Unidade II

3 ESTRUTURAS DE DECISÃO

Nem todos os problemas possuem uma sequência única no processo de solução. Em alguns casos, uma decisão precisa ser tomada durante o processo de resolução a fim de definir a sequência correta de comandos a ser realizada. Para problemas que apresentam essa característica, aplicam-se as estruturas de decisão.

Uma estrutura de decisão promove um desvio na sequência de comandos do algoritmo, determinando qual bloco de comandos será executado após a condição ser avaliada. A condição é uma expressão lógica ou relacional que pode ou não ser satisfeita.

A maioria das linguagens de programação implementam duas estruturas de decisão, o comando **se... então** e a estrutura **escolha**. Ambas podem ser simples ou encadeadas e podem implementar outras estruturas diferentes.

3.1 Estrutura de decisão simples — SE... ENTÃO

O comando **SE... ENTÃO** é uma estrutura de decisão que avalia uma condição e, se ela for **verdadeira**, escolhe um comando ou bloco de comandos a ser executado. Se dois ou mais comandos forem executados, eles devem estar dentro de um bloco de comandos. Se a condição for **falsa**, o desvio é feito e o comando ou o bloco de comandos não é executado. Observe a figura a seguir, que representa o desvio condicional.

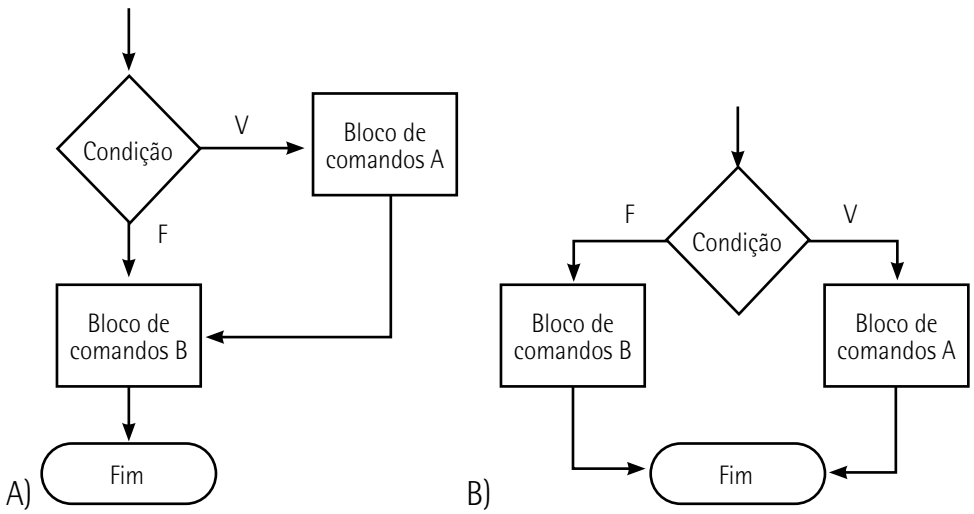

Figura 48 – Fluxogramas da estrutura de decisão simples SE... ENTÃO: A) estrutura de decisão simples; B) estrutura de decisão composta

A parte A) da figura anterior corresponde ao fluxograma da estrutura de decisão simples, **SE... ENTÃO**, ou seja, o bloco de comandos A será executado se, e somente, a condição for **verdadeira**; já o bloco de comandos B sempre será executado.

A parte B) da figura anterior apresenta o fluxograma da estrutura de decisão composta, **SE... SENÃO**, ou seja, se a condição for **verdadeira**, o bloco de comandos A será executado, se não (se for **falsa**) o bloco de comandos B será executado.

O pseudocódigo da estrutura de decisão simples é apresentado na figura seguinte. Conforme demonstrado à esquerda da figura, em A), um único comando é executado se a condição for **verdadeira**. Já à direita, em B), dois ou mais comandos são executados quando a condição resultar **verdadeira**. O bloco dentro da condição é expresso pelos delimitadores **início** e **fimse**.

se (<condição>) então <comando> Fimse A)	se (<condição>) então início <comando 1> <comando 2> : <comando n> Fimse B)

Figura 49 – Pseudocódigo para a estrutura de decisão simples SE... ENTÃO A) o bloco de comandos é composto por uma única linha B) o bloco de comandos é composto por duas ou mais linhas

Algumas linguagens de programação, tais como C/C++, Java e C# utilizam como delimitadores de bloco os símbolos { e } para início e fim do bloco, respectivamente. Em Python, o bloco é definido pela indentação. Portanto, é importante entender quando a estrutura de blocos é necessária e escrever códigos formatados como boa prática.

A estrutura **SE... ENTÃO** funciona como a implicação lógica (A → B, lida da seguinte forma: **se A então B**) estudada em lógica matemática (ALENCAR FILHO, 2017).

A figura a seguir traz exemplos de implicação lógica:

Se fizer sol **então** Aline irá à praia.	**Se** João passar no vestibular **então** ganhará um carro.
A: Está sol.	**A:** João passou no vestibular.
B: Aline irá à praia.	**B:** João ganhou um carro.
Se A **então** B	**Se** A **então** B
Se A for verdade, ou seja, se está sol, **então** Aline irá à praia.	**Se** A for verdade, ou seja, se João passou no vestibular, **então** João ganhará o carro.

Figura 50 – A estrutura de decisão simples e a implicação lógica

O comando **SE... ENTÃO** é muito importante na programação de computadores e muito utilizado em diferentes contextos pela característica de promover um desvio na execução dos comandos como

em tomadas de decisão. O algoritmo da figura seguinte codifica a implicação lógica explicada na figura anterior. Observe que a linha 10 não será executada se, em todas as execuções do algoritmo, a resposta à pergunta apresentada no comando de saída da linha 6 capturada no comando de entrada da linha 7 for diferente de "**S**".

```
1.  Algoritmo "Exemplo Estrutura de Decisão Simples"
2.  Var
3.          A : caractere
4.   Início
5.          //entrada
6.          escreva("Está sol? Responda S para sim ou N para não ")
7.          leia(A)
8.          //processamento
9.          se (A = "S") entao
10.           escreva("Aline irá à praia")
11.         fimse
12. Fimalgoritmo
```

Figura 51 – Exemplo de pseudocódigo da estrutura de decisão simples (A → B)

A estrutura de decisão promove um desvio no algoritmo. No exemplo da figura anterior, se a resposta do usuário for S, a tela mostrará a mensagem **Aline irá à praia**. Contudo, qualquer caractere diferente de S produzirá o encerramento do algoritmo sem nenhuma interação adicional.

É possível colocar várias condições dentro do algoritmo. No algoritmo da figura seguinte, os comandos são executados na ordem sequencial. Na linha 11, a condição é testada. Se x for maior que y, a linha 12 será executada e a mensagem **x é maior que y** será escrita. Se x não for maior que y, a linha 12 não será executada e o algoritmo encontrará o **fimse**, instrução que finaliza a estrutura de decisão. Da mesma forma, as linhas 15 e 18 serão executadas apenas se as condições das linhas que as antecedem resultarem verdadeiras.

```
1.  Algoritmo "Exemplo com varias estruturas de Decisão Simples"
2.  Var
3.          x, y : inteiro
4.  Início
5.          //entrada
6.          escreva("Digite um valor para X: ")
7.          leia(x)
8.          escreva("Digite um valor para Y: ")
9.          leia(y)
10.         //processamento
11.         se (x>y) então
12.           escreva(x, " é maior que ", y)
13.         fimse
14.         se (x<y) então
15.           escreva(x, " é menor que ", y)
16.         fimse
17.         se (x=y) então
18.           escreva(x, " é igual a ", y)
19.         fimse
20.         escreva("Fim dos testes")
21. fimalgoritmo
```

Figura 52 – Algoritmo em pseudocódigo com três decisões simples

A figura subsequente ilustra a lógica da programação por um fluxograma que representa o pseudocódigo da figura anterior.

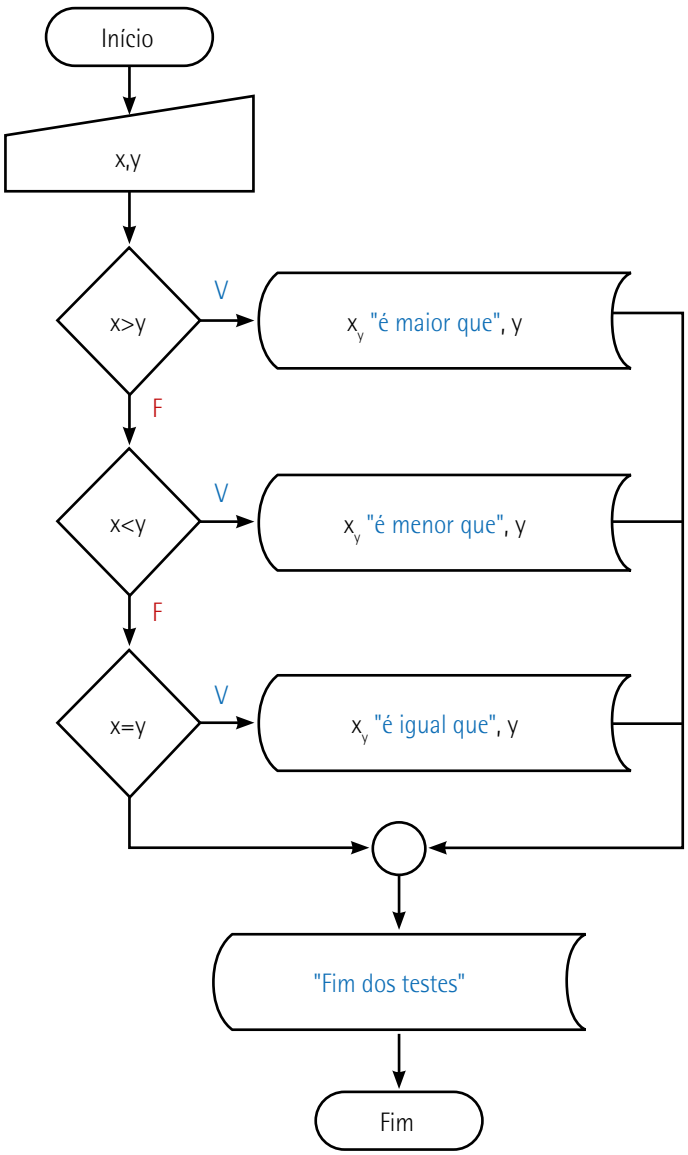

Figura 53 – Algoritmo em fluxograma com quatro decisões simples

O exemplo a seguir propõe um problema para ser resolvido por meio de algoritmos. No exemplo da figura seguinte, foram criados cabeçalhos para a criação do programa. Essa é uma boa prática e não é aplicada em todos os algoritmos deste livro-texto por limitação de espaço.

Exemplo

Projete um algoritmo que receba a altura e o sexo de uma pessoa, calcule e mostre o peso ideal, utilizando as seguintes regras:

- sexo feminino: o peso ideal será (62.1 * altura) – 44.7
- sexo masculino: o peso ideal será (72.7 * altura) – 58

```
1.  Algoritmo "Peso Ideal"
2.  // Disciplina  : Lógica de Programação e Algoritmos
3.  // Professora  : Eliane Oliveira Santiago
4.  // Descrição   : Programa para calcular o peso ideal
5.  // Data atual  : 02/02/2021
6.  Var
7.     altura, peso_ideal : real
8.     sexo : caracter
9.
10. Inicio
11.       //entrada
12.       escreva("Qual é a sua altura? ")
13.       leia(altura)
14.
15.       escreva("Qual é o seu sexo [M/F]?")
16.       leia (sexo)
17.
18.       //processamento
19.       se (sexo="F") então
20.          peso_ideal <- (62.1*altura)-44.7
21.       senão
22.          peso_ideal <- (72.7*altura)-58
23.       fimse
24.
25.       //saida
26.       escreva("Seu peso ideal é : ", peso_ideal)
27. Fimalgoritmo
```

Figura 54 – Algoritmo em pseudocódigo para calcular o peso ideal de uma pessoa

Este algoritmo utiliza uma estrutura de decisão simples e, se o usuário digitar qualquer letra diferente de F, o algoritmo calculará o peso ideal com a segunda fórmula.

3.2 Estrutura de decisão composta — SE... SENÃO

A estrutura de decisão composta decide um entre dois comandos ou blocos de comandos. **Se** a condição for verdadeira, **então** execute o bloco A, **senão**, execute o bloco B. Essa estrutura de decisão composta é chamada **SE... SENÃO** e sua lógica é representada no fluxograma da figura seguinte.

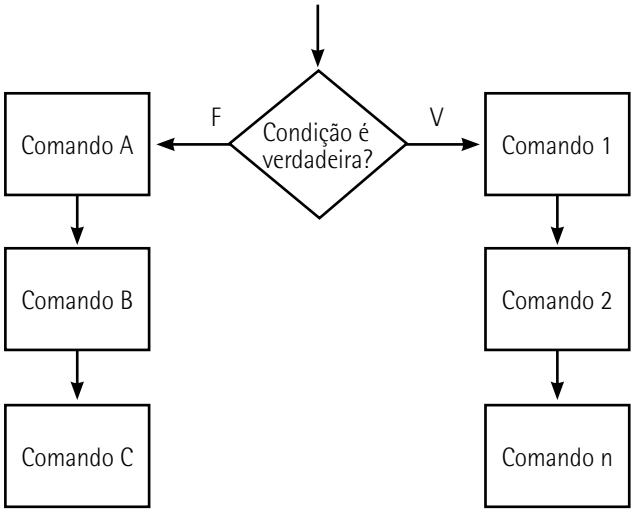

Figura 55 – Fluxograma da estrutura de decisão composta SE... SENÃO

Um problema matemático que aplica a estrutura de decisão composta é a equação de segundo grau dada por:

$f(x) = ax^2 + bx + c$

Resolver essa equação implica primeiro calcular o delta (Δ) e na sequência as raízes x_1 e x_2. **Se** $\Delta \geq 0$, calcular as raízes, **senão** escrever a mensagem "Não existem raízes reais". As fórmulas do Δ e para extrair as raízes do problema são, respectivamente:

$\Delta = b^2 - 4 . a . c$

$$x = \frac{-b \pm \sqrt{\Delta}}{2.a}$$

As variáveis do problema para resolver essa equação são **a**, **b** e **c**. Uma instância para esse problema é:

$f(x) = x^2 - 3x - 10$

Para essa instância, a solução é dada na figura seguinte:

$a = 1$	$b = -3$	$c = -10$	$\Delta > 0$ implica calcular raízes
$\Delta = b^2 - 4 . a . c$	$\leftrightarrow \Delta = (-3)^2 - 4 * 1 * (-10)$ $\leftrightarrow \Delta = 9 + 40$ $\leftrightarrow \Delta = 49$		$x = \frac{-b \pm \sqrt{\Delta}}{2.a} \rightarrow$

Figura 56 – Solução matemática para a equação $f(x) = x^2 - 3x - 10$

O algoritmo da figura anterior é apresentado na próxima figura:

A) Pseudocódigo

```
Algoritmo "Equação de Segundo Grau"
Var
        //declaração das variáveis
        a, b, c, delta, x1, x2 : real
Início
        //entrada
        escreval("Programa para calcular f(x)=ax2+bx+c")
        escreval("a: ")
        leia(a)
        escreval("b: ")
        leia(b)
        escreval("c: ")
        leia(c)

        //processamento
        delta ← b*b-4*a*c
        se (delta>=0) então
          x1 <- (-1*b+raiz(delta))/(2*a)
          x2 <- (-1*b-raiz(delta))/(2*a)
          //saída se condição verdadeira.
          escreva("A raiz 1 é ", x1, "e a raiz 2 é ", x2")
        senão
          //saída se condição falsa
          escreva("Não existem raízes reais")
        fimse
Fimalgoritmo
```

B) Fluxograma

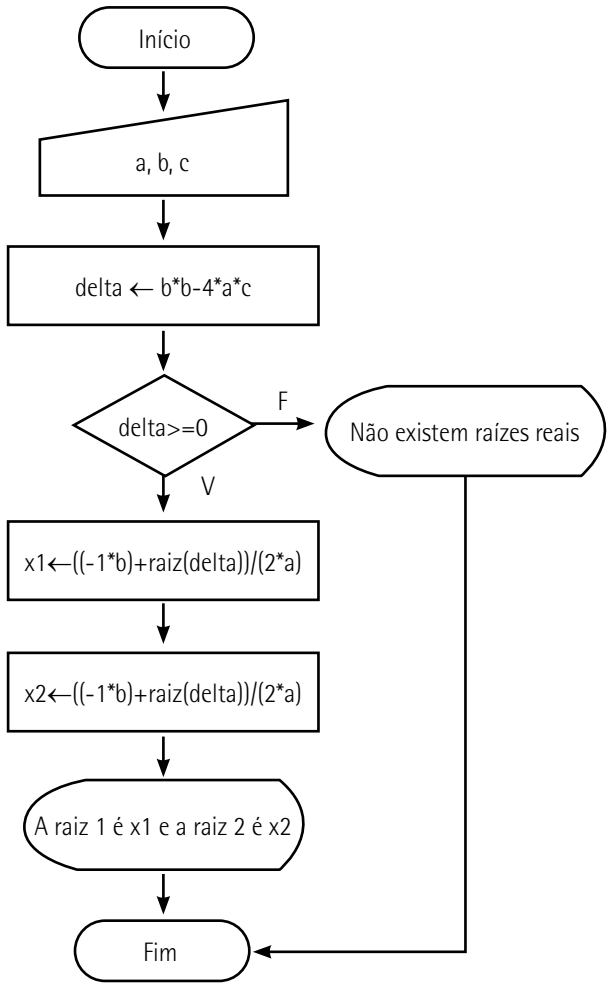

Figura 57 – Algoritmo para a resolução da equação de 2º grau

O algoritmo em descrição narrativa para este problema é:

> Informar os valores para as variáveis a, b e c.
>
> Calcular o delta.
>
> Se o delta é maior ou igual a zero, então calcule as raízes.
>
> Senão escreva a mensagem "Não existem raízes reais".

