

Capítulo 6

Técnicas para a Solução de Problemas

Neste capítulo serão consolidadas as técnicas para a solução de problemas que podem ter tamanho arbitrário. Percebeu-se, por alguns exemplos e exercícios propostos nos capítulos anteriores, que a solução em um único fluxograma de problemas com complexidade média tornava árdua a sua representação, bem como seu entendimento. As técnicas apresentadas neste capítulo seguem o lema de “dividir para conquistar”, ou seja, “quebrar” uma solução única, complexa e difícil de entender, por um conjunto de soluções menores, inteligíveis, que juntas formam a solução final. Neste capítulo será apresentada a técnica *top-down* como aquela que irá permitir “dividir para conquistar”, bem como a divisão de um sistema em sub-routine, permitindo assim modularizar soluções por meio de funções e procedimentos.

6.1 A técnica *top-down*

A técnica *top-down* possui esse nome porque se **analisa** primeiramente um problema como um todo (*top*), identificando a seguir uma primeira divisão deste em um conjunto de subproblemas menores (*down*). O processo é realizado dessa forma até que não mais seja necessária nenhuma subdivisão.

Depois da identificação e solução dos subproblemas menores, percorre-se o caminho inverso na **síntese** da solução global: juntam-se as soluções menores obtidas de maneira ordenada até se formar a solução procurada.

Em síntese, essa técnica pode ser assim descrita:

1. Entender o problema a ser resolvido.
2. Estabelecer o objetivo a ser alcançado.
3. Dividir o problema (solução desconhecida) em problemas menores, com solução mais simples (ou conhecida) e cujo total permita atingir o objetivo.
4. Continuar a subdividir os problemas gerados até que seja possível solucionar a todos.
5. A solução do problema original é feita pela junção ordenada das soluções dos problemas finais.

Como motivação para a utilização dessa técnica, considere a criação de um projeto mecânico. Algumas pessoas poderiam desenvolver o projeto mostrando todos os detalhes em um único desenho. Sabe-se, porém, que isso é simples se o projeto for rudimentar, mas, caso a complexidade aumente, isso nunca funcionará.

O desenho de uma peça para posterior fabricação segue um processo conhecido. O desenho deve ser representado com uso de pelo menos três vistas: a planta, a elevação e a lateral. Qual a razão disso? A razão está em exibir os diferentes detalhes construtivos da peça de uma forma racional, simples e que não deixe margem a dúvidas.

Mesmo assim, existem situações em que surgem dúvidas; nesse caso deve-se efetuar cortes na peça, mostrando os detalhes de forma a torná-los evidentes e dirimir as eventuais dúvidas. Não exibir esses detalhes pode fazer com que o produto final não tenha a função originalmente projetada (a peça não serve para aquilo a que foi especificada) e expor todos os detalhes em um único desenho dificulta demais a visualização.

Na solução de problemas de âmbito computacional, ocorrem situações semelhantes. Com o crescimento do número de comandos a serem utilizados, pode-se facilmente perder a visão do todo, e aquele comando mal posicionado pode pôr a perder todo o trabalho de desenvolvimento (identificar um comando errôneo em um único e grande programa é como achar uma agulha no palheiro).

6.1.1 Exemplo de aplicação

Como exemplo de aplicação da técnica *top-down*, será analisado o seguinte problema: deseja-se construir um sistema automatizado para calcular as notas finais, em todas as disciplinas, de todos os alunos de uma escola. Como isso pode ser feito? A seguir, repetem-se os passos da técnica com foco neste problema.

Entendimento do problema

O problema exige que se tenha as notas de cada aluno por disciplina. Com esses dados, deve-se calcular e exibir as notas finais.

Estabelecer o objetivo a ser alcançado

O objetivo é exibir as notas finais de todos os alunos para todas as disciplinas da escola.

Divisão do problema

Analizando esse problema de maneira informal, apenas para se ter um primeiro contato com a técnica *top-down*, seria possível elaborar o fluxograma da Figura 6.1, representando o ponto de partida em busca da solução do problema das notas.

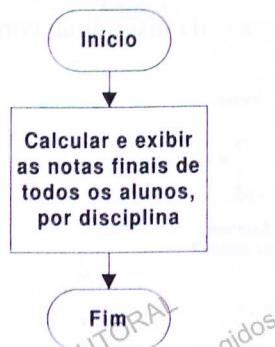
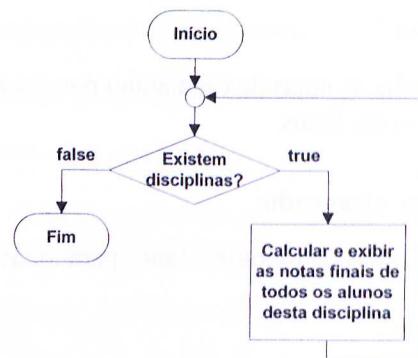


Figura 6.1 Ponto de partida na busca da solução do problema das notas.

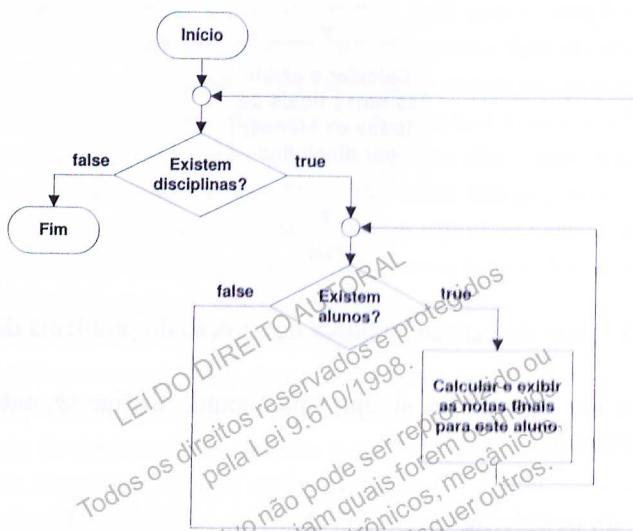
Percebe-se que são necessários alguns refinamentos. Dirige-se, então, para o particionamento do problema.

Particionamento do problema

Em uma primeira divisão do problema, descobre-se que é necessário o cálculo das notas dos alunos para cada disciplina, conforme a Figura 6.2.

