On the Net

In some for loops, the variable referenced in the parameters is only needed inside the for loop. To accommodate this problem, C++ allows you to declare and initialize variables in the initializing expression of a for loop. For example, in Exercise 8-2 the variable `i` *is used within the loop only. The program in Exercise 8-2 could be replaced with the source code below.*

```
#include <iostream.h>

main ()
{
  for(int i = 10; i >= 0; i--)
    cout << i << endl;
  cout << "End of the loop.\n";
  return 0;
}
```

To learn more about advanced features of for loops see http://www.ProgramCPP.com. See Topic 8.1.1.

SECTION 8.1 QUESTIONS

1. What category of structures includes loops?
2. What is the name of the for loop parameter that ends the loop?
3. What for loop parameter changes the counter variable?
4. What happens if you key a semicolon after the parentheses of a for statement?
5. How many statements can be included in a loop?
6. Write a for statement that will print the numerals 3, 6, 12, 24.
7. Write a for statement that will print the numerals 24, 12, 6, 3.

PROBLEM 8.1.1

Write a program that uses a for loop to print the odd numbers from 1 to 21. Save the source file as *ODDLOOP.CPP*.

while Loops

A while loop is similar to a for loop. Actually, while loops are sometimes easier to use than for loops and are better suited for many loops. With a for loop, the parameters in the parentheses control the number of times the loop iterates and the statements in the loop structure are just along for the ride. In a while loop, something inside the loop triggers the loop to stop.

For example, a while loop may be written to ask a user to input a series of numbers until the number 0 is entered. The loop would repeat until the number 0 is entered.

There are two kinds of while loops: the standard while loop and the do while loop. The difference between the two is where the control expression is tested. Let's begin with the standard while loop.

THE WHILE LOOP

The *while loop* repeats a statement or group of statements as long as a control expression is true. Unlike a for loop, a while loop does not use a counter variable. The control expression in a while loop can be any valid expression. The program in Figure 8-5 uses a while loop to repeatedly divide a number by 2 until the number is less than or equal to 1.

In a while loop, the control expression is tested before the statements in the loop begin. Figure 8-6 shows a flow chart of the program in Figure 8-5. If the number provided by the user is less than or equal to 1, the statements in the loop are never executed.

PITFALLS

As with the for loop, placing a semicolon after the closing parenthesis of a while loop will prevent any lines from being iterated.

```
#include<iostream.h>

main()
{
 float num;
 cout << "Please enter the number to divide:";
 cin >> num;
 while (num > 1.0)
  {
   cout << num << '\n';
   num = num / 2;
  }
 return 0;
}
```

FIGURE 8-5
A while loop does not use a counter variable.