\[
c = a;
\]
\[
\text{else}
\]
\[
c = b;
\]

Compared to the \texttt{if else} sequence, the conditional operator is more concise but, at first, less obvious. One difference between the two approaches is that the conditional operator produces an expression and hence a single value that can be assigned or be incorporated into a larger expression, as the program did when it assigned the value of the conditional expression to the variable \(c\). The conditional operator's concise form, unusual syntax, and overall weird appearance make it a great favorite among programmers who appreciate those qualities. One favorite trick for the reprehensible goal of concealing the purpose of code is to nest conditional expressions within one another, as the following mild example shows:

```cpp
const char x[2][20] = {"Jason ", "at your service\n"};
const char * y = "Quillstone ";

for (int i = 0; i < 3; i++)
    cout << ((i < 2)? !i ? x[i] : y : x[1]);
```

This is merely an obscure (but, by no means, maximally obscure) way to print the three strings in the following order:

Jason Quillstone at your service

In terms of readability, the conditional operator is best suited for simple relationships and simple expression values:

\[
x = (x > y) \ ? x : y;
\]

If the code becomes more involved, it probably can be expressed more clearly as an \texttt{if else} statement.

**The switch Statement**

Suppose you create a screen menu that asks the user to select one of five choices, for example, Cheap, Moderate, Expensive, Extravagant, and Excessive. You can extend an if