

Capítulo 3

Algoritmos e Fluxogramas

No Capítulo 1 foi apresentado o conceito de algoritmo, suas características tais como a sua formalização via sintaxe e semântica adequadas e como o desenvolvimento de um algoritmo representa o desenvolvimento da solução de um problema. Neste capítulo será revisado o conceito de algoritmo para depois ser introduzida uma representação gráfica amplamente conhecida de algoritmos: fluxogramas.

3.1 Revisão do conceito de algoritmo

No Capítulo 1 houve um primeiro contato com a palavra algoritmo. Lá fazia-se a descrição de seu significado, de características desejáveis que todo algoritmo deve possuir, como sintaxe e semântica bem-definidas, e relacionava-se sua utilização com a solução de problemas. Para fixar o conceito de algoritmo, é fornecida a sua definição segundo o dicionário Aurélio:

Algoritmo: 1. *Mat.* Processo de cálculo ou de resolução de um grupo de problemas semelhantes, em que se estipulam, com generalidade e sem restrições, regras formais para a obtenção do resultado ou da solução do problema. 2. *Inform.* Conjunto de regras e operações bem-definidas e ordenadas, destinadas à solução de um problema ou de uma classe de problemas, em um número finito de etapas.

Em outras palavras, o algoritmo representa o **caminho de solução para um problema**. A elaboração do algoritmo é de importância crucial para a criação de um programa de computador e nas soluções de qualquer tipo de problema. Da definição exposta anteriormente, pode-se extrair as seguintes características evidentes:

- Um algoritmo representa uma sequência de regras.
- Essas regras devem ser executadas em uma ordem preestabelecida.
- Cada algoritmo possui um conjunto finito de regras.
- Essas regras devem possuir um significado e ser formalizadas segundo alguma convenção.

3.2 Aplicabilidade dos algoritmos

Existe um algoritmo embutido em toda tarefa, independentemente de ela ser relacionada a um programa de computador. Em nosso cotidiano, executamos toda e qualquer tarefa utilizando algoritmos, mesmo não percebendo isso. Atos como comer, respirar, ir para a escola, dirigir um automóvel, resolver uma prova, estudar, cozinhar, fazer uma refeição, consertar o motor de um automóvel etc. são tarefas que podem ser descritas por meio de algoritmos.

Por outro lado, existem algoritmos que precisamos aprender para poder realizar certas tarefas específicas, como, por exemplo, aquelas ligadas à Engenharia e à Computação. Assim, para especificar um processo de montagem de um circuito eletrônico, um processo químico industrial e um programa eficiente de pesquisa de informações em um banco de dados, entre tantos, são necessários informações e conhecimentos adicionais aos que já possuímos. Desse modo, conclui-se que:

*Algoritmos não servem apenas para programar computadores!
São de uso geral!*

3.2.1 Exemplo não computacional de um algoritmo

Um exemplo concreto de um algoritmo que está fora do ambiente computacional é a receita para se preparar um sorvete de chocolate. Assim, o problema a ser resolvido é a definição dos passos necessários para se obter um sorvete de chocolate.

Aqui precisa-se saber inicialmente quais são os ingredientes essenciais para se fazer o sorvete. Uma sugestão seria utilizar os seguintes ingredientes:

- 1 tablete de chocolate meio amargo;
- 1 lata de leite condensado;

- a mesma medida da lata com leite;
- raspas de chocolate ou chocolate granulado.

Com esses ingredientes, especificam-se os passos da receita que resolve o problema da preparação do sorvete, representado pelo Algoritmo 3.1:

Algoritmo 3.1 Algoritmo para fazer um sorvete de chocolate.

Início

1. Ponha o chocolate em uma tigela refratária.
2. Deixe a tigela no micro-ondas durante um minuto em potência média.
3. Tire o chocolate do forno com cuidado e mexa-o até esfriar.
4. Bata-o no liquidificador com o leite condensado e o leite.
5. Despeje tudo em uma forma de gelo e espere congelar por três horas.
6. Distribua o sorvete em taças.
7. Decore com as raspas ou com o chocolate granulado.
8. Sirva.

Fim

3.2.2 Exemplo computacional de um algoritmo

Outro exemplo concreto, agora no domínio da Matemática, é o algoritmo de Euclides (definido entre 400-300 a.C.) para a determinação do *máximo divisor comum* entre dois números inteiros x e y . Os valores das variáveis x e y representam os valores de entrada do problema ou – fazendo uma comparação com o exemplo anterior – os “ingredientes”. O algoritmo que resolve esse problema pode ser descrito conforme o Algoritmo 3.2.

Algoritmo 3.2 Algoritmo para calcular o máximo divisor comum entre dois números.

Início

1. Pedir para o usuário fornecer valores inteiros para x e y .
2. **Enquanto** $y \neq 0$ **Faça**
3. $r \leftarrow$ o resto da divisão entre x e y
4. $x \leftarrow y$
5. $y \leftarrow r$
6. **Fim Enquanto**
7. Exibir para o usuário o MDC procurado e que está em x .

Fim

Para verificar se esse algoritmo está correto, é necessário **simulá-lo** de acordo com suas regras. A linha 1 apresenta um comando que pede o fornecimento de dois valores inteiros, um para x e outro para y . Podem ser quaisquer valores, por exemplo, $x = 18$ e $y = 15$.

A linha 2 representa um comando que indica uma repetição, isto é, o que existe entre as linhas 2 e 6 deve ser repetido, enquanto a condição expressa na linha 2 ($y \neq 0$) for verdadeira. Dessa maneira, com os valores propostos, tem-se a geração de valores segundo a Tabela 3.1.

Tabela 3.1 Simulação do algoritmo de Euclides.

Linha	Comando	Valor das variáveis		
		<i>x</i>	<i>y</i>	<i>r</i>
1	Pedir para o usuário fornecer valores inteiros para x e y .	18	15	?
2	Enquanto $y \neq 0$ Faça (verdadeiro: $y = 15$)	18	15	?
3	$r \leftarrow$ o resto da divisão entre x e y	18	15	3
4	$x \leftarrow y$	15	15	3
5	$y \leftarrow r$	15	3	3
6	Fim Enquanto	15	3	3
2	Enquanto $y \neq 0$ Faça (verdadeiro: $y = 3$)	15	3	3
3	$r \leftarrow$ o resto da divisão entre x e y	15	3	0
4	$x \leftarrow y$	3	3	0
5	$y \leftarrow r$	3	0	0
6	Fim Enquanto	3	0	0
2	Enquanto $y \neq 0$ Faça (falso: $y = 0$)	3	0	0
7	Exibir para o usuário o MDC procurado e que está em x .	3	0	0

A Tabela 3.1 indica a execução dos comandos do algoritmo de Euclides, supondo valores iniciais como $x = 18$ e $y = 15$. Observe que a variável r nas duas primeiras linhas da tabela ainda não havia sido considerada pelo algoritmo, daí seu valor ser indeterminado. É importante notar que alguns comandos alteram o valor das variáveis e então novos valores passam a valer. Dessa forma, foi obtido como MDC entre x e y propostos o último valor que foi armazenado em x ($x = 3$).

Embora esses dois exemplos sejam algoritmos, existem ainda algumas deficiências em suas descrições. Entre elas:

- no primeiro exemplo consegue-se criar uma quantidade fixa de sorvete (essa receita rende seis taças). E se desejar obter somente uma taça? Não seria interessante conseguir uma solução mais geral, em que se possa variar as quantidades de in-

gredientes para obter a quantidade que quiser de sorvete? Lembre-se das soluções literais discutidas no Capítulo 1;

- no segundo caso, apesar de ser um algoritmo que determina o máximo divisor comum entre quaisquer valores inteiros de x e y , o algoritmo precisa ser formalizado. Nesse caso, é necessário definir uma sintaxe e semântica para representar esse algoritmo de uma maneira livre de interpretações ambíguas (a interpretação, por enquanto, está na tabela de simulação).

De qualquer modo, ambos os algoritmos apresentados possuem algumas propriedades em comum, que serão mais bem detalhadas na próxima seção.

3.3 Propriedades de um algoritmo

Todo algoritmo possui uma série de propriedades que serão descritas a seguir:

- **Valores de entrada.** Todo algoritmo deve possuir zero, uma ou mais entradas de dados. O exemplo do sorvete visto anteriormente na Seção 3.2.1 representa um algoritmo que possui zero entradas, pois seu algoritmo opera com quantidades fixas de ingredientes. Já o exemplo da Seção 3.2.2 representa um algoritmo com duas entradas de dados, atribuídas às variáveis x e y .
- **Valores de saída.** Todo algoritmo possui uma ou mais saídas, que simboliza(m) seu(s) resultado(s). Assim, tanto o algoritmo do sorvete quanto o de Euclides possuem uma única saída. No primeiro, a saída é o próprio sorvete; no segundo, o valor do máximo divisor comum entre x e y .
- **Finitude.** Costuma-se dizer que toda tarefa a ser realizada possui um início, meio e fim. Como os algoritmos representam os passos de solução de um problema – executando assim uma tarefa –, também possuem um início, meio e fim. Portanto, uma primeira propriedade do algoritmo é a finitude. Todo algoritmo deve ser finito, isto é, deve possuir um início e um conjunto de passos que, ao serem executados, levarão sempre ao seu término ou fim, executando a tarefa a que se propõe. Ambos os exemplos vistos nas Seções 3.2.1 e 3.2.2 são algoritmos finitos, pois chegam a um resultado em um número finito de passos.

Deve-se uma atenção especial a essa propriedade. Muitas vezes, por desatenção, pode-se criar um algoritmo que nunca chegará a um resultado, tornando-se infinito. Por exemplo, altere a condição da linha 2 do Algoritmo 3.2 para $y \geq 0$.

Simule-o, então, para os valores $x = 5$ e $y = 2$ e responda: você consegue chegar à linha 7?

- **Passos elementares.** Um algoritmo computacional deve ser explicitado por meio de operações elementares, sem que possam haver diferenças de interpretação, de forma tal que possa ser executado até por *máquinas bastante limitadas*, como o computador.

Dos exemplos vistos, o algoritmo de Euclides possui essa propriedade, pois utiliza somente operações matemáticas e comparações, operações que qualquer computador realiza por natureza. Já o algoritmo do sorvete deve ainda ser bem-refinado, para que suas operações possam ser representadas, de alguma maneira, em passos elementares.

- **Correção.** Um algoritmo deve ser correto, isto é, deve permitir que, com sua execução, se chegue à(s) saída(s) com resultados coerentes com a(s) entrada(s). Para saber se um algoritmo está correto ou não, deve-se realizar testes com diversos valores de entrada (simulação), cujos valores a serem produzidos já se conhece *a priori* e, então, comparar esses resultados com os valores produzidos pelo algoritmo em questão.

Por exemplo: o máximo divisor entre os números 12 e 9 é 3. Faça $x = 12$ e $y = 9$ e então execute o algoritmo de Euclides mostrado na Seção 3.2.2. Você chegou ao mesmo resultado? Para outros pares de valores x e y , o algoritmo está correto?

3.4 Fluxogramas

Para um algoritmo ser útil, deve ser entendido da mesma forma por todas as pessoas que o utilizarem. Até o presente momento, as descrições de algoritmos que foram apresentadas usaram uma linguagem informal para representar os passos a serem executados. Apesar de cômodo, o uso de linguagens informais para a descrição de algoritmos pode levar ao surgimento de ambiguidades por diferentes pessoas.

Existem diversas maneiras de formalizar a representação de um algoritmo – neste livro utiliza-se uma forma de representação gráfica denominada fluxograma. A definição da palavra fluxograma, pelo dicionário Michaelis, é:

Fluxograma: *Inform.* (fluxo+grama). 1. Diagrama para representação de um algoritmo. 2. Representação gráfica, por símbolos especiais, da definição, análise ou método de solução de um problema.

O uso de fluxogramas neste livro deve-se primeiramente ao fato de que o engenheiro deva possuir grande familiaridade com diagramas esquemáticos com linguagem matemática e expressão gráfica. Os fluxogramas possuem um grande apelo visual e aplicação no entendimento de processos industriais, os quais são muito importantes na formação e na vida prática do engenheiro. Além disso, a utilização de fluxogramas como elemento de representação na solução de problemas computacionais é ainda muito grande na ciência da computação.

Todo fluxograma deve possuir uma *sintaxe* e uma *semântica* bem-definidas. A sintaxe de um fluxograma é definida pela forma correta de empregar seus elementos, os quais são:

- símbolos gráficos específicos;
- expressões admissíveis a serem escritas no interior dos símbolos;
- sub-rotinas predefinidas que podem ser utilizadas nas expressões.

A semântica de um fluxograma indica como interpretá-lo. São regras de como entender e simular a solução que ele propõe. Tanto a sintaxe quanto a semântica de fluxogramas serão tratadas nas Seções 3.5, 3.6, 3.7, 3.8 e 3.9.

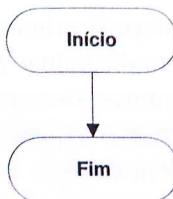
3.5 Construindo fluxogramas

Os símbolos de fluxograma a serem adotados neste livro seguirão a norma ISO 5807/1985. Uma descrição completa desses símbolos e de suas aplicações encontra-se no Apêndice B. Para se criar um fluxograma que represente um algoritmo, deve-se construir cada um de seus passos de acordo com um símbolo apropriado da norma. Serão apresentadas as regras básicas para a construção de fluxogramas nas seções a seguir.

3.5.1 Fluxograma mínimo

O menor fluxograma que se pode escrever é aquele que não executa absolutamente nada. De qualquer forma, todo fluxograma deve possuir um **íncio** e um **fim**. Os símbolos que denotam o íncio e fim de um fluxograma são representados por “retângulos arredondados”, conhecidos por **terminadores**, contendo, respectivamente, os textos **Íncio** e **Fim**, conforme ilustrado na Figura 3.1.

Esses símbolos não representam nenhum tipo de operação, mas são essenciais para a determinação do íncio e fim do fluxograma. Os elementos de um fluxograma são conectados por **setas** que indicam o caminho a ser seguido a partir de um símbolo. A regra básica para a interpretação de um fluxograma pode ser sintetizada pelo Algoritmo 3.3.

**Figura 3.1** Fluxograma mínimo.**Algoritmo 3.3** Algoritmo para interpretar um fluxograma.**Início**

1. Vá para o símbolo **Início**.
2. **Repita**
3. Seguindo a direção indicada pela seta, vá para o próximo símbolo.
4. Interprete esse símbolo.
5. **Até chegar no símbolo Fim.**

Fim

Assim, a interpretação do fluxograma da Figura 3.1 não leva à execução de nenhum comando, pois, após o símbolo **Início**, chega-se ao símbolo **Fim** sem passar por qualquer outro.

3.5.2 Fluxograma com comandos sequenciais

Um fluxograma contendo apenas comandos sequenciais é aquele que, a partir do símbolo **Início**, permite a execução das instruções contidas nos símbolos subsequentes sem desvio algum na direção até se alcançar o símbolo **Fim**.

