Introdução à Computação

Rosane Minghim e Guilherme P. Telles

9 de Agosto de 2012

Capítulo 4

Subprogramas

4.1 Introdução

No desenvolvimento de um algoritmo qualquer, como já foi mencionado, é útil subdivir uma tarefa em várias, e desenvolvê-las individualmente. Cada uma dessas tarefas pode ser executada por um subprograma específico, que recebe os dados produzidos na etapa anterior e produz os resultados necessários para a etapa seguinte do algoritmo.

A utilização de subprogramas no desenvolvimento de sistemas computacionais é uma tarefa corriqueira e deve fazer parte do costume de desenvolvimento de qualquer programador.

Um exemplo de subprograma é qualquer das funções pré-definidas no Capítulo 2. Cada uma delas é um subprograma. Por exemplo, quando a função `seno` é acionada, ela recebe o dado que é fornecido a ela (no caso, o valor do ângulo), executa uma tarefa (o cálculo do seno) e produz um resultado (o valor do seno daquele ângulo). Com isso não é necessário embutir o código do cálculo do seno no algoritmo que necessita empregar este valor, simplificando o código e o entendimento do algoritmo original, e também tornando todo cálculo de seno uniformemente consistente.

Além da vantagem da consitência e do apoio ao desenvolvimento de programas, o uso de subprogramas evita repetição de trechos de código no caso de determinadas tarefas serem realizadas repetidas vezes.

Diz-se que um algoritmo que utiliza o subprograma '**chama**' o subprograma. Um subprograma possui uma lista de **parâmetros**, que são os dados que ele troca com o algoritmo que o utiliza. Um subprograma pode ser executado quantas vezes for necessário, e a partir de qualquer parte do algoritmo, cada vez com um conjunto de parâmetros diferentes, se necessário. Ou seja, ele é adaptado a receber qualquer dado de tipo compatível com as definições

de tipo dos seus parâmetros. Por exemplo, ora o `seno` calculado é de um ângulo, ora de outro, conforme a necessidade.

Por essas razões um subprograma deve ser genérico o suficiente para poder ser utilizado a partir de dados distintos e algoritmos diversos, produzindo o resultado esperado. A seguir utilizamos um exemplo de algoritmo apresentado no Capítulo 3 para ilustrar a utilidade do emprego de subprogramas.

4.2 Subprogramas: exemplo introdutório

No Capítulo 3 foi apresentado como exemplo o cálculo das médias, em que a média final de um aluno num curso foi calculada com base em três notas de prova (com uma possível substitutiva) e três notas de trabalho (ver Exercício Resolvido 1). O algoritmo final é repetido abaixo para referência. Nele observa-se um trecho de algoritmo (linhas 17 a 34), onde é calculada a menor dentre as três notas, para que ela possa posteriormente ser substituida.

```
1  Algoritmo médias
2    variável
3      menor_prova: real
4      prova1, prova2, prova3, substitutiva: real
5      trabalho1, trabalho2, trabalho3: real
6      resposta: caracter
7      média_prova, média_trabalho, média_final: real
8
9    {Leia o valor das notas das três provas}
10   leia (prova1, prova2, prova3)
11
12   leia (resposta)
13   se (resposta = 's') ou (resposta = 'S') então
14     {Leia o valor da prova substitutiva}
15     leia(substitutiva)
16
17     {Obtenha o valor da menor prova}
18     se prova1 ≤ prova2 então
19        se (prova2 ≤ prova3)  então
20          menor_prova ← prova1
21        senão
22          se prova1<prova3 então
23            menor_prova ← prova1
24          senão
```

```
25              menor_prova ← prova3
26            fim se
27          fim se
28        senão
29          se (prova2 ≤ prova3) então
30            menor_prova ← prova2
31          senão
32            menor_prova ← prova3
33          fim se
34        fim se
35        {Calcule a média de provas - com substitutiva}
36        se menor_prova = prova1 então
37          média_prova ← (substitutiva + prova2 + prova3)/3
38        senão
39          se menor_prova = prova2 então
40            média_prova ← (prova1 + substitutiva + prova3)/3
41          senão
42            média_prova ← (prova1 + prova2 + substitutiva)/3
43          fim se
44        fim se
45      senão
46        {Calcule a média de provas - sem substitutiva}
47        média_prova ← (prova1 + prova2 + prova3)/3
48      fim se
49      {Leia o valor das notas de trabalho}
50      leia(trabalho1,trabalho2,trabalho3)
51      {Calcule a média dos trabalhos}
52      média_trabalho ← (trabalho1 + trabalho2 + trabalho3)/3
53      {Calcule a média final}
54      se  (média_trabalho ≥  5,0) e (média_prova ≥ 5,0) então
55         média_final ← média_prova*0,7 + média_trabalho*0,3
56      senão
57        se média_trabalho < média_prova então
58          média_final ← média_trabalho
59        senão
60          média_final ← média_prova
61        fim se
62      fim se
63      {Imprima o resultado}
64      escreva(prova1,prova2,prova3,substitutiva,
65      média_prova,média_trabalho,média_final)
```

```
66 fim
```

Suponha agora que o critério mude, e que também seja possível, para um aluno qualquer, apresentar um trabalho substitutivo cuja nota substitua a menor nota entre as três de trabalho.

