

ПРИЛОЖЕНИЕ

Программа PST написана на языке Object Pascal с помощью программной среды Borland Delphi 6.

Программа состоит из следующих модулей:

1. Constants.pas

Здесь определяются основные величины и функции, используемые в программе. Так, здесь определяются координаты атомов в элементарной ячейке, коэффициенты приближений Мольер и Дойля-Тёрнера, ядерный и электронный логарифмы, а также следующие функции: потенциал и его производные, ядерная и электронная плотности и коэффициенты диффузии.

2. Trajectories.pas

Главный модуль программы. Здесь задаются параметры каналирующей частицы и кристалла, выбирается приближение для потенциала изолированного атома и форм-фактора электронной плотности, задаются число учитываемых членов разложения в ряд Фурье, число шагов численного решения системы дифференциальных уравнений движения.

3. AddTrjDialog.pas

Окно для задания точек и углов влёта частиц в кристалл. Также можно задать вид распределения частиц по углам влёта – равномерное или нормальное. По точкам влёта реализуется только равномерное распределение.

4. RandomGauss.pas

Создаёт случайное нормальное распределение частиц по углам влёта. С помощью алгоритма, реализованного в этом модуле, находятся значения начальных углов. Плотность распределения получаемых значений имеет нормальный вид. Каждое, найденное таким образом, значение угла влёта сопоставляется со значением точки влёта случайным образом. При этом мы получаем равномерное распределение по точкам влёта и нормальное распределение по углам влёта.

5. MonteCarloDlg.pas

Форма для задания условий розыгрыша траекторий.

6. Solver.pas

Решатель траекторий. Алгоритм решения основан на методе Рунге-Кутты 4 порядка для численного решения систем дифференциальных уравнений. На каждом шаге решения кроме нахождения новых значений динамических переменных, проверяются условия розыгрыша, в случае выполнения которых выполняется розыгрыш.

7. GraphU.pas

Окно для построения графиков потенциала, его производных, электронной и ядерной плотностей и коэффициентов диффузии.

8. ChartXVT.pas

Окно графического представления результатов расчёта траекторий. Здесь можно просмотреть графики траекторий, поперечных скоростей, фазовых траекторий, вторых моментов и потерь энергии.

9. RaspredV.pas

Модуль для расчёта пространственных и угловых распределений.

10. RaspredE.pas

Модуль расчёта потерь энергии на электронах. Здесь также можно построить спектры потерь энергии и рассчитать деканализирование, связанное с потерями энергии.

11. Dechanneling.pas

Модуль расчёта деканализирования по модели критического приближения.

12. OrientZav.pas

Модуль для построения ориентационных зависимостей.

```

unit Constants;

interface

uses Math;

type
  En_arr_Type=array of array of double;

const
  e2=14.4;    // заряд электрона^2
  a0=0.529;   // радиус Бора
  m0c2=931.4943222E+6; // энергия, экв. массе покоя нуклона
  mec2=0.51099906E+6;  // энергия, экв. массе покоя электрона

  alpha: array [1..3] of double=(0.35,0.55,0.10); // коэф.Мольер для любого кр.

var
  nxRange,
  Z1, Z2: integer; // Зарядовые числа иона и атома кристалла
  Znak_Z1: integer;
  PhiEpsilon, Theta_kl,
  Pi4Z1Z2e2d2, DvaPi, DvaPiax,
  d_small, d3, a_small, a_st,
  sigma, sigma_st, sigma_st2,
  Vmax, Umin, XUmin,
  ax, ay, az,
  EnergyT,
  MassOfIon, // MassOfIon * m0c2 - масса налетающего иона в эВ
  epsilon,
  MA_max, MC_max, //Пороговые значения вторых моментов
  Tmin_MonteCarlo, Tmax_MonteCarlo,
  le, le_d, lnucl, lnucl_d, betaE2, gammaE, deltaE, IE,
  NE, hwp,
  CoefForMC,
  Curv, // Постоянная кривизна
  kt, // Переменная кривизна
  PlDec, sigma_dec, // Для деканализирования по глубине
  glubina, dep_Start,
  DeltaTheta: double;
  k, l, ntr: integer;
  Energy_arr: En_arr_Type;
  Energy_Start, Energy_Last: array of double;
  Ploskost: String;
  Crystal: String;
  Approximation: String;
  SolveSys5DE, SolveMonteCarlo, NoExceedMmax, CombineRozigrish, SolveME,
  LnuclGemmell, LoadedTrj, UxxForSolver0, UxxForSolver3D,
  PeriodPoX, AutoSolve: Boolean;
  NormV: Boolean;
  x_stj, y_stj, z_stj: array [1..8] of double;
  xdax, kyday_lzdaz: array [1..8] of double;
  VxU, VklU, VxUx, VklUx, VxUxx, VklUxx, Fgx2exp,
  expSgx2, Pi2nx: array of double;
  Sba: array [1..3] of double;
  alpha_UDT, beta_UDT, alpha_roeDT, beta_roeDT: array [1..4] of double;
  c_roeDT: double; // Коэф-ты. Дойля-Тернера
  NumOfLayers, FirstLayer: integer;
  NameOfFile, NameOfFileGrOut: String;
  AutoClose, SaveGrOut, Archiving: Boolean;
  Settings: array [1..93] of String; // Настройки
  StartTime, WorkTime: TDateTime;

Procedure InitConstants;
Procedure InitCrystal;
Procedure InitPloskost;
Procedure InitDoyle;
Procedure FinalizeConst;
function gx(nx:integer):double;
function gx2(nx:integer):double;
function g2_kl(nx:integer):double;

```

```

function V(g2:double):double;
function V_Molier(g2:double):double;
function V_DT(g2:double):double;
function F(g2:double):double;
function F_Molier(g2:double):double;
function F_DT(g2:double):double;
function U(x_st:double):double;
function U_x(x_st:double):double;
function U_xx(x_st:double):double;
function UxxForSolver(x_st,T_st:double):double;
function Usp(x_st,T_st:double):double;
function Usp_x(x_st,T_st:double):double;
function Usp_xx(x_st,T_st:double):double;
function ro_e(x_st:double):double;
function ro_nucl(x_st:double):double;
function Diffuz_e(x:double):double;
function Diffuz_nucl(x:double):double;
function Diffuz(x:double):double;

implementation

Procedure InitConstants;
var
  Massa_1, Massa_2: double;
  Phi_kl: double;
begin
  Theta_kl:=-ArcTan(1/k*ay/az);
  Phi_kl:=k*d_small/ay*sin(Theta_kl+DeltaTheta)+
    1*d_small/az*cos(Theta_kl+DeltaTheta);
  PhiEpsilon:=Phi_kl/epsilon;

{Электронный коэффициент диффузии:}
  NE:=8/(d_small*d_small*d_small);
  betaE2:=1-1/Sqr((EnergyT/MassOfIon)+1); // beta^2, beta=(v/c)
  gammaE:=1/Sqrt(1-betaE2); // gamma = (1-beta^2)^(-1/2)
  IE:=11.5*Z2;
  hwp:=Sqrt(2*DvaPi*Z2*NE*e2*Sqr(1972.8)/mec2);
  deltaE:=2*ln(hwp*Sqr(betaE2)*gammaE/IE)-1;
  Le:=ln(2*mec2*betaE2*sqr(gammaE)/IE)-betaE2; //-deltaE/2;
  Le_d:=Le*DvaPi*Sqr(e2*Z1);

{Ядерный коэффициент диффузии:}
  if Lnuc1Gemmell=False then
    begin
      // Lnuc1:=ln(EnergyT*sigma_st/1972.8); <-- по Базылеву: не используется
      Lnuc1:=2*ln(183/Power(Z2,1/3));
    end
  else
    begin
      Massa_1:=MassOfIon;
      if Crystal='Ge' then Massa_2:=72.59*m0c2
      else if Crystal='C' then Massa_2:=12.01*m0c2
      else Massa_2:=28.085*m0c2;
      Lnuc1:=ln(1.29*a_small*EnergyT*Massa_2/(Z1*Z2*(Massa_1+Massa_2)*e2));
    end;
    Lnuc1_d:=Lnuc1*DvaPi*Sqr(e2*Z1*Z2);
    CoefForMC:=epsilon*d_small/Sqr(Vmax); {Коэффициент для расчёта MC в Solver}
  end;

Procedure InitCrystal;
begin
  if Crystal='Ge' then
    begin
      Z2:=32;
      sigma:=0.085;
      d_small:=5.657;
    end
  else if Crystal='C' then
    begin
      Z2:=6;
      sigma:=0.04;
    end
  end;
end;

```

```

    d_small:=3.567;
end
else
begin
    Z2:=14;
    sigma:=0.075;
    d_small:=5.431;
end;
d3:=d_small*d_small*d_small;
end;

Procedure InitPloskost;
var
i, nx: integer;
begin
    Umin:=0;
    DvaPi:=2*Pi;
    Pi4Z1Z2e2d2:=2*DvaPi*Z1*Z2*e2;
    a_small:=0.88534*a0/Sqrt(Power(Z1,(2/3))+ Power(Z2,(2/3))+);
    a_st:=a_small;
    sigma_st:=sigma;
    sigma_st2:=Sqr(sigma_st)/2;

    Sba[1]:=Sqr(0.3/a_st);    // Sba[i]:=Sqr(beta[i]/a_st);
    Sba[2]:=Sqr(1.2/a_st);    // beta: array [1..3] of double=(0.3,1.2,6.0);
    Sba[3]:=Sqr(6.0/a_st);

if Ploskost='(110)' then
begin
    x_stj[1]:=0; x_stj[2]:=sqrt(2)/4; x_stj[3]:=sqrt(2)/4;
    x_stj[4]:=sqrt(2)/2; x_stj[5]:=sqrt(2)/4; x_stj[6]:=sqrt(2)/2;
    x_stj[7]:=sqrt(2)/2; x_stj[8]:=3*sqrt(2)/4;

    y_stj[1]:=0; y_stj[2]:=sqrt(2)/4; y_stj[3]:=-sqrt(2)/4; y_stj[4]:=0;
    y_stj[5]:=0; y_stj[6]:=sqrt(2)/4; y_stj[7]:=-sqrt(2)/4; y_stj[8]:=0;

    z_stj[1]:=0; z_stj[2]:=1/2; z_stj[3]:=1/2; z_stj[4]:=0; z_stj[5]:=1/4;
    z_stj[6]:=3/4; z_stj[7]:=3/4; z_stj[8]:=1/4;

    ax:=d_small/sqrt(2);
    ay:=d_small/sqrt(2);
    az:=d_small;
    XUmin:=0.25;    {Значение X, при котором U(X)=Umin}
    sigma_dec:=0.075/ax;
    PlDec:=0.5;
end
else if Ploskost='(111)' then
begin
    x_stj[1]:=0; x_stj[2]:=1/sqrt(3); x_stj[3]:=1/sqrt(3);
    x_stj[4]:=1/sqrt(3); x_stj[5]:=3/(4*sqrt(3)); x_stj[6]:=7/(4*sqrt(3));
    x_stj[7]:=7/(4*sqrt(3)); x_stj[8]:=7/(4*sqrt(3));

    y_stj[1]:=0; y_stj[2]:=sqrt(2)/4; y_stj[3]:=-sqrt(2)/4; y_stj[4]:=0;
    y_stj[5]:=0; y_stj[6]:=sqrt(2)/4; y_stj[7]:=-sqrt(2)/4; y_stj[8]:=0;

    z_stj[1]:=0; z_stj[2]:=sqrt(2)/(4*sqrt(3));
    z_stj[3]:=sqrt(2)/(4*sqrt(3)); z_stj[4]:=-1/(sqrt(2)*sqrt(3));
    z_stj[5]:=0; z_stj[6]:=sqrt(2)/(4*sqrt(3));
    z_stj[7]:=sqrt(2)/(4*sqrt(3)); z_stj[8]:=-sqrt(2)/(2*sqrt(3));

    ax:=d_small/sqrt(3);
    ay:=d_small/sqrt(2);
    az:=d_small/(sqrt(3)/sqrt(2));
    XUmin:=0.375;
    sigma_dec:=0.075/ax;
    PlDec:=1;
end
else // if Ploskost='(100)' then
begin
    x_stj[1]:=0; x_stj[2]:=0; x_stj[3]:=1/2; x_stj[4]:=1/2;
    x_stj[5]:=1/4; x_stj[6]:=1/4; x_stj[7]:=3/4; x_stj[8]:=3/4;

```

```

y_stj[1]:=0; y_stj[2]:=1/2; y_stj[3]:=0; y_stj[4]:=1/2;
y_stj[5]:=1/4; y_stj[6]:=3/4; y_stj[7]:=1/4; y_stj[8]:=3/4;

z_stj[1]:=0; z_stj[2]:=1/2; z_stj[3]:=1/2; z_stj[4]:=0;
z_stj[5]:=1/4; z_stj[6]:=3/4; z_stj[7]:=3/4; z_stj[8]:=1/4;

ax:=d_small;
ay:=d_small;
az:=d_small;
XUmin:=0.125;
sigma_dec:=sigma/ax;
PlDec:=0.25;
end;

for i:=1 to 8 do
begin
xdax[i]:=x_stj[i]*d_small/ax;
kyday_lzdaz[i]:=k*y_stj[i]*d_small/ay+l*z_stj[i]*d_small/az;
end;

Finalize(Pi2nx); SetLength(Pi2nx, 2*nxRange+1);
Finalize(expSgx2); SetLength(expSgx2, 2*nxRange+1);
Finalize(VxU); SetLength(VxU, 2*nxRange+1);
Finalize(VklU); SetLength(VklU, 2*nxRange+1);
Finalize(VxUx); SetLength(VxUx, 2*nxRange+1);
Finalize(VklUx); SetLength(VklUx, 2*nxRange+1);
Finalize(VxUxx); SetLength(VxUxx, 2*nxRange+1);
Finalize(VklUxx); SetLength(VklUxx, 2*nxRange+1);
Finalize(Fgx2exp); SetLength(Fgx2exp, 2*nxRange+1);

For i:=0 to (2*nxRange) do
begin
nx:=i-nxRange;
expSgx2[i]:=exp(-sigma_st2*gx2(nx));
Pi2nx[i]:=DvaPi*nx;
VxU[i]:=V(gx2(nx))*expSgx2[i];
VklU[i]:=V(g2_kl(nx))*exp(-sigma_st2*g2_kl(nx));
VxUx[i]:=Pi2nx[i]*VxU[i];
VklUx[i]:=Pi2nx[i]*VklU[i];
VxUxx[i]:=Sqr(Pi2nx[i])*VxU[i];
VklUxx[i]:=Sqr(Pi2nx[i])*VklU[i];
Fgx2exp[i]:=F(gx2(nx))*expSgx2[i];
end;

Umin:=U(XUmin);
Vmax:=U(0);
end;

Procedure InitDoyle;
begin
if Crystal='Ge' then
begin
alpha_UDT[1]:=2.4467; alpha_UDT[2]:=2.7015; alpha_UDT[3]:=1.6157;
alpha_UDT[4]:=0.6009;
beta_UDT[1]:=55.893; beta_UDT[2]:=14.393; beta_UDT[3]:=2.4461;
beta_UDT[4]:=0.3415;
alpha_roeDT[1]:=10.0816; alpha_roeDT[2]:=6.3747; alpha_roeDT[3]:=3.7068;
alpha_roeDT[4]:=3.6830;
beta_roeDT[1]:=2.8509; beta_roeDT[2]:=0.2516; beta_roeDT[3]:=11.4468;
beta_roeDT[4]:=54.7625;
c_roeDT:=2.1313;
end
else if Crystal='C' then
begin
alpha_UDT[1]:=0.7307; alpha_UDT[2]:=1.1951; alpha_UDT[3]:=0.4563;
alpha_UDT[4]:=0.1247;
beta_UDT[1]:=36.9951; beta_UDT[2]:=11.2966; beta_UDT[3]:=2.8139;
beta_UDT[4]:=0.3456;
alpha_roeDT[1]:=2.31; alpha_roeDT[2]:=1.02; alpha_roeDT[3]:=1.5886;
alpha_roeDT[4]:=0.865;