3.3 Estrutura de decisão encadeada SE... SENÃO

Há situações em que as decisões podem ser dependentes de muitas e diferentes condições e, nesse caso, as decisões devem ser encadeadas, ou seja, uma estrutura dentro da outra, conforme ilustrado na figura a seguir.

```
se (<condiçao1>) entao
se (<condiçao2>) entao
        se (<condiçao3>) entao
          ... // continua com outra estrutura de decisao ou comandos. _      senao
          ... // continua com outra estrutura de decisao ou comandos. _      fimse // fim da condicao 3
 fimse // fim da condicao 2
senao
        se (<condição4>) entao
          se (<condição5>) entao
            ... // continua com outra estrutura de decisao ou comandos.      senao
            ... // continua com outra estrutura de decisao ou comandos.      fimse // fim da condicao 5
        fimse // fim da condicao 4
fimse // fim da condicao 1
```

Figura 58 – Sintaxe da estrutura de decisão encadeada

Exemplo

Escreva um algoritmo que receba três valores reais, verifique e mostre se esses valores podem ser o comprimento dos lados de um triângulo.

Solução

A condição para ser triângulo é a soma de cada um dos lados ser menor ou igual que a soma dos outros dois. Se a condição for verdadeira, verificar se formam um triângulo equilátero, isósceles ou escaleno, senão escrever a mensagem ao usuário: "**As medidas não formam um triângulo!**".

Dicas:

- Triângulo equilátero tem os três lados iguais.
- Triângulo isósceles tem dois lados iguais e um diferente.
- Triângulo escaleno tem os três lados diferentes.

O algoritmo que resolve este problema é:

```
1.  Algoritmo "Triangulo"
2.  Var
3.  L1, L2, L3: real
4.  eh_triangulo : lógico
5.  Início
6.  escreva("Digite a medida do Lado 1")
7.  leia(L1)
8.  escreva("Digite a medida do Lado 2")
9.  leia(L2)
10. escreva("Digite a medida do Lado 3")
11. leia(L3)
12.
13.         eh_triangulo ← ((L1<=L2+L3) e (L2<=L1+L3) e (L3<=L1+L2))
14.
15.         se (eh_triangulo = FALSO) então
16. escreva("As medidas não formam um triângulo!")
17.     senão
18. se ((L1=L2) e (L2=L3)) então
19.         escreva("Triangulo equilátero")
20.         senão
21.           se (L1=L2) ou (L2=L3) ou (L1=L3) então
22. escreva("Triangulo isósceles")
23.           senão
24. escreva("Triangulo escaleno")
25.           fimse
26.         fimse
27.         fimse
28. Fimalgoritmo
```

Figura 59 – Pseudocódigo que aplica a estrutura de decisão encadeada. Algoritmo para decidir se três medidas podem ser as bases de um triângulo e, em caso afirmativo, decidir qual o tipo de triângulo (equilátero, isósceles ou escaleno)

No exemplo da figura anterior, a variável lógica eh_triangulo recebe uma expressão lógica, logo guardará **verdadeiro** ou **falso**, a depender dos valores das variáveis L1, L2 e L3. Se eh_triangulo armazena o valor lógico **falso**, então a condição é verdadeira e o comando de saída que escreve a mensagem de que as medidas não formam um triângulo é executado e o algoritmo é finalizado. No entanto, se eh_triangulo não guarda o valor lógico **falso**, significa que é **verdadeiro**, ou seja, as medidas formam um triângulo, a condição testada é falsa e o fluxo segue para decidir qual é o tipo de triângulo. A figura posterior mostra a lógica da programação expressa pelo fluxograma equivalente.

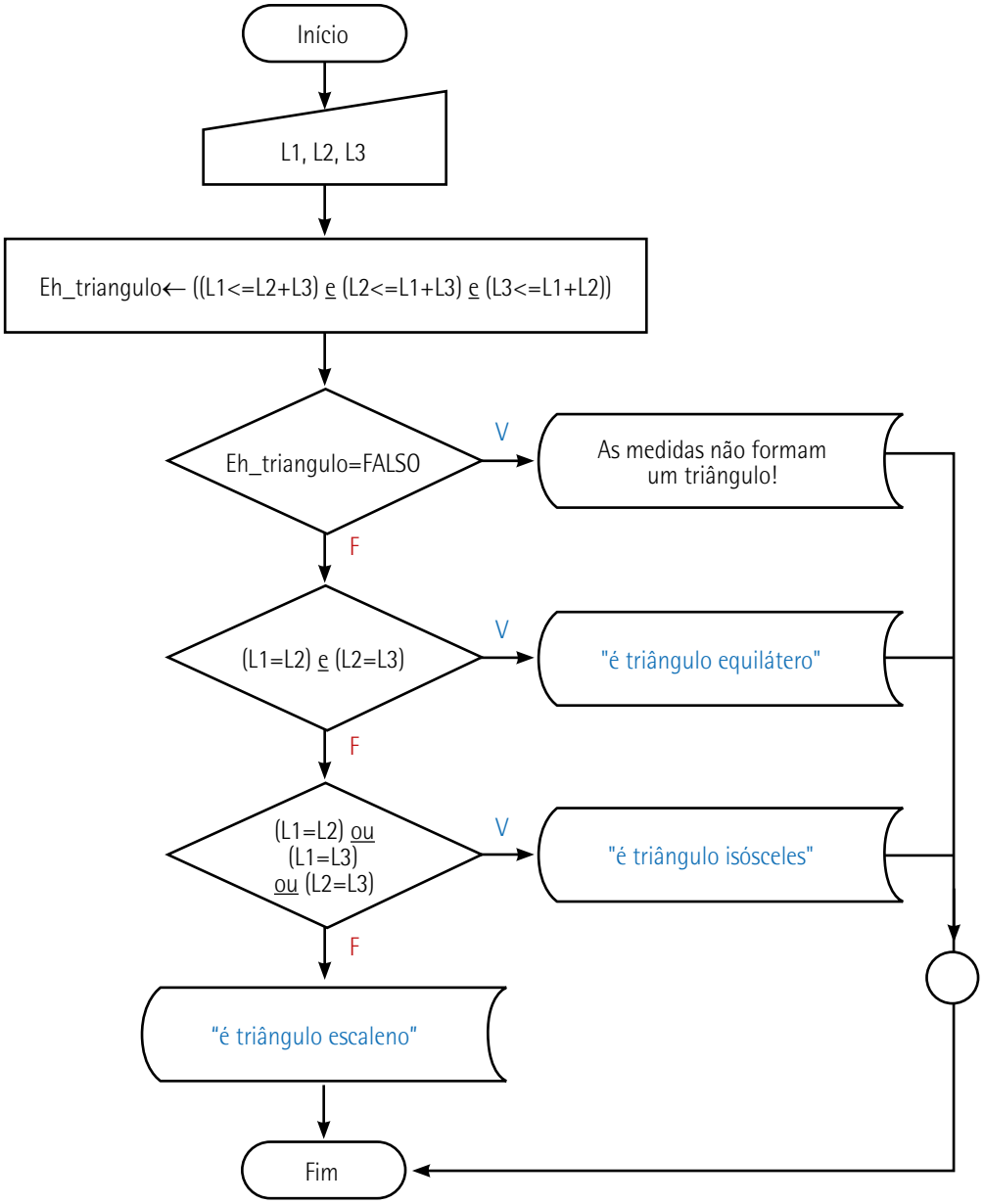

Figura 60 – Fluxograma que aplica a estrutura de decisão encadeada. Algoritmo para decidir se três medidas podem ser as bases de um triângulo e, em caso afirmativo, decidir qual o tipo de triângulo (equilátero, isósceles ou escaleno)

O encadeamento deve ser evitado sempre que possível. O uso de expressões lógicas ajuda a compreender melhor a lógica do algoritmo.

Observe as duas expressões encadeadas da forma a seguir:

```
se (condição1 = verdadeira) então
  se (condição2 = verdadeira) então
        //faça alguma coisa
  fimse
fimse
```

Figura 61

Elas mostram que se a condição 1 e a condição 2 forem verdadeiras, o comando no interior da estrutura deverá ser executado.

No exemplo da figura seguinte, é apresentada a condicional usando a conjunção ao combinar ambas as condições com o operador **e** lógico.

```
se(condição1 = verdadeira) e (condição2 = verdadeira) então
        //faça alguma coisa
fimse
```

Figura 62

A condição analisada na estrutura de decisão **SE... SENÃO** retornará sempre **verdadeiro** ou **falso**. O algoritmo da próxima figura escreve os valores informados nas variáveis n1, n2 e n3 em ordem crescente. A decisão tem sempre dois termos e, quando três ou mais devem ser testados, é importante usar os operadores lógicos para combinar as expressões e o uso do encadeamento da decisão pode ser necessário.

Exemplo

Escreva um algoritmo que receba três diferentes números inteiros e escreva-os em ordem crescente (figura a seguir).

```
1.  Algoritmo "Ordenar_3_Numeros"
2.  Var
3.  n1, n2, n3: inteiro
4.  Inicio
5.  escreva("Número 1: ")
6.  leia(n1)
7.  escreva("Número 2: ")
8.  leia(n2)
9.  escreva("Número 3: ")
10. leia(n3)
11.
12.         // validação dos dados de entrada
13.         se (n1=n2) e (n1=n3) e (n2=n3) entao
14.           escreval("Dados de entrada são iguais")
15.         senao // os dados sao diferentes
16.         se (n1<n2) e (n1 < n3) então
17.           se (n2<n3) então
18.                   escreva(n1, " < ", n2, " < " , n3)
19.           senão
20.               escreva(n1, " < ", n3, " < " , n2)
21.           fimse
22.         senao
23.         se (n2<n1) e (n2<n3) entao
24.                   se (n1<n3) entao
25.                           escreva(n2, " < ", n1, " < " , n3)
26.                   senão
27.                           escreva(n2, " < ", n3, " < " , n1)
28.                   fimse
29.           senao
30.                   se (n3<n1) e (n3 < n2) entao
31.                           se (n1<n2) entao
32.                                   escreva(n3," < ", n1," < " , n2)
33.                           senão
34.                                   escreva(n3," < ", n2," < " , n1)
35.                           fimse
36.                   fimse
37.           fimse
38.         fimse
39.         fimse
40. Fimalgoritmo
```

Figura 63 – Algoritmo em pseudocódigo para ordenar três números. Exemplo de aplicação de estrutura de decisão encadeada

Lembrete

O deslocamento, que na programação é chamado de indentação, ajuda a delimitar o início e o fim dos blocos. Como se pode ver no exemplo da figura anterior, na qual há várias estruturas de decisão encadeadas, a indentação torna-se imprescindível para o entendimento do código, portanto, dar o deslocamento no encadeamento dos comandos SE...SENÃO é importante para entender qual bloco de comando será executado em cada condição.

As estruturas encadeadas podem ter suas lógicas corretas, mas o encadeamento de instruções torna o código mais difícil de ser compreendido. Sempre que existem dois ou mais comandos **SE** encadeados, é importante avaliar se a lógica não pode ser reescrita usando operadores lógicos.

O enunciado sugere a entrada de três números diferentes e observe que as linhas 16 e 17 poderiam ser escritas em uma única linha de comando usando o operador lógico de conjunção da forma:

```
se ((n1<n2) e (n1<n3) e (n2<n3)) então
```

Figura 64

O uso de parênteses externos é importante para tornar clara a expressão lógica composta por dois termos e cada um deles é formado por duas operações relacionais. Para que a condição seja verdadeira, ambas as condições devem ser verdadeiras e o resultado é mostrado na tela do usuário, afirmando que n1 é menor que n2 e n2 é menor que n3.

O algoritmo da figura subsequente resolve o mesmo problema do anterior, mas foi simplificado com o uso de expressões lógicas.

Exemplo

A figura seguinte traz um exemplo de aplicação de estrutura de decisão encadeada.

```
1.  Algoritmo "Ordenar_3_Numeros"
2.  Var
3.  n1, n2, n3: inteiro
4.  Inicio
5.          escreva("Número 1: ")
6.          leia(n1)
7.          escreva("Número 2: ")
8.          leia(n2)
9.          escreva("Número 3: ")
10.         leia(n3)
11.         se (n1<n2) e (n1<n3) e (n2<n3) então
12.     escreva(n1, " < ", n2, " < " , n3)
13.         senão
14.           se (n1<n2) e (n1<n3) e (n3<n2) então
15.       escreva(n1, " < ", n3, " < " , n2)
16.     senão
17.       se (n2<n1) e (n2<n3) e (n1<n3) então
18.         escreva(n2, " < ", n1, " < " , n3)
19.       senão
20.         se (n2<n1) e (n2<n3) e (n3<n1) então
21.           escreva(n2, " < ", n3, " < " , n1)
22.         senão
23.                 se (n3<n1) e (n3<n2) e (n1<n2) então
24.                         escreva(n3, " < ", n1, " < " , n2)
25.           senão
26.               escreva(n3, " < ", n2, " < " , n1)
27.           fimse
28.         fimse
29.       fimse
30.     fimse
31.   fimse
32. fimalgoritmo
33.
```

Figura 65 – Algoritmo em pseudocódigo para ordenar três números – encadeamento reduzido pelo uso de expressões lógicas

Nesse exemplo está sendo usada apenas a estrutura de decisão composta. Os muitos níveis do comando **SE... SENÃO** ocorrem porque cada decisão tem apenas uma instrução, que é outro **SE**.

Para esse problema, existem seis possibilidades de ordenação possível, conforme demonstrado no quadro a seguir:

Quadro 16

N1	N2	N3	Linhas 12 e 13
N1	N3	N2	Linhas 15 e 16
N2	N1	N3	Linhas 18 e 19
N2	N3	N1	Linhas 21 e 22
N3	N1	N2	Linhas 24 e 25
N3	N2	N1	Linha 27

Na linha 12 do exemplo, está sendo verificado se n1 é o menor e n2 o segundo menor. **Se** a expressão for verdadeira, os três números são escritos na ordem crescente, **senão** uma nova condição será testada na linha 15.

Na condição da linha 15 se testa se é verdade que n1 é o menor e n3 é o segundo menor. Se for verdade, será escrito n1 < n3 < n2.

Se as duas condições anteriores não forem verdadeiras, na condição da linha 18, é verificado se n2 é o menor e o n1 é o segundo menor. Se a condição resultar verdadeira, será escrito na linha 19 n2 < n1 < n3.

Da mesma forma, a condição da linha 21 será verificada apenas se as anteriores forem falsas. Na linha 21, está sendo verificado se n2 é o menor e se n3 é o segundo menor. Se a condição for verdadeira, o comando da linha 22 escreverá n2 < n3 < n1.

Nesse ponto, se todas as condições forem falsas, é porque o menor número é o n3. De acordo com a lógica apresentada, o segundo menor ainda precisa ser verificado. Portanto, na linha 24 é verificado se é verdade que n3 é o menor e se n1 é o segundo menor. Se a resposta for afirmativa, a sequência crescente dos números será n3 < n1 < n2.

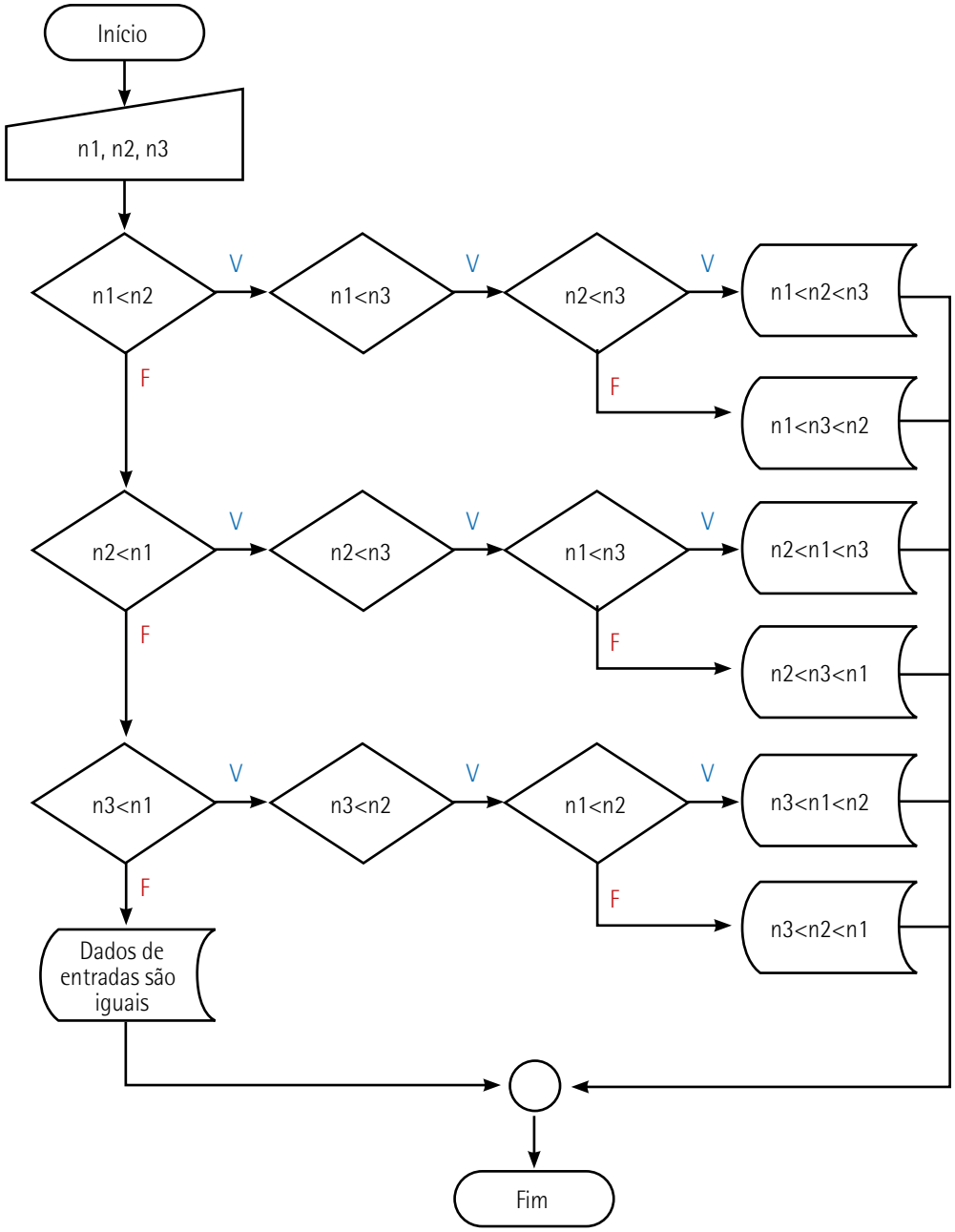

Figura 66 – Fluxograma do algoritmo para escrever três números em ordem crescente – exemplo de estrutura de decisão encadeada

Dica

Nas linguagens de programação derivadas da linguagem C, tais como C++ e Java, a estrutura de decisão é codificada com os comandos **if (condição) else...**, mas não existe o comando correspondente ao comando **fimse** responsável por delimitar o fim do bloco **SE** no pseudocódigo. Adicionalmente, a estrutura aceita um único comando, tornando opcional o uso dos delimitadores { de início e } de fim do bloco, mas obrigatório para blocos.

