Library algorithms, this task seems trivial: (1) sum all of the salaries using `std::accumulate`, (2) calculate the average salary by dividing this sum by the total number of employees, and (3) count the number of employees with above-average salaries using `std::count_if`. Unfortunately, both `std::accumulate` and `std::count_if` require callbacks to return the salary for an `Employee` and to supply the criterion for counting, respectively. The callback for `std::accumulate` must take two parameters — the current running sum and an element from the sequence being summed — and must return the new running sum:

```
struct SalaryAccumulator
{
    long operator()(long currSum, const Employee& e) const
        // returns the sum of currSum and the salary field of e
    {
        return currSum + e.salary;
    }
};
```

The callback for `std::count_if` is a **predicate** (i.e., an expression that yields a Boolean result in response to a yes-or-no question) that takes a single argument and returns **`true`** if an element having that value should be counted and **`false`** otherwise. In this case, we are concerned with `Employee` objects having salaries above the average. Our **predicate functor** must, therefore, carry around that average so it can compare the average to the salary of the employee that is supplied as an argument:

```
class SalaryIsGreater  // function object constructed with a threshold salary
{
    const long d_thresholdSalary;

public:
    explicit SalaryIsGreater(long ts) : d_thresholdSalary(ts) { }
        // construct with a threshold salary, ts

    bool operator()(const Employee& e) const
        // return true if the salary for Employee e is greater than the
        // threshold salary supplied on construction, false otherwise
    {
        return e.salary > d_thresholdSalary;
    }
};
```

Note that, unlike our previous functor classes, `SalaryIsGreater` has a member variable; i.e., it has *state*. This member variable must be initialized, necessitating a constructor. The call operator compares its input argument against this member variable to compute the predicate value.

With these two functor classes defined, we can finally implement the simple three-step algorithm for determining the number of employees with salaries greater than the average: