

Help

```

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2008+2) //The "#els
#else
/*****
/*                                vector.c                                */
/*****
/*                                */
/* type VECTOR                    */
/*                                */
/* Copyright (C) 1992-1995 Tomas Skalicky. All rights reserved.          */
/*                                */
/*****
/*                                */
/*      ANY USE OF THIS CODE CONSTITUTES ACCEPTANCE OF THE TERMS          */
/*      OF THE COPYRIGHT NOTICE (SEE FILE COPYRIGHT.H)                    */
/*                                */
/*****

#include <stddef.h>
#include <stdlib.h>
#include <string.h>

#include "laspack/highdim_vector.h"
#include "laspack/errhandl.h"
#include "laspack/copyright.h"

void V_Constr(Vector *V, char *Name, size_t Dim, InstanceType Instance,
              Boolean OwnData)
/* constructor of the type Vector */
{
    V->Name = (char *)malloc((strlen(Name) + 1) * sizeof(char));
    if (V->Name != NULL)
        strcpy(V->Name, Name);
    else
        LASError(LASMemAllocErr, "V_Constr", Name, NULL, NULL);
    V->Dim = Dim;
    V->Instance = Instance;
    V->LockLevel = 0;
    V->Multipl = 1.0;
    V->OwnData = OwnData;
}

```

```
if (OwnData)
{
    if (LASResult() == LASOK)
    {
        V->Cmp = (double *)malloc((Dim + 1) * sizeof(double));
        if (V->Cmp == NULL)
            LASError(LASMemAllocErr, "V_Constr", Name, NULL, NULL);
    }
    else
    {
        V->Cmp = NULL;
    }
}

void V_Destr(Vector *V)
/* destructor of the type Vector */
{
    if (V->Name != NULL)
        free(V->Name);
    if (V->OwnData)
    {
        if (V->Cmp != NULL)
        {
            free(V->Cmp);
            V->Cmp = NULL;
        }
    }
}

void V_SetName(Vector *V, char *Name)
/* (re)set name of the vector V */
{
    if (LASResult() == LASOK)
    {
        free(V->Name);
        V->Name = (char *)malloc((strlen(Name) + 1) * sizeof(char));
        if (V->Name != NULL)
            strcpy(V->Name, Name);
        else
            LASError(LASMemAllocErr, "V_SetName", Name, NULL, NULL);
    }
}
```

```
    }
}

char *V_GetName(Vector *V)
/* returns the name of the vector V */
{
    if (LASResult() == LASOK)
        return (V->Name);
    else
        return ("");
}

size_t V_GetDim(Vector *V)
/* returns dimension of the vector V */
{
    size_t Dim;

    if (LASResult() == LASOK)
        Dim = V->Dim;
    else
        Dim = 0;
    return (Dim);
}

void V_SetCmp(Vector *V, size_t Ind, double Val)
/* set a value of a vector component */
{
    if (LASResult() == LASOK)
    {
        if (Ind > 0 && Ind <= V->Dim && V->Instance == Normal && V->OwnData == True)
        {
            V->Cmp[Ind] = Val;
        }
        else
        {
            LASError(LASRangeErr, "V_SetCmp", V->Name, NULL, NULL);
        }
    }
}

void V_SetAllCmp(Vector *V, double Val)
```

```
/* set all vector components equal Val */
{
    size_t Dim, Ind;
    double *VCmp;

    if (LASResult() == LASOK)
    {
        Dim = V->Dim;
        VCmp = V->Cmp;
        for (Ind = 1; Ind <= Dim; Ind++)
            VCmp[Ind] = Val;
        V->Multipl = 1.0;
    }
}

void V_SetRndCmp(Vector *V)
/* set random components of the vector V */
{
    size_t Dim, Ind;
    double *VCmp;

    if (LASResult() == LASOK)
    {
        Dim = V_GetDim(V);
        VCmp = V->Cmp;
        for (Ind = 1; Ind <= Dim; Ind++)
        {
            VCmp[Ind] = (double)rand() / ((double)RAND_MAX + 1.0);
        }
        V->Multipl = 1.0;
    }
}

double V_GetCmp(Vector *V, size_t Ind)
/* returns the value of a vector component */
{
    double Val;

    if (LASResult() == LASOK)
    {
        if (Ind > 0 && Ind <= V->Dim)
```

```
        {
            Val = V->Cmp[Ind];
        }
    else
    {
        LASError(LASRangeErr, "V_GetCmp", V->Name, NULL, NULL);
        Val = 0.0;
    }
}
else
{
    Val = 0.0;
}
return (Val);
}

void V_AddCmp(Vector *V, size_t Ind, double Val)
/* add a value to a vector component */
{
    if (LASResult() == LASOK)
    {
        if (Ind > 0 && Ind <= V->Dim && V->Instance == Normal && V->OwnData == TRUE)
        {
            V->Cmp[Ind] += Val;
        }
        else
        {
            LASError(LASRangeErr, "V_AddCmp", V->Name, NULL, NULL);
        }
    }
}

void V_Lock(Vector *V)
/* lock the vector V */
{
    if (V != NULL)
        V->LockLevel++;
}

void V_Unlock(Vector *V)
/* unlock the vector V */
```

```
{
  if (V != NULL)
  {
    V->LockLevel--;
    if (V->Instance == Tempor && V->LockLevel <= 0)
    {
      V_Destr(V);
      free(V);
    }
  }
}

#endif //PremiaCurrentVersion
```