Para introduzir essa mudança no algoritmo acima, seria também necessário determinar a menor nota de trabalho, antes de calcular a média dos trabalhos (linha 52). Nesse ponto do algoritmo, um pseudo-código idêntico àquele usado para calcular a menor nota de prova teria que ser repetido, agora usando as variáveis `trabalho1`, `trabalho2` e `trabalho3`, para obter o menor valor de trabalho.

Tanto a consistência dessa nova versão do algoritmo quanto sua clareza e compacidade seriam beneficiadas pelo uso de um subprograma capaz de determinar o menor entre três números reais. Este subprograma admitiria três valores reais quaisquer e teria como resultado um valor igual ao menor entre os três. Dessa forma, o subprograma admitiria como parâmetros hora as variáveis `prova1`, `prova2` e `prova3`, e hora as variáveis `trabalho1`, `trabalho2` e `trabalho3`.

A seguir, uma versão para esse subprograma é apresentada com a notação do pseudo-código, que será formalizada na Seção 4.4.

Exemplo 4.1

```
Subprograma menor(valor1,valor2,valor3):real

e:valor1:real
  valor2:real
  valor3:real
r:o menor valor entre valor1, valor2 e valor3, real.

variável
  menor_valor: real

início
  se valor1 ≤ valor2 então
    se (valor2 ≤ valor3)  então
      menor_valor ← valor1
    senão
      se valor1<valor3 então
        menor_valor ← valor1
      senão
        menor_valor ← valor3
```

```
      fim se
    fim se
  senão
    se (valor2 ≤ valor3) então
      menor_valor ← valor2
    senão
      menor_valor ← valor3
    fim se
  fim se
  retorne (menor_valor)
fim
```

O subprograma acima define, genericamente, um algoritmo que calcule o menor entre três valores reais, e seu uso ocorre como se fosse uma das funções pré-definidas, ou seja, ao chamar:

```
menor(x,y,z)
```

o menor valor entre `x`, `y` e `z` é retornado, da mesma forma que uma função pré-definida retorna um valor, como seno ou raiz quadrada de um parâmetro qualquer. O resultado pode ser usado na atribuição a uma variável, dentro de uma expressão, no meio dos parâmetros de impressão, etc..

Na notação do subprograma do Exemplo 4.1, a cláusul indicada por `e:` lista os parâmetros que o subprograma recebe como dados de entrada e a cláusula `r:` expressa o que ele retorna como resultado.

Com o uso desse subprograma, o algoritmo original do cálculo das médias (Exercício Resolvido 1) do Capítulo 3 fica alterado da forma apresentada no algoritmo abaixo (note o comando da linha 17).

Exemplo 4.2

```
1  Algoritmo médias
2    variável
3      menor_prova: real
4      prova1, prova2, prova3, substitutiva: real;
5      trabalho1, trabalho2, trabalho3: real;
6      resposta: caracter;
7      média_prova, média_trabalho, média_final: real;
8
9    {Leia o valor das notas das três provas}
10   leia (prova1, prova2, prova3)
11
```

```
12    leia (resposta)
13    se (resposta = 's') ou (resposta = 'S') então
14      {Leia o valor da prova substitutiva}
15      leia(substitutiva)
16      {Obtenha o valor da menor prova}
17      menor_prova ← menor(prova1,prova2,prova3)
18      {Calcule a média de provas - com substitutiva}
19      se menor_prova = prova1 então
20        média_prova ← (substitutiva + prova2 + prova3)/3
21      senão
22        se menor_prova = prova2 então
23          média_prova ← (prova1 + substitutiva + prova3)/3
24        senão
25          média_prova ← (prova1 + prova2 + substitutiva)/3
26        fim se
27      fim se
28    senão
29      {Calcule a média de provas - sem substitutiva}
30      média_prova ← (prova1 + prova2 + prova3)/3
31    fim se
32    {Leia o valor das notas de trabalho}
33    leia(trabalho1,trabalho2,trabalho3)
34    {Calcule a média dos trabalhos}
35    média_trabalho ← (trabalho1 + trabalho2 + trabalho3)/3
36    {Calcule a média final}
37    se  (média_trabalho ≥  5,0) e (média_prova ≥ 5,0) então
38       média_final ← média_prova*0,7 + média_trabalho*0,3
39    senão
40      se média_trabalho < média_prova então
41        média_final ← média_trabalho
42      senão
43        média_final ← média_prova
44      fim se
45    fim se
46
47    {Imprima o resultado}
48    escreva(prova1,prova2,prova3,substitutiva,média_prova,
49            média_trabalho,média_final)
50  fim
```

