algoritmos e programação

com exemplos em Pascal e C

nina edelweiss
maria aparecida castro livi

bookman

as autoras

Nina Edelweiss é engenheira eletricista e doutora em Ciência da Computação pela Universidade Federal do Rio Grande do Sul. Durante muitos anos, lecionou em cursos de Engenharia e de Ciência da Computação na UFRGS, na UFSC e na PUCRS. Foi, ainda, orientadora do Programa de Pós-Graduação em Ciência da Computação da UFRGS. É coautora de três livros, tendo publicado diversos artigos em periódicos e em anais de congressos nacionais e internacionais. Participou de diversos projetos de pesquisa financiados por agências de fomento como CNPq e FAPERGS, desenvolvendo pesquisas nas áreas de bancos de dados e desenvolvimento de software.

Maria Aparecida Castro Livi é licenciada e bacharel em Letras, e mestre em Ciência da Computação pela Universidade Federal do Rio Grande do Sul. Desenvolveu sua carreira profissional na UFRGS, onde foi programadora e analista de sistema, antes de ingressar na carreira docente. Ministrou por vários anos a disciplina de Algoritmos e Programação para alunos dos cursos de Engenharia da Computação e Ciência da Computação. Sua área de interesse prioritário é o ensino de Linguagens de Programação, tanto de forma presencial quanto a distância.

E22a Edelweiss, Nina.
Algoritmos e programação com exemplos em Pascal e C [recurso eletrônico] / Nina Edelweiss, Maria Aparecida Castro Livi. – Dados eletrônicos. – Porto Alegre : Bookman, 2014.

Editado também como livro impresso em 2014.
ISBN 978-85-8260-190-7

1. Informática. 2. Algoritmos – Programação. I. Livi, Maria Aparecida Castro. II. Título.

CDU 004.421

Catalogação na publicação: Ana Paula M. Magnus – CRB 10/2052

4.5 → comando de seleção múltipla

O **comando de seleção múltipla** seleciona uma dentre diversas opções, com base na avaliação de uma expressão. Na pseudolinguagem aqui utilizada, a sintaxe deste comando é:

```
caso <expressão> seja
  <rótulo 1>: <comando 1>
  <rótulo 2>: <comando 2>
        ...
  <rótulo n>: <comando n>
  [ senão <comandos> ]
fim caso
```

onde os símbolos "[" e "]" significam que essa linha é opcional.

O comando inicia com um cabeçalho `caso <expressão> seja`, seguido de uma série de comandos rotulados, ou seja, comandos precedidos por um valor seguido do caractere ":". O resultado da avaliação da expressão utilizada no cabeçalho e os valores representados nos rótulos devem ser todos do mesmo tipo e corresponder a um valor ordinal como, por exemplo, inteiro ou caractere. Cada rótulo corresponde a somente um comando. Um comando composto pode ser utilizado caso se queira executar mais de uma ação para um determinado rótulo.

Depois de avaliada a expressão, seu resultado é comparado com cada um dos rótulos, na ordem em que são definidos. Somente o comando associado ao primeiro rótulo que for igual ao resultado da expressão é executado. Só a igualdade é verificada. Se o valor da expressão não for igual a qualquer dos rótulos, nada será executado pelo comando. Opcionalmente, poderá ser utilizada a cláusula `senão`, que indica o comando a ser executado caso nenhum dos rótulos corresponda ao valor da expressão do cabeçalho. O final do comando de seleção múltipla é indicado pelas palavras reservadas `fim caso`. Por exemplo, supondo que a variável a seja uma variável do tipo inteiro:

```
caso a seja
   1: a ← 0          {COMANDO SIMPLES PARA O CASO a = 1}
   2: início         {COMANDO COMPOSTO PARA O CASO a = 2}
      ler(a)
      escrever(a)
      fim
   3: a ← a + 1      {COMANDO SIMPLES PARA O CASO a = 3}
   senão escrever(a){CLÁUSULA OPCIONAL PARA OUTROS VALORES DE a}
fim caso
```

Um comando de seleção múltipla equivale a um comando de seleção dupla com outros comandos de seleção dupla aninhados nele. O exemplo anterior equivale a:

```
se a = 1
então a ← 0                    {COMANDO SIMPLES PARA O CASO a = 1}
senão se a = 2
      então início             {COMANDO COMPOSTO PARA O CASO a = 2}
            ler(a)
            escrever(a)
            fim
      senão se a = 3
            então a ← a + 1    {COMANDO SIMPLES PARA O CASO a = 3}
            senão escrever(a)  {CLÁUSULA OPCIONAL}
```

Uma vantagem de usar o comando de seleção múltipla em lugar desses comandos aninhados está na possibilidade de utilizar somente um nível de indentação, o que torna mais clara a visualização das alternativas existentes.

