```
/*************************************************************************/
/* LCRAMAC w.2.0                                                         */
/* Obliczanie ram plaskich sprezystych                                   */
/* prowadzone w jednostkach kN, cm, podczas gdy dane wpisywane w roznych */
/* jednostkach                                                           */
/* (C) Copyright Leszek CHODOR , grudzien 1989, styczen 1992             */
/*************************************************************************/
#include <process.h>
#include <conio.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <conio.h>
#include <ctype.h>
#include <io.h>

#define DOUBLE sizeof(double)
#define INT    sizeof(int)
#define NF  3                   /* Node Free    */
#define EF  6                   /* Element Free */

struct DescriptData
     { int NodeNumber;    double *Node[2];       /* [X,Y] */
       int ElementNumber; int *Element[3]; char **TypeElement;
                                              /* [W1,W2,TP], [TE] */
       int SectionNumber; double *Section[3];   /* [E, Ix, A ]  */
       int BondNumber; int *BondNode; char *BondDegree; double *BondStif[2];
                                              /* [W], [S], [Podat., Blad]*/
       int Size,           /* Liczba stopni swobody       */
           Band;           /* Szerokosc polpasma          */
     } DataGeom[1];

 struct DescriptForce
    { int SchemeNumber; int *Scheme; /*Schematy obciazen*/
      int ConcenTypeNumber; double *ConcenTypeForce[3];  /*[Py,Px,M]*/
      int ConcenNumber; int *ConcenForce[3]; double *ConcenX1;
                                /*[Scheme, Typ, Element(-Wezel)] [X1]*/
      int UniformTypeNumber; double *UniformTypeForce[4]; /*[qx1,qy1,qx2,qy2]*/
      int UniformNumber, **UniformForce;       /*[Typy w schemacie]*/
      int ThermalTypeNumber; double *ThermalTypeForce[4]; /*[dt1,dt2,at,h]*/
      int ThermalNumber, **ThermalForce;       /*[Typy w schemacie]*/
   } DataForce[1];

static double StifElement[EF*EF], CosSin[2], StifLokal[EF*EF];
static char Key[101];

/*struct FORCE { double x,
               Mmax, QlMmax, QrMmax,
               Mmin, QlMmin, QrMmin,
               MQl , Qlextr,
               MQr , Qrextr; };*/

/*************************************************************************/
/*- WriteXY  wersja 1.0 - PISZ NA EKRAN                                 -*/
/*************************************************************************/
int WriteXY( int x, int y, int Color, char *Format, ...)
{ static char Inscript[161];  va_list ArgPtr;
  int Length; register Count;
  va_start(ArgPtr, Format);
  if( ((Length=vsprintf(Inscript,Format,ArgPtr))==EOF) || (Length>80) )
      return(-1);
  for(Count=Length-1; Count>=0; Count--)
      {Inscript[2*Count]=Inscript[Count]; Inscript[2*Count+1]=Color;}
```

```
  puttext(x,y,x+Length-1,y,Inscript);
  va_end(arg_ptr);
  return(0);
}/*WriteXY*/

  #define  START       0
  #define  PROBLEM     1
  #define  DATA        2
  #define  STATIC      3
  #define  END         4
/*-------------------------------------------------------------------------*/
/*-  LoadDataGEOM()   CZYTANIE DANYCH z pliku dyskowego                    -*/
/*- Zwrot: 0=Sukces;

          10=Zla struktura pliku danych
          11=Liczba wezlow mniejsza od 2
          12=Brak pamieci na wspolrzedne X wezl˘w
          13=Brak pamieci na wspolrzedne Y wezl˘w
          14=Wezly systemu o jednakowych wspolrzednych

          15=Liczba elementow mniejsza od 1
          16=Brak pamieci na wezly poczatkowe element˘w
          17=Brak pamieci na wezly koncowe element˘w
          18=Brak pamieci na typy przekroj˘w element˘w
          19=Brak pamieci na typy element˘w
          20=Numer wezla wiekszy od liczby wezlow
          21=Numer wezla mniejszy od 1
          22=Numer typu przekroju mniejszy od 1
          23=Typ Elementu inny niz < xx, ox, xo lub oo >
          24=Jednakowy lewy i prawy wezel Elementu
          25=Elementy o jednakowych wezlach

          26=Liczba typ˘w przekroj˘w mniejsza do 1
          27=Liczba typ˘w przekroj˘w wieksza od liczby element˘w
          28=Typ przekroju elementu wiekszy od liczby typ˘w
          29=Brak pamieci na pola przekroj˘w
          30=Brak pamieci na momenty bezwladnosci
          31=Brak pamieci na moduly odksztalcalnosci podluznej
          32=Ujemna lub zerowa charakterystyka geometryczna

          33=Liczba wiezow mniejsza od 3
          34=Brak pamieci na wezly z podporami
          35=Brak pamieci na stopnie swobody podp˘r
          36=Brak pamieci na podatnosci podp˘r
          37=Brak pamieci na bledy polozenia podp˘r
          38=Numer wezla z podpora wiekszy od liczby wezl˘w
          39=Stopien swobody wiezi r˘zny od < 1, 2 lub 3 >
          40=Ujemna podatnosc podpory
          */
/*-------------------------------------------------------------------------*/
int LoadDataGeom(FILE *File, struct DescriptData *DataGeom)
{ int i, j, Etap; long Sum=0L;
  int ix=0;

  for(i=0;i<6;i++) fgets(Key,100,File);
    for(i=0;i<(int)strlen(Key);i++) Sum+=(long)(i*(int)Key[i]);
        if(Sum!=186466L) return(10);
  for(i=0;i<4;i++) fgets(Key,100,File);
/*  fscanf(File,"\nEtap zadania : %s (%d)\n",Key,&Etap);
      if((Etap<DATA)||(Etap>END)) return(10);*/

/* WEZLY */
  for(i=0;i<6;i++) fgets(Key,100,File);
  ix=-1; fscanf(File," Liczba wezlow ... %d\n\n",&ix);
   if(ix<2) return(11);
  DataGeom->NodeNumber=ix;
  if((DataGeom->Node[0]= /*X*/
     (double *)farcalloc(ix,DOUBLE))==NULL) return(12);
```

```
  if((DataGeom->Node[1]= /*Y*/
     (double *)farcalloc(ix,DOUBLE))==NULL) return(13);
  for(i=0;i<3;i++) fgets(Key,100,File);
  for(i=0;i<ix;i++)
    { fscanf(File," %*d %lf %lf\n",
                        &DataGeom->Node[0][i], &DataGeom->Node[1][i]);
      DataGeom->Node[0][i]*=100; DataGeom->Node[1][i]*=100; /*zamiana z [m] na
[cm]*/
    }
 for(i=0;i<ix-1;i++)
  for(j=i+1;j<ix;j++)
   if( (fabs(DataGeom->Node[0][i]-DataGeom->Node[0][j])<DBL_EPSILON)&&
       (fabs(DataGeom->Node[1][i]-DataGeom->Node[1][j])<DBL_EPSILON))
return(14);
/*ELEMENTY*/
  for(i=0;i<2;i++) fgets(Key,100,File);
