```
//  file: diffeq_test.cpp
//
//  Program to study the error in differential equation algorithms
//
//  Programmer:  Dick Furnstahl  furnstahl.1@osu.edu
//
//  Revision history:
//      02/07/04  original version, translated from diffeq_test.c
//      01/30/05  comments improved and function names changed
//      01/22/06  improved code organization
//
//  Notes:
//   * Based on the discussion of differential equations in Chap. 9
//      of "Computational Physics" by Landau and Paez
//   * As a convention (advocated in "Practical C++"), we'll append
//      "_ptr" to all pointers.
//
//*************************************************************

// include files
#include <iostream>              // note that .h is omitted
#include <iomanip>               // note that .h is omitted
#include <fstream>               // note that .h is omitted
using namespace std;             // we need this when .h is omitted
#include <cmath>
#include "diffeq_routines.h"     // diffeq routine prototypes

// function prototypes
double rhs (double t, double y[], int i, void *params_ptr);
double exact_answer (double t, void *params_ptr);

// structures
typedef struct                   // define a type to hold parameters
{
  double alpha;
  double beta;
}
f_parameters;                    // now we can define a structure of this type
                                 //   using the keyword "f_parameters"

//************************* main program ***************************
int
main (void)
{
  void *params_ptr;                // void pointer passed to functions
  f_parameters funct_parameters;   // parameters for the function

  const int N = 1;                 // size of arrays of y functions
  double y_euler[N], y_rk4[N];     // arrays of y functions

  ofstream out ("diffeq_test.dat");          // open the output file

  funct_parameters.alpha = 1.;  // function parameter to be passed
  funct_parameters.beta = 1.;   // function parameter to be passed
  params_ptr = &funct_parameters;       // structure to pass to function

  double tmin = 0.;                // starting t value
  double tmax = 3.;                // last t value
  y_euler[0] = 1.0;                // initial condition for y(t)
  y_rk4[0] = 1.0;                  // initial condition for y(t)

  // print out a header line and the first set of points
  out << "#   t      y_euler(t)      y_rk4(t)     y_exact(t) \n";
  out << scientific << setprecision (9)
    << tmin << " "
    << y_euler[0] << " "
    << y_rk4[0] << " " << exact_answer (tmin, params_ptr) << endl;

  double h = 0.1;                  // initialize mesh spacing
```



```
  for (double t = tmin; t <= tmax; t += h)
  {

    // find y(t+h) and output vs. t+h
    euler (N, t, y_euler, h, rhs, params_ptr);  // Euler's algorithm

    runge4 (N, t, y_rk4, h, rhs, params_ptr);           // 4th order R-K

    out << scientific << setprecision (9)
      << t + h << " "
      << y_euler[0] << " "
      << y_rk4[0] << " " << exact_answer (t + h, params_ptr) << endl;
  }

  cout << "data stored in diffeq_test.dat\n";
  out.close ();                    // close the output file

  return (0);                      // successful completion
}


//************************* rhs ************************************
//
//  * This is the function defining the right hand side of the diffeq
//
//*****************************************************************
double
rhs (double t, double y[], int i, void *params_ptr)
{
  double a = ((f_parameters *) params_ptr)->alpha;
  // double b = ((f_parameters *) params_ptr)->beta;

  if (i == 0)
  {
    return (-a * t * y[0]);
  }

  return (1);                      // something's wrong if we get here
}

//********************** exact_answer **************************
//
//  * This is the exact answer for y(t)
//
//**************************************************************
double
exact_answer (double t, void *params_ptr)
{
  // recover a and b from the void pointer params_ptr
  double a = ((f_parameters *) params_ptr)->alpha;
  double b = ((f_parameters *) params_ptr)->beta;

  return (b * exp (-a * t * t / 2.));
}
```