```cpp
cout << "The expression x > 3 has the value ";
cout << (x > 3) << "\n";
return 0;
}
```

**Compatibility Note**

Older implementations of C++ may require using `ios::boolalpha` instead of `ios_base::boolalpha` as the argument for `cout.setf()`. Yet older implementations might not recognize either form.

Here is the output:

The expression x = 100 has the value 100
Now x = 100
The expression x < 3 has the value 0
The expression x > 3 has the value 1
The expression x < 3 has the value false
The expression x > 3 has the value true

Normally, `cout` converts `bool` values to `int` before displaying them, but the `cout.setf(ios::boolalpha)` function call sets a flag that instructs `cout` to display the words `true` and `false` instead of 1 and 0.

**Remember**

A C++ expression is a value or a combination of values and operators, and every C++ expression has a value.

To evaluate the expression `x = 100`, C++ must assign the value 100 to `x`. When the very act of evaluating an expression changes the value of data in memory, we say the evaluation has a *side effect*. Thus, evaluating an assignment expression has the side effect of changing the assignee's value. You might think of assignment as the intended effect, but from the standpoint of how C++ is constructed, evaluating the expression is the primary effect. Not all expressions have side effects. For example, evaluating `x + 15`