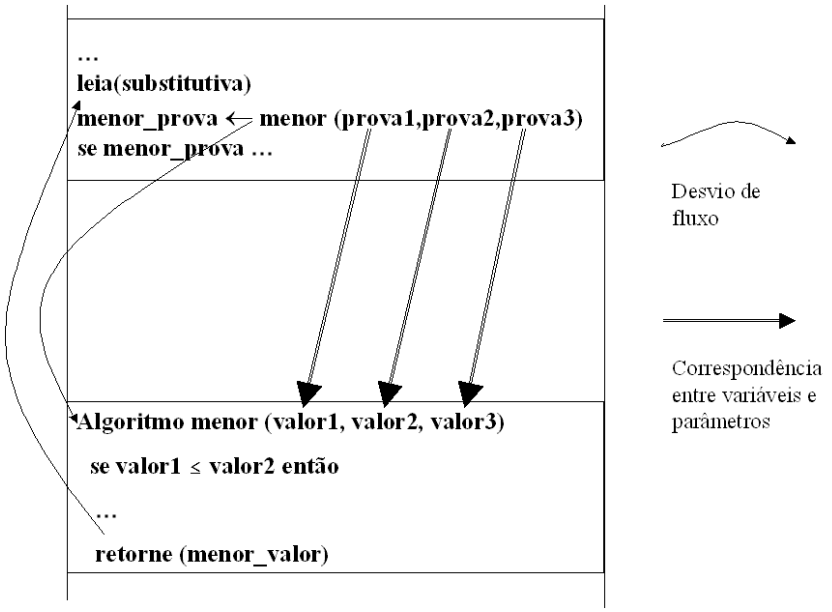

Figura 4.1: Ilustração do conceito de desvio do controle para um subprograma

O algoritmo incluindo a mudança sugerida acima, referente ao trabalho substitutivo, é resolvido no final deste capítulo.

Um subprograma funciona a partir do algoritmo que o chamou como um comando, que desvia o fluxo de execução do algoritmo que chamou para o algoritmo do subprograma. Neste ponto, o algoritmo do subprograma é executado, enquanto o algoritmo que aciona o subprograma fica à espera do resultado. É como se existissem dois ambientes: o do algoritmo que 'chamou' e o do algoritmo que 'é chamado'. No ambiente que chama o subprograma, valem as variáveis e outras definições deste ambiente. Quando o subprograma está em ação o ambiente muda e novas declarações e variáveis estão valendo. A partir do retorno do subprograma (cláusula `fim` ou comando `retorna`), o ambiente do subprograma deixa de existir e volta a valer o ambiente do algoritmo que acionou o subprograma. A Figura 4.1 ilustra esse mecanismo de troca de ambientes durante o acionamento de um subprograma, através do exemplo do algoritmo `médias` que aciona o algoritmo `menor` na sua linha 17.

Pode-se notar, pela figura, que os nomes das variáveis do subprograma são distintos dos nomes das variáveis no algoritmo que o chamou. Isso é natural, uma vez que um subprograma pode ser desenvolvido independentemente de qualquer programa que o utilize, como é o caso típico do subprograma `menor`.

A correspondência entre as variáveis dos dois algoritmos (o que chama e o que é chamado) é feita no momento da chamada. A lista de parâmetros

da chamada possui o mesmo número de variáveis da lista de parâmetros do subprograma. No momento da chamada, as variáveis são correspondidas umas com as outras, na ordem em que aparecem na lista de parâmetros. Por exemplo, na chamada:

```
menor_prova ← menor(prova1,prova2,prova3)
```

O conteúdo das variáveis `prova1`, `prova2`, e `prova3` são 'passados' para o subprograma `menor`, de forma que, no início do subprograma, o valor da váriavel `valor1` é o mesmo que o valor de `prova1`, o da variável `valor2` é o mesmo que `prova2`, e o valor da variável `valor3` é o mesmo que o da variável `prova3`.

Por exemplo, se no momento da chamada `prova1` = 6,0, `prova2` = 3,0 e `prova3` = 8,0, então o subprograma menor começa com `valor1` = 6,0, `valor2` = 3,0 e `valor3` = 8,0, retornando o resultado 3,0 como resposta. Após a linha 17 do algoritmo do Exemplo 4.2, então, o valor da variável `menor_prova` seria 3,0.

4.3 Parâmetros

Um subprograma tem o objetivo de executar uma tarefa específica, dado um conjunto de dados de entrada, produzindo um resultado. Dados de entrada e de saída dos subprogramas são chamados parâmetros e são considerados variáveis locais dentro do subprograma. Os parâmetros de entrada de um subprograma não devem ser alterados no código do subprograma. Os parâmetros de saída são alterados no código para que possam ecoar seus resultados para o exterior (o algoritmo que chamou o subprograma). Alguns parâmetros são chamados de parâmetros de entrada e saída. Eles têm o efeito de carregar para o interior do subprograma alguma informação do ambiente externo, e retornam com um novo valor, determinado pelo próprio subprograma.

Como exemplo, supõe-se que os funcionários de um setor de uma empresa serão submetidos a um teste sobre o funcionamento daquele setor para efeito de promoções. Cada uma das questões do teste é de múltipla escolha (A até E). Uma questão correta adiciona um certo número de pontos (dependendo da questão) e uma questão errada decresce um número de pontos obtidos pelo candidato (dependendo da questão). Uma questão não respondida decresce um ponto.

Um subprograma que altere o valor obtido por um candidato, acrescendo ou decrescento o número adequado de pontos de acordo com a resposta de uma questão, poderia ter os seguintes parâmetros:

Entrada: o número de pontos obtidos até o momento pelo candidato (nas questões anteriores), o número da questão e a resposta fornecida pelo candidato.

Saída: o número de pontos do candidato após conferência do acerto ou não da questão.

