When an argument is passed *by-value*, a copy of the argument is assigned implicitly to the corresponding parameter. For example, a copy of of the constant expression **1000.00** is assigned to parameter *P* when function *Payment* is called on line 9 in Figure 1. In the same way, copies of the constant expressions **8.00/1200** and **12** are assigned to parameters *i* and *n* when function *Payment* is called.

```
 1.   #include <iostream>
 2.   #include <iomanip>
 3.   #include <cmath>
 4.   using namespace std;
 5.   double Payment(double P,double i,int n)
 6.   {    return P*i/(1-pow(1+i,-n));
 7.   }
 8.   int main()
 9.   {    double R=Payment(1000.00,8.00/1200,12);
10.        cout << "Your monthly payment is $" << fixed << setprecision(2) << R << ".";
11.        cout << endl;
12.        return 0;
13.   }
```

Figure 1. Function *Payment*.

Storage for parameters is created when the function is called and reclaimed when the function returned. Storage for parameters *P*, *i*, and *n* is created when function *Payment* is called and reclaimed when it returns.