

Help

```
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2008+2) //The "#els
#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 "
href../../common/math/highdim_solver/cps_stencil_h_src.pdfcps_stencil.h"
#include "
href../../common/math/highdim_solver/cps_stencil_pattern_h_src.pdfcps_stencil
#include "
href../../common/math/highdim_solver/cps_assertions_h_src.pdfcps_assertions.h
#include "
href../../common/math/highdim_solver/cps_utils_h_src.pdfcps_utils.h"

int stencil_pattern_create(stencil_pattern **s)
{

    STANDARD_CREATE(s, stencil_pattern);

    return OK;
}

int stencil_pattern_destroy(stencil_pattern **s)
{

    int k;
    stencil_application *sapp;

    for (k = 0; k < MAX_STENCIL_SIZE; k++)
    {
        sapp = (*s)->application[k];
        if (sapp)
```

```

        {
            stencil_application_destroy(&sapp);
        }
    }
    STANDARD_DESTROY(s);
    return OK;
}

int stencil_pattern_put(stencil_pattern *s, unsigned int entry, stencil_application *sapp)
{
    /* put a couple (pos,value) at entry */
    REQUIRE("stencil_pattern_not_null", s != NULL);
    REQUIRE("valid_entry", entry >= 0 && entry < MAX_STENCIL_SIZE);

    s->application[entry] = sapp;
    s->count++;
    return OK;
}

int stencil_pattern_item(const stencil_pattern *s, stencil_application **sapp)
{
    /* get a couple (pos,value) stored at entry */
    REQUIRE("stencil_pattern_not_null", s != NULL);
    REQUIRE("valid_cursor", s->cursor >= 0 && s->cursor < MAX_STENCIL_SIZE);

    *sapp = s->application[s->cursor];

    ENSURE("result_not_null", (*sapp) != NULL);
    return OK;
}

int stencil_pattern_start(stencil_pattern *s)
{
    /* set cursor at first element not null */
    REQUIRE("stencil_pattern_not_null", s != NULL);

    s->cursor = 0;

    while (!stencil_pattern_after(s) && (s->application[s->cursor] == NULL))
    {
        stencil_pattern_forth(s);
    }
}

```

```

    }
    return OK;
}

int stencil_pattern_after(const stencil_pattern *s)
{
    /* is cursor at end */
    REQUIRE("stencil_pattern_not_null", s != NULL);
    return (s->cursor >= MAX_STENCIL_SIZE);
}

int stencil_pattern_forth(stencil_pattern *s)
{
    /* move cursor forth till next not-null element is found */
    REQUIRE("stencil_pattern_not_null", s != NULL);
    REQUIRE("not_after", !stencil_pattern_after(s));

    do
    {
        s->cursor = s->cursor + 1;
    }
    while (!(stencil_pattern_after(s)) && (s->application[s->cursor] == NULL));

    return OK;
}

int stencil_application_create(stencil_application **sapp)
{
    STANDARD_CREATE(sapp, stencil_application);
    return OK;
}

int stencil_application_destroy(stencil_application **sapp)
{
    STANDARD_DESTROY(sapp);
    return OK;
}

int stencil_application_is_internal(const stencil_application *sapp)

```

```

{
    /* is application inside current grid limits ? */
    REQUIRE("stencil_application_not_null", sapp != NULL);

    return (sapp->grid_location == GLOC_INTERNAL);
}

int stencil_application_is_external(const stencil_application *sapp)
{
    /* is application outside current grid limits ? */
    REQUIRE("stencil_application_not_null", sapp != NULL);

    return (sapp->grid_location == GLOC_EXTERNAL);
}

int stencil_application_is_boundary(const stencil_application *sapp)
{
    /* is application on boundary ? */
    REQUIRE("stencil_application_not_null", sapp != NULL);

    return (sapp->grid_location == GLOC_BOUNDARY);
}

int stencil_application_set_internal(stencil_application *sapp)
{
    REQUIRE("stencil_application_not_null", sapp != NULL);

    sapp->grid_location = GLOC_INTERNAL;
    return OK;
}

int stencil_application_set_external(stencil_application *sapp)
{
    REQUIRE("stencil_application_not_null", sapp != NULL);

    sapp->grid_location = GLOC_EXTERNAL;
    return OK;
}

int stencil_application_set_boundary(stencil_application *sapp)
{

```

```

    REQUIRE("stencil_application_not_null", sapp != NULL);

    sapp->grid_location = GLOC_BOUNDARY;
    return OK;
}

int stencil_application_set_order(stencil_application *sapp, unsigned int ord)
{
    REQUIRE("stencil_application_not_null", sapp != NULL);

    sapp->order = ord;
    return OK;
}
/* end -- stencil_pattern.c */

#endif //PremiaCurrentVersion

```