```
   Help
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2008+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
#else
/**********************************************************
*   CPS - A simple C PDE solver                           *
*                                                         *
*   Copyright (c) 2007,                                   *
*     Maya Briani       <m.briani@iac.rm.cnr.it>,         *

*     Francesco Ferreri <francesco.ferreri@gmail.com>,    *
*     Roberto Natalini  <r.natalini@iac.rm.cnr.it>,       *
*     Marco Papi        <m.papi@iac.rm.cnr.it>            *
*                                                         *

**********************************************************/
#include "cps_grid_tuner.h"
#include "cps_grid.h"
#include "cps_grid_node.h"
#include "cps_utils.h"
#include "cps_assertions.h"

/* private implementation functions */
static int generic_tuner(grid_tuner *tuner, grid *g){
  /* generically tune grid time */
  REQUIRE("tuner_not_null", tuner != NULL);
  REQUIRE("grid_not_null", g != NULL);

  g->delta[T_DIM] = (g->max_value[T_DIM]-g->min_value[T_
    DIM])/(double)(g->ticks[T_DIM]);
  g->is_tuned = 1;

  ENSURE("grid_is_tuned",g->is_tuned);
  return OK;
}

/* public interface */

int grid_tuner_create(grid_tuner **tuner){
  STANDARD_CREATE(tuner,grid_tuner);
  (*tuner)->tuners[GENERIC_TUNER] = generic_tuner;
  return OK;
}

int grid_tuner_destroy(grid_tuner **tuner){
  STANDARD_DESTROY(tuner);
  return OK;
}

int grid_tuner_set_argument(grid_tuner *tuner, void *arg){
  /* set argument to tuner */
  REQUIRE("tuner_not_null", tuner != NULL);
  REQUIRE("argument_not_null", arg != NULL);

  tuner->argument = arg;

  ENSURE("argument_set", tuner->argument == arg);
  return OK;
}

int grid_tuner_set_tuner(grid_tuner *tuner, int type, grid_
    tuner_proc proc){
  /* set a tuner procedure of given type */
  REQUIRE("tuner_not_null", tuner != NULL);
  REQUIRE("valid_type", type > 0 && type < MAX_TUNERS);
  REQUIRE("tuner_proc_not_null", proc != NULL);

  tuner->tuners[type] = proc;

  return OK;
}

int grid_tuner_apply(grid_tuner *tuner, int type, grid *
    grid){
  /* apply given type of tuner */
  grid_tuner_proc proc;
  REQUIRE("tuner_not_null", tuner != NULL);
  REQUIRE("valid_type", type >= 0 && type < MAX_TUNERS);
  REQUIRE("grid_not_null", grid != NULL);

    proc = tuner->tuners[type];
  proc(tuner,grid);
  grid->is_tuned = 1;
  ENSURE("grid_is_tuned", grid->is_tuned);
  return OK;
}

#endif //PremiaCurrentVersion
```

References