Por exemplo, deseja-se construir um fluxograma que represente o algoritmo para calcular a força exercida pela coluna de um líquido sobre a área da válvula de um reservatório, conforme a Figura 3.2.

Nesse problema, conhece-se a altura h (m) do reservatório, o diâmetro d (m) da válvula e o peso específico γ do líquido (N/m^3). A força F , em Newtons, calculada é o peso da coluna do líquido. Como já temos o peso específico da substância, o cálculo da força é dado por:

$$F = \gamma \times VolumeColuna = \gamma \times AreaTampa \times h = \frac{\pi \times \gamma \times d^2 \times h}{4}$$

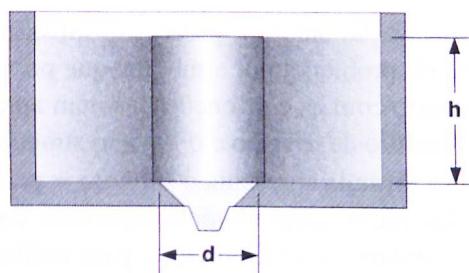


Figura 3.2 Problema da força exercida pela coluna de um líquido.

As variáveis existentes na fórmula anterior deverão também estar presentes na solução apresentada pelo fluxograma. No entanto, as **variáveis de um fluxograma** possuem um significado adicional daquele encontrado nas variáveis da Matemática: as variáveis em um fluxograma representam simbolicamente espaços da memória nos quais serão armazenados seus valores.

Deve-se ter em mente que uma variável, d , por exemplo, ao aparecer em um fluxograma, representa algum espaço da memória onde seu valor será armazenado, conforme indicado pela Figura 3.3. Não é necessário conhecer qual é a posição desse espaço na memória: sabendo-se apenas o nome da variável, pode-se **ler** ou **alterar** seu conteúdo.

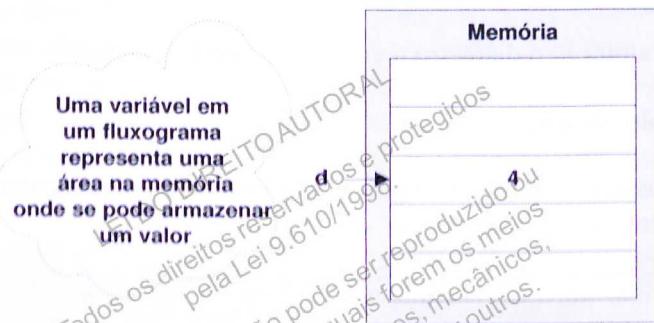


Figura 3.3 Significado de variável em fluxogramas.

Além disso, é necessário renomear as variáveis do problema original em alguns casos. Por exemplo, as variáveis π e γ desse problema serão escritas futuramente no texto de uma linguagem de programação, que contém apenas caracteres ASCII. Assim, seguindo as convenções que serão apresentadas na Seção 3.7, renomeiam-se as variáveis π e γ do problema, respectivamente, para pi e $gama$. Deve-se considerar, inclusive, que o símbolo pi é predefinido (veja a Seção 3.9) com o valor 3,14159.

Utilizando a mesma observação anterior, precisa-se substituir o símbolo de divisão e multiplicação da fórmula do problema por símbolos que possam ser escritos de uma forma mais simples. De acordo com as convenções a serem apresentadas na Seção 3.8, serão utilizados a / como símbolo de **divisão** e o * como símbolo de **multiplicação**.

Observando, ainda, que a fórmula apresenta o símbolo = para denotar que a variável F vai possuir o mesmo valor que o calculado pela expressão à sua direita, é necessário fazer outra observação. O símbolo = será utilizado para realizar comparações, isto é, verificar igualdades. No caso desse problema, a variável F armazenará um valor calculado pela expressão. Dessa forma, para indicar essa operação, denominada **atribuição**, se utilizará o símbolo ← para indicar que F armazenará o valor calculado.

Por fim, deve-se notar que na fórmula aparece o termo d^2 . Para simplificar, será utilizada a sub-rotina predefinida *sqr*, que eleva ao quadrado o valor indicado em seu argumento (veja a Seção 3.9).

Observa-se que, para resolver esse problema de forma geral, devem-se obter, via algum **dispositivo de entrada** – por exemplo, um **teclado** –, os **valores de entrada** necessários para solucionar o problema. Essa operação é comumente chamada de **leitura dos dados**. De forma análoga, após a obtenção do resultado pela aplicação da fórmula, é preciso **exibir** o valor para o usuário, via algum **dispositivo de saída** – por exemplo, um **monitor de vídeo** –, o valor produzido, nomeado **valor de saída**.

Nesse problema, os valores de entrada e de saída são os seguintes:

- Entrada: a altura h , o diâmetro d e o peso específico $gama$.
- Saída: o valor da força F .

Agora já é possível esboçar um algoritmo informal para representar a solução desse problema, descrito pelo Algoritmo 3.4.

Algoritmo 3.4 Algoritmo para calcular a força exercida pela coluna de um líquido.

Início

1. Ler os valores h , d e $gama$.
2. Calcular $F \leftarrow pi * gama * sqr(d) * h / 4$.
3. Exibir o valor de F .

Fim

Nesse instante se inicia a construção do fluxograma. Da forma que foi apresentado na Seção 3.5.1, a construção desse fluxograma começa com o desenho do símbolo de **Início**, conforme ilustrado na Figura 3.4.

```

    graph TD
        Inicio([Início]) --> Entrada[/h, d, gama/]
    
```

Início

Figura 3.4 Passo 1 na construção do fluxograma para o problema da força.

Na sequência, conecta-se esse símbolo ao primeiro passo a ser executado, usando-se uma seta. O primeiro passo, de acordo com o Algoritmo 3.4, é a leitura dos valores das variáveis h , d e $gama$. O símbolo do fluxograma para se ler os valores externos a serem atribuídos a variáveis do algoritmo tem a forma de um trapézio. Indica-se no seu interior os nomes das variáveis que receberão os valores a serem digitados, separados por vírgula, de acordo com a Figura 3.5.

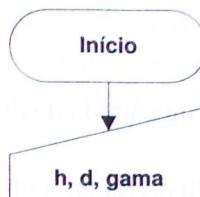


Figura 3.5 Passo 2 na construção do fluxograma para o problema da força.

Deve-se fazer duas observações sobre o significado desse símbolo. Em primeiro lugar, esse símbolo não indica qual tipo de tela os dados serão digitados. Isso reflete a característica de independência dos fluxogramas. Portanto, é de “imaginação livre” como esse comando funcionaria na prática. Por exemplo, ele poderia ser futuramente implementado em Pascal ou em Delphi e ser apresentado em telas como mostra a Figura 3.6.

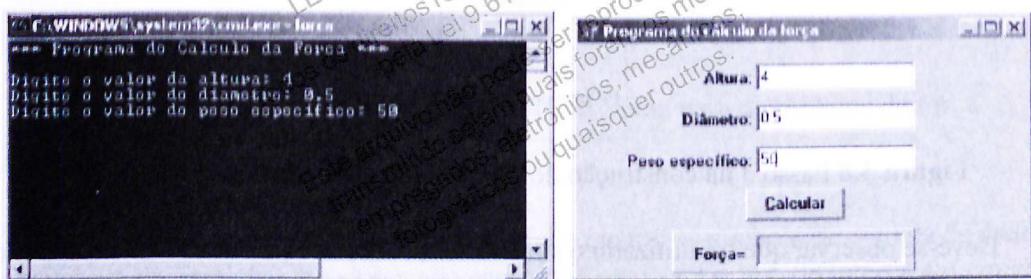


Figura 3.6 Duas interpretações concretas do símbolo de entrada.

Em segundo lugar, a execução do símbolo de entrada realiza uma **atribuição im-**

plícita dos valores digitados às variáveis que estão indicadas no seu interior, na ordem que foram escritas. Dessa forma, a execução do símbolo de entrada apresentado na Figura 3.5 com os valores 4, 0,5 e 50 vai atribuir esses valores, respectivamente, às variáveis h , d e $gama$, isto é, serão armazenados nas posições de memória reservadas para essas variáveis, conforme ilustrado pela Figura 3.7.

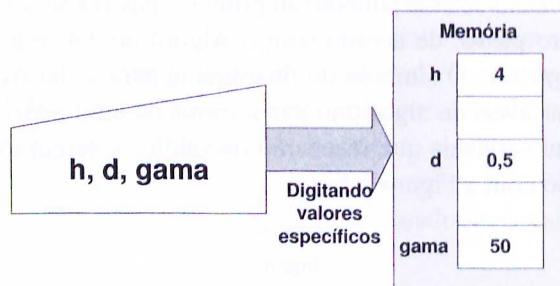


Figura 3.7 O efeito da entrada de dados nas variáveis.

Seguindo com a elaboração do fluxograma, o próximo passo representa um cálculo direto, portanto, **um processo**. O símbolo de fluxograma para representar cálculos ou processos a serem **executados sem questionamento** tem a forma de um retângulo. Em seu interior, escrevem-se as expressões ou o nome do processo que será executado. Nesse exemplo, calcula-se a força F , de acordo com a Figura 3.8.



Figura 3.8 Passo 3 na construção do fluxograma para o problema da força.

Deve-se observar que foi utilizado o símbolo \leftarrow , conforme justificado anteriormente para **atribuir** o valor do cálculo realizado pela expressão à variável F . Assim, com os valores $h = 4$, $d = 0,5$ e $gama = 50$, será armazenado o valor 32,2699 na posição de memória que a variável F representa, como mostra a Figura 3.9.

No próximo passo, deve-se exibir o valor calculado da variável F . O símbolo do

Memória	
h	4
d	0,5
F	32,2699
gama	50

Figura 3.9 O efeito do comando de atribuição em uma variável.

fluxograma para **exibir** os valores está representado a seguir e, em seu interior, escrevem-se os nomes das variáveis a serem exibidas separados por vírgula. Nesse caso, deve-se exibir apenas o valor de F , conforme indicado na Figura 3.10.

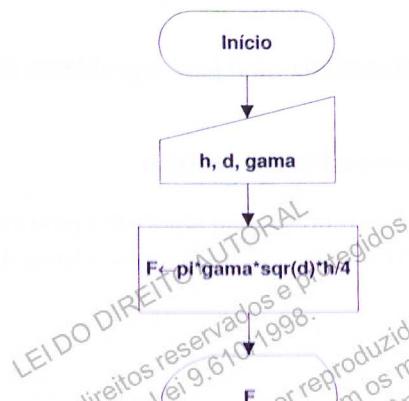


Figura 3.10 Passo 4 na construção do fluxograma para o problema da força.

Aqui, vale a mesma observação que foi feita quanto ao símbolo de entrada de dados. O símbolo de exibição de valores não especifica qual será o tipo de interface na qual o dado aparecerá (veja novamente a Figura 3.6).

Por fim, é necessário indicar o término do fluxograma por seu símbolo **Fim**. O fluxograma final é apresentado na Figura 3.11.

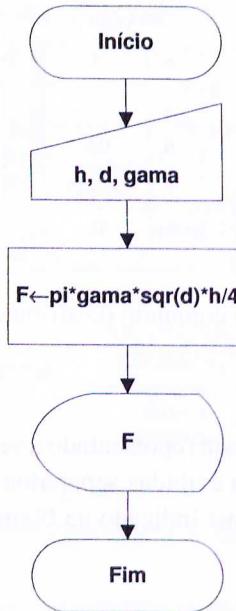


Figura 3.11 Fluxograma final para o problema da força.

3.5.3 Fluxograma com comandos de decisão

Deseja-se criar um fluxograma que represente o algoritmo para calcular as raízes de uma equação de segundo grau tipo $Ax^2 + Bx + C$, utilizando a fórmula de Bhaskara a seguir:

$$x = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

Esse algoritmo deverá gerar uma resposta para quaisquer valores de A , B e C fornecidos, como mostra o Algoritmo 3.5.

Nesse algoritmo foi usada a rotina predefinida *sqr* para representar a operação raiz quadrada (veja a Seção 3.9).

Observe com atenção a linha 2 do Algoritmo 3.5. Nota-se que é utilizado um comando de decisão. Nesse caso, é necessário verificar se $A \geq 0$ e, dependendo do resultado, tomar uma das seguintes decisões (mutualmente exclusivas):

- se for verdade, exiba mensagem “Não é equação de 2º grau” e então pare;
- senão, continuar em frente com os cálculos.

Algoritmo 3.5 Algoritmo para calcular as raízes de uma equação de 2º grau .**Início**

1. Ler A, B, C .
2. **Se** $A = 0$ **Então**
3. Exiba a mensagem “Não é equação de 2º grau!”.
4. **Senão** { A equação é de 2º grau.}
5. Calcule $D \leftarrow \text{sqr}(B) - 4 * A * C$
6. **Se** $D < 0$ **Então**
7. Exiba a mensagem “Não existem raízes reais!”.
8. **Senão** {Calcule as raízes.}
9. $r1 \leftarrow (-B + \text{sqrt}(D))/(2 * A)$
10. $r2 \leftarrow (-B - \text{sqrt}(D))/(2 * A)$
11. Exibir $r1$ e $r2$
12. **Fim Se**
13. **Fim Se**

Fim

Repetindo o mesmo que na Seção 3.5.2, o início do fluxograma fica conforme a Figura 3.12.



Figura 3.12 Passo 1 na construção do fluxograma para o problema das raízes.

A norma ISO 5807/1985 possui um símbolo específico para realizar esse tipo de operação – chamada **decisão** – que possibilita escolher um de dois caminhos a partir de um teste. Seu símbolo possui a forma de um losango, no interior do qual se escreve a expressão do teste a ser avaliado. Desse símbolo partem duas setas que, dependendo do valor da expressão, indicarão o caminho a ser seguido.

O resultado dessa expressão é um valor **lógico**, considerando um de dois valores possíveis: **verdadeiro** ou **falso**. Convencionamos escrever esses valores, respectivamente, pelas palavras em inglês **true** e **false** (veja a Seção 3.6).

Dessa forma, até a linha 2, o fluxograma ficaria de acordo com a Figura 3.13.

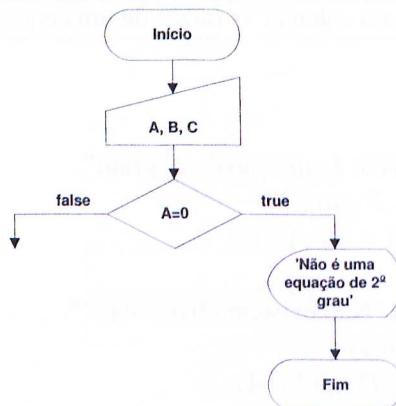


Figura 3.13 Passo 2 na construção do fluxograma para o problema das raízes.

Indicam-se nas retas que saem do símbolo de decisão os dois valores possíveis de um teste ou comparação (*true* e *false*), sendo, por convenção, *true* no lado direito e *false* no lado esquerdo. Mensagens constantes (cadeias de caracteres fixas) são delimitadas por apóstrofes (veja a Seção 3.6), para evitar que sejam confundidas com as variáveis.

O caminho a ser tomado após a avaliação da expressão lógica em um símbolo de decisão depende do resultado dessa expressão. A Figura 3.14 exibe dois casos que, dependendo do valor da variável *A*, vão conduzir a caminhos distintos.

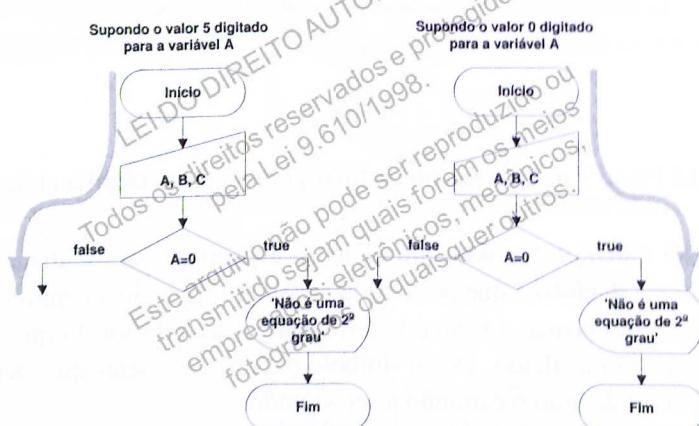


Figura 3.14 Encaminhamento após um comando de decisão.

O comando da linha 5 do Algoritmo 3.5 só deverá ser executado se o valor da variável A não for zero. Logo, o fluxograma fica conforme a Figura 3.15.

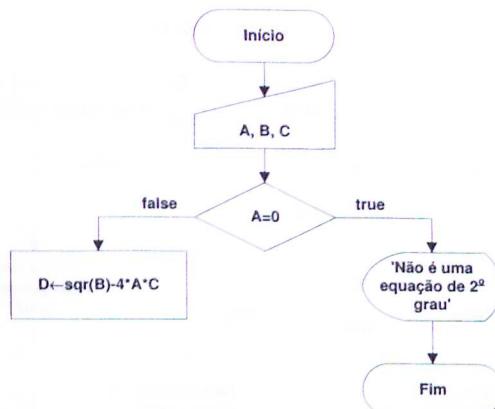


Figura 3.15 Passo 3 na construção do fluxograma para o problema das raízes.

Por fim, o comando da linha 6 do Algoritmo 3.5 introduz outra estrutura condicional. Seguindo o raciocínio análogo ao feito anteriormente, completa-se o fluxograma. Sua versão final está representada na Figura 3.16. Observe o uso do símbolo representado por uma “bolinha”. Ela não representa comando algum, mas uma forma de “juntar” os fluxos que provêm de caminhos diferentes.

Nota-se que na Figura 3.16 foram adicionadas anotações nas laterais dos símbolos utilizados, representando **comentários**. Não executam comando algum e servem para tornar mais claras as partes do fluxograma para o leitor. Os comentários tornam mais rápido o entendimento de um fluxograma, porém aqueles que forem óbvios devem ser evitados.

3.5.4 Fluxograma com comandos de repetição

Nesta seção serão apresentados os fluxogramas que contêm *comandos de repetição*. Para tanto, será utilizado como exemplo o algoritmo de Euclides para o cálculo de máximo divisor comum entre dois números inteiros, segundo o Algoritmo 3.2.

Antes de iniciar a construção do fluxograma propriamente dito, devem ser feitas algumas observações. Observe que na linha 2 desse algoritmo aparece a expressão $y \neq 0$. Seguindo as convenções que serão apresentadas na Seção 3.8, o símbolo para representar a **desigualdade** será o $<>$ (justaposição dos símbolos $<$ e $>$). Já na linha 3, deve

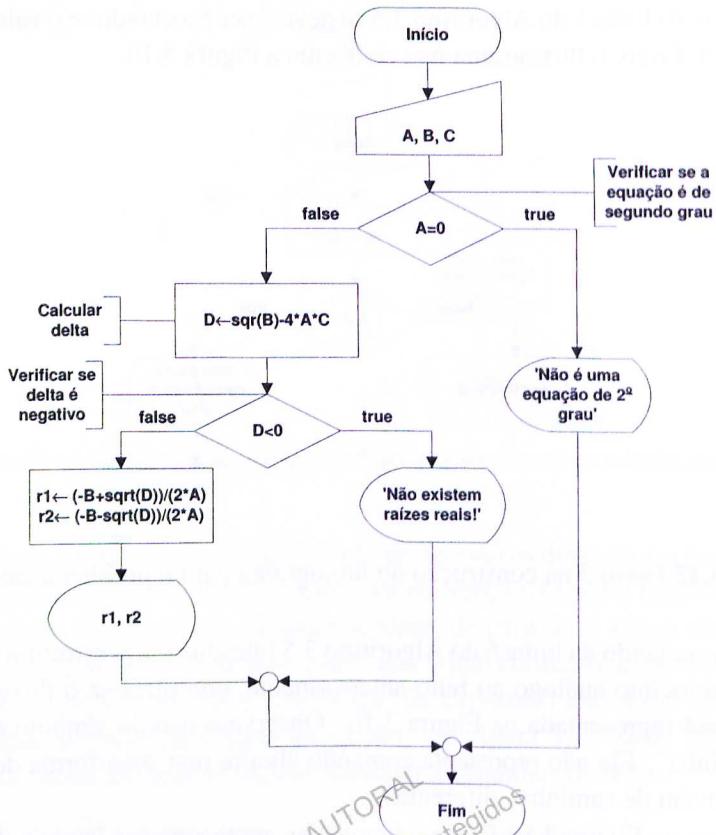


Figura 3.16 Fluxograma final, comentado, para o problema das raízes.

ser executado o cálculo do **resto da divisão entre dois números inteiros**. Acompanhando as convenções que serão apresentadas na Seção 3.8, o símbolo de operador para representar essa operação será **mod**.

O fluxograma para representar esse algoritmo está na Figura 3.17. Devem ser feitos alguns comentários:

1. Embora seja empregado nesta seção o mesmo símbolo do comando de decisão da norma ISO 5807/1985, aqui ele possui um significado diferente: é parte integral de um comando de repetição, servindo como um teste para indicar se os comandos em seu interior deverão ser executados novamente.
2. A repetição é indicada pelo caminho fechado que sai do símbolo de decisão e que

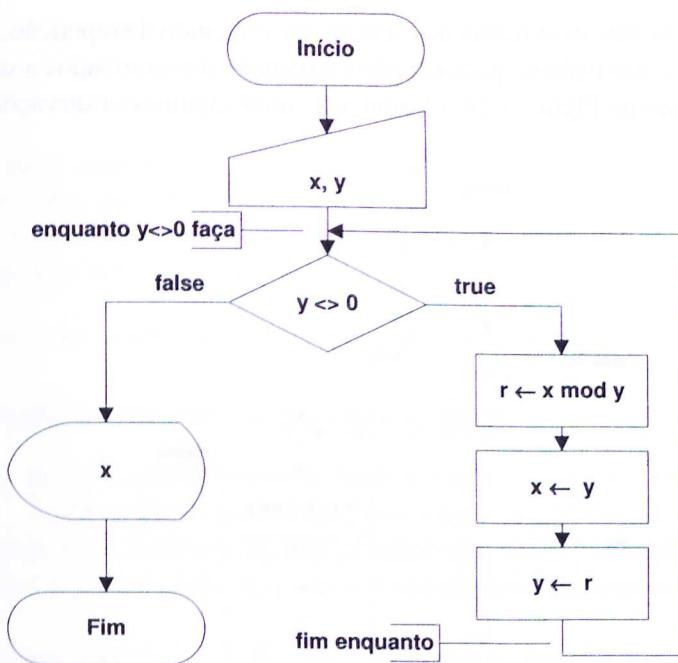


Figura 3.17 Fluxograma para o algoritmo de Euclides.

volta para ele.

3. A linha de retorno deve ser sempre desenhada imediatamente antes do símbolo de decisão.
4. A expressão lógica escrita internamente ao símbolo de decisão, no caso de comandos de repetição, representa um **critério ou condição de parada** da repetição. No caso desse fluxograma, os símbolos desenhados no caminho fechado representam os comandos que deverão ser executados **enquanto** a condição $y \neq 0$ for verdadeira. O bloco que exibe o resultado contido na variável x apenas será executado quando a condição $y \neq 0$ for falsa. Esse comando de repetição será formalizado no Capítulo 4 com o nome de estrutura **enquanto-faça**.
5. As expressões de condição de parada devem ser testadas, pois, se estiverem erradas poderão levar a uma **repetição infinita**. Por exemplo, se for trocada a condição $y \neq 0$ para $y \geq 0$ no fluxograma apresentado na Figura 3.17, nunca se alcançará o símbolo **Fim**, tornando-o infinito, portanto, deixando de representar um algoritmo.

Não existe somente essa forma de escrever um comando de repetição. De fato, é possível posicionar a condição de parada após a execução dos comandos a serem repetidos, conforme ilustrado na Figura 3.18. Cabem aqui mais algumas observações:

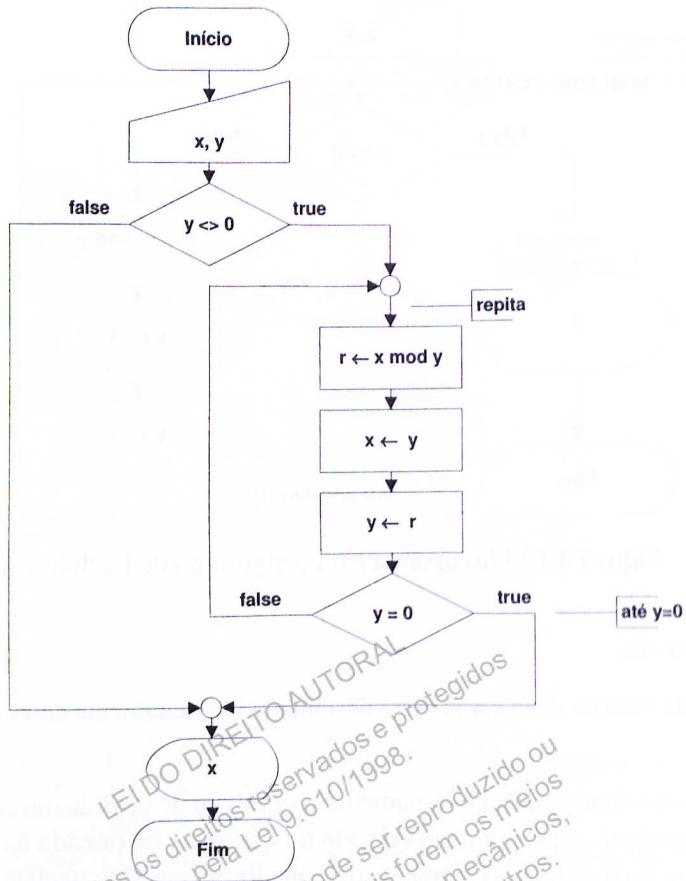


Figura 3.18 Outro fluxograma para o algoritmo de Euclides.

1. Note que o comando de repetição foi escrito agora com a condição de parada *após* os comandos que se desejam repetir. Esse comando pode ser entendido como “**repita** os comandos **até que** a condição de parada seja verdadeira”. Com efeito, esse comando será formalizado em uma estrutura de programação denominada **repita-até**, a ser apresentada no Capítulo 4.
2. Como esse comando de repetição primeiro executa os comandos para depois ve-

rificar a condição de parada, foi necessário incluir um teste ($y \neq 0$) antes dessa repetição, pois, caso contrário, existe a possibilidade da realização de uma divisão por zero (no comando $r \leftarrow x \bmod y$).

3. A condição de parada mudou. Nesse caso, a repetição será executada se a condição $y = 0$ for falsa, que, em termos lógicos, é o mesmo que executar a repetição se a condição $y \neq 0$ for verdadeira. Nota-se, ainda, que a repetição termina quando a condição $y = 0$ é verdadeira.

Os símbolos vistos neste capítulo até este ponto estão resumidos na Tabela 3.2.

3.5.5 Simulação de algoritmos com fluxogramas

A simulação de um fluxograma é feita da mesma maneira que a de um algoritmo (veja a Seção 3.2.2). A vantagem é que os símbolos possuem um significado preciso e as setas que conectam esses símbolos permitem seguir o fluxo das instruções de uma forma mais visível. Uma sequência para se entender e simular um fluxograma foi definido pelo Algoritmo 3.3.

Como exemplo final desta seção, será projetado e testado um fluxograma para resolver o seguinte problema: exibir a média de N temperaturas fornecidas pelo usuário.

Deve-se entender o problema, com a experiência pessoal obtida ou por meio de dados fornecidos. Os dados do problema induzem:

1. O número de temperaturas é variável e indeterminado, representado simbolicamente pela variável N .
2. Para realizar a média de N temperaturas é necessário, primeiramente, que esses N valores sejam fornecidos pelo usuário.
3. Não é possível resolver esse problema se $N = 0$, pois não tem sentido fazer a média de um conjunto com zero ou menos valores.
4. Com esses valores, agora é possível calcular a média: basta somar todos os N valores e depois dividir o resultado por N .

A primeira questão que surge é a seguinte: se o número N de valores de temperatura for indeterminado e se para cada temperatura for necessária uma variável para armazená-la, como resolver o problema?

Na realidade, a solução é bem simples. Esse problema não exige que se armazenem todas as variáveis de temperatura (como armazenar conjuntos variáveis de dados será

Tabela 3.2 Resumo dos símbolos vistos no Capítulo 3.

Símbolo	Nome	Utilidade
	<i>Terminador</i>	Representar a saída para ou entrada do ambiente externo, por exemplo, início ou final de programa, uso externo e origem ou destino de dados etc.
	<i>Processo</i>	Representar qualquer tipo de processo, processamento de função, por exemplo, executando uma operação definida ou grupo de operações, resultando na mudança de valor, forma ou localização de uma informação ou determinação de uma, entre as várias direções de fluxo a serem seguidas.
	<i>Linha básica</i>	Representar o fluxo dos dados ou controles. Podem ser utilizadas pontas de seta, sólidas ou abertas, na extremidade para indicar a direção do fluxo onde necessário ou para enfatizá-lo e facilitar a legibilidade.
	<i>Entrada manual</i>	Representar os dados, de qualquer tipo de mídia, que sejam fornecidos, manualmente, em tempo de processamento, por exemplo, teclado <i>on-line</i> , mouse, chaveamento, caneta óptica <i>light pen</i> , leitor de código de barras etc.
	<i>Exibição</i>	Representar os dados, cuja mídia seja de qualquer tipo, na qual a informação seja mostrada para uso humano, tais como monitores de vídeo, indicadores <i>on-line</i> , mostradores etc.
	<i>Decisão</i>	Representar uma decisão ou um desvio tendo uma entrada; porém pode ter uma série de saídas alternativas, uma única das quais deverá ser ativada como consequência da avaliação das condições internas ao símbolo. O resultado apropriado de cada saída deverá ser escrito adjacente à linha, representando o caminho respectivo.

visto no Capítulo 5). Particularizando o problema, se N fosse igual a um, o problema seria resolvido de forma direta, bastando apenas exibir a única temperatura digitada.