A Figura 4.5 representa o fluxograma correspondente ao exemplo anterior. Pode-se observar que o fluxo do programa somente passará por um dos comandos associados aos rótulos.

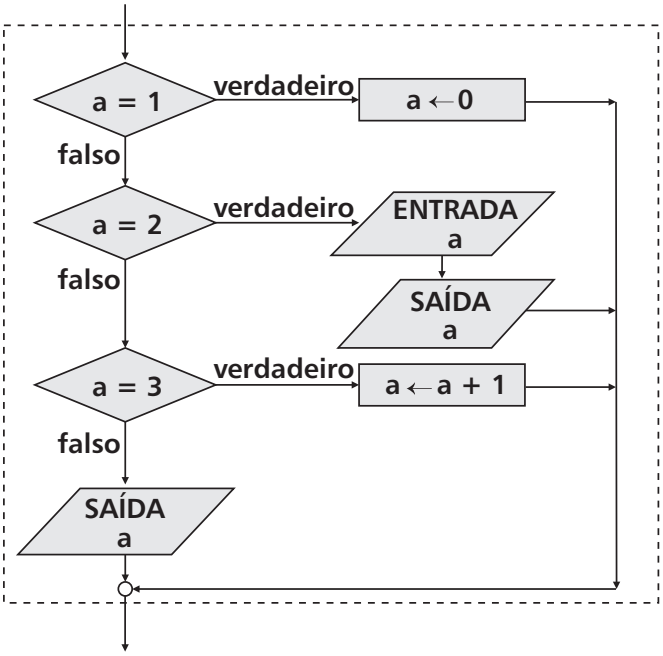

figura 4.5 Fluxograma de um comando de seleção múltipla.

O algoritmo a seguir, que calcula e informa a média e o conceito de um aluno, ilustra a utilização de rótulos do tipo caractere em um comando de seleção múltipla:

```
Algoritmo 4.5 - MédiaConceito2
{INFORMA MÉDIA E CONCEITO DE UM ALUNO}
  Entradas: nota1, nota2, nota3 (real)
  Saídas: média (real)
          Informação do conceito do aluno
  Variável auxiliar: conceito (char)
```

```
início
  ler (nota1, nota2, nota3)                          {ENTRADA DAS 3 NOTAS}
  média ← (nota1 + nota2 + nota3) / 3                {CALCULA MÉDIA}
  escrever (média)                                   {INFORMA MÉDIA}
  se média ≥ 9                                       {CÁLCULO DO CONCEITO}
  então conceito ← 'A'
  senão se média ≥ 7,5
        então conceito ← 'B'
        senão se média ≥ 6,0
              então conceito ← 'C'
              senão conceito ← 'D' {MÉDIA < 6}
  caso conceito seja                                 {INFORMA CONCEITO}
     'A': escrever('Conceito A - Parabéns!')
     'B': escrever('Conceito B')
     'C': escrever('Conceito C')
     'D': escrever('Conceito D - Você foi reprovado')
  fim caso
 fim
```

A implementação do comando de seleção múltipla varia, dependendo da linguagem de programação utilizada. Algumas linguagens permitem que um comando seja rotulado com uma lista de valores, ou mesmo com um intervalo. No exemplo a seguir, `nota` somente pode conter um valor inteiro:

```
caso nota seja
            0..5: escrever('Reprovado'){RÓTULO DO TIPO INTERVALO}
   6, 7, 8, 9, 10: escrever('Aprovado') {LISTA DE RÓTULOS}
fim caso
```

O primeiro comando é rotulado com o intervalo `0..5`, representando os valores inteiros `0`, `1`, `2`, `3`, `4` e `5`. O comando associado ao intervalo será executado quando o valor da variável `nota` for um desses valores. O segundo comando apresenta uma lista de rótulos. Se o valor da variável `nota` for igual a um deles, o segundo comando será executado. Observar que a utilização desses dois tipos de rótulos gerou um comando conciso e muito fácil de compreender.