A)

```
if(condicao)
  //comando
else
{
  //comando1
  //comando2
     :
  //comando n
}
```

B)

```
if(condicao)
  if(condicao)
    if(condicao){
      //comandos
    }
```

Figura 67 – Estrutura de decisão codificada em Linguagem C: A) composta; B) aninhada

Lembrete

Dentro da condicional, se há apenas uma única linha de comando, o uso dos delimitadores { e } são opcionais. Esses delimitadores significam, respectivamente, início e fim de bloco.

Quando há dois ou mais comandos, é obrigatório o uso dos delimitadores { e }

Exemplo

O índice de massa corporal (IMC) é uma medida internacional usada para calcular se uma pessoa está no peso ideal. Trata-se de um método fácil e rápido para a avaliação do nível de gordura de cada pessoa, sendo, por isso, um preditor internacional de obesidade adotado pela Organização Mundial da Saúde (OMS).

O IMC é determinado pela divisão da massa do indivíduo pelo quadrado de sua altura, em que a massa está em quilogramas e a altura em metros.

Quadro 17 – Classificação de IMC

Resultado	Situação
Abaixo de 17	Muito abaixo do peso
Entre 17 e 18,49	Abaixo do peso
Entre 18,50 e 24,99	Peso normal
Entre 25 e 29,99	Acima do peso
Entre 30 e 34,99	Obesidade I
Entre 35 e 39,99	Obesidade II (severa)
Acima de 40	Obesidade III (mórbida)

Adaptado de: Brasil (2009).

Projete um algoritmo para ler a altura e o peso de uma pessoa, calcular o IMC e mostrar a situação em que se encontra usando como base a classificação no quadro anterior.

Resolução

```
1.  Algoritmo "IMC"
2.  Var
3.    altura, massa, imc : real
4.  Inicio
5.      //entrada
6.      escreva("Peso.....: ")
7.      leia(massa)
8.      escreva("Altura...: ")
9.      leia(altura)
10.
11.     //processamento
12.     imc <- massa/(altura*altura))
13.
14.     //saida
15.     se (imc<17) então
16.       escreva("Muito abaixo do peso")
17.     senao
18.       se ((imc>=17) E (imc<18.5)) entao
19.         escreva("Abaixo do peso")
20.       senao
21.         se ((imc>=18.5) E (imc<25)) entao
22.           escreva("Peso Normal")
23.         senao
24.           se ((imc>=25) E (imc<30)) entao
25.             escreva("Acima do peso")
26.           senao
27.             se ((imc>=30) E (imc<35)) entao
28.               escreva("Obesidade I")
29.             senao
30.               se ((imc>=35) E (imc<40)) entao
31.                 escreva("Obesidade II")
32.               senao
33.                 escreva("Obesidade III")
34.               fimse
35.             fimse
36.           fimse
37.         fimse
38.       fimse
39.     fimse
40. Fimalgoritmo
```

Figura 68 – Algoritmo para calcular o IMC – estrutura encadeada

3.4 Estrutura de decisão ESCOLHA–CASO

A estrutura **ESCOLHA-CASO** oferece uma forma organizada para agrupar os comandos. A lógica dessa estrutura consiste em verificar o valor da variável que controlará a decisão e uma ação diferente será executada para cada valor que a variável poderá assumir.

Nessa estrutura, a variável de verificação deve ser discreta, por isso, em algumas linguagens de programação, o tipo deve ser caractere ou inteiro. Uma variável discreta pode assumir um número finito e contável de valores. A sintaxe do comando **ESCOLHA-CASO** é apresentada na figura que segue.

```
escolha (<variavel>)
        caso <valor1>
          <comandos>
        caso <valor2>, <valor3>, <valor4>
          <comandos>
        caso <valor5>
          <comandos>
        outrocaso
          <comandos>
fimescolha
```

Figura 69 – Pseudocódigo para a estrutura de decisão simples ESCOLHA-CASO

Exemplo 1

Escreva um algoritmo para perguntar ao usuário "Como está o dia hoje?" e decidir se Aline, de um dos nossos exemplos anteriores, irá à praia. Caso o tempo esteja ensolarado, ela irá à praia, caso esteja nublado ou chuvoso, ela não irá.

Solução

Neste exemplo, a variável tempo:

- é discreta, porque poderá assumir apenas quatro valores: ensolarado, nublado, chuvoso e nevoso;
- possui um dado do tipo caractere;
- será verificada quanto ao seu valor.

Caso o dado verificado na variável tempo seja "**E**" então os comandos **escreva("O dia está ensolarado")** e **escreva("Aline irá à praia")** serão executados. Caso o dado verificado na variável tempo seja "**N**" ou "**C**", então os comandos **escreva("O dia está nublado ou chuvoso")** e **escreva("Aline não irá à praia")** serão executados. Para qualquer outro valor que a variável tempo possa assumir, será executado o comando **escreva("Talvez esteja nevando!!!")**.

```
Algoritmo "Aline irá à praia"
Var
   tempo : caracter
Inicio
        escreva("Como está o dia hoje?")
        leia(tempo)
        escolha (tempo)
         caso "E"
          escreva("O dia está ensolarado")
          escreva("Aline irá à praia")
         caso "N", "C"
          escreva("O dia está nublado ou chuvoso")
          escreva("Aline não irá à praia")
         outrocaso:
          escreva("Talvez esteja nevando!!!")
        fimescolha
Fimalgoritmo
```

Figura 70 – Algoritmo para decidir com base na estrutura ESCOLHA-CASO, quando um caso trata dois valores

Exemplo 2

No algoritmo da próxima figura, a variável a ser verificada é do tipo inteiro.

```
1.  Algoritmo "Múltipla Escolha em cada caso"
2.  Var
3.     valor : inteiro
4.  Inicio
5.           escreva("Escreva um número inteiro")
6.           leia(valor)
7.
8.           escolha (valor)
9.       caso 1
10.          escreva("Número igual a 1")
11.      caso 2, 3, 4, 5, 6, 7, 8
12.          escreva("número entre 2 e 8")
13.      caso 10, 11, 12
14.          escreva("número 10, 11 ou 12")
15.      outrocaso
16.          escreva("outro número")
17.    fimescolha
18. Fimalgoritmo
```

Figura 71 – Algoritmo para decidir com múltiplas escolhas em cada caso

A variável valor é arbitrária, pois poderá assumir qualquer valor inteiro e os números são infinitos. No entanto, apenas o subconjunto dos números inteiros que consiste dos números de 1 a 12 são considerados no programa.

Na estrutura **ESCOLHA... CASO**, expressões com operadores relacionais não são aceitas. Portanto, se a ideia é considerar uma lista de valores, o programador poderá listar os valores separados por vírgula, como mostrado nas linhas 11 e 13 do algoritmo da figura anterior.

Dica

No VisuAlg, o intervalo entre valores usando pontos (..) não é aceito. Esta notação poderá ser usada na linguagem de programação C/C++. A figura seguinte mostra as listas aceitas na estrutura de decisão ESCOLHA-CASO.

caso 2, 3, 4, 5, 6, 7, 8	Forma correta
	Forma correta.
caso 2 .. 8	
	Esta sintaxe não é aceita no VisuAlg, mas é aceita em alguns compiladores da Linguagem C/C++.
caso (valor>=2 E valor <=8)	Errado

Figura 72 – Listas de opções aceitas para cada caso da estrutura ESCOLHA-CASO

A figura a seguir representa o fluxograma da estrutura de decisão ESCOLHA-CASO. Cada instrução representada no fluxograma pode ser substituída por um bloco de comandos, ou seja, uma lógica de programação que implementa uma regra de negócio.

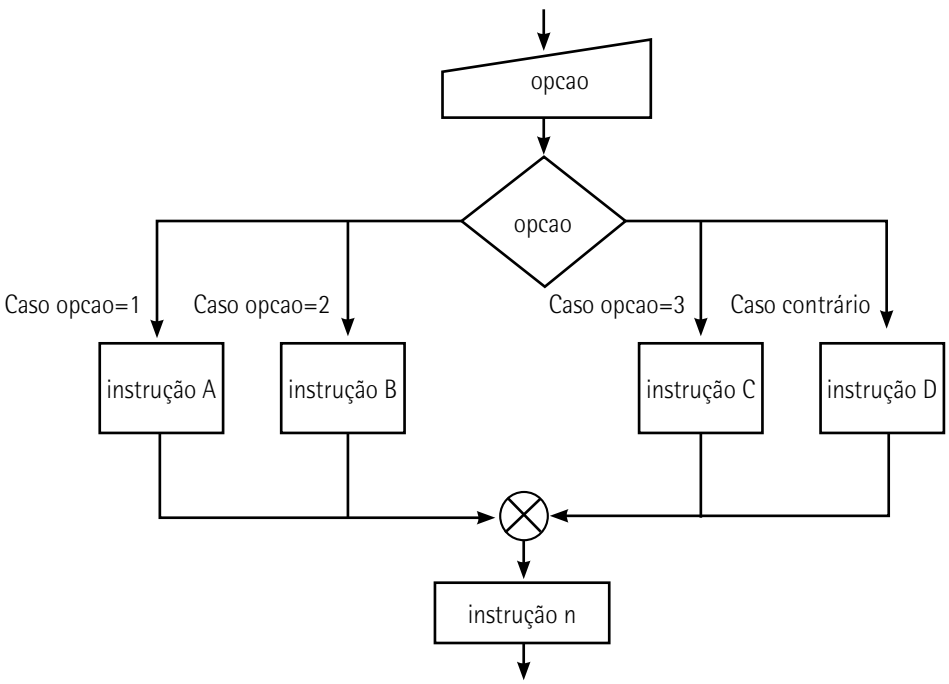

Figura 73 – Fluxograma da estrutura de decisão ESCOLHA-CASO

3.5 Estrutura de decisão encadeada ESCOLHA–CASO

A estrutura de decisão ESCOLHA-CASO encadeada é adequada para decidir com base em dados relacionados. O encadeamento se dá pela inserção de uma estrutura ESCOLHA-CASO dentro de outra, ou seja, o algoritmo dentro de um caso consiste de uma outra estrutura ESCOLHA. Poderão ser encadeadas múltiplas estruturas, a depender do contexto do problema a ser resolvido. A próxima figura mostra a sintaxe da estrutura escolha encadeada.

```
escolha (<variavel>)
        caso <valor1>
          escolha (<variavel>)
                  caso <valor1>
                          escolha (<variavel>)
                                  caso <valor1>
                                          <comandos>
                                  caso <valor2>, <valor4>
                                          <comandos>
                                  caso <valor5>
                                          <comandos>
                                  outrocaso
                                          <comandos>
                          fimescolha
                  caso <valor2>, <valor3>, <valor4>
                          <comandos>
                  caso <valor5>
                          <comandos>
                  outrocaso
                          <comandos>
          fimescolha
        caso <valor2>, <valor3>, <valor4>
          escolha (<variavel>)
                  caso <valor1>
                          <comandos>
                  caso <valor2>, <valor3>, <valor4>
                          <comandos>
                  caso <valor5>
                          <comandos>
                  outrocaso
                          <comandos>
          fimescolha
        caso <valor5>
          <comandos>
        outrocaso
          <comandos>
Fimescolha
```

Figura 74 – Pseudocódigo para a estrutura de decisão aninhada ESCOLHA-CASO

Exemplo

Desenvolva um algoritmo que receba a idade e o peso de uma pessoa, verifique e mostre em qual grupo a pessoa está, conforme o quadro a seguir:

Quadro 18

Idade	Peso		
	Até 60 kg	Entre 61 kg e 90 kg	Acima de 60 kg
0 a 17 anos	Grupo A	Grupo B	Grupo C
18 a 59 anos	Grupo D	Grupo E	Grupo F
Acima de 60 anos	Grupo G	Grupo H	Grupo I

Solução

```
1.  Algoritmo "Grupo de Risco"
2.  Var
3.     grupo : caracter
4.     idade, peso : inteiro
5.  Inicio
6.     //entrada
7.     escreva("Idade.....: ")
8.     leia(idade)
9.
10.    escreva("Peso (parte inteira).....: ")
11.    leia(peso)
12.
13.    //processamento e saída
14.    escolha (idade)
15.       caso 0 .. 17
16.          escolha (peso)
17.                caso 0 .. 60
18.                   escreva("Grupo A")
19.                caso 61 .. 90
20.                   escreva("Grupo B")
21.                outrocaso
22.                   escreva("Grupo C")
23.          fimescolha
24.       caso 18 .. 59
25.          escolha (peso)
26.                caso 0 .. 60
27.                   escreva("Grupo D")
28.                caso 61 .. 90
29.                   escreva("Grupo E")
30.                outrocaso
31.                   escreva("Grupo F")
32.          fimescolha
33.       outrocaso
34.          escolha (peso)
35.                caso 0 .. 60
36.                   escreva("Grupo G")
37.                caso 61 .. 90
38.                   escreva("Grupo H")
39.                outrocaso
40.                   escreva("Grupo I")
41.          fimescolha
42.       fimescolha
43. Fimalgoritmo
```

Figura 75 – Algoritmo baseado na estrutura de decisão ESCOLHA-CASO para decidir o grupo de risco de uma pessoa

Saiba mais

Uma excelente fonte de consulta sobre o tema de estruturas de decisão é a obra a seguir:

MANZANO, J. A. N. G.; OLIVEIRA, J. F. de. *Algoritmos*: lógica para desenvolvimento de programação de computadores. 28. ed. São Paulo: Érica, 2016.

3.6 Algoritmos de decisão

Estruturas são logicamente sequenciais e em algumas situações a sequência precisa sofrer desvios de acordo com alguma decisão sobre qual a sequência correta de comandos que deve ser executada para a resolução de um problema. A partir de agora apresentaremos exemplos de algoritmos que aplicam estruturas de decisão estudadas até aqui neste livro-texto.

Exemplo 1

Escreva um algoritmo para calcular a diferença entre dois números. O algoritmo deve receber dois números inteiros e apresentar a diferença do maior pelo menor.

Solução

```
1.  Algoritmo "Diferença de 2 números"
2.  Var
3.     x, y : inteiro
4.  Inicio
5.        //entrada
6.        escreva("X = ")
7.        leia(x)
8.        escreva("Y = ")
9.        leia(y)
10.
11.       se (x>y) entao
12.          //Saída e processamento.
13.          escreva("O resultado de ",x," - ",y," é ",(x-y))
14.       senão
15.          //Saída e processamento.
16.          escreva("O resultado de ",y," - ",x," é ",(y-x))
17.       fimse
18. Fimalgoritmo
```

Figura 76 – Algoritmo para calcular a diferença entre dois números

No exemplo da figura anterior, uma condição é aplicada para decidir se um valor de entrada é maior do que o outro. Então, na linha 3, declaram-se duas variáveis que serão usadas como entradas de valores, nas linhas 6 e 8 exibe-se mensagem ao usuário para que ele informe o valor de cada variável e nas linhas 7 e 9 os valores são capturados e atribuídos às variáveis **x** e **y**, respectivamente.

Após a leitura das variáveis, a condição **x>y** é explicitada assim: sendo verdadeira, a operação **x-y** será executada, caso contrário, a operação **y-x** será executada e então o resultado será exibido de acordo com a operação executada.

Exemplo 2

Escreva um algoritmo que leia três valores e determine o maior entre os três números. Esse algoritmo deve receber três entradas diferentes, decidir e apresentar qual é o maior valor dentre as três.

Solução

```
1.  Algoritmo "MAIOR DE 3 NÚMEROS"
2.  Var
3.    N1, N2, N3: inteiro
4.
5.  Inicio
6.    //entrada
7.    escreva("N1 = ")
8.    leia(N1)
9.    escreva("N2 = ")
10.   leia(N2)
11.   escreva("N3 = ")
12.   leia(N3)
13.
14. se ((N1>N2) e (N1>N3)) então
15.       //Saída e processamento.
16.       escreva("O maior entre: ",N1,", ", N2, " e ", N3, " é: ", N1)
17.   senão
18.       se ((N2>N1) e (N2>N3)) então
19.           //Saída e processamento.
20.           escreva("O maior entre: ",N1,", ", N2, " e ", N3, " é: ", N2)
21.       senão
22.           //Saída e processamento.
23.           escreva("O maior entre: ",N1,", ", N2, " e ", N3, " é: ", N3)
24.       fimse
25.   fimse
26. Fimalgoritmo
```

Figura 77 – Algoritmo para determinar o maior valor entre três números

No exemplo da figura anterior, três variáveis do tipo inteiro foram declaradas para capturarem os valores de entrada. Os comandos de entrada nas linhas 8, 10 e 12 esperam que o usuário digite os valores para serem armazenados nas variáveis **N1**, **N2** e **N3**. Na linha 14, a condição é encadeada para decidir qual é o maior valor entre os três valores de entrada.