  ix=-1; fscanf(File,"Liczba element%*s%*s %d\n\n",&ix);
   if(ix<1) return(15);
  DataGeom->ElementNumber=ix;
  if((DataGeom->Element[0]=(int *)farcalloc(ix,INT))==NULL) return(16);/*W1*/
  if((DataGeom->Element[1]=(int *)farcalloc(ix,INT))==NULL) return(17);/*W2*/
  if((DataGeom->Element[2]=(int *)farcalloc(ix,INT))==NULL) return(18);/*TP*/
  if((DataGeom->TypeElement= /*TE*/
              (char **)farcalloc(ix,sizeof(char *)))==NULL) return(19);
  for(i=0;i<ix;i++)
    if((DataGeom->TypeElement[i]=
          (char *)farmalloc(3*sizeof(char)))==NULL) return(19);
  for(i=0;i<3;i++) fgets(Key,100,File);
  for(i=0;i<DataGeom->ElementNumber;i++)
   { fscanf(File,"%*d %d %d %s %d\n",
                    &DataGeom->Element[0][i], &DataGeom->Element[1][i],
                    DataGeom->TypeElement[i], &DataGeom->Element[2][i]);
     if(DataGeom->Element[0][i]>DataGeom->NodeNumber||
        DataGeom->Element[1][i]>DataGeom->NodeNumber )        return(20);
     if(DataGeom->Element[0][i]<1||DataGeom->Element[1][i]<0) return(21);
     if(DataGeom->Element[2][i]<1)                            return(22);
     if( !strcmp(strlwr(DataGeom->TypeElement[i]),"xx")&&
         !strcmp(strlwr(DataGeom->TypeElement[i]),"xo")&&
         !strcmp(strlwr(DataGeom->TypeElement[i]),"ox")&&
         !strcmp(strlwr(DataGeom->TypeElement[i]),"oo") )     return(23);
     DataGeom->Element[0][i]--; DataGeom->Element[1][i]--;
     DataGeom->Element[2][i]--;
   }/*for i*/
 for(i=0;i<DataGeom->ElementNumber;i++)
   if(DataGeom->Element[0][i]==DataGeom->Element[1][i]) return(24);
 for(i=0;i<DataGeom->ElementNumber-1;i++)
     for(j=i+1; j<DataGeom->ElementNumber;j++)
        if((DataGeom->Element[0][i]==DataGeom->Element[0][j])&&
           (DataGeom->Element[1][i]==DataGeom->Element[1][j])) return(25);
 /*PRZEKROJE*/
  for(i=0;i<4;i++) fgets(Key,100,File);
  ix=-1; fscanf(File,"Liczba typ%*s%*s%*s %d\n\n",&ix);
  if(ix<1) return(26);
  if(ix>DataGeom->ElementNumber) return(27);
  for(i=0;i<DataGeom->ElementNumber;i++)
    if(DataGeom->Element[2][i]>(ix-1)) return(28);
  DataGeom->SectionNumber=ix;
  for(i=0;i<3;i++) fgets(Key,100,File);
  /*A,Ix,E*/
  if((DataGeom->Section[0]=(double *)farcalloc(ix,DOUBLE))==NULL) return(29);
  if((DataGeom->Section[1]=(double *)farcalloc(ix,DOUBLE))==NULL) return(30);
  if((DataGeom->Section[2]=(double *)farcalloc(ix,DOUBLE))==NULL) return(31);
  for(i=0;i<DataGeom->SectionNumber;i++)
    { fscanf(File,"%*d %lf %lf %lf\n",
               &DataGeom->Section[0][i], &DataGeom->Section[1][i],
               &DataGeom->Section[2][i]);
      DataGeom->Section[2][i]*=100; /*[GPa] na [kN/cm^2]*/
      if( DataGeom->Section[0][i]<DBL_EPSILON ||
```

```
          DataGeom->Section[1][i]<DBL_EPSILON ||
          DataGeom->Section[2][i]<DBL_EPSILON) return(32); }
 /*PODPORY*/
 for(i=0;i<2;i++) fgets(Key,100,File);
 ix=-1; fscanf(File,"Liczba wiez%*s%*s%*s %d\n\n",&ix);
   if(ix<3) return(33);
  DataGeom->BondNumber=ix;
  if((DataGeom->BondNode=(int *)farcalloc(ix,INT))==NULL) return(34); /*W*/
  if((DataGeom->BondDegree=
              (char *)farcalloc(ix,sizeof(char)))==NULL) return(35); /*S*/
  if((DataGeom->BondStif[0]=(double *)farcalloc(ix,DOUBLE))==NULL) return(36);
                                                          /*podatnosc*/
  if((DataGeom->BondStif[1]=(double *)farcalloc(ix,DOUBLE))==NULL) return(37);
                                                       /*blad polozenia*/
  for(i=0;i<3;i++) fgets(Key,100,File);
  for(i=0;i<DataGeom->BondNumber;i++)
  { fscanf(File,"%*d %d %c %lf %lf\n",
              &DataGeom->BondNode[i], &DataGeom->BondDegree[i],
              &DataGeom->BondStif[0][i], &DataGeom->BondStif[1][i]);
    if(DataGeom->BondNode[i]>DataGeom->NodeNumber) return(38);
    if(!strchr("YyXxOo",DataGeom->BondDegree[i]))  return(39);
    if(DataGeom->BondStif[0][i]<0.0)               return(40);
    DataGeom->BondNode[i]--;
    DataGeom->BondStif[0][i]/=10.0;  /*[mm] na [cm]*/
    DataGeom->BondDegree[i]=toupper(DataGeom->BondDegree[i]);
    if(DataGeom->BondDegree[i]=='O') DataGeom->BondStif[1][i]/=100.0;
                                else DataGeom->BondStif[1][i]/=10.0;
                    /*[rad/kNm] na [rad/kNcm]l lub [mm/kN] na [cm/kN]*/
   }/*for i*/

  for(i=0;i<DataGeom->ElementNumber;i++)
   DataGeom->Band=
    max(DataGeom->Band,abs(DataGeom->Element[1][i]-DataGeom->Element[0][i]));
  DataGeom->Band=NF*(DataGeom->Band+1);
  DataGeom->Size=NF*DataGeom->NodeNumber;
return(0);
}/*LoadData*/

/*-------------------------------------------------------------------------*/
/*-  LoadDataForce()   CZYTANIE DANYCH O OBCIAZENIACH                      -*/
/*  Zwrot: 0=Sukces,
          50=Nie ma zadanych obciazen ( zerowa liczba schematow )
          51=Brak pamieci na kombinacje obciazen

          52=Ujemna liczba schemat˘w obciazen skupionych
          53=Brak pamieci na obciazenia skupione Px
          54=Brak pamieci na obciazenia skupione Py
          55=Brak pamieci na obciazenia skupione M
          56=Ujemna liczba punkt˘w obciazen skupionych
          57=Brak pamieci na przynaleznosc obciazen skupionych do schemat˘w
          58=Brak pamieci na typy obciazen skupionych w punktach
          59=Brak pamieci na przylozenie obc. skupionych do element˘w
          60=Brak pamieci na wsp˘lrzedne obciazen skupionych
          61=Zly numer przynaleznego do obciazen skupionych schematu
          62=Zly numer typ obciazen skupionych w punkcie
          63=Zly numer elementu z obciazeniem skupionym
          64=Zly numer wezla z obciazeniem skupionym
          65=Ujemna wspolrzedna polozenia obciazenia skupionego,

          66=Ujemna liczba schemat˘w obciazen rozlozonych
          67=Brak pamieci na obciazenia rozlozone qx1
          68=Brak pamieci na obciazenia rozlozone qy1
          69=Brak pamieci na obciazenia rozlozone qx2
          70=Brak pamieci na obciazenia rozlozone qy2
          71=Ujemna liczba element˘w z obciazeniem rozlozonym
          72=Brak pamieci na schematy obciazen rozlozonych na elementach

          73=Ujemna liczba schemat˘w obciazen termicznych
```

```