No entanto, existem N temperaturas a considerar e sabe-se que o problema somente será resolvido de forma correta para os valores de N maior que zero. Como não se deve confiar nos valores que o usuário digita, uma primeira versão do fluxograma que conte com a solução desse problema deverá permitir a entrada do valor N e então decidir se N é maior que zero ou não. No caso negativo, basta exibir uma mensagem de erro adequada e, então, terminar, conforme ilustrado pela Figura 3.19.

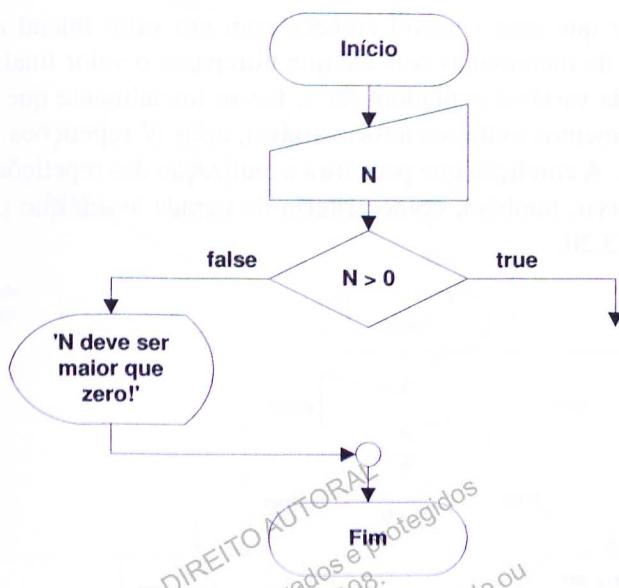


Figura 3.19 Passo 1 na construção do fluxograma para o problema das temperaturas.

Agora surge o ponto crucial: como ler, somar e realizar a média de todas as temperaturas? Basta pensar na utilização de um comando de repetição dentro do qual será lido um valor de temperatura e, a seguir, acrescentá-lo aos outros que já foram somados. Quando se somarem todos os valores, basta realizar a divisão por N .

Conduzindo a solução por partes, primeiro é preciso saber como controlar o número de repetições. Deseja-se que, para um valor $N > 0$, sejam repetidas a leitura e a soma de valores. Para criar um comando de repetição que vai executar N repetições, é necessária uma variável adicional, denominada comumente *variável contadora*, ou simplesmente, *contador*.

A ideia é fazer que essa variável comece com um valor inicial adequado e então repetir a instrução de incremento dela até que ultrapasse o valor final, que, nesse caso, é N . Se o nome da variável contadora for i , faz-se inicialmente que i tenha o valor 1 e, realizando incrementos unitários nessa variável, após N repetições, ela alcançará um valor maior que N . A condição que permitirá a realização das repetições pode ser escrita como $i \leq N$ e servir, também, como critério de parada assim que $i > N$, conforme descrito na Figura 3.20.

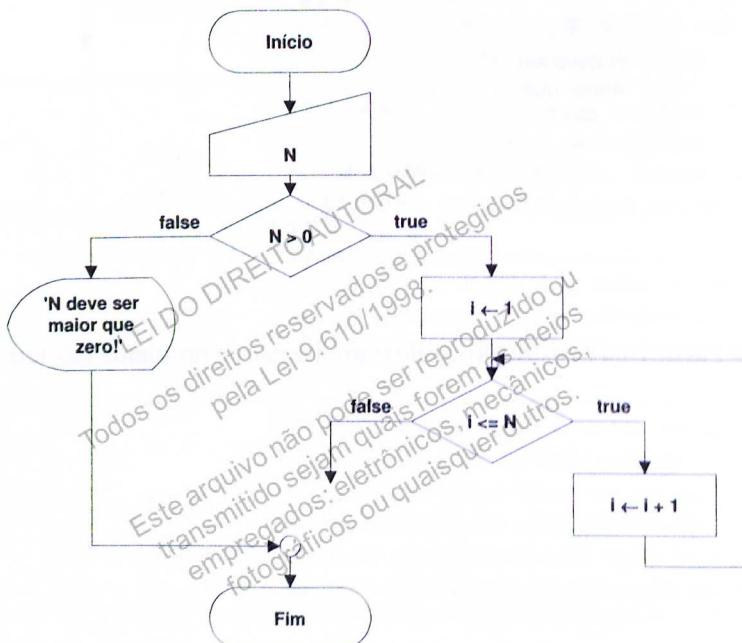


Figura 3.20 Passo 2 na construção do fluxograma para o problema das temperaturas.

Deve-se observar que a instrução $i \leftarrow i + 1$ é interpretada da seguinte forma: soma-se

1 ao valor da variável i e então atribui-se esse valor à própria variável i . Como uma variável representa uma área da memória que contém um valor, nada impede que esse valor seja somado a 1 e então gravado de volta na mesma posição.

Agora que já existe uma forma de repetir N vezes alguma instrução, este é o momento de introduzir o comando de leitura da temperatura. Utilizando a variável T para representar uma temperatura, coloca-se o comando de leitura no interior da repetição apresentada, de acordo com a Figura 3.21.

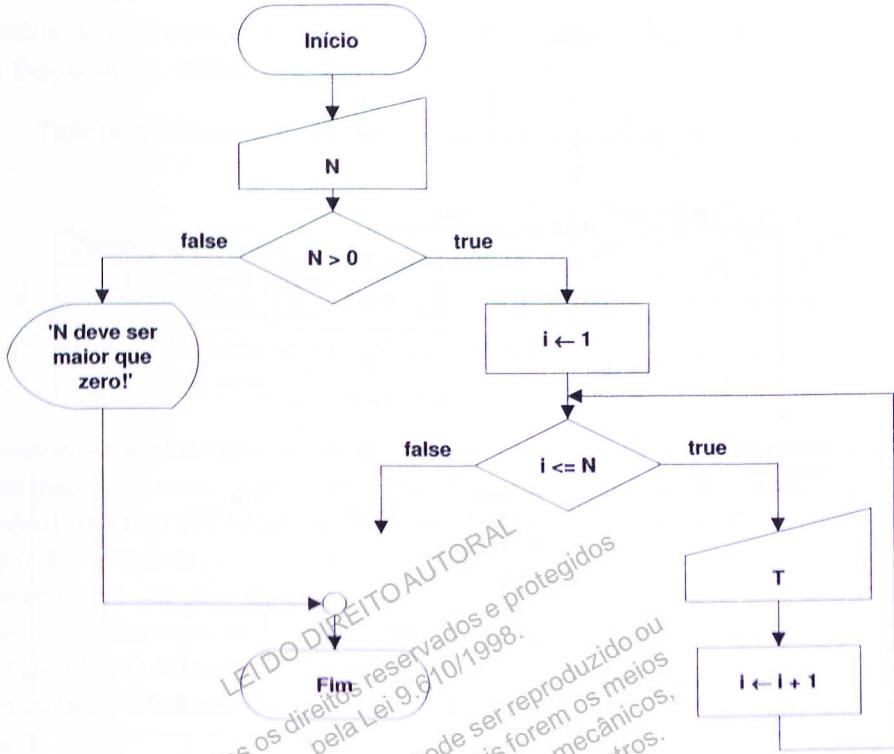


Figura 3.21 Passo 3 na construção do fluxograma para o problema das temperaturas.

O fluxograma da Figura 3.21 permite a entrada de N temperaturas, porém, armazena-se apenas o último valor delas, pois se está utilizando uma única variável para guardar a temperatura, a variável T . No próximo passo da construção do fluxograma, deve-se adicionar um comando que possibilite a soma dessa variável com a soma parcial existente anteriormente.

Para se fazer isso, basta acrescentar mais uma variável ao problema, a qual armazenará a soma das temperaturas (uma variável com esse propósito é denominada

acumulador). A técnica para isso é iniciar uma variável fora do comando de repetição com o valor zero, por exemplo, $S \leftarrow 0$ e, então, após a leitura de uma temperatura, somar essa temperatura a essa variável, como em $S \leftarrow S + T$. Assim, a cada repetição, lê-se uma temperatura e, então, prontamente é feita sua soma com a soma anterior.

Por fim, após a execução das N somas, basta dividir o valor da variável S por N , obtendo-se a média e, então, exibindo esse valor para o usuário. O fluxograma final está representado na Figura 3.22.

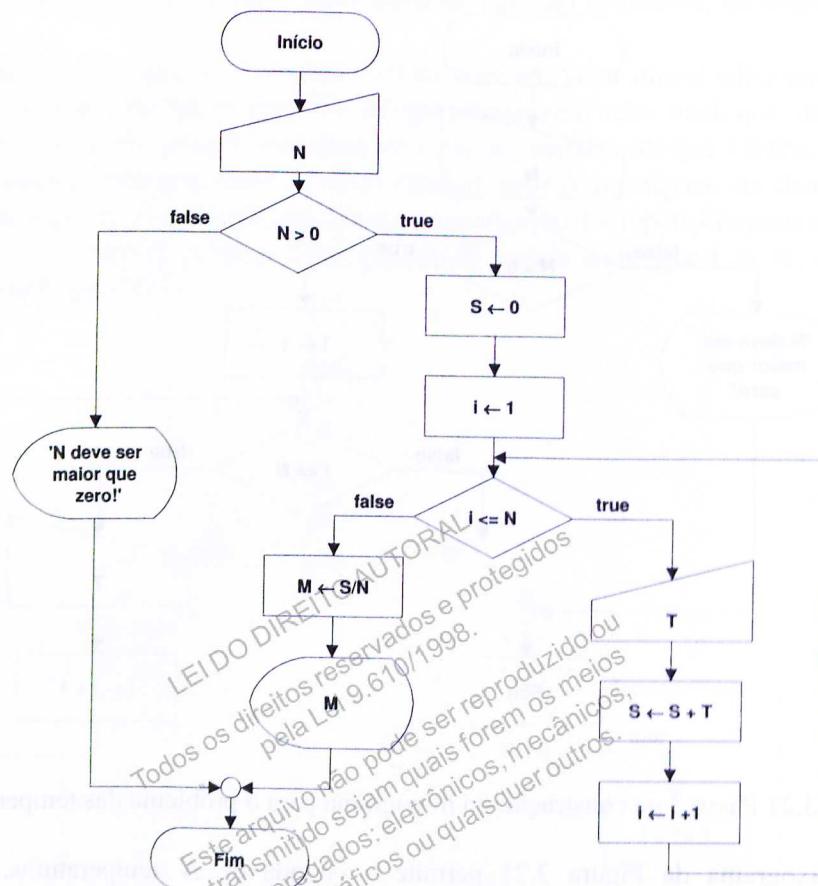


Figura 3.22 Fluxograma final para o problema das temperaturas.

Agora é o momento de verificar se esse fluxograma está correto ou não. É necessário realizar testes com diversos valores de entrada (corretos ou não) e, ao simular o fluxograma, sempre alcançar o símbolo **Fim** com valores coerentes.

Serão definidos dois conjuntos de testes:

- com valores de entrada suspeitos: $N = -1$ e temperaturas da lista (10, 11, 12);
- com valores de entrada corretos: $N = 3$ e temperaturas da lista (10, 11, 12).

Utilizando o primeiro conjunto de valores e seguindo o Algoritmo 3.3 de interpretação de fluxogramas, obtém-se a Tabela 3.3.

Tabela 3.3 Simulação do fluxograma com valores de entrada errados.

Passo	Comando	Valor das variáveis				
		<i>N</i>	<i>S</i>	<i>i</i>	<i>T</i>	<i>M</i>
1	Entrada (digitar: -1)	-1	?	?	?	?
2	Decisão (<i>false</i> : $N < 0$)	-1	?	?	?	?
3	Exibição (exibir: ‘N deve ser maior que zero!’)	-1	?	?	?	?

Nesse caso, o fluxograma apresentou como resposta a exibição da mensagem ‘N deve ser maior que zero!’ e terminou. Isso era esperado, já que para os valores negativos de N não deveria ser produzida nenhuma média. Realizando a simulação com o segundo conjunto de valores de entrada, obtém-se a Tabela 3.4.

Observe que por essa simulação verificou-se que o fluxograma da Figura 3.22 fornece os resultados corretos também. Basta checar: com $N = 3$ valores digitados que foram (10, 11, 12) foi produzido e exibido o valor $M = 11$, que é a média desses valores particulares. É interessante, para praticar, testar esse mesmo fluxograma com outros valores de teste.

3.6 Convenções para tipos de dados

Um fluxograma não indica explicitamente o tipo do valor que opera. Números como 4 e 67 podem ser utilizados como números inteiros ou reais. Por sua vez, números como 45,78 e 55,0987 são com certeza números reais. Existem ainda os tipos para as cadeias de caracteres, caracteres simples e valores lógicos. Assim, com o objetivo de implementar futuramente os algoritmos em uma linguagem de programação, será convencionado o formato desses tipos de dados.

Tabela 3.4 Simulação do fluxograma com valores de entrada corretos.

Passo	Comando	Valor das variáveis				
		<i>N</i>	<i>S</i>	<i>i</i>	<i>T</i>	<i>M</i>
1	Entrada (digitar: 3)	3	?	?	?	?
2	Decisão (true: $N > 0$)	3	?	?	?	?
3	Processo ($S \leftarrow 0$)	3	0	?	?	?
4	Processo ($i \leftarrow 1$)	3	0	1	?	?
5	Decisão (true: $i \leq N$)	3	0	1	?	?
6	Entrada (digitar: 10)	3	0	1	10	?
7	Processo ($S \leftarrow S + T$)	3	10	1	10	?
8	Processo ($i \leftarrow i + 1$)	3	10	2	10	?
9	Decisão (true: $i \leq N$)	3	10	2	10	?
10	Entrada (digitar: 11)	3	10	2	11	?
11	Processo ($S \leftarrow S + T$)	3	21	2	11	?
12	Processo ($i \leftarrow i + 1$)	3	21	3	11	?
13	Decisão (true: $i \leq N$)	3	21	3	11	?
14	Entrada (digitar: 12)	3	21	3	12	?
15	Processo ($S \leftarrow S + T$)	3	33	3	12	?
16	Processo ($i \leftarrow i + 1$)	3	33	4	12	?
17	Decisão (false: $i > N$)	3	33	4	12	?
18	Processo ($M \leftarrow S/N$)	3	33	4	12	11
19	Exibição (exibir: 11)	3	33	4	12	11

3.6.1 Números

Os números manipulados em um algoritmo podem ser inteiros ou reais. Os números inteiros são escritos sem separador de decimal, como, por exemplo, 10, 334, 13 etc. Os números reais serão escritos separando-se a parte decimal com um *ponto*. Portanto, entende-se como números reais os valores como 3,4415, 45,98, 1,0 etc.

Alternativamente, esses números reais (principalmente quando muito grandes) podem ser escritos em notação científica (ou notação exponencial), expressando o número em potências de dez. Dessa forma, a constante de Avogadro, 6.02×10^{23} , pode ser escrita em um algoritmo com a seguinte notação científica: 6.02E23. Utiliza-se, por conseguinte, o símbolo *E* para separar o número de sua potência de dez.