4.6 → exercícios de fixação

exercício 4.1 Ler um número inteiro. Se o número lido for positivo, escrever uma mensagem indicando se ele é par ou ímpar.

```
Algoritmo ParOuÍmpar
{INFORMA SE UM VALOR LIDO DO TECLADO É PAR OU ÍMPAR}
  Entrada: valor (inteiro)            {VALOR A SER TESTADO}
  Saída: Mensagem de 'par' ou 'ímpar'
```

```
  Variável auxiliar: ehPar (lógica)
início
  ler(valor)                          {ENTRADA DO VALOR A SER TESTADO}
  se  valor ≥ 0                       {VALOR LIDO É POSITIVO}
  então início
        ehPar ← (valor mod 2) = 0     {VERIFICA SE VALOR É PAR}
        se ehPar                      {VERDADEIRO SE ehPar FOR VERDADEIRO}
        então escrever('Par ! ')
        senão escrever('Ímpar ! ')
        fim
fim
```

exercício 4.2 Dados os coeficientes de uma equação do 2º grau, calcular e informar os valores de suas raízes.

```
Algoritmo EquaçãoSegundoGrau
{INFORMA OS VALORES DAS RAÍZES DE UMA EQUAÇÃO DO SEGUNDO GRAU}
  Entradas: a, b, c (real)              {COEFICIENTES DA EQUAÇÃO}
  Saídas: r1, r2 (real)                 {RAÍZES}
  Variável auxiliar: disc (real)        {DISCRIMINANTE}
início
  ler(a, b, c)  {ENTRADA DOS VALORES DOS COEFICIENTES DA EQUAÇÃO}
  se a = 0
  então início
        escrever('Não é equação do segundo grau! ')
        escrever('Raiz = ', (- c / b ))
        fim
  senão início
        disc ← sqr(b) - 4 * a * c       {CÁLCULO DO DISCRIMINANTE}
        se disc < 0
        então escrever('Raízes imaginárias !')
        senão início
              r1 ← ( - b + sqrt ( disc ) ) / ( 2 * a )
              r2 ← ( - b - sqrt ( disc ) ) / ( 2 * a )
              escrever('Raízes: ', r1, r2)
              fim
       fim
fim
```

exercício 4.3 Processar uma venda de livros em uma livraria. O cliente poderá comprar diversas unidades de um mesmo tipo de livro. O código que define o tipo do livro vendido (A, B, C) e o número de unidades desse livro são fornecidos como dados de entrada.

Preços: Tipo A – R$ 10,00
Tipo B – R$ 20,00
Tipo C – R$ 30,00

Calcular e informar o preço a pagar. Caso tenham sido vendidos mais de 10 livros, exibir uma mensagem informando isso.

A solução deste problema é mostrada em duas etapas. Inicialmente, é apresentado um algoritmo em passos gerais para dar uma visão global da solução. Depois, cada um dos passos é detalhado, dando origem ao algoritmo completo.

```
Algoritmo UmaVenda - PASSOS GERAIS
{PROCESSA UMA VENDA DE LIVROS}
  Entradas: tipo do livro ('A', 'B' ou 'C')
            número de livros
  Saídas: preço a pagar
          mensagem indicando que foram vendidas mais de 10 unidades
```

1. Obter dados
2. Calcular preço de venda
3. Emitir mensagem caso necessário
4. Terminar

Detalhamento do algoritmo:

```
Algoritmo UmaVenda
{PROCESSA UMA VENDA DE LIVROS}
  Entradas: código (caractere)          {CÓDIGO DO LIVRO}
            numeroUnidades (inteiro)    {NR. UNIDADES VENDIDAS}
  Saídas: aPagar (real)                 {PREÇO A PAGAR}
          {Mensagem indicando que foram vendidas mais de 10
           unidades}
início
  ler(código, numeroUnidades)           {ENTRADA DE DADOS}
  se código = 'A'                       {CÁLCULO DO PREÇO DA VENDA}
  então aPagar ← numeroUnidades * 10
  senão se código = 'B'
        então aPagar ← numeroUnidades * 20
        senão se código = 'C'
              então aPagar ← numeroUnidades * 30
              senão início              {CÓDIGO ESTÁ INCORRETO}
                    aPagar ← 0
                    escrever('Código errado')
                    fim
  se aPagar > 0                         {CÓDIGO ERA VÁLIDO}
  então início
        escrever(aPagar)                {INFORMA VALOR A PAGAR}
        se numeroUnidades > 10
        então escrever ('Foram vendidas mais de 10 unidades')
        fim
  fim
```

exercício 4.4 Processar uma venda em um estabelecimento comercial. São fornecidos o código do produto vendido e seu preço. Deverá ser dado um desconto, de acordo com a seguinte tabela:

