e primo.")
    senao
      escreval("O numero:",num," nao e primo.")
    fimse
  senao
     escreva("O numero deve ser inteiro e maior do que um!")
  fimse
FimAlgoritmo
```

Figura 141

Com base no programa e nos seus conhecimentos, assinale a alternativa correta.

A) Se o usuário entrar com o número 27 no programa do enunciado, a saída será: **O numero:27 e primo**. Isso indica que o programa está funcionando de forma correta, pois o número 27 é um número ímpar, e todos os números ímpares são números primos.

B) Se o usuário entrar com o número 27 no programa do enunciado, a saída será: **O numero:27 e primo**. Isso indica que o programa está funcionando de forma errada, pois o número 27 não é um número primo.

C) Se o usuário entrar com o número 27 no programa do enunciado, a saída será: **O numero:27 nao e primo**. Como o número 27 não é um número primo, podemos afirmar que a saída do programa está correta. Além disso, não é necessário fazer mais nenhum teste no programa para afirmarmos que o seu funcionamento está correto, já que o programa funcionou corretamente para uma entrada. Para tal, basta testarmos um único número para estarmos certos de que o programa funciona corretamente.

D) Se o usuário entrar com o número 27 no programa do enunciado, a saída será: **O numero:27 nao e primo**. Como o número 27 não é um número primo, podemos afirmar que a saída do programa está correta. Contudo, se quisermos realmente investigar se o programa está funcionando de forma correta para todas as entradas, devemos fazer mais testes com outros casos e analisar cuidadosamente o código do programa. Um teste simples para apenas uma única entrada pode não ser suficiente para determinarmos se um programa está funcionando de forma correta ou não.

E) Se o usuário entrar com o número 1 no programa do enunciado, a sua saída será: **O numero:1 e primo**.

Resposta correta: alternativa D.

Análise das alternativas

A) Alternativa incorreta.

Justificativa: a saída do programa não é **O numero:27 e primo** para a entrada 27. Além disso, o número 27 não é um número primo (nem todos os números ímpares são primos, como é o caso dos números 9, 15, 81, entre muitos outros).

B) Alternativa incorreta.

Justificativa: como afirmado anteriormente, nota-se que a saída do programa para a entrada 27 não é **O numero:27 e primo**. Na verdade, a saída é: **O numero:27 não e primo**.

C) Alternativa incorreta.

Justificativa: a alternativa apresenta a saída correta do programa, mas erra ao dizer que nenhum teste adicional é necessário. Para sabermos se um programa funciona de forma correta, é necessário testarmos diversas condições diferentes, buscando explorar os diversos estados possíveis do programa.

Em programas muito simples, pode ser possível fazermos poucos testes para avaliarmos seu funcionamento geral. Contudo, na maioria das situações reais, devemos ser cuidadosos e buscar sempre testar nossos programas em várias situações, especialmente naquelas próximas a limites específicos, como os limites de contagens, por exemplo.

D) Alternativa correta.

Justificativa: a alternativa apresenta a saída correta do programa para a entrada especificada e também ressalta a importância dos testes. É fundamental que um programa seja adequadamente testado, para evitar erros possivelmente catastróficos em produção, que podem afetar a vida dos usuários e de outras pessoas.

E) Alternativa incorreta.

Justificativa: a saída do programa será **O numero deve ser inteiro e maior do que um!**, e não **O numero:1 e primo**. Observe que o primeiro teste, logo após a entrada do usuário, existe justamente para nos certificarmos de que o valor inserido pelo usuário esteja dentro dos limites especificados pelo professor, ou seja, que o número inserido pelo usuário seja maior do que 1.

Análise da questão

O algoritmo apresentado no enunciado é bastante simples, mas muito lento. Existem vários algoritmos melhores para o cálculo de números primos. Por exemplo, poderíamos checar apenas os números entre 2 e a raiz quadrada do número inserido pelo usuário, diminuindo consideravelmente o número de testes e tornando a saída mais rápida, especialmente para números maiores.

Fora isso, convém mencionar que não foram utilizados acentos no programa para evitar problemas de codificação, que podem acontecer em algumas máquinas. De forma geral, o VisuAlg tolera bem o uso de acentos, mas eles foram evitados para garantir o seu funcionamento em diversas situações.

Questão 2. Um estudante estava escrevendo um pequeno jogo de adivinhação de números. A ideia do jogo é muito simples: o programa "sorteia" um número inteiro aleatório entre 1 e 10 e armazena esse valor em uma variável. Esse número não deve ser alterado até o fim da execução do programa. O usuário não sabe qual é o número sorteado (o número só é mostrado no final da execução, para que o usuário veja qual é o número que ele deveria ter "adivinhado") e tem três tentativas para tentar adivinhar o número.

A cada tentativa, o programa deve solicitar ao usuário que entre com um número inteiro entre 1 e 10. Ele armazena esse valor em uma variável de entrada e compara esse valor com o número anteriormente sorteado. Se os valores forem iguais, o programa deve interromper as perguntas e dizer ao usuário que ele acertou. Se os valores forem diferentes, o programa deve repetir a pergunta e decrementar em uma unidade o contador de tentativas. Se não restarem mais tentativas e o usuário não tiver acertado o número, o programa informa ao usuário qual foi o número sorteado e termina a execução. A seguir, temos o programa feito pelo estudante, escrito em Portugol com a sintaxe do VisuAlg.

