The Flag-controlled Pattern

A third way to implement the read-until-sentinel pattern is to use a flag-controlled loop, where you introduce an additional *Boolean* variable just before the loop starts and set it to **`false`**. Inside the loop you read a data value and check the sentinel, just as in the loop-and-a-half.

Instead of a **`break`** statement, set your flag variable to **`true`** when the sentinel is read. Otherwise, you process that data value as normal:

```
Set finished to false   // Boolean control flag
while not finished
  read the value
  if value is the sentinel then
    set finished to true
  else
    process the variable
```

As we've done with the other two methods, here is the same program implemented as a **flag-controlled** sentinel loop:

```
bool finished{false};       // control flag
while (! finished)
{
    cout << "> ";               // Prompt and read item
    cin >> value;
    if (value == 0)
    {
        finished = true;
    }
    else
    {
        total += value;
    }
}
```

Which of these thre versions **should** you use? Eric Roberts, a professor at Stanford for many years, suggests that empirical studies demonstrate that students are more **likely to write correct programs if they use the loop-and-a-half version** than if they are forced to use some other strategy. If you're interested, follow the link to read Roberts' paper.