```

```

    beta_roeDT[1]:=20.8439; beta_roeDT[2]:=10.2075; beta_roeDT[3]:=0.5687;
    beta_roeDT[4]:=51.6512;
    c_roeDT:=0.2156;
end
else
begin
    alpha_UDT[1]:=2.1293; alpha_UDT[2]:=2.5333; alpha_UDT[3]:=0.8349;
    alpha_UDT[4]:=0.3216;
    beta_UDT[1]:=57.7748; beta_UDT[2]:=16.4756; beta_UDT[3]:=2.8796;
    beta_UDT[4]:=0.3860;
    alpha_roeDT[1]:=6.2915; alpha_roeDT[2]:=3.0353; alpha_roeDT[3]:=1.9891;
    alpha_roeDT[4]:=1.5410;
    beta_roeDT[1]:=2.4386; beta_roeDT[2]:=32.3337; beta_roeDT[3]:=0.6785;
    beta_roeDT[4]:=81.6937;
    c_roeDT:=1.1407;
end;
end;

Procedure FinalizeConst;
begin
    Finalize(Pi2nx);
    Finalize(expSgx2);
    Finalize(VxU);
    Finalize(VklU);
    Finalize(VxUx);
    Finalize(VklUx);
    Finalize(VxUxx);
    Finalize(VklUxx);
    Finalize(Fgx2exp);
end;

function gx(nx:integer):double;
begin
    gx:=nx*DvaPi/ax;
end;

function gx2(nx:integer):double;
begin
    gx2:=Sqr(nx*DvaPi/ax);
end;

function g2_kl(nx:integer):double;
begin
    g2_kl:=Sqr(DvaPi)*(Sqr(nx/ax)+Sqr(k/ay)+Sqr(1/az));
end;

// Приближение Мольер:

function V_Molier(g2:double):double;
var
    sum: double;
    i: integer;
begin
    sum:=0;
    for i:=1 to 3 do
        sum:=sum+alpha[i]/(Sba[i]+g2);
    end;
    V_Molier:=Pi4Z1Z2e2d2*sum;
end;

function F_Molier(g2:double):double;
var
    sum: double;
    i: integer;
begin
    sum:=0;
    for i:=1 to 3 do
        sum:=sum+alpha[i]*Sba[i]/(Sba[i]+g2);
    end;
    F_Molier:=Z2*sum;
end;

// Приближение Дойля-Тернера:

```

```

function V_DT(g2:double):double;
var
  sum: double;
  i: integer;
begin
  sum:=0;
  for i:=1 to 4 do
    sum:=sum+alpha_UDT[i]*exp(-beta_UDT[i]/Sqr(DvaPi)*g2/4);
  V_DT:=DvaPi*Z1*e2*a0*sum;
end;

function F_DT(g2:double):double;
var
  sum: double;
  i: integer;
begin
  sum:=0;
  for i:=1 to 4 do
    sum:=sum+alpha_roeDT[i]*exp(-beta_roeDT[i]*g2/Sqr(4*Pi));
  F_DT:=sum+c_roeDT;
end;

function V(g2:double):double;
begin
  if Approximation='Doil-Terner'
  then V:=V_DT(g2)
  else V:=V_Molier(g2);
end;

function F(g2:double):double;
begin
  if Approximation='Doil-Terner'
  then F:=F_DT(g2)
  else F:=F_Molier(g2);
end;

function U(x_st:double):double; {Двухмерный потенциал U(x)}
var
  sum: double;
  n, nMax, j: integer;
begin
  nMax:=2*nxRange; {nMax -> nx=+nxRange}
  sum:=0;
  for j:=1 to 8 do
  //   for nx:=(-nxRange) to nxRange do
  //     sum:=sum+V(gx2(nx))*exp(-sigma_st2*gx2(nx))
  //     *cos(2*Pi*nx*(x_st-x_stj[j]*d_small/ax));
  for n:=0 to nMax do
    sum:=sum+VxU[n]*cos(Pi2nx[n]*(x_st-xdax[j]));

  U:=Znak_Z1*sum/d3-Umin;
end;

function U_x(x_st:double):double; {Первая производная от U(x)}
var
  sum: double;
  n, nMax, j: integer;
begin
  nMax:=2*nxRange; {nMax -> nx=+nxRange}
  sum:=0;
  for j:=1 to 8 do
  //   for nx:=(-nxRange) to nxRange do
  //     sum:=sum-2*Pi*nx*V(gx2(nx))*exp(-sigma_st2*gx2(nx))
  //     *sin(2*Pi*nx*(x_st-x_stj[j]*d_small/ax));
  for n:=0 to nMax do
    sum:=sum-VxUx[n]*sin(Pi2nx[n]*(x_st-xdax[j]));

  U_x:=Znak_Z1*sum/d3;
end;

```



```

function U_xx(x_st:double):double;    {Вторая производная от U(x)}
var
    sum: double;
    n, nMin, nMax, j: integer;
begin
    nMin:=nxRange+1;    {nMin -> nx=1}
    nMax:=2*nxRange;    {nMax -> nx=+nxRange}
    sum:=0;
    for j:=1 to 8 do
//      for nx:=(-nxRange) to nxRange do
//          sum:=sum-Sqr(2*Pi*nx)*V(gx2(nx))*exp(-sigma_st2*gx2(nx))
//              *cos(2*Pi*nx*(x_st-x_stj[j]*d_small/ax));
        for n:=nMin to nMax do // n:=nx+nxRange
            sum:=sum-VxUxx[n]*cos(Pi2nx[n]*(x_st-xdax[j]));

    sum:=2*sum; // nx=0, Sum_j(*[0])=0*Vgx2exp*cos(0)=0

    U_xx:=Znak_Z1*sum/d3;
end;

function UxxForSolver(x_st,T_st:double):double;    //Вторая производная
begin                                     //для расчёта вторых моментов
    if UxxForSolver0=True
    then UxxForSolver:=0
    else if UxxForSolver3D=False
        then UxxForSolver:=U_xx(x_st)
        else UxxForSolver:=Usp_xx(x_st,T_st);
end;

function Usp(x_st,T_st:double):double;    {Трёхмерный потенциал U(x,t)}
var
    sum: double;
    nx, n, j: integer;
begin
    sum:=0;

    for j:=1 to 8 do
        for nx:=(-nxRange) to nxRange do
            begin
                n:=nx+nxRange;
                sum:=sum+VxU[n]*cos(Pi2nx[n]*(x_st-xdax[j]))
                    +VklU[n]*cos(DvaPi*(nx*(x_st-xdax[j])
                        +PhiEpsilon*T_st
                        -kyday_lzdaz[j]));
            end;

    Usp:=Znak_Z1*sum/d3 - Umin;
end;

function Usp_x(x_st,T_st:double):double;    {Первая производная от U(x,t)}
var
    sum, xj: double;
    nx, n, j: integer;
begin
    sum:=0;

    for j:=1 to 8 do begin
        xj:=x_st-xdax[j];
        for nx:=(-nxRange) to nxRange do
            begin
                n:=nx+nxRange;
                sum:=sum-VxUx[n]*sin(Pi2nx[n]*xj)
                    -VklUx[n]*sin(DvaPi*(nx*xj
                        +PhiEpsilon*T_st
                        -kyday_lzdaz[j]));
            end;
        end;

    Usp_x:=Znak_Z1*sum/d3;
end;

```

```

function Usp_xx(x_st,T_st:double):double;    {Вторая производная от U(x,t)}
var
    sum: double;
    nx, n, j: integer;
begin
    sum:=0;

    for j:=1 to 8 do
        for nx:=(-nxRange) to nxRange do
            begin
                n:=nx+nxRange;
                sum:=sum-VxUxx[n]* cos(Pi2nx[n]*(x_st-xdax[j]))
                    -VklUxx[n]*cos(DvaPi*(nx*(x_st-xdax[j])
                        +PhiEpsilon*T_st
                        -kyday_lzdaz[j]));
            end;

        Usp_xx:=Znak_Z1*sum/d3;
    end;

function ro_e(x_st:double):double;
var
    sum: double;
    n, nMin, nMax, j: integer;
begin
    nMin:=nxRange+1;    {nMin -> nx=1}
    nMax:=2*nxRange;    {nMax -> nx=+nxRange}

    sum:=0;
    for j:=1 to 8 do
        for n:=nMin to nMax do
            sum:=sum+Fgx2exp[n]*cos(Pi2nx[n]*(x_st-xdax[j]));
        sum:=2*sum;
        sum:=sum+Fgx2exp[nxRange]*8;    // n=nxRange -> nx=0, Sum_j(cos(0))=8*1=8
    ro_e:=sum/d3;
end;

function ro_nucl(x_st:double):double;
var
    sum: double;
    n, nMin, nMax, j: integer;
begin
    nMin:=nxRange+1;    {nMin -> nx=1}
    nMax:=2*nxRange;    {nMax -> nx=+nxRange}
    sum:=0;
    for j:=1 to 8 do
        for n:=nMin to nMax do
            sum:=sum+expSgx2[n]*cos(Pi2nx[n]*(x_st-xdax[j]));
        sum:=2*sum+8;    // nx=0, expSgx2[0]=1, Sum_j(cos(0))=8*1=8
    ro_nucl:=sum/d3;
end;

function Diffuz_e(x:double):double;
begin
    Diffuz_e:=ro_e(x)*Le_d;
end;

function Diffuz_nucl(x:double):double;
begin
    Diffuz_nucl:=ro_nucl(x)*Lnucld;
end;

function Diffuz(x:double):double;
begin
    Diffuz:=Diffuz_e(x)+Diffuz_nucl(x);
end;
end.

```

unit Trajectories;

interface

uses

Constants, Solver, Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms, Dialogs, ExtCtrls, StdCtrls, ComCtrls, ClipBrd,
AddTrjDialog, ExtDlgs, AboutProgram, Buttons, Gauges, Spin, OleServer,
Excel2000, Menus, ToolWin, Math, RandomGauss;

type

TForm1 = class(TForm)
SolveTrBtn: TButton;
LabeledEdit1: TLabelledEdit;
LabeledEdit2: TLabelledEdit;
SavePictureDialog1: TSavePictureDialog;
Label2: TLabel;
LEGraphT0: TLabelledEdit;
LEGraphTLast: TLabelledEdit;
LEEnergyT: TLabelledEdit;
LEDelThGrad: TLabelledEdit;
Label3: TLabel;
LEortL: TLabelledEdit;
LEortK: TLabelledEdit;
Edit1: TEdit;
LabelCurv: TLabel;
Edit2: TEdit;
Label7: TLabel;
Label6: TLabel;
SaveDialog1: TSaveDialog;
Opendialog1: TOpenDialog;
LEZ1: TLabelledEdit;
LabelEV: TLabel;
LEEpsilon: TLabelledEdit;
LEMassOfIon: TLabelledEdit;
LabelAEM: TLabel;
LEDelThEps: TLabelledEdit;
LEnx: TLabelledEdit;
ChBoxSys5: TCheckBox;
ExcelApplication1: TExcelApplication;
MainMenu1: TMainMenu;
ApplicatMenu: TMenuItem;
OptionsMenu: TMenuItem;
HelpMenu: TMenuItem;
Graph2DMenu: TMenuItem;
AboutMenu: TMenuItem;
ChartXVTMenu: TMenuItem;
RaspredXVMenu: TMenuItem;
DechanMenu: TMenuItem;
EnergyMenu: TMenuItem;
FileMenu: TMenuItem;
SaveResultsMenu: TMenuItem;
MenuRazdelitel1: TMenuItem;
ExitMenu: TMenuItem;
ToolBar1: TToolBar;
ToolButton1: TToolButton;
U2dPlotBtn: TBitBtn;
RaspredBtn: TBitBtn;
XtBtn: TBitBtn;
EnergyBtn: TBitBtn;
Bevel1: TBevel;
Bevel2: TBevel;
LabelGrad: TLabel;
OpenTrjMenu: TMenuItem;
RestartMenu: TMenuItem;
MenuRazdelitel2: TMenuItem;
LabelProton: TLabel;
SolveNextBtn: TBitBtn;
AutoMenu: TMenuItem;
PrintDialog1: TPrintDialog;
Timer1: TTimer;

```

Menu1000ToNormal: TMenuItem;
DechanBtn: TBitBtn;
RestartBtn: TBitBtn;
OpenFileBtn: TBitBtn;
SaveResultsBtn: TBitBtn;
HelpBtn: TBitBtn;
AddTrBtn: TBitBtn;
ToolButton2: TToolButton;
ToolButton7: TToolButton;
ToolButton8: TToolButton;
CBCrystal: TComboBox;
CBPloskost: TComboBox;
CBApproximation: TComboBox;
CBLnucl: TComboBox;
Gaugel: TGauge;
ToolButton3: TToolButton;
Bevel3: TBevel;
ReportMemo: TMemo;
procedure OnCreate(Sender: TObject);
procedure SolveTrBtnClick(Sender: TObject);
procedure AddTrBtnClick(Sender: TObject);
procedure LEEnergyTChange(Sender: TObject);
procedure RaspredBtnClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure XtBtnClick(Sender: TObject);
procedure EnergyBtnClick(Sender: TObject);
procedure LabeledEdit1MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
procedure LENxChange(Sender: TObject);
procedure AboutMenuClick(Sender: TObject);
procedure SaveResultsMenuClick(Sender: TObject);
procedure ExitMenuClick(Sender: TObject);
procedure U2dPlotBtnClick(Sender: TObject);
procedure OpenTrjMenuClick(Sender: TObject);
procedure RestartMenuClick(Sender: TObject);
procedure LabelProtonClick(Sender: TObject);
procedure LabeledEdit2MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
procedure SolveNextBtnClick(Sender: TObject);
procedure AutoMenuClick(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
procedure Menu1000ToNormalClick(Sender: TObject);
procedure DechanBtnClick(Sender: TObject);
private
    { Private declarations }
    procedure ReadSettings;
    procedure DefaultSettings;
    procedure WriteSettings;
    procedure SolveNextLayer;
    procedure SaveResults(SaveFile: String; NoSaveEnergy: Boolean);
public
    { Public declarations }
    procedure Zapolnenie;
    procedure LoadTrjFormFile(FileWithTrjs: String);
    procedure Archiv7Zip(FileForArhiv: String);
    procedure Save1000(FileForSave1000: String);
    procedure Save1000ToNormal(FileWithSave1000: String; n1000s: integer);
    procedure ProcessingSave1000(FileForProcessing: String);
end;

TAllTraject=object
    Traject:array of OneTrajectory;
    CntOfTraject: integer;
    constructor Init(ser:PSeriaObj;dep_Start,depth:double;NodeCnt:integer);
    destructor Done;
    procedure SolveAll;
end;

procedure AutomaticSolve;
procedure RedErrors(ObjLE: TLabelledEdit; Cond: String);

```

```

var
    Form1: TForm1;
    Trajects: TAllTraject;

implementation

uses GraphU, Dechanneling, ChartXVT, RaspredV, RaspredE,
    MonteCarloDlg, OrientZav, Automatica;

{$R *.dfm}

Procedure TAllTraject.SolveAll;
var
    i:integer;
begin
    Form1.Gauge1.MaxValue:=CntOfTraject;
    for i:=0 to (CntOfTraject-1) do
        with Traject[i] do
            begin
                While not Solved do Solve;
                Form1.Gauge1.Progress:=i+1;
            end;
    FormRaspredE.Gauge3.Progress:=0;
end;

constructor TAllTraject.Init(ser:PSeriaObj;dep_Start,depth:double;NodeCnt:integer);
var
    i,j,k: integer;
    x,v: double;
begin
    k:=-1;
    Randomize;
    CntOfTraject:=ser^.AllTrCnt;
    SetLength(Traject,CntOfTraject);

    if NormV=True then RandomGauss.InitGauss(CntOfTraject,
                                                ser^.Seria[0].v1st,ser^.Seria[0].v2nd);
    for i:=0 to (ser^.SeriaCnt-1) do
        begin
            if (ser^.Seria[i].count > 1) then
                for j:=1 to ser^.Seria[i].count do
                    begin
                        inc(k);
                        x:=ser^.Seria[i].x1st+(j-1)*(ser^.Seria[i].x2nd-ser^.Seria[i].x1st)
                                                                    /(ser^.Seria[i].count-1);

                        if NormV=False
                        then
                            v:=ser^.Seria[i].v1st+(j-1)*(ser^.Seria[i].v2nd-
                                                                ser^.Seria[i].v1st)/(ser^.Seria[i].count-1)
                        else v:=RandomGauss.GetGauss(k);
                        //v:=RandG(ser^.Seria[i].v1st,ser^.Seria[i].v2nd);
                        Traject[k].Init(dep_Start,depth,NodeCnt,v,x);
                    end
                end
            else
                begin
                    inc(k);
                    Traject[k].Init(dep_Start,depth,NodeCnt,
                                    ser^.Seria[i].v1st,ser^.Seria[i].x1st);
                end;
            end;

    if NormV=True then Finalize(RandomGauss.v_Gauss);
end;

destructor TAllTraject.Done;
var
    k: integer;
begin
    for k:=0 to (CntOfTraject-1) do Traject[k].Done;
    Finalize(Traject);
end;