            74=Brak pamieci na temperatury dt1
            75=Brak pamieci na temperatury dt2
            76=Brak pamieci na wsp lczynniki rozszerzalnosci termicznej
            77=Brak pamieci na wysokosci przekroj w (obciazenia termiczne )
            78=Ujemna liczba element w z obciazeniem termicznym
            79=Brak pamieci na obciazenia termiczne ba elementach
*/
/*-------------------------------------------------------------------------*/
int LoadDataForce(FILE *File, struct DescriptForce *DataForce,
                  struct DescriptData *DataGeom)
{ int i, j, ix=0;
  int Scheme, Element, Typ;
  /*Kombinacje obciazen*/
  for(i=0;i<2;i++) fgets(Key,100,File);
  ix=-1; fscanf(File,"Liczba schemat%*s%*s%*s %d\n\n",&ix);
    if(ix<=0)  return(50);
  DataForce->SchemeNumber=ix;
  if((DataForce->Scheme=(int *)farcalloc(ix,INT))==NULL) return(51);
  for(i=0;i<3;i++) fgets(Key,100,File);
  for(i=0;i<DataForce->SchemeNumber;i++)
                        fscanf(File,"%*d %d\n", &DataForce->Scheme[i]);
  /*Obciazenie skupione*/
  for(i=0;i<9;i++) fgets(Key,100,File);
  ix=-1; fscanf(File,"Liczba typ%*s%*s%*s%*s %d\n\n",&ix);
  if(ix<0) return(52);
  DataForce->ConcenTypeNumber=ix;
  if(DataForce->ConcenTypeNumber>0)
  { for(i=0;i<3;i++) fgets(Key,100,File);
    if((DataForce->ConcenTypeForce[0]= /*Px*/
              (double *)farcalloc(ix,DOUBLE))==NULL) return(53);
    if((DataForce->ConcenTypeForce[1]= /*Py*/
              (double *)farcalloc(ix,DOUBLE))==NULL) return(54);
    if((DataForce->ConcenTypeForce[2]= /*M*/
              (double *)farcalloc(ix,DOUBLE))==NULL) return(55);
    for(i=0;i<DataForce->ConcenTypeNumber;i++)
     { fscanf(File,"%*d %lf %lf %lf\n",
        &DataForce->ConcenTypeForce[0][i], &DataForce->ConcenTypeForce[1][i],
        &DataForce->ConcenTypeForce[2][i]);
        DataForce->ConcenTypeForce[2][i]*=100.0; /*[m] na [cm]*/
     }
    ix=-1; fscanf(File,"Liczba punkt%*s%*s%*s%*s%*s %d\n\n",&ix);
    if(ix<0) return(56);
    DataForce->ConcenNumber=ix;
    if(DataForce->ConcenNumber>0)
    { for(i=0;i<3;i++) fgets(Key,100,File);
      /*[Scheme, Typ, Element (-Wezel)] [X1]*/
      if((DataForce->ConcenForce[0]=(int *)farcalloc(ix,INT))==NULL) return(57);
      if((DataForce->ConcenForce[1]=(int *)farcalloc(ix,INT))==NULL) return(58);
      if((DataForce->ConcenForce[2]=(int *)farcalloc(ix,INT))==NULL) return(59);
      if((DataForce->ConcenX1=(double *)farcalloc(ix,DOUBLE))==NULL) return(60);
       for(i=0;i<DataForce->ConcenNumber;i++)
       { fscanf(File,"%*d %d %d %d %lf\n",
          &DataForce->ConcenForce[0][i],&DataForce->ConcenForce[1][i],
          &DataForce->ConcenForce[2][i],&DataForce->ConcenX1[i]);
      if(DataForce->ConcenForce[0][i]<1 ||
         DataForce->ConcenForce[0][i]>DataForce->SchemeNumber) return(61);
      if(DataForce->ConcenForce[1][i]<1 ||
         DataForce->ConcenForce[1][i]>DataForce->ConcenTypeNumber) return(62);
      DataForce->ConcenForce[0][i]--; DataForce->ConcenForce[1][i]--;
      if(DataForce->ConcenForce[2][i]>0)
        { if(DataForce->ConcenForce[2][i]>DataGeom->ElementNumber) return(63);
                          else DataForce->ConcenForce[2][i]--;
        }
       else if(-DataForce->ConcenForce[2][i]>DataGeom->NodeNumber) return(64);
      if(DataForce->ConcenX1[i]<0.0) return(65);
         else DataForce->ConcenX1[i]*=100; /* zamiana [m] na [cm] */
     }
    }/*if(ConcenNumber)*/
```

```
 }/*if(ConcenTypeNumber)*/

/*Obciazenia rozlozone*/
for(i=0;i<2;i++) fgets(Key,100,File);
ix=-1; fscanf(File,"Liczba typ%*s%*s%*s%*s %d\n\n",&ix);
if(ix<0) return(66);
DataForce->UniformTypeNumber=ix;
if(DataForce->UniformTypeNumber>0)
{  for(i=0;i<3;i++) fgets(Key,100,File);
   /*[qx1,qy1,qx2,qy2]*/
   if((DataForce->UniformTypeForce[0]=(double *)farcalloc(ix,DOUBLE))==NULL)
            return(67);
   if((DataForce->UniformTypeForce[1]=(double *)farcalloc(ix,DOUBLE))==NULL)
            return(68);
   if((DataForce->UniformTypeForce[2]=(double *)farcalloc(ix,DOUBLE))==NULL)
            return(69);
   if((DataForce->UniformTypeForce[3]=(double *)farcalloc(ix,DOUBLE))==NULL)
            return(70);
   for(i=0;i<DataForce->UniformTypeNumber;i++)
     { fscanf(File,"%*d %lf %lf %lf %lf\n",
              &DataForce->UniformTypeForce[0][i],
              &DataForce->UniformTypeForce[1][i],
              &DataForce->UniformTypeForce[2][i],
              &DataForce->UniformTypeForce[3][i]);
       for(j=0;j<4;j++) DataForce->UniformTypeForce[j][i]/=100;
                                                  /*[kN/m] na [kN/cm]*/
     }
   ix=-1; fscanf(File,"Liczba element%*s%*s%*s%*s%*s %d\n\n",&ix);
   if(ix<0) return(71);
   DataForce->UniformNumber=ix;
   if(DataForce->UniformNumber>0)
   { for(i=0;i<3;i++) fgets(Key,100,File);
     if((DataForce->UniformForce=
       (int**)farcalloc(DataForce->SchemeNumber, sizeof(int*)))==NULL)
                  return(72);
     for(j=0;j<DataForce->SchemeNumber;j++)
      if((DataForce->UniformForce[j]=
         (int *)farcalloc(DataGeom->ElementNumber,INT))==NULL) return(72);
     for(j=0;j<DataForce->UniformNumber;j++)
      { fscanf(File,"%d %d %d\n", &Scheme, &Element, &Typ);
        DataForce->UniformForce[Scheme-1][Element-1]=Typ;
        /*Uwaga: Typ od 1. Nie zmniejszany -> nieobciazony , typ=0 */
      }
   }/*if(UniformNumber)*/
}/*if(UniformTypeNumber)*/

/*Obciazenia termiczne*/
for(i=0;i<2;i++) fgets(Key,100,File);
ix=-1; fscanf(File,"Liczba typ%*s%*s%*s%*s %d\n\n",&ix);
if(ix<0) return(73);
DataForce->ThermalTypeNumber=ix;
if(DataForce->ThermalTypeNumber>0)
{  for(i=0;i<3;i++) fgets(Key,100,File);
    /*[dt1,dt2,at,h]*/
   if((DataForce->ThermalTypeForce[0]=(double *)farcalloc(ix,DOUBLE))==NULL)
            return(74);
   if((DataForce->ThermalTypeForce[1]=(double *)farcalloc(ix,DOUBLE))==NULL)
            return(75);
   if((DataForce->ThermalTypeForce[2]=(double *)farcalloc(ix,DOUBLE))==NULL)
            return(76);
   if((DataForce->ThermalTypeForce[3]=(double *)farcalloc(ix,DOUBLE))==NULL)
            return(77);
   for(i=0;i<DataForce->ThermalTypeNumber;i++)
    { fscanf(File,"%*d %lf %lf %lf %lf\n",