Para os números negativos, precede-se o número em questão com o sinal “-”. Assim, são exemplos de números negativos: -3, -0.9, -12 etc. Em notação científica, se o expoente for negativo, precede-se seu valor com o sinal “-”. Exemplos: 4.5E-3, -3.5E-45 etc.

3.6.2 Caracteres e cadeias de caracteres

Os caracteres representam uma única letra ou símbolo de texto. Os caracteres em um algoritmo serão representados limitando-os com apóstrofes. Será utilizada essa convenção para evitar que haja confusões entre os caracteres e os nomes de variáveis. Dessa maneira, o caractere *W* será representado em um algoritmo como ‘W’. Com essa notação, não há perigo de confundir *h*, que se entende por nome de variável, com ‘h’, que se entende pelo caractere *h* minúsculo.

Os símbolos fora do alfabeto também são representados como caracteres: ‘@’, ‘\’, ‘/’ etc. O espaço em branco entra nessa categoria, também: ‘ ’. As cadeias de caracteres ou simplesmente *strings* (cordões de caracteres) representam um conjunto ordenado de caracteres, significando palavras, mensagens ou ainda pequenos textos. Uma cadeia de caracteres será representada, limitando-a com apóstrofes. Assim, são exemplos de cadeias de caracteres: ‘Olá’, ‘123XYZ’, ‘098-33#’ etc.

As cadeias de caracteres podem conter quaisquer caracteres, incluindo os espaços em branco. Dessa forma, as cadeias ‘Bom dia’, ‘Erro no Aplicativo’ contêm espaços em branco. Deve-se ter cuidado com a diferença entre os números e uma cadeia de caracteres que contém caracteres que representam números. São completamente diferentes os valores 123, que é um inteiro, de ‘123’, que é uma cadeia que contém os caracteres ‘1’, ‘2’, e ‘3’.

Normalmente, em linguagens de programação, existe uma limitação para o número de caracteres em uma cadeia. Será convencionado o limite de 255 caracteres.

3.6.3 Valores lógicos

Os valores lógicos ou ainda booleanos (em homenagem a George Boole, que elaborou a lógica booleana) são aqueles que representam apenas dois estados: um estado verdadeiro ou um estado falso. Os valores booleanos serão convencionados segundo a grafia inglesa. Assim, o valor **verdadeiro** será escrito **true** e o valor **falso**, **false**. O uso desses valores em expressões lógicas será estudado adiante.

3.7 Convenções para os nomes de variáveis

As variáveis utilizadas em um algoritmo devem ser escritas de modo claro, inteligível. Nesse sentido, convencionam-se algumas regras para nomear as variáveis:

1. Os nomes podem ter até 63 caracteres de comprimento.

2. Os nomes devem ser iniciados por um caractere alfabético (letra) ou pelo caractere ‘_’.
3. Os nomes podem possuir números, desde que se inicie por letra.
4. Além de letras, números e o caractere ‘_’, não é aceito nenhum outro símbolo.
5. Letras fora do alfabeto ocidental, como as letras gregas, não são aceitas.
6. Não se fará diferenciação entre as letras maiúsculas e minúsculas nos nomes de variáveis.

Exemplos de nomes válidos de variáveis: *A*, *Ba12*, *A1543T*, *CustoTotal*, *Pagamento*, *Nome*, *Lista*, *Contador*, *Valores*, *Indice*, *Resultado_Final*, *C3PO*, *NCC1701*, *R2D2*.

Exemplos de nomes inválidos: *1QA*, por começar com valor numérico; *Preço*, por possuir caracter especial (ç); *C&A* pelo mesmo motivo anterior (&); *Resultado Final* (espaço em branco); *Media – final*, pois o sinal de menos é reservado para a operação subtração.

3.8 Convenções para as expressões

3.8.1 Operação de atribuição

A atribuição é a operação que permite armazenar um valor em uma variável. Para essa operação, será utilizado o símbolo \leftarrow (seta para a esquerda), e a variável que receberá esse valor ou resultado de uma expressão deverá estar do lado esquerdo da seta.

Dessa forma, pode-se ler a expressão $A \leftarrow 6$ como **armazenar o valor 6 na variável A** ou ainda como **A recebe 6**. Ainda é possível se escrever $A \leftarrow A + 1$ que significa **A recebe o valor de A + 1**. O símbolo \leftarrow foi escolhido para evitar confusões com o operador $=$, reservado para a realização de comparações.

Em um fluxograma, a atribuição somente pode ser utilizada em blocos que representam processos ou comandos (veja a Figura 3.23).

Nesse exemplo, ao se entrar com um valor para a variável *A*, será obtido como resultado o valor original acrescido de 1. Se *A* for 5, o resultado exibido será 6.

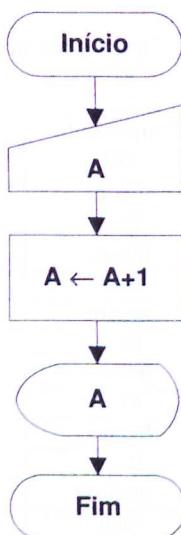


Figura 3.23 Exemplo do comando de atribuição.

3.8.2 Operações aritméticas

São definidas as quatro operações aritméticas: adição, subtração, multiplicação e divisão. Dependendo dos operadores e do tipo de resultado, a divisão pode ter de ser feita de forma diferente: a **divisão real**, a **divisão inteira** e o **resto da divisão inteira**. Essas operações matemáticas são consideradas operações binárias, pois envolvem sempre dois operandos.

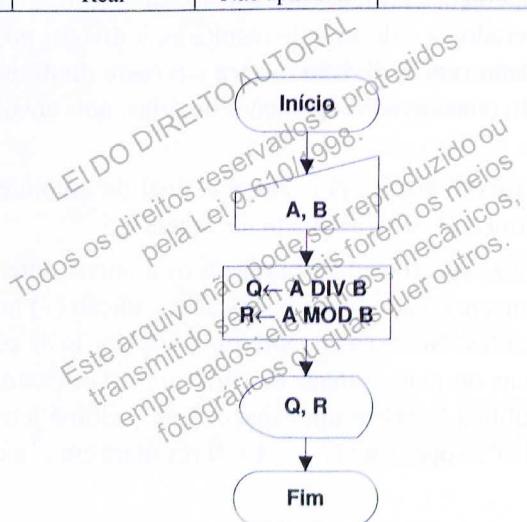
A operação troca de sinal é unária, pois altera o sinal de um único operando. Um resumo das operações aritméticas é apresentado na Tabela 3.5.

As operações aritméticas apresentadas abrangem os números inteiros e os números reais. No entanto, será convencionado que a operação de adição (+) também poderá ser aplicada à cadeia de caracteres. Nesse caso, representa a operação de **concatenação**, isto é, permite a junção de duas ou mais cadeias de caracteres. Por exemplo, se a variável *A* contém a cadeia 'República' (existe um espaço após a última letra) e a variável *B* contém a cadeia 'Federativa', a operação $C \leftarrow A + B$ resultará em *C* a cadeia 'República Federativa'.

Como exemplo do uso de operações aritméticas com números, segue na Figura 3.24 o fluxograma que, dado dois números inteiros *A* e *B*, calcula o cociente *Q* e o resto *R* da divisão inteira de *A* por *B*.

Tabela 3.5 Operações aritméticas.

Operação	1º operando (A)	2º operando (B)	Tipo resultante (C)	Simbologia
Adição	Inteiro	Inteiro	Inteiro	$C \leftarrow A + B$
	Real	Real	Real	
	Real	Inteiro	Real	
	Inteiro	Real	Real	
Subtração	Inteiro	Inteiro	Inteiro	$C \leftarrow A - B$
	Real	Real	Real	
	Real	Inteiro	Real	
	Inteiro	Real	Real	
Multiplicação	Inteiro	Inteiro	Inteiro	$C \leftarrow A * B$
	Real	Real	Real	
	Real	Inteiro	Real	
	Inteiro	Real	Real	
Divisão real	Inteiro	Inteiro	Real	$C \leftarrow A / B$
	Real	Real	Real	
	Real	Inteiro	Real	
	Inteiro	Real	Real	
Divisão inteira	Inteiro	Inteiro	Inteiro	$C \leftarrow A \text{ div } B$
Resto	Inteiro	Inteiro	Inteiro	$C \leftarrow A \text{ mod } B$
Troca de sinal	Inteiro	Não aplicável	Inteiro	$C \leftarrow -A$
	Real	Não aplicável	Real	

**Figura 3.24** Exemplo de operadores aritméticos.

3.8.3 Operações relacionais

As operações relacionais permitem efetuar comparações entre duas variáveis. Essas operações são largamente utilizadas em estruturas condicionais e em repetitivas. Já que essas operações dependem de testes de valores e, **todo resultado de um teste só pode ser *true* ou *false***, seu resultado é um valor **booleano**. Os operadores relacionais são apresentados na Tabela 3.6.

Tabela 3.6 Operações relacionais.

Operador	Significado	Resultado
=	Igualdade	Booleano (<i>true</i> ou <i>false</i>). Sendo <i>true</i> , se o 1º operando for igual ao 2º e <i>false</i> , se diferentes.
<	Menor	Booleano (<i>true</i> ou <i>false</i>). Sendo <i>true</i> , se o 1º operando for menor que o 2º e <i>false</i> , caso contrário.
>	Maior	Booleano (<i>true</i> ou <i>false</i>). Sendo <i>true</i> , se o 1º operando for maior que o 2º e <i>false</i> , caso contrário.
<=	Menor ou igual	Booleano (<i>true</i> ou <i>false</i>). Sendo <i>true</i> , se o 1º operando for menor ou igual ao 2º e <i>false</i> , caso contrário.
>=	Maior ou igual	Booleano (<i>true</i> ou <i>false</i>). Sendo <i>true</i> , se o 1º operando for maior ou igual ao 2º e <i>false</i> , caso contrário.
<>	Diferente	Booleano (<i>true</i> ou <i>false</i>). Sendo <i>true</i> , se o 1º operando for diferente do 2º e <i>false</i> , caso sejam iguais.

As operações relacionais também são definidas para as cadeias de caracteres. Nesse caso, vale a avaliação **lexicográfica**, como nos dicionários (mas se baseando na ordem da tabela ASCII). Assim, se a variável *A* possuir a cadeia ‘Banana’ e se a variável *B* contém ‘Banana d’água’, então *A* < *B* é *true*; *A* > *B* é *false* e assim por diante. A comparação de cadeias é sensível ao caso; ‘Banana’, ‘banana’ e ‘Banana’ < ‘banana’ (pois o caractere ‘B’ tem código ASCII menor que o caractere ‘b’).

3.8.4 Operações lógicas

São operações efetuadas com os valores booleanos (*true* ou *false*), resultando em valores booleanos. Essas operações são largamente utilizadas em estruturas condicionais e em repetitivas, já que essas estruturas dependem de testes de valores. Os operadores lógicos têm a função de advérbio na linguagem de programação. Cada operador lógico tem a

sua “tabela verdade” composta por: **not**, **and** e **or**.

O operador **not** representa a negação do valor booleano – sua inversão. O operador **and** realiza o teste simultâneo de duas condições – resulta o valor *true* somente quando ambos operandos são *true*. Já para o operador **or**, basta que uma das condições seja *true*, para que o resultado seja *true*. A Tabela 3.7 exibe a tabela verdade dos operadores lógicos.

Tabela 3.7 Operações lógicas.

Operador lógico	1º operando (A)	2º operando (B)	Resultado (C)	Simbologia
<i>NOT</i>	<i>true</i>	Não aplicável	<i>false</i>	$C \leftarrow \text{not } A$
	<i>false</i>	Não aplicável	<i>true</i>	
<i>AND</i>	<i>true</i>	<i>true</i>	<i>true</i>	$C \leftarrow A \text{ and } B$
	<i>true</i>	<i>false</i>	<i>false</i>	
	<i>false</i>	<i>true</i>	<i>false</i>	
	<i>false</i>	<i>false</i>	<i>false</i>	
<i>OR</i>	<i>true</i>	<i>true</i>	<i>true</i>	$C \leftarrow A \text{ or } B$
	<i>true</i>	<i>false</i>	<i>true</i>	
	<i>false</i>	<i>true</i>	<i>true</i>	
	<i>false</i>	<i>false</i>	<i>false</i>	

Assim, considerando-se as seguintes variáveis com valores iniciais indicados, $A = 3$, $B = 7$, $C = -1$, a expressão $\text{not}((C > 0) \text{ and } ((B - A) > 2))$ resulta em *true*, pois $C > 0$ é *false* e $((B - A) > 2)$ é *true*, resultando a operação *and* aplicado nessas expressões em *false*. Por fim, o resultado $\text{not}(\text{false})$ fornece seu inverso lógico, ou seja, *true*.

3.8.5 Expressões

Se prestarmos atenção, todas as operações discutidas são levadas a efeito com um ou dois operandos de cada vez. Uma expressão contendo diversos operandos deve ser avaliada de acordo com a **precedência** dos operadores envolvidos. A precedência dos operadores indica, em uma expressão, qual operação será realizada antes das outras.

Os operadores aritméticos possuem a seguinte precedência, do maior para o menor:

1. Troca de sinal (“-” unário).
2. $*$, $/$, mod , div na ordem em que aparecerem.
3. $+$ e $-$.

Por exemplo, a expressão $A \leftarrow 24/6/2 - 127 \text{ div } 7 \text{ mod } 5$ deve se assim avaliada:

$$\begin{aligned} A &\leftarrow 2 - 18 \text{ mod } 5 \\ A &\leftarrow 2 - 3 \\ A &\leftarrow -1 \end{aligned}$$

De forma diferente da Matemática, na qual as expressões podem ser agrupadas com chaves, colchetes e parênteses, aqui o único elemento de agrupamento de expressões válido serão os parênteses. Assim, em expressões contendo parênteses, a precedência é ligeiramente alterada:

1. Resolver o nível mais interno de parênteses, usando a precedência de operações:
 - (a) Troca de sinal (“–” unário).
 - (b) \ast , $/$, mod , div na ordem em que aparecerem.
 - (c) $+$ e $-$.
2. Passar para o próximo nível, sempre do mais interno para o mais externo, utilizando a precedência de operações.

No caso de operadores lógicos, a precedência é (do maior para o menor): ***not, and*** e por fim ***or***. Esta última também pode ser alterada, da mesma forma, pelo uso de parênteses. Finalmente, em um último nível da hierarquia dos operadores, têm-se os operadores relacionais.

A Tabela 3.8 fornece a precedência de todos os operados vistos até este ponto.

Tabela 3.8 Precedência dos operadores.

Hierarquia	Operadores
1	<i>not</i>
2	<i>*, /, div, mod, and</i>
3	<i>+, -, or</i>
4	<i><, <=, =, ></i>

3.9 Sub-rotinas predefinidas

Até este ponto foram apresentados os tipos de dados básicos para a construção de fluxogramas e suas operações. No entanto, surge um problema, caso seja necessário definir as

operações com funções matemáticas conhecidas, como *seno*, *logaritmos* e outras. Além disso, a única operação que foi apresentada para as cadeias de caracteres foi a concatenação. Como fazer, então, para se extrair parte de um nome, por exemplo?