```

```

procedure TForm1.SolveTrBtnClick(Sender: TObject);
var
  cnt, code: integer;
  dep: double;
  s: String;
begin
  s:=Edit1.Text;  Val(s,Curv,code);  Curv:=Curv*d_small/Sqr(epsilon);
  s:=Edit2.Text;  Val(s,kt,code);
  s:=LabeledEdit1.Text;  Val(s,glubina,code);

if Form1.ChBoxSys5.Checked=False then
begin
  s:=LabeledEdit1.Text;      Val(s,dep,code);
  s:=LabeledEdit2.Text;      Val(s,cnt,code);
  dep_Start:=0;
  SolveSys5DE:=False;
  Gauge1.BackColor:=clYellow;  Gauge1.ForeColor:=clBlue;  Gauge1.Progress:=0;
  StartTime:=Now;
  ReportMemo.Lines.Add([' '+TimeToStr(Now)+'']  Расчёт начал
                      '+DateTimeToStr(StartTime));

  ReportMemo.Repaint;
  Trajects.Init(@AllSeria,dep_Start,dep_Start+dep,cnt);
  Trajects.SolveAll;
  Gauge1.BackColor:=clBtnFace;
  Finalize(Energy_Start);      Finalize(Energy_Last);
  SetLength(Energy_Start,Trajects.CntOfTraject);
  SetLength(Energy_Last,Trajects.CntOfTraject);
  SolveNextBtn.Enabled:=True;
if AutoSolve=True then AutomaticSolve;
  WorkTime:=Now-StartTime;
  ReportMemo.Lines.Add([' '+TimeToStr(Now)
                      +'']  Расчёт окончен, выполнен за время '
                      +TimeToStr(WorkTime));

end
else
begin
  MonteCarloDlgForm.Visible:=True;  MonteCarloDlgForm.Show;
end;
end;

procedure TForm1.SolveNextBtnClick(Sender: TObject);
begin
  StartTime:=Now;
  ReportMemo.Lines.Add([' '+TimeToStr(Now)+'']  Расчёт следующего слоя начал '
                      +DateTimeToStr(StartTime));

  ReportMemo.Repaint;
  SolveNextLayer;
if AutoSolve=True then AutomaticSolve;
  WorkTime:=Now-StartTime;
  ReportMemo.Lines.Add([' '+TimeToStr(Now)
                      +'']  Расчёт окончен, выполнен за время '
                      +TimeToStr(WorkTime));

end;

procedure TForm1.SolveNextLayer;
var
  i, cnt, code, Last: integer;
  dep: double;
  s: String;
  LastXV_arr: array of array of double;
begin
  s:=Edit1.Text;  Val(s,Curv,code);  Curv:=Curv*d_small/Sqr(epsilon);
  s:=Edit2.Text;  Val(s,kt,code);
  s:=LabeledEdit1.Text;  Val(s,glubina,code);
  s:=LabeledEdit1.Text;      Val(s,dep,code);
  s:=LabeledEdit2.Text;      Val(s,cnt,code);
  Gauge1.BackColor:=clWhite;  Gauge1.ForeColor:=clMoneyGreen;
  Gauge1.Progress:=0;
  SetLength(LastXV_arr,Trajects.CntOfTraject,7);

```

```

for i:=0 to Trajects.CntOfTraject-1 do
begin
    Last:=Trajects.Traject[i].NodeCount-1;
    LastXV_arr[i,0]:= Trajects.Traject[i].x_arr[Last];
    LastXV_arr[i,1]:= Trajects.Traject[i].v_arr[Last];
    LastXV_arr[i,2]:= Trajects.Traject[i].MA_arr[Last];
    LastXV_arr[i,3]:= Trajects.Traject[i].MB_arr[Last];
    LastXV_arr[i,4]:= Trajects.Traject[i].MC_arr[Last];
    LastXV_arr[i,5]:= Trajects.Traject[i].tmin0;
    LastXV_arr[i,6]:= Trajects.Traject[i].tmax0;
end;

dep_Start:=Trajects.Traject[0].t_arr[Trajects.Traject[0].NodeCount-1];
dep:=dep+dep_Start;

Str(dep_Start:6:3,s);    LEGraphT0.Text:=s;
Str(dep:6:3,s);          LEGraphTLast.Text:=s;

Trajects.Init(@AllSeria,dep_Start,dep,cnt);

for i:=0 to Trajects.CntOfTraject-1 do
begin
    Trajects.Traject[i].x_arr[0]:=LastXV_arr[i,0];
    Trajects.Traject[i].v_arr[0]:=LastXV_arr[i,1];
    Trajects.Traject[i].MA_arr[0]:=LastXV_arr[i,2];
    Trajects.Traject[i].MB_arr[0]:=LastXV_arr[i,3];
    Trajects.Traject[i].MC_arr[0]:=LastXV_arr[i,4];
    Trajects.Traject[i].tmin0:=LastXV_arr[i,5];
    Trajects.Traject[i].tmax0:=LastXV_arr[i,6];
    Energy_Start[i]:=Energy_Last[i];
end;

Finalize(LastXV_arr);    Trajects.SolveAll;
Gaugel.BackColor:=clBtnFace;
end;

procedure TForm1.AddTrBtnClick(Sender: TObject);
begin
    Form2.Visible:=true;    Form2.Show;
end;

procedure TForm1.LEEnergyTChange(Sender: TObject); // Изменение параметров:
var
    s: String; // Плоскость, приближение,
    DelThGrad, DelThEps: double; // Lnucl, k, l, Z1, MassOfIon,
    code: integer; // EnergyT, DelTh(в град.)
begin
    SolveTrBtn.Enabled:=False; //При неправильном задании "расчёт траекторий"
                                // остаётся недоступным
    Crystal:=CBCrystal.Text;    //Выбор кристалла
    InitCrystal;

    Ploskost:=CBPloskost.Text; //Выбор плоскости

    if CBLnucl.ItemIndex=1 then LnuclGemmell:=False else LnuclGemmell:=True;

    if CBApproximation.ItemIndex=1 then // Выбор приближения потенциала
    begin // (Approximation)
        Approximation:='Doil-Terner';
        InitDoyle;
    end
    else
        Approximation:='Molier';

    s:=LEortL.Text;    //Чтение значения L
    Val(s,l,code);
    if code<>0 then exit; //В случае ошибки - выход из процедуры (end)

    s:=LEortK.Text;    //Чтение значения K: Должно быть <> 0
    Val(s,k,code);
    if (code<>0) or (k=0) then exit;

```

```

s:=LEZ1.Text; //Чтение значения Z1
Val(s,Z1,code);
if (code<>0) then exit;
if Z1>0 then Znak_Z1:=1 else Znak_Z1:=-1;
Z1:=abs(Z1);

InitPloskost; // Эта процедура задаёт X_stj, Y_stj, Z_stj, ax, ay, az, XUmin,
// соответствующие выбранной плоскости

s:=LEMassOfIon.Text; //Масса частицы в а.е.м. должна быть >= 0
Val(s,MassOfIon,code);
if (code<>0) or (MassOfIon<=0) then exit;
MassOfIon:=MassOfIon*m0c2; //Т.к. в расчётах используется масса в эВ

s:=LEEnergyT.Text; //Энергия 'T' - кинетическая энергия частицы
Val(s,EnergyT,code);
if (code<>0) or (EnergyT<1000) then exit;

epsilon:=sqrt(abs(Vmax/(EnergyT+MassOfIon/(1+MassOfIon/EnergyT)))); //eps
Str(epsilon,s); LEEpsilon.Text:=s;

s:=LEDelThGrad.Text; //DelTh в градусах
Val(s,DelThGrad,code);
if code<>0 then exit;
LEDelThGrad.Hint:='Угол к цепочке = '+#13+FloatToStr(DelThGrad)+' Град.';
DelThEps:=DelThGrad*Pi/(epsilon*180); //DelTh: градусы --> eps
Str(DelThEps,s);
LEDelThEps.Text:=s; //Значение DelThEps
LEDelThEps.Hint:='delTh = '+#13+s+' x eps'+#13+'='
+FloatToStr(DelThEps)+' x eps';
DeltaTheta:=DelThEps*epsilon; //Т.к. в расчётах используется такое значение

InitConstants;

SolveTrBtn.Enabled:=True;

Form1.Caption:='Расчёт траекторий частиц с Z1= '+IntToStr(Z1)
+' , массой = '+FloatToStr(MassOfIon/m0c2)+' а.е.м.'
+' в плоскости '+Ploskost+' кристалла '+Crystal;
end;

procedure TForm1.LEnxChange(Sender: TObject); // Изменение 'nx'
var
s: String;
code, nxNew: integer;
begin
s:=LEnx.Text;
Val(s,nxNew,code);
if code<>0 then exit;
nxRange:=nxNew;
LEnx.EditLabel.Caption:='nx='+IntToStr(nxRange);
Form1.LEEnergyTChange(Form1);
end;

procedure TForm1.LabeledEdit1MouseMove(Sender: TObject; Shift: TShiftState;
X, Y: Integer); // Всплывающая подсказка: показывает глубину в Анг.
var
s: String;
code: integer;
gl: double;
begin
s:=LabeledEdit1.Text; Val(s,gl,code);
if code<>0 then exit;
gl:=gl*d_small/epsilon; Str(gl,s);
LabeledEdit1.Hint:='Глубина: '+LabeledEdit1.Text+' тау'+#13
+s+' Анг.='+#13+'=' +FloatToStr(gl)+' Анг.'+#13
+'=' +FloatToStr(gl/1E7)+' мм.';
end;

procedure TForm1.LabeledEdit2MouseMove(Sender: TObject; Shift: TShiftState;

```



```

    X, Y: Integer);          // Всплывающая подсказка: показывает кол-во узлов,
var                          // глубину в тау и Анг.,
    gl: double;              // шаг по глубине в тау и Анг.
    nuzlov, code: integer;
begin
    Val(LabeledEdit2.Text,nuzlov,code);
    if (code<>0) or (nuzlov<3) then begin LabeledEdit2.Hint:='???'; exit; end;
    Val(LabeledEdit1.Text,gl,code);
    if code<>0 then begin LabeledEdit2.Hint:='???'; exit; end;
    if SolveTrBtn.Enabled=False then begin LabeledEdit2.Hint:='???'; exit; end;
    LabeledEdit2.Hint:='Кол-во узлов = '+IntToStr(nuzlov)+'13+13
        +'Глубина = '+FloatToStr(gl)+' тау = '+13
        +' = '+FloatToStr(gl*d_small/epsilon)+' Анг.'+13
        +' = '+FloatToStr(gl*d_small/epsilon/1E7)+' мм.'+13+13
        +'Шаг = '+FloatToStr(gl/(nuzlov-1))+' тау = '+13
        +' = '+FloatToStr(gl*d_small/epsilon/(nuzlov-1))+' Анг.';

end;

procedure TForm1.LabelProtonClick(Sender: TObject);
begin
    // Устанавливает Z1 и m для протона
    LEZ1.Text:='1';    LEMassOfIon.Text:='1.00727647';
end;

procedure TForm1.ReadSettings; //Процедура чтения файла с настройками
var
    i: integer;              // записывает настройки в массив Settings[i]
    Text: String;            // или обращается к процедуре DefaultSettings
    SetEd: TextFile;         // для восстановления массива Settings по умолчанию.
begin
    DefaultSettings;
    try
        Text:=ExtractFilePath(paramstr(0));
        Text:=Text+'SetEd.txt';    AssignFile(SetEd,Text);
        Reset(SetEd);
        For i:=1 to Length(Settings) do Readln(SetEd,Settings[i]);
        CloseFile(SetEd);
    except
        Application.MessageBox('Т.к. файл SetEd.txt недоступен,'+13+13
            +'будут применены установки'+13+13
            +'по умолчанию',
            'Traject.exe - Проблемы',MB_IconWarning);
    end;
end;

procedure TForm1.DefaultSettings; // Восстанавливает массив Settings
begin
    // по умолчанию:
    Settings[1]:='-0.375';          Settings[2]:='-1.4';
    Settings[3]:='0.375';          Settings[4]:='1.4';
    Settings[5]:='5';              Settings[6]:='1';
    Settings[7]:='500';            Settings[8]:='1E+6';
    Settings[9]:='1';              Settings[10]:='1';
    Settings[11]:='1.00727647';     Settings[12]:='0';
    Settings[13]:='-1000';          Settings[14]:='1000';
    Settings[15]:='1';              Settings[16]:='0';
    Settings[17]:='0';              Settings[18]:='0';
    Settings[19]:='156.871';        Settings[20]:='65.775';
    Settings[21]:='2192';           Settings[22]:='742';
    Settings[23]:='40';             Settings[24]:='CheckBoxSys5=True';
    Settings[25]:='0';              Settings[26]:='150';
    Settings[27]:='0.073739403';    Settings[28]:='0.141421356';
    Settings[29]:='0,01843485075';  Settings[30]:='0,035355339';
    Settings[31]:='-5';             Settings[32]:='5';
    Settings[33]:='0,001';          Settings[34]:='0,1';
    Settings[35]:='0,00';           Settings[36]:='1,00';
    Settings[37]:='0,02';           Settings[38]:='0';
    Settings[39]:='1';              Settings[40]:='-1';
    Settings[41]:='1';              Settings[42]:='0';
    Settings[43]:='1';              Settings[44]:='CheckBoxAngstrem=True';
    Settings[45]:='CheckBoxRoe07=False'; Settings[46]:='CheckBoxGammaDeltha=True';
    Settings[47]:='0';              Settings[48]:='ChBDecShow=False';
    Settings[49]:='ChBAngstrem=True'; Settings[50]:='ChBeVAngstrem=False';

```

```

Settings[51]:='0,01';
Settings[53]:='100';
Settings[55]:='-1';
Settings[57]:='0';
Settings[59]:='1000';
Settings[61]:='CBLeaveDec=False';
Settings[63]:='ChBoxCombineRozigrish=True';
Settings[65]:='1';
Settings[67]:='1';
Settings[69]:='CheckBoxDec=False';
Settings[71]:='AutoScaleMorgunGraph=True';
Settings[73]:='0';
Settings[75]:='LnuclGemmell=True';
Settings[77]:='0';
Settings[79]:='0.01';
Settings[81]:='ChBoxNormGistEL=True';
Settings[83]:='UxxForSolver0=False';
Settings[85]:='ChBoxTAngTau=True';
Settings[87]:='UxxForSolver3D=False';
Settings[89]:='ChBoxWithhE=False';
Settings[91]:='NormV=False';
Settings[93]:='100';

Settings[52]:='0';
Settings[54]:='ChBGisteVAng=True';
Settings[56]:='1';
Settings[58]:='1';
Settings[60]:='CBAngstrem=True';
Settings[62]:='ChBoxNoExceedMmax=True';
Settings[64]:='ChBoxSolveME=False';
Settings[66]:='1';
Settings[68]:='1';
Settings[70]:='CheckBoxAngstrem=True';
Settings[72]:='500';
Settings[74]:='0';
Settings[76]:='0';
Settings[78]:='1';
Settings[80]:='0';
Settings[82]:='ChBoxDecELG1Ang=True';
Settings[84]:='100';
Settings[86]:='ChBoxNormDec=False';
Settings[88]:='PeriodPoX=True';
Settings[90]:='0';
Settings[92]:='1000';

end;

```

```

procedure TForm1.Zapolnenie; //процедура заполняет данными форму Form1
begin

```

```

    LabeledEdit1.Text:=Settings[6];    LabeledEdit2.Text:=Settings[7];
    LEEnergyT.Text:=Settings[8];    LEDELthGrad.Text:=Settings[9];
    LEZ1.Text:=Settings[10];    LEMassOfIon.Text:=Settings[11];
    CBPloskost.ItemIndex:=StrToIntDef(Settings[12],0);
    LEortk.Text:=Settings[15];    LEortl.Text:=Settings[16];
    Edit1.Text:=Settings[17];    Edit2.Text:=Settings[18];
    LENx.Text:=Settings[23];
    CBApproximation.ItemIndex:=StrToIntDef(Settings[25],0);
    if Settings[24]='ChBoxSys5=True'
    then ChBoxSys5.Checked:=True    else ChBoxSys5.Checked:=False;
    if Settings[75]='LnuclGemmell=True'
    then LnuclGemmell:=True    else LnuclGemmell:=False;
    CBCrystal.ItemIndex:=StrToIntDef(Settings[90],0);
    LEEnergyTChange(Form1);

```

```

end;

```

```

procedure TForm1.OnCreate(Sender: TObject);

```

```

begin
    nxRange:=40;    dep_Start:=0;
    ReadSettings;    Zapolnenie;
    SavePictureDialog1.DefaultExt:='*.emf';
    SavePictureDialog1.Filter := 'Win32 Enhanced Metafiles (*.emf)|*.emf';
    Label3.Caption:=Chr($B4)+Chr($65);
    Application.HintHidePause:=10000;
    LEMassOfIon.Hint:='Масса налетающего иона в а.е.м.'+#13
        +'1 а.е.м. = '+FloatToStr(mOc2)+' эВ'+#13
        +'для протона m = 1.00727647 а.е.м.';
    Edit1.Hint:='Заданное здесь число *d / eps^2'+#13+
        'Curv = 0 - для прямого кристалла,'+#13+
        'Curv = 7E-11 - для Ризгиба = 148 см';

```

```

    ReportMemo.Lines.Add(['+TimeToStr(Now)+']    Всё в порядке!');

```

```

end;

```

```

procedure TForm1.WriteSettings; // Процедура составляет массив Settings