        &DataForce->ThermalTypeForce[0][i],
        &DataForce->ThermalTypeForce[1][i],
        &DataForce->ThermalTypeForce[2][i],
        &DataForce->ThermalTypeForce[3][i]);
```

```
        DataForce->ThermalTypeForce[3][i]/=10;  /* [mm] na [cm] */
        DataForce->ThermalTypeForce[2][i]/=1e6; /* wpisano at=14.0 ->
                                                   at=14/1000000=0.000014*/
      }
     ix=-1; fscanf(File,"Liczba element%*s%*s%*s%*s %d\n\n",&ix);
     if(ix<0) return(78);
     DataForce->ThermalNumber=ix;
     if(DataForce->ThermalNumber>0)
     {  for(i=0;i<3;i++) fgets(Key,100,File);
        if((DataForce->ThermalForce=
         (int**)farcalloc(DataForce->SchemeNumber, sizeof(int*)))==NULL)
                    return(79);
        for(j=0;j<DataForce->SchemeNumber;j++)
         if((DataForce->ThermalForce[j]=
           (int *)farcalloc(DataGeom->ElementNumber,INT))==NULL) return(79);
        for(j=0;j<DataForce->ThermalNumber;j++)
         { fscanf(File,"%d %d %d\n", &Scheme, &Element, &Typ);
           DataForce->ThermalForce[Scheme-1][Element-1]=Typ;
           /*Uwaga: Typ od 1. Nie zmniejszany -> nieobciazony , typ=0 */
         }
     }/*if(ThermalNumber)*/
  }/*if(ThermalTypeNumber)*/

 return(0);
}/*LoadForce*/

/*-------------------------------------------------------------------------*/
/*-  WriteResult()   DRUKOWANIE WYNIKOW                                    -*/
/*-------------------------------------------------------------------------*/
int WriteResult(FILE *File, struct DescriptData *DataGeom,
                double *Displacement, double *Force)

{ int j, i1, j1;

    fprintf(File,"PRZEMIESZCZENIA:\n");
    fprintf(File," Wezel          v[cm]          w[cm]         fi[deg]\n");
     for(j=0;j<DataGeom->NodeNumber;j++)
        fprintf(File,"%4d %15.3f%16.3f%16.3f\n",j+1,
                  Displacement[NF*j],Displacement[NF*j+1],
                  Displacement[NF*j+2]*180/3.142);
    fprintf(File,"\nSILY PRZEKROJOWE:\n");
    fprintf(File,"Element                  Wezel W1->W2
Wezel W2->W1          \n");
    fprintf(File,"   ( W1-W2 )     N[kN]      V[kN]     M[kNm]      N[kN]
V[kN]     M[kNm]\n");
       for(j=0;j<DataGeom->ElementNumber;j++)
          { i1=DataGeom->Element[0][j];  j1=DataGeom->Element[1][j];
           fprintf(File,"%-3d(%3d-%-3d)%10.3f%11.3f%11.3f%11.3f%11.3f%11.3f\n",
                  j+1,i1+1,j1+1,
                  -Force[EF*j],Force[EF*j+1],Force[EF*j+2]/100,Force[EF*j+3],
                  -Force[EF*j+4],-Force[EF*j+5]/100);
          }
 return(0);
}/*WriteResult*/

/*-------------------------------------------------------------------------*/
/* Cosinusy kierunkowe oraz dlugosc Elementu */
int CosDirectElement(int Elem, struct DescriptData *DataGeom, double *Length,
                double *CosSin)
/* Ret=0= Sukces;
       7= Zerowa dlugosc elementu konstrukcji;                           */
/*-------------------------------------------------------------------------*/
{ int      wl=DataGeom->Element[0][Elem],
           wp=DataGeom->Element[1][Elem];
  double   dx=DataGeom->Node[0][wp]-DataGeom->Node[0][wl],
           dy=DataGeom->Node[1][wp]-DataGeom->Node[1][wl];
  double Len;
  if(fabs(Len=sqrt(dx*dx+dy*dy))<DBL_EPSILON) return(7);
```

```
  CosSin[0]=dx/Len; CosSin[1]=dy/Len;
  *Length=Len;
  return(0);
}/*CosDirectElement*/

/*-------------------------------------------------------------------------*/
/*-  EqualForce()   SILY WEZLOWE I TRANSFORMACJA SIL
-*/
/*-------------------------------------------------------------------------*/
int EqualForce(int Scheme, struct DescriptData *DataGeom,
               struct DescriptForce *DataForce, double *Force,
               double *ForceDisplac, double *Displacement)
/* Zwrot=0  Sukces
        7 Zerowa dlugosc lub wysokosc elementu
        8 Do wezla z przegubem przylozono skupiony moment
*/
/*zapisuje obciazenia do wektora przemieszczen Displacement*/
{
  register int m, l, i, j;
  int i1, j1, Elem, Node, Type, Ret=0;
  double qy1,qy2,qx1,qx2, Py, Px, M, dt1, dt2, dh, at, Length, CosSin[2];
  double py1,py2,px1,px2;
  double t[EF]={0}, xL=0, x=0, E=0, EI=0, EA=0, s=0, s1=0;
  int TypPrzekroju; char TypElementu[3];

  memset(Force,0,EF*DataGeom->ElementNumber*DOUBLE);
  memset(Displacement,0,DataGeom->Size*DOUBLE);

 /* OBCIAZENIA ROZLOZONE */
 if(DataForce->UniformTypeNumber>0)
 for(Elem=0;Elem<DataGeom->ElementNumber;Elem++)
  { Type=DataForce->UniformForce[Scheme][Elem]-1;
    if((Type>=0)&&(Type<DataForce->UniformTypeNumber))
    { qy1=DataForce->UniformTypeForce[1][Type];
      qy2=DataForce->UniformTypeForce[3][Type];
      qx1=DataForce->UniformTypeForce[0][Type];
      qx2=DataForce->UniformTypeForce[2][Type];
      if((Ret=CosDirectElement(Elem, DataGeom, &Length, CosSin))>0)
                                                    return(Ret);
      px1=CosSin[0]*qx1+CosSin[1]*qy1; py1=-CosSin[1]*qx1+CosSin[0]*qy1;
      px2=CosSin[0]*qx2+CosSin[1]*qy2; py2=-CosSin[1]*qx2+CosSin[0]*qy2;
      t[0]=-Length/6*(2*px1+px2); t[1]=-Length/20*(7*py1+3*py2);
      t[2]=-Length*Length/60*(3*py1+2*py2);
      t[3]=-Length/6*(2*px2+px1); t[4]=-Length/20*(7*py2+3*py1);
      t[5]=Length*Length/60*(3*py2+2*py1);
      for(j=0;j<EF;j++)  Force[EF*Elem+j]+=t[j];
    }
  }
 /*OBCIAZENIA SKUPIONE*/
 for(i=0;i<DataForce->ConcenNumber;i++)
  {  if(DataForce->ConcenForce[0][i]!=Scheme) continue;
     Type=DataForce->ConcenForce[1][i];
     Px=DataForce->ConcenTypeForce[0][Type];
     Py=DataForce->ConcenTypeForce[1][Type];
     M=DataForce->ConcenTypeForce[2][Type];
     Elem=DataForce->ConcenForce[2][i];
     if(Elem>=0)
       { i1=DataGeom->Element[0][Elem];  j1=DataGeom->Element[1][Elem];
         xL=DataForce->ConcenX1[i];
         if((Ret=CosDirectElement(Elem, DataGeom, &Length, CosSin))>0)
                                                    return(Ret);
         if(fabs(xL)>DBL_EPSILON && fabs(xL-Length)>DBL_EPSILON)
           { /*jesli obc. na elemencie */
             px1=CosSin[0]*Px+CosSin[1]*Py; py1=-CosSin[1]*Px+CosSin[0]*Py;
             for(j=0;j<2;j++)
              { x=xL/Length*(1-2*j)+j;