Código A – 20%
Código B – 10%

Produtos com código diferente de A ou B não terão desconto. Além disso, se o total a pagar for maior ou igual a R$ 80,00, deverá ser dado um desconto adicional de 10%. Calcular e informar o preço a pagar e o desconto dado na compra, se for o caso.

```
Algoritmo UmaVendaComércio
{PROCESSA UMA VENDA EM UM ESTABELECIMENTO COMERCIAL}
   Entradas: preço (real)
             código (caractere)
   Saídas: desconto (real)
           aPagar (real)
início
  ler(código, preço)              {ENTRADA DE DADOS}
  aPagar ← preço                  {INICIALIZA aPagar}
  desconto ← 0                    {INICIALIZA desconto}
  se código = 'A'                 {CALCULA desconto}
  então desconto ← preço / 5
  senão se código = 'B'
        então desconto ← preço / 10
  aPagar ← aPagar - desconto      {CALCULA aPagar}
  se aPagar ≥ 80,00               {VERIFICA SE COMPRA ≥ 80,00}
  então início
        desconto ← desconto + aPagar / 10
        aPagar ← aPagar * 0,9     {MAIS 10% DE DESCONTO}
        fim
  escrever(aPagar)                {INFORMA VALOR A PAGAR}
  se desconto ≠ 0
  então escrever(desconto)        {INFORMA DESCONTO RECEBIDO}
  fim
```

exercício 4.5 Escrever um programa que, dados um determinado mês (representado por um número inteiro) e um ano, informe quantos dias tem esse mês. Para determinar o número de dias de fevereiro, verificar primeiro se o ano é bissexto. Um ano será bissexto se terminar em 00 e for divisível por 400, ou se não terminar por 00, mas for divisível por 4.

```
Algoritmo DiasDoMês1
{INFORMA QUANTOS DIAS TEM UM DETERMINADO MÊS}
  Entrada: mês, ano (inteiro)   {DADOS DE ENTRADA - MÊS E ANO}
  Saída: dias (inteiro)         {NÚMERO DE DIAS DO MÊS NESTE ANO}
  Variável auxiliar: ehBissexto (lógica)
```

```
início
  ler(mês, ano)        {ENTRADA DE DADOS}
  se mês = 2           {SE FEVEREIRO, VERIFICA SE ANO EH BISSEXTO}
  início
     eh Bissexto  ← falso
     se (ano mod 100) = 0
     então início
           se (ano mod 400) = 0
           então ehBissexto  ←  verdadeiro
           fim
     senão se (ano mod 4) = 0
           então ehBissexto  ←  verdadeiro
  fim
  se mês = 2                   {FEVEREIRO}
  então se ehBissexto
        então dias ← 29
        senão dias ← 28
  senão se (mês=4) ou (mês=6) ou (mês=9) ou (mês=11)
        então dias ← 30        {ABRIL, JUNHO, SETEMBRO, NOVEMBRO}
        senão dias ← 31        {DEMAIS MESES}
  escrever(dias)               {INFORMA NÚMERO DE DIAS DO MÊS}
fim
```

Algoritmo DiasDoMês2

```
{INFORMA QUANTOS DIAS TEM UM DETERMINADO MÊS}
  Entrada: mês, ano (inteiro)   {DADOS DE ENTRADA - MÊS E ANO}
  Saída: dias (inteiro)         {NÚMERO DE DIAS DO MÊS NESTE ANO}
  Variável auxiliar: ehBissexto (lógica)
início
  ler(mês, ano)                 {ENTRADA DE DADOS}
  caso mês seja
    2 : {FEVEREIRO}
        eh_Bissexto ← falso
        se (ano mod 100) = 0
        então início
              se (ano mod 400) = 0
              então ehBissexto  ←  verdadeiro
              fim
        senão se (ano mod 4) = 0
              então ehBissexto  ←  verdadeiro
        se ehBissexto
        então dias ← 29         {ANO BISSEXTO}
        senão dias ← 28
    4, 6, 9, 11 : dias ← 30     {ABRIL, JUNHO, SETEMBRO, NOVEMBRO}
```

```
    senão dias ← 31              {DEMAIS MESES}
 fim caso                        {FIM COMANDO DE SELEÇÃO MÚLTIPLA}
  escrever(dias)                 {INFORMA NÚMERO DE DIAS DO MÊS}
fim
```

exercício 4.6 Fazer um programa que leia um caractere e exiba uma mensagem informando se ele é uma letra, um dígito numérico ou um caractere de operação aritmética. Se o caractere não for de qualquer desses três tipos, informar que se trata de um caractere desconhecido.

```
Algoritmo IdentificarCaractere
{INFORMA TIPO DE CARACTERE LIDO}
  Entradas: lido (caractere)      {CARACTERE A SER IDENTIFICADO}
  Saídas: Mensagem indicando o tipo de caractere lido
início
  ler(lido)                       {CARACTERE A SER IDENTIFICADO}
  caso lido seja
    'A'..'Z', 'a'..'z' : escrever('Letra')
    '0'..'9': escrever('Dígito')
    '+', '-', '*', '/' : escrever('Operador aritmético')
    senão escrever('Caractere desconhecido')
  fim caso
fim
```

Sugestão: resolva este exercício sem utilizar o comando de seleção múltipla.