**Figura 6.2** Primeira partição do problema.

Continuando a subdividir o problema, o próximo refinamento seria calcular e exibir as notas finais de cada aluno para cada disciplina, conforme ilustrado na Figura 6.3.

**Figura 6.3** Segunda partição do problema.

O próximo refinamento seria definir como calcular e exibir as notas finais para um único aluno, conforme indicado na Figura 6.4. Neste ponto, cabe refinar o processo de cálculo da nota final. Tome como exemplo a expressão apresentada a seguir (supondo que o aluno realize quatro provas no ano):

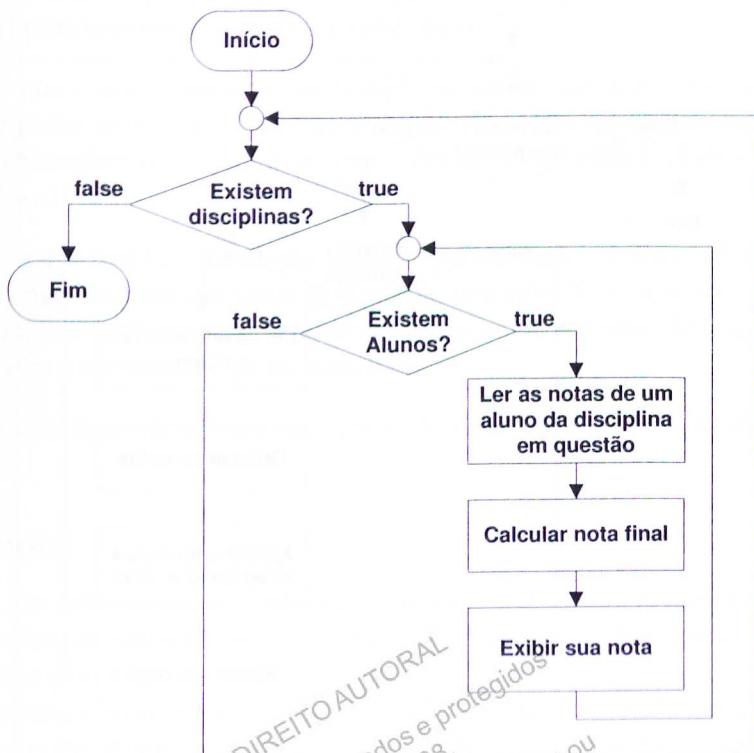


Figura 6.4 Terceira participação do problema.

Em que:

- P'' é a maior nota entre P_1, P_2 e P_3 ;
- P' é a segunda maior nota entre P_1, P_2 e P_3 .

Com o conhecimento dessa fórmula, basta agora refinrar o processo *Calcular Nota Final* a fim de que primeiro ordene as notas de modo decrescente (para descobrir a maior

e a segunda maior notas) e então aplicar a fórmula. O fluxograma final está apresentado na Figura 6.5.

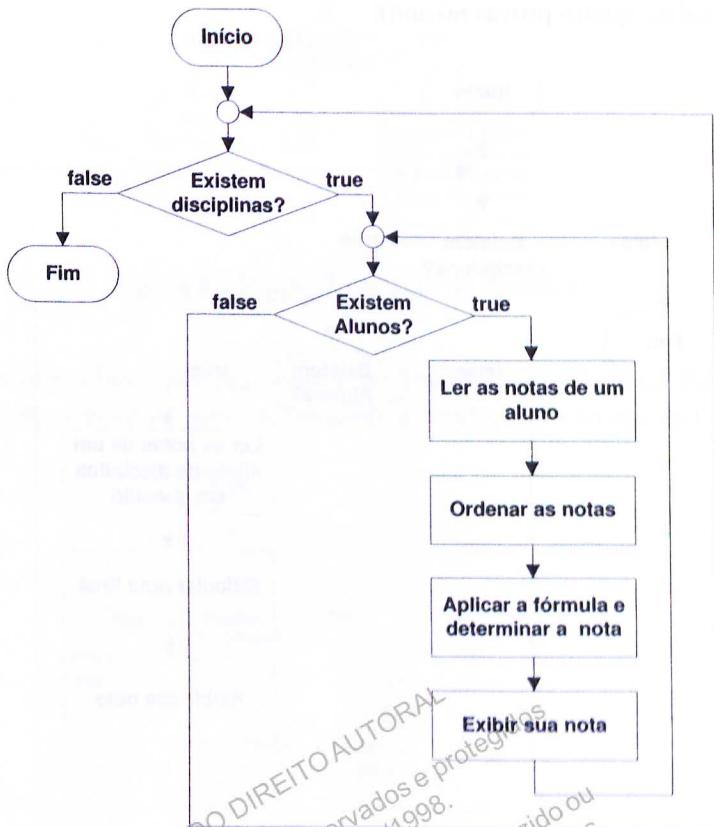


Figura 6.5 Partição final do problema

Solução do problema original

Percebe-se que as soluções encontradas já foram colocadas em seus lugares. Para uma versão final, ainda é necessário definir com exatidão os processos apresentados informalmente nesse fluxograma. Para se fazer isso, pode-se contar com *sub-rotinas*, assunto das próximas seções.

6.2 Sub-rotinas

Sub-rotina ou ainda subalgoritmo é o nome dado a um algoritmo que realiza uma tarefa específica, mas não representa um algoritmo (ou solução) completo. As sub-rotinas são utilizadas para a resolução de partes (integrantes e distintas) de algoritmos. Seu uso é aconselhável principalmente para os seguintes casos:

- Evitar que o algoritmo torne-se complexo: nesse caso a sub-rotina pode representar partes específicas da solução de um problema maior, tornando mais simples o entendimento da solução como um todo (é uma aliada poderosa da técnica *top-down*).
- Incentivar a reutilização de algoritmos: por exemplo, se fosse escrita uma sub-rotina para ordenar um vetor qualquer de tamanho N , esta poderia ser utilizada em qualquer problema que exigisse a ordenação de um vetor de tamanho predefinido, sem a necessidade de reescrevê-lo.

Existem dois tipos de sub-rotinas, que serão descritas a seguir: funções e procedimentos.

6.2.1 Funções

As funções são sub-rotinas que retornam um valor calculado. O conceito de função torna-se simples de entender se for lembrado o conceito de funções na Matemática, como, por exemplo, a função *seno*.