O número de pontos, então, é parâmetro de entrada e saída, pois entra com um valor (número de pontos até o momento), e sai com outro (número de pontos após conferência da questão atual). Além disso, um parâmetro lógico de saída poderia sinalizar se a questão está correta ou não.

O cabeçalho do subprograma poderia ser:

```
calcule_pontos (questão,resposta,pontos,acerto)
```

onde: `questão` e `resposta` são parâmetros de entrada, indicando o número da questão respondida e a resposta fornecida; o parâmetro `pontos` é de de entrada e saída; possui, na entrada, o valor de pontos obtidos até o momento; na saída, está atualizado com o número de pontos após verificação da questão. o parâmeto `acerto` é de saída, e indica se aquela questão em particular está correta (verdadeiro) ou errada (falso).

Na chamada deste subprograma o resultado não é atribuído a nenhuma variável, uma vez que os resultados que ele gera são ecoados no valor das variáveis `pontos` e `acerto` usadas como parâmetros, as quais voltam do subprograma alteradas. Nada é retornado no nome do subprograma.

O exemplo abaixo fornece a definição do subprograma `calcule_pontos`.

Exemplo 4.3

```
Subprograma calcule_pontos (questão, resp, pontos, correta)

e: questão: inteiro {número da questão respondida}
   resp: caracter {resposta dada pelo candidato}

e/s: pontos {pontos obtidos anteriormente pelo candidato,
     atualizado pelo número de pontos conferido pela questão }

s: correta: lógico {retorna verdadeiro se a resposta
             à questão é correta}

início
  se resp = 'F' então
    correta ← falso
    pontos ← pontos - 1
```

```
senão
  escolha questão entre
  1:
    Se resp = 'C' então
      pontos ← pontos + 15
      correta ← verdadeiro
    senão
      pontos ← pontos - 10
      correta ← falso
    fim se
  2:
    Se resp = 'A' então
      pontos ← pontos + 15
      correta ← verdadeiro
    senão
      pontos ← pontos - 10
      correta ← falso
    fim se
  3:
    Se resp = 'B' então
      pontos ← pontos + 15
      correta ← verdadeiro
    senão
      pontos ← pontos - 5
      correta ← falso
    fim se
  4:
    Se resp = 'A' então
      pontos ← pontos + 15
      correta ← verdadeiro
    senão
      pontos ← pontos - 5
      correta ← falso
    fim se
  5:
    Se resp = 'B' então
      pontos ← pontos + 10
      correta ← verdadeiro
    senão
      pontos ← pontos - 5
      correta ← falso
```

```
  fim se
6:
  Se resp = 'C' então
    pontos ← pontos + 10
    correta ← verdadeiro
  senão
    pontos ← pontos - 5
    correta ← falso
  fim se
7:
  Se resp = 'D' então
    pontos ← pontos + 10
    correta ← verdadeiro
  senão
    pontos ← pontos - 5
    correta ← falso
  fim se
8:
  Se resp = 'D' então
    pontos ← pontos + 10
    correta ← verdadeiro
  senão
    pontos ← pontos - 5
    correta ← falso
  fim se
9:
  Se resp = 'C' então
    pontos ← pontos + 5
    correta ← verdadeiro
  senão
    pontos ← pontos - 5
    correta ← falso
  fim se
10:
  Se resp = 'A' então
    pontos ← pontos + 5
    correta ← verdadeiro
  senão
    pontos ← pontos - 2
    correta ← falso
  fim se
```

```
    fim escolha
  fim se
fim
```

Em resumo, exitem várias possibilidades quanto a passagem de parâmetors em subprogramas expressos em pseudo-código.

- Parâmetros podem ser:
  - De entrada: Seus valores são utilizados no subprograma mas não são modificados por ele (ex: parâmetros `valor1, valor2` e `valor3` no subprograma `menor`, Exemplo 4.1).
  - De saída: Carregam resultados do subprograma para o ambiente que o chamou, mas seu valor no início do subprograma é irrelevante. É o caso do parâmetro `correta` do subprograma `calcule_pontos`, Exemplo 4.3.
  - De entrada e saída: Seus valores são utilizados e alterados pelo subprograma, como o parâmetro `pontos` do subprograma `calcule_pontos`, Exemplo 4.3.
- Subprogramas podem retornar resultados nos seus nomes, e nesse caso a chamada normalmente é empregada numa expressão, como no uso do subprograma `menor` abaixo:

  ```
  menor_prova ← menor(prova1,prova2,prova3)
  ```

- Subprogramas podem não retornar resultados nos seus nomes. Nesse caso, os resultados são transmitidos apenas pela alteração das variáveis utilizadas como parâmetros, e a chamada é feita numa linha de comando independente, como no caso da chamada do subprograma `calcule_pontos` abaixo:

  ```
  calcule_pontos (questão,resposta,pontuação,acerto)
  ```

Os subprogramas que retornam valor no seu nome são chamados de **função**, a exemplo de todas aquelas pré-definidas apresentadas no Capítulo 2 e da função `menor` do Exemplo 4.1. Os que não retornam valores no seu nome são usualmente chamados de **procedimentos**.

Fica claro que um tipo de subprograma pode sempre ser convertido no outro. No entanto, uma função é mais adequada quando existe um único

resultado retornando do subprograma, e esse resultado pode ser representado através de um tipo simples. Quando o resultado tende a ser usado em expressões, é mais ainda indicado o uso de funções.

