The default constructor guarantees that all stacks are created empty. The code for `pop()` and `push()` guarantees that the top of the stack is managed properly. Guarantees like this are one of the things that make object-oriented programming more reliable. Suppose, instead, you create a separate array to represent the stack and an independent variable to represent the index of the top. Then, it is your responsibility to get the code right each time you create a new stack. Without the protection that private data offers, there's always the possibility of making some program blunder that alters data unintentionally.

Let's test this stack. **Listing 10.12** models the life of a clerk who processes purchase orders from the top of his in-basket, using the *LIFO (last-in, first-out)* approach of a stack.

**Listing 10.12 stacker.cpp**

```
// stacker.cpp -- test Stack class
#include <iostream>
using namespace std;
#include <cctype> // or ctype.h
#include "stack.h"
int main()
{
    Stack st; // create an empty stack
    char c;
    unsigned long po;
    cout << "Please enter A to add a purchase order,\n";
    << "P to process a PO, or Q to quit.\n";
    while (cin >> c && toupper != 'Q')
    {
        while (cin.get() != '\n')
            continue;
        if (isalpha)
        {
            cout << 'a';
            continue;
        }
        switch
        {
```