```
thread_local int local_counter = 0;
std::atomic<int> global_counter(0);
std::mutex m;

void increment(int howmany) {
  for (int i = 0; i < howmany; ++i) {
    local_counter++;
    global_counter++;
  }
  std::lock_guard<std::mutex> g(m);
  std::cout << "Thread exiting with local = " << local_counter
            << " and global = " << global_counter << std::endl;
}
int run_threads(int thread_count, int increments_per_thread) {
  std::vector<std::thread> threads;
  for (int i = 0; i < thread_count; ++i)
    threads.push_back(std::thread(increment, increments_per_thread));
  for (int i = 0; i < thread_count; ++i)
    threads[i].join();
}
```

FIGURE A.9 A C++ program that uses thread-local variables.