Data Loops

Let's look at how the *filter* program works. C++ input streams read a single character using the **member function** `cin.get()`. To **read successive characters**, until all of the data has been processed, use a **data** (or `eof-controlled`) loop. (`eof` is shorthand for **end-of-file)**.

```
while there is still data to process
    read a data item
    process the data item
```

Translate this into C++ by **using streams as conditions**, as shown here:

```
char ch;
while (cin.get(ch))
{
    cout.put(ch);   // print the output
}
```

The expression `cin.get(ch)` does two things.

1. **It reads** the next character from the stream into the `char` variable `ch` (which is passed to the function by reference). Whitespace **is not** skipped.
2. **It returns the input stream** (in this case `cin`) after reading the variable so you can determine whether the I/O operation succeeded.

The `cin` object has a member function, named `fail()`, which indicates whether the last operation succeeded. `fail()` **is implicitly called when a stream is used as a condition.** In a condition, the stream is interpreted as `true` if it is still good, and as false on failure.

When reading characters using `cin.get()`, input fails **only if there are no characters left** in the stream. The effect of the **basic data loop** is to execute the body of the `while` loop once for **each** character until the stream reaches what is known as **end of the file**.

For **output streams**, the `put()` member function takes a `char` value as its argument and writes that character to the stream.