```
//  file: eigen_test.cpp
//
//  Program to test the GSL eigenvalue/eigenvector routines
//
//  Programmer:  Dick Furnstahl  furnstahl.1@osu.edu
//
//  Revision history:
//      01/04/04  original version, translated from eigen_test.cpp
//
//  Notes:
//   * Uses the GSL functions for computing eigenvalues
//      and eigenvectors of matrices.  The basic flowchart is:
//       * define names for matrices and vectors
//       * allocate space for these matrices and vector
//       * load the matrix to be diagonalized (pointed to by Amat_ptr)
//       * find the eigenvalues and eigenvectors with gsl_eigensymmv
//       * sort the results and print them out
//   * Based on the documentation for the GSL library under
//      "Eigensystems" and Chap. 15 of "Computational Physics"
//      by Landau and Paez.
//   * As a convention (advocated in "Practical C"), we'll append
//      "_ptr" to all pointers.
//   * We've added two calls to "clock" to time the calculation.
//
//  To do:
//   * Output to a file, suitable for plotting
//
///*****************************************************************

// include files
#include <iostream>             // note that .h is omitted
#include <iomanip>              // note that .h is omitted
using namespace std;
#include <time.h>
#include <gsl/gsl_eigen.h>      // include the appropriate GSL header file


//************************** main program **************************
int
main ()
{
  clock_t start, end;           // start and stop times
  int dimension;                // dimension of the matrices and vectors
  double hilbert;               // an entry in a Hilbert matrix

  gsl_matrix *Amat_ptr;         // original gsl matrix to process
  gsl_vector *Eigval_ptr;       // gsl vector with eigenvalues
  gsl_matrix *Eigvec_ptr;       // gsl matrix with eigenvectors
  gsl_eigen_symmv_workspace *worksp;    // the workspace for gsl

  // the following two objects are for output only
  double eigenvalue;            // one of the eigenvalues of the matrix
  gsl_vector *eigenvector_ptr;  // one of the eigen vectors of the matrix

  // pick the dimension of the matrix
  cout << "Enter the dimension of the matrix: ";
  cin >> dimension;

  // allocate space for the vectors, matrices, and workspace
  Amat_ptr = gsl_matrix_alloc (dimension, dimension);
  Eigval_ptr = gsl_vector_alloc (dimension);
  Eigvec_ptr = gsl_matrix_alloc (dimension, dimension);
  worksp = gsl_eigen_symmv_alloc (dimension);
  eigenvector_ptr = gsl_vector_alloc (dimension);
  eigenvalue = 0;

  // Load the Hilbert matrix pointed to by Amat_ptr
  for (int i = 0; i < dimension; i++)
    {
```

```
      for (int j = 0; j < dimension; j++)
        {
          hilbert = 1. / ((float) (i + j + 1)); // i,j start at 0
          gsl_matrix_set (Amat_ptr, i, j, hilbert);
        }
    }

  // Find the eigenvalues and eigenvectors of the real, symmetric
  // matrix pointed to by Amat_ptr.  It is partially destroyed
  // in the process. The eigenvectors are pointed to by
  // Eigvec_ptr and the eigenvalues by Eigval_ptr.

  start = clock ();             // start the clock to time the next routine
  gsl_eigen_symmv (Amat_ptr, Eigval_ptr, Eigvec_ptr, worksp);
  end = clock ();               // stop the clock and print the elapsed time

  cout << " Finding the eigenvalues/vectors took " << fixed
        << setprecision(3)
        <<  (double) (end - start) / (double) CLOCKS_PER_SEC
        << " seconds\n\n";

  // sort the eigenvalues and eigenvectors in ascending order
  gsl_eigen_symmv_sort (Eigval_ptr, Eigvec_ptr, GSL_EIGEN_SORT_ABS_ASC);

  // print out the results
  // comment starting here when running large matrices
  for (int i = 0; i < dimension; i++)
    {
      eigenvalue = gsl_vector_get (Eigval_ptr, i);
      gsl_matrix_get_col (eigenvector_ptr, Eigvec_ptr, i);

      cout << "eigenvalue = " << scientific << eigenvalue << endl;

      cout << "eigenvector = \n";
      for (int j = 0; j < dimension; j++)
      {
          cout << scientific << gsl_vector_get (eigenvector_ptr, j) << endl;
      }
    }
  // end the comment here when running large matrices

  // free the space used by the vector and matrices and workspace
  gsl_matrix_free (Eigvec_ptr);
  gsl_vector_free (Eigval_ptr);
  gsl_matrix_free (Amat_ptr);
  gsl_vector_free (eigenvector_ptr);
  gsl_eigen_symmv_free (worksp);

  return (0);                   // successful completion
}

//*************************************************************
```