                t[j*NF+2]=(2*j-1)*py1*Length*(x-2*x*x+x*x*x)+ M*(1-x)*(3*x-1);
                t[j*NF+1]=-py1*(1-3*x*x+2*x*x*x)+ 6*M*(1-2*j)*x*(1-x)/Length;
```

```
                t[j*NF]=-px1*x;
              }
             for(j=0;j<EF;j++)  Force[EF*Elem+j]+=t[j];
            }
          }/*if Elem>0*/
        if(Elem<0 || fabs(xL)<DBL_EPSILON || fabs(xL-Length)<DBL_EPSILON)
         { /*obciazenie w wezle*/
           if(Elem>=0)
            { Node=(fabs(xL)<=DBL_EPSILON)?i1:j1;
              strcpy(TypElementu,DataGeom->TypeElement[Elem]);
              strlwr(TypElementu);
              if((Node==i1)&&(!strcmp(TypElementu,"ox")||
                              !strcmp(TypElementu,"oo"))) Force[EF*Elem+2]-=M;
                else if((Node==j1)&&(!strcmp(TypElementu,"xo")||
                              !strcmp(TypElementu,"oo"))) Force[EF*Elem+5]-=M;
                  else Displacement[NF*Node+2]+=M; /*moment bez przegubu */
            }
           else { Node=-Elem-1;
                  for(Elem=0;Elem<DataGeom->ElementNumber;Elem++)
                  { i1=DataGeom->Element[0][Elem];
                    j1=DataGeom->Element[1][Elem];
                    strcpy(TypElementu,DataGeom->TypeElement[Elem]);
                    strlwr(TypElementu);
                    if((fabs(M)>DBL_EPSILON) &&
                       ( ((Node==i1)&&(!strcmp(TypElementu,"ox")||
                                       !strcmp(TypElementu,"oo")))||
                         ((Node==j1)&&(!strcmp(TypElementu,"xo")||
                                       !strcmp(TypElementu,"oo")))
                       ) )
                    return(8); /*Moment w przegubie*/
                  }
                  Displacement[NF*Node+2]+=M;
                }
             Displacement[NF*Node]+=Px; Displacement[NF*Node+1]+=Py;
          }/*if obciazenie w wezle*/
 }/*for i*/
/*OBCIAZENIE TEMPERATURA*/
if(DataForce->ThermalTypeNumber>0)
for(Elem=0;Elem<DataGeom->ElementNumber;Elem++)
 { Type=DataForce->ThermalForce[Scheme][Elem]-1;
   if((Type>=0)&&(Type<DataForce->ThermalTypeNumber))
    { TypPrzekroju=DataGeom->Element[2][Elem];
      E=DataGeom->Section[2][TypPrzekroju],       /*kN/cm^2 */
      EI=E*DataGeom->Section[1][TypPrzekroju],    /*kNcm^2  */
      EA=E*DataGeom->Section[0][TypPrzekroju];    /*kN      */
      dt1=DataForce->ThermalTypeForce[0][Type];
      dt2=DataForce->ThermalTypeForce[1][Type];
      at=DataForce->ThermalTypeForce[2][Type];
      dh=DataForce->ThermalTypeForce[3][Type];
      t[3]=-(t[0]=EA*at*(dt1+dt2)/2);   t[1]=t[4]=0.0;
       if(dt1==dt2) t[5]=t[2]=0.0;
           else { if(dh<DBL_EPSILON) return(7);
                  t[5]=-(t[2]=EI*at*(dt2-dt1)/dh); }
      for(j=0;j<EF;j++)  Force[EF*Elem+j]+=t[j];
    }
 }/*for Elem*/
/*Modyfikacja rownowaznikow dla Elementow innych niz sztywno-sztywny*/
 for(Elem=0;Elem<DataGeom->ElementNumber;Elem++)
  { strcpy(TypElementu,DataGeom->TypeElement[Elem]); strlwr(TypElementu);
    if(strcmp(TypElementu,"xx"))
    { i=EF*Elem;
      if((Ret=CosDirectElement(Elem, DataGeom, &Length, CosSin))>0)
                                                        return(Ret);
      if(!strcmp(TypElementu,"oo"))
         { x=(Force[i+2]+Force[i+5])/Length; Force[i+2]=Force[i+5]=0.0; }
      else if(!strcmp(TypElementu,"xo"))
             { x=3.0/2.0*Force[i+5]/Length;
               Force[i+2]-=Force[i+5]/2.0; Force[i+5]=0.0; }
```

```
        else if(!strcmp(TypElementu,"ox"))
               { x=3.0/2.0*Force[i+2]/Length;
                 Force[i+5]-=Force[i+2]/2.0; Force[i+2]=0.0; }
        Force[i+1]-=x; Force[i+4]+=x;
      }
    }
  /*modyfikacja wektora obciazen*/
  for(l=0;l<DataGeom->ElementNumber;l++)
    { i1=DataGeom->Element[0][l];  j1=DataGeom->Element[1][l];
      if((Ret=CosDirectElement((int)l, DataGeom, &Length, CosSin))>0)
                                                        return(Ret);
      for(m=0;m<2;m++)
       { i=NF*(i1*(1-m)+j1*m); s=-Force[l*EF+NF*m]; s1=-Force[l*EF+NF*m+1];
         Displacement[i]+=s*CosSin[0]-s1*CosSin[1];
         Displacement[i+1]+=s*CosSin[1]+s1*CosSin[0];
         Displacement[i+2]-=Force[EF*l+NF*m+2];
       }
    }
  for(l=0;l<DataGeom->NodeNumber;l++)
    for(j=0;j<NF;j++)  { i=NF*l+j; Displacement[i]+=ForceDisplac[i]; }
  return(Ret);
}/*EqualForce*/

/*-------------------------------------------------------------------------*/
/* Macierz sztywnosci Elementu w ukladzie lokalnym */
int StifElemLokal( int Elem, struct DescriptData *DataGeom,
                   double Length, double *StifLokal)
/*-------------------------------------------------------------------------*/
{
  int TypPrzekroju=DataGeom->Element[2][Elem];
  double E=   DataGeom->Section[2][TypPrzekroju],      /*kN/cm/cm*/
         EI=E*DataGeom->Section[1][TypPrzekroju],      /*kNcm^2  */
         EA=E*DataGeom->Section[0][TypPrzekroju];      /*kN      */
         /* obliczenia prowadzone w jednostkach kN, cm        */
  char TypElementu[3];
  strlwr(strcpy(TypElementu,DataGeom->TypeElement[Elem]));
  memset(StifLokal,0,DOUBLE*EF*EF);
  StifLokal[EF*0+0]=StifLokal[EF*3+3]=-(StifLokal[EF*0+3]=-EA/Length);
  if(!strcmp(TypElementu,"xx")) /*sztywno-sztywny*/
   { StifLokal[EF*1+1]=StifLokal[EF*4+4]=
                       -(StifLokal[EF*1+4]=-12*EI/Length/Length/Length);
     StifLokal[EF*1+2]=StifLokal[EF*1+5]=
                 -(StifLokal[EF*4+5]=StifLokal[EF*2+4]=-6*EI/Length/Length);
     StifLokal[EF*2+2]=StifLokal[EF*5+5]=4*EI/Length;
     StifLokal[EF*2+5]=2*EI/Length;
   }
  else if(!strcmp(TypElementu,"ox")) /*przegub-sztywny*/
   { StifLokal[EF*1+4]=-3*EI/Length/Length/Length;
     StifLokal[EF*1+1]=StifLokal[EF*4+4]=-StifLokal[EF*1+4];
     StifLokal[EF*1+5]=-(StifLokal[EF*4+5]=-3*EI/Length/Length);
     StifLokal[EF*5+5]=3*EI/Length;
   }
  else if(!strcmp(TypElementu,"xo")) /*sztywno-przegub*/
   { StifLokal[EF*1+1]=StifLokal[EF*4+4]=
                       -(StifLokal[EF*1+4]=-3*EI/Length/Length/Length);
     StifLokal[EF*1+2]=-(StifLokal[EF*2+4]=-3*EI/Length/Length);
     StifLokal[EF*2+2]=3*EI/Length;
   }
   /*pozostale - w tym oo=przegub-przegub*/
  StifLokal[EF*3+0]=StifLokal[EF*0+3];