```

```

begin

```

```

    Settings[1]:=Form2.LabeledEdit1.Text;    Settings[2]:=Form2.LabeledEdit2.Text;
    Settings[3]:=Form2.LabeledEdit3.Text;    Settings[4]:=Form2.LabeledEdit4.Text;
    Settings[5]:=Form2.LabeledEdit5.Text;    Settings[6]:=Form1.LabeledEdit1.Text;
    Settings[7]:=Form1.LabeledEdit2.Text;    Settings[8]:=Form1.LEEnergyT.Text;
    Settings[9]:=Form1.LEDELthGrad.Text;    Settings[10]:=Form1.LEZ1.Text;
    Settings[11]:=Form1.LEMassOfIon.Text;
    Settings[12]:=IntToStr(Form1.CBPloskost.ItemIndex);
    Settings[13]:='-1E10'; Settings[14]:='1E10';
    Settings[15]:=Form1.LEortk.Text;    Settings[16]:=Form1.LEortl.Text;

```

```

Settings[17]:=Form1.Edit1.Text;    Settings[18]:=Form1.Edit2.Text;
Settings[19]:='1'; Settings[20]:='1'; Settings[21]:='1'; Settings[22]:='1';
Settings[23]:=Form1.LEnx.Text;
if Form1.ChBoxSys5.Checked=True
then Settings[24]:='ChBoxSys5=True'
    else Settings[24]:='ChBoxSys5=False';
Settings[25]:=IntToStr(Form1.CBApproximation.ItemIndex);
Settings[26]:=MonteCarloDlgForm.LabeledEdit1.Text;
Settings[27]:=MonteCarloDlgForm.LabeledEdit2.Text;
Settings[28]:=MonteCarloDlgForm.LabeledEdit3.Text;
Settings[29]:=FormRaspredV.LEMAmin.Text;
Settings[30]:=FormRaspredV.LEMCmin.Text;
Settings[31]:=FormRaspredV.LEXVot.Text;
Settings[32]:=FormRaspredV.LEXVdo.Text;
Settings[33]:=FormRaspredV.LEXVstep.Text;
Settings[34]:=FormRaspredV.LEhV.Text;
Settings[35]:=FormRaspredV.LECalcVot.Text;
Settings[36]:=FormRaspredV.LECalcVdo.Text;
Settings[37]:=FormRaspredV.LEhX.Text;
Settings[38]:=FormRaspredV.LECalcXot.Text;
Settings[39]:=FormRaspredV.LECalcXdo.Text;
Settings[40]:=FormRaspredV.LEVminGl.Text;
Settings[41]:=FormRaspredV.LEVmaxGl.Text;
Settings[42]:=FormRaspredV.LECalcGlot.Text;
Settings[43]:=FormRaspredV.LECalcGldo.Text;
if FormRaspredV.ChBoxAngstrem.Checked=True
    then Settings[44]:='ChBoxAngstrem=True'
        else Settings[44]:='ChBoxAngstrem=False';
if FormRaspredE.ChBoxRoe07.Checked=True
    then Settings[45]:='ChBoxRoe07=True'    else Settings[45]:='ChBoxRoe07=False';
if FormRaspredE.ChBoxGammaDeltha.Checked=True
    then Settings[46]:='ChBoxGammaDeltha=True'
        else Settings[46]:='ChBoxGammaDeltha=False';
Settings[47]:='0';    Settings[48]:='ChBDecShow=False';
if FormRaspredE.ChBANGstrem.Checked=True
    then Settings[49]:='ChBANGstrem=True'    else Settings[49]:='ChBANGstrem=False';
if FormRaspredE.ChBeVANGstrem.Checked=True
    then Settings[50]:='ChBeVANGstrem=True'
        else Settings[50]:='ChBeVANGstrem=False';
Settings[51]:=FormRaspredE.LEhE.Text;
Settings[52]:=FormRaspredE.LECalcEot.Text;
Settings[53]:=FormRaspredE.LECalcEdo.Text;
if FormRaspredE.ChBGisteVAng.Checked=True
    then Settings[54]:='ChBGisteVAng=True'
        else Settings[54]:='ChBGisteVAng=False';
Settings[55]:=FormRaspredE.LEEminGl.Text;
Settings[56]:=FormRaspredE.LEEmaxGl.Text;
Settings[57]:=FormRaspredE.LECalcGlot.Text;
Settings[58]:=FormRaspredE.LECalcGldo.Text;
Settings[59]:=FormRaspredE.LERazreshSpectra.Text;
if FormDechanneling.CBANGstrem.Checked=True
    then Settings[60]:='CBANGstrem=True'    else Settings[60]:='CBANGstrem=False';
if FormDechanneling.CBLeaveDec.Checked=True
    then Settings[61]:='CBLeaveDec=True'    else Settings[61]:='CBLeaveDec=False';
if MonteCarloDlgForm.ChBoxNoExceedMmax.Checked=True
    then Settings[62]:='ChBoxNoExceedMmax=True'
        else Settings[62]:='ChBoxNoExceedMmax=False';
if MonteCarloDlgForm.ChBoxCombineRozigrish.Checked=True
    then Settings[63]:='ChBoxCombineRozigrish=True'
        else Settings[63]:='ChBoxCombineRozigrish=False';
Settings[64]:='ChBoxSolveME=False';
Settings[65]:=FormRaspredE.LEStepUpDownGistE.Text;
Settings[66]:=FormRaspredV.LEStepUpDownFxv.Text;
Settings[67]:=FormRaspredV.LEStepUpDownGistX.Text;
Settings[68]:=FormRaspredV.LEStepUpDownGistV.Text;
if FormChartXVT.CheckBoxDec.Checked=True
    then Settings[69]:='CheckBoxDec=True'    else Settings[69]:='CheckBoxDec=False';
if FormChartXVT.CheckBoxAngstrem.Checked=True
    then Settings[70]:='CheckBoxAngstrem=True'
        else Settings[70]:='CheckBoxAngstrem=False';
Settings[71]:='AutoScaleMorgunGraph=True';

```

```

Settings[72]:='500';    Settings[73]:='0';    Settings[74]:='1';
if LnuclGemmell=True
  then Settings[75]:='LnuclGemmell=True'    else Settings[75]:='LnuclGemmell=False';
  Settings[76]:=IntToStr(FormGraphU.CBGraph.ItemIndex);
  Settings[77]:=FormGraphU.LEXot.Text;
  Settings[78]:=FormGraphU.LEXdo.Text;
  Settings[79]:=FormGraphU.LEStepX.Text;
  Settings[80]:=FormGraphU.LETst.Text;
if FormRaspredE.ChBoxNormGistEL.Checked=False
  then Settings[81]:='ChBoxNormGistEL=False'
    else Settings[81]:='ChBoxNormGistEL=True';
if FormRaspredE.ChBoxDecELG1Ang.Checked=False
  then Settings[82]:='ChBoxDecELG1Ang=False'
    else Settings[82]:='ChBoxDecELG1Ang=True';
if UxxForSolver0=True
  then Settings[83]:='UxxForSolver0=True'    else Settings[83]:='UxxForSolver0=False';
  Settings[84]:=MonteCarloDlgForm.LabeledEdit4.Text;
if MonteCarloDlgForm.ChBoxTAngTau.Checked=False
  then Settings[85]:='ChBoxTAngTau=False'    else Settings[85]:='ChBoxTAngTau=True';
if FormDechanneling.ChBoxNormDec.Checked=False
  then Settings[86]:='ChBoxNormDec=False'
    else Settings[86]:='ChBoxNormDec=True';
if UxxForSolver3D=True
  then Settings[87]:='UxxForSolver3D=True'
    else Settings[87]:='UxxForSolver3D=False';
if PeriodPoX=True
  then Settings[88]:='PeriodPoX=True'    else Settings[88]:='PeriodPoX=False';
if FormRaspredE.ChBoxWithhE.Checked=True
  then Settings[89]:='ChBoxWithhE=True'    else Settings[89]:='ChBoxWithhE=False';
  Settings[90]:=IntToStr(Form1.CBCrystal.ItemIndex);
if NormV=True
  then Settings[91]:='NormV=True'    else Settings[91]:='NormV=False';
  Settings[92]:=FormRaspredE.LECenaKanala.Text;
  Settings[93]:=FormRaspredE.LEStepSpectraE.Text;
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
var
  i: integer;
  Text: String;
  SetEd: TextFile;
begin
  Finalize(Energy_arr);  Finalize(Energy_Start);  Finalize(Energy_Last);
  Trajects.Done;  //Finalize(Trajects.Traject);
  AllSeria.Done;  //Finalize(AddTrjDialog.AllSeria.Seria);
  FinalizeConst;

  if LoadedTrj=True then begin
    if Application.MessageBox(' Были установлены настройки'+#13+ ' '+#13+
      ' из файла с сохранёнными'+#13+ ' '+#13+
      ' траекториями'+#13+ ' '+#13+
      'Сохранить настройки в SetEd.txt?',
      'Что делать с настройками?',
      MB_YesNo+MB_IconQuestion) = mrNo then exit;
  end;

  WriteSettings;

  try
    Text:=ExtractFilePath(paramstr(0));
    Text:=Text+'SetEd.txt';
    AssignFile(SetEd,Text);
    Rewrite(SetEd);
    For i:=1 to Length(Settings) do WriteLn(SetEd,Settings[i]);
    CloseFile(SetEd);
  except {Если сохранить файл SetEd.txt невозможно}
  MessageDlg('Сохранение настроек невозможно!',mtError,[mbOk],0);
  end;
end;

// Запуск 'приложений':

```

```

procedure TForm1.U2dPlotBtnClick(Sender: TObject); // Графики функций U, ro, ...
begin
    FormGraphU.Visible:=True;  FormGraphU.Show;
end;

procedure TForm1.XtBtnClick(Sender: TObject);      // Графики траекторий
var
    i: integer;
    Text, Sx, Sv: String;
begin
    FormChartXVT.Visible:=True;
    FormChartXVT.Show;
    FormChartXVT.Gauge1.Progress:=0;
    FormChartXVT.ListBox1.Clear;
    For i:=0 to Trajects.CntOfTraject-1 do begin
        Str(Trajects.Traject[i].x_arr[0], Sx);
        Str(Trajects.Traject[i].v_arr[0], Sv);
        Text:='Tp.№ '+IntToStr(i+1)+' (i='+IntToStr(i)+') | Хнач= '+Sx
            +' | Vнач= '+Sv;
        FormChartXVT.ListBox1.Items.Add(Text);
        FormChartXVT.LabelForAllTrjClick(self);
        FormChartXVT.LabelForAllUzelsClick(self);
    end;
end;

procedure TForm1.RaspredBtnClick(Sender: TObject); // Распределения по X и V
begin
    FormRaspredV.Visible:=True;
    FormRaspredV.Show;
    FormRaspredV.LabelForAllTrjClick(self);
    FormRaspredV.LabelForAllUzelsClick(self);
    if Trajects.Traject[0].NodeCount<=32767 then
        begin
            FormRaspredV.UpDownFxv.Max:=(Trajects.Traject[0].NodeCount-1);
            FormRaspredV.UpDownGistV.Max:=(Trajects.Traject[0].NodeCount-1);
            FormRaspredV.UpDownGistX.Max:=(Trajects.Traject[0].NodeCount-1);
        end
    else
        begin
            FormRaspredV.UpDownFxv.Enabled:=False;
            FormRaspredV.UpDownGistV.Enabled:=False;
            FormRaspredV.UpDownGistX.Enabled:=False;
        end;
    end;

procedure TForm1.EnergyBtnClick(Sender: TObject); // Потери энергии
begin
    FormRaspredE.Visible:=True;
    FormRaspredE.Show;
    FormRaspredE.LabelForAllTrjClick(self);
    FormRaspredE.LabelForAllUzelsClick(self);
    if Trajects.Traject[0].NodeCount<=32767 then
        begin
            FormRaspredE.UpDownGistE.Max:=(Trajects.Traject[0].NodeCount-1);
            FormRaspredE.UpDownGistE.Position:=(Trajects.Traject[0].NodeCount-1);
            FormRaspredE.UpDownSpectraE.Max:=(Trajects.Traject[0].NodeCount-1);
            FormRaspredE.UpDownSpectraE.Position:=(Trajects.Traject[0].NodeCount-1);
        end
    else
        begin
            FormRaspredE.UpDownGistE.Enabled:=False;
            FormRaspredE.UpDownSpectraE.Enabled:=False;
        end;
    end;

    // Меню:

procedure TForm1.AboutMenuClick(Sender: TObject);
begin
    FormAbout.Visible:=true;  FormAbout.LEDate.Text:=DateToStr(Now);

```

```

end;

procedure TForm1.SaveResultsMenuClick(Sender: TObject);
var
  Text: String;
  NoSaveEnergy: Boolean;
begin
  NoSaveEnergy:=False;

  try
    Str(Energy_arr[0,1],Text);
  except
    if Application.MessageBox('Проблемы с сохранением потерь энергии'+#13+ ' '+#13+
      'Сохранять без потерь энергии?',
      'Сохранение:',MB_YESNO+MB_ICONWARNING) = mrNo
    then exit
    else NoSaveEnergy:=True;
  end;

  SaveDialog1.DefaultExt:='xvt';
  SaveDialog1.Filter := 'Файлы траекторий XV (*.xvt)|*.XVT';
  if SaveDialog1.Execute then
    begin
      Text:=SaveDialog1.FileName;
      ReportMemo.Lines.Add('['+TimeToStr(Now)+'] Записываем траектории в файл '+Text);
      ReportMemo.Repaint;
      SaveResults(Text, NoSaveEnergy);
      ReportMemo.Lines.Add('['+TimeToStr(Now)+'] Траектории записаны нормально');
    end;
  end;

procedure TForm1.SaveResults(SaveFile: String; NoSaveEnergy: Boolean);
var
  // Запись (сохранение) результатов работы в один файл:
  i, ti, NumTrj, NumTrPerBun, code: integer;
  Text: String;
  SaveF: TextFile;
begin
  Gauge1.BackColor:=clAqua; Gauge1.ForeColor:=clMaroon; Gauge1.Progress:=0;
  Gauge1.MaxValue:=Trajects.CntOfTraject;

  AssignFile(SaveF,SaveFile);
  Rewrite(SaveF);

  Writeln(SaveF,'Save TRAJECTORIES XV 04-05'); {Вариант сохранения}
  Writeln(SaveF,Length(Settings));

  WriteSettings;
  For i:=1 to Length(Settings) do Writeln(SaveF,Settings[i]);

  if SolveMonteCarlo=True
  then Writeln(SaveF,'SolveMonteCarlo=True')
  else Writeln(SaveF,'SolveMonteCarlo=False');

  Writeln(SaveF,Trajects.Traject[0].t_arr[0]);
  Writeln(SaveF,Trajects.Traject[0].t_arr[Trajects.Traject[0].NodeCount-1]);
  Writeln(SaveF,Trajects.Traject[0].tau);
  Writeln(SaveF,Trajects.Traject[0].NodeCount);
  Writeln(SaveF,Trajects.CntOfTraject);

  Writeln(SaveF,Form2.ListBox1.Count);
  NumTrj:=0;
  For i:=0 to Form2.ListBox1.Count-1 do
    begin
      Text:=Form2.ListBox1.Items.Strings[i];
      Delete(Text,Pos('T',Text),Length(Text));
      Val(Text,NumTrPerBun,code);
      Str((Trajects.Traject[NumTrj].x_arr[0]),Text); Writeln(SaveF,Text);
      Str((Trajects.Traject[NumTrj].v_arr[0]),Text); Writeln(SaveF,Text);
      NumTrj:=NumTrj+NumTrPerBun-1;
      Str((Trajects.Traject[NumTrj].x_arr[0]),Text); Writeln(SaveF,Text);
      Str((Trajects.Traject[NumTrj].v_arr[0]),Text); Writeln(SaveF,Text);
    end;
  end;

