such as in the test condition of a while loop. Furthermore, the bool value for the conversion is true if the last attempted read was successful and false otherwise. This means you can rewrite the while test to look like this:

```
while (cin)    // while input is successful
```

This is a bit more general than using !cin.fail() or !cin.eof(), for it detects other possible causes of failure, such as disk failure.

Finally, because the return value of cin.get(char) is cin, you can condense the loop to this format:

```
while (cin.get(ch)) // while input is successful
{
   ...               // do stuff
}
```

To evaluate the loop test, the program first has to execute the call to cin.get(ch), which, if successful, places a value into ch. Then, the program obtains the return value from the function call, which is cin. Then, it applies the bool conversion to cin, which yields true if input worked, false otherwise. The three guidelines (identifying the termination condition, initializing the condition, and updating the condition) all are compressed into one loop test condition.

**Yet Another cin.get()**

The more nostalgic of the C users among you might yearn for C's character I/O functions, getchar() and putchar(). They still are available if you want them. Just use the stdio.h header file as you would in C (or use the more current cstdio). Or, you can use member functions from the istream and ostream classes that work in much the same way. We look at that approach now.

**Compatibility Note**

Some older implementations don't support the cin.get() member function (no arguments) discussed here.