  StifLokal[EF*2+1]=StifLokal[EF*1+2];  StifLokal[EF*4+1]=StifLokal[EF*1+4];
  StifLokal[EF*5+1]=StifLokal[EF*1+5];
  StifLokal[EF*4+2]=StifLokal[EF*2+4];  StifLokal[EF*5+2]=StifLokal[EF*2+5];
  StifLokal[EF*5+4]=StifLokal[EF*4+5];
  return(0);
}/*StifElemLokal*/

/*-------------------------------------------------------------------------*/
```

```
int StifElemGlobal(int Elem, struct DescriptData *DataGeom, double *StifElement)
/*-------------------------------------------------------------------------*/
{ double  Length=0.0;
  int i, j, t, t1, t2, r, r1, r2, Ret=0;

  if( (Ret=CosDirectElement(Elem, DataGeom, &Length, CosSin))!=0)
                                                    return(Ret);
  StifElemLokal(Elem, DataGeom,  Length, StifLokal);
  for(i=0;i<=2;i++)
  {    t=(i==1)?3:0; r=(i==0)?0:3;
       t1=t+1; t2=t+2;
       r1=r+1; r2=r+2;  /*  CosSin[0]=cosinus; CosSin[1]=sinus  */
   StifElement[EF*t+r] =CosSin[0]*CosSin[0]*StifLokal[EF*t+r]+
                        CosSin[1]*CosSin[1]*StifLokal[EF*t1+r1];
   StifElement[EF*t+r1]=StifElement[EF*t1+r]=
              CosSin[0]*CosSin[1]*(StifLokal[EF*t+r]-StifLokal[EF*t1+r1]);
   StifElement[EF*t+r2]=-CosSin[1]*StifLokal[EF*t1+r2];
   StifElement[EF*t1+r1]=CosSin[1]*CosSin[1]*StifLokal[EF*t+r]+
                        CosSin[0]*CosSin[0]*StifLokal[EF*t1+r1];
   StifElement[EF*t1+r2]=CosSin[0]*StifLokal[EF*t1+r2];
   StifElement[EF*t2+r] =-CosSin[1]*StifLokal[EF*t2+r1];
   StifElement[EF*t2+r1]=CosSin[0]*StifLokal[EF*t2+r1];
   StifElement[EF*t2+r2]=StifLokal[EF*t2+r2];
  }
 for(i=0;i<NF;i++)
     for(j=NF;j<EF;j++)
        StifElement[EF*j+i]=StifElement[EF*i+j];
 return(Ret);
}/*StifElemGlobal*/

/*-------------------------------------------------------------------------*/
/*Agregacja macierzy StifElement do macierzy StifSystem*/
AgregatElement(int Elem, struct DescriptData *DataGeom,
               double *StifElement, double *StifSystem)
/*-------------------------------------------------------------------------*/
{ int wl=DataGeom->Element[0][Elem], wp=DataGeom->Element[1][Elem],
      i=NF*wl, j=NF*(wp-wl),
      r1, r2, t1, t2;
  register int t, m, r, k;

  for(k=0;k<2;k++)
   for(r=0;r<NF;r++)
    { r1=k*NF+r; r2=i+k*j+r;
      for(m=0;m<2;m++)
       for(t=0;t<NF;t++)
        { t1=m*NF+t; t2=i+m*j+t-r2; if(t2<0) continue;
          StifSystem[DataGeom->Band*r2+t2]+=StifElement[EF*r1+t1];
        }
    }
 return(0);
}/*AgregatElement*/

/*************************************************************************/
/*Uwzglednienie Warunkow brzegowych*/
BondSystem(struct DescriptData *DataGeom,
           double *ForceDisplac, double *StifSystem)
/*************************************************************************/
{ register int i, j, j1;
  double NotStif, ErrorSituation;
  int Degree, LocalDegree;
for(i=0;i<DataGeom->BondNumber;i++)
 { LocalDegree=(DataGeom->BondDegree[i]=='X')?0:
               (DataGeom->BondDegree[i]=='Y')?1:2;
   Degree=NF*DataGeom->BondNode[i]+LocalDegree;
   NotStif=DataGeom->BondStif[0][i]; ErrorSituation=DataGeom->BondStif[1][i];
   if(ErrorSituation!=0.0)
    { for(j=0;j<DataGeom->Band;j++)
      { if(((j1=Degree-j-1)>0)&&(j<DataGeom->Band-1))
```

```
        ForceDisplac[j1]-=ErrorSituation*StifSystem[DataGeom->Band*j1+j+1];
        if((j1=Degree+j)<DataGeom->Size)
        ForceDisplac[j1]-=ErrorSituation*StifSystem[DataGeom->Band*Degree+j];
      }
    }
  StifSystem[DataGeom->Band*Degree+0]+=( (NotStif==0.0)?1.0E40:1/NotStif );
 }
for(i=0;i<DataGeom->NodeNumber;i++)
   { j=DataGeom->Band*(NF*i+2);  if(StifSystem[j]==0.0) StifSystem[j]=1.0; }
return(0);
}/*BondSystem*/

/*************************************************************************/
/*Budowa Macierzy Sztywnosci Systemu oraz modyfikacja macierzy*/
int StifnessSystem(struct DescriptData *DataGeom, double *StifSystem,
                   double *ForceDisplac)
/*************************************************************************/
{ register int Elem; int Ret=0;
  for(Elem=0;Elem<DataGeom->ElementNumber;Elem++)
   { Ret=StifElemGlobal(Elem,DataGeom,StifElement);
                                        if(Ret>0) return(Ret);
     AgregatElement(Elem,DataGeom,StifElement,StifSystem);
   }
  BondSystem(DataGeom, ForceDisplac, StifSystem);
  return(Ret);
}/*StifnessSystem*/

#define max(a,b)        (((a) > (b)) ? (a) : (b))
#define min(a,b)        (((a) < (b)) ? (a) : (b))

/*************************************************************************/
/*-  ForwardBand()  Rozwiazywanie ukladu rownan. Krok prosty metody       -*/
/*-                 Gaussa dla macierzy pasmowej Matrix o rozmiarze Free  -*/
/*-                 oraz szerokosci pasma Band.                           -*/
/*- Zwrot: 0 Sukces, -1 Uklad osobliwy                                    -*/
/*************************************************************************/
ForwardBand(double *Matrix, int Free, int Band)
{ register int j, k, i; int ik, jk, Band1=Band-1;
  double C=0.0, *PtrMatrix;
  for(i=1;i<Free;i++)
  {if(fabs(Matrix[i*Band])<DBL_EPSILON) return(-1);
   for(k=0;k<Band1;k++)
    { ik=i+k; jk=max(0,ik-Band1); C=Matrix[i*Band+k];
      for(j=jk;j<i;j++)
       { PtrMatrix=Matrix+j*Band;
         if(fabs(PtrMatrix[0])<DBL_EPSILON) return(-1);
         C-=(PtrMatrix[i-j]*(PtrMatrix[ik-j]/PtrMatrix[0]));
       }/*for j*/
      Matrix[i*Band+k]=C;
    }/*for k*/
  }/*for i*/
  return(0);
}/*ForwardBand*/

/*************************************************************************/
/*-  BackBand()  Krok odwrotny rozwiazywania ukladu rownan z macierza pasm.*/
/*************************************************************************/
void BackBand(double *Matrix, int Free, int Band, double *FreeTerm)
{ register int j, i1,i; int jk, Free1=Free-1;
  double C;
  for(i=0;i<Free1;i++)
   { C=FreeTerm[i]/Matrix[i*Band]; jk=min(Free-i,Band);
     for(j=1;j<jk;j++) FreeTerm[i+j]-=Matrix[i*Band+j]*C;
   }/*for i*/
  for(i1=0;i1<Free;i1++)
   { i=Free-i1-1;
     for(j=1;j<Band;j++)
        { if(i+j-Free+1>0) break;
```

```
          FreeTerm[i]-=Matrix[i*Band+j]*FreeTerm[i+j];
        }
       FreeTerm[i]/=Matrix[i*Band];
     }/*for i1*/