```

```

        WriteLn(SaveF, NumTrPerBun);
        NumTrj:=NumTrj+1;
    end;
    For i:=0 to (Trajects.CntOfTraject-1) do begin
        for ti:=0 to (Trajects.Traject[i].NodeCount-1) do begin
            WriteLn(SaveF, (Trajects.Traject[i].x_arr[ti]));
            WriteLn(SaveF, (Trajects.Traject[i].v_arr[ti]));
            if (Trajects.Traject[i].MA_arr[ti])=0
            then WriteLn(SaveF, 0)
            else WriteLn(SaveF, (Trajects.Traject[i].MA_arr[ti]));
            if (Trajects.Traject[i].MB_arr[ti])=0
            then WriteLn(SaveF, 0)
            else WriteLn(SaveF, (Trajects.Traject[i].MB_arr[ti]));
            if (Trajects.Traject[i].MC_arr[ti])=0
            then WriteLn(SaveF, 0)
            else WriteLn(SaveF, (Trajects.Traject[i].MC_arr[ti]));
            WriteLn(SaveF, 0)
        end;
        Form1.Gauge1.Progress:=i+1;
    end;
    if NoSaveEnergy=False then
    begin
        WriteLn(SaveF, 'Save Energy_arr');
        For i:=0 to (Trajects.CntOfTraject-1) do begin
            for ti:=0 to (Trajects.Traject[i].NodeCount-1) do begin
                WriteLn(SaveF, Energy_arr[i, ti]);
            end;
        end;
    end;
    WriteLn(SaveF, 'Save-T0-MonteCarlo');
    For i:=0 to (Trajects.CntOfTraject-1) do begin
        WriteLn(SaveF, Trajects.Traject[i].tmin0);
        WriteLn(SaveF, Trajects.Traject[i].tmax0);
    end;
    CloseFile(SaveF);
    Gauge1.BackColor:=clBtnFace;
end;

procedure TForm1.OpenTrjMenuClick(Sender: TObject);
var
    Text: String;
begin
    Opndialog1.DefaultExt:='xvt';
    Opndialog1.Filter := 'Файлы траекторий XV (*.xvt)|*.XVT';
    if Opndialog1.Execute then
    begin
        Text:=Opndialog1.FileName;
        ReportMemo.Lines.Add([' '+TimeToStr(Now)+'']    Открываем файл '+Text);
        ReportMemo.Refresh;
        LoadTrjFormFile(Text);
        ReportMemo.Lines.Add([' '+TimeToStr(Now)+'']    Траектории загружены нормально');
    end;
end;

procedure TForm1.RestartMenuClick(Sender: TObject); // Сброс траекторий
begin
    if (Application.MessageBox(' Перезапуск программы'+#13+' '+#13+
        ' Удаляются все траектории'+#13+' '+#13+
        'Продолжить выполнение ?',
        'Restart ?',
        MB_YesNo+MB_IconQuestion) = mrNo) then exit;

    Gauge1.Progress:=0;
    Finalize(Energy_arr); Finalize(Energy_Start); Finalize(Energy_Last);
    Trajects.Done; AllSeria.Done;
    AllSeria.Init(25, @Form2);
    Form2.ListBox1.Clear; Form2.ChBoxNormV.Checked:=False;
    Trajects.CntOfTraject:=0;

    ReportMemo.Lines.Add([' '+TimeToStr(Now)+'']    Restart ');
end;

```

```

procedure TForm1.ExitMenuClick(Sender: TObject);
begin
    Form1.Close;
end;

// Другое:

procedure RedErrors(ObjLE: TLabelledEdit; Cond: String);
var
    i, code: integer; // Процедура для проверки правильности вводимых значений
    r: double;
    S: String;
begin
    ObjLE.Font.Color:=clRed;
    try
        S:=ObjLE.Text;
        if Length(S)=0 then exit;
        For i:=0 to Length(S) do if S[i]='.' then S[i]:=',';
        ObjLE.Text:=S;    r:=StrToFloat(ObjLE.Text);
    except
        on EConvertError do exit;
    end;
    if (Cond='Integer') then
        begin
            Val(ObjLE.Text, i, code);
            if code<>0 then exit;
            r:=i;
        end;
    if (Cond='No <=0') and (r<=0) then exit;
    ObjLE.Font.Color:=clBlack;
end;

procedure TForm1.AutoMenuClick(Sender: TObject);
begin
    AutomaticaForm.Visible:=True;
end;

procedure TForm1.LoadTrjFormFile(FileWithTrjs: String);
Label
    CloseSaveF;
var
    i, j, ti, code, CntBunch, n_tr, nuzlov, imax: integer;
    Progress: integer;
    dep, left, right, tau, xv: double;
    SaveF: TextFile;
    Text, Version: String;
    t_Start: array of double;
begin

    if LoadedTrj=True then
        begin
            SetLength(t_Start,Trajects.CntOfTraject);
            For i:=0 to (Trajects.CntOfTraject-1) do
                t_Start[i]:=Trajects.Traject[i].t_arr[0];
            end;

    Gauge1.BackColor:=clAqua;
    Gauge1.Progress:=0;

    AssignFile(SaveF,FileWithTrjs);
    Reset(SaveF);

    Readln(SaveF,Version);
    if (Version<>'Save TRAJECTORIES XV 03-05')
        and (Version<>'Save TRAJECTORIES XV 03-05+')
        and (Version<>'Save TRAJECTORIES XV 04-05') then
        begin
            Application.MessageBox('Указан неверный файл',
                                    'Загрузка остановлена !',
                                    MB_Iconstop);

```



```

        CloseFile(SaveF);
        Gage1.BackColor:=clBtnFace;
        exit;
    end;
DefaultSettings;
Readln(SaveF,Text);
if Text='+' then imax:=82
else imax:=StrToInt(Text);
For i:=1 to imax do Readln(SaveF,Settings[i]);
Form1.Zapolnenie; FormGraphU.Zapolnenie; FormChartXVT.Zapolnenie;
FormDechanneling.Zapolnenie; FormRaspredV.Zapolnenie;
FormRaspredE.Zapolnenie; MonteCarloDlgForm.Zapolnenie;
Readln(SaveF,Text);
if Text='SolveMonteCarlo=True' then
begin
    SolveSys5DE:=True;
    SolveMonteCarlo:=True;
    MonteCarloDlgForm.DoMonteCarloBtn.Font.Color:=clBlue;
    MonteCarloDlgForm.DoNoMonteCarloBtn.Font.Color:=clBlack;
    Val(Settings[26],Tmax_MonteCarlo,code);
    Val(Settings[84],Tmin_MonteCarlo,code);
    if Settings[85]='ChBoxTAngTau=True' then
    begin
        Tmax_MonteCarlo:=Tmax_MonteCarlo*epsilon/d_small;
        Tmin_MonteCarlo:=Tmin_MonteCarlo*epsilon/d_small;
    end;
    Val(Settings[27],MA_max,code);
    Val(Settings[28],MC_max,code);
    FormRaspredV.LabelMAmax.Caption:='MA_max = '+FloatToStr(MA_max);
    FormRaspredV.LabelMCmax.Caption:='MC_max = '+FloatToStr(MC_max);
    if Settings[62]='ChBoxNoExceedMmax=True'
    then NoExceedMmax:=True
    else NoExceedMmax:=False;
    if Settings[63]='ChBoxCombineRozigrish=False'
    then CombineRozigrish:=False
    else CombineRozigrish:=True;
    if Settings[64]='ChBoxSolveME=True'
    then SolveME:=True
    else SolveME:=False;
end
else
begin
    SolveSys5DE:=True;
    SolveMonteCarlo:=False;
    MonteCarloDlgForm.DoMonteCarloBtn.Font.Color:=clBlack;
    if Settings[24]='ChBoxSys5=True' then
    begin
        MonteCarloDlgForm.DoNoMonteCarloBtn.Font.Color:=clBlue;
        if Settings[64]='ChBoxSolveME=True'
        then SolveME:=True
        else SolveME:=False;
    end;
end;

if Settings[24]='ChBoxSys5=False'
then SolveSys5DE:=False;

if Version='Save TRAJECTORIES XV 04-05' then
begin
    Readln(SaveF,Text); Val(Text,dep_Start,code);
    Readln(SaveF,Text);
end
else dep_Start:=0;

Readln(SaveF,Text); Val(Text,tau,code);
Readln(SaveF,Text); Val(Text,nuzlov,code);
Readln(SaveF,Text); Val(Text,n_tr,code);

Readln(SaveF,Text); Val(Text,CntBunch,code);
For i:=0 to CntBunch-1 do
begin

```

```

ReadLn (SaveF, Text);
Val (Text, xv, code);
Text:=FloatToStr(xv);
  For j:=0 to Length(Text) do
    if Text[j]=',' then Text[j]:='.';
    Form2.LabeledEdit1.Text:=Text;
ReadLn (SaveF, Text);
Val (Text, xv, code);
Text:=FloatToStr(xv);
  For j:=0 to Length(Text) do
    if Text[j]=',' then Text[j]:='.';
    Form2.LabeledEdit2.Text:=Text;
ReadLn (SaveF, Text);
Val (Text, xv, code);
Text:=FloatToStr(xv);
  For j:=0 to Length(Text) do
    if Text[j]=',' then Text[j]:='.';
    Form2.LabeledEdit3.Text:=Text;
ReadLn (SaveF, Text);
Val (Text, xv, code);
Text:=FloatToStr(xv);
  For j:=0 to Length(Text) do
    if Text[j]=',' then Text[j]:='.';
    Form2.LabeledEdit4.Text:=Text;
ReadLn (SaveF, Text); Form2.LabeledEdit5.Text:=Text;
Form2.BitBtn1Click(self);
end;

Text:=LabeledEdit1.Text; Val (Text, dep, code); glubina:=dep;

Gaugel.ForeColor:=clMaroon;
Gaugel.MaxValue:=n_tr;
Trajects.Init (@AllSeria, 0, dep, nuzlov);
SetLength(t_Start, Trajects.CntOfTraject);
Progress:=0;
For i:=(Trajects.CntOfTraject-n_tr) to (Trajects.CntOfTraject-1) do begin
  for ti:=0 to nuzlov-1 do begin
    ReadLn (SaveF, Text); Val (Text, Trajects.Traject[i].x_arr[ti], code);
    ReadLn (SaveF, Text); Val (Text, Trajects.Traject[i].v_arr[ti], code);
    ReadLn (SaveF, Text); Val (Text, Trajects.Traject[i].MA_arr[ti], code);
    ReadLn (SaveF, Text); Val (Text, Trajects.Traject[i].MB_arr[ti], code);
    ReadLn (SaveF, Text); Val (Text, Trajects.Traject[i].MC_arr[ti], code);
    ReadLn (SaveF, Text);
    Trajects.Traject[i].t_arr[ti]:=tau*(ti)+dep_Start;
  end;
  t_Start[i]:=dep_start;
  Progress:=Progress+1;
  Form1.Gaugel.Progress:=Progress;
end;
if LoadedTrj=True then
begin
  For i:=0 to (Trajects.CntOfTraject-1) do
    Trajects.Traject[i].t_arr[0]:=t_Start[i];
  Finalize(t_Start);
end;

SetLength(Energy_Start, Trajects.CntOfTraject);
SetLength(Energy_Last, Trajects.CntOfTraject);

ReadLn (SaveF, Text);

if Text='Save Energy_arr' then
begin
  SetLength(Energy_arr, Trajects.CntOfTraject, nuzlov);
  FormRasprede.Gauge3.ForeColor:=clMaroon;
  For i:=(Trajects.CntOfTraject-n_tr) to (Trajects.CntOfTraject-1) do begin
    for ti:=0 to nuzlov-1 do begin
      ReadLn (SaveF, Text); Val (Text, Energy_arr[i, ti], code);
    end;
    Energy_Start[i]:=Energy_arr[i, 0]; Energy_Last[i]:=Energy_arr[i, nuzlov-1];
  end;
end;

```

```

    FormRaspredE.Gauge3.Progress:=100;
    FormRaspredE.EnergyReady;
end

else if Text='SaveEnergy-Start-Last' then
begin
    For i:=(Trajects.CntOfTraject-n_tr) to (Trajects.CntOfTraject-1) do begin
        Readln(SaveF,Text); Val(Text,Energy_Start[i],code);
        Readln(SaveF,Text); Val(Text,Energy_Last[i],code);
    end;
end

else
begin
    Application.MessageBox('Т.к. в открываемом файле нет'+#13+#13
        +'данных о потерях энергии'+#13+#13
        +'применяем нулевые начальные'+#13+#13
        +'условия: Energy_arr[i,0]=0',
        'Обратите внимание!',MB_IconInformation);

    if Text='Save-T0-MonteCarlo' then
    begin
        For i:=(Trajects.CntOfTraject-n_tr) to (Trajects.CntOfTraject-1) do begin
            Readln(SaveF,Text); Val(Text,Trajects.Traject[i].tmin0,code);
            Readln(SaveF,Text); Val(Text,Trajects.Traject[i].tmax0,code);
        end;
        goto CloseSaveF;
    end;

end;

Readln(SaveF,Text);
if Text='Save-T0-MonteCarlo' then
begin
    For i:=(Trajects.CntOfTraject-n_tr) to (Trajects.CntOfTraject-1) do begin
        Readln(SaveF,Text); Val(Text,Trajects.Traject[i].tmin0,code);
        Readln(SaveF,Text); Val(Text,Trajects.Traject[i].tmax0,code);
    end;
end
else
begin
    For i:=(Trajects.CntOfTraject-n_tr) to (Trajects.CntOfTraject-1) do begin
        Trajects.Traject[i].tmin0:=dep_Start+dep;
        Trajects.Traject[i].tmax0:=dep_Start+dep;
    end;
end;
CloseSaveF:
    CloseFile(SaveF);
    Form2.Zapolnenie;
    ntr:=StrToInt(Form2.LabeledEdit5.Text);
    LoadedTrj:=True;
    Gauge1.BackColor:=clBtnFace;
    SolveNextBtn.Enabled:=True;
    Str(dep_Start:6:3,Text);      LEGraphT0.Text:=Text;
    Str((dep_Start+dep):6:3,Text); LEGraphTLast.Text:=Text;
end;

procedure TForm1.DechanBtnClick(Sender: TObject);
begin
    FormDechanneling.Visible:=True;
    FormDechanneling.Show;
    FormDechanneling.ChBoxNormDec.Caption:='Нормировать на количество частиц'
        +' ( No = '+IntToStr(Trajects.CntOfTraject)+' )';
    FormDechanneling.LENomTrjDo.Text:=IntToStr(Trajects.CntOfTraject-1);
end;

end.

```

```

unit AddTrjDialog;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Constants, Buttons;

type
  TForm2 = class(TForm)
    LabeledEdit1: TLabeledEdit;
    StaticText1: TStaticText;
    LabeledEdit2: TLabeledEdit;
    StaticText2: TStaticText;
    LabeledEdit3: TLabeledEdit;
    LabeledEdit4: TLabeledEdit;
    LabeledEdit5: TLabeledEdit;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    ListBox1: TListBox;
    Label1: TLabel;
    AddTrOrientBtn: TBitBtn;
    ChBoxNormV: TCheckBox;
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure AddTrOrientBtnClick(Sender: TObject);
    procedure ListBox1Click(Sender: TObject);
    procedure ChBoxNormVClick(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
    procedure Zapolnenie;
  end;
  PForm2=^TForm2;

  TSeriesProperties=record
    x1st,v1st:double;
    x2nd,v2nd:double;
    count:integer;
  end;
  PSeriesProperties=^TSeriesProperties;

  TSerialObj=object
    Serial:array of TSeriesProperties;
    SerialCnt:integer;
    MaxCnt,AllTrCnt:integer;
    Dlg:PForm2;
    constructor Init(cn:integer;d:PForm2);
    destructor Done;
    procedure Add1Series(ser:PSeriesProperties);
    function Execute:boolean;
  end;
  PSerialObj=^TSerialObj;

var
  Form2: TForm2;
  SeriesProperties:TSeriesProperties;
  AllSerial:TSerialObj;

implementation

uses AddTrOrientZav;

{$R *.dfm}

function TSerialObj.Execute;
begin
  Execute:=true;

```

```

end;

procedure TSerialObj.Add1Series(ser:PseriesProperties);
begin
    if (SerialCnt < MaxCnt) then
        begin
            Serial[SerialCnt].count:=ser^.count;
            Serial[SerialCnt].x1st:=ser^.x1st;
            Serial[SerialCnt].v1st:=ser^.v1st;
            Serial[SerialCnt].x2nd:=ser^.x2nd;
            Serial[SerialCnt].v2nd:=ser^.v2nd;
            AllTrCnt:=AllTrCnt+ser^.count;
            inc(SerialCnt);
        end;
    end;

constructor TSerialObj.Init(cn:integer;d:PForm2);
begin
    SerialCnt:=0;
    AllTrCnt:=0;
    Dlg:=d;
    MaxCnt:=cn;
    SetLength(Serial,cn);
end;

destructor TSerialObj.Done;
begin
    Finalize(Serial);
end;

procedure TForm2.BitBtn1Click(Sender: TObject);
var
    s:String;
    c1,c2,c3,c4,c5:integer;
begin
    with SeriesProperties do
        begin
            s:=LabeledEdit1.Text;    val(s,x1st,c1);
            s:=LabeledEdit2.Text;    val(s,v1st,c2);
            s:=LabeledEdit3.Text;    val(s,x2nd,c3);
            s:=LabeledEdit4.Text;    val(s,v2nd,c4);
            s:=LabeledEdit5.Text;    val(s,count,c5);
            ntr:=count;
        end;
    if (c1=0) and (c2=0) and (c3=0) and (c4=0) and (c5=0) then
        begin
            AllSerial.Add1Series(@SeriesProperties);
            Form2.Visible:=false;
        end;

    s:=LabeledEdit5.Text+' tp.   X = '
      +LabeledEdit1.Text+' ... '
      +LabeledEdit3.Text+'   V = '
      +LabeledEdit2.Text+' ... '
      +LabeledEdit4.Text;
    ListBox1.Items.Add(s)
end;

procedure TForm2.BitBtn2Click(Sender: TObject);
begin
    Form2.Visible:=false;
end;

procedure TForm2.Zapolnenie; //процедура заполняет данными форму Form2
begin
    LabeledEdit1.Text:=Settings[1];
    LabeledEdit2.Text:=Settings[2];
    LabeledEdit3.Text:=Settings[3];
    LabeledEdit4.Text:=Settings[4];
    LabeledEdit5.Text:=Settings[5];
    if Settings[91]='NormV=True'

```

```

    then ChBoxNormV.Checked:=True
    else ChBoxNormV.Checked:=False;
    ChBoxNormVClick(self);
end;

procedure TForm2.FormCreate(Sender: TObject);
begin
    Zapolnenie;
    AllSeria.Init(25,@Form2);
end;

procedure TForm2.AddTrOrientBtnClick(Sender: TObject);
begin
    FormAddTrOrientZav.Visible:=True;
    FormAddTrOrientZav.Show;
end;

procedure TForm2.ListBox1Click(Sender: TObject);
var
    Text: String;
begin
    if ListBox1.ItemIndex>(-1) then
    begin
        Text:=ListBox1.Items.Strings[ListBox1.ItemIndex];
        ListBox1.Hint:=Text;
    end;
end;

procedure TForm2.ChBoxNormVClick(Sender: TObject);
begin
    if ChBoxNormV.Checked=False
    then
    begin
        LabeledEdit2.EditLabel.Caption:='Vот =';
        LabeledEdit4.EditLabel.Caption:='Vдо =';
        NormV:=False;
    end
    else
    begin
        LabeledEdit2.EditLabel.Caption:='Vср =';
        LabeledEdit4.EditLabel.Caption:='dV (ср.кв.откл.) =';
        NormV:=True;
    end;
end;

end.