Neste exemplo, a condição é aplicada usando uma expressão relacional e lógica. Na linha 14, é verificado se N1 é o maior de todos. Na linha 18, é verificado se N2 é o maior de todos. E por serem

3 números, se nem o primeiro nem o segundo forem o maior, por dedução o maior é o terceiro N3 e, por isso, não precisa ser testado. Para cada condição verificada, é executado um comando de saída informando qual é o maior valor. Por estar dentro de uma estrutura de decisão, quando uma condição é verificada como verdadeira, as demais não são executadas.

Exemplo 3

Escreva um algoritmo que informe o dia da semana e se há aula nesse dia a partir de uma entrada.

Solução

```
1.  Algoritmo "Dias de Aula"
2.  Var
3.    dia_da_semana : inteiro
4.  Inicio
5.    escreva("*================================ *")
6.    escreva("*     DIA DA SEMANA                *")
7.    escreva("* [1] Domingo                     *")
8.    escreva("* [2] Segunda-feira               *")
9.    escreva("* [3] Terça-feira                 *")
10.   escreva("* [4] Quarta-feira                *")
11.   escreva("* [5] Quinta-feira                *")
12.   escreva("* [6] Sexta-feira                 *")
13.   escreva("* [7] Sábado                      *")
14.   escreva("*================================ *")
15.   leia(dia_da_semana)
16.   escolha(dia_da_semana)
17.     caso 1
18.         escreva("Domingo não tem aula")
19.     caso 2
20.         escreva("Aulas da Segunda-feira")
21.         escreva("Lógica de Programação e Algoritmos")
22.     caso 3
23.         escreva("Aulas da Terça-feira")
24.         escreva("Lógica Matemática")
25.     caso 4
26.         escreva("Aulas da Quarta-feira")
27.         escreva("Interação Humano-Computador")
28.     caso 5
29.         escreva("Aulas da Quinta-feira")
30.         escreva("Introdução à Programação Estruturada")
31.     caso 6
32.         escreva("Aulas da Sexta-feira")
33.         escreva("Progr. Dispositivos Móveis")
34.     caso 7
35.         escreva("Aulas do Sábado")
36.         escreva("autoestudo")
37.     outrocaso
38.         escreva("Dia inválido!")
39.   fimescolha
40. Fimalgoritmo
```

Figura 78 – Algoritmo para determinar o dia da semana e a respectiva aula

No exemplo da figura anterior, um menu será exibido ao usuário e uma pausa será dada quando a linha 15 for executada, aguardando o usuário digitar um número à sua escolha. A ideia do algoritmo é que o usuário consulte qual é a aula do dia da semana informado.

A decisão é definida pela estrutura ESCOLHA-CASO, a qual receberá o dia informado na variável **dia_da_semana** e escolherá qual o caso aplicável. Como a decisão envolve sete possibilidades de dias da semana, estruturar o algoritmo usando a estrutura SE... SENÃO encadeada tornaria o código mais extenso e complexo.

A estrutura ESCOLHA... CASO para este problema deixa muito claro o que será exibido dependendo do dia da semana escolhido. Cada caso possui um algoritmo e o código fica mais bem organizado.

Neste exemplo, na linha 3 a variável **dia_da_semana** é declarada. Na linha 15 é lida e, a partir da linha 16, inicia a decisão a partir da escolha do valor para a variável **dia_da_semana**.

Para cada valor que a variável **dia_da_semana** poderá assumir, há um caso a ser verificado e para cada caso, um algoritmo. Para o caso de o usuário digitar um valor que não foi tratado nos casos da estrutura ESCOLHA... CASO será exibida a mensagem **Dia Inválido!**.

Exemplo 4

Escreva um algoritmo que receba a idade e o grupo de uma pessoa e mostre o nome das pessoas de cada grupo e em qual faixa etária estão, conforme o quadro a seguir:

Quadro 19

	17 a 20 anos	21 a 24 anos
A	João, José, Joaquim	Pedro e Carlos
B	Maria, Joana e Josefa	Catarina e Sofia
Outros	Não há	Não há

Solução

```
1.  Algoritmo "Escolha-Caso Encadeado"
2.  Var
3.    idade: inteiro
4.    grupo: caractere
5.  Inicio
6.      //entrada
7.      escreva("Digite sua idade: ")
8.      leia(idade)
9.
10.     //processamento e saida
11.     escolha(idade)
12.           caso 17, 18, 19, 20
13.             escreva("Digite seu grupo: ")
14.             leia(grupo)
15.            escolha(grupo)
16.           caso "A"
17.                           escreva("João, José, Joaquim")
18.           caso "B"
19.                           escreva("Maria, Joana e Josefa")
20.           outrocaso
21.                            escreva("Não há pessoas entre 17 e 20 anos em outros grupos")
22.             fimescolha
23.       caso 21, 22, 23, 24
24.             escreva("Digite seu sexo: ")
25.             leia(grupo)
26.             escolha(grupo)
27.             caso "A"
28.                           escreva("Pedro e Carlos")
29.           caso "B"
30.                           escreva("Catarina e Sofia")
31.           outrocaso
32.             escreva("Não há pessoas entre 21 e 24 anos em outros grupos")
33.             fimescolha
34.           outrocaso
35.           escreva("Não há alunos nesta faixa etária!")
36.     fimescolha
37. Fimalgoritmo
```

Figura 79 – Algoritmo para calcular o valor de delta

Neste exemplo de estrutura escolha encadeada, as variáveis idade e grupo controlam as estruturas ESCOLHA... CASO externa e interna. Para cada faixa etária, a variável grupo é lida. A decisão consiste em combinar idade e grupo pra mostrar quem são as pessoas do respectivo grupo.

Exemplo 5

Escreva um algoritmo que permita escolher quais cálculos das disciplinas Matemática e Física efetuar. 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, calcular também a velocidade média.

Solução

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

```
49.         escreva("|                                                    |")
50.         escreva("------------------------------------------------ ")
51.         escreva("")
52.         escreva(" Opção escolhida: ")
53.         leia(opmat)
54.         escolha(opmat)
55.           caso 1
56.             escreva("Digite o valor da aresta: ")
57.             leia(aresta)
58.             q <- aresta * aresta
59.             escreva("A área do quadrado é: ", q)
60.
61.           caso 2
62.             escreva("Digite o valor da base: ")
63.             leia(base)
64.             escreva("Digite o valor da altura: ")
65.             leia(altura)
66.             r <- base * altura
67.             escreva("A área do retângulo é: ", r)
68.
69.           caso 3
70.             escreva("Para calcular a equação de 1ºGrau, considere: y=a*i+b")
71.             escreva("Digite o valor de A...: ")
72.             leia(a1)
73.             escreva("Digite o valor de B...: ")
74.             leia(b1)
75.             escreva("Digite o valor de I...: ")
76.             leia(i1)
77.             eq1 <- a1*i1+b1
78.             escreva("O valor de y é:", eq1)
79.
80.           caso 4
81.             escreva("Programa para calcular f(x)=ax2+bx+c")
82.             escreva("a: ")
83.             leia(a)
84.             escreva("b: ")
85.             leia(b)
86.             escreva("c: ")
87.             leia(c)
88.             delta <- b*b-4*a*c
89.             se (delta>=0) então
90.               x1 <- (-1*b+raiz(delta))/(2*a)
91.               x2 <- (-1*b-raiz(delta))/(2*a)
92.               escreva("A raiz 1 é ",x1,"e a raiz 2 é ",x2)
93.             senão
94.               escreva("Não existem raízes reais")
95.             fimse
96.
97.           outrocaso
98.             escreva("-------------------------------------------")
99.             escreva("|                                           |")
100.              escreva("|    A opção escolhida não existe!          |")
101.              escreva("|                                           |")
102.              escreva("-------------------------------------------")
103.
104.          fimescolha
105.
106.      caso 2
107.        escreva("------------------------------------------------- ")
108.        escreva("|                                                 |")
109.        escreva("|         Calculadora - Física                    |")
110.        escreva("|                                                 |")
111.        escreva("------------------------------------------------- ")
112.        escreva("|                                                 |")
113.        escreva("| Qual tipo de cálculo deseja realizar?           |")
114.        escreva("| [1] Volume                                      |")
```

```
115.        escreva("| [2] Velocidade média                                  |")
116.        escreva("|                                                       |")
117.        escreva("-------------------------------------------------------")
118.        escreva("")
119.        escreva(" Opção escolhida: ")
120.        leia(opfis)
121.        escolha(opfis)
122.          caso 1
123.            escreva("-------------------------------------------------")
124.            escreva("|                                               |")
125.            escreva("|     Calculadora - Física - VOLUME             |")
126.            escreva("|                                               |")
127.            escreva("-------------------------------------------------")
128.            escreva("|                                               |")
129.            escreva("| Deseja calcular VOLUME de qual objeto?        |")
130.            escreva("| [1] Volume do um Paralelepípedo               |")
131.            escreva("| [2] Volume de um Cilindro                     |")
132.            escreva("|                                               |")
133.            escreva("-------------------------------------------------")
134.            escreva("")
135.            escreva(" Opção escolhida: ")
136.            leia(opfisvol)
137.            escolha(opfisvol)
138.              caso 1
139.                escreva("Comprimento do paralelepípedo (C): ")
140.                leia(C1)
141.                escreva("Largura do paralelepípedo.... (L): ")
142.                leia(L1)
143.                escreva("Altura do paralelepípedo..... (A): ")
144.                leia(A2)
145.                escreva("O Volume do Paralelepípedo é: ", C1*L1*A2)
146.              caso 2
147.                escreva("Informe o valor do RAIO: ")
148.                leia(raio)
149.                escreva("Informe o valor da ALTURA: ")
150.                leia(ALTURA1)
151.                volcil <- 3.14 * (raio * raio) * ALTURA1
152.                escreva("O Volume do Paralelepípedo é: ", volcil)
153.              outrocaso
154.                escreva("-------------------------------------------")
155.                escreva("|                                         |")
156.                escreva("|   A opção escolhida não existe          |")
157.                escreva("|                                         |")
158.                escreva("-------------------------------------------")
159.            fimescolha
160.
161.          caso 2
162.            escreva("Informe a distância percorrida (em KM): ")
163.            leia(dist)
164.            escreva("Informe o tempo gasto (em Horas): ")
165.            leia(tmp)
166.            vmedia <- dist/tmp
167.            escreva("A velocidade média é de: ", vmedia," Km/h")
168.
169.          outrocaso
170.            escreva("-------------------------------------------------")
171.            escreva("|                                               |")
172.            escreva("|     A opção escolhida não existe!             |")
173.            escreva("|                                               |")
174.            escreva("-------------------------------------------------")
175.
176.        fimescolha
177.
178.    outrocaso
179.        escreva("-------------------------------------------------------")
180.        escreva("|                                                       |")
```

```
181.        escreva("|    A opção escolhida não existe!                    |")
182.        escreva("|                                                      |")
183.        escreva("------------------------------------------------------ ")
184.
185.    fimescolha
186. Fimalgoritmo
```

Figura 80 – Algoritmo calculadora

No exemplo da figura anterior, foram implementados algoritmos já vistos anteriormente e novos adicionados a fim de explicar que o algoritmo pode ser grande e reunir várias funcionalidades. As variáveis foram incluídas no início do bloco entre as linhas 3 e 18.

4 ESTRUTURAS DE REPETIÇÃO

As estruturas de repetição são recursos para projetar algoritmos de modo a repetir um determinado bloco, evitando a repetição de códigos. Repetição de códigos torna o código vulnerável a defeitos, pois uma correção implementada num bloco de comandos poderá não ser replicada em outro bloco que faz a mesma coisa.

Uma estrutura de repetição também é conhecida como um laço ou um fluxo de controle que será repetido por um número indeterminado e finito de vezes, ou até que uma condição seja satisfeita. A condição é também chamada de teste e pode ser representada por uma expressão aritmética, relacional ou lógica.

Há três formas de estruturar os laços de repetição: com teste no início, no fim ou com controle do número de iterações, conforme ilustrado na figura a seguir:

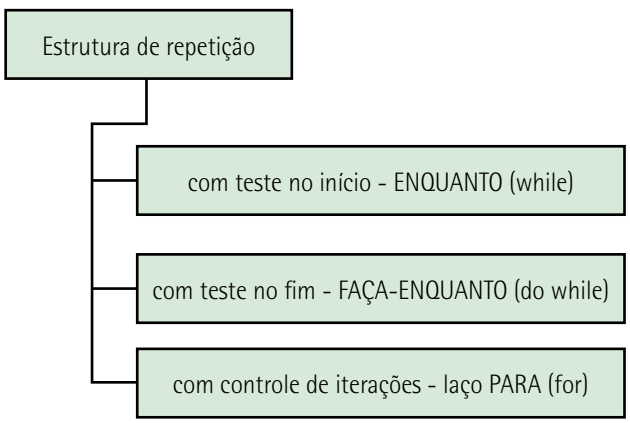

Figura 81 – Fluxograma da estrutura de repetição

4.1 Estrutura de repetição simples com teste no início (laço ENQUANTO)

A estrutura de repetição com teste no início estabelece o começo do laço com uma condição de teste. Se o teste resultar verdadeiro, então o bloco de comandos dentro do laço é executado e, ao término, a condição é novamente testada. Enquanto a condição for verdadeira, o bloco interno será repetido.

A figura a seguir apresenta a sintaxe da estrutura de repetição com teste no início (A) e o fluxograma da lógica (B). Primeiro a condição é testada e, se for verdadeira, o bloco de comandos que pertence a essa estrutura deve ser executado enquanto a condição permanecer verdadeira. A condição é testada antes da execução do bloco, o que significa que se a condição de teste for falsa, o algoritmo não executará o bloco e seguirá para a próxima instrução.

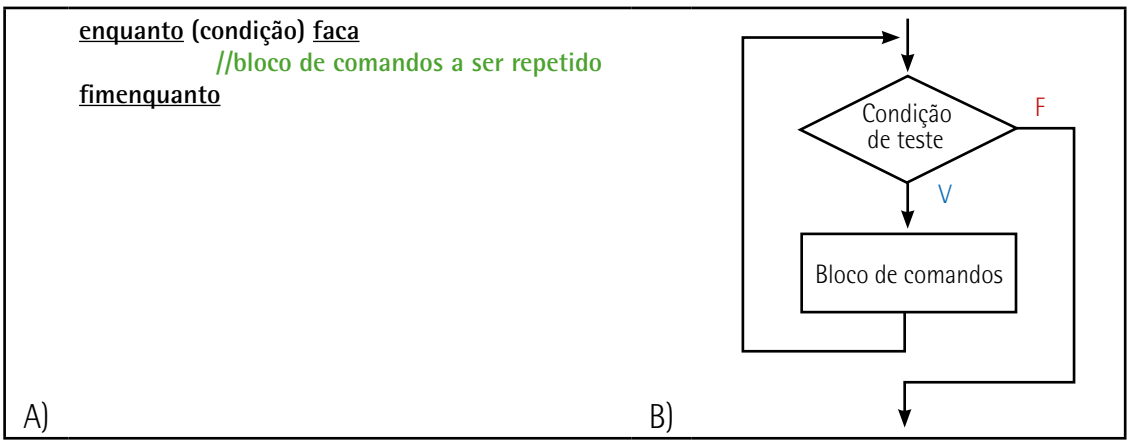

Figura 82 – Estrutura de repetição com teste no início: A) sintaxe do pseudocódigo; B) fluxograma da estrutura de repetição ENQUANTO... FAÇA

Um exemplo é apresentado na figura seguinte. O objetivo desse algoritmo é escrever os números do intervalo entre x e 10. A variável de entrada x é obtida pelo teclado, logo, não se sabe qual valor o usuário irá digitar.

Suponha que o usuário digite o número 5. Primeiro o programa testará se a condição (x<10) é satisfeita. Como 5 é menor do que 10, a condição é verdadeira e o bloco composto pelas linhas 8 e 9 será executado. Observe que o valor de x deve ser igual ou maior que 10 para que a condição seja falsa e a linha 11 seja executada. Para que o algoritmo não fique em um laço infinito, é necessário incrementar o valor de x dentro do bloco que será repetido.

Adicionalmente, se o usuário digitar um valor igual ou maior que 10, o bloco composto pelas linhas 8 e 9 nunca será executado.

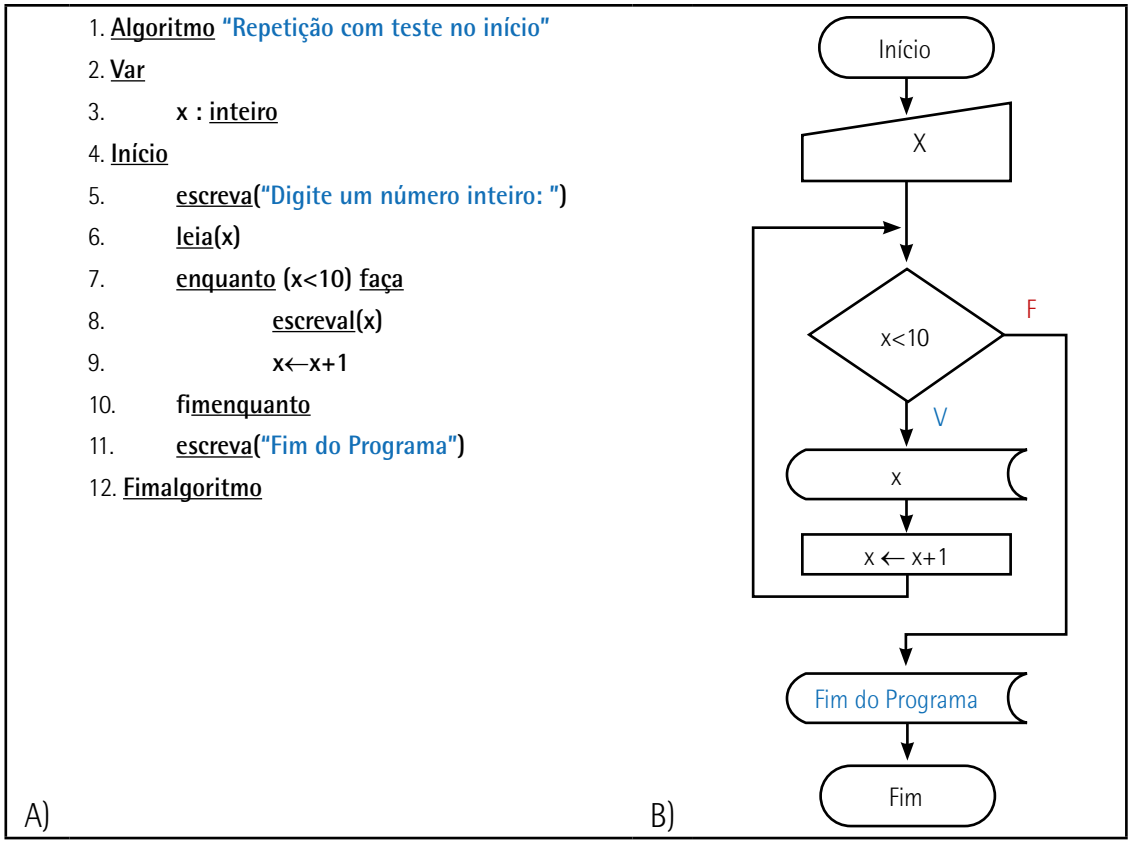

Figura 83 – Exemplo de estrutura de repetição com teste no início: A) pseudocódigo; B) fluxograma

No próximo exemplo, é apresentado o algoritmo que calcula a tabuada de um número inteiro.

