EXAMPLE 4.9 The Factorial Numbers

The factorial numbers 0!, 1!, 2!, 3!, ⋯ are defined recursively by the equations

\[
\begin{align*}
0! &= 1 \\
n! &= n(n-1)
\end{align*}
\]

For example, letting \( n = 1 \) in the second equation yields

\[1! = 1((1-1)!) = 1(0!) = 1(1) = 1\]

Similarly, with \( n = 2 \):

\[2! = 2((2-1)!) = 2(1!) = 2(1) = 2\]

and with \( n = 3 \):

\[3! = 3((3-1)!) = 3(2!) = 3(2) = 6\]

The first seven factorial numbers are shown in the table at right.

This program prints all the factorial numbers up to an input limit:

```c
int main()
{
    long bound;
    cout << "Enter a positive integer: ";
    cin >> bound;
    cout << "Factorial numbers < ":
    long f=1, i=1;
    do
        { f *= ++i;
          cout <<", " << f;
        } while (f < bound);
}
```

Enter a positive integer: **1000000**
Factorial numbers < 1000000:
1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880

The do...while loop iterates until its control condition \((f < \text{bound})\) is false.

4.4 THE for STATEMENT

The syntax for the for statement is

```c
for (initialization; condition; update) statement;
```

where initialization, condition, and update are optional expressions, and statement is any executable statement. The three-part \((\text{initialization}; \text{condition}; \text{update})\) controls the loop. The initialization expression is used to declare and/or initialize control variable(s) for the loop; it is evaluated first, before any iteration occurs. The condition expression is used to determine whether the loop should continue iterating; it is evaluated immediately after the initialization; if it is true, the statement is executed. The update expression is used to update the control variable(s); it is evaluated after the statement is executed. So the sequence of events that generate the iteration are:

1. evaluate the initialization expression;
2. if the value of the condition expression is false, terminate the loop;
3. execute the statement;
4. evaluate the update expression;
5. repeat steps 2–4.