Na Matemática, ao se escrever $x = \text{sen}(0,77)$, está se aplicando a função *seno* sobre um argumento que, neste exemplo, é 0,77. Esse resultado é então atribuído à variável x . Não é necessário saber *a priori* como realmente funciona essa função, pois esses detalhes se encontram no algoritmo de cálculo da função seno de uma calculadora e em textos de cálculo numérico. Assim, em todas as situações em que se deseja o seno de um número, acaba-se por reutilizar essa função.

Em fluxogramas, as funções são representadas da seguinte forma (notar o símbolo de início para funções), conforme ilustrado na Figura 6.6:

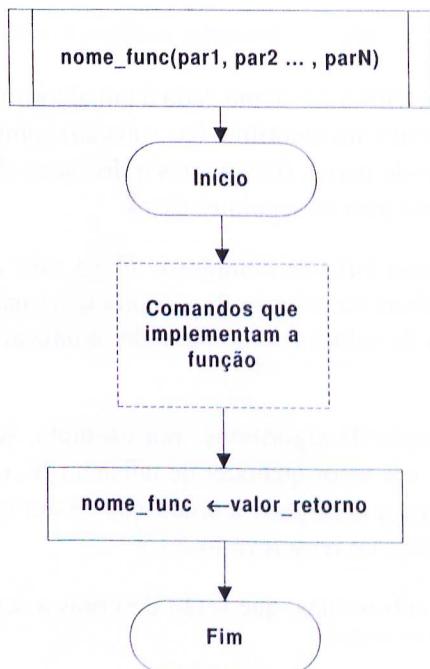


Figura 6.6 Representação de função em fluxograma.

- *nome_func*: representa o nome da função. Utiliza-se qualquer nome para uma função, desde que esteja de acordo com as regras para as variáveis descritas no Capítulo 3.
- *(par1, par2, ..., parN)*: representa uma lista de parâmetros sobre os quais a função vai operar. É com esses valores que ela vai calcular um resultado final a ser retornado. Pode-se escrever uma função sem parâmetro algum; nesse caso, não se utilizam parênteses. Cada parâmetro pode ser de qualquer tipo (inteiro, real, lógico, caractere, cadeia de caracteres, vetor ou matriz).
- *nome_func ← valor_retorno*: essa expressão é obrigatória para funções e indica que esse fluxograma representa uma função. Se a função se chamar *CalcAlgumaCoisa* e for retornar um valor final que está em uma variável *X* da função, essa expressão ficaria: *CalcAlgumaCoisa ← X*. O retorno pode ser de qualquer tipo (inteiro, real, lógico, caractere, cadeia de caracteres, vetor ou matriz).

6.2.2 Exemplos de funções

Deseja-se escrever uma função que calcule a contribuição para o INSS (a contribuição é calculada segundo a Tabela 6.1).

Tabela 6.1 Tabela de contribuições ao INSS.

TABELA VIGENTE	
Tabela de contribuição dos segurados empregado, empregado doméstico e trabalhador avulso, para pagamento de remuneração a partir de 16 de junho de 2010	
	Portaria nº 408, de 17 de agosto de 2010
Salário de contribuição (R\$)	Alíquota para fins de recolhimento ao INSS (%)
até R\$ 1.040,22	8,00 %
de R\$ 1.040,23 a R\$ 1.733,70	9,00 %
de R\$ 1.733,71 até R\$ 3.467,40	11,00 %
acima de R\$ 3.467,40	valor fixo de R\$ 381,41

A primeira decisão a ser tomada refere-se aos parâmetros da função. Para esse exemplo, nota-se que o cálculo da contribuição ao INSS é feito de acordo com o salário do contribuinte. Assim, uma função para se calcular a contribuição deve ter um parâmetro: salário (um número real). O retorno da função será um número real que representa o valor da contribuição. Dessa forma, o fluxograma que define essa função é exposto na Figura 6.7.

O identificador *S* é o parâmetro dessa função. Ao ser utilizada, deverá ser passado algum valor para *S* de modo que a função retorne um resultado. Observe que, internamente a essa função, utiliza-se a variável *C* para armazenar de forma temporária o resultado a ser retornado. Essa função poderia ser utilizada para o cálculo do desconto do IRRF (Imposto de Renda Retido na Fonte). Esse cálculo é feito sobre o salário líquido após a dedução da contribuição ao INSS, de acordo com a Tabela 6.2.

Tabela 6.2 Tabela de descontos para o IRRF.

Base de Cálculo em R\$	Aliquota %	Parcela a Deduzir do Imposto em R\$
Até 1.499,15	–	–
De 1.499,16 até 2.246,75	7,5	112,43
De 2.246,76 até 2.995,70	15	280,94
De 2.995,71 até 3.743,19	22,5	505,62
Acima de 3.743,19	27,5	692,78

Líquido = Bruto-INSS-IR

Líquido = Bruto-INSS-(base*alíquota - parcela)

