23

série livros didáticos informática ufrgs

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

5.3 → comando de repetição condicional `enquanto/faça` por avaliação prévia de condição

Existem situações em que as repetições não estão condicionadas a um número definido de vezes. Por exemplo, em uma loja, não é possível saber quantas vendas ocorrerão ao longo de um dia, mas se sabe que essas serão encerradas no horário definido para que a loja feche suas portas. O processamento de cada venda (registro do valor da venda, do vendedor que efetuou a venda, etc.) é semelhante, compondo um conjunto de comandos a serem repetidos. A repetição nesse caso está condicionada à ocorrência de duas condições: venda efetuada e horário adequado.

O **comando de repetição condicional `enquanto/faça`** faz que um comando, simples ou composto, tenha sua execução condicionada ao resultado de uma expressão lógica, isto é, a execução desse comando é repetida enquanto o valor lógico resultante da avaliação da expressão de controle for verdadeiro. A sintaxe de um comando de repetição condicional `enquanto/faça` é:

```
enquanto <expressão lógica> faça
   <comando>
```

A Figura 5.4 representa o fluxograma referente ao comando de repetição condicional `enquanto/faça`.

O laço nunca será executado caso o valor inicial da expressão lógica seja falso logo de início, já que a avaliação da condição de controle ocorre antes da execução do comando a ser repe-

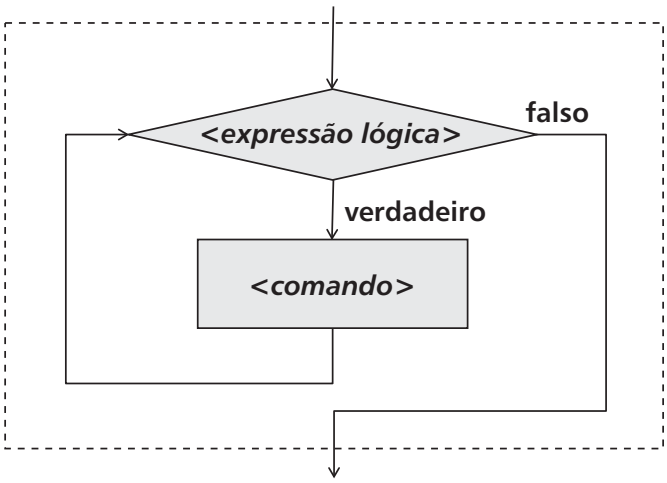

figura 5.4 Fluxograma do comando de repetição `enquanto/faça`.

tido. Se o valor inicial da expressão lógica de controle for verdadeiro, é necessário que algum componente dessa expressão seja alterado dentro do laço para que ela, em algum momento, seja avaliada como falsa, fazendo que a repetição encerre, evitando assim a ocorrência de um laço – *loop* – infinito. Portanto, esse comando somente é indicado para situações em que a expressão de controle inclui algum dado a ser modificado pelo programa dentro do próprio laço, determinando assim o encerramento das repetições.

No trecho de programa a seguir, a variável `a` é inicializada com 1 e, com esse valor, é iniciada a execução do comando `enquanto/faça`. A cada repetição do laço, `a` é incrementada em uma unidade e isso se repete enquanto `a` for inferior a 5. Quando `a` atingir o valor 5, o comando `enquanto/faça` será encerrado, sendo executado o próximo comando, que imprime o valor final de `a`, que é 5.

```
a ← 1
enquanto a < 5 faça
  a ← a + 1
escrever (a)
```

Variáveis lógicas são muito utilizadas na expressão lógica de comandos `enquanto/faça`. Por exemplo, no trecho de programa a seguir, a leitura e a escrita de um valor são repetidas até que seja lido o valor zero. A variável lógica `segue` é inicializada com verdadeiro antes do comando `enquanto/faça` e é testada a cada repetição, tornando-se falsa no momento em que é lido o valor zero:

```
segue ← verdadeiro
enquanto segue faça
  início
  ler(valor)
  se valor ≠ 0
  então escrever(valor)
  senão segue ← falso
  fim
```

5.3.1 sinalização de final de dados

Para exemplificar o uso do comando `enquanto/faça`, considere uma situação em que o número de alunos que terão as notas processadas é desconhecido. Neste caso, como saber quando encerrar o processo de leitura de notas, cálculo de média e informação da média obtida pelo aluno? O encerramento de um comando de repetição condicional que inclui a entrada dos dados a serem processados pode ser implementado de três formas, apresentadas a seguir.

