A estrutura de repetição `for`

A estrutura de repetição **`for`** trata todos os detalhes da repetição controlada por contador. Para ilustrar o poder do **`for`**, iremos reescrever o programa da Fig. 2.16. O resultado é mostrado na Fig. 2.17. O programa opera como segue.

Quando a estrutura **`for`** começa a ser executada, a variável de controle **`counter`** é declarada e inicializada com 1. Então, é verificada a condição de continuação do laço, **`counter <= 10`**. Como o valor inicial de **`counter`** é 1, a condição é satisfeita; assim, o comando do corpo imprime o valor de **`counter`**, i.e., 1. A variável de controle **`counter`** é então incrementada na expressão **`counter++`** e o laço começa novamente com o teste de continuação do laço. Como a variável de controle agora é igual a 2, o valor final não é excedido e assim o programa executa novamente o comando do corpo. Este processo continua até que a variável de controle **`counter`** seja incrementada para 11 – isto faz com que o teste de continuação do laço não seja satisfeito e a repetição termine. O programa continua, executando o primeiro comando depois da estrutura **`for`** (neste caso, o comando **`return`** no fim do programa).

```
1  // Fig. 2.16: fig02_16.cpp
2  // Repetição controlada por contador
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  int main()
9  {
10    int counter = 1;              // inicialização
11
12    while ( counter <= 10 ) {     // condição de repetição
13       cout << counter << endl;
14       ++counter;                 // incremento
15    }
16
17    return 0;
18 }
```

Fig. 2.16 Repetição controlada por contador.

```
1  // Fig. 2.17: fig02_17.cpp
2  // Repetição controlada por contador com a estrutura for
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  int main ( )
9  {
10    // inicialização, condição de repetição e incremento
11    // estão todas incluidas no cabeçalho da estrutura for.
12
13    for ( int counter = 1; counter <= 10; counter++ )
14       cout << counter << endl;
15
16    return 0;
17 }
```

Fig. 2.17 Repetição controlada por contador com a estrutura **`for`**.

Note que a Fig. 2.17 usa a condição de continuação de laço **`counter <= 10`**. Se o programador escrevesse, incorretamente, **`counter < 10`**, então o ciclo seria executado somente 9 vezes. Este é um erro de lógica comum, chamado *saindo do laço uma iteração antes (off-by-one errar).*

A Fig. 2.18 examina mais de perto a estrutura **for** da Fig. 2.17. Note que a estrutura **for** "faz tudo" – especifica cada um dos itens necessários para uma repetição controlada por contador com uma variável de controle. Se existir mais de um comando no corpo do **for**, são necessárias chaves para definir o corpo do laço.