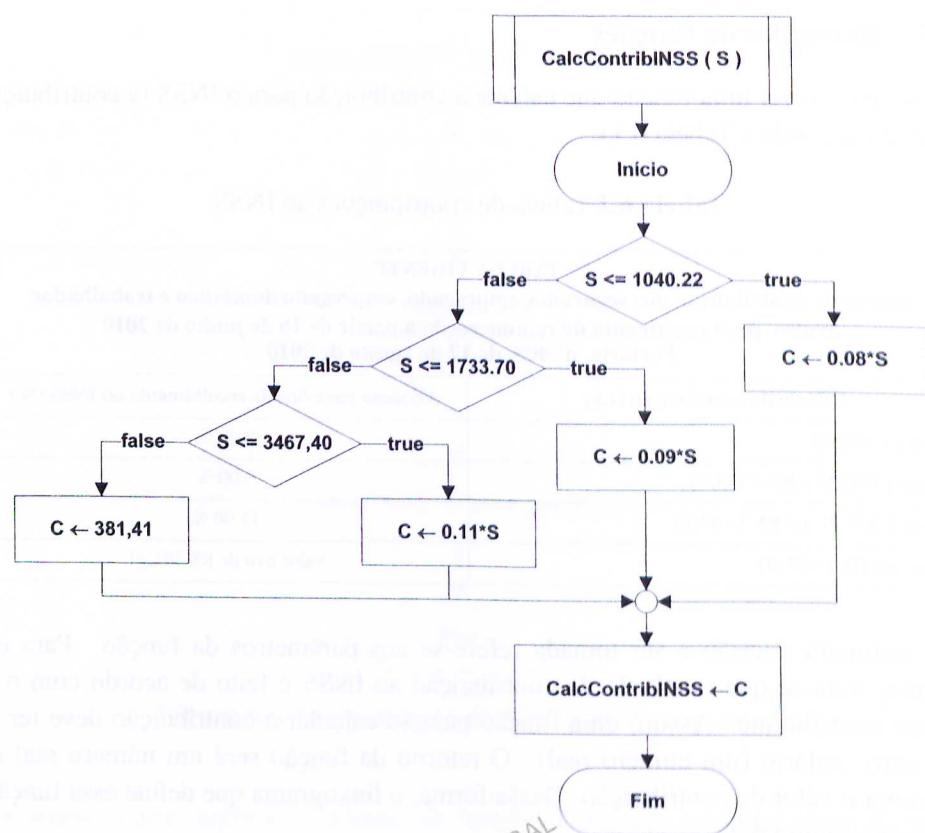
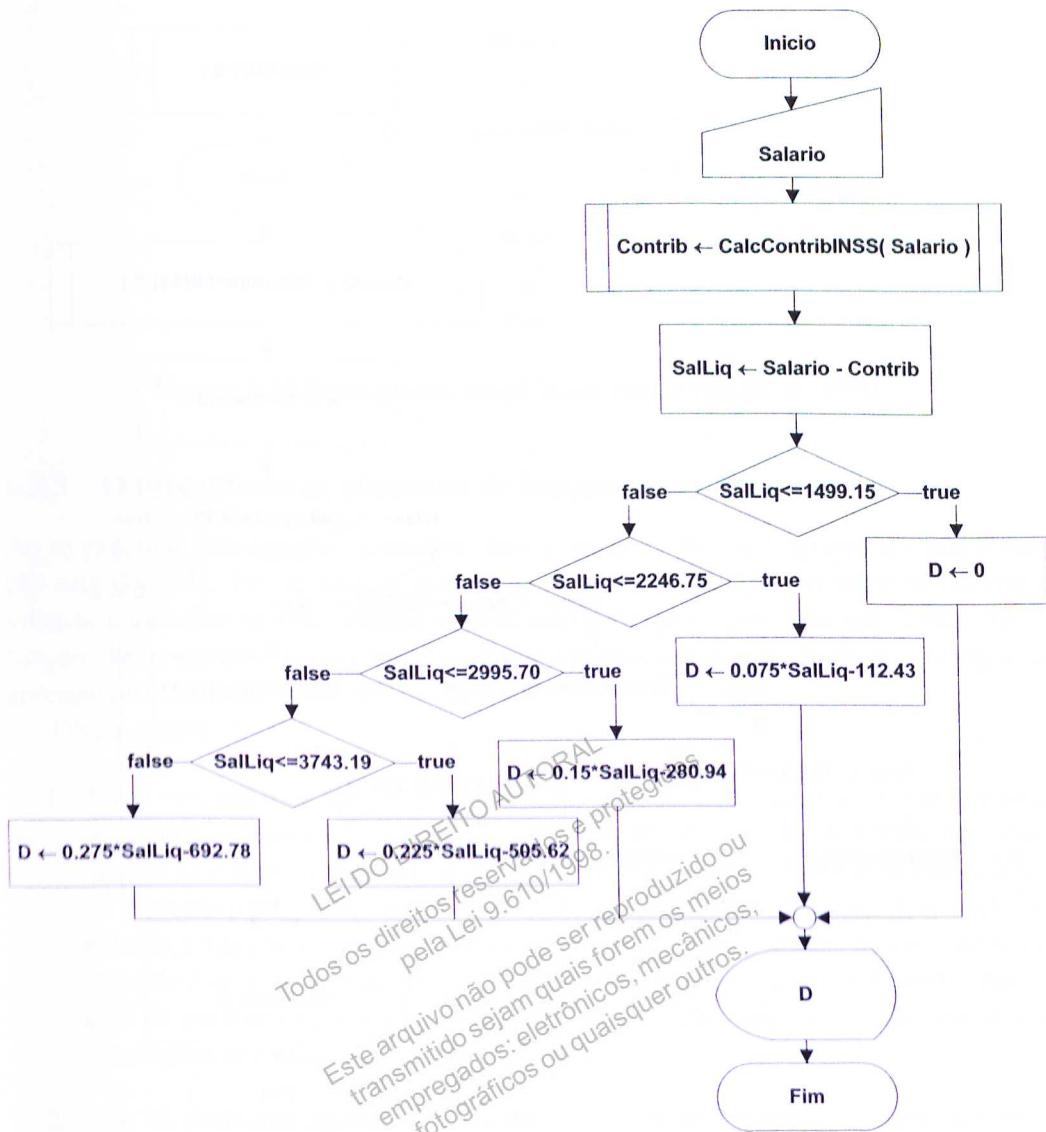


Figura 6.7 Função para calcular a contribuição do INSS.

Utilizando a função `CalcContribINSS` definida anteriormente, o fluxograma para esse cálculo ficaria conforme a Figura 6.8.

Como o cálculo do IRRF é muito executado em várias situações (folha de pagamento, por exemplo), cria-se também uma função que faça esse cálculo, segundo a Figura 6.9. Nota-se que as funções podem executar outras funções!

Finalmente, a solução para o cálculo de IRRF pode ser assim simplificada, conforme a Figura 6.10.

Figura 6.8 Uso da função *CalcContribINSS*.