Um procedimento é mais adequado quando existe mais de um parâmetro de saída (incluindo aí parâmetros de entrada e saída), ou quando ele é primariamente usado para organizar um algoritmo, expressando um passo de nível de detalhamento mais alto.

A notação usada em pseudo-código para expressar os vários itens de procedimentos e funções é formalizada a seguir.

4.4 Notação de subprogramas em pseudo-código

Os dois tipos de subprograma admitidos em pseudo-código (respectivamente procedimento e função) são especificados nos formatos abaixo:

```
Subprograma nome (parâmetro, parâmetro, ..., parâmetro)

e:   parâmetro: tipo
     parâmetro: tipo
     parâmetro: tipo
     ...

s:   parâmetro: tipo
     parâmetro: tipo
     parâmetro: tipo
     ...

e/s: parâmetro: tipo
     parâmetro: tipo
     parâmetro: tipo
     ...

declarações

início
  comandos do algoritmo do subprograma
fim
```

e

```
Subprograma nome (parâmetro, parâmetro, ..., parâmetro):tipo

e:   parâmetro: tipo
     parâmetro: tipo
     parâmetro: tipo
     ...

s:   parâmetro: tipo
     parâmetro: tipo
     parâmetro: tipo
     ...

e/s: parâmetro: tipo
     parâmetro: tipo
     parâmetro: tipo
     ...

r: tipo

declarações

início
  comandos do algoritmo do subprograma
  retorne(expressão)
fim
```

onde:

- e: indica a lista de parâmetros de entrada
- s: indica a lista de parâmetros de saída
- e/s: indica a lista de parâmetros de entrada e saída
- r: indica o tipo retornado por uma função

Muito embora tanto funções quanto procedimentos possam ter qualquer número de parâmetros de entrada, parâmetros de saída e parâmetros de entrada e saída, conforme mencionado anteriormente, esses dois tipos de

subprogramas se prestam a tarefas ligeiramente diferentes, além do objetivo comum de realizar um passo individual de um algoritmo mais complexo. As próximas recomendações destacam as aplicações mais indicadas para funções e procedimentos, incluindo um exemplo adicional.

Sugestão 6 *Use um subprograma do tipo função quando:*

- *Existe um único resultado do subprograma*[1] *e ele é expresso na forma de tipo simples.*
- *O resultado tende a ser usado em expressões ou em chamadas de outros procedimentos e funções.*
- *O resultado é do tipo lógico e tende a ser usado para controlar um processo repetitivo.*

O Exemplo 4.4 apresenta um algoritmo que ilustra o último item da Recomendação 6.

Exemplo 4.4

Suponha que um trecho de algoritmo vai ser executado até que o usuário digite um valor fora de uma faixa de valores 'aceitáveis'. Os extremos dessa faixa são controlados por duas constantes, definidas da seguinte forma:

```
constante
  INI_FAIXA = -600
  FIM_FAIXA = 1200
```

Uma função para testar esta condição seria dada pelo seguinte pseudo-código:

```
Subprograma é_válido (val): lógico

e:val:inteiro {valor a ser verificado}

r: lógico {verdadeiro se valor val é considerádo válido e falso
caso contrário}

início
```

[1]Exceção feita a parâmetros que indiquem tratamento de erros e exceções. Este tema será tratado em maior detalhe em capítulo futuro.

```
  retorne(val ≥ INI_FAIXA e val ≤ FIM_FAIXA)
fim
```

Na função é_válido*, se a expressão lógica for verdadeira é retornado* verdadeiro *e se ela for falsa é retornado* falso*. Assim, o algoritmo que tem a repetição controlada pela condição testada por esta função assumiria a forma:*

```
  ...

  leia(valor_entrada)
  enquanto é_válido(valor_entrada) faça
    {comandos repetidos do algoritmo}
    ...

    leia(valor_entrada)
  fim enquanto
```

Exercício Sugerido 12 *Usando o esquema acima, re-escreva o algoritmo do Exemplo 3.12, re-escrevendo também a função* é_válido*.*

Sugestão 7 *Use um subprograma do tipo procedimento quando:*

- *Existem vários parâmetros de saída como resultado de um subprograma.*
- *O subprograma implementa um passo de algoritmo de um nível de refinamento inicial, ou seja, que ainda será subdividido em vários outros passos.*
- *O(s) resultado(s) do subprograma são representados por tipos de dados complexos (compostos ou heterogêneos), ou por parâmetro(s) de entrada e saída.*

Em princípio um subprograma é auto-contido, isto é, uma vez definido seu objetivo, qualquer programa que precise daquela tarefa que ele executa pode empregá-lo. Alguns subprogramas, no entanto, têm tarefas muito específicas de um problema particular que o algoritmo pretende resolver, e assumem uma característica dedicada à aplicação específica daquele algoritmo.

Na próxima seção são apresentados algoritmos baseados em subprogramas, na forma de exercícios resolvidos e sugeridos. Eles têm o objetivo de reforçar os conceitos ilustrados e formalizados acima.