```
Algoritmo "jogo"
Var
segredo: inteiro
entrada: inteiro
tentativas: inteiro
acertou: inteiro
Inicio
  tentativas<-3
  acertou<-0
  segredo<-randi(10)+1
  escreval("Vou pensar em um numero inteiro entre 1 e 10.")
  escreval("Voce tem 3 tentativas para adivinhar o numero!")
  repita
      escreval("Adivinhe o numero que eu estou pensando!")
      leia(entrada)
      se entrada = segredo entao
        acertou<-1
      senao
        escreval("Errou! Tente novamente.")
      fimse
      tentativas<-tentativas-1
  ate (acertou=1) ou (tentativas=0)
  se acertou=1 entao
    escreval("Voce adivinhou!")
  senao
    escreval("Acabaram as tentativas...o valor correto era:",segredo)
  fimse
FimAlgoritmo
```

Figura 142

No programa apresentado, a função **randi(10)** gera (ou "sorteia") um número inteiro aleatório entre 0 e 9. Note que tanto 0 quanto 9 podem ser sorteados pela função. Veja também que o número 9, o limite superior, é igual ao argumento passado para a função "randi" menos 1. Para podermos gerar números inteiros aleatórios entre 1 e 10, basta somarmos 1 ao número retornado(ou "sorteado") pela função "randi". Com base no programa apresentado, nos requisitos desejados do programa e nos seus conhecimentos, avalie as afirmativas:

I – O programa não funciona de forma correta, pois permite quatro tentativas, e não três, que era o valor especificado.

II – Mesmo que o usuário entre com o valor correto na primeira tentativa, o programa continua solicitando números para o usuário, até que o contador de tentativas se torne nulo.

III – Para cada tentativa errada, o programa mostra para o usuário o seguinte texto: "Errou! Tente novamente". Isso acontece mesmo após a última tentativa. Dessa forma, se o usuário errar todas as vezes e esgotar as tentativas, o programa mostrará na tela o seguinte texto: "Errou! Tente novamente.". Logo em seguida, em uma nova linha, teremos: "Acabaram as tentativas...o valor correto era:", com o valor do número a ser adivinhado. Finalmente, o programa termina a execução. Isso denota um problema: no último caso, o programa indica que o usuário pode tentar novamente, o que não é possível nessa situação.

É correto o que se afirma em:

A) I, apenas.

B) II, apenas.

C) III, apenas.

D) I e III, apenas.

E) I, II e III.

Resposta correta: alternativa C.

Análise das afirmativas

I – Afirmativa incorreta.

Justificativa: se analisarmos o código com cuidado, veremos que o programa permite a execução de, no máximo, três tentativas. Observe que, a cada tentativa, no final do laço REPITA... ATÉ, o valor da variável **tentativas** é decrementado de uma unidade. Uma das condições para a interrupção da execução do laço é quando esse valor se tornar nulo. Note ainda que a variável **tentativas** é inicializada com o valor 3.

II – Afirmativa incorreta.

Justificativa: sempre que o usuário entrar com o valor correto, o programa atribui o valor 1 à variável **acertou**. No começo do programa, essa variável é inicializada com o valor nulo. Observe que essa variável é utilizada como uma das condições do laço REPITA... ATÉ. Se o valor da variável **acertou** for igual a 1, o laço é interrompido.

III – Afirmativa correta.

Justificativa: o programa apresenta um problema de usabilidade caso o usuário erre todas as tentativas, indicando que uma nova tentativa pode ser feita, mas terminando a execução. Isso acontece porque a linha com o comando **escreval("Errou! Tente novamente.")** é executada sem a checagem

de quantas tentativas ainda restam para o usuário, fazendo com que a sua execução seja feita mesmo no caso do fim do programa. Como programadores, devemos tomar cuidado em não comunicar informações contraditórias e equivocadas ao usuário. Isso pode induzi-lo a pensar que determinada ação é possível quando esse não é o caso. Normalmente, esses são problemas relacionados à área de usabilidade. Entretanto, é importante vermos como essas questões estão também relacionadas à lógica de programação, como é o caso do programa em questão.

Análise da questão

Como solução alternativa, poderíamos ter utilizado o tipo de dados **logico** para a variável **acertou**, atribuindo valores como **verdadeiro** ou **falso**, em vez de utilizarmos uma variável inteira, atribuindo valores como 0 e 1. Poderíamos dizer que a utilização do tipo de dados **logico** seria mais adequada, uma vez que o comportamento da variável **acertou** é binário na lógica do programa apresentado. Contudo, como esse tipo de situação ainda é bastante comum (especialmente em algumas linguagens de programação nas quais esse tipo de dados não esteve disponível durante muito tempo, como é o caso da linguagem C), é importante lembrar que esse tipo de situação existe.