Exemplo 1

A tabuada de um número é o produto desse número pelos inteiros do intervalo entre 1 e 10. Desse modo, o número digitado pelo usuário é fixo e a estrutura de repetição deverá controlar uma outra variável responsável por mudar o seu valor a cada linha. Neste exemplo, será computada a tabuada do n, logo, n vezes i, onde i corresponde aos valores de 1 a 10 para cada iteração do laço de repetição.

A descrição narrativa fica da seguinte forma:

```
Descrição narrativa
pergunte ao usuário: "Qual tabuada deseja calcular? "
leia a resposta do usuário.
inicialize i com 1
enquanto i <=10 repita
    escreva n * i = (n*i)
    incremente i
fim
```

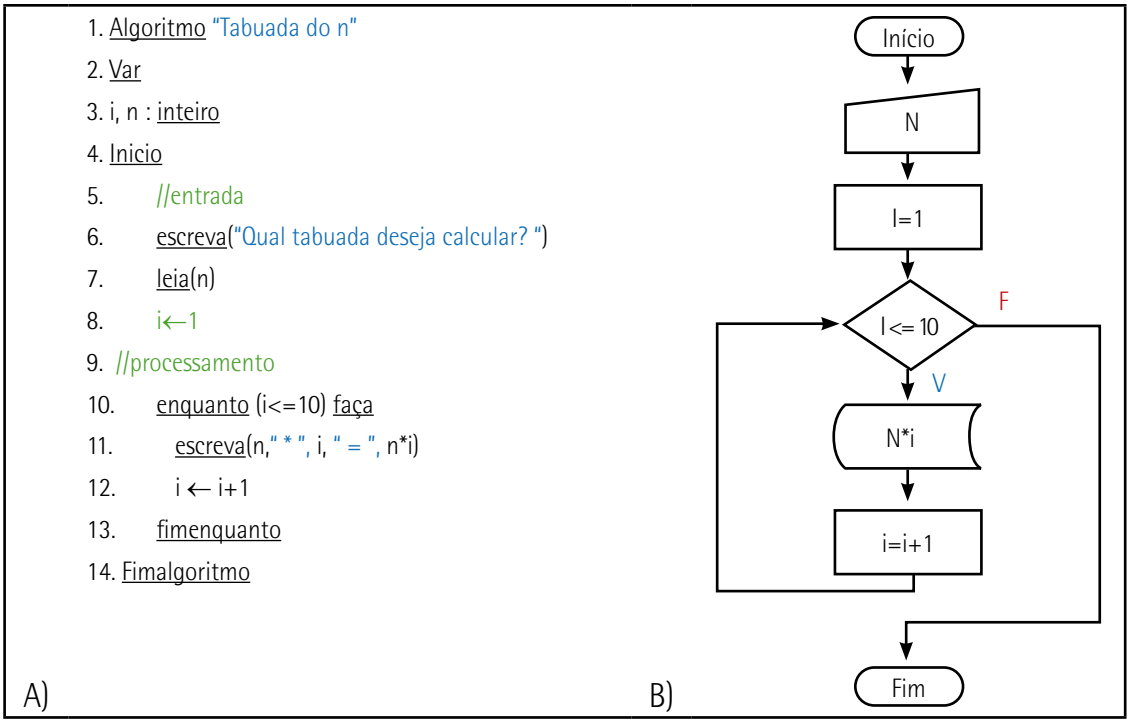

Figura 84 – Algoritmo para calcular e mostrar a tabuada do valor atribuído à variável n:
A) pseudocódigo; B) fluxograma

Um fator importante nos laços com teste no início é inicializar a variável que controla o laço de repetição e incrementá-la dentro do bloco para evitar laços infinitos – nesse caso, a variável **i**.

O próximo exemplo é o cálculo do fatorial de um número.

Exemplo 2

Por definição, o fatorial é um processo matemático definido como "o produto dos números inteiros positivos menores ou iguais a **n**. A fórmula seguinte demonstra essa definição.

$$n! = \prod_{k=1}^{n} k = n \times (n-1) \times (n-2) \times \ldots \times 3 \times 2 \times 1, \ \forall n \in \mathbb{N}$$

Para calcular o fatorial de um número inteiro **n**, duas lógicas podem ser aplicadas. A primeira é multiplicar o **n** por todos os valores menores até chegar ao elemento neutro da multiplicação, que é o 1. O fatorial de 5 será calculado da seguinte forma:

5! é igual a 5 x 4 x 3 x 2 x 1

ou seja, a ideia do algoritmo para obter o fatorial de **n** é multiplicar o **n** por n-1, n-2, até 1. Nessa lógica, a variável **n** é decrementada.

A segunda lógica é obter o fatorial de **n** multiplicando uma variável auxiliar **i** que será inicializada com 1 e incrementada até que seja igual a **n**, da seguinte forma:

5! = 1 x 2 x 3 x 4 x 5

A lógica é a maneira de raciocinar sobre o problema e diferentes algoritmos podem resolver o mesmo problema. No exemplo do fatorial, as duas lógicas conduzem ao cálculo correto. Quando você tiver desenvolvido habilidades de escrever algoritmos, poderá analisá-los a fim de identificar qual apresentará melhor desempenho em termos de tempo de processamento ou uso de memória.

O algoritmo para calcular o fatorial com a primeira lógica solicitará uma entrada numérica, executará a repetição enquanto a condição (n>1) for satisfeita. Durante a repetição, o número dado será multiplicado pelos seus anteriores e a variável de controle será decrementada. O algoritmo finaliza apresentando o fatorial. Se **n** for igual a 100, as linhas 14 e 15 serão repetidas 100 vezes.

A)

```
1. Algoritmo "Fatorial"
2. //declaração das variáveis
3. Var
4.       n, x, fatorial: inteiro;
5. Inicio
6. //entrada
7. escreval("Fatorial")
8. escreval("Digite um número: ")
9. leia(n)
10. fatorial ← 1
11. x <- n
12. //processamento
13. enquanto (n > 1) faça
14. fatorial←fatorial*n
15. n← n-1
16. fimenquanto
17.
18. //saída
19. escreva("O fatorial de ", x, " é ", fatorial)
20. Fimalgoritmo
```

B)

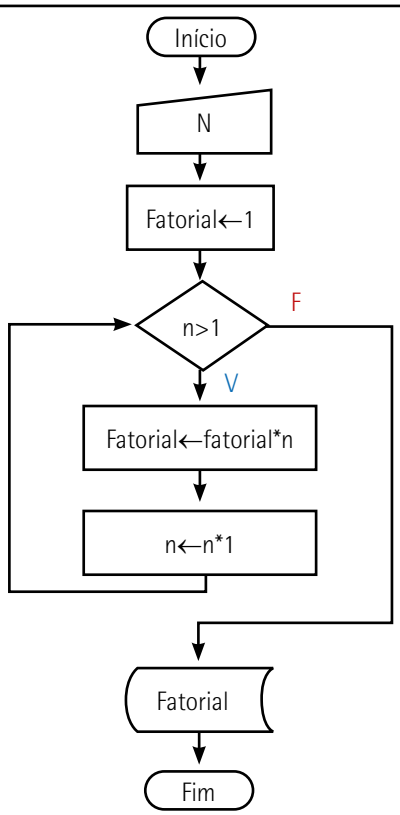

Figura 85 – Algoritmo para calcular o fatorial de n usando a estrutura ENQUANTO... FAÇA:
A) pseudocódigo que implementa o cálculo do fatorial na lógica: n! = n x (n-1) x (n-2) x ... (n-(n-2)) x (n-(n-1)); B) fluxograma

As estruturas de repetição e decisão podem ser combinadas num processo de solução de problemas. A seguir é apresentado outro exemplo que combina a estrutura de repetição com teste no início com as estruturas de decisão já estudadas na unidade 2.

Exemplo 3

Elabore um algoritmo que receba dois números inteiros, verifique qual é o maior entre eles, calcule e mostre o resultado do somatório dos números que sejam ímpares compreendidos entre esses dois números.

Solução

Se os números inteiros forem representados pelas variáveis x e y, por exemplo, se x receber 10 e y receber 20, o programa deverá somar os inteiros ímpares do intervalo entre 10 e 20, a saber: 11, 13, 15, 17 e 19.

Um aspecto a ser considerado é que o usuário poderá digitar um valor para x menor do que o valor de y, que corresponde ao fluxo normal de interpretação do problema. Contudo, o usuário também poderá digitar um valor para x maior do que para y e, nesse caso, o projeto de algoritmo deverá decidir qual é o menor e o maior dentre os dois números para encontrar os ímpares.

O algoritmo da figura seguinte decide primeiro qual dos números é o menor e o maior, e usa variáveis menor e maior para guardar. Enquanto a variável **menor** for menor do que a variável **maior**, então, decide se o número é ímpar. Se é ímpar, então soma-o.

A variável menor é incrementada na linha 29 e, após algumas repetições, o valor será maior que o da variável maior e o laço será interrompido. O comando de saída é instruído dentro de uma estrutura de decisão porque, nesse ponto do algoritmo, a variável menor já não tem mais o valor original obtido na estrutura de decisão das linhas 16 e 22.

```
1.  Algoritmo "Somatorio dos Impares entre dois Numeros"
2.
3.  Var
4.    x, y, menor, maior, soma : inteiro
5.
6.  Inicio
7.    //entrada
8.    escreva("Digite um valor....: ")
9.    leia(x)
10.   escreva("Digite outro valor.: ")
11.   leia(y)
12.
13.   //processamento
14.   soma <- 0
15.
16.   se (x<y) então
17.     menor<-x
18.     maior<-y
19.   senão
20.     menor<-y
21.     maior<-x
22.   fimse
23.
24.   x<-menor
25.   enquanto (menor <= maior) faca
26.     se (menor mod 2 <> 0) entao
27.       escreva(" + ", menor, " ")
28.       soma<-soma+menor
29.     fimse
30.     menor<-menor+1
31.   fimenquanto
32.
33.   //saida
34.   escreval(" ")
35.   escreva("Somatório dos ímpares entre ", x, " e ", menor, " é: ", soma)
36. Fimalgoritmo
```

Figura 86 – Algoritmo somatório dos ímpares entre dois números

A lógica desse algoritmo é também descrita no fluxograma da figura seguinte:

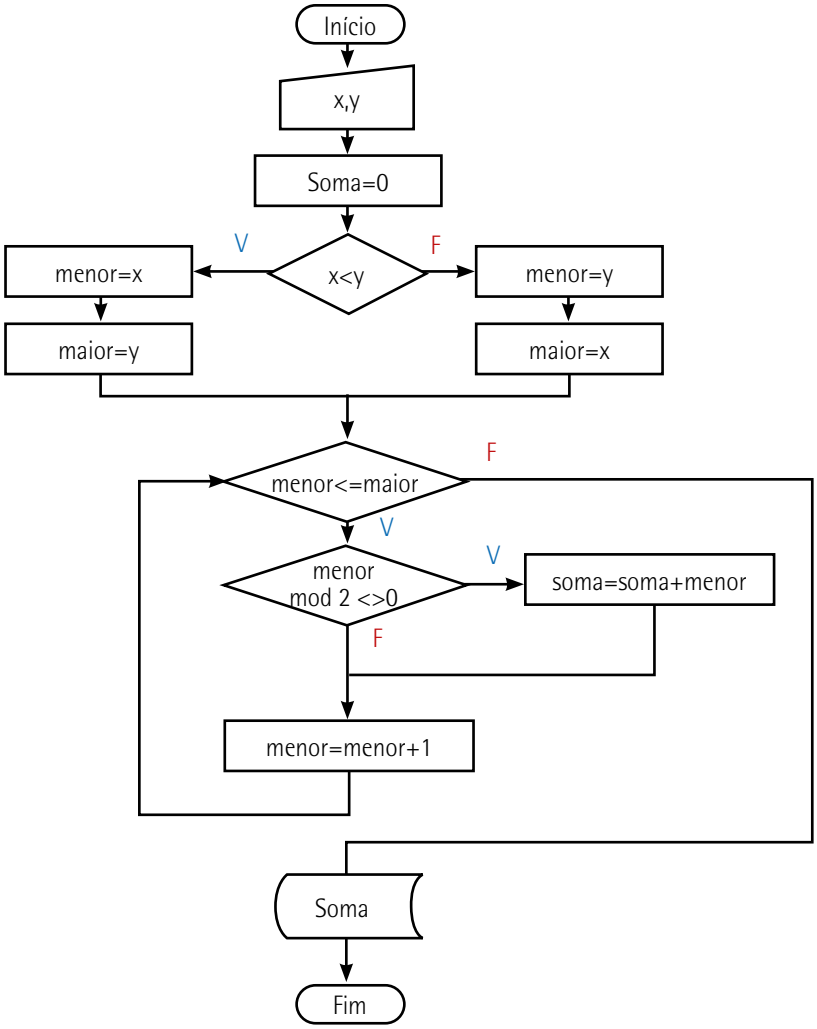

Figura 87 – Fluxograma somatório dos ímpares entre dois números

O algoritmo em narração descritiva é:

1. Leia dois números inteiros (x e y).
2. Se x for menor que y, então, menor recebe x e maior recebe y.
3. Senão menor recebe y e maior recebe x.
4. Inicialize a variável soma com zero
5. Enquanto a variável **menor** for menor que ou igual à variável **maior** faça.
6. Se o resto da divisão de **menor** por dois for diferente de zero (menor é ímpar) então
   a) **soma** recebe o **soma + menor**.
7. Incremente a variável menor (incrementar é somar 1).
8. Reteste a condição da linha 5 (**menor<=maior**) (linha 5) e, se for verdadeiro, repita.
9. Se x<y, então escreva somatório dos ímpares entre x e y é soma.
10. Senão escreva somatório dos ímpares entre y e x é soma.

4.1.1 Estrutura de repetição encadeada com teste no início

Similar ao encadeamento da estrutura condicional, a estrutura de repetição pode ser encadeada de modo a repetir blocos de comandos a partir de uma condição anterior. Também é possível criar condições em que se utilizem mais do que dois eixos de coordenadas, entretanto, à medida que o número de encadeamento aumenta, o algoritmo passa a ficar mais lento ou mais complexo.

A necessidade de projetar algoritmos com laços de repetição encadeados deve ser muito bem analisada, a fim de definir o método mais adequado a ser aplicado. Ao encadear uma estrutura de repetição, a primeira estrutura somente finalizará após a estrutura do bloco interno ser finalizada.

O algoritmo da figura seguinte possui dois laços de repetição com teste no início. O primeiro laço, também chamado de laço externo, é iniciado na linha 9 e controlado pela variável **i**. O segundo laço, chamado de laço interno, é iniciado na linha 11 e controlado pela variável **j**.

Para entrar no primeiro laço, a condição é i<10. Como a variável **i** é inicializada no início do algoritmo com 0 e o laço externo será repetido enquanto i<10, na primeira vez que a linha 9 for executada, a condição será verdadeira e o bloco interno será executado. Fazem parte do bloco os comandos entre as linhas 10 e 19. A variável **i** é incrementada na linha 18, assim, cada vez que o algoritmo encontra o **fimenquanto** na linha 20, ele volta para testar na linha 10 a condição de entrada na estrutura de repetição e, sendo esta verdadeira, repetir o bloco novamente.

Observação

A fim de evitar que o laço fique repetindo ininterruptamente, o valor da variável i deve ser alterado dentro do bloco.

De modo similar, o laço interno se repetirá dez vezes e, quando o laço interno for encerrado, o algoritmo estará preso no laço externo.

Para cada iteração que o laço externo executar, o laço interno repetirá dez vezes o bloco de comandos. Os laços encadeados desse algoritmo se repetirão 100 vezes. A figura a seguir (B) mostra o teste de mesa para as iterações desse algoritmo.

A)

```
1.  Algoritmo "Estrutura de Repetição Encadeada"
2.  Var
3.          //declaração das variáveis
4.  Inicio
5.    i←0
6.    j←0
7.    //...bloco de código
8.          //laço de repetição externo
9.    enquanto (i<10) faça
10.           //laço de repetição interno
11.     enquanto (j<10) faça
12.       //processamento
13.       //...bloco de código
14.       j←j+1
15.     fimenquanto
16.     //processamento
17.     //...bloco de código
18.     i←i+1
19.     j←0
20.   fimenquanto
21. Fimalgoritmo
```

B) teste de mesa

i	j	j<10	i<10
0	0	V	V
	1	V	
	2	V	
	3	V	
	:	:	
	:	:	
	9	V	
	10	F	
1	0	V	V
	1	V	
	2	V	
	:		
	9	V	
	10	F	
2	0	V	V
	1	V	
	2	V	
	:		
	10	F	
:			
:			
9	0	V	V
	1	V	
	2	V	
	:		
	10	F	
10	0		F

Figura 88 – A) algoritmo em pseudocódigo que aplica a estrutura de repetição encadeada; B) teste de mesa

Há várias aplicações para a estrutura de repetição encadeada. Similar ao algoritmo exemplificado na figura anterior, o exemplo de algoritmo da figura 90 mostra o pseudocódigo para calcular a tabuada do 1 ao 10.