```

```

unit RandomGauss;

interface

uses
    Math;

type
    flp=function (x:double):double;

var
    Sigma_Gauss, a_Gauss, alpha_Gauss, beta_Gauss, epsilon_Gauss: double;
    n_Gauss, itermax: integer;
    PabTrap: double;
    v_Gauss: array of double;
    sigma2_Gauss: double;
    sigma122_Gauss: double; {1/(2*sigma^2)}
    s1sqrt2pi_Gauss: double; {1/sigma*sqrt(2pi)}
    s13sqrt2pi_Gauss: double; {1/(sigma^3)*sqrt(2pi)}
    s15sqrt2pi_Gauss: double; {1/(sigma^5)*sqrt(2pi)}

function RhoGauss(v:double):double;
function RhoGaussX(x:double):double;
function RhoGaussXX(x:double):double;
function NonLinF(x:double;k:integer):double;
function NonLinFEulerMcLoren(x:double;k:integer):double;
function NonLinFx(x:double;k:integer):double;
function NonLinFxEulerMcLoren(x:double;k:integer):double;
function IntegrateTrap(f:flp;a,b:double;nmax:integer):double;
function IntegrateEulerMcLoren(f,fl:flp;a,b:double;nmax:integer):double;
procedure FindVEuler(k:integer);
procedure InitGauss(ntr: integer; a0, sigma0: double);
procedure SolveGauss;
function GetGauss(k: integer): double;
procedure RandomeGauss;

implementation

function RhoGauss(v:double):double;
begin
    RhoGauss:=s1sqrt2pi_Gauss * exp(- sqr(v-a_Gauss)*sigma122_Gauss );
end;

function RhoGaussX(x:double):double;
begin
    RhoGaussX:=(a_Gauss-x)*s13sqrt2pi_Gauss * exp(- sqr(x-a_Gauss)*sigma122_Gauss );
end;

function RhoGaussXX(x:double):double;
begin
    RhoGaussXX:=(Sqr(a_Gauss-x)-sigma2_Gauss)*s15sqrt2pi_Gauss * exp(- sqr(x-
a_Gauss)*sigma122_Gauss );
end;

function NonLinF(x:double;k:integer):double;
begin
    NonLinF:= 0.5*( RhoGauss(v_Gauss[k-1]) + RhoGauss(x) ) *
        ( x - v_Gauss[k-1]) - PabTrap/n_Gauss;
end;

function NonLinFEulerMcLoren(x:double;k:integer):double;
begin
    NonLinFEulerMcLoren:=NonLinF(x,k) + ( RhoGaussX(v_Gauss[k-1]) - RhoGaussX(x) ) *
        Sqr( x - v_Gauss[k-1])/12;
end;

function NonLinFx(x:double;k:integer):double;
begin
    NonLinFx:= 0.5*( RhoGauss(v_Gauss[k-1]) + RhoGauss(x) +
        RhoGaussX(x)*( x - v_Gauss[k-1]));
end;

```

```

function NonLinFxEulerMcLoren(x:double;k:integer):double;
begin
    NonLinFxEulerMcLoren:= NonLinFx(x,k) + ( 2*(RhoGaussX(v_Gauss[k-1]) - RhoGaussX(x))
+RhoGaussXX(x)*( x - v_Gauss[k-1]) )*( x - v_Gauss[k-1])/12;
end;

function IntegrateTrap(f:flp;a,b:double;nmax:integer):double;
var
    i:integer;
    tmpI,dx,x:double;
begin
    dx:=(b-a)/nmax;
    x:=a;
    tmpI:=0;
    for i:=1 to nmax do
        begin
            tmpI:=tmpI+0.5*( f(x)+f(x+dx) ) *dx;
            x:=x+dx;
        end;
    IntegrateTrap:=tmpI;
end;

function IntegrateEulerMcLoren(f,fl:flp;a,b:double;nmax:integer):double;
var
    i:integer;
    tmp,dx,x:double;
begin
    dx:=(b-a)/nmax;
    x:=a+dx;
    tmp:=f(x);
    for i:=2 to nmax-1 do
        begin
            x:=x+dx;
            tmp:=tmp+f(x);
        end;
    IntegrateEulerMcLoren:=tmp+0.5*( f(a)+f(b) )+(fl(a)-fl(b))/12;
end;

procedure FindVEuler(k:integer);
var
    i:integer;
    p,xk:double;
begin
    xk:=v_Gauss[k-1];
    i:=0;
    repeat
        inc(i);
        p:= NonLinFEulerMcLoren(xk,k)/NonLinFxEulerMcLoren(xk,k);
        xk:=xk - p;
    until (i >= itermax) or (abs(p) < epsilon_Gauss);
    v_Gauss[k]:=xk;
end;

procedure InitGauss(ntr: integer; a0, sigma0: double);
begin
    sigma_Gauss := sigma0;
    a_Gauss := a0;
    alpha_Gauss := a_Gauss - 3*sigma_Gauss;
    beta_Gauss := a_Gauss + 3*sigma_Gauss;
    n_Gauss := ntr - 1;

    itermax:=10; {макс. кол-во итераций метода Ньютона}
    epsilon_Gauss:=1E-20;

    SetLength(v_Gauss,(n_Gauss+1));

    Sigma2_Gauss:=Sqr(sigma_Gauss);
    sigma122_Gauss := 1/(2*sigma2_Gauss);
    slsqrt2pi_Gauss := 1/(sigma_Gauss*sqrt(2*pi));
    slsqrt2pi_Gauss := 1/(sigma_Gauss*sigma2_Gauss*sqrt(2*pi));

```



```

s15sqrt2pi_Gauss := 1/(sigma_Gauss*sqr(sigma2_Gauss)*sqrt(2*pi));
PabTrap:=IntegrateTrap(RhoGauss,alpha_Gauss,beta_Gauss,n_Gauss);

SolveGauss;
end;

procedure SolveGauss;
var
  k: integer;
begin
  v_Gauss[0]:=alpha_Gauss;
  for k:=1 to n_Gauss do
    begin
      FindVEuler(k);
    end;
  end;
  RandomGauss;
end;

function GetGauss(k: integer): double;
begin
  Result:=v_Gauss[k];
end;

procedure RandomGauss;
var
  i, nn: integer;
  v1: double;
begin
  Randomize;
  for i:=0 to n_Gauss do
    begin
      nn:=Round(RandomRange(0,n_Gauss+1));
      v1:=v_Gauss[nn];
      v_Gauss[nn]:=v_Gauss[i];
      v_Gauss[i]:=v1;
    end;
  end;
end.

```

```

unit MonteCarloDlg;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, AddTrjDialog, Constants;

type
  TMonteCarloDlgForm = class(TForm)
    Label1: TLabel;
    LabeledEdit1: TLabeledEdit;
    DoMonteCarloBtn: TBitBtn;
    DoNoMonteCarloBtn: TBitBtn;
    CloseBtn: TBitBtn;
    LabeledEdit2: TLabeledEdit;
    LabeledEdit3: TLabeledEdit;
    Label2: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    ChBoxNoExceedMmax: TCheckBox;
    ChBoxCombineRozigrish: TCheckBox;
    Bevel1: TBevel;
    Bevel2: TBevel;
    Bevel3: TBevel;
    Label10: TLabel;
    ChBoxUxx0: TCheckBox;
    LabeledEdit4: TLabeledEdit;
    Label11: TLabel;
    Label4: TLabel;
    Label3: TLabel;
    ChBoxTAngTau: TCheckBox;
    Bevel5: TBevel;
    Label12: TLabel;
    Label13: TLabel;
    ChBoxUxx3D: TCheckBox;
    procedure DoMonteCarloBtnClick(Sender: TObject);
    procedure DoNoMonteCarloBtnClick(Sender: TObject);
    procedure CloseBtnClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure ChBoxUxx0Click(Sender: TObject);
    procedure ChBoxTAngTauClick(Sender: TObject);
    procedure LabeledEdit1Change(Sender: TObject);
    procedure LabeledEdit4Change(Sender: TObject);
    procedure ChBoxUxx3DClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure Zapolnenie;
  end;

var
  MonteCarloDlgForm: TMonteCarloDlgForm;

implementation

uses Trajectories, RaspredV;

{$R *.dfm}

procedure TMonteCarloDlgForm.DoMonteCarloBtnClick(Sender: TObject);
var
  cnt, code: integer;
  dep: double;
  s: String;
begin

```

```

s:=Form1.LabeledEdit1.Text;      Val(s,dep,code);
s:=Form1.LabeledEdit2.Text;      Val(s,cnt,code);

dep_Start:=0;

Str(dep_Start:6:3,s);      Form1.LEGraphT0.Text:=s;
Str(dep:6:3,s);      Form1.LEGraphTLast.Text:=s;

SolveSys5DE:=True;
SolveMonteCarlo:=True;

if MonteCarloDlgForm.ChBoxNoExceedMmax.Checked=True
then NoExceedMmax:=True
else NoExceedMmax:=False;

if MonteCarloDlgForm.ChBoxCombineRozigrish.Checked=True
then CombineRozigrish:=True else CombineRozigrish:=False;

if MonteCarloDlgForm.ChBoxUxx3D.Checked=False
then UxxForSolver3D:=False else UxxForSolver3D:=True;

if MonteCarloDlgForm.ChBoxUxx0.Checked=False
then UxxForSolver0:=False else UxxForSolver0:=True;

s:=LabeledEdit1.Text;
Val(s,Tmax_MonteCarlo,code);      if code<>0 then exit;
if ChBoxTAngTau.Checked=True
then Tmax_MonteCarlo:=Tmax_MonteCarlo*epsilon/d_small;
s:=LabeledEdit4.Text;
Val(s,Tmin_MonteCarlo,code);      if code<>0 then exit;
if ChBoxTAngTau.Checked=True
then Tmin_MonteCarlo:=Tmin_MonteCarlo*epsilon/d_small;
s:=LabeledEdit2.Text;      Val(s,MA_max,code);      if code<>0 then exit;
s:=LabeledEdit3.Text;      Val(s,MC_max,code);      if code<>0 then exit;
FormRaspredV.LabelMamax.Caption:='MA_max = '+FloatToStr(MA_max);
FormRaspredV.LabelMCmax.Caption:='MC_max = '+FloatToStr(MC_max);
MonteCarloDlgForm.Visible:=False;
Form1.Gaugel.BackColor:=clYellow;      Form1.Gaugel.ForeColor:=clBlue;
Form1.Gaugel.Progress:=0;
StartTime:=Now;
Form1.ReportMemo.Lines.Add('['+TimeToStr(Now)+']      Расчёт начал
'+DateTimeToStr(StartTime));
Form1.ReportMemo.Repaint;
Trajects.Init(@AllSeria,dep_Start,dep,cnt);
Trajects.SolveAll;
Form1.Gaugel.BackColor:=clBtnFace;
Finalize(Energy_Start);      Finalize(Energy_Last);
SetLength(Energy_Start,Trajects.CntOfTraject);
SetLength(Energy_Last,Trajects.CntOfTraject);
Form1.SolveNextBtn.Enabled:=True;
if AutoSolve=True then AutomaticSolve;
WorkTime:=Now-StartTime;
Form1.ReportMemo.Lines.Add('['+TimeToStr(Now)+']      Расчёт окончен, выполнен за
время '+TimeToStr(WorkTime));
end;

procedure TMonteCarloDlgForm.DoNoMonteCarloBtnClick(Sender: TObject);
var
cnt,code: integer;
dep: double;
s: String;
begin
s:=Form1.LabeledEdit1.Text;      Val(s,dep,code);
s:=Form1.LabeledEdit2.Text;      Val(s,cnt,code);
dep_Start:=0;
Str(dep_Start:6:3,s);      Form1.LEGraphT0.Text:=s;
Str(dep:6:3,s);      Form1.LEGraphTLast.Text:=s;
SolveSys5DE:=True;      SolveMonteCarlo:=False;
if MonteCarloDlgForm.ChBoxUxx3D.Checked=False
then UxxForSolver3D:=False else UxxForSolver3D:=True;

```

```

if ChBoxUxx0.Checked=False
  then UxxForSolver0:=False else UxxForSolver0:=True;
  MonteCarloDlgForm.Visible:=False;
  Form1.Gauge1.BackColor:=clYellow;   Form1.Gauge1.ForeColor:=clBlue;
  Form1.Gauge1.Progress:=0;
  StartTime:=Now;
  Form1.ReportMemo.Lines.Add(' [' +TimeToStr(Now)+' ]   Расчёт начат
'+DateTimeToStr(StartTime));
  Form1.ReportMemo.Repaint;
  Trajects.Init(@AllSeria,dep_Start,dep,cnt);
  Trajects.SolveAll;
  Form1.Gauge1.BackColor:=clBtnFace;
  Finalize(Energy_Start);
  Finalize(Energy_Last);
  SetLength(Energy_Start,Trajects.CntOfTraject);
  SetLength(Energy_Last,Trajects.CntOfTraject);
  Form1.SolveNextBtn.Enabled:=True;
  if AutoSolve=True then AutomaticSolve;
  WorkTime:=Now-StartTime;
  Form1.ReportMemo.Lines.Add(' [' +TimeToStr(Now)+' ]   Расчёт окончен, выполнен за
время ' +TimeToStr(WorkTime));
  end;

procedure TMonteCarloDlgForm.CloseBtnClick(Sender: TObject);
begin
  MonteCarloDlgForm.Close;
end;

procedure TMonteCarloDlgForm.FormShow(Sender: TObject);
var
  Text: String;
begin
  Label9.Caption:=' ' +FloatToStr(Le);
  if LnuclGemmell=True
    then Text:=' [Gemmell] '
    else Text:=' [2*ln(183/Z2^(1/3))] ' ;
  Label10.Caption:='Lnucl ' +Text+ ' = ' +FloatToStr(Lnucl);
end;

procedure TMonteCarloDlgForm.Zapolnenie; //процедура заполняет данными форму
begin
  LabeledEdit1.Text:=Settings[26];
  LabeledEdit2.Text:=Settings[27];
  LabeledEdit3.Text:=Settings[28];
  if Settings[62]='ChBoxNoExceedMmax=True'
    then ChBoxNoExceedMmax.Checked:=True
    else ChBoxNoExceedMmax.Checked:=False;
  if Settings[63]='ChBoxCombineRozigrish=False'
    then ChBoxCombineRozigrish.Checked:=False
    else ChBoxCombineRozigrish.Checked:=True;
  if Settings[83]='UxxForSolver0=True'
    then ChBoxUxx0.Checked:=True
    else ChBoxUxx0.Checked:=False;
  LabeledEdit4.Text:=Settings[84];
  if Settings[85]='ChBoxTAngTau=False'
    then ChBoxTAngTau.Checked:=False
    else ChBoxTAngTau.Checked:=True;
  if Settings[87]='UxxForSolver3D=True'
    then ChBoxUxx3D.Checked:=True
    else ChBoxUxx3D.Checked:=False;
end;

procedure TMonteCarloDlgForm.FormCreate(Sender: TObject);
begin
  Zapolnenie;
end;

procedure TMonteCarloDlgForm.ChBoxUxx0Click(Sender: TObject);
begin
  if ChBoxUxx0.Checked=False then
    begin

```

```

    UxxForSolver0:=False;
    if UxxForSolver3D=False
    then Label5.Caption:='Uxx = U_xx (x)' else Label5.Caption:='Uxx = U_xx (x,T)';
    end
else
begin
    UxxForSolver0:=True; Label5.Caption:='Uxx = 0';
end;
end;

procedure TMonteCarloDlgForm.ChBoxUxx3DClick(Sender: TObject);
begin
    if ChBoxUxx3D.Checked=False then
    begin
        UxxForSolver3D:=False;
        if UxxForSolver0=False then Label5.Caption:='Uxx = U_xx (x)';
        end
    else
    begin
        UxxForSolver3D:=True;
        if UxxForSolver0=False then Label5.Caption:='Uxx = U_xx (x,T)';
        end;
    end;
end;

procedure TMonteCarloDlgForm.ChBoxTAngTauClick(Sender: TObject);
begin
    if ChBoxTAngTau.Checked=False then
    begin
        Label8.Caption:='tay'; Label11.Caption:='tay';
    end
    else
    begin
        Label8.Caption:='AHT.'; Label11.Caption:='AHT.';
    end;
    LabeledEdit1Change(self); LabeledEdit4Change(self);
end;

procedure TMonteCarloDlgForm.LabeledEdit1Change(Sender: TObject);
var
    Doub: double;
    Code: integer;
    S: String;
begin
    S:=LabeledEdit1.Text;
    Val(S,Doub,Code);
    if (code<>0) then exit;
    if ChBoxTAngTau.Checked=False
    then Label8.Caption:='tay = '+FloatToStrF(Doub*d_small/epsilon,ffGeneral,6,3)+'
AHT.'
    else Label8.Caption:='AHT. = '+FloatToStrF(Doub*epsilon/d_small,ffGeneral,6,3)+'
tay';
end;

procedure TMonteCarloDlgForm.LabeledEdit4Change(Sender: TObject);
var
    Doub: double;
    Code: integer;
    S: String;
begin
    S:=LabeledEdit4.Text;
    Val(S,Doub,Code);
    if (code<>0) then exit;
    if ChBoxTAngTau.Checked=False
    then Label11.Caption:='tay = '+FloatToStrF(Doub*d_small/epsilon,ffGeneral,6,3)+'
AHT.'
    else Label11.Caption:='AHT. = '+FloatToStrF(Doub*epsilon/d_small,ffGeneral,6,3)+'
tay';
end;
end.

