Feb 14, 06 8:46 **RandomWalk_test.cpp** Page 1/1

```
//  file: RandomWalk_test.cpp
//
//  Test program to for RandomWalk class
//
//  Programmer:  Dick Furnstahl  furnstahl.1@osu.edu
//
//  Revision history:
//      05/24/04  Original version
//      02/10/06  Updated and switched names of files
//
//*****************************************************************

// include files
#include <iostream>             // cout and cin
#include <iomanip>              // manipulators like setprecision
#include <fstream>              // file input and output
using namespace std;            // we need this when .h is omitted

#include "RandomWalk.h"

//*****************************************************************
int
main (void)
{
  int npts = 1;                 // number of random numbers to generate

  double x = 0.;                // current x
  double y = 0.;                // current y

  RandomWalk my_random_walk (x, y);     // create a random walk starting
                                        //  at (x,y)

  cout << "How many random numbers? ";
  cin >> npts;

  // output file random_walk_test.dat holds a single walk of length npts
  ofstream out;
  out.open ("RandomWalk_test.dat");

  out << "# (x,y) coordinates of a random walk with " << npts << " points"
    << endl;
  // output the first point
  out << my_random_walk.get_x () << " " << my_random_walk.get_y () << endl;

  // do the walk and output to a file
  for (int i = 0; i < npts; i++)
    {
      my_random_walk.step ();   // take a step
      x = my_random_walk.get_x ();
      y = my_random_walk.get_y ();
      out << x << " " << y << endl;
    }

  out.close ();                 // close the output file
  cout << endl << "Output " << npts << " random walk coordinates to "
    << "RandomWalk_test.dat." << endl;

  return (0);
}
```

Feb 14, 06 10:53 **RandomWalk.h** Page 1/1

```
//  file: RandomWalk.h
//
//  Header file for RandomWalk class
//
//  Programmer:  Dick Furnstahl  furnstahl.1@osu.edu
//
//  Revision history:
//      05/24/04  adapted random_walk.cpp to define a RandomWalk class
//      02/13/06  made get_x and get_y inline
//
#include <gsl/gsl_rng.h>        // GSL random number generators
#include <gsl/gsl_randist.h>    // GSL random distributions

class RandomWalk
{
 public:
  RandomWalk (double x0, double y0);    // constructor: initialize the walk

  ~RandomWalk ();               // destructor

  // use the automatically generated copy constructor

  void step ();                 // take a random step

  double get_x () { return x;}; // get the current x coordinate

  double get_y () {return y;};  // get the current y coordinate

 private:
  int npts;                     // number of random steps so far

  double x;                     // current x value
  double y;                     // current y value

  double lower_limit;           // lower limit of uniform range of step size
  double upper_limit;           // upper limit of uniform range of step size

  double delta_x;               // x-component of step
  double delta_y;               // y-component of step

  gsl_rng *rng_ptr;             // pointer to random number generator (rng)

};
```

```
//  file: RandomWalk.cpp
//
//  Member functions for RandomWalk class that generates random walks
//
//  Programmer:  Dick Furnstahl  furnstahl.1@osu.edu
//
//  Revision history:
//      05/24/04  switched random_walk.cpp to define a RandomWalk class
//
//  Notes:
//   * implements method 2 from the list in section 6.10
//      of the Landau/Paez text.
//   * random numbers are generated uniformly from a to b
//   * uses the GSL random number functions
//   * both the gsl_rng.h and gsl_randist.h header files are needed
//   * the current version uses the gsl_rng_taus random number
//      generator.  There are many other choices (just change
//      the name in the gsl_rng_alloc statement).  See the GSL
//      manual for a list of generators and their properties.
//
//**********************************************************************

// include files
#include <cmath>

#include "RandomWalk.h"

// function prototypes
extern unsigned long int random_seed ();        // routine to generate a seed

//**********************************************************************

// Constructor for RandomWalk
RandomWalk::RandomWalk (double x0, double y0)
{
  npts = 1;                     // start with one point

  unsigned long int seed = random_seed ();      // seed for rng

  // initialize step size
  lower_limit = -sqrt (2.);     // lower limit of uniform range
  upper_limit = sqrt (2.);      // upper limit of uniform range

  x = x0;                       // current x
  y = y0;                       // current y
  delta_x = 0.;                 // uniform random number from a to b
  delta_y = 0.;                 // 2nd uniform random number from a to b

  rng_ptr = gsl_rng_alloc (gsl_rng_taus);       // allocate the rng
  seed = random_seed ();        // generate a seed
  gsl_rng_set (rng_ptr, seed);  // seed the rng
}

// Destructor for RandomWalk
RandomWalk::~RandomWalk ()
{
  gsl_rng_free (rng_ptr);       // free the random number generator
}

// Take a single random step with lower_limit < delta_x, delta_y < upper_limit
void
RandomWalk::step ()
{
  delta_x = gsl_ran_flat (rng_ptr, lower_limit, upper_limit);
  delta_y = gsl_ran_flat (rng_ptr, lower_limit, upper_limit);

  // increment the current coordinates
  x += delta_x;
  y += delta_y;
```



```
  npts++;                       // increment the point counter
}
```