marca de parada no último dado válido. Define-se, dentre os valores a serem lidos, qual o último valor que deve ser processado, valor esse que funciona como marca de parada. Como essa marca de parada é um dado válido, só depois de processá-la e de processar os demais dados a ela associados é que o processamento deve ser encerrado. No exemplo das notas de alunos, esse controle poderia ser o código do último aluno a ser processado, como mostrado no Algoritmo 5.3, que lê as notas e o código de um conjunto de alunos e informa as suas médias:

```
Algoritmo 5.3 - MédiaAlunos_1
{INFORMA MÉDIA DOS ALUNOS DE UMA TURMA}
{CONDIÇÃO DE PARADA: CÓDIGO DO ÚLTIMO ALUNO}
  Entradas: nota1, nota2, nota3 (real)
            código, cod_último (inteiro)
  Saídas: média (real)
início
  ler(cod_último)              {ENTRADA DO CÓDIGO DO ÚLTIMO ALUNO}
  código ← 0 {INICIALIZA CÓDIGO SÓ PARA COMPARAR A PRIMEIRA VEZ}
  enquanto código ≠ cod_último faça
    início
    ler(código)                                {LÊ CÓDIGO DO ALUNO}
    ler(nota1, nota2, nota3)                 {ENTRADA DAS 3 NOTAS}
    média ← (nota1 + nota2 + nota3) / 3              {CALCULA MÉDIA}
    escrever(código, média) {INFORMA CÓDIGO DO ALUNO E SUA MÉDIA}
    fim
fim
```

marca de parada após os dados válidos. É definido um valor de parada que não constitui um dado válido. Esse valor não deve ser processado, funcionando somente como indicação de parada das repetições. Por exemplo, no caso dos alunos pode ser uma primeira nota negativa:

```
Algoritmo 5.4 - MédiaAlunos_2
{INFORMA MÉDIA DOS ALUNOS DE UMA TURMA}
{CONDIÇÃO DE PARADA: PRIMEIRA NOTA LIDA É NEGATIVA}
  Entradas: nota1, nota2, nota3 (real)
```

```
          código (inteiro)
  Saídas: média (real)
início
  ler (nota1, nota2, nota3)                    {ENTRADA DE 3 NOTAS}
  enquanto nota1 ≥ 0 faça
    início
    ler(código)                                {LÊ CÓDIGO DO ALUNO}
    média ← (nota1 + nota2 + nota3) / 3          {CALCULA MÉDIA}
    escrever(código, média) {INFORMA CÓDIGO DO ALUNO E SUA MÉDIA}
    ler(nota1, nota2, nota3)       {ENTRADA DAS PRÓXIMAS 3 NOTAS}
    fim
fim
```

Observar que foi necessário ler as notas do primeiro aluno antes de entrar no comando de repetição para que o teste do comando `enquanto/faça` pudesse ser realizado adequadamente já na sua primeira execução. Ao final do processamento das notas de um aluno, são lidas as do próximo, devolvendo o controle ao comando `enquanto/faça` para que seja realizado novamente o teste da primeira nota, definindo se vai ser realizada nova repetição ou se o comando deve ser terminado. Assim tem-se a garantia de que não são processadas, como se fossem dados válidos, as notas que contêm a marca de parada.

parada solicitada pelo usuário. Ao final de cada iteração, o usuário decide se deseja continuar ou parar, respondendo a uma pergunta explícita, conforme mostrado no Algoritmo 5.5:

```
Algoritmo 5.5 – MédiaAlunos_3
{INFORMA MÉDIA DOS ALUNOS DE UMA TURMA}
{CONDIÇÃO DE PARADA: INFORMADA PELO USUÁRIO}
  Entradas: nota1, nota2, nota3 (real)
            código (inteiro)
            continuar (caractere)
  Saídas: média (real)
início
  continuar ← 'S'{INICIALIZA CÓDIGO PARA COMPARAR A PRIMEIRA VEZ}
  enquanto continuar = 'S' faça
    início
    ler(código)                                 {LÊ CÓDIGO DO ALUNO}
    ler(nota1, nota2, nota3)                  {ENTRADA DAS 3 NOTAS}
    média ← (nota1 + nota2 + nota3) / 3           {CALCULA MÉDIA}
    escrever(código, média) {INFORMA CÓDIGO DO ALUNO E SUA MÉDIA}
    escrever('Mais alunos? Responda S ou N ')
    ler(continuar)           {USUÁRIO INFORMA SE TEM MAIS ALUNOS}
    fim
  fim
```