```

```

unit Solver;

interface

uses
    Constants, Math;

type
    DbArr=array of double;

    OneTrajectory=object
        t_arr, v_arr, x_arr,
        MA_arr, MB_arr, MC_arr: DbArr;
        t_Start, t_Depth,
        v_Start, x_Start: double;
        NodeCount: integer;
        kx, kv, kMA, kMB, kMC: array[1..4] of double;
        Solved, CanSolve: boolean;
        tau: double;
        tmin0, tmax0: double;
        n, Nmax: integer;
        constructor Init(start,depth:double;NodeCnt:integer;vst,xst:double);
        destructor Done;
        function RightForV(t,v,x:double):double;
        function RightForX(t,v,x:double):double;
        function RightFormA(a,b,c,x,t:double):double;
        function RightFormB(a,b,c,x,t:double):double;
        function RightFormC(a,b,c,x,t:double):double;
        Procedure Solve;
    end;
    Procedure CalcUxxD(x:double);

var
    Uxx_sol, Diff_sol, De_solME: double;

implementation

constructor OneTrajectory.Init(start,depth:double;NodeCnt:integer;vst,xst:double);
begin
    t_Start:=start;    t_Depth:=depth;
    NodeCount:=NodeCnt;
    v_Start:=vst;    x_Start:=xst;
    Solved:=false;
    SetLength(t_arr,NodeCount);    SetLength(x_arr,NodeCount);
    SetLength(v_arr,NodeCount);    SetLength(MA_arr,NodeCount);
    SetLength(MB_arr,NodeCount);    SetLength(MC_arr,NodeCount);
    Nmax:=NodeCount-1;
    tau:=(t_Depth-t_Start)/Nmax;
    v_arr[0]:=v_Start;
    x_arr[0]:=x_Start;
    t_arr[0]:=t_Start;
    MA_arr[0]:=0;    MB_arr[0]:=0;    MC_arr[0]:=0;
    n:=0;
    tmin0:=t_Start;
    tmax0:=t_Start;
    CanSolve:=true;
end;

destructor OneTrajectory.Done;
begin
    Finalize(t_arr);    Finalize(x_arr);    Finalize(v_arr);
    Finalize(MA_arr);    Finalize(MB_arr);    Finalize(MC_arr);
end;

function OneTrajectory.RightForV(t,v,x:double):double;
begin
    RightForV:=-Znak_Z1*Usp_x(x,t)/Vmax+Curv+kt*t;
end;

function OneTrajectory.RightForX(t,v,x:double):double;
begin

```

```

    RightForX:=v;
end;

function OneTrajectory.RightFormA(a,b,c,x,t:double):double;
begin
    RightFormA:=2*b;
end;

function OneTrajectory.RightFormB(a,b,c,x,t:double):double;
begin
    RightFormB:=c-Uxx_sol/Vmax*a;
end;

function OneTrajectory.RightFormC(a,b,c,x,t:double):double;
begin
    RightFormC:=Diff_sol*CoefFormC-2*Uxx_sol/Vmax*b;
end;

Procedure OneTrajectory.Solve;
begin
    if SolveSys5DE=False then //Решение без вторых моментов (только ур-е. движения)
    begin
        While (n <= (Nmax-1)) and CanSolve do
        begin
            t_arr[n+1]:=t_arr[n]+tau;

            kv[1]:=RightForV(t_arr[n],v_arr[n],x_arr[n]);
            kx[1]:=RightForX(t_arr[n],v_arr[n],x_arr[n]);

            kv[2]:=RightForV(t_arr[n]+tau/2,v_arr[n]+tau*kv[1]/2,x_arr[n]+tau*kx[1]/2);
            kx[2]:=RightForX(t_arr[n]+tau/2,v_arr[n]+tau*kv[1]/2,x_arr[n]+tau*kx[1]/2);

            kv[3]:=RightForV(t_arr[n]+tau/2,v_arr[n]+tau*kv[2]/2,x_arr[n]+tau*kx[2]/2);
            kx[3]:=RightForX(t_arr[n]+tau/2,v_arr[n]+tau*kv[2]/2,x_arr[n]+tau*kx[2]/2);

            kv[4]:=RightForV(t_arr[n+1],v_arr[n]+tau*kv[3],x_arr[n]+tau*kx[3]);
            kx[4]:=RightForX(t_arr[n+1],v_arr[n]+tau*kv[3],x_arr[n]+tau*kx[3]);

            v_arr[n+1]:=v_arr[n]+tau*(kv[1]+2*kv[2]+2*kv[3]+kv[4])/6;
            x_arr[n+1]:=x_arr[n]+tau*(kx[1]+2*kx[2]+2*kx[3]+kx[4])/6;

            MA_arr[n+1]:=0; MB_arr[n+1]:=0; MC_arr[n+1]:=0;

            inc(n);
        end;
        if (n=Nmax) then Solved:=true;
    end
else {if SolveSys5DE=True;} //Решение 5 ур-ний. (на X, V и вторые моменты)
begin
    Randomize;
    While (n <= (Nmax-1)) and CanSolve do
    begin
        t_arr[n+1]:=t_arr[n]+tau;

        kv[1]:=RightForV(t_arr[n],v_arr[n],x_arr[n]);
        kx[1]:=RightForX(t_arr[n],v_arr[n],x_arr[n]);

        kv[2]:=RightForV(t_arr[n]+tau/2,v_arr[n]+tau*kv[1]/2,x_arr[n]+tau*kx[1]/2);
        kx[2]:=RightForX(t_arr[n]+tau/2,v_arr[n]+tau*kv[1]/2,x_arr[n]+tau*kx[1]/2);

        kv[3]:=RightForV(t_arr[n]+tau/2,v_arr[n]+tau*kv[2]/2,x_arr[n]+tau*kx[2]/2);
        kx[3]:=RightForX(t_arr[n]+tau/2,v_arr[n]+tau*kv[2]/2,x_arr[n]+tau*kx[2]/2);

        kv[4]:=RightForV(t_arr[n+1],v_arr[n]+tau*kv[3],x_arr[n]+tau*kx[3]);
        kx[4]:=RightForX(t_arr[n+1],v_arr[n]+tau*kv[3],x_arr[n]+tau*kx[3]);

        CalcUxxD(x_arr[n]);

        kMA[1]:=RightFormA(MA_arr[n],MB_arr[n],MC_arr[n],x_arr[n],t_arr[n]);
        kMB[1]:=RightFormB(MA_arr[n],MB_arr[n],MC_arr[n],x_arr[n],t_arr[n]);
    end
end
end

```

```

kMC[1]:=RightFormC(MA_arr[n],MB_arr[n],MC_arr[n],x_arr[n],t_arr[n]);

CalcUxxD(x_arr[n]+tau*kx[1]/2);

kMA[2]:=RightFormA(MA_arr[n]+tau*kMA[1]/2,MB_arr[n]+tau*kMB[1]/2,
    MC_arr[n]+tau*kMC[1]/2,x_arr[n]+tau*kx[1]/2,t_arr[n]+tau/2);
kMB[2]:=RightFormB(MA_arr[n]+tau*kMA[1]/2,MB_arr[n]+tau*kMB[1]/2,
    MC_arr[n]+tau*kMC[1]/2,x_arr[n]+tau*kx[1]/2,t_arr[n]+tau/2);
kMC[2]:=RightFormC(MA_arr[n]+tau*kMA[1]/2,MB_arr[n]+tau*kMB[1]/2,
    MC_arr[n]+tau*kMC[1]/2,x_arr[n]+tau*kx[1]/2,t_arr[n]+tau/2);

CalcUxxD(x_arr[n]+tau*kx[2]/2);

kMA[3]:=RightFormA(MA_arr[n]+tau*kMA[2]/2,MB_arr[n]+tau*kMB[2]/2,
    MC_arr[n]+tau*kMC[2]/2,x_arr[n]+tau*kx[2]/2,t_arr[n]+tau/2);
kMB[3]:=RightFormB(MA_arr[n]+tau*kMA[2]/2,MB_arr[n]+tau*kMB[2]/2,
    MC_arr[n]+tau*kMC[2]/2,x_arr[n]+tau*kx[2]/2,t_arr[n]+tau/2);
kMC[3]:=RightFormC(MA_arr[n]+tau*kMA[2]/2,MB_arr[n]+tau*kMB[2]/2,
    MC_arr[n]+tau*kMC[2]/2,x_arr[n]+tau*kx[2]/2,t_arr[n]+tau/2);

CalcUxxD(x_arr[n]+tau*kx[3]);

kMA[4]:=RightFormA(MA_arr[n]+tau*kMA[3],MB_arr[n]+tau*kMB[3],
    MC_arr[n]+tau*kMC[3],x_arr[n]+tau*kx[3],t_arr[n+1]);
kMB[4]:=RightFormB(MA_arr[n]+tau*kMA[3],MB_arr[n]+tau*kMB[3],
    MC_arr[n]+tau*kMC[3],x_arr[n]+tau*kx[3],t_arr[n+1]);
kMC[4]:=RightFormC(MA_arr[n]+tau*kMA[3],MB_arr[n]+tau*kMB[3],
    MC_arr[n]+tau*kMC[3],x_arr[n]+tau*kx[3],t_arr[n+1]);

v_arr[n+1]:=v_arr[n]+tau*(kv[1]+2*kv[2]+2*kv[3]+kv[4])/6;
x_arr[n+1]:=x_arr[n]+tau*(kx[1]+2*kx[2]+2*kx[3]+kx[4])/6;

MA_arr[n+1]:=MA_arr[n]+tau*(kMA[1]+2*kMA[2]+2*kMA[3]+kMA[4])/6;
MB_arr[n+1]:=MB_arr[n]+tau*(kMB[1]+2*kMB[2]+2*kMB[3]+kMB[4])/6;
MC_arr[n+1]:=MC_arr[n]+tau*(kMC[1]+2*kMC[2]+2*kMC[3]+kMC[4])/6;

if (SolveMonteCarlo=True) and ((t_arr[n+1]-tmin0)>=Tmin_MonteCarlo) then
begin //С розыгрышем, но розыгрыш работает если
    // после предыдущего розыгрыша пройдено Tmin_MonteCarlo

if NoExceedMmax=True then //Розыгрыш при превышении M_max,
begin //независимо от глубины Tmax_MonteCarlo
    if CombineRozigrish=True then //При выполнении одного из условий
    begin //переопределяются x, v, MA, MB
        if (MA_arr[n+1]>=MA_max) or (MC_arr[n+1]>=MC_max) then
        begin
            x_arr[n+1]:=RandG(x_arr[n+1],Sqrt(abs(MA_arr[n+1])));
            v_arr[n+1]:=RandG(v_arr[n+1],Sqrt(abs(MC_arr[n+1])));
            {Функция RandG(среднее значение, ср.кв.откл.) возвращает
             случайное число из гауссового распределения}
            MA_arr[n+1]:=0; MB_arr[n+1]:=0; MC_arr[n+1]:=0;
            tmin0:=t_arr[n+1]; tmax0:=t_arr[n+1];
        end;
    end
else // if CombineRozigrish=False
    begin
        if (MA_arr[n+1]>=MA_max) then
        begin
            x_arr[n+1]:=RandG(x_arr[n+1],Sqrt(MA_arr[n+1]));
            MA_arr[n+1]:=0; MB_arr[n+1]:=0; MC_arr[n+1]:=0;
            tmin0:=t_arr[n+1]; tmax0:=t_arr[n+1];
        end;
        if (MC_arr[n+1]>=MC_max) then
        begin
            v_arr[n+1]:=RandG(v_arr[n+1],Sqrt(MC_arr[n+1]));
            MA_arr[n+1]:=0; MB_arr[n+1]:=0; MC_arr[n+1]:=0;
            tmin0:=t_arr[n+1]; tmax0:=t_arr[n+1];
        end;
    end;
end;

```



```

if (t_arr[n+1]-tmax0)>=Tmax_MonteCarlo then //Розыгрыш по глубине:
  begin
    tmin0:=t_arr[n+1];
    tmax0:=t_arr[n+1];

    if (MA_arr[n+1]<MA_max) and (MC_arr[n+1]<MC_max) then
      begin // Розыгрыш по X и V без обнуления МА,МВ,МС:
        x_arr[n+1]:=RandG(x_arr[n+1],Sqrt(abs(MA_arr[n+1])));
        v_arr[n+1]:=RandG(v_arr[n+1],Sqrt(abs(MC_arr[n+1])));
      end;

    if CombineRozigrish=True then
      begin
        if (MA_arr[n+1]>=MA_max) or (MC_arr[n+1]>=MC_max) then
          begin
            x_arr[n+1]:=RandG(x_arr[n+1],Sqrt(abs(MA_arr[n+1])));
            v_arr[n+1]:=RandG(v_arr[n+1],Sqrt(abs(MC_arr[n+1])));
            MA_arr[n+1]:=0; MB_arr[n+1]:=0; MC_arr[n+1]:=0;
          end;
        end
      else // if CombineRozigrish=False
        begin
          if (MA_arr[n+1]>=MA_max) then
            begin
              x_arr[n+1]:=RandG(x_arr[n+1],Sqrt(MA_arr[n+1]));
              MA_arr[n+1]:=0; MB_arr[n+1]:=0; MC_arr[n+1]:=0;
            end;
          if (MC_arr[n+1]>=MC_max) then
            begin
              v_arr[n+1]:=RandG(v_arr[n+1],Sqrt(MC_arr[n+1]));
              MA_arr[n+1]:=0; MB_arr[n+1]:=0; MC_arr[n+1]:=0;
            end;
          end;
        end; // конец розыгрыша по глубине
      end; // конец розыгрыша
      inc(n);
    end;
    if (n=Nmax) then Solved:=true;
  end; //Конец совместного решения уравнений движения и вторых моментов

end;

Procedure CalcUxxD(x:double);
var
  xxj, Fcos, FUxx, Fre, Frn: double;
  n, nMin, nMax, j: integer;
begin
  nMin:=nxRange+1; {nMin -> nx=1}
  nMax:=2*nxRange; {nMax -> nx=+nxRange}
  FUxx:=0; Fre:=0; Frn:=0;
  for j:=1 to 8 do begin
    xxj:=x-xdax[j];
    for n:=nMin to nMax do begin
      Fcos:=cos(Pi2nx[n]*xxj);
      FUxx:=FUxx-VxUxx[n]*Fcos;
      Fre:=Fre+Fgx2exp[n]*Fcos;
      Frn:=Frn+expSgx2[n]*Fcos;
    end;
  end;
  if UxxForSolver0=true then Uxx_sol:=0
  else Uxx_sol:=2*FUxx/d3;
  Fre:=2*Fre+Fgx2exp[nxRange]*8;
  Frn:=2*Frn+8;
  Diff_sol:=Le_d*Fre+Lnuc1_d*Frn;
  Diff_sol:=Diff_sol/d3;
end;

end.