A figura seguinte mostra que cada tabuada é controlada pela variável **n**, ou seja, quando **n** vale 1, calcula-se a tabuada do 1, quando **n** vale 2, calcula-se a tabuada do 2, e quando **n** vale 10, calcula-se a tabuada do 10.

A variável **m** controla o laço de repetição interno. Para cada n, **m** deve variar de 0 a 10 (os números em vermelho na tabuada). E quando encerrar a multiplicação de **n** por todos os **m**, a variável **m** deverá ser redefinida para zero a fim de computar a próxima tabuada com **m** variando novamente de 0 a 10.

n		m		n*m
1	x	0	=	0
1	x	1	=	1
1	x	2	=	2
1	x	3	=	3
1	x	4	=	4
1	x	5	=	5
1	x	6	=	6
1	x	7	=	7
1	x	8	=	8
1	x	9	=	9
1	x	10	=	10

n		m		n*m
2	x	0	=	0
2	x	1	=	2
2	x	2	=	4
2	x	3	=	6
2	x	4	=	8
2	x	5	=	10
2	x	6	=	12
2	x	7	=	14
2	x	8	=	16
2	x	9	=	18
2	x	10	=	20

n		m		n*m
3	x	0	=	0
3	x	1	=	3
3	x	2	=	6
3	x	3	=	9
3	x	4	=	12
3	x	5	=	15
3	x	6	=	18
3	x	7	=	21
3	x	8	=	24
3	x	9	=	27
3	x	10	=	30

.....

n		m		n*m
10	x	0	=	0
10	x	1	=	10
10	x	2	=	20
10	x	3	=	30
10	x	4	=	40
10	x	5	=	50
10	x	6	=	60
10	x	7	=	70
10	x	8	=	80
10	x	9	=	90
10	x	10	=	100

Figura 89 – Tabuada do 1 ao 10 – demonstração das variáveis n e m no laço de repetição encadeado

O algoritmo da próxima figura possui dois laços encadeados. O laço interno codificado nas linhas 11 a 13 controlará a variável **m** que, a cada iteração da variável **n**, será inicializada com 0 para processar a tabuada do próximo número. A variável **n**, controlada pelo laço externo, vai variar de 1 a 10. O teste de mesa representa o estado das variáveis durante a execução do algoritmo.

Exemplo de aplicação

```
1. Algoritmo "Tabuada do 1 ao 10"
2. Var
3. n, m : inteiro
4. Inicio
5.       //entrada
6.       escreva("Este programa calcula a tabuada do 1 ao 10 ")
7.       n←1
8.  //processamento
9.       enquanto (n<=10) faça
10.        m←0
11.        enquanto (m<=10) faça
12.                  escreva(n," * ", m, " = ", n*m)
13.                  m ← m+1
14.        fimenquanto
15.      fimenquanto
16. Fimalgoritmo
```

teste de mesa

n	m	n*m
1	0	
1	1	
1	2	
	:	
1	10	

2	0	
2	1	
2	2	
	:	
2	10	

10	0	
10	1	
10	2	
	:	
10	10	

Figura 90 – Algoritmo e teste de mesa para calcular e escrever a tabuada do 1 ao 10 – demonstração das variáveis n e m no laço de repetição encadeado

Um fator importante a ser observado nos laços de repetição é a variável que controla as iterações do laço. A atribuição m←0 na linha 10 torna a condição (m<=10) verdadeira e os comandos do laço interno são executados.

Observe que o comando da linha 13 é o incremento da variável m ←m+1, para que m possa variar e, quando valer 11, a condição de teste do laço retornará **falso** e o laço será interrompido.

4.2 Estrutura de repetição com teste no fim (laço FAÇA)

Diferentemente da estrutura de repetição com o teste no início, em que os processos são executados apenas quando a condição é satisfeita, na estrutura de repetição com teste no final o processo é executado pelo menos uma vez, independentemente de a condição ser ou não satisfeita.

Na lógica de programação, o teste no final pode ser interpretado como **FAÇA...ENQUANTO <condição>**, ou **REPITA... ATÉ <condição>**, conforme mostra a próxima figura:

Faça //bloco de comandos enquanto (<condição for verdadeira>)	repita //bloco de comandos até (condição seja verdadeira)

Figura 91 – Sintaxe da estrutura de decisão com teste no final: A) FAÇA... ENQUANTO; B) REPITA ... ATÉ

Ambas são interpretações corretas da mesma estrutura. As linguagens de programação implementam esses comandos como **DO... WHILE** ou **REPEAT... UNTIL**, embora nunca ambos.

Observação

As linguagens de programação derivadas da Linguagem C utilizam a lógica FAÇA... ENQUANTO (**DO... WHILE**).

Nas linguagens de programação derivadas da linguagem Pascal, tal como o Delphi, a estrutura de repetição com teste no início é o REPITA... ATÉ, ou REPEAT... UNTIL.

O comando funciona como uma palavra de ordem **Faca (execute) o bloco enquanto a condição for verdadeira**. A execução do bloco será realizada independentemente de a condição ser ou não verdadeira, visto que o teste só será realizado no final, após a execução do bloco.

Quando a condição for testada, se ela não for satisfeita, pelo menos uma vez o bloco de comandos terá sido executado. A figura seguinte apresenta a lógica de uma estrutura de repetição com teste no fim.

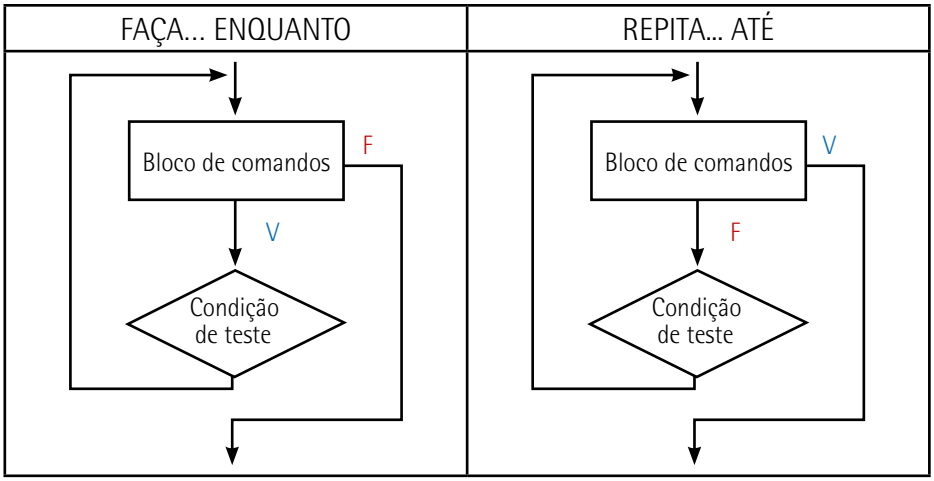

Figura 92 – Fluxograma do laço de repetição com teste no fim: A) FAÇA... ENQUANTO; B) REPITA... ATÉ

Em ambos os casos, a variável de controle é testada a cada iteração do laço. Dentro do laço, a variável precisa ser atualizada para evitar repetições infinitas. A diferença entre as duas lógicas de repetição com teste no fim está na interpretação da condição de teste e saída do laço.

O comando FAÇA... ENQUANTO executará o bloco de comandos e ao término, testará a condição. O bloco será executado enquanto a condição for verdadeira, ou seja, até que a condição seja falsa, continuará repetindo a execução.

O comando REPITA... ATÉ também executará o bloco de comandos e ao término testará a condição. O bloco será executado até que a condição seja verdadeira, ou seja, enquanto a condição não for verdadeira (ou seja, enquanto for falsa), o bloco será repetido. A figura 94 apresenta um algoritmo codificado nos dois laços de repetição.

Observação

O IDE VisuAlg, usado para testar os algoritmos deste livro-texto, aceita apenas o REPITA... ATÉ, assim, os algoritmos são exemplificados nesta lógica.

4.2.1 Estrutura de repetição encadeada com teste no fim

O encadeamento da estrutura de repetição com teste no fim pode ser realizado com outras estruturas de repetição com teste no fim, no início ou com o laço **PARA** que veremos mais adiante em nosso percurso de estudos neste livro-texto. Agora discutiremos apenas exemplos de encadeamento de estruturas de repetição encadeadas com teste no fim.

Exemplo

O objetivo deste exemplo é desenvolver um algoritmo que receba um número N, calcule e mostre o valor da seguinte série:

Série = 1 + 1/2 + 1/3 + ... + 1/N

```
1. Algoritmo "Série"
2. Var
3.       n , y : inteiro
4.       serie : real
5. Inicio
6.       //inicializacao das variaveis
7.
8.       y←1
9.       serie←0
10.      //entrada
11.      escreva("Digite um número inteiro: ")
12.      leia(n)
13.      escreva("S = ")
14.      //processamento
15.      repita
16. serie←serie+(1/y)
17. escreva("+", 1, "/", y)
18. y←y+1
19.      até(y>=n)
20.      //saida
21.      escreva(serie)
22. Fimalgoritmo
```

Figura 93 – Algoritmo em pseudocódigo que aplica estrutura de repetição com teste no fim para calcular a série = 1 + 1/2 + 1/3 + ... + 1/N

No pseudocódigo da figura anterior, o bloco entre as linhas 15 e 19 será repetido até que y seja igual a n. Mesmo que o n seja maior do que y na primeira iteração, o bloco será executado porque a condição é testada apenas no final.

Os exemplos de algoritmos da figura seguinte aplicam as duas lógicas de estrutura de repetição com teste no fim, a saber, REPITA... ATÉ e FAÇA ... ENQUANTO. Em ambos os algoritmos, o número de valores será informado pelo usuário e não é conhecido pelo algoritmo. Outra abordagem interessante do algoritmo da figura seguinte em relação ao exemplo da figura precedente é que a contagem das repetições é regressiva.

Os algoritmos da figura a seguir identificam o menor e o maior dentre os n números digitados. A variável n controla o número de iterações do laço e, de acordo com o enunciado, devem ser lidos dez números, e cada iteração do laço corresponde à execução do bloco entre as linhas 11 e 19.

No algoritmo **MenorMaior** da próxima figura, em que a lógica REPITA... ATÉ é aplicada, as linhas 11 a 18 serão repetidas até que a variável de controle n seja igual a 0. A variável n foi inicializada na linha 8 com o valor 10, e é decrementada na linha 17. Na primeira vez que a condição de teste é executada na linha 19, a variável n vale 9 e a condição de teste resulta **falso**, porque 9 não é igual a 0 e o bloco de comandos composto pelas linhas 11 a 18 é repetido. A repetição será recorrente até que a variável n seja igual a 0.

Observe que o mesmo problema é resolvido no algoritmo **MenorMaior2** da figura a seguir, em que a lógica FAÇA... ENQUANTO é aplicada: o bloco entre as linhas 11 e 19 é repetido enquanto n > 0. A leitura dessas estruturas deve ser feita da seguinte forma:

- repita até que a condição seja verdadeira;
- faça enquanto a condição é verdadeira.

Exemplo

Escreva um algoritmo para ler 10 números inteiros e apresentar o menor e o maior.

A)

```
1. Algoritmo "MenorMaior"
2. var
3. n, número, maior, menor : inteiro
4. Inicio
5. //entrada do primeiro valor
6. escreva("Digite um numero: ")
7. leia(numero)
8.
9. //inicialização das variáveis
10. maior ← numero
11. menor ← numero
12. n<-9
13. //processamento
14. repita
15. escreva("Digite um numero: ")
16. leia(numero)
17. se (numero>maior) então
18.    maior←numero
19. fimse
20. se (numero<menor) então
21.    menor←numero)
22. fimse
23. n←n-1
24. ate n=0
25. //saida
26. escreva("O maior número é ", maior)
27. escreva("O menor número é ", menor)
28. Fimalgoritmo
```

B)

```
1. Algoritmo "MenorMaior2"
2. var
3. n, numero, maior, menor : inteiro
4. Inicio
5. //entrada do primeiro valor
6. escreva("Digite um numero: ")
7. leia(numero)
8.
9. //inicialização das variáveis
10. maior<-numero
11. menor<-numero
12. n<-9
13. //processamento
14. faca
15. escreva("Digite um numero: ")
16. leia(numero)
17. se (numero>maior) então
18.      maior←numero
19. fimse
20. se (numero < menor) então
21.      menor←numero)
22. fimse
23. n←n-1
24. enquanto (n>0)
25. //saida
26. escreva("O maior número é ", maior)
27. escreva("O menor número é ", menor)
28. Fimalgoritmo
```

Figura 94 – Algoritmo aplicando o laço de repetição com teste no fim para calcular o maior e o menor dentre n números digitados: A) Estrutura REPITA... ATÉ; B) Estrutura FAÇA... ENQUANTO

O fluxograma da figura a seguir resolve o mesmo problema do pseudocódigo da figura anterior. A interpretação do código muda, apesar de a estrutura do algoritmo ser bem parecida, diferindo apenas na condição de teste da linha 19 da figura precedente.

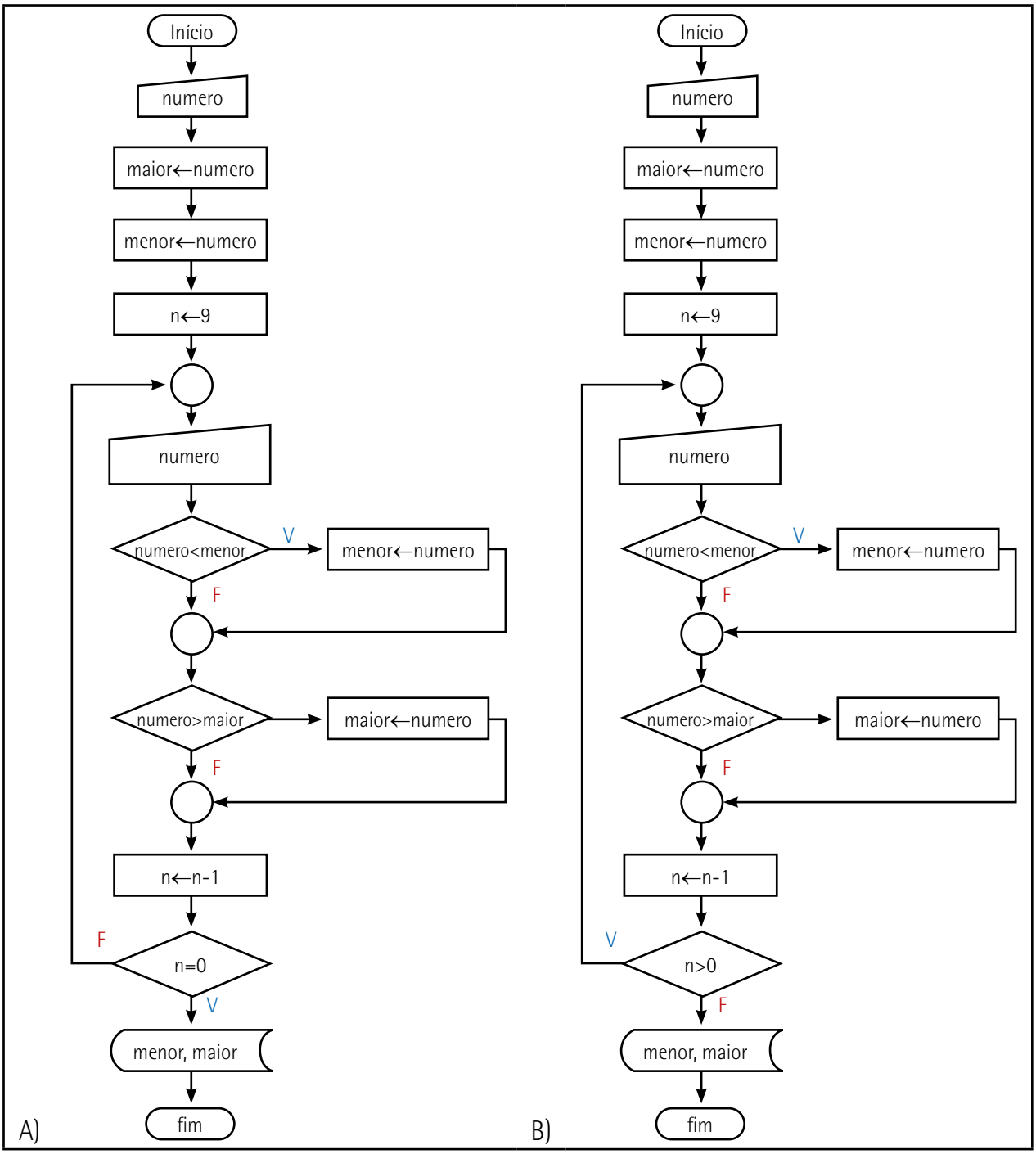

Figura 95 – Fluxograma aplicando o laço de repetição com teste no fim para calcular o maior e o menor dentre n números digitados: A) estrutura REPITA... ATÉ; B) estrutura FAÇA... ENQUANTO

Observação

Observe na parte A da figura anterior que o resultado falso para a primeira condição implica permanecer no bloco de repetição. Já a parte B da mesma figura mostra que quando a terceira condição retorna falso, o laço é interrompido.

Em narração descritiva, o algoritmo da figura anterior pode ser descrito da seguinte forma:

```
Leia o primeiro valor.
Inicialize as variáveis maior e menor com o primeiro número.
Inicialize a variável de controle n com 9, pois faltam 9 números para serem lidos.
Repita o bloco abaixo até que a variável de controle n seja igual a 0.
    Leia um número.
    Se numero>maior, então atribua o número à variável maior.
    Se número<menor, então atribua o número à variável menor.
    Atualize a variável de controle n decrementando-a. (n ← n-1).
Quando a variável de controle n for igual a 0, interrompa a repetição.
Escreva o maior e o menor número dentre os 10 digitados.
```

A fim de demonstrar que diferentes algoritmos podem ser soluções corretas para o mesmo problema, o exemplo da figura seguinte é o cálculo do fatorial de N.