5.3.2 contagem de repetições

Caso se necessite saber quantas repetições foram realizadas, uma vez que esse número é desconhecido, é preciso fazer uso de uma variável do tipo `contador`, incrementada dentro do laço a cada iteração. O algoritmo a seguir estende o Algoritmo 5.4, informando também a média da turma. Para o cálculo dessa média é necessário conhecer o número de alunos, informado através do contador `cont_al`:

```
Algoritmo 5.6 - MédiaAlunoETurma_2
{INFORMA MÉDIA DOS ALUNOS DE UMA TURMA E A MÉDIA GERAL DESSA TURMA.
PARA INDICAR FIM DE PROCESSAMENTO, O CONTEÚDO INFORMADO EM NOTA1 SERÁ
NEGATIVO}
  Entradas: nota1, nota2, nota3 (real)
  Saídas: média (real)
          soma_médias (real)
          média_turma (real)
  Variável auxiliar:
      cont_al (inteiro) {CONTADOR DE ALUNOS PROCESSADOS}
início
  soma_médias ← 0  {SOMA MÉDIAS INDIVIDUAIS: VALOR INICIAL ZERO}
  cont_al ← 0           {CONTADOR DE ALUNOS: VALOR INICIAL ZERO}
  ler(nota1, nota2, nota3)        {ENTRADA DAS 3 PRIMEIRAS NOTAS}
  enquanto nota1 ≥ 0 faça
    início
    cont_al ← cont_al + 1                 {CONTA ALUNO LIDO}
    média ← (nota1 + nota2 + nota3) / 3      {CALCULA MÉDIA}
    escrever(cont_al, média)                 {INFORMA MÉDIA}
    soma_médias ← soma_médias + média      {SOMA DAS MÉDIAS}
    ler(nota1, nota2, nota3)   {ENTRADA DAS PRÓXIMAS 3 NOTAS}
    fim {DO ENQUANTO}
  média_turma ← soma_médias / cont_al       {MÉDIA DA TURMA}
  escrever(média_turma)
fim
```

5.3.3 comandos de repetição aninhados

Assim como para o comando `para/faça`, o comando incluído dentro de um laço de repetições do comando `enquanto/faça` pode ser um comando qualquer, inclusive outro comando de repetição `para/faça` ou `enquanto/faça`. O exemplo a seguir mostra o aninhamento de um comando `para/faça` dentro de um `enquanto/faça`. A variável `mais_um` é do tipo `caractere`. Os comandos do laço do `enquanto/faça` serão repetidos enquanto não for lido um caractere "N". A cada repetição, todo o comando `para/faça` é executado:

```
ler(mais_um)
enquanto mais_um ≠ 'N' faça
  início
  ler(lim_sup)
  para i de 1 incr 1 até lim_sup faça
    escrever(i)
  ler(mais_um)
  fim
```

5.4 → comando de repetição condicional `repita/até` por avaliação posterior de condição

O **comando de repetição condicional** por avaliação posterior **`repita/até`** também vincula a execução de um conjunto de comandos ao resultado da avaliação de uma expressão lógica. O comando inicia pela execução do laço e, quando essa execução é concluída, a expressão é avaliada: se o valor lógico obtido for falso, o laço é executado novamente; se for verdadeiro, o comando é encerrado. Isso significa que o laço é sempre executado pelo menos uma vez, independentemente do valor lógico inicial resultante da avaliação da expressão de controle. Observar que, normalmente, o valor inicial da expressão lógica será falso, pois se deseja repetir o laço mais de uma vez. Portanto, é necessário que, em algum momento, o conteúdo de alguma variável utilizada nesta expressão lógica tenha o valor alterado dentro do laço, de forma a modificar o valor resultante de sua avaliação para verdadeiro, evitando assim a ocorrência de um laço – *loop* – infinito.

A sintaxe de um comando de repetição condicional `repita/até` é a seguinte:

```
repita
   <comandos>
até <expressão lógica>
```

Observar que, diferentemente dos comandos anteriores, aqui não é necessário um comando composto, pois a sintaxe aceita múltiplos comandos, delimitados pela cláusula `até`.

O fluxograma representado na Figura 5.5 mostra o funcionamento desse comando, em que o laço de repetição é sempre executado pelo menos uma vez.

O aninhamento de comandos de repetição também se aplica ao comando `repita/até`, incluindo os outros comandos de repetição já vistos.

O Algoritmo 5.7, a seguir, adapta o Algoritmo 5.6, utilizando no laço de repetição um comando `repita/até` em lugar do `enquanto/faça`. Observar que, como o laço desse comando é sempre executado pelo menos uma vez, se tornou necessária a inclusão de um comando condicional logo no início para condicionar a execução do laço ao valor inicial de `nota1`.