Em princípio, todas as operações necessárias poderiam ser definidas via novos algoritmos. Entretanto, este é um trabalho desnecessário, pois a maior parte das linguagens de programação (em uma continuação do estudo apresentado neste livro) mostra um conjunto de **sub-rotinas** que resolvem problemas básicos matemáticos e de cadeias de caracteres.

Considere, por enquanto, como uma sub-rotina um algoritmo que foi escrito e que está pronto para o uso. Uma sub-rotina pode apresentar **parâmetros**, que servem para se passar valores a serem calculados. Uma sub-rotina tipo **função** é aquela que produz um valor diretamente. E uma sub-rotina tipo **procedimento** é a que não produz um valor ou produz um valor indiretamente, via um de seus parâmetros. Os conceitos de como escrever novas sub-rotinas é assunto do Capítulo 6.

Nas seções a seguir serão convencionadas as sub-rotinas válidas para serem utilizadas neste livro. Qualquer operação não mencionada nas tabelas a seguir poderão ser implementadas com o uso de uma ou mais sub-rotinas apresentadas.

3.9.1 Funções matemáticas

Consideram-se válidas as seguintes funções predefinidas, comuns a várias linguagens de programação de alto nível, expostas na Tabela 3.9.

Os argumentos dessas funções podem ser números reais ou inteiros e o tipo de resultado produzido é apresentado na própria tabela. Dessa tabela, seguem algumas observações. Na primeira delas, as funções *int* e *frac* servem para “destrinchar” números reais, resultando em novos números reais. Por exemplo, a expressão $A \leftarrow \text{int}(3.1415)$ resulta em $A = 3.0$ e a expressão $A \leftarrow \text{frac}(3.1415)$, em $A = 0.1415$. Já as funções *trunc* e *round* servem, respectivamente, para truncar e arredondar um número real em um número inteiro. Dessa forma, a expressão $A \leftarrow \text{trunc}(3.1415)$ resulta em $A = 3$ (inteiro), e a expressão $A \leftarrow \text{round}(3.1415)$, em $A = 3$ (o arredondamento é realizado para o inteiro mais próximo).

Na segunda, as funções trigonométricas apresentadas utilizam e resultam ângulos em radianos. Assim, caso seja necessário calcular o seno de 33° , esse arco deverá ser convertido em radianos com a expressão $33 \times \pi / 180$, resultado em ≈ 0.58 radianos.

Um exemplo de aplicação das funções dessa tabela é o fluxograma para calcular x^y para todo $x > 0$. Nota-se que não existe uma função pronta para realizar esse cálculo. No entanto, uma expressão equivalente utilizando funções da Tabela 3.9 pode ser

Tabela 3.9 Funções matemáticas.

Função	Utilidade	Tipo do resultado
$\ln(x)$	Retorna o valor do logaritmo neperiano de x .	Real
$\exp(x)$	Retorna o valor de e^x .	Real
$\text{int}(x)$	Retorna a parte inteira de x .	Real
$\text{frac}(x)$	Retorna a parte fracionária de x .	Real
$\text{trunc}(x)$	Retorna a parte inteira de x .	Inteiro
$\text{round}(x)$	Arredonda x para o próximo inteiro.	Inteiro
$\text{sqr}(x)$	Retorna x^2 .	Real
\sqrt{x}	Retorna \sqrt{x} .	Real
$\sin(x)$	Retorna o seno de x .	Real
$\cos(x)$	Retorna o cosseno de x .	Real
$\arctan(x)$	Retorna o arco, em radianos, cuja tangente é x .	Real
$\text{abs}(x)$	Retorna o módulo de x ($ x $).	Real ou Inteiro

desenvolvida, considerando-se as seguintes igualdades:

$$z = x^y$$

$$\ln(z) = y \ln x$$

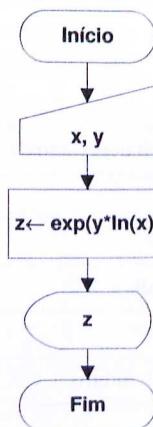
$$z = e^{y \ln x}$$

Assim, a expressão para calcular x^y , utilizando as funções da Tabela 3.9, é $\exp(y * \ln(x))$. Entretanto, deve-se prestar atenção para valores negativos na base, ou seja, para a variável x . Um fluxograma bem simples para calcular o valor de x e y fornecidos está descrito na Figura 3.25.

3.9.2 Funções e procedimentos para as cadeias de caracteres

As cadeias de caracteres também podem ser manipuladas. Convencionam-se as operações de cadeias de caracteres segundo a Tabela 3.10.

Por exemplo, um algoritmo que lê o nome de um estudante e então exibe a mensagem de que ele será aprovado em computação, ficaria assim representado pela Figura 3.26.

**Figura 3.25** Exemplo do uso de funções matemáticas.**Tabela 3.10** Funções e procedimentos para as cadeias de caracteres.

Operação	Significado	Exemplos
<i>length(s)</i>	Fornece como resultado o número de caracteres que compõe uma cadeia <i>S</i> .	$N \leftarrow \text{length}(\text{'Olá'})$ O valor de <i>N</i> é 3.
<i>concat(<i>S</i>₁, <i>S</i>₂)</i>	Une duas cadeias, a 2^{a} (<i>S</i> ₂) no final da 1^{a} (<i>S</i> ₁), formando uma nova cadeia.	$S \leftarrow \text{concat}(\text{'Bom'}, \text{'Dia'})$ O valor de <i>S</i> é 'BomDia'.
<i>copy(<i>S</i>, <i>ini</i>, <i>num</i>)</i>	Retorna (copia) a uma nova cadeia de caracteres com os <i>num</i> elementos da cadeia <i>S</i> a partir da posição <i>ini</i> .	$S \leftarrow \text{copy}(\text{'Turbo Pascal'}, 7, 6)$ O valor de <i>S</i> é 'Pascal'.
<i>insert(<i>S</i>₁, <i>S</i>₂, <i>ini</i>)</i>	Insere uma nova cadeia (<i>S</i> ₁) na posição <i>ini</i> de <i>S</i> ₂ , deslocando para a esquerda o resto da cadeia original (<i>S</i> ₂).	$S \leftarrow \text{"Turbo 7.0"}$ $\text{insert}(\text{'Pascal'}, S, 7)$ O valor de <i>S</i> é 'Turbo Pascal7.0'
<i>pos(<i>S</i>₁, <i>S</i>₂)</i>	Fornece como resultado a posição, na qual a cadeia <i>S</i> ₁ começa dentro da cadeia <i>S</i> ₂ . Se a cadeia <i>S</i> ₁ não existir em <i>S</i> ₂ , o resultado será zero.	$I \leftarrow \text{pos}(\text{'Pascal'}, \text{'Turbo Pascal'})$ Aqui o valor de <i>I</i> é 7, mas $I \leftarrow \text{pos}(\text{'pascal'}, \text{'Turbo Pascal'})$ O valor de <i>I</i> é 0. Existe diferença entre maiúsculas e minúsculas nessa função.

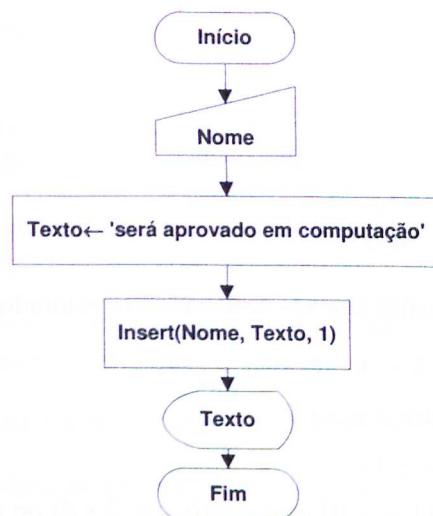


Figura 3.26 Exemplo do uso de operações com cadeias de caracteres.

Se o valor da variável *Nome* for ‘Douglas’, o valor da variável *Texto* a ser apresentado será ‘Douglas será aprovado em Computação’.

3.10 Exercícios

3.1. Elabore um fluxograma que calcule quantas notas de 50, 10 e 1 são necessárias para se pagar uma conta cujo valor é fornecido.

3.2. Reescreva as expressões abaixo, utilizando as convenções adotadas para os fluxogramas:

a) $C = 3 \cdot \frac{(5 + 3)}{2 \cdot 3} - 4 + 7$

b) $q = \sqrt{\frac{g \cdot P_m^5}{L \cdot V_m}}$

c) $\eta = 1 - \frac{\omega}{\mu} \cdot \frac{tg \epsilon}{sen(\beta + \epsilon)}$

d) $k_0 = \frac{3 \cdot 10^{-7}}{\sqrt[5]{Re}}$

Todos os direitos reservados ao autor
pela Lei 9.609/1998.

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

e) $D = \frac{3}{2 + \frac{5}{2 + \frac{1}{3}}}$

f) $L_p = \frac{(d - t)^4 \cdot \pi}{32}$

g) $H_{vl} = T_A \cdot \frac{\omega^2}{2 \cdot g} \cdot \frac{L}{t} \cdot \frac{\operatorname{tg} \varepsilon}{\operatorname{sen} \beta}$

3.3. ☀ Resolva as expressões abaixo, destacando o resultado final:

a) $A \leftarrow (18/3/2 - 1) * 5 - 4 - (2 + 3 + 5)/2$

b) $B \leftarrow 26/6/2 - 127 \operatorname{div} 7 \operatorname{mod} 5$

c) $C \leftarrow 7 \operatorname{mod} 4 - 8/(3 + 1)$

d) $D \leftarrow (2 >= 5) \operatorname{and} (1 <> 0) \operatorname{and} \operatorname{not}(6 <= 2 * 3) \operatorname{or} (10 <> 10)$

e) $E \leftarrow (5 <> 2) \operatorname{or} \operatorname{not}(7 > 4) \operatorname{and} (4 <= pi)$

3.4. ☀ Considerando as seguintes atribuições, $R \leftarrow 2$, $S \leftarrow 5$, $T \leftarrow -1$, $X \leftarrow 3$, $Y \leftarrow 1$ e $Z \leftarrow 0$, resolver as expressões abaixo:

a) $A \leftarrow (R >= 5) \operatorname{or} (T > Z) \operatorname{and} (X - Y + R > 3 * Z)$

b) $B \leftarrow (\operatorname{abs}(T) + 3 >= 4) \operatorname{and} \operatorname{not}(3 * R/2 < S * 3)$

c) $C \leftarrow (X = 2) \operatorname{or} (Y = 1) \operatorname{and} ((Z > 0) \operatorname{or} (R > 3)) \operatorname{and} (S < 10)$

d) $D \leftarrow (R <> S) \operatorname{or} \operatorname{not}(\operatorname{sqrt}(R) < \operatorname{sqrt}(X)) \operatorname{and} (4327 * X * S * Z = 0)$

3.5. ☀ Elabore um fluxograma que calcule o alcance de um projétil, dada a velocidade inicial v_0 e o ângulo θ entre o cano do canhão e o solo. A fórmula a ser utilizada é:

$$S = \frac{v_0^2 \sin(2\theta)}{g}$$

3.6. ☀ Elabore um fluxograma que calcule a área de um triângulo pela fórmula de Hierão:

$$K = \sqrt{s(s - a)(s - b)(s - c)}$$

em que K é a área do triângulo, s o semiperímetro e a , b e c são os lados do triângulo.

3.7. ☀ Elabore um fluxograma que permita a entrada de um número inteiro e diga se ele é par ou ímpar.

- 3.8.** ☀ Elabore um fluxograma que leia dois números (x e y) e escreva como resultado o maior entre eles.
- 3.9.** ☀ ☺ Elabore um fluxograma que permita a entrada de dois valores, x e y , troque seus valores entre si e então exiba os novos resultados.
- 3.10.** ☀ O mesmo do Exercício 3.9, com as seguintes exigências: só podem ser utilizadas duas variáveis e operações de adição e subtração.
- 3.11.** ☀ ☺ Elabore um fluxograma que permita a entrada de n (lido pelo teclado) valores reais e apresente como resultado o maior entre esses valores.
- 3.12.** ☀ Idem ao Exercício 3.8, só que escreva o menor deles.
- 3.13.** ☀ Elabore um fluxograma que calcule e exiba a média de dois números digitados.
- 3.14.** ☀ ☺ Elabore um fluxograma que calcule e exiba a média de 500 números fornecidos pelo usuário.
- 3.15.** ☀ Elabore um fluxograma que calcule e exiba a soma dos números pares contidos entre zero e um número par fornecido via teclado.
- 3.16.** ☀ Elabore um fluxograma que calcule e exiba a soma dos números ímpares contidos entre zero e um número ímpar fornecido via teclado.
- 3.17.** ☀ A contribuição para o INSS (interessante para estrutura condicional por ser progressivo) é calculada a partir da tabela a seguir:

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\$)	Aliquota 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

Elabore um algoritmo que, para uma entrada do salário bruto, calcule a contribuição ao INSS e o salário líquido restante.

3.18. ☀ O desconto do IRRF (Imposto de Renda Retido na Fonte), também denominado “Mordida do Leão”, é calculado sobre o salário líquido após a dedução da contribuição ao INSS, de acordo com a seguinte tabela:

Base de Cálculo em R\$	Alíquota %	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

$$\text{Líquido} = \text{Bruto} - \text{INSS} - \text{IR}$$

$$\text{Líquido} = \text{Bruto} - \text{INSS} - (\text{base} * \text{alíquota} - \text{parcela})$$

Elabore um fluxograma que, para uma entrada do salário bruto e após a dedução da contribuição (veja o Exercício 3.17), calcule o desconto do IRRF.

3.19. ☀ Elabore um fluxograma que leia as quatro notas de prova (P_1, P_2, P_3 e P_4) e quatro notas de trabalho (T_1, T_2, T_3 e T_4) e posteriormente exiba a mensagem ‘Aprovado’ ou ‘Não aprovado’ dependendo dos valores obtidos, conforme as regras de cálculo definidas a seguir:

- Média de provas: $MP = \frac{P_1 + P_2 + P_3 + P_4}{4}$
- Média de trabalhos: $MT = \frac{T_1 + T_2 + T_3 + T_4}{4}$
- Média final: $MF = 0,8MP + 0,2MT$
- Situação:
 - se $MF \geq 6,0 \Rightarrow$ aprovado;
 - se $MF < 6,0 \Rightarrow$ não aprovado.

3.20. ☀ Elabore um fluxograma que transforme uma temperatura fornecida em °C para a correspondente em °F. A fórmula de conversão de °F para °C é:

$$C = \frac{5}{9}(\text{°F} - 32)$$

3.21. ☀ Elabore um fluxograma que permita a entrada de dois valores inteiros e faça uma contagem decrescente desde o maior deles até o menor.

3.22. ☀ Elabore um fluxograma que permita a entrada de três valores e faça a contagem desde o primeiro deles até o segundo com passo dado pelo terceiro.

3.23. ☀ Um número inteiro é considerado triangular se este for o produto de três números inteiros consecutivos, como, por exemplo $120 = 4 \times 5 \times 6$. Elabore um fluxograma que, após ler um número n , verifique se o mesmo é ou não triangular.

