```
10
9
8
7
6
5
4
3
2
1
0
End of loop.
```

FIGURE 8-3
A for loop can decrement the counter variable.

The counter variable can do more than step by one. In the program in Figure 8-4, the counter variable is doubled each time the loop iterates. In the next exercise, you will see the effect of this for loop.

```
#include <iostream.h>

main()
{
 int i;   // counter variable
 for(i = 1; i <= 100; i = i + i)
   cout << i << '\n';
 return 0;
}
```

FIGURE 8-4
The counter variable in a for loop can be changed by any valid expression.

EXERCISE 8-3 INCREMENTING BY A STEP OTHER THAN ONE

1. Key the program from Figure 8-4 into a blank editor screen.
2. Save the source file as *DBLSTEP.CPP*. Can you predict the program's output?
3. Compile and run the program to see if your prediction was right.
4. Close the source file.

The for statement gives you a lot of flexibility. As you have already seen, the step expression can increment, decrement, or count in other ways. Some more examples of for statements are shown in Table 8-1.

for STATEMENT	COUNT PROGRESSION
`for (i = 2; i <= 10; i = i + 2)`	2, 4, 6, 8, 10
`for (i = 1; i < 10; i = i + 2)`	1, 3, 5, 7, 9
`for (i = 10; i <= 50; i = i + 10)`	10, 20, 30, 40, 50

TABLE 8-1

USING A STATEMENT BLOCK IN A for LOOP

If you need to include more than one statement in the loop, use braces to make a statement block below the for statement. If the first character following a for statement is an open brace ({), all of the statements between the braces are repeated. The same rule applies as with if structures.

EXERCISE 8-4 USING A STATEMENT BLOCK IN A for LOOP

1. Open *BACKWARD.CPP* and edit the source code to match the following:

```
#include <iostream.h>

main()
{
  int i;
  for(i = 10; i >= 0; i--)
    {
      cout << i << '\n';
      cout << "This is in the loop.\n";
    }
  return 0;
}
```

2. Compile and run the program to see that the phrase *This is in the loop* prints on every line. The second cout statement is now part of the loop because it is within the braces.
3. Close the source file without saving changes.

Extra for Experts

Earlier in this chapter, you learned that placing a semicolon at the end of the parentheses in a for statement will cause the loop to do nothing. While it is true that the statements that follow will not be iterated, there are cases where you might actually want to do just that.

Suppose you are given an integer and asked to calculate the largest three-digit number that can be produced by repeatedly doubling the given integer. For example, if the given integer is 12, repeatedly doubling the integer would produce the sequence 12, 24, 48, 96, 192, 384, 768, 1536. Therefore 768 is the largest three-digit number produced by repeatedly doubling the number 12.

The program below uses an empty loop to achieve the result outlined above.

```
#include <iostream.h>
main()
 {
  long i;
  cin >> i;
  for ( ; i <= 1000; i = i * 2);
  cout << i/2 << '\n';
  return 0;
 }
```

The first parameter of the for statement is left blank because `i` *is initialized by the user in the cin statement. Even though the loop is empty, the stepping of the counter variable continues and the value is available after the loop terminates. The value of* `i` *is 1000 or more when the loop ends. The cout statement then divides the counter by two to return it to the highest three-digit number.*