```
#include <omp.h>
#include <iostream>
#include <cmath>

#include "timer.h"

// OpenMP simple example notes:
// - not meant for distributed memory parallel systems (e.g., clusters)
// - key concepts: shared memory, synchronization

#define CHUNKSIZE       2500
#define N               50000

int main()
{
        int             nthreads, tid;
        int             i;              // private to each thread (copied mem)
        int             chunk=CHUNKSIZE;// shared among threads (shared mem)
        float           a[N],b[N],c[N]; // shared among threads (shared mem)
        atd::timer_t    timer;


  // initializ arrays
  for(i=0; i<N; i++)
    a[i] = b[i] = i * 1.0;

  // fork a team of threads with each thread having a private tid variable
  #pragma omp parallel private(tid)
  {
    // get master thread id and obtain number of available threads (cores)
    if((tid = omp_get_thread_num())==0) {
      nthreads = omp_get_num_threads();
      std::cerr << "number of threads = " << nthreads << std::endl;
    }
  }// implicit barrier: all threads join master thread and terminate

  timer.start();
  #pragma omp parallel shared(a,b,c,chunk) private(i)
  {
    #pragma omp for schedule(static,chunk) nowait
    for(i=0; i<N; i++)
      c[i] = a[i] + b[i];
  }// implicit barrier: all threads join master thread and terminate
  timer.end();
  std::cerr << "parallel execution: ";
  std::cerr << " (" << timer.elapsed_us() << " us) " << std::endl;

  timer.start();
    for(i=0; i<N; i++)
      c[i] = a[i] + b[i];
  timer.end();
  std::cerr << "serial execution: ";
  std::cerr << " (" << timer.elapsed_us() << " us) " << std::endl;
}
```

```
#include        <iostream>
#include        <string>
#include        <cstdlib>
#include        <sys/time.h>

#include        "timer.h"

namespace atd {

std::ostream& operator<<(std::ostream& s,const timer_t& rhs)
{
  s << "time elapsed: " << rhs.tt << " us" << std::endl;

  return(s);
}


void timer_t::start()
{
  ts = stamp_us();
}


void timer_t::end()
{
  te = stamp_us();
  tt = te - ts;
}


double timer_t::stamp_us()
{
        double          s,us,tod;
        struct timeval  tp;

  // get time of day (tod), return in microseconds

  gettimeofday(&tp,NULL);
  s = static_cast<double>(tp.tv_sec);
  us = static_cast<double>(tp.tv_usec);
  tod = s*1000000.0 + us;
  return(tod);
}

} // namespace atd
```