3.24. ☀ Elabore um fluxograma que leia um valor n inteiro e verifique se este é ou não primo (número primo é divisível por um e por ele mesmo).

3.25. ☀ Um número palíndromo é aquele que se lido da esquerda para a direita e da direita para a esquerda possui o mesmo valor (por exemplo: 34543). Elabore um fluxograma que leia um número n , inteiro, e verifique se ele é um palíndromo.

3.26. ☀ Elabore um fluxograma que receba três valores digitados A , B e C , informando se estes podem ser os lados de um triângulo. O ABC é triângulo se $A < B + C$ e $B < A + C$ e $C < A + B$.

3.27. ☀ Elabore um fluxograma que permita a entrada de 30 valores e mostre a soma de seus inversos. Observação: o inverso de x é $1/x$.

3.28. ☀ Elabore um fluxograma que permita a entrada de n valores e mostre a soma de seus quadrados.

3.29. ☀ Elabore um fluxograma que permita a entrada de dez valores e calcule o produto de todos eles.

3.30. ☀ Elabore um fluxograma que represente o algoritmo para calcular a soma dos primeiros 40 termos da sequência definida a seguir, com o valor de A fornecido via teclado:

$$\frac{7 \cdot A}{3}, \frac{7 \cdot A}{6}, \frac{7 \cdot A}{12}, \frac{7 \cdot A}{24}, \frac{7 \cdot A}{48} \text{ ou}$$

3.31. ☀ Elabore um fluxograma que represente o algoritmo para calcular a soma dos primeiros N termos da sequência definida a seguir, com N fornecido via teclado:

$$\frac{1}{2}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}, \frac{1}{10}, \dots$$

3.32. ☀ Elabore um fluxograma que represente o algoritmo para calcular a soma dos primeiros N termos da sequência definida a seguir, com N fornecido via teclado:

$$S = 1 + 2 + 3 + 4 + 5 + \dots + N$$

3.33. ☀ O número π pode ser calculado por meio da série infinita:

$$\pi = 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \dots\right)$$

Elabore um fluxograma que calcule e exiba o valor do número π , utilizando a série anterior, até que o valor absoluto da diferença entre o número calculado em uma iteração e o da anterior seja menor ou igual a 0.00000000005.

3.34. ☀ Elabore um fluxograma que, dados dois números complexos $c1$ e $c2$, calcule as seguintes operações: soma, subtração e multiplicação.

Lembrando: um número complexo possui duas partes, uma real (re) e uma imaginária (im), representado genericamente como $c = re + j \cdot im$.

3.35. ☀ O número 3025 possui uma característica interessante, sendo a seguinte: $30 + 25 = 55$ e $55^2 = 3025$. Elaborar um fluxograma que verifique se um número inteiro de quatro algarismos (digitado) tem essa propriedade ou não.

3.36. ☀ Elabore um fluxograma que represente o algoritmo do cálculo dos 50 primeiros termos da série apresentada a seguir, sendo X um valor digitado:

$$\frac{2 \cdot 3}{X + 3}, \frac{2 \cdot 5}{X + 5}, \frac{2 \cdot 7}{X + 7}, \frac{2 \cdot 9}{X + 9}, \dots$$

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.

3.37. ☀ Qual o resultado exibido pelo fluxograma da Figura 3.27, se o dado de entrada digitado for sua idade?

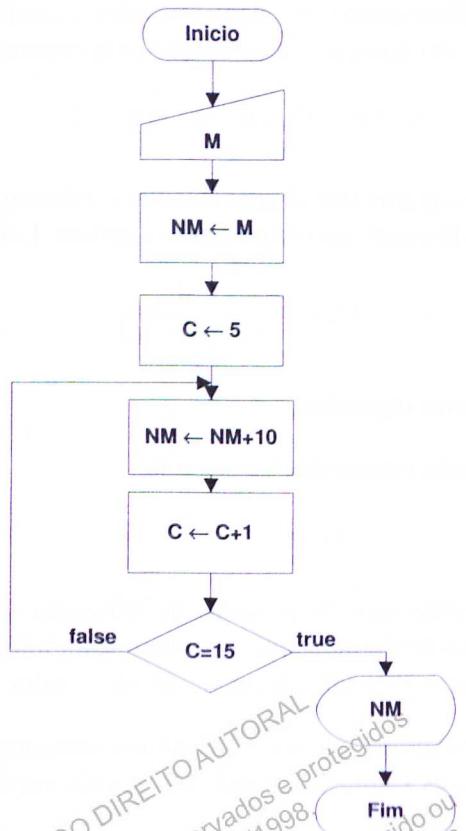


Figura 3.27 Fluxograma para o Exercício 3.37.

3.38. ☀ Elabore um fluxograma que calcule e exiba a tensão S de uma barra cilíndrica de diâmetro D submetida a uma carga Q . Os valores de D e Q devem ser digitados via teclado. Utilize a fórmula $S = \frac{4 \cdot Q}{\pi \cdot D^2 \cdot n}$, considerando as seguintes condições:

- se $D > 100$, então $n = 2$;
- se $D < 50$, então $n = 6$;
- caso contrário, $n = 4$.

3.39. ☀ Altere o fluxograma da Figura 3.25 para calcular x^y , para quaisquer valores de x e y (incluindo 0).

3.40. ☀ ☈ Elabore um fluxograma que, dado um valor n inteiro, calculará seu fatorial. Lembrando, o fatorial de um número n é calculado pela expressão:

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

3.41. ☀ ☈ Elabore um fluxograma que deverá calcular o número de maneiras de se escorrer r dentre n objetos diferentes, não importando a ordem. Lembrando:

$$C_{n,r} = \frac{n!}{r! \cdot (n - r)!}$$

em que n e r são valores digitados.

3.42. ☀ A fórmula de juros compostos é a seguinte:

$$V_f = (1 + i)^N \cdot V_i$$

V_f é o valor final obtido após N períodos de aplicação com juros i . V_i é o valor inicial, à vista. Elabore um fluxograma que, após ler o valor inicial, o número de períodos (que normalmente são meses) e a taxa de juros, calcule o valor final desejado.

3.43. ☀ Escreva um fluxograma que leia três valores quaisquer para as variáveis A , B e C . A seguir, ordene esses valores, exibindo as mesmas variáveis A , B e C , agora já ordenadas.

3.44. ☀ Um número da série de Fibônnaci é gerado à partir da soma de dois valores imediatamente anteriores. Convenciona-se que o primeiro número, f_0 , é 0, e o segundo, f_1 , é 1. A partir desses valores, é possível calcular o n -ésimo elemento da série assim (para $n > 2$):

$$f_n = f_{n-1} + f_{n-2}$$

Elabore um fluxograma que, a partir de um valor n lido ($n \geq 0$), calcule f_n .

3.45. ☀ Para o fluxograma apresentado pela Figura 3.28, responda:

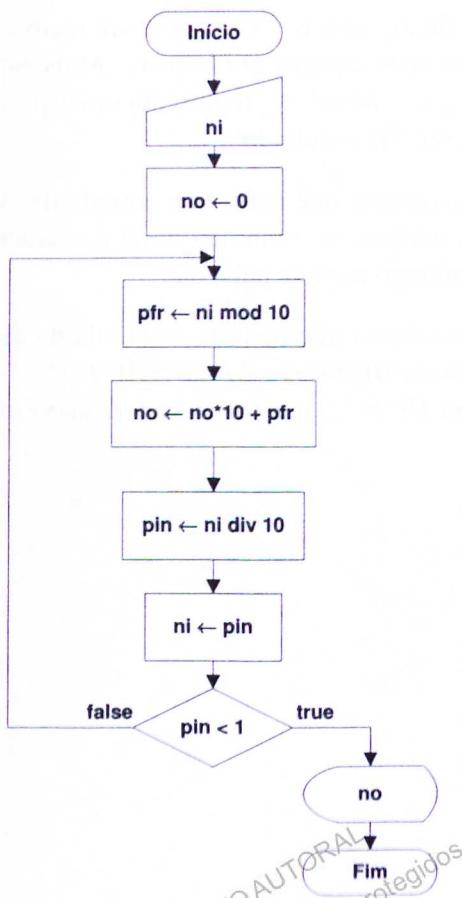


Figura 3.28 Fluxograma para o Exercício 3.45.

- a) determine o valor de no para $ni = 8730$;
 b) determine o valor de no para $ni = 1234$;
 c) o que faz esse fluxograma?

3.46. ☀ Escreva um fluxograma que leia o nome do usuário e o cumprimento. Por exemplo, se você se chamar Aníbal, a resposta a ser exibida será: ‘Olá Aníbal, meu nome é Chuck. Você quer brincar comigo?’

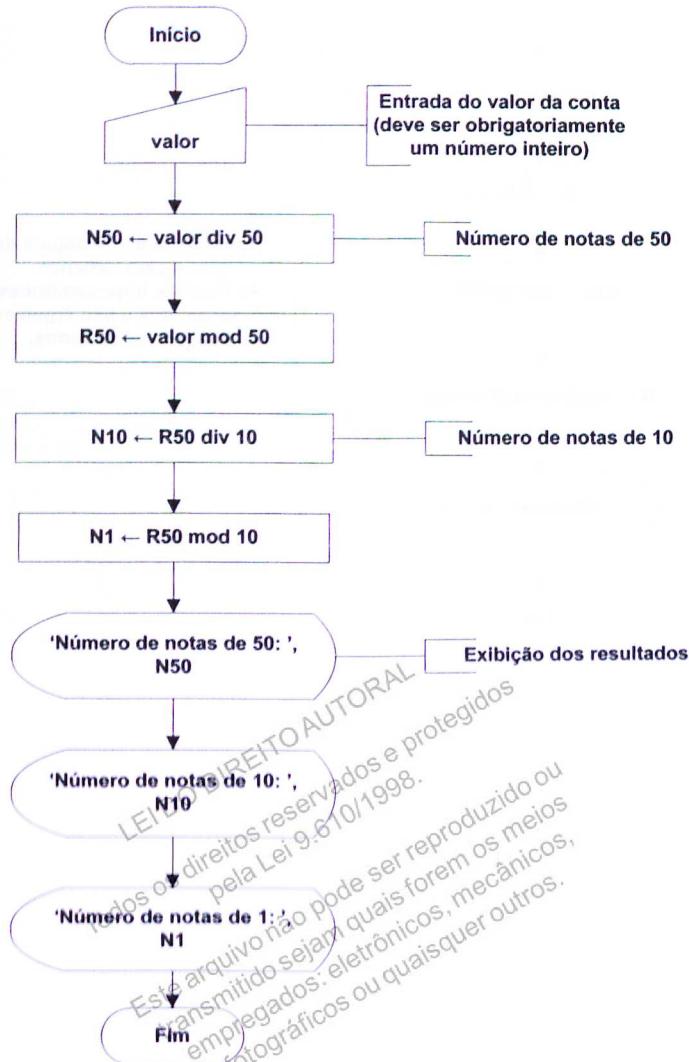
3.47. ☀ Elabore um fluxograma que permita a entrada de um número inteiro entre 1 e 999 e escreva seu valor por extenso.

- 3.48.** Elabore um fluxograma que compacte um texto contido em uma cadeia de caracteres, eliminando os seus espaços em branco. Mais especificamente, se o texto fornecido for ‘O amor é lindo!’ (representa um espaço em branco), o resultado a ser apresentado deverá ser ‘Oamorélindo!’.
- 3.49.** Elabore um fluxograma que permita a entrada de N cadeias de caracteres e então exiba as seguintes estatísticas: o número total de caracteres digitados (incluindo espaços em branco) e o número total de palavras.
- 3.50.** Elabore um fluxograma que permita a entrada de duas cadeias de caracteres, respectivamente, às variáveis *BUSCA* e *MENSAGEM*, e então exiba todas as posições da cadeia contida em *BUSCA* que foram localizadas em *MENSAGEM*.

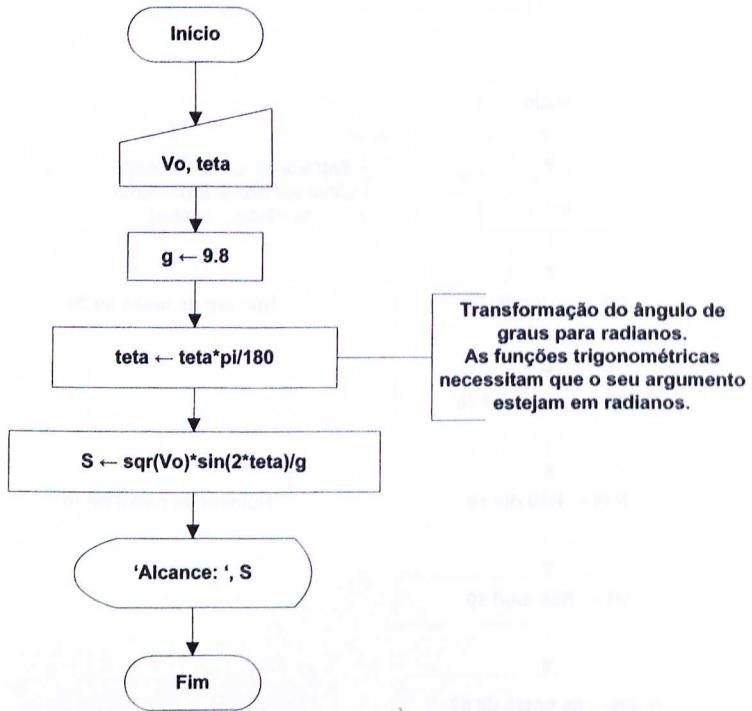
LEI DO DIREITO AUTORAL
Todos os direitos reservados e protegidos
pela Lei 9.610/1998.
Este arquivo não pode ser reproduzido ou
transmitido sejam quais forem os meios
empregados: eletrônicos, mecânicos,
fotográficos ou quaisquer outros.

3.11 Exercícios resolvidos

3.1.



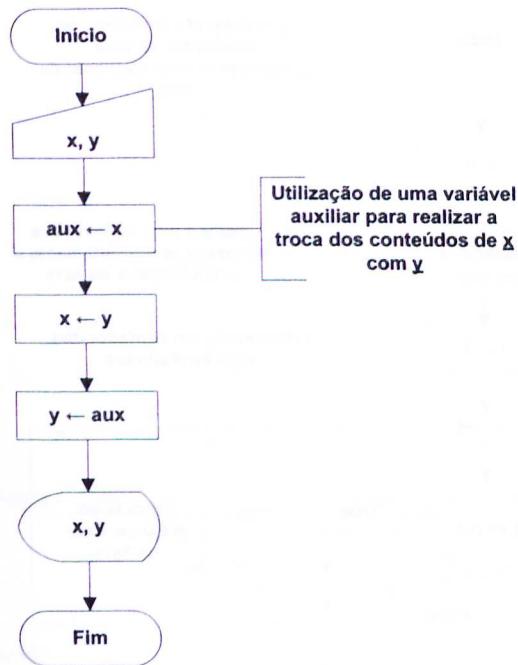
3.5. ☀



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.