4.5 Exercícios e Exemplos

Primeiramente, o algoritmo do cálculo das médias desenvolvido no Capítulo 3 é repetido aqui para ilustrar um possível desenvolvimento do mesmo problema tendo por base o uso de subprogramas capazes de resolver diversas partes do algoritmo. O enunciado é ligeiramente modificado para incluir a modificação sugerida na Seção 4.2.

Exercício Resolvido 5 *Cálculo da média de um aluno em um curso.*

Enunciado*: Calcular a média final de um aluno em um curso. No curso, existem três notas de prova e três notas de trabalho, sendo que a menor das notas de prova pode ser substituída pela nota de uma prova substitutiva, que é opcional para o aluno. A menor nota de trabalho também pode ser substituída pela nota de um trabalho substitutivo opcional.*

As notas variam de 0,0 a 10,0. A média de prova (MP) é dada pela média aritmética entre as três notas de prova, e a média de trabalhos (MT) é dada pela média aritmética entre as três notas de trabalho.

A média final (MF) é então calculada pela seguinte regra:

- *MF = 0,7∗MP + 0,3∗MT, se MP e MT forem ambas maiores ou iguais a 5,0.*
- *MF = min(MP,MT) caso contrário.*

obs*: min(a,b) significa o menor valor entre a e b.*

O resultado do programa é uma linha impressa apresentando todas as notas para o aluno, incluindo a média final e as médias intermediárias (prova e trabalho).

Resolução 5 *O algoritmo abaixo representa uma possível sequência inicial de passos para a solução do problema acima.*

```
Algoritmo médias
  Leia o valor das notas das três provas
  Leia o valor da prova substitutiva
  Obtenha o menor valor de prova
  Calcule a média de provas
  Leia o valor das notas de trabalho
  Obtenha o menor valor de trabalho
  Calcule a média de trabalhos
  Calcule a média final
  Imprima o resultado
fim
```

Neste ponto, podemos resolver individualmente cada passo.

Os passos de leitura são ligeiramente simples, portanto eles podem ser resolvidos na última versão do algoritmo principal.

O passo `Obtenha o menor valor de prova` *implica em utilizar o subprograma* `menor(a,b,c)`*, desenvolvido na Seção 4.2, Exemplo 4.1. O mesmo ocorre para o passo* `Obtenha o menor valor de trabalho`*.*

Ambos os passos: `Calcule a média de provas` *e* `Calcule a média de trabalhos` *envolvem o cálculo de uma média aritmética. Para solução desse passo pode-se projetar uma função genérica para o cálculo da média aritmética de três valores reais. O subprograma para isso é dado a seguir:*

```
Subprograma média_aritmética (valor1, valor2, valor3):real

e: valor1, valor2, valor3: real
   {valores dos quais se deseja calcular a média aritmética}
r: média aritmética dos três parâmetros de entrada, real.

início
  retorne((valor1+valor2+valor3)/3)
fim
```

O passo `Calcule a média final` *envolve, entre outras coisas, definir o menor entre dois valores (médias de trabalho e prova). Para isso, pode ser desenvolvida uma função que determina o menor entre dois valores. O algoritmo abaixo é uma possível solução para essa função.*

```
Subprograma mínimo (valor1, valor2):real

e: valor1, valor2: real
   {valores dos quais se deseja calcular a média aritmética}
r: o menor valor entre valor1 e valor2, real.

início
  se valor1 ≤ valor2 então
     retorne(valor1)
  senão
     retorne(valor2)
  fim se
fim
```

Com base nessas funções e no algoritmo `menor` *desenvolvido na Seção 4.2, uma possível versão para o algoritmo principal do cálculo das médias seria:*

```
Algoritmo médias
  variável
    menor_prova, menor_trabalho: real
    prova1, prova2, prova3, substitutiva: real
    trabalho1, trabalho2, trabalho3, substitutivo: real
    resposta: caracter
    média_prova, média_trabalho, média_final: real

  {Este algoritmo utiliza as funções:
    mínimo(a,b), menor(a,b,c), média_aritmética(a,b,c)}

  {Leia o valor das notas das três provas}
  leia (prova1, prova2, prova3)

  leia (resposta)
  se (resposta = 's') ou (resposta = 'S') então
    {Leia o valor da prova substitutiva}
    leia(substitutiva)
    menor_prova ← menor(prova1,prova2,prova3)
    {Calcule a média de provas - com substitutiva}
    se menor_prova = prova1 então
      média_prova ← media_aritmética
           (substitutiva, prova2, prova3)
    senão
      se menor_prova = prova2 então
        média_prova ← média_aritmética
             (prova1, substitutiva, prova3)
      senão
        média_prova ← média_aritmética
            (prova1, prova2, substitutiva)
      fim se
    fim se
  senão
    {Calcule a média de provas - sem substitutiva}
    média_prova ← média_aritmética (prova1, prova2, prova3)
  fim se
  {Leia o valor das notas de trabalho}
  leia(trabalho1,trabalho2,trabalho3)
  {Calcule a média dos trabalhos}
  leia (resposta)
  se (resposta = 's') ou (resposta = 'S') então
```

```
    {Leia o valor do trabalho substitutivo}
    leia(substitutivo)
    menor_trabalho ← menor(trabalho1,trabalho2,trabalho3)
    {Calcule a média de trabalhos - com substitutivo}
    se menor_trabalho = trabalho1 então
      média_trabalho ←
           media_aritmética (substitutivo, trabalho2, trabalho3)
    senão
      se menor_trabalho = trabalho2 então
        média_trabalho ← média_aritmética
             (trabalho1, substitutivo, trabalho3)
      senão
        média_trabalho ← média_aritmética
             (trabalho1, trabalho2, substitutivo)
      fim se
    fim se
  senão
    {Calcule a média de trabalhos - sem substitutivo}
    média_trabalho ← média_aritmética
         (trabalho1, trabalho2, trabalho3)
  fim se
  {Calcule a média final}
  se  (média_trabalho ≥  5,0) e (média_prova ≥ 5,0) então
     média_final ← média_prova*0,7 + média_trabalho*0,3
  senão
    média_final ← mínimo(média_prova,média_trabalho)
  fim se
  {Imprima o resultado}
  escreva(prova1,prova2,prova3,substitutiva,média_prova,
          média_trabalho,média_final)
fim
```