Nesse procedimento, o algoritmo executará o cálculo do fatorial antes de testar a condição de laço, dessa forma, quando a condição for executada, um bloco de comandos já terá sido executado.

Uma característica importante da estrutura de repetição com teste no final é a garantia da primeira execução do programa, uma vez que algumas situações de linha de código ou de mensagem necessitam de dados ou informações para que possam realizar o procedimento subsequente, o que evita duplicidade de instrução. Vamos ao exemplo.

Exemplo

```
Algoritmo "Fatorial"
Var
        //declaracao das variaveis
        n, fatorial: inteiro
Inicio
        //entrada
        escreval("Fatorial")
        escreval("Informe o número a calcular: ")
        leia(n)
        fatorial ← 1

        //processamento
  repita
          fatorial ← fatorial*n
          n ← n-1
  até (n < 1)
        se (fatorial=0) então
          fatorial←1
        fimse
        //saida
        escreva("O valor do fatorial é: ", fatorial)

Fimalgoritmo
```

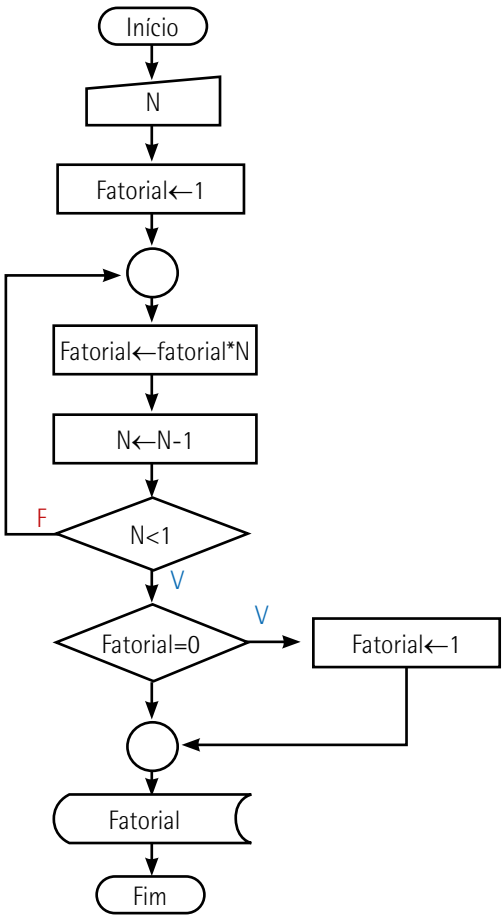

Figura 96 – Algoritmo em pseudocódigo e fluxograma para calcular o fatorial de n usando a estrutura de repetição simples com teste no fim

4.3 Estrutura de repetição com controle de iterações (laço PARA)

As estruturas de repetição com controle de iterações são usadas em algoritmos que necessitam executar blocos de códigos que deverão ser repetidos por um número específico de vezes.

A estrutura é organizada em três partes: declaração e inicialização da variável que controla o laço, uma condição de teste necessária para interromper o laço e a atualização da variável de controle.

Em pseudocódigo, a sintaxe é definida como mostrado na figura seguinte. O <valor> do passo é o incremento ou decremento da variável que controla o laço PARA.

```
para <variável> de <valor_inicial> até <valor_final> passo <valor> faça
        //bloco de comandos a ser repetido
fimpara
```

Figura 97 – Sintaxe do pseudocódigo da estrutura de repetição com controle de iterações – laço PARA

No quadro a seguir são apresentados exemplos de diferentes implementações do laço PARA. Dependendo do problema a ser resolvido, outras variações poderão ser aplicadas de acordo com a lógica estruturada para a resolução.

Quadro 20 – Exemplos de implementações do laço PARA

Exemplos de implementação do laço PARA	Explicação
para x de 1 até 10 passo 1 faça escreva(x, " ") fimpara	x é a variável que controla o laço, ela deve variar de 1 até 10 e o passo igual a 1 significa que x será incrementada de 1 em 1 O comando escreva será executado dez vezes, e escreverá 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
para x de 1 até 10 passo 2 faça escreva(x, " ") fimpara	x é a variável que controla o laço, ela deve variar de 1 até 10 e o passo igual a 1 significa que x será incrementada de 2 em 2 O comando escreva será executado cinco vezes e escreverá 1, 3, 5, 7, 9
para y de 10 até 1 passo -1 faça escreva(y, " ") fimpara	y é a variável que controla o laço, ela deve variar de 10 até 1 e o passo igual a -1 significa que y será decrementada de 1 em 1 O comando escreva será executado dez vezes e escreverá 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
para y de 12 até 1 passo -3 faça escreva(y, " ") fimpara	y é a variável que controla o laço, ela deve variar de 12 até 1 e o passo igual a -3, significa que y será decrementada de 3 em 3 O comando escreva será executado quatro vezes e escreverá 12, 9, 6, 3

O exemplo da figura seguinte é o algoritmo para calcular a tabuada de N, sendo que N será informado pelo usuário a partir de comando de entrada. As iterações da tabuada são as multiplicações de N pelos valores de 0 a 10. A variável que controla o laço de repetição do exemplo da figura anterior é a variável x, visto que está variando no intervalo de 0 a 10.

```
1.  Algoritmo "Tabuada de N usando o laço para"
2.  Var
3.    x, n : inteiro
4.  Inicio
5.    escreva("Digite um Número: ")
6.    leia(n)
7.
8.    para x de 0 ate 10 passo 1 faça
9.      escreval(n, " * ", x, " = ", n*x)
10.   fimpara
11. Fimalgoritmo
```

n		x		n*x
3	x	0	=	0
3	x	1	=	3
3	x	2	=	6
3	x	3	=	9
3	x	4	=	12
3	x	5	=	15
3	x	6	=	18
3	x	7	=	21
3	x	8	=	24
3	x	9	=	27
3	x	10	=	30

Figura 98 – Algoritmo para calcular a tabuada usando o laço PARA

A implementação do laço PARA condensa em uma única linha a lógica das outras estruturas de repetição vistas anteriormente, ficando a cargo da estrutura a atualização da variável que controla o laço.

O algoritmo para calcular o enésimo número da série de Fibonacci usando o laço PARA é implementado no exemplo da figura seguinte e a lógica é apresentada no fluxograma da figura 100. A sequência é dada pela soma dos dois números anteriores e assim sucessivamente até que seja atingido o número desejado.

Por definição:

fib(0) = 0

fib(1) = 1

fib(n) = fib(n-1) + fib(n-2)

para qualquer natural n > 1.

Na sequência de Fibonacci, quando n é 10, fib(10) será 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, pois a contagem dos termos da sequência se inicia em 0.

A descrição narrativa é a seguinte:

```
Leia n
Se n=0 então escreva Fib(0) = 0
Se n=1 então escreva Fib(0) = 1
Senão, escreva fib(n) = fib(n-1) + fib(n-2)
```

Generalizando a fórmula fib(n) = fib(n-1) + fib(n-2) em variáveis, tem-se as variáveis a, b e aux, responsáveis por computar os números da série. O objetivo do algoritmo é mostrar a sequência de Fibonacci e informar o valor do enésimo termo da série. O algoritmo da figura seguinte considera Fib(0) = 0 o primeiro termo da série, então Fib(10) retornará o décimo primeiro elemento da série.

Exemplo

A figura a seguir traz um algoritmo em pseudocódigo para calcular o enésimo termo da sequência de Fibonacci usando o laço PARA:

```
1.  Algoritmo "Fibonacci"
2.
3.  Var
4.    a, b, aux, i, n : inteiro
5.  Inicio
6.    //inicializando as variáveis a e b
7.    a<-0
8.    b<-1
9.
10.   //entrada
11.         escreval("Série de Fibonacci Fib(n)")
12.         escreva("Informe o número: ")
13.         leia(n)
14.
15.   //processamento
16.   se (n=0) então
17.     escreva("Fib(",n,") = ", a)
18.   senão
19.     se (n=1) então
20.       escreva("Fib(",n,") = ", a, ", ", b)
21.     senão
22.       escreva("Fib(",n,") = ", a)
23.       para i<-2 ate n passo 1 faca
24.         aux<-a+b
25.         a<-b
26.         b<-aux
27.         escreva(" ", a)
28.       fimpara
29. //saída
30.         escreval(" ")
31.         escreval("Fib(",n,")",a)
32.     fimse
33.   fimse
34. Fimalgoritmo
```

Figura 99

O fluxograma correspondente ao exemplo da figura anterior é apresentado na figura seguinte. Os objetos do fluxograma destacados em amarelo expressam a estrutura do laço PARA. Na estrutura do laço PARA em pseudocódigo, esses três comandos estão descritos na sintaxe do laço:

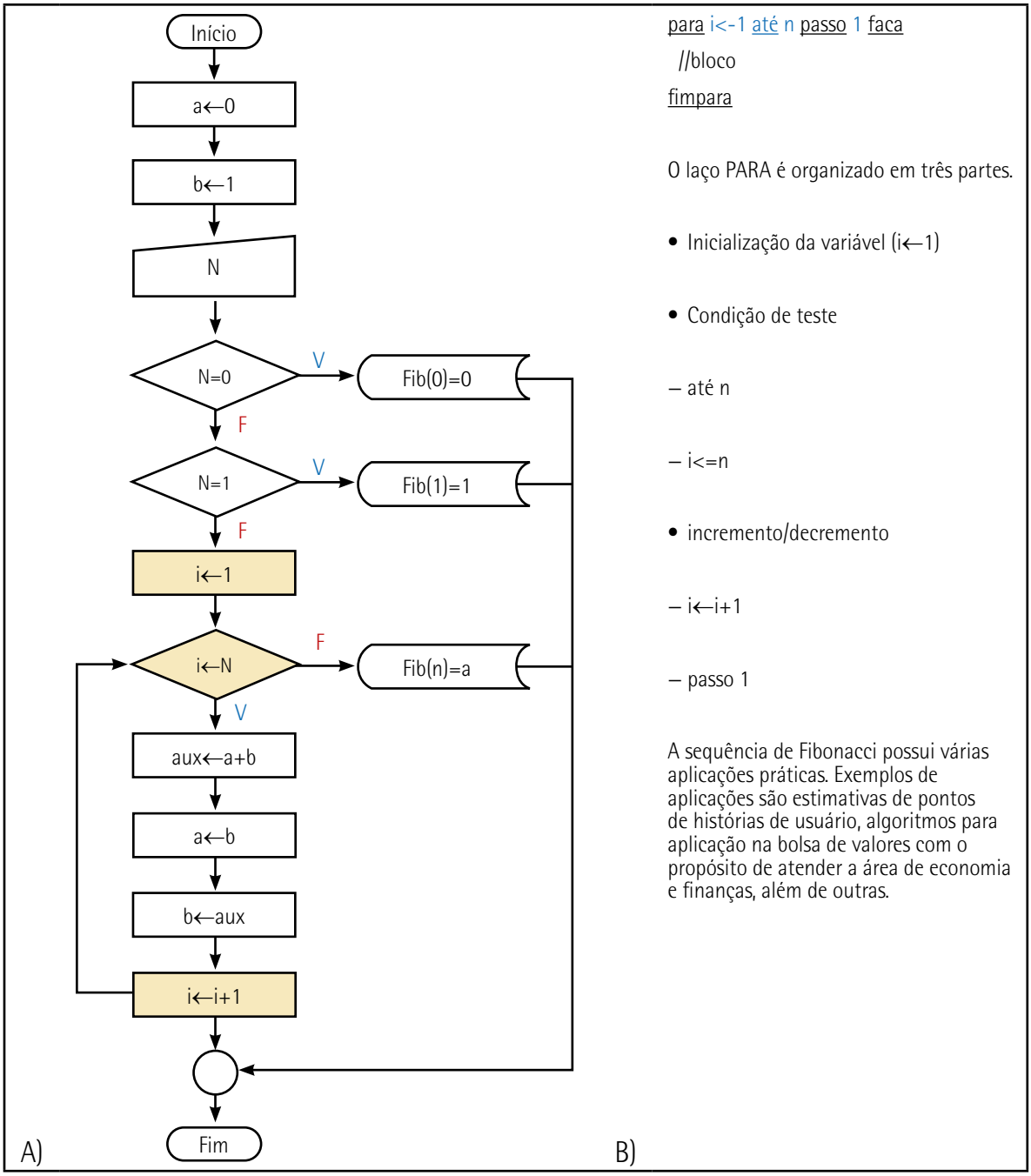

Figura 100 – Fluxograma para resolver o problema da série de Fibonacci

Saiba mais

Leia mais sobre estruturas de repetição no capítulo:

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

4.4 Algoritmos com estruturas de repetição

Algoritmos são projetados com estruturas de repetição a fim de evitar a duplicidade de códigos. O bloco será repetido enquanto a condição for satisfeita. Essas estruturas reduzem a quantidade de linhas de código, organizam o algoritmo e facilitam a sua interpretação do algoritmo. Trabalharemos a partir deste ponto com exemplos de algoritmos com laços de repetição que aplicam os conceitos que estudamos até agora.

Exemplo 1

Elabore um algoritmo que efetue a soma de números ímpares entre um dado intervalo informado pelo usuário.

Solução

```
1.    Algoritmo "Soma entre números ímpares"
2.    Var
3.      x, y, menor, maior, soma : inteiro
4.
5.    Inicio
6.      //entrada
7.      escreva("Digite um valor....: ")
8.      leia(x)
9.      escreva("Digite outro valor.: ")
10.     leia(y)
11.
12.     //processamento
13.     soma <- 0
14.
15.     se ( x > y ) entao
16.       maior <- x
17.       menor <- y
18.     senão
19.       maior <- y
20.       menor <- x
21.     fimse
22.
23.     enquanto (menor <= maior) faca
24.       se (menor mod 2 <> 0) entao
25.         escreva(" + ", menor, " ")
26.         soma <- soma+menor
27.       fimse
28.       menor <- menor+1
29.     fimenquanto
30.
31.     //saida
32.     escreva(" ")
33.     se (x<y) entao
34.       escreva("Somatório dos ímpares entre ", x, " e ", y, " é: ", soma)
35.     senão
36.       escreva("Somatório dos ímpares entre ", y, " e ", x, " é: ", soma)
37.     fimse
38.   Fimalgoritmo
```

Figura 101 – Algoritmo soma dos números ímpares

O propósito do algoritmo apresentado é somar números ímpares entre um intervalo de números dados pelo usuário. Como o usuário pode informar um número maior primeiro e em seguida um número menor, uma condição foi estabelecida para determinar qual dos dois números é o menor e qual é o maior, entre as linhas 15 e 21. A partir da linha 23 até a linha 29, o comando de laço de repetição ENQUANTO é executado até que o menor se iguale ao maior.

Exemplo 2

Elabore um algoritmo para calcular a tabuada de um número informado pelo usuário.

Solução

```
1.  Algoritmo "Tabuada de N"
2.  Var
3.     i, n : inteiro
4.  Inicio
5.     //entrada
6.     escreva("Qual número deseja calcular a Tabuada?: ")
7.     leia(n)
8.     i<-0
9.     //processamento
10.    faca
11.       escreva(n," * ", i, " = ", n*i)
12.            i<-i+1
13.    enquanto (i<10)
14. Fimalgoritmo
```

Figura 102 – Algoritmo tabuada de N

Neste exemplo, foi utilizado o laço de repetição com teste no fim, nesse caso, o laço inicia sua execução, na linha 10, executa os comandos pertencentes ao laço e finaliza executando a condição. Caso a condição seja **falsa**, é dada a saída do laço.

Neste caso, os comandos do laço são executados pelo menos uma vez, diferentemente de quando a condição é executada no início de um laço.

Exemplo 3

Elabore um algoritmo para calcular a tabuada de um número informado pelo usuário até um determinado valor informado por ele também. O algoritmo deve continuar em execução até o usuário solicitar que pare.

Solução

```
1.  Algoritmo "Tabuada de N limitada pelo usuário"
2.  Var
3.    n, inicial, final : inteiro
4.    opcao : caractere
5.  Inicio
6.    faca
7.      //entrada
8.      escreva("Qual número deseja calcular a ?: ")
9.      leia(n)
10.     escreva("Até qual valor? ")
11.     leia(final)
12.     inicial<-0
13.     //processamento
14.     faca
15.       escreva(n," * ", inicial, " = ", n*inicial)
16.           inicial<-inicial+1
17.     enquanto (inicial<=final)
18.     escreva("Deseja continuar? [s] ou [n]")
19.     leia(opcao)
20.   enquanto(opcao!='n')
21. Fimalgoritmo
```

Figura 103 – Algoritmo tabuada de N com limitação e condição de saída

Neste exemplo, foram utilizados dois laços de repetição com teste no fim. O primeiro, que se inicia na linha 5 termina na linha 18, foi usado para que o algoritmo se mantenha em execução até o usuário solicitar seu encerramento, na linha 17, e o segundo, que se inicia na linha 14 e termina na linha 17, foi usado para calcular a tabuada de um número, variável **n**, até um número informado pelo usuário, variável **f**.

Exemplo 4

Elabore um algoritmo para calcular a série de Fibonacci a partir de entradas positivas ou negativas.

Solução

```
1.  Algoritmo "Fibonacci – Positivos ou Negativos"
2.  Var
3.    a, b, c, aux, i, n : inteiro
4.  Inicio
5.    //inicializando as variáveis a, b, c
6.    a <- 0
7.    b <- 1
8.    c <- -1
9.     //entrada
10.         escreval("Série de Fibonacci Fib(n)")
11.         escreva("Informe o número: ")
12.         leia(n)
13.
14.   //processamento
15.   se(n<0) então
16.     escreva("Fib(",n,") = ", a)
17.     i <- -1
18.     enquanto (n <= i) faca
19.       aux <- a+c
20.       a <- c
21.       c <- aux
22.       escreva(", ", a)
23.       n <- n + 1
24.     fimenquanto
25.   senão
26.     escreva("Fib(",n,") = ", a)
27.     para i<-2 ate n passo 1 faca
28.       aux<-a+b
29.       a<-b
30.       b<-aux
31.       escreva(", ", a)
32.     fimpara
33.   fimse
34. Fimalgoritmo
```

Figura 104 – Algoritmo série de Fibonacci, números positivos e negativos

Neste exemplo, foram utilizadas duas estruturas de repetição diferentes: ENQUANTO... FAÇA e PARA... ATÉ... PASSO... FAÇA. No primeiro laço de repetição, que está entre as linhas 18 e 24, é realizado o processo de cálculo da série de Fibonacci para números negativos, para os números positivos o cálculo é feito entre as linhas 27 e 32.

