```
done:
if (d < n)
  printf("%d is divisible by %d\n", n, d);
else
  printf("%d is prime\n", n);
```

Q&A

exit function ➤9.5

The goto statement, a staple of older programming languages, is rarely needed in everyday C programming. The break, continue, and return statements—which are essentially restricted goto statements—and the exit function are sufficient to handle most situations that might require a goto in other languages.

Nonetheless, the goto statement can be helpful once in a while. Consider the problem of exiting a loop from within a switch statement. As we saw earlier, the break statement doesn't quite have the desired effect: it exits from the switch, but not from the loop. A goto statement solves the problem:

```
while (...) {
  switch (...) {
    ...
    goto loop_done;    /* break won't work here */
    ...
  }
}
loop_done: ...
```

The goto statement is also useful for exiting from nested loops.

PROGRAM

Balancing a Checkbook

Many simple interactive programs are menu-based: they present the user with a list of commands to choose from. Once the user has selected a command. the program performs the desired action, then prompts the user for another command. This process continues until the user selects an "exit" or "quit" command.

The heart of such a program will obviously be a loop. Inside the loop will be statements that prompt the user for a command, read the command, then decide what action to take:

```
for (;;) {
  prompt user to enter command;
  read command;
  execute command;
}
```

Executing the command will require a switch statement (or cascaded if statement):

```
for (;;) {
  prompt user to enter command;
  read command;
  switch (command) {
    case command1: perform operation1; break;
```