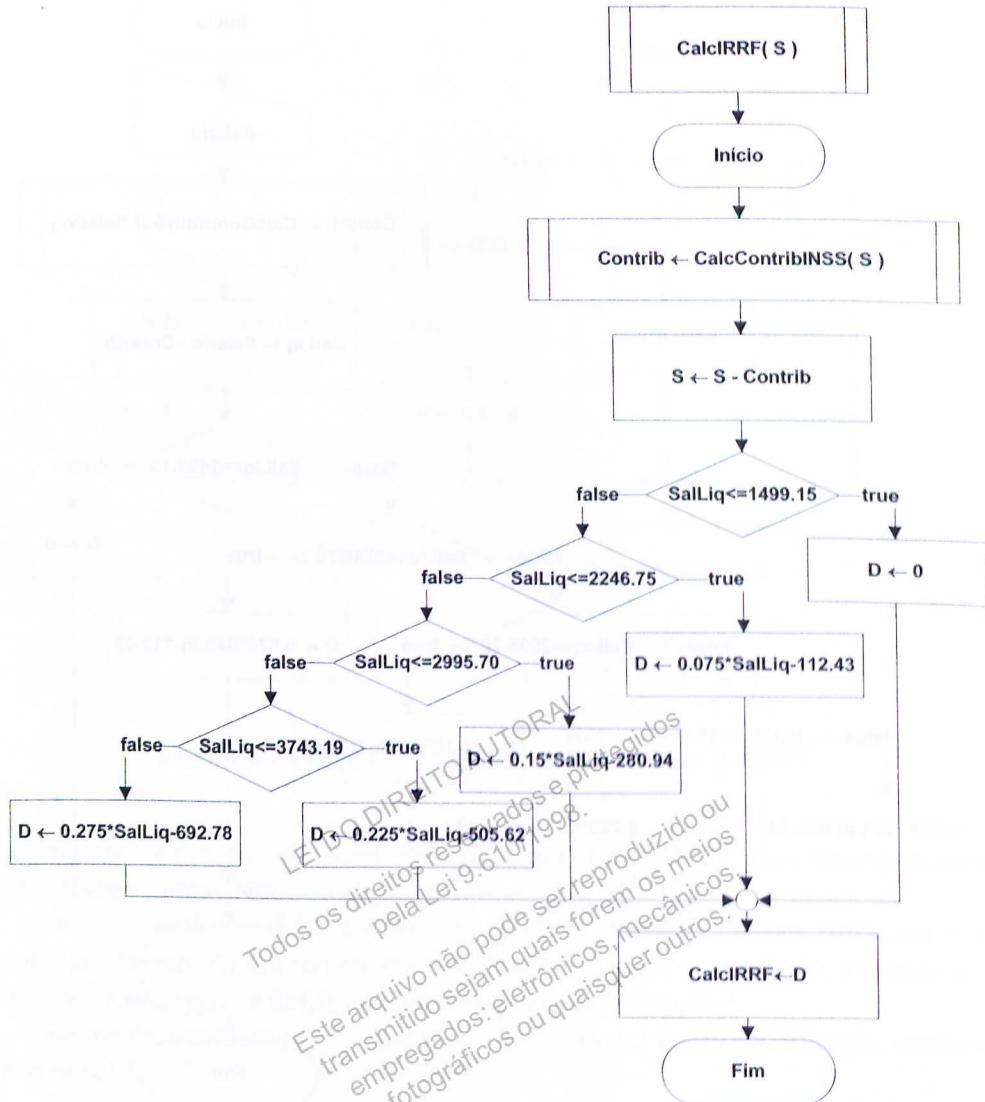


Figura 6.9 Função para calcular o desconto do IRRF.

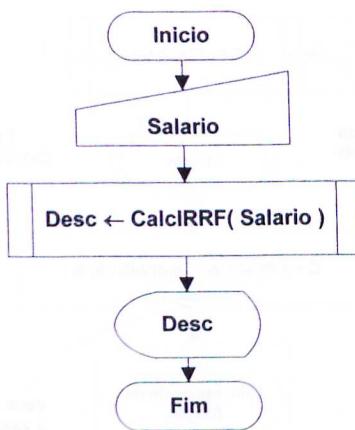


Figura 6.10 Fluxograma simplificado para o cálculo do IRRF.

6.2.3 O mecanismo de chamada de funções

Ao se executar uma função, ocorre um desvio no fluxo para o diagrama no qual a função está definida. Nesse desvio, executa-se a função, obtém-se um valor de retorno e volta-se para o fluxograma original no comando logo após o ponto em que se executou a função. Veja esse mecanismo, tomando como exemplo a execução do último fluxograma apresentado, conforme ilustrado na Figura 6.11.

Observações:

1. Ao se executar uma função, o seu parâmetro recebe implicitamente o valor de uma constante ou variável que se substitui em seu lugar. Quando se escreve uma chamada a uma função como `Desc ← CalcIRR(Salario)` entende-se que se o valor digitado para a variável `Salario` for 10, ocorrerá implicitamente `S ← Salario` e então a função vai executar os cálculos sobre seu parâmetro `S`, que contém o valor de `Salario`. A esse tipo de passagem de parâmetro, dá-se o nome de “passagem de parâmetro por valor” ou, simplesmente, “parâmetro por valor” ou, ainda, “passagem por valor”;
2. Não há problema algum em se reutilizar nomes de variáveis e parâmetros idênticos em funções diferentes. Reconheça que a função “protege” suas variáveis e parâmetros de outras de mesmo nome existentes em funções externas a ela;
3. Uma função retorna sempre um único valor.

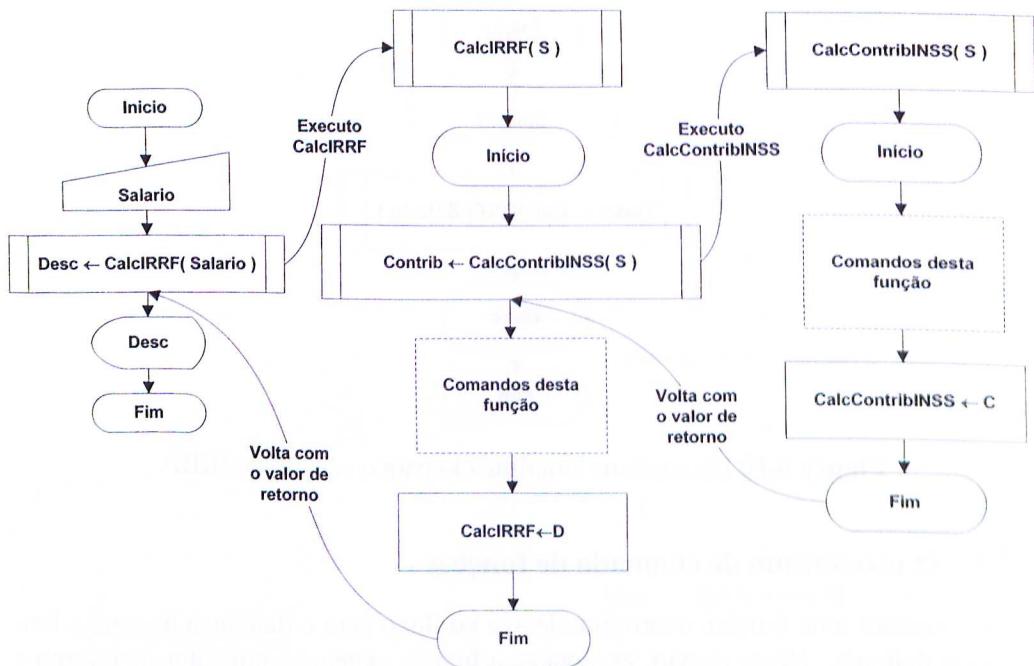


Figura 6.11 O mecanismo de chamada de uma função.