Supondo agora que o critério de avaliação vai ser alterado para que se admita pesos 2, 3 e 3 para as provas, ficou muito mais fácil alterar o algoritmo das médias consistentemente. Sabe-se, olhando o algoritmo acima, que seria alterado apenas o trecho do cálculo das médias que, ao invés de empregar a função `média_aritmética`*, iria agora utilizar uma nova função,* `média_ponderada`*.*

Para generalizar essa função, admite-se que, para cada nota de prova passada como parâmetro seja passado também seu peso. O algoritmo pode ser expresso da seguinte forma:

```
Subprograma
média_ponderada(valor1,peso1,valor2,peso2,valor3,peso3):real

e:valor1,valor2,valor3: real
  {valores dos quais se deseja calcular a média}
  peso1,peso2,peso3:inteiro
  {pesos dos valores valor1, valor2 e valor3 respectivamente}
r:a média dos valores valor1, valor2 e valor3, ponderada pelos
  respectivos pesos, real

variável
  média:real {armazena temporariamente a média ponderada}

início
  média ← (valor1*peso1 + valor2*peso2 + valor3*peso3)/
             (peso1 + peso2 + peso3)
  retorne(média)
fim
```

Abaixo é apresentado o trecho do algoritmo principal modificado para atender a essa mudança de critério de prova.

```
Algoritmo médias

constante
  peso_prova1 = 2
  peso_prova2 = 2
  peso_prova3 = 3

  ...
  {Calcule a média de provas - com substitutiva}
  se menor_prova = prova1 então
    média_prova ← media_ponderada
           (substitutiva,peso_prova1,
           prova2,peso_prova2,
           prova3,peso_prova3)
  senão
    se menor_prova = prova2 então
      média_prova ← media_ponderada
           (prova1,peso_prova1,
           substitutiva,peso_prova2,
```

```
            prova3,peso_prova3)
    senão
      média_prova ← media_ponderada
            (prova3,peso_prova1,
            prova2,peso_prova2,
            substitutiva,peso_prova3)
    fim se
  fim se
  ...

fim
```

Exercício Sugerido 13 *O algoritmo médias apresentado acima está detalhado demais para um algoritmo principal. Idealmente, a leitura do algoritmo principal de uma tarefa como esta, que envolve várias tarefas menores, deve identificar claramente os passos de alto nível. Desenvolva os subprogramas adequados e complete a documentação abaixo para que o algoritmo principal do algoritmo do cálculo das médias se resuma ao algoritmo abaixo:*

```
Algoritmo médias

declarações

  leia_provas(prova1,prova2,prova3,substitutiva)
  média_prova ← calcule_média_prova(prova1,
                                    prova2,
                                    prova3,
                                    substitutiva)
  leia_trabalhos(trabalho1,trabalho2,trabalho3,substitutivo)
  média_trabalho ← calcule_média_trabalho(trabalho1,
                                    trabalho2,
                                    trabalho2,
                                    substitutivo)
  média_final ← calcule_média_final(media_prova,
                                    media_trabalho)
  impressões

fim
```

Exercício Resolvido 6 *Cálculo de uma nota fiscal.*

Enunciado*: Suponha que um programa vá calcular o total de uma nota fiscal. O usuário deve digitar, para cada item da nota fiscal, o número de itens*

adquiridos, e o valor unitário do item. Valores serão digitados pelo usuário até que ele forneça um valor zero para o número de itens. O programa deve calcular e imprimir o valor de cada item, e ir adicionando esse valor ao valor total da nota. O programa deve imprimir o valor total da nota fiscal no final do processo. O valor total de cada item possui desconto de 5% se o número de itens adquiridos ultrapassar 100 unidades. Se o total do valor do item ultrapassar R$ 200,00, existe um desconto de 10%. A cada item calculado, deve ser impresso, além do total do item, o valor do desconto em espécie.

