```
//  file: GslSpline.h
//
//  Header file for the GslSpline C++ class.
//
//  Programmer:  Dick Furnstahl  furnstahl.1@osu.edu
//
//  Revision history:
//      02/10/09  original version
//
//  Notes:
//   * We encapsulate GSL spline functions in this class
//   * The GSL header file is included in GslSpline.cpp and
//      anywhere we want to create GslSpline objects.
//   * Based on the documentation for the GSL library under
//      "Interpolation" and on the "Using GSL Interpolation Functions"
//      780.20 handout.
//   * See the GSL documentation for error handling details.
//
//  To do:
//   * add other splining methods
//
//*****************************************************************

// The ifndef/define macro ensures that the header is only included once
#ifndef GSLSPLINE_H
#define GSLSPLINE_H

// include files
#include <string>     // C++ strings
#include <gsl/gsl_spline.h>       // header for gsl spline routines

class Spline
{
  public:
    Spline (double* x_array, double* y_array, int num_pts,
            std::string type);  // constructor
    ~Spline ();  // destructor

    // accessor functions
    double y (const double x);    // find y(x)
    double yp (const double x);   // find y'(x)
    double ypp (const double x);  // find y''(x)

  private:
    std::string spline_type;       // type of spline
    gsl_interp_accel *accel_ptr;   // accelerator pointer
    gsl_spline *spline_ptr;        // spline pointer

};

#endif
```



```
//  file: GslSpline.cpp
//
//  Definitions for the GslSpline C++ class.
//
//  Programmer:  Dick Furnstahl  furnstahl.1@osu.edu
//
//  Revision history:
//     01/25/09  Original version using gsl_cubic_spline_test.cpp as a guide.
//
//*****************************************************************
// include files
#include <iostream>
#include <string>     // C++ strings
#include <gsl/gsl_errno.h>       // header for gsl error handling
#include <gsl/gsl_spline.h>      // header for gsl splining routines
#include "GslSpline.h"       // include the header for this class

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

// Constructor for Spline
Spline::Spline(double* x_array, double* y_array, int num_pts, std::string type)
{
  spline_type = type;    // set the private variable for the spline type

  // Allocate the accelerator
  accel_ptr = gsl_interp_accel_alloc ();

  // Allocate the spline according to spline_type
  if (spline_type == "cubic")
  {
    spline_ptr = gsl_spline_alloc (gsl_interp_cspline, num_pts);
  }
  else if (spline_type == "linear")
  {
    spline_ptr = gsl_spline_alloc (gsl_interp_linear, num_pts);
  }
  else
  {
    std::cout << "Illegal spline type!" << std::endl;
    exit (1);  // time to quit!
  }

  // Initialize the spline
  gsl_spline_init (spline_ptr, x_array, y_array, num_pts);
}

Spline::~Spline () // Destructor for Spline
{
  // Free the accelerator and spline object
  gsl_spline_free (spline_ptr);
  gsl_interp_accel_free (accel_ptr);
}

double Spline::y (const double x)  // find y(x)
{
  return gsl_spline_eval (spline_ptr, x, accel_ptr);
}

double Spline::yp (const double x)  // find y'(x)
{
  return gsl_spline_eval_deriv (spline_ptr, x, accel_ptr);
}

double Spline::ypp (const double x)  // find y''(x)
{
  return gsl_spline_eval_deriv2 (spline_ptr, x, accel_ptr);
}

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

```
//  file: gsl_spline_test_class.cpp
//
//  Test program for the gsl spline routines using the Spline class
//
//  Programmer:  Dick Furnstahl  furnstahl.1@osu.edu
//
//  Revision history:
//      02/10/09 -- created from gsl_cubic_spline_test.cpp
//
//  Notes:
//   * uses the GSL interpolation functions (see online documentation)
//
//*************************************************************
// include files
#include <iostream>    // cout and cin
#include <iomanip>     // manipulators like setprecision
#include <cmath>
#include <string>     // C++ strings
using namespace std;
#include "GslSpline.h"  // Header file for the GSL Spline class

inline double sqr (double z) {return z*z;}  // inline function for z^2

int
main (void)
{
  const int NMAX = 300;   // maximum number of array points
  double x_values[NMAX], y_values[NMAX];

  // Test: interpolate y = sin(x^2) from 0 to 2 with 20 points
  double xmin = 1.;
  double xmax = 3.;
  int npts = 20;
  double deltax = (xmax - xmin)/double(npts-1);
  for (int i = 0; i < npts; i++)
  {
    double x_temp = double(i) * deltax;   // grid of x points
    x_values[i] = x_temp;
    y_values[i] = sin (x_temp * x_temp);
  }

  // Make the spline object
  string type = "cubic";
  Spline my_cubic_spline (x_values, y_values, npts, type);

  double x;
  cout << "Enter x: ";
  cin >> x;    // test point

  // Evaluate the spline and derivatives
  double y = my_cubic_spline.y (x);
  double y_deriv = my_cubic_spline.yp (x);
  double y_deriv2 = my_cubic_spline.ypp (x);

  double x_sq = sqr(x);

  cout << "  x   y_exact  y_spline  y'_exact y'_spline";
  cout << " y''_exact y''_spline" << endl;
  cout << fixed << setprecision(6)
       << x << " " << sin(x_sq) << " " <<  y << " "
       <<  2.*x*cos(x_sq) << " " << y_deriv << " "
       <<  -4.*x_sq* sin(x_sq) + 2.*cos(x_sq) << " " <<  y_deriv2
       << endl;

  return (0);      // successful completion
}
```