6.2.4 Procedimentos

Os procedimentos são sub-rotinas que, ao contrário de funções, não retornam um valor explicitamente. O uso de procedimentos é aconselhável quando se deseja realizar uma operação na qual uma função não se encaixa muito bem.

Um exemplo de aplicação de procedimento poderia ser no algoritmo que ordene os N valores de um vetor. Nesse caso, a tarefa se resume em ordenar um vetor existente, sem a necessidade de se retornar um valor.

Em fluxogramas, os procedimentos são representados de acordo com a Figura 6.12, sendo:

- *nome_proc*: representa o nome do procedimento. Utiliza-se qualquer nome para um procedimento desde que esteja de acordo com as regras para variáveis vistas no Capítulo 3.

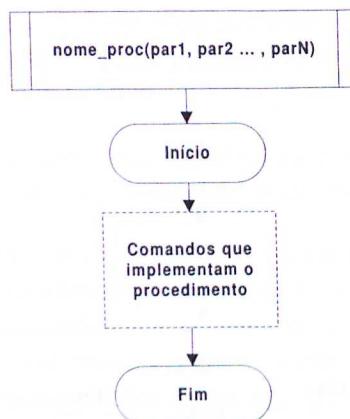


Figura 6.12 Representação de um procedimento.

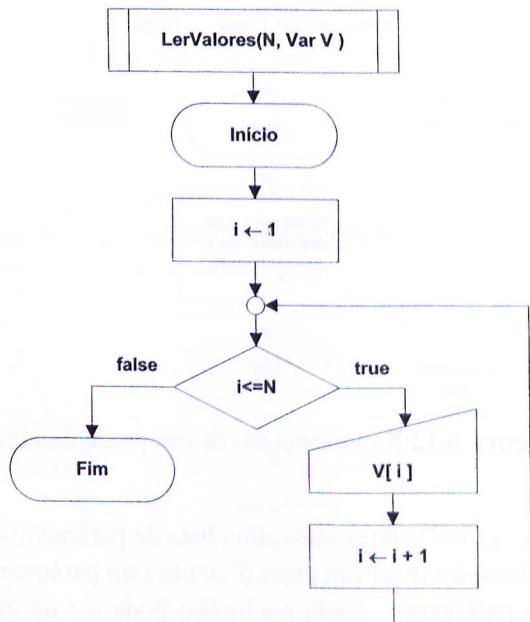
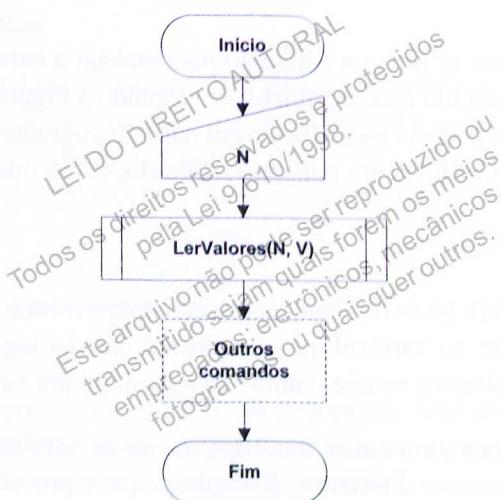
- $(par1, par2, \dots, parN)$: representa uma lista de parâmetros que o procedimento vai receber. Pode-se escrever um procedimento sem parâmetro algum; nesse caso, não se utilizam parênteses. Cada parâmetro pode ser de qualquer tipo (inteiro, real, lógico, caractere, cadeia de caracteres, vetor ou matriz).
- Como um procedimento não retorna os valores, não existe o bloco indicando esse retorno.

Como exemplo, tem-se um procedimento para realizar a leitura de N valores de um vetor qualquer, representado pelo parâmetro V , segundo a Figura 6.13.

Esse procedimento poderia ser utilizado em outro fluxograma que necessite ler os valores para um vetor específico com tamanho arbitrário, como ilustrado pela Figura 6.14.

Algumas observações:

1. Ao se executar um procedimento, o seu parâmetro recebe implicitamente o valor de uma constante ou variável que se substitui em seu lugar (como em funções). Esse tipo de parâmetro recebe o nome de parâmetro por valor (veja o item 6.2.3);
2. Não há problema algum em se reutilizar nomes de variáveis e parâmetros idênticos em procedimentos diferentes. Reconheça que o procedimento “protege” suas variáveis e parâmetros de outras de mesmo nome existentes em procedimentos externos a ele. Nesse tipo de passagem por valor, se o conteúdo do parâmetro for alterado por alguma expressão dentro da sub-rotina, ele não será enviado para

**Figura 6.13** Procedimento para realizar a leitura de um vetor.**Figura 6.14** Utilização de um procedimento.

quem o chamou. Assim, pode-se dizer que esse tipo de parâmetro tem a característica de ser um “parâmetro de entrada”, ou seja, apenas recebendo e nunca enviando o seu conteúdo;

3. Presume-se que os parâmetros tanto de funções quanto de procedimentos possam ser alterados internamente. Veja o fluxograma do exemplo anterior;
4. Na Figura 6.14, o parâmetro *V* sofre a passagem por referência. Nesse tipo de passagem é enviado ao parâmetro o endereço de memória onde a variável foi criada. Por meio dessa característica, se o valor do parâmetro for alterado, também sofrerá alteração a variável referenciada pelo parâmetro. Assim, pode-se dizer que esse parâmetro tem a característica de ser um “parâmetro de entrada e saída”, ou seja, recebe e envia o seu conteúdo;
5. Para diferenciar os dois tipos de passagem por parâmetro, adota-se nesse livro a palavra *VAR* no cabeçalho da sub-rotina antes do parâmetro que sofre a passagem por referência (veja a Figura 6.13);
6. Já a chamada da sub-rotina continua sendo feita da maneira usual, ou seja, sem nenhuma alteração (veja a Figura 6.14).