Resolução 6 *Uma versão inicial para o algoritmo deste problema é dado a seguir:*

```
Algoritmo calcula_nota_fiscal

  total_da_nota ← 0,0
  leia(nro_de_itens)
  enquanto (nro_de_itens) ≠ 0 faça
    leia (valor_do_item)
    calcule_valor_do_item(nro_de_itens,valor_do_item,
                          total_do_item,desconto_do_item)
    escreva(valor_do_item, desconto_do_item)
    total_da_nota ← total_da_nota + total_do_item
    leia(nro_de_itens)
  fim enquanto
  escreva (total_da_nota)
fim
```

Pode-se observar, pelo algoritmo acima, que foram tomadas as seguintes decisões com relação ao subprograma `some_valor_do_item` *que apoia a execução do algoritmo:*

- *Ele calcula o valor total do item, portanto deve processar o cálculo dos descontos.*
- *Ele não faz impressões, devendo para isso devolver como resultado o valor do desconto aplicado para cada item calculado.*
- *Uma vez que existem dois valores de saída (o total do item e o desconto do item), ele não retorna nada no nome do subprograma, sendo, portanto, do tipo procedimento.*

Assim, o algoritmo do subprograma `calcule_valor_do_item` *pode ser expresso da seguinte forma:*

```
Subprograma
calcule_valor_do_item(quantidade,valor_unitário,total_item,desconto)

e: quantidade: inteiro, número de itens adquiridos
   valor_unitário: real, valor unitário do item, em reais.
s: total_item: real, valor total do item, em reais.
   desconto: real, desconto aplicado ao item, em reais.


variável
  desconto_quantidade, desconto_valor: real
       {auxiliares no cálculo do desconto}
constante
  fator_desconto_quantidade = 0,05
  fator_desconto_valor = 0,1

início
  desconto_quantidade ← 0,0
  desconto_valor ← 0,0
  total_item ← quantidade * valor_unitário
  se quantidade > 100 então
    desconto_quantidade ← total_item*0,05
    total_item ← total_item - desconto_quantidade
  fim se
  se total_item > 100,00 então
    desconto_valor ← total_item * 0,1
    total_item ← total_item - desconto_valor
  fim se
  desconto ← desconto_quantidade + desconto_valor
fim
```

Observa-se que as impressões necessárias são realizadas no algoritmo principal. Em princípio, essa é uma regra interessante a ser seguida, ou seja, evitar fazer impressões nos subprogramas que realizam cálculos. As impressões devem ser feitas nos algoritmos que controlam as chamada dos subprogramas (como o algoritmo principal), ou em subprogramas específicos (que só fazem escrita, não calculam nada). A mesma regra se aplica para as leituras.

Exercício Sugerido 14 *Alterar o programa acima para que ele imprima o total do desconto, em reais.*

Exercício Sugerido 15 *Documentar todos os algoritmos deste capítulo.*

Exercício Sugerido 16 *Transformar em subprogramas os algoritmos dos capítulos anteriores.*

Exemplo 4.5 Encadeamento de chamadas entre subprogramas

Tanto algoritmos principais quanto subprogramas podem chamar subprogramas. Este exemplo ilustra tal fato.

A algoritmo abaixo é de um subprograma que retorna verdadeiro se um conjunto de três valores está ordenado.

```
Subprograma está_ordenado(v1, v2, v3):lógico

e: v1: real {primeiro valor}
   v2: real {segundo valor}
   v3: real {terceiro valor}
r: {verdadeiro se estão ordenados
   falso caso contrário}

variável
  resultado: lógico

início
  resultado ← (v1 ≤ e v2 ≤ v3)
  retorne(resultado)
fim
```

O subprograma abaixo ordena um conjunto de dois valores.

```
Subprograma ordena_dois(v1, v2)

e/s: v1: real {na entrada, o primeiro valor,
              na saída o menor valor}
     v2: real {na entrada, o segundo valor,
              na saída o maior valor}

variável
  auxiliar: real

início

  se v2 < v1 então
```

```
     auxiliar ← v2
     v2 ← v1
     v1 ← auxiliar
  fim se
fim
```

O subprograma abaixo ordena um conjunto de três valores utilizando os subprogramas anteriores.

```
Subprograma ordena_três(v1, v2, v3)

e/s: v1, v2, v3: real
     {na entrada, três valores em qualquer ordem}
     {na saída, três valores em ordem crescente}

 {Usa os subprogramas está_ordenado(a,b,c) e
   ordena_dois(t,u)}

início
  se não está_ordenado(v1,v2,v3) então
     ordena_dois(v1,v2)
     ordena_dois(v2,v3)
     ordena_dois(v1,v2)
  fim se
fim
```

O algoritmo a seguir auxilia o teste do subprograma `ordena_três`.

```
Algoritmo Ordena_Trios

{Usa o subprogramas  ordena_três (x,y,z)}

constante

     max_repetições = 4

variável
     valor1,valor2,valor3:real
     repetição: inteiro

início
```

```
  leia(max_repetições)
  para repetição de 1 até  max_repetições faça
     leia(valor1,valor2,valor3)
     ordena_três(valor1,valor2,valor3)
     escreva(valor1,valor2,valor3)
  fim para
fim
```

Exercício Sugerido 17 *Qual a saída do programa acima para os seguintes dados de entrada:*
0 0 0
1 0 0
3 2 1
3 3 1
0 1 2
1 0 1
1 3 3
1 2 3

Exercício Sugerido 18 *Desenvolver um subprograma para converter um número binário de oito bits em um número inteiro.*

Desenvolver um subprograma para converter um número inteiro em um número binário de 8 bits.
Sugestão: os bits podem ser caracteres.