Para que um ou outro laço seja usado, uma condição SE foi imposta para determinar se o número informado pelo usuário, na linha 12, é positivo ou negativo. Um número negativo é um número menor do que zero, portanto, essa é a primeira condição testada na linha 15, se não se entende que é um número positivo (linha 25).

Em ambos os laços, à medida que a série é calculada, os números são escritos na tela, nas linhas 22 e 31.

Exemplo 5

Elabore um algoritmo que capture a altura, a idade e o sexo de um número determinado de pessoas e informe a maior altura e a menor altura, a idade do mais velho e do mais novo, o total de pessoas por sexo e o total de pessoas.

Solução

```
1.  Algoritmo "Maior e Menor Altura e Idade"
2.
3.  Var
4.  // Seção de Declarações das variáveis
5.    idade, maior_idade, menor_idade : inteiro
6.    total_homens, total_mulheres,total_outros, total_pessoas : inteiro
7.    sexo, resposta : caractere
8.    altura, maior_altura, menor_altura : real
9.
10. Inicio
11.   //inicialização das variáveis
12.   maior_idade <- 0
13.   menor_idade <- 200
14.   total_masculino <- 0
15.   total_feminino <- 0
16.   total_outros <- 0
17.   total_pessoas <- 0
18.   maior_altura <- 0.0
19.   menor_altura <- 10.0
20.
21.   repita
22.         //entrada
23.         escreval(" Idade.....: ")
24.         leia(idade)
25.         escreval(" Altura.....: ")
26.         leia(altura)
27.         escreva(" Sexo [M] masculino, [F] feminino [O] outro..: ")
28.         leia(sexo)
29.         total_pessoas <- total_pessoas + 1
30.
31.         //processamento do mais velho e mais novo
32.         se (idade>maior_idade) entao
33.           maior_idade <- idade
34.         fimse
35.         se (idade<menor_idade) entao
36.           menor_idade <- idade
37.         fimse
38.
```

```
39.             //processamento do mais alto e o mais baixo
40.          se (altura>maior_altura) entao
41.             maior_altura <- altura
42.          fimse
43.          se(altura<menor_altura) entao
44.             menor_altura <- altura
45.          fimse
46.          //processamento dos totais por sexo
47.          se ((sexo="m") ou (sexo="M")) entao
48.            total_masculino <- total_masculino + 1
49.          senao
50.            se((sexo="f") ou (sexo="F")) entao
51.                        total_feminino <- total_feminino + 1
52.            senao
53.                        total_outros <- total_outros +1
54.            fimse
55.          fimse
56.
57.          escreva(" Deseja continuar? [s/n] ")
58.          leia(resposta)
59.    até (resposta<>"s")
60.
61.          //saída
62.    escreval("  Maior Idade.....: ", maior_idade)
63.    escreval("  Menor Idade.....: ", menor_idade)
64.    escreval("  Maior Altura.....: ", maior_altura)
65.    escreval("  Menor Altura.....: ", menor_altura)
66.    escreval("  Total de Homens.....: ", total_masculino)
67.    escreval("  Total de Mulheres.....: ", total_feminino)
68.    escreval("  Total Outros.....: ", total_outros)
69.    escreval("  Total de Pessoas.....: ", total_pessoas)
70.  Fimalgoritmo
```

Figura 105 – Algoritmo pessoas e seus dados

O propósito do algoritmo apresentado é efetuar uma pesquisa sobre um número indeterminado de pessoas e, ao término, apresentar a maior idade e a menor idade dentro de uma faixa etária, a maior altura e a menor altura dentre as pessoas cadastradas, o total de homens, mulheres e pessoas de outras identidades de gênero, além de apresentar o número total de pessoas cujos dados foram coletados.

Nas linhas de 11 a 19, foram inicializadas as variáveis responsáveis por armazenar os totais. Essas inicializações são necessárias para que a soma que acumula o valor anterior mais o novo valor seja computada corretamente.

O laço de repetição entre as linhas 21 e 59 é o responsável por repetir as perguntas idade, altura e sexo para cada pessoa. Assim que o usuário responde a altura, a idade e o sexo, o algoritmo processa a idade do mais velho e do mais novo, bem como a maior e a menor altura. Adicionalmente, o algoritmo computa os totais de homens, mulheres e, caso o usuário não tenha digitado as letras f minúscula ou maiúscula ou m minúscula ou maiúscula, o algoritmo computa o total de pessoas que declararam outros para a variável sexo.

Dentro do laço de repetição, as instruções capturam as respostas individuais de cada pessoa a cada iteração. E como condição de saída do laço é perguntar ao usuário se deseja continuar, o laço será repetido até que a resposta seja diferente de **s** (repetindo o bloco até que a opção seja diferente de

sim para continuar, ou, em outros termos, quando a resposta for **não**). Utilizar o laço com teste no final foi necessário para que o algoritmo executasse pelo menos uma vez o bloco dentro do laço pois, dessa forma, se apenas dois dados de uma única pessoa forem inseridos, esses serão os valores a serem apresentados.

Quando a resposta for **N** (não) o laço será interrompido e os totais computados serão apresentados. Os comandos das linhas 61 a 69 são os comandos de saída. Observe que as variáveis foram identificadas com nomes que indicam seus propósitos.

Exemplo 6

Um usuário precisa somar notas fiscais (NF) a partir de um número de NF informado pelo usuário, sendo que cada NF possui um número indeterminado de itens e cada item possui um valor que o usuário deve comunicar.

Quando o usuário terminar de digitar cada NF, deve ser exibido o valor total da soma dos itens da NF digitada; e, quando ele terminar de digitar todas as NFs, deve ser exibido o valor total da soma de todas elas.

A figura seguinte é um exemplo de NF.

Nota fiscal de venda ao consumidor – MOD 2 Série D

Nome: CNPJ:

Endereço:

Data de Emissão: ___ / ___ / _____ Nº

Destinatário: _____

Endereço: _____

CPF / CNPJ: _____

Quantidade	Descrição	Preço unitário	Total

Total

Figura 106 – Modelo de NF

Solução

```
1.  Algoritmo "Soma das Notas Fiscais"
2.  Var
3.    somaNF, somaTotal, valorItem : real
4.    contadorNF, contadorItens, quantidadeNF, numeroItens : inteiro
5.  Inicio
6.    escreva("Informe a quantidade de Notas Fiscais a somar: ")
7.    leia(quantidadeNF)
8.    somaNF <- 0
9.    somaTotal <- 0
10.   para contadorNF de 1 ate quantidadeNF faca
11.     escreva("", contadorNF,"ª Nota Fiscal")
12.     escreva("Digite o total de itens dessa NF", contadorNF,": ")
13.     leia(numeroItens)
14.     somaNF <- 0
15.     para contadorItens de 1 ate numeroItens faca
16.       escreva("Informe o valor do item", contadorItens, ": ")
17.       leia(valorItem)
18.       somaNF <- somaNF + valorItem
19.     fimpara
20.     somaTotal <- somaTotal + somaNF
21.     escreval("Soma dos itens da NF", contadorNF, "é: ", somaNF)
22.   fimpara
23.   escreval("")
24.   escreval("Soma dos totais de Todas as NFs é: ", somaTotal)
25. Fimalgoritmo
```

Figura 107 – Algoritmo soma das NFs

O algoritmo da figura anterior é o encadeamento do comando PARA a fim de solucionar dois somatórios, o primeiro dos itens de cada nota fiscal e o segundo para somar o total de todas as notas fiscais. Assim, o laço externo PARA que se inicia na linha 12 irá armazenar o total de cada nota fiscal e o laço interno que se inicia na linha 17 computa o valor de todos os itens.

Resumo

A estrutura de decisão encadeada SE... ENTÃO pode tornar a lógica da programação difícil para compreender e implementar manutenções futuras, especialmente quando o algoritmo dentro de cada bloco de comandos é muito grande.

São muitas as situações dependentes de uma decisão e a escolha dentre as duas é uma decisão do programador.

A diferença entre as estruturas SE... SENÃO e ESCOLHA-CASO é que a primeira aceita expressões lógicas e relacionais na condição de teste, enquanto a segunda, por padrão, trabalha com variáveis discretas ou valores de intervalos finitos e conhecidos e não aceita expressões. O quadro a seguir resume as principais diferenças entre as duas estruturas de decisão existentes:

Quadro 21 – Comparação entre as estruturas de decisão SE... SENÃO e ESCOLHA-CASO

SE... SENÃO	ESCOLHA-CASO
Usa variáveis discretas e contínuas	Usa variáveis discretas
Condições de verificação e teste, aceita operadores relacionais >, >=, <, <=, ==, !=	Não aceita expressões relacionais e é verificada apenas a relação de igualdade
Opera com variáveis de todos os tipos	Opera com variáveis do tipo inteiro e caractere

A estrutura de repetição é amplamente usada e necessária para permitir que um programa possa executar um bloco de código quantas vezes forem necessárias até que seu teste esteja satisfeito. Isso é muito útil quando se quer trabalhar com múltiplos dados sem que seja necessário criar e repetir rotinas de código.

Os métodos de repetição são similares, contudo, há pontos de divergência em sua utilização. Em alguns algoritmos é necessário que o bloco do código de repetição seja executado ainda que o valor atribuído antes do teste faça com que o processamento não siga adiante. Nesse caso, se utilizasse a estrutura ENQUANTO... FAÇA o bloco de código existente não seria executado, mas com o FAÇA... ENQUANTO ou REPITA... ATÉ seria.

Exercícios

Questão 1. Considere o algoritmo apresentado a seguir, escrito em Portugol.

```
Algoritmo "Pergunta"
Var
val1, val2: inteiro
num3, num4: inteiro
Inicio
  escreva("Primeiro valor: ")
  leia(val1)
  escreva("Segundo valor: ")
  leia(val2)
  num3<-10
  num4<-20
  se (val1 < num4) e (val1>num3) entao
    escreva("1")
  fimse
  se (val2<num4) e (val2>num3) entao
    escreva("2")
  fimse

  se (val2>val1) entao
    escreva("3")
  fimse
  se (val1<>val2) entao
    escreva("4")
  fimse
FimAlgoritmo
```

Figura 108

Suponha que um usuário execute o programa do enunciado para diferentes entradas e considere apenas o texto mostrado após a digitação das duas entradas solicitadas pelo programa. Nesse cenário, avalie as afirmativas:

I – Se a primeira entrada do usuário for o número 12 e a segunda o número 15, o programa deverá mostrar na tela a seguinte saída: 1234.

II – Se a primeira entrada do usuário for o número 2 e a segunda o número 3, o programa deverá mostrar na tela a seguinte saída: 34.

III – Se a primeira entrada do usuário for o número 40 e a segunda o número 30, o programa deverá mostrar na tela a seguinte saída: 4030.

IV – Se a primeira entrada do usuário for o número 16 e a segunda o número 5, o programa deverá mostrar na tela a seguinte saída: 14.

É correto o que se afirma apenas em:

A) I.

B) II.

C) I e III.

D) III e IV.

E) I, II e IV.

Resposta correta: alternativa E.

Análise das afirmativas

I – Afirmativa correta.

Justificativa: na afirmativa, o primeiro valor inserido pelo usuário é o número 12, que é maior do que **num3** (igual a 10) e menor do que **num4** (igual a 20). Assim, sabemos que o programa deve escrever pelo menos o número 1, resultado da primeira comparação. O mesmo ocorre com a variável **val2**, que é igual a 15, e, portanto, o número 2 também deve ser mostrado na tela, resultado da segunda comparação. Além disso, 15 é maior do que 12, fazendo com que o número 3 seja impresso, de acordo com a terceira comparação. Finalmente, como **val1** e **val2** são diferentes, o número 4 também será mostrado, resultado da quarta comparação e da última comparação.

II – Afirmativa correta.

Justificativa: na afirmativa, devemos observar que as entradas do usuário são 2 e 3, ambas menores do que o conteúdo da variável **num3** (igual a 10). Em função disso, as condições das duas primeiras comparações não são verdadeiras e, portanto, os números 1 e 2 não são impressos. Contudo, como o conteúdo da variável **val2** é maior do que o conteúdo da variável **val1**, uma vez que 3 > 2, o número 3 deve ser impresso, como resultado da terceira comparação. Finalmente, como o conteúdo das variáveis **val1** e **val2** são números diferentes, a última comparação é verdadeira, levando à impressão na tela do número 4.

III – Afirmativa incorreta.

Justificativa: se observarmos o código do programa, notaremos que ele não imprime em nenhum momento o número 0 – apenas os números 1, 2, 3 e 4 são impressos pelo programa. Logo, uma saída como 4030 não pode acontecer, motivo pelo qual a afirmativa é incorreta.

IV – Afirmativa correta.

Justificativa: como o primeiro valor entrado pelo usuário é igual a 16, ambas as condições da primeira comparação se tornam verdadeiras, uma vez que 16 é maior do que 10 e menor do que 20. Contudo, como **num4** é igual a 5 e menor do que **num3** (que é igual a 10), uma das condições da segunda comparação torna-se falsa, o que faz com que o comando **escreva("2")** não seja executado e, portanto, o número 2 não é mostrado na tela. Além disso, como 5 é menor do que 16, a condição da terceira comparação torna-se falsa, fazendo com que o comando **escreva("3")** não seja executado. Finalmente, na última comparação, como 5 e 16 são diferentes, a condição torna-se verdadeira, o que leva à impressão do valor 4 na tela. Dessa forma, a saída do programa após os números digitados pelo usuário deve ser 14, como dito na afirmativa.

Uma observação relevante é a de que o comando **escreva** não pula linhas na saída do texto, de forma que todos os números são impressos na mesma linha.

Questão 2. Observe atentamente o algoritmo apresentado a seguir, escrito utilizando a sintaxe do Portugol usada pelo programa VisuAlg:

```
Algoritmo "Pergunta"
Var
x, y, z: inteiro
Inicio
   escreva("Primeiro valor: ")
   leia(x)
   escreva("Segundo valor: ")
   leia(y)
   escreva("Terceiro valor: ")
   leia(z)
   escreva("a")
   se (x-y < 0) e (x-z<0) entao
      escreva("b")
      se y-z < 0 entao
         escreva("c")
      senao
         escreva("d")
      fimse
      escreva("e")
   fimse
   escreva("f")
FimAlgoritmo
```

Figura 109

Com base no código e nos seus conhecimentos e considerando apenas a saída do programa após o usuário ter digitado os três valores solicitados, avalie as afirmativas. Considere também que o usuário digitou apenas números inteiros válidos.

I – Após a última entrada do usuário, o programa deve imprimir na tela um conjunto de letras, obrigatoriamente começando com letra **a** e sempre terminando com a letra **f**, independentemente de quais sejam os valores entrados previamente pelo usuário.

II – Se a primeira entrada do usuário for o número 3, a segunda o número 2 e a terceira o número 1, o programa deverá mostrar na tela a seguinte saída: **af**.

III – Independentemente dos valores entrados pelo usuário, o programa nunca poderá imprimir as letras **c** ou **d** seguidas uma da outra. Por exemplo, uma saída como **abcdef** não pode ser obtida.

IV – Independentemente da entrada do usuário, a saída do programa deve sempre terminar com as letras **ef**.

É correto o que se afirma apenas em:

A) I.

B) II.

C) I, II e III.

D) III e IV.

E) I, II e IV.

Resposta correta: alternativa C.

Análise das afirmativas

I – Afirmativa correta.

Justificativa: podemos observar que as linhas com os comandos **escreva("a")** e **escreva("f")** ocorrem fora dos comandos de desvio condicional **SE**. Dessa forma, esses comandos devem ser executados em todas as condições. Além disso, o comando **escreva("a")** é executado antes dos comandos para a escrita das outras letras, fazendo com que a letra **a** seja a primeira a ser impressa. O comando **escreva("f")** é o último comando de impressão executado, independentemente de qualquer condição e, por esse motivo, a letra **f** deve ser sempre a última a ser impressa.

II – Afirmativa correta.

Justificativa: se o valor de x (atribuído na primeira entrada do usuário) for maior do que o valor de y (atribuído na segunda entrada do usuário), a condição do primeiro **SE** não é satisfeita, e os seus comandos internos não são executados (inclusive o segundo **SE**, mais interno). Dessa forma, apenas os seguintes comandos de impressão de resultados são executados: **escreva("a")** e **escreva("f")**, fazendo com que a saída seja **af**.

III – Afirmativa correta.

Justificativa: nesse programa, as letras **c** e **d** não podem ser impressas simultaneamente. Devemos observar que os comandos **escreva("c")** e **escreva("d")** são executados de acordo com a condição do **SE** mais interno. A condição em questão é:

y-z<0

Alternativamente, podemos afirmar que:

y<z

Se essa condição for verdadeira, o comando **escreva("c")** é executado. Caso contrário (indicado pelo **senao** no código), o comando **escreve("d")** é executado. Dessa forma, temos a execução de um dos dois comandos, mas nunca dos dois simultaneamente. Assim, cadeias de texto como **abcdef** não podem ser uma saída válida desse programa.

IV – Afirmativa incorreta.

Justificativa: a linha com o comando **escreva("e")** só é executada se a entrada do usuário satisfizer a condição do primeiro comando de desvio condicional do código, o comando **SE**. No caso, a condição é a seguinte:

x-y<0 e x-z<0

De forma alternativa, mas com efeito equivalente, temos:

x<y e x<z

Caso essa condição não seja atendida, todos os comandos internos ao **SE** não serão executados, inclusive a linha **escreva("e")**. Nesses casos, a saída do programa não terminará com as letras **ef**, como indicado na afirmativa. Por exemplo, esse é o caso citado na afirmativa II, em que a saída do programa é apenas **af**, que não termina em **ef**.