LEI DO DIREITO AUTORAL
Todos os direitos reservados e protegidos
pela Lei 9.610/1998.

Este arquivo não pode ser reproduzido ou
transmitido sejam quais forem os meios
empregados: eletrônicos, mecânicos,
fotográficos ou quaisquer outros.

6.3 Exercícios

6.1. ☀ Escreva o fluxograma para a função fatorial (utilize os conceitos de sub-rotinas apresentados). Lembrando:

$$n! = n \cdot (n - 1) \cdot (n - 2) \cdot (n - 3) \cdots \cdot 1$$

6.2. ☀ Escreva um fluxograma para calcular a soma dos N primeiros termos da série:

$$S = \frac{1!}{2!} + \frac{3!}{4!} + \frac{5!}{6!} + \dots$$

Em que N é lido pelo teclado. Utilize a função fatorial escrita anteriormente.

6.3. ☀ Deseja-se ler N pares de números e então calcular o produto dos maiores números em cada par. Como se procederia na solução desse problema, empregando a técnica *top-down*? Devem ser definidas as funções e os procedimentos necessários para a solução.

6.4. ☀ Utilizando a técnica *top-down*, como poderia ser implementada uma calculadora que realizasse as seguintes operações, guiadas por um conjunto de opções exibidas em uma tela, conforme a figura a seguir:

CALCULADORA	
0. SOMA 1. SUBTRAÇÃO 2. MULTIPLICAÇÃO 3. DIVISÃO 4. SENO 5. COSSENO 6. TANGENTE 7. MÓDULO 8. POTENCIAÇÃO 9. EXPONENCIAL 10. FATORIAL 11. LOGARITMO NEPERIANO 12. LOGARITMO NA BASE DEZ	
Entre com uma opção:	

Ao ser digitada a opção, deve-se realizar a operação que foi determinada, entretanto, devendo obedecer às condições apresentadas a seguir:

- a) Ao ser digitada a opção, deve-se certificar de que esta é válida, ou seja, pertence ao intervalo de 0 a 12.
- b) Para a soma, subtração, multiplicação e divisão, dois números serão digitados para a realização da operação. Deve-se verificar ainda que, na divisão, o denominador não poderá ser zero.
- c) No cálculo do seno, cosseno e tangente, o valor que será digitado estará em graus, no entanto, para a realização da operação, o ângulo deverá ser transformado para radianos. Deve-se certificar de que, no cálculo da tangente, o argumento da função não poderá ser múltiplo de 90° .
- d) No cálculo da potenciação, o programa deverá tratar o caso de se utilizar números negativos na base.
- e) Na opção fatorial, o cálculo só pode ser realizado para números inteiros, maiores ou iguais a zero.
- f) No cálculo do logaritmo neperiano e na base dez, deve-se verificar se o argumento é um valor maior que zero.
- g) Após o cálculo de cada função da calculadora, o programa emitirá uma mensagem, perguntando se o usuário deseja continuar com outro cálculo ou não. Em caso afirmativo, o menu principal será reapresentado e a nova opção escolhida; senão, o programa será encerrado.

6.5. ☀ Escreva uma função denominada *InverteCadeia*, que, dada uma cadeia de caracteres *S*, vai retornar o inverso dessa cadeia. Exemplo: se *S*='banana', então *InverteCadeia*(*S*)='ananab'. Dica: utilize as funções de manipulação de caracteres vistas no Capítulo 3.

6.6. ☀ Utilizando a função anterior, escreva um fluxograma que leia uma cadeia de caracteres e exiba uma mensagem, dizendo se ela é ou não um palíndromo. Lembrando: uma cadeia de caracteres é um palíndromo se possuir o mesmo significado se for lida da esquerda para a direita ou vice-versa.

6.7. ☀ Escreva um fluxograma para a função média aritmética de um vetor de tamanho *N*. A média aritmética é assim definida:

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

6.8. ☀ Escreva um fluxograma para a função desvio-padrão de um vetor de tamanho N . O desvio-padrão é, então, definido:

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

$$DM = \frac{\sum_{i=1}^n |x_i - \bar{x}|}{n}$$

6.9. ☀ As coordenadas de um vetor (da Geometria Analítica) no espaço R^3 podem ser representadas dentro de um vetor do tipo real de tamanho 3 assim: $V[1] = x$, $V[2] = y$, $V[3] = z$. Escreva fluxogramas que implementem as seguintes operações:

- procedimento para armazenar três valores reais (x, y, z) dentro de um vetor;
- procedimento que some dois vetores passados como parâmetros e escreva o resultado em um terceiro vetor;
- função que retorne o produto escalar entre dois vetores, passados como parâmetros;
- função que retorne o módulo de um vetor passado como parâmetro.

6.10. ☀ Escreva o procedimento *TROCA* que inverterá o valor de seus dois parâmetros x e y .

6.11. ☀ Com o procedimento *TROCA* anteriormente definido, escreva o fluxograma que vai representar o procedimento de ordenação crescente de um vetor de tamanho N pelo “Método das Trocas”, descrito no Algoritmo 6.4.

A seguir, verifique a correção do algoritmo, simulando o fluxograma para os seguintes vetores:

- vetor $A = 4$, com $N = 4$;
- vetor $B = 3, 7, -1$, com $N = 3$;
- vetor $C = 0, -1, 9, 3, -1$, com $N = 5$.

Algoritmo 6.1 Algoritmo de ordenação por trocas.**Início**

1. $i \leftarrow i$
2. **Enquanto** $i <= N - 1$ **Faça**
3. $j \leftarrow N$
4. **Enquanto** $j > i$ **Faça**
5. Se $VETOR[j - 1] > VETOR[j]$ **Então**
6. $TROCA(VETOR[j], VETOR[j - 1])$
7. **Fim Se**
8. $j \leftarrow j - 1$
9. **Fim Enquanto**
10. $i \leftarrow i + 1$
11. **Fim Enquanto**

Fim

6.12. ☀ Construa um fluxograma que leia dois números inteiros (A e B), calculando as seguintes expressões: $X = A!$, $Y = 2A!$ e $Z = (2A)!$. O cálculo do fatorial deve estar escrito em uma sub-rotina.