}/*BackBand*/

/*-------------------------------------------------------------------------*/
/*-  Cross()   SILY PRZEKROJOWE                                           -*/
/*-------------------------------------------------------------------------*/
int CrossForce(struct DescriptData *DataGeom, double *Displacement,
               double *Force)
{ register int i, m, l, k, Ret=0;
  int i1,j1;
  double t[6], Length, CosSin[2];
 for(l=0;l<DataGeom->ElementNumber;l++)
 { i1=DataGeom->Element[0][l];  j1=DataGeom->Element[1][l];
   if((Ret=CosDirectElement((int)l, DataGeom, &Length, CosSin))>0) return(Ret);
   for(m=0;m<2;m++)
    { i=NF*(i1*(1-m)+j1*m);
      t[3*m]=Displacement[i]*CosSin[0]+Displacement[i+1]*CosSin[1];
      t[3*m+1]=Displacement[i+1]*CosSin[0]-Displacement[i]*CosSin[1];
      t[3*m+2]=Displacement[i+2];
    }
  StifElemLokal(l, DataGeom,Length, StifLokal);
  for(m=0;m<NF;m++) for(k=NF;k<EF;k++) StifLokal[k*EF+m]=StifLokal[m*EF+k];
  for(m=0;m<EF;m++) for(k=0;k<EF;k++)  Force[EF*l+m]+=StifLokal[EF*m+k]*t[k];
 }
  return(Ret);
}/*CrossForce*/

  struct DescriptData _DataGeom[1];
  struct DescriptForce _DataForce[1];

/*-------------------------------------------------------------------------*/
void main(int argc, char *argv[])
/*-------------------------------------------------------------------------*/
/* Exit=0  Sukces,
        1  Nie mozna otworzyc pliku wynikow (argv[1] lub "LCRAMA.DRK"),
        2  Za duzy system. Brak pamieci do zapamietania macierzy sztywnosci,
        3  Za duzy system. Brak pamieci do zapamietania wektora przemieszczen,
        4  Za duzy system. Brak pamieci do zapamietania wektora przemieszczen,
        5  Za duzy system. Brak pamieci do zapamietania wektora sil,
        7 Zerowa dlugosc elementu,
        8 Do wezla z przegubem przylozono skupiony moment
        9 System konstrukcyjny jest osobliwy
        10 do 40 - bledy w pliku danych ( LoadDataGeom()  )
        >50      - bledy w pliku danych ( LoadDataForce() )
*/
/*-------------------------------------------------------------------------*/
{ double *StifSystem, *ForceDisplac, *Displacement, *Force, ErrorSituation;
  char NameDataFile[81]="LCRAMA.DRK", NameRunFile[81]={0};
  FILE *DataFile, *ErrorFile;
  int i, j, Bond, Ret=0, Color=79, y=12, LocalDegree=0;

  if(argc>1) strcpy(NameDataFile,argv[1]);
  if(argc>2) Color=atoi(argv[2]);
  if(argc>3) y=atoi(argv[3]);
  if(argc>4) strcpy(NameRunFile,argv[4]);
  /*---------      Czytanie danych       ---------*/
  if((DataFile=fopen(NameDataFile,"r+"))==NULL) { Ret=1; goto END_PROG; };
  if(Color) WriteXY( 20, y, Color, "%-41.41s", "      Czytanie danych o
geometrii");
  Ret=LoadDataGeom(DataFile,DataGeom);
  if(Ret>0) goto END_PROG;
  if(Color) WriteXY( 20, y, Color, "%-41.41s", "      Czytanie danych o
obciazeniach");
  Ret=LoadDataForce(DataFile,DataForce,DataGeom);
  if(Ret>0) goto END_PROG;
```

```
  chsize(fileno(DataFile),ftell(DataFile));
  fclose(DataFile);

  /*---------       Rozwiazanie systemu bez obciazen       ---------*/
  if((StifSystem=
     (double *) farcalloc(DataGeom->Size*DataGeom->Band,DOUBLE))==NULL)
                                                   { Ret=2; goto END_PROG; }
  if((ForceDisplac=(double *) farcalloc(DataGeom->Size,DOUBLE))==NULL)
                                                   { Ret=3; goto END_PROG; }
  if((Displacement=(double *) farcalloc(DataGeom->Size,DOUBLE))==NULL)
                                                    { Ret=4; goto END_PROG; }
  if((Force=(double *) farcalloc(EF*DataGeom->ElementNumber,DOUBLE))==NULL)
                                                    { Ret=5; goto END_PROG; }

  if(Color) WriteXY( 20, y, Color, "%-41.41s", "       Budowa macierzy
sztywnosci");
  Ret=StifnessSystem(DataGeom, StifSystem, ForceDisplac);
                                                   if(Ret>0) goto END_PROG;
  if(Color) WriteXY( 20, y, Color, "%-41.41s", "       Rozwiazanie wlasne
systemu");
  if(ForwardBand(StifSystem, DataGeom->Size, DataGeom->Band)==-1)
                                                    { Ret=9; goto END_PROG; }

  /* -------- Rozwiazanie systemu dla zadanych obciazen ------- */
  if((DataFile=fopen(NameDataFile,"a"))==NULL) { Ret=1; goto END_PROG; };
  fprintf(DataFile,"                           WYNIKI\n");

fprintf(DataFile,"ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄ\n");
  if(Color) WriteXY( 20, y, Color-1, " Schemat ");
  for(j=0;j<DataForce->SchemeNumber;j++)
  { fprintf(DataFile,"\nSCHEMAT %d\n-------------\n", j+1);
    if(Color) WriteXY( 29, y, Color-1, "%-3d:   ", j+1);
    if(Color) WriteXY( 34, y, Color, "%-27.27s", " R˘wnowazniki wezlowe");
    Ret=EqualForce(j, DataGeom, DataForce, Force, ForceDisplac, Displacement);
    if(Ret>0) goto END_PROG;
    if(Color) WriteXY( 34, y, Color, "%-27.27s", " Rozwiazanie dla obciazen");
    BackBand(StifSystem, DataGeom->Size, DataGeom->Band, Displacement);
    for(Bond=0;Bond<DataGeom->BondNumber;Bond++)
    { if((ErrorSituation=DataGeom->BondStif[1][Bond])!=0.0)
      LocalDegree=(DataGeom->BondDegree[Bond]=='X')?0:
                  (DataGeom->BondDegree[Bond]=='Y')?1:2;
      Displacement[NF*DataGeom->BondNode[Bond]+LocalDegree]+=ErrorSituation;
    }
    if(Color) WriteXY( 34, y, Color, "%-27.27s", " Sily przekrojowe");
    CrossForce(DataGeom, Displacement, Force);
    if(Color) WriteXY( 34, y, Color, "%-27.27s", " Zapis wynik˘w");
    WriteResult(DataFile, DataGeom, Displacement, Force);
  }/*end loop for Scheme*/

  fprintf(DataFile,"\nUWAGI: 1) Katy obrotu fi dotycza sztywnych koncow
elementow.\n"
                 "            Na koncach elementow zakonczonych przegubem katy
obrotu sa\n"
                 "            nieokreslone niezaleznie od podanych wyzej
wartosci.\n\n");
  fprintf(DataFile,"       2) ZNAKOWANIE OBCIAZEN I PRZEMIESZCZEN:\n"
                 "          Dodatnie zwroty przemieszczen (v,w,fi) oraz
obciazen (Px,Py),(qx,qy) \n"
                 "          w globalnym ukladzie wspolrzednych (X,Y) :\n\n"
                 "                          Y,Py,qy,w       \n"
                 "                          ÄżM,fi      \n"
