sum += number;
cout << "Five exquisite choices indeed! ";

The compiler ignores indentation, so only the first statement would be in the loop. Thus, the loop would print the five prompts and do nothing more. After the loop completed, the program would move to the following lines, reading and summing just one number.

Compound statements have another interesting property. If you define a new variable inside a block, the variable persists only as long as the program is executing statements within the block. When execution leaves the block, the variable is deallocated. That means the variable is known only within the block:

```
#include <iostream>
using namespace std;
int main()
{
    int x = 20;
    {
        // block starts
        int y = 100;
        cout << x << "\n";   // ok
        cout << y << "\n";   // ok
    }  // block ends
    cout << x << "\n";  // ok
    cout << y << "\n";           // invalid, won't compile
    return 0;
}
```

Note that a variable defined in an outer block still is defined in the inner block.

What happens if you declare a variable in a block that has the same name as one outside the block? The new variable hides the old one from its point of appearance until the end of the block. Then, the old one becomes visible again.

```
int main()
{
    int x = 20;  // original x
    {
        // block starts
        cout << x << "\n";  // use original x
```