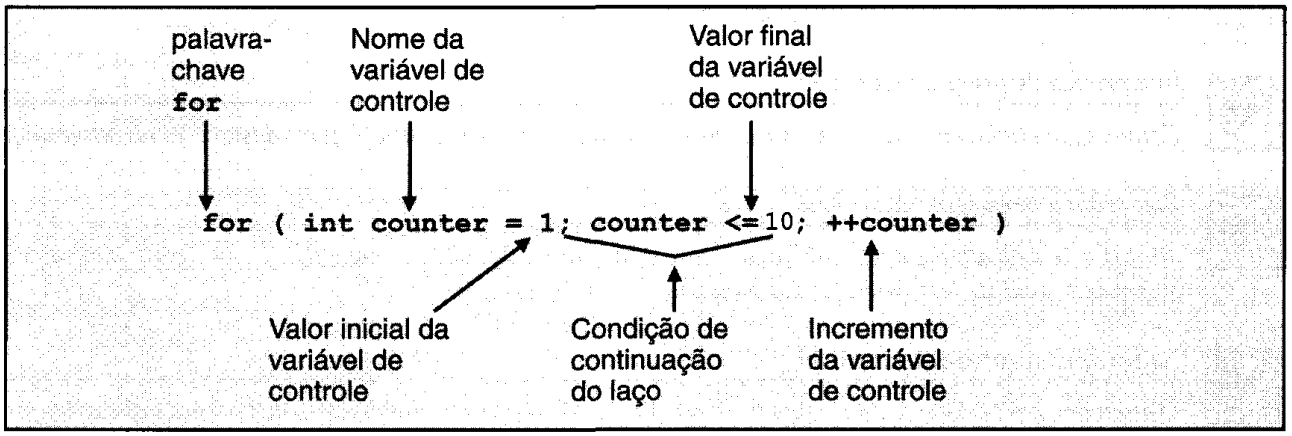

Fig. 2.18 Componentes de um cabeçalho **for** típico.

O fluxograma da estrutura **for** é muito semelhante ao da estrutura **while**. A Fig. 2.19 mostra o fluxograma do comando **for**

```
for ( int counter = 1; counter <= 10; counter++ )
   cout << counter << endl;
```

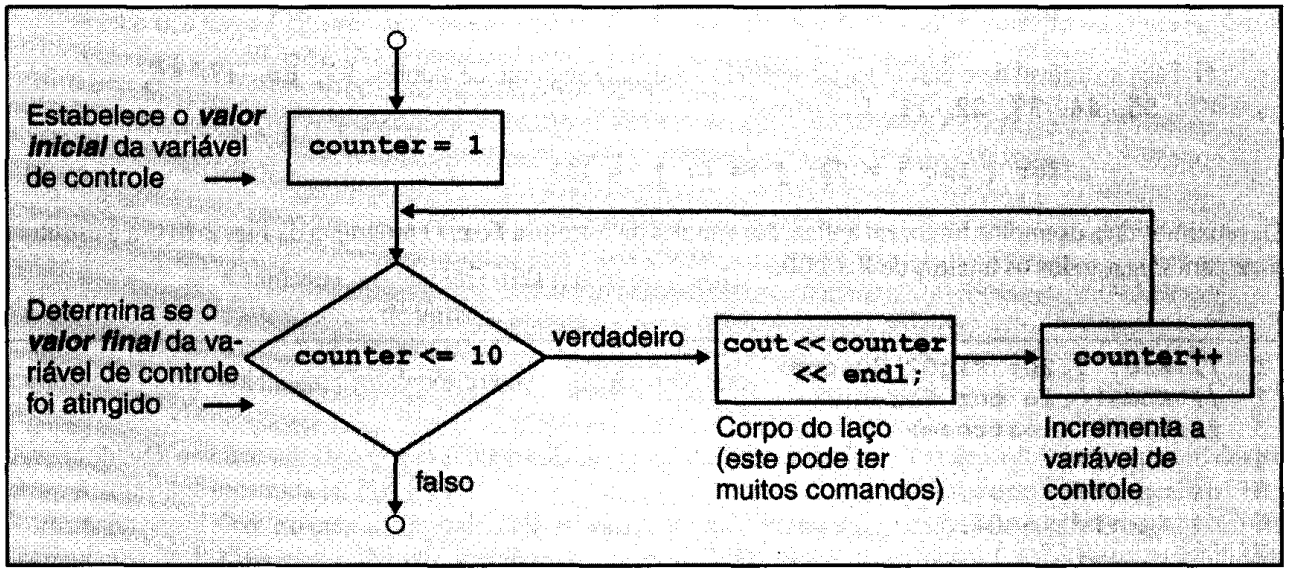

Fig. 2.19 Fluxograma de uma estrutura de repetição **for** típica.

O formato geral da estrutura **for** é

```
for ( initialization; loopContinuationTest; increment )
   statement
```

onde *initialization* inicializa a variável de controle do laço, *loopContinuationTest* é a condição de continuação do laço e *increment* incrementa a variável de controle. Na maioria dos casos, a estrutura **for** pode ser representada com uma estrutura **while** equivalente, como segue:

```
initialization;
while ( loopContinuationTest ) {
   statement
   increment;
}
```