9.14 `static` Class Members (cont.)

Demonstrating `static` Data Members

- The program of Figs. 9.27–9.29 demonstrates a `private static` data member called `count` (Fig. 9.27, line 24) and a `public static` member function called `getCount` (Fig. 9.27, line 18).

```cpp
// Fig. 9.27: Employee.h
// Employee class definition with a static data member to
// track the number of Employee objects in memory
#ifndef EMPLOYEE_H
#define EMPLOYEE_H

#include <string>

class Employee
{
public:
   Employee( const std::string &, const std::string & ); // constructor
   ~Employee(); // destructor
   std::string getFirstName() const; // return first name
   std::string getLastName() const; // return last name

   // static member function
   static unsigned int getCount(); // return # of objects instantiated
private:
   std::string firstName;
   std::string lastName;

```

Fig. 9.27 | Employee class definition with a static data member to track the number of Employee objects in memory. (Part 1 of 2.)

```
   // static data
   static unsigned int count; // number of objects instantiated
}; // end class Employee

#endif
```

Fig. 9.27 | `Employee` class definition with a static data member to track the number of `Employee` objects in memory. (Part 2 of 2.)

```cpp
// Fig. 9.28: Employee.cpp
// Employee class member-function definitions.
#include <iostream>
#include "Employee.h" // Employee class definition
using namespace std;

// define and initialize static data member at global namespace scope
unsigned int Employee::count = 0; // cannot include keyword static

// define static member function that returns number of
// Employee objects instantiated (declared static in Employee.h)
unsigned int Employee::getCount()
{
   return count;
} // end static function getCount

// constructor initializes non-static data members and
// increments static data member count
Employee::Employee( const string &first, const string &last )
   : firstName( first ), lastName( last )
{
```

Fig. 9.28 | Employee class member-function definitions. (Part 1 of 2.)

```
    ++count; // increment static count of employees
    cout << "Employee constructor for " << firstName
       << ' ' << lastName << " called." << endl;
} // end Employee constructor

// destructor deallocates dynamically allocated memory
Employee::~Employee()
{
    cout << "~Employee() called for " << firstName
       << ' ' << lastName << endl;
    --count; // decrement static count of employees
} // end ~Employee destructor

// return first name of employee
string Employee::getFirstName() const
{
    return firstName; // return copy of first name
} // end function getFirstName

// return last name of employee
string Employee::getLastName() const
{
    return lastName; // return copy of last name
} // end function getLastName
```

Fig. 9.28 | Employee class member-function definitions. (Part 2 of 2.)

```cpp
// Fig. 9.29: fig09_29.cpp
// static data member tracking the number of objects of a class.
#include <iostream>
#include "Employee.h" // Employee class definition
using namespace std;

int main()
{
   // no objects exist; use class name and binary scope resolution
   // operator to access static member function getCount
   cout << "Number of employees before instantiation of any objects is "
      << Employee::getCount() << endl; // use class name

   // the following scope creates and destroys
   // Employee objects before main terminates
   {
      Employee e1( "Susan", "Baker" );
      Employee e2( "Robert", "Jones" );

```

Fig. 9.29 | `static` data member tracking the number of objects of a class. (Part 1 of 3.)

```
      // two objects exist; call static member function getCount again
      // using the class name and the scope resolution operator
      cout << "Number of employees after objects are instantiated is "
         << Employee::getCount();

      cout << "\n\nEmployee 1: "
         << e1.getFirstName() << " " << e1.getLastName()
         << "\nEmployee 2: "
         << e2.getFirstName() << " " << e2.getLastName() << "\n\n";
   } // end nested scope in main

   // no objects exist, so call static member function getCount again
   // using the class name and the scope resolution operator
   cout << "\nNumber of employees after objects are deleted is "
      << Employee::getCount() << endl;
} // end main
```

Fig. 9.29 | `static` data member tracking the number of objects of a class. (Part 2 of 3.)

```
Number of employees before instantiation of any objects is 0
Employee constructor for Susan Baker called.
Employee constructor for Robert Jones called.
Number of employees after objects are instantiated is 2

Employee 1: Susan Baker
Employee 2: Robert Jones

~Employee() called for Robert Jones
~Employee() called for Susan Baker

Number of employees after objects are deleted is 0
```

Fig. 9.29 | `static` data member tracking the number of objects of a class. (Part 3 of 3.)

Common Programming Error 9.6

Using the `this` pointer in a `static` member function is a compilation error.

Common Programming Error 9.7

Declaring a `static` member function `const` is a compilation error. The `const` qualifier indicates that a function cannot modify the contents of the object on which it operates, but `static` member functions exist and operate independently of any objects of the class.