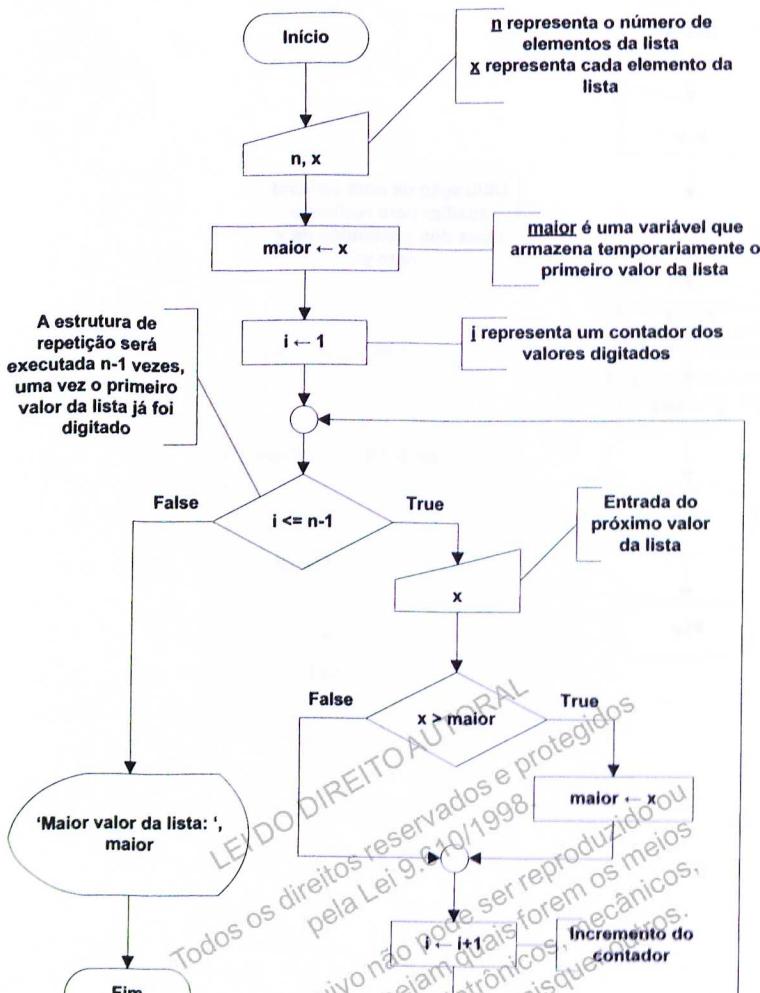
3.9. ☼



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.

3.11.

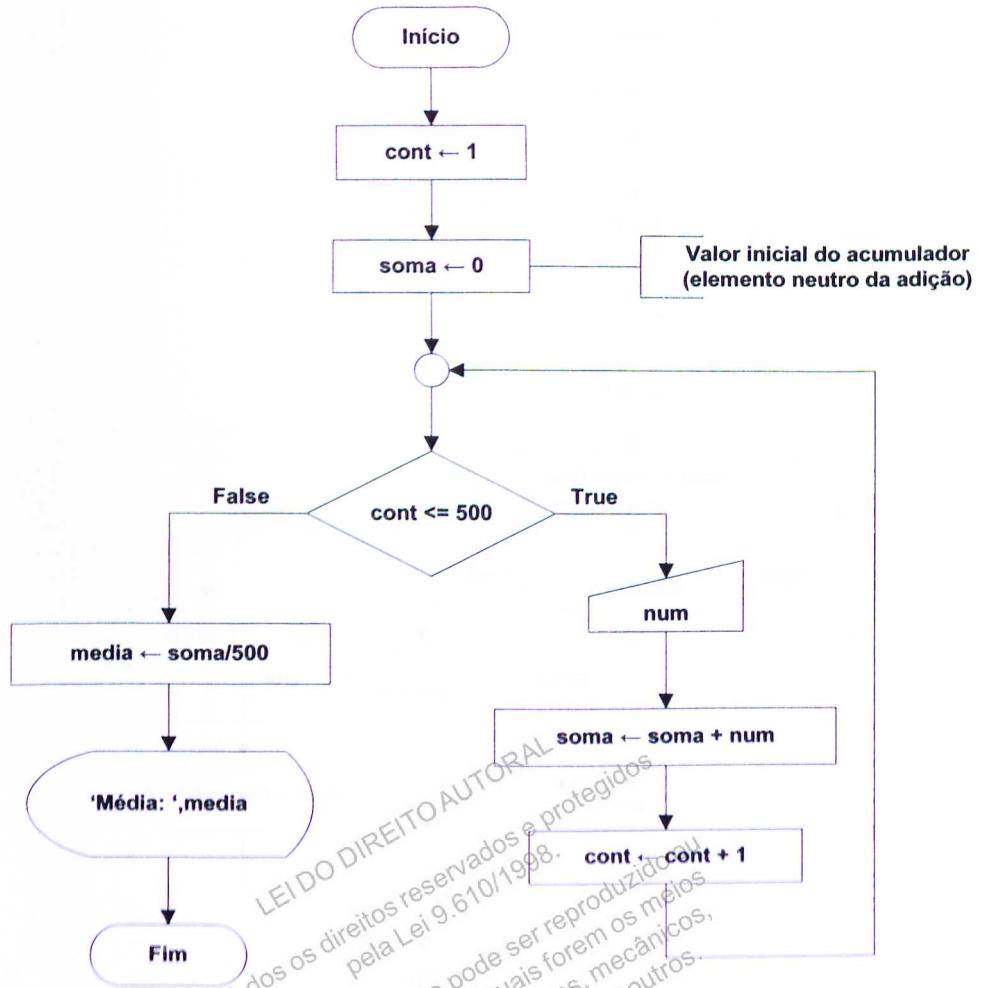


A solução apresentada permite encontrar o maior valor de uma lista contendo tanto números positivos quanto negativos.

Como ficaria a solução para encontrar o menor valor de uma lista?

O que aconteceria se a variável maior tivesse como valor inicial 0?

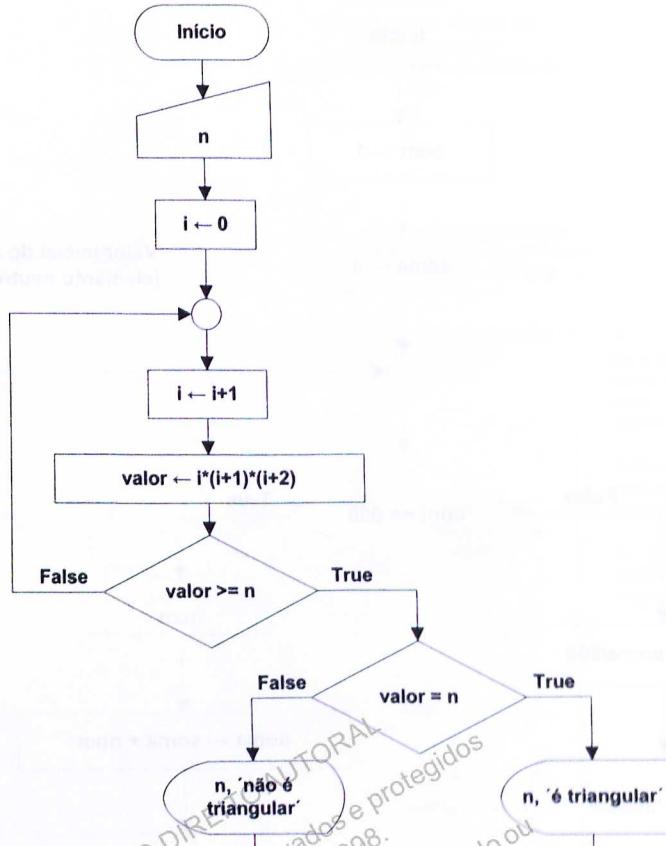
3.14. ☀



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

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

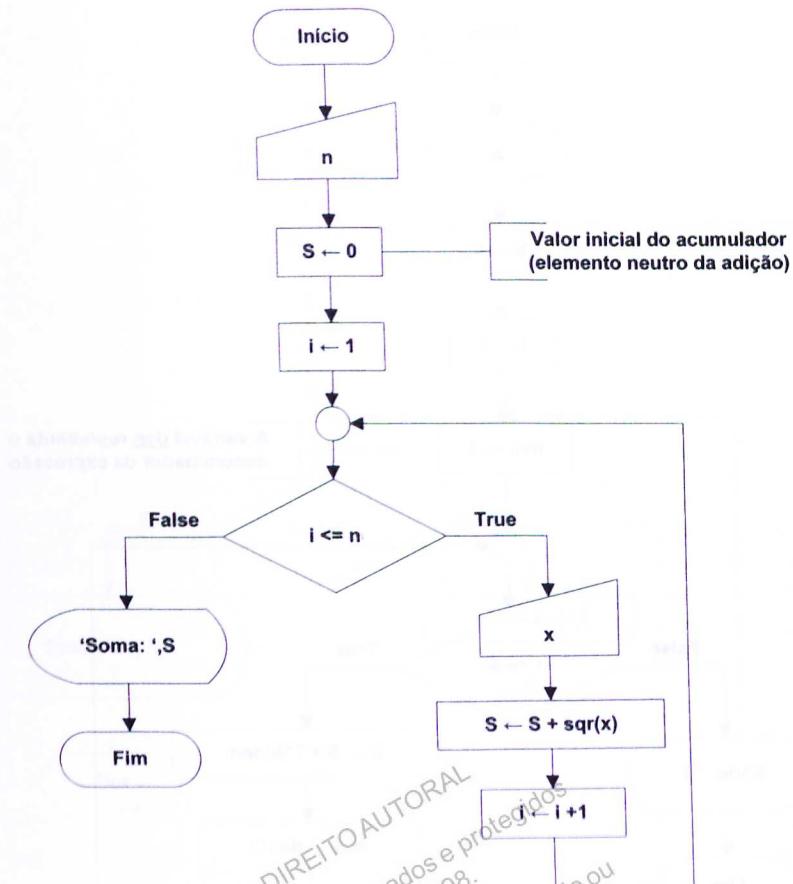
3.23. ☀



LEI DO DIRETORIAL
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.

Fim

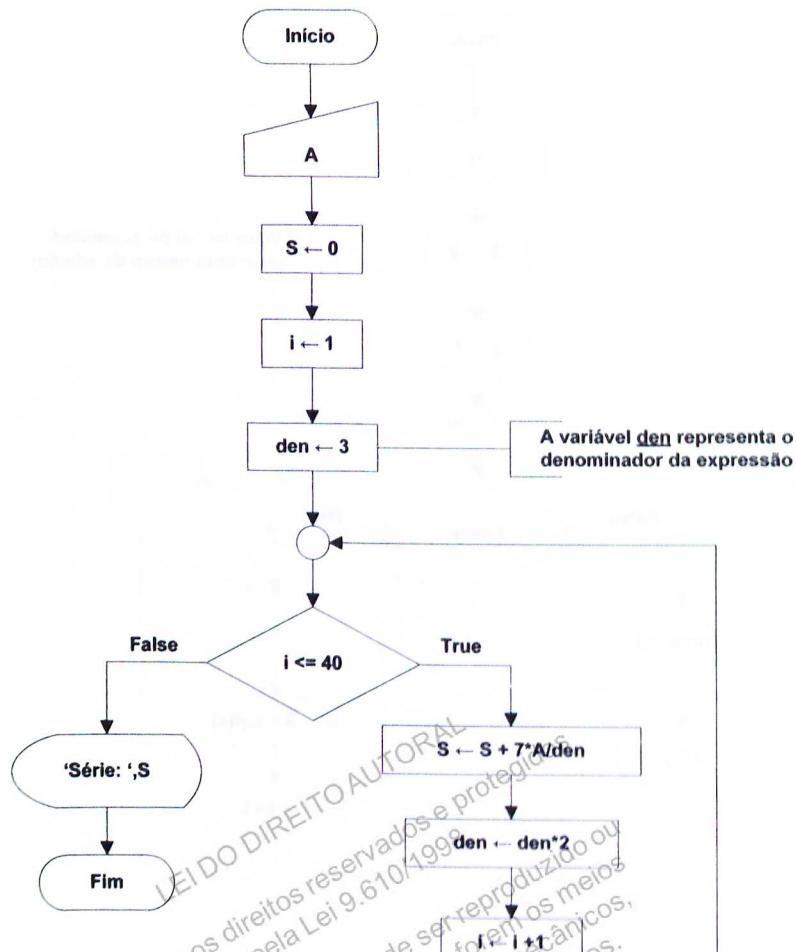
3.28. ☀



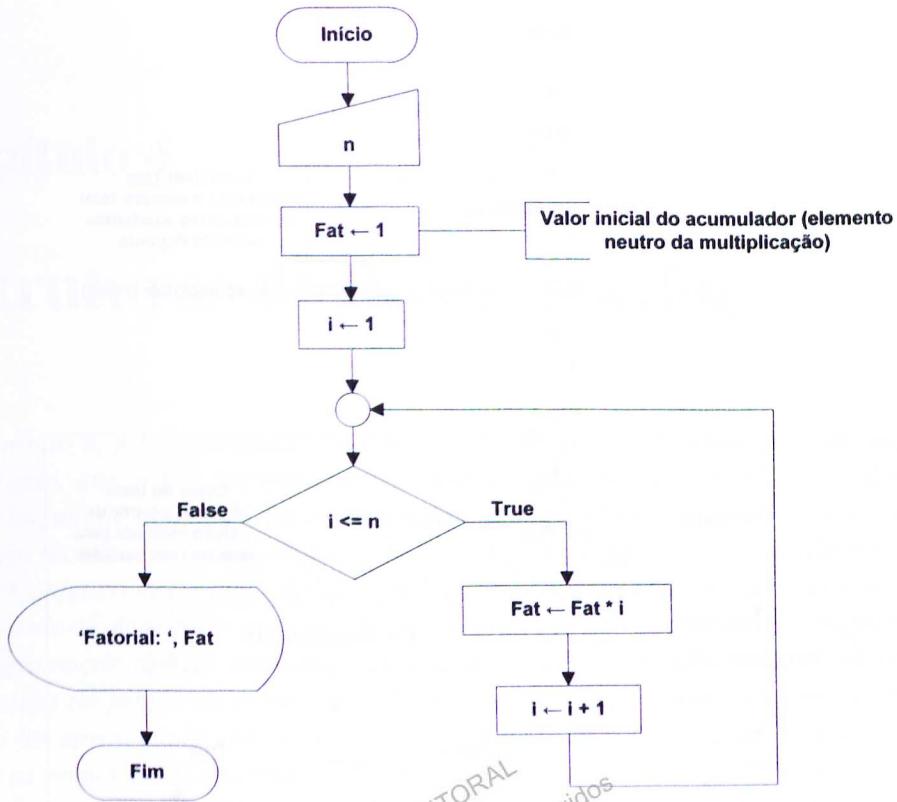
LEI DO DIREITO AUTORAL
Todos os direitos reservados e protegidos
pela Lei 9.610/1998.

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

3.30. ☀



3.40. ☀



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.

3.48. ☀