                 "                          ŔÄÄÄÄ X,Px,qx,v\n\n");
  fprintf(DataFile,"       3) ZNAKOWANIE SIL PRZEKROJOWYCH: \n"
                 "          a) sily osiowe N sa dodatnie jesli sa
rozciagajace,\n"
                 "          b) sily poprzeczne V oraz momenty zginajace M sa
dodatnie\n"
```

```
                "                    jesli powoduja rozciaganie wlokien polozonych
po stronie\n");
  fprintf(DataFile,              "                    dodatniej lokalnego
prawoskretnego ukladu wspolrzednych:\n\n"
                "                                   V,M (+)       \n"
                "                                   ł             \n"
                "                      Wezel 1  ŔÄÄÄÄ Wezel 2\n"
                "                    Polozenie wezla 1 oraz wezla 2 elementow
okreslono przy\n"
                "                    zadawaniu geometrii.\n");
END_PROG:
  fclose(DataFile);
  if(Ret<10&&Ret!=1)
    { /*dane o geometrii*/
      for(i=0;i<2;i++) farfree(DataGeom->Node[i]);
      for(i=0;i<3;i++) farfree(DataGeom->Element[i]);
      for(i=0;i<DataGeom->ElementNumber;i++) farfree(DataGeom->TypeElement[i]);
      farfree(DataGeom->TypeElement);
      for(i=0;i<3;i++) farfree(DataGeom->Section[i]);
      farfree(DataGeom->BondNode);    farfree(DataGeom->BondDegree);
      farfree(DataGeom->BondStif[0]); farfree(DataGeom->BondStif[1]);
      /*dane o obciazeniach*/
      farfree(DataForce->Scheme);
      if(DataForce->ConcenTypeNumber>0)
      { for(i=0;i<3;i++) farfree(DataForce->ConcenTypeForce[i]);
        if(DataForce->ConcenNumber>0)
         { for(i=0;i<3;i++) farfree(DataForce->ConcenForce[i]);
           farfree(DataForce->ConcenX1);
         }
       }
      if(DataForce->UniformTypeNumber>0)
       { for(i=0;i<4;i++) farfree(DataForce->UniformTypeForce[i]);
         if(DataForce->UniformNumber>0)
         { for(j=0;j<DataForce->UniformNumber;j++)
                      farfree(DataForce->UniformForce[j]);
           farfree(DataForce->UniformForce);
         }
       }
      if(DataForce->ThermalTypeNumber>0)
       { for(i=0;i<4;i++) farfree(DataForce->ThermalTypeForce[i]);
         if(DataForce->ThermalNumber>0)
         { for(j=0;j<DataForce->ThermalNumber;j++)
                      farfree(DataForce->ThermalForce[j]);
           farfree(DataForce->ThermalForce);
         }
       }
     /*rozwiazanie*/
     farfree(StifSystem);   farfree(ForceDisplac);
     farfree(Displacement); farfree(Force);
    }/*if(Ret==0)*/
  else
    { /*Bledy danych geometrii*/
      if(Ret>12) farfree(DataGeom->Node[0]);
      if(Ret>13) farfree(DataGeom->Node[1]);
      if(Ret>16) farfree(DataGeom->Element[0]);
      if(Ret>17) farfree(DataGeom->Element[1]);
      if(Ret>18) farfree(DataGeom->Element[2]);
      if(Ret>19) { for(i=0;i<DataGeom->ElementNumber;i++)
                           farfree(DataGeom->TypeElement[i]);
                   farfree(DataGeom->TypeElement);
                 }
      if(Ret>29) farfree(DataGeom->Section[0]);
      if(Ret>30) farfree(DataGeom->Section[1]);
      if(Ret>31) farfree(DataGeom->Section[2]);
      if(Ret>34) farfree(DataGeom->BondNode);
      if(Ret>35) farfree(DataGeom->BondDegree);
      if(Ret>36) farfree(DataGeom->BondStif[0]);
      if(Ret>37) farfree(DataGeom->BondStif[1]);
```

```
      /*Bledy danych o obciazeniach*/
      if(Ret>51) farfree(DataForce->Scheme);
      /*Obciazenie skupione*/
      if(DataForce->ConcenTypeNumber>0)
      { if(Ret>53) farfree(DataForce->ConcenTypeForce[0]);
        if(Ret>54) farfree(DataForce->ConcenTypeForce[1]);
        if(Ret>55) farfree(DataForce->ConcenTypeForce[2]);
        if(DataForce->ConcenNumber>0)
         { if(Ret>57) farfree(DataForce->ConcenForce[0]);
           if(Ret>58) farfree(DataForce->ConcenForce[1]);
           if(Ret>59) farfree(DataForce->ConcenForce[2]);
           if(Ret>60) farfree(DataForce->ConcenX1);
         }/*if(ConcenNumber)*/
       }/*if(ConcenTypeNumber)*/
      /*Obciazenia rozlozone*/
      if(DataForce->UniformTypeNumber>0)
      { if(Ret>67) farfree(DataForce->UniformTypeForce[0]);
        if(Ret>68) farfree(DataForce->UniformTypeForce[1]);
        if(Ret>69) farfree(DataForce->UniformTypeForce[2]);
        if(Ret>70) farfree(DataForce->UniformTypeForce[3]);
        if(DataForce->UniformNumber>0)
        { if(Ret>72)
           { for(j=0;j<DataForce->UniformNumber;j++)
                              farfree(DataForce->UniformForce[j]);
             farfree(DataForce->UniformForce);
           }
        }/*if(UniformNumber)*/
       }/*if(UniformTypeNumber/)*/
      /*Obciazenia termiczne*/
      if(DataForce->ThermalTypeNumber>0)
      { if(Ret>74) farfree(DataForce->ThermalTypeForce[0]);
        if(Ret>75) farfree(DataForce->ThermalTypeForce[1]);
        if(Ret>76) farfree(DataForce->ThermalTypeForce[2]);
        if(Ret>77) farfree(DataForce->ThermalTypeForce[3]);
        if(DataForce->ThermalNumber>0)
        { if(Ret>79)
           { for(j=0;j<DataForce->ThermalNumber;j++)
                              farfree(DataForce->ThermalForce[j]);
             farfree(DataForce->ThermalForce);
           }
        }/*if(ThermalNumber)*/
       }/*if(ThermalTypeNumber/)*/
   }/*else*/
  if(Ret>0)
  { DataFile=fopen(NameDataFile,"a");
    fprintf(DataFile,"\n\nObliczenia zako¤czono NIEPOMY□LNIE:");
    if(Color) WriteXY( 20, y, Color, "  BŚŹD:      ");
    if((ErrorFile=fopen("LCRAMA.ERR","r"))!=NULL)
    { j=1; do { if(!feof(ErrorFile)) fgets(Key,100,ErrorFile);
                else { j=0; break; }
              } while(atoi(Key)!=Ret);
      if(j) fprintf(DataFile,"\nBŚŹD %s",Key);
    }
   if(ErrorFile==NULL || j==0)  fprintf(DataFile,"\nBLAD %d",Ret);
   fclose(DataFile); fclose(ErrorFile);
  }
  else
  { /*Wyznaczanie obwiedni*/
    if(Color) WriteXY( 20, y, Color, " SUKCES:    ");
    fclose(DataFile);
  }

  if(Color) WriteXY( 31, y, Color+128, "%-30.30s", "Naciżnij dowolny klawisz");
  getch();
  if(*NameRunFile!=0)
      execl(NameRunFile, NameRunFile, "4", "1", "1", "1", NULL);
/* argv[]=["Panel","N","Opcja1","Opcja2",...,"OpcjaN","Run"] */
exit(Ret);
```

```
}/*main*/
```