```cpp
/*
 * Compute pi by Monte Carlo calculation of area of a circle
 *
 * parallel version using OpenMP
 */
#include <iostream>
#include <cstdlib>
#include <omp.h>
using namespace std;

int main(int argc, char *argv[]) {

  const char Usage[] = "Usage: pi <steps> <repeats> (try 1000000 4)";
  if (argc < 3) {
    cerr << Usage << endl; return(1);
  }
  int num_steps = atoi(argv[1]);
  int num_repeats = atoi(argv[2]);

  printf("Computing pi via Monte Carlo using %d steps, repeating %d ti
      num_steps, num_repeats);

  // A little throwaway parallel section just to show num threads
  #pragma omp parallel
  #pragma omp master
  printf("Using %d threads\n", omp_get_num_threads());

  double tot_time=0;
  for (int r=0; r<num_repeats; r++) {
    int count=0;

    double start_time = omp_get_wtime();    // start timing

    #pragma omp parallel for reduction(+:count)
    for (int i=0; i < num_steps; i++) {
      double x = (double) rand()/RAND_MAX;
      double y = (double) rand()/RAND_MAX;
      if (x*x + y*y < 1) count++;
    }

    double end_time = omp_get_wtime();      // stop timing

    double pi = 4.0 * count / num_steps;
    printf("pi = %27.25f (%g sec)\n", pi, end_time - start_time);
    tot_time += end_time - start_time;
  }

  printf("Average ns/iteration: %5.2f\n",
      10e6*tot_time/(num_repeats*num_steps));
  return 0;
}

```