6.13. ☀ Seja a função abaixo:

$$F(X) = X^3 + 3X^2$$

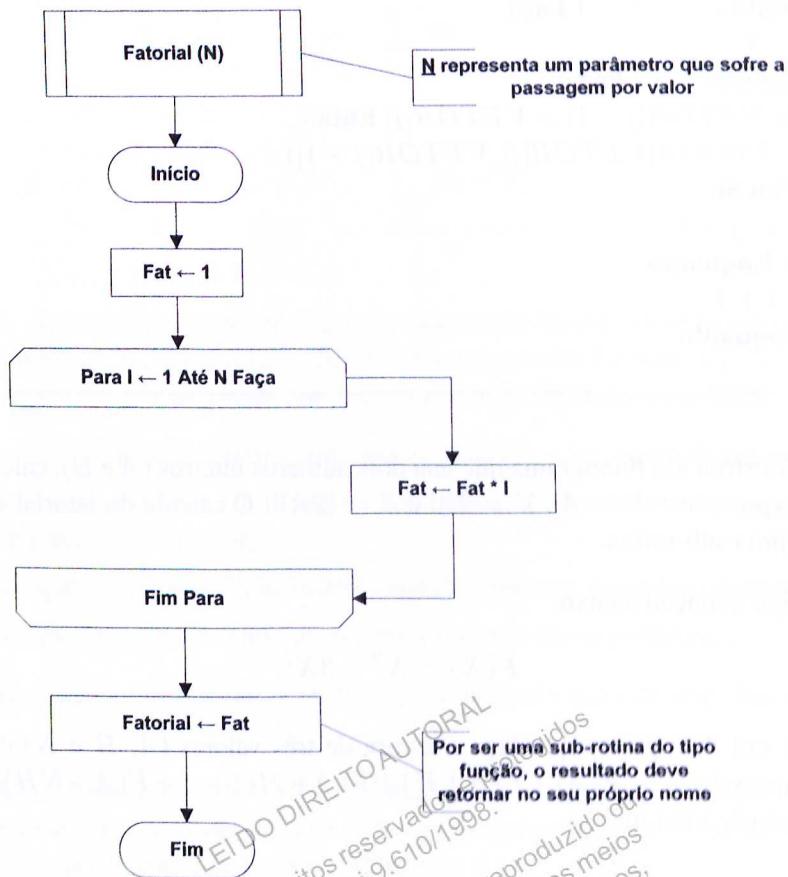
Elabore um fluxograma que faça a leitura de três valores (A , H e N) e a seguir calcule a expressão $P = F(A) + F(A+H) + F(A+2H) + \dots + F(A+NH)$, definida pela aplicação de uma função.

LEI DO DIREITO AUTORAL
Todos os direitos reservados. Fábio Góes
pela Lei 9.610/1998.

Este arquivo não pode ser reproduzido ou transmitido sejam quais forem os meios empregados: eletrônicos, mecânicos, fotográficos ou quaisquer outros.

6.4 Exercícios resolvidos

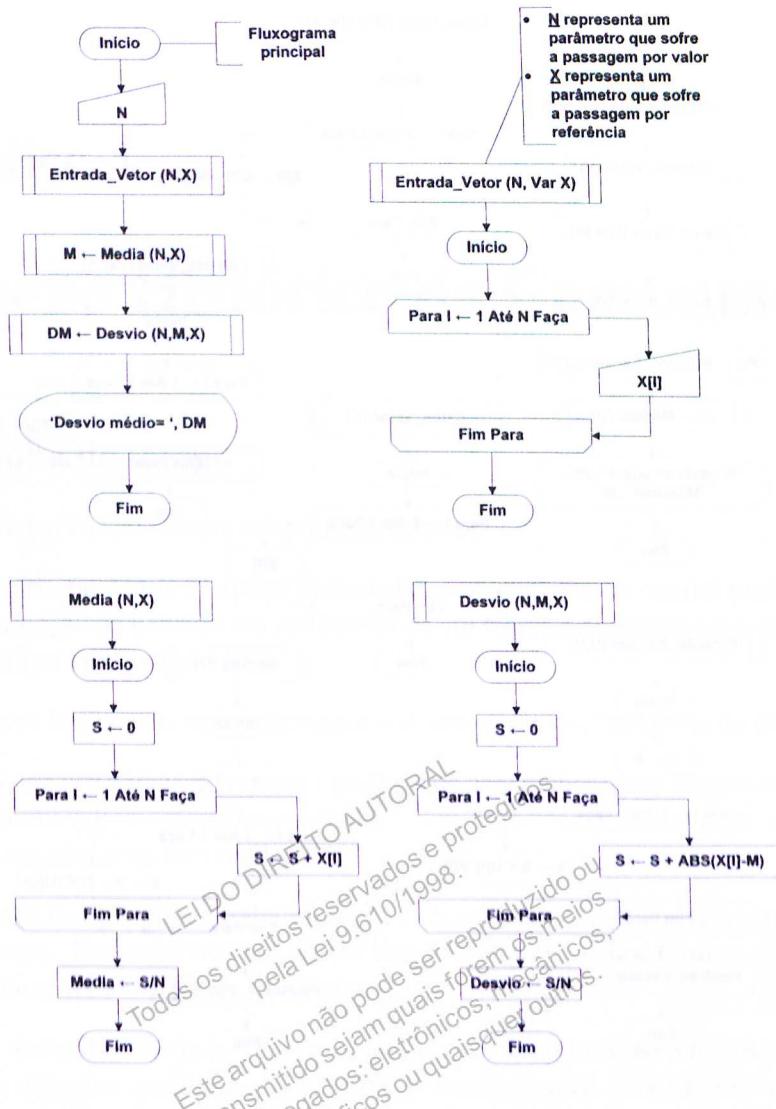
6.1. ☀



Todos os direitos reservados
pela Lei 9.610/1998.

Este arquivo não pode ser reproduzido ou
transmitido sejam quais forem os meios
empregados: eletrônicos, mecânicos,
fotográficos ou quaisquer outros.

6.8. ☀



6.9.