```

```

unit GraphU;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, TeEngine, Series, ExtCtrls, TeeProcs, Chart, StdCtrls, Buttons,
  Constants, Trajectories, ExtDlgs, Menus;

type
  TFormGraphU = class(TForm)
    BitBtn1: TBitBtn;
    Chart1: TChart;
    Series1: TLineSeries;
    CBGraph: TComboBox;
    LabelGrafik: TLabel;
    LEXot: TLabeledEdit;
    LEXdo: TLabeledEdit;
    LESTepX: TLabeledEdit;
    LETst: TLabeledEdit;
    Label1: TLabel;
    ListBox1: TListBox;
    LabelIntRes: TLabel;
    PopupMenuGraphU: TPopupMenu;
    ShowGrid: TMenuItem;
    ShowText: TMenuItem;
    N3: TMenuItem;
    SaveGraph: TMenuItem;
    PrintGraph: TMenuItem;
    ToClipBrd: TMenuItem;
    SerferUxtBtn: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure ShowGridClick(Sender: TObject);
    procedure ShowTextClick(Sender: TObject);
    procedure SaveGraphClick(Sender: TObject);
    procedure PrintGraphClick(Sender: TObject);
    procedure ToClipBrdClick(Sender: TObject);
    procedure FormDechannelingExcellClick(Sender: TObject);
    procedure SerferUxtBtnClick(Sender: TObject);
  private
    { Private declarations }
    function GraphFunction(x,T:double):double;
    procedure WriteCoordinates;
  public
    { Public declarations }
    procedure Zapolnenie;
  end;

var
  FormGraphU: TFormGraphU;

implementation

{$R *.dfm}

uses ComObj;

procedure TFormGraphU.WriteCoordinates;
var
  i: integer;
  Si, Sx, Sy, Sz, Sa, Sl, SAppr, SLnucl: String;
  s_i, s_x, s_y, s_z: array [1..8] of String;
begin
  ListBox1.Items.Clear;

  Si:=' j '; Sx:='x_j '; Sy:='y_j '; Sz:='z_j ';
For i:=1 to 8 do
    begin
      Str(i:10,s_i[i]); Si:=Si+' | '+s_i[i];
      Str(x_stj[i]:8:4,s_x[i]); Sx:=Sx+' | '+s_x[i];
    
```

```

Str(y_stj[i]:8:4,s_y[i]); Sy:=Sy+' | '+s_y[i];
Str(z_stj[i]:8:4,s_z[i]); Sz:=Sz+' | '+s_z[i];
end;

Str(ax:7:4,Sa); Sx:=Sx+' || ax = '+Sa;
Str(ay:7:4,Sa); Sy:=Sy+' || ay = '+Sa;
Str(az:7:4,Sa); Sz:=Sz+' || az = '+Sa;

if Approximation='Molier'
then SAppr:=' > Приближение Мольер'
else SAppr:=' > Приближение Дойля-Тернера';
if Lnuc1Gemmell=True
then SLnuc1:=' | Lnuc1 по Геммеллу (Бору)'
else SLnuc1:=' | Lnuc1=2ln[183/Z2^(1/3)]';

S1:=' Плоскость '+Ploskost+' < k = '+IntToStr(k)+'; l = '+IntToStr(l)
+SAppr+SLnuc1;

ListBox1.Items.Add(S1); ListBox1.Items.Add(Si); ListBox1.Items.Add(Sx);
ListBox1.Items.Add(Sy); ListBox1.Items.Add(Sz);

end;

function TFormGraphU.GraphFunction(x,T:double):double;
begin
case CBGraph.ItemIndex of
1: GraphFunction:=U_x(x)/ax;
2: GraphFunction:=U_xx(x)/sqr(ax);
3: GraphFunction:=ro_e(x);
4: GraphFunction:=ro_nucl(x);
5: GraphFunction:=Diffuz_e(x)*sqr(sqr(epsilon))*1E+16/sqr(Vmax);
6: GraphFunction:=Diffuz_nucl(x)*sqr(sqr(epsilon))*1E+16/sqr(Vmax);
7: GraphFunction:=Diffuz(x)*sqr(sqr(epsilon))*1E+16/sqr(Vmax);
8: GraphFunction:=Usp(x,T);
9: GraphFunction:=Usp_x(x,T)/ax;
10: GraphFunction:=Usp_xx(x,T)/sqr(ax);
11: GraphFunction:=UxxForSolver(x,T);
else GraphFunction:=U(x);
end;
end;

procedure TFormGraphU.BitBtn1Click(Sender: TObject);
var
sotx, sdox, sdoT, sstepx: String;
otx, dox, stepx, x, Integral, T: double;
code: integer;
begin
WriteCoordinates;
Series1.Clear;
sotx:=LEXot.Text; Val(sotx,otx,code);
sdox:=LEXdo.Text; Val(sdox,dox,code);
sstepx:=LEStepX.Text; Val(sstepx,stepx,code);
sdoT:=LETst.Text; Val(sdoT,T,code);
x:=otx;
case CBGraph.ItemIndex of
1: begin
Chart1.Title.Text.Text:='Напряжённость электрического поля';
Chart1.LeftAxis.Title.Caption:='Ux(x), эВ/Анг';
end;
2: begin
Chart1.Title.Text.Text:='График второй производной Uxx(x)';
Chart1.LeftAxis.Title.Caption:='Uxx(x), эВ/Анг^2';
end;
3: begin
Chart1.Title.Text.Text:='График электронной плотности ro_e(x)';
Chart1.LeftAxis.Title.Caption:='ro_e(x)';
end;
4: begin
Chart1.Title.Text.Text:='График ядерной плотности ro_nucl(x)';
Chart1.LeftAxis.Title.Caption:='ro_nucl(x)';
end;
end;
end;

```

```

5: begin
  Chart1.Title.Text.Text:='График электронного коэф-та диффузии Diffuz_e(x)';
  Chart1.LeftAxis.Title.Caption:='Diffuz_e(x), мкрад^2 / мкм';
  end;
6: begin
  Chart1.Title.Text.Text:='График ядерного коэф-та диффузии Diffuz_nucl(x)';
  Chart1.LeftAxis.Title.Caption:='Diffuz_nucl(x), мкрад^2 / мкм';
  end;
7: begin
  Chart1.Title.Text.Text:='График суммарного коэф-та диффузии Diffuz(x)';
  Chart1.LeftAxis.Title.Caption:='Diffuz(x), мкрад^2 / мкм';
  end;
8: begin
  Chart1.Title.Text.Text:='Сечение трёхмерного потенциала U(x,T) на глубине T';
  Chart1.LeftAxis.Title.Caption:='U(x,T)';
  end;
9: begin
  Chart1.Title.Text.Text:='Сечение первой производной Usp_x(x,T) на глубине T';
  Chart1.LeftAxis.Title.Caption:='Usp_x(x,T), эВ/Анг';
  end;
10: begin
  Chart1.Title.Text.Text:='Сечение второй производной Usp_xx(x,T) на глубине T';
  Chart1.LeftAxis.Title.Caption:='Usp_xx(x,T), эВ/Анг^2';
  end;
11: begin
  Chart1.Title.Text.Text:='Сечение Uxx(x,T) [для вторых моментов] на глубине T';
  Chart1.LeftAxis.Title.Caption:='Uxx(x,T) [для вторых моментов]';
  end;
else begin
  Chart1.Title.Text.Text:='График потенциала U(x)';
  Chart1.LeftAxis.Title.Caption:='U(x), эВ';
  end;
end;
  Integral:=0;
  While x<=dox do begin
    Series1.AddXY(x,GraphFunction(x,T),'',clBlack);
    Integral:=Integral+GraphFunction(x,T);
    x:=x+stepx;
  end;
  LabelIntRes.Caption:='Площадь под графиком = '+#13+FloatToStr(Integral*stepx);
end;

procedure TFormGraphU.Zapolnenie; //процедура заполняет данными форму
begin
  CBGraph.ItemIndex:=StrToIntDef(Settings[76],0);
  LEXot.Text:=Settings[77]; LEXdo.Text:=Settings[78];
  LESTepX.Text:=Settings[79]; LETst.Text:=Settings[80];
end;

procedure TFormGraphU.FormCreate(Sender: TObject);
begin
  Zapolnenie;
end;

procedure TFormGraphU.ShowGridClick(Sender: TObject);
begin
  if ShowGrid.Checked=False then
  begin
    Chart1.LeftAxis.Grid.Visible:=True; Chart1.BottomAxis.Grid.Visible:=True;
    ShowGrid.Checked:=True;
  end
  else
  begin
    Chart1.LeftAxis.Grid.Visible:=False; Chart1.BottomAxis.Grid.Visible:=False;
    ShowGrid.Checked:=False;
  end;
end;

procedure TFormGraphU.ShowTextClick(Sender: TObject);
begin
  if ShowText.Checked=False then

```

```

    begin
        Chart1.Foot.Visible:=True;    Chart1.Title.Visible:=True;
        ShowText.Checked:=True;
    end
else
    begin
        Chart1.Foot.Visible:=False;    Chart1.Title.Visible:=False;
        ShowText.Checked:=False;
    end;
end;

procedure TFormGraphU.SaveGraphClick(Sender: TObject);
begin
    if Form1.SavePictureDialog1.Execute then
        begin
            Chart1.SaveToMetafileEnh(Form1.SavePictureDialog1.FileName);
        end;
end;

procedure TFormGraphU.PrintGraphClick(Sender: TObject);
begin
    if Form1.PrintDialog1.Execute then Chart1.Print;
end;

procedure TFormGraphU.ToClipBrdClick(Sender: TObject);
begin
    Chart1.CopyToClipboardMetafile(true);
end;

procedure TFormGraphU.FormDechannelingExcel1Click(Sender: TObject);
var
    i: integer;
    ExcelApplication: Variant;
begin
    ExcelApplication:=CreateOleObject ('Excel.Application');
    ExcelApplication.Visible:=True;    ExcelApplication.WorkBooks.Add;

    for i:=1 to Series1.Count do
        begin
            ExcelApplication.Cells[i,1].Value:=Series1.XValue[i-1];
            ExcelApplication.Cells[i,2].Value:=Series1.YValue[i-1];
        end;
end;

procedure TFormGraphU.SerferUxtBtnClick(Sender: TObject);
var
    StepX,StepY,CurrentX,CurrentY: double;
    DatFileT: String;
    DatFile: TextFile;
    i,j: integer;
begin
    Form1.SaveDialog1.DefaultExt:='dat';
    Form1.SaveDialog1.Filter:= 'Файлы dat (*.dat)|*.DAT';
    if Form1.SaveDialog1.Execute then
        begin
            DatFileT:=Form1.SaveDialog1.FileName;
            AssignFile(DatFile,DatFileT);
            ReWrite(DatFile);
            StepX:=1E-2;    StepY:=1E-2;
            for i:=0 to 99 do begin
                CurrentX:=i*StepX;
                for j:=0 to 99 do begin
                    CurrentY:=j*StepY;
                    WriteLn(DatFile,CurrentX,' ',CurrentY,' ',Uspace(CurrentX,CurrentY));
                end;
            end;
            CloseFile(DatFile);
        end;
end;

end.

```

```

unit ChartXVT;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, TeEngine, Series, ExtCtrls, TeeProcs, Chart, StdCtrls, Buttons,
  Constants, Gauges, ExtDlgs, Trajectories, Menus, Math;

type
  TFormChartXVT = class (TForm)
    Chart1: TChart;
    Series1: TPointSeries;
    Gauge1: TGauge;
    Series2: TPointSeries;
    Series3: TLineSeries;
    ListBox1: TListBox;
    CheckBoxDec: TCheckBox;
    CheckBoxAngstrem: TCheckBox;
    ChartXtBtn: TBitBtn;
    ChartVtBtn: TBitBtn;
    ChartVXBtn: TBitBtn;
    Series4: TLineSeries;
    CheckBoxSelect: TCheckBox;
    ChartMABtn: TSpeedButton;
    ChartMBBtn: TSpeedButton;
    ChartMCBtn: TSpeedButton;
    LEGraphYmin: TLabeledEdit;
    LEGraphXmin: TLabeledEdit;
    LEGraphXmax: TLabeledEdit;
    LEGraphYmax: TLabeledEdit;
    ChBoxGraphYAuto: TCheckBox;
    ChBoxGraphXAuto: TCheckBox;
    Label1: TLabel;
    Label2: TLabel;
    SBChangeGraph: TSpeedButton;
    Bevel1: TBevel;
    Bevel2: TBevel;
    ChartEnBtn: TBitBtn;
    ChartEeVAngBtn: TBitBtn;
    Label6: TLabel;
    LENomTrjOt: TLabeledEdit;
    LENomTrjDo: TLabeledEdit;
    LabelForAllTrj: TLabel;
    Bevel6: TBevel;
    Label10: TLabel;
    LENomUzlaOt: TLabeledEdit;
    LENomUzlaDo: TLabeledEdit;
    LabelForAllUzels: TLabel;
    Bevel3: TBevel;
    PopupMenuChartXVT: TPopupMenu;
    ShowGrid: TMenuItem;
    ShowText: TMenuItem;
    N3: TMenuItem;
    SaveGraph: TMenuItem;
    PrintGraph: TMenuItem;
    ToClipBrd: TMenuItem;
    CheckBoxExcel: TCheckBox;
    procedure ChartXtBtnClick(Sender: TObject);
    procedure ChartVtBtnClick(Sender: TObject);
    procedure ChartVXBtnClick(Sender: TObject);
    procedure ChartMABtnClick(Sender: TObject);
    procedure ChartMBBtnClick(Sender: TObject);
    procedure ChartMCBtnClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure SBChangeGraphClick(Sender: TObject);
    procedure LEGraphYminChange(Sender: TObject);
    procedure LEGraphXminChange(Sender: TObject);
    procedure ChBoxGraphYAutoClick(Sender: TObject);
    procedure ChartEnBtnClick(Sender: TObject);
    procedure ChartEeVAngBtnClick(Sender: TObject);
  end;

```

```

procedure LENomTrjOtChange(Sender: TObject);
procedure LENomTrjDoChange(Sender: TObject);
procedure LENomUzlaOtChange(Sender: TObject);
procedure LENomUzlaDoChange(Sender: TObject);
procedure LabelForAllTrjClick(Sender: TObject);
procedure LabelForAllUzelsClick(Sender: TObject);
procedure ShowGridClick(Sender: TObject);
procedure ShowTextClick(Sender: TObject);
procedure PrintGraphClick(Sender: TObject);
procedure SaveGraphClick(Sender: TObject);
procedure ToClipBrdClick(Sender: TObject);

private
  { Private declarations }
  NomTrjOt, NomTrjDo, NomUzlaOt, NomUzlaDo: integer;
  GetTrjUzelOk: Boolean;
procedure GetTrjUzel;
procedure DoChartXVT(YAxis:String);
procedure RepaintGraphChartXVT;
public
  { Public declarations }
procedure Zapolnenie;
end;

var
  FormChartXVT: TFormChartXVT;

implementation

{$R *.dfm}

uses ComObj;

procedure TFormChartXVT.DoChartXVT(YAxis:String);
var
  i, j: integer;
  x, y, t: double;
  ExcelApplication: Variant;
  ColorsOfPoints: TColor;

begin
  Chart1.Visible:=True;
  Chart1.FreeAllSeries;

  if CheckBoxExcel.Checked then begin
    ExcelApplication:=CreateOleObject ('Excel.Application');
    ExcelApplication.Visible:=True;
    ExcelApplication.WorkBooks.Add;
  end;

  GetTrjUzel; if GetTrjUzelOk=False then exit;

  if YAxis='x' then Chart1.LeftAxis.Title.Caption:='x (t) / ax'
  else if YAxis='v' then Chart1.LeftAxis.Title.Caption:='Vx (t)'
  else if YAxis='MA' then Chart1.LeftAxis.Title.Caption:='MA (t)'
  else if YAxis='MB' then Chart1.LeftAxis.Title.Caption:='MB (t)'
  else if YAxis='MC' then Chart1.LeftAxis.Title.Caption:='MC (t)'
  else if YAxis='En-eV' then Chart1.LeftAxis.Title.Caption:=
    'Еп (потерянная энергия), эВ'
  else if YAxis='eV-Ang' then Chart1.LeftAxis.Title.Caption:=
    'Еп (потерянная энергия), эВ/Анг.'
  else Chart1.LeftAxis.Title.Caption:='0';

  if CheckBoxAngstrem.Checked=False
  then Chart1.BottomAxis.Title.Caption:='г л у б и н а, т а у'
  else Chart1.BottomAxis.Title.Caption:='г л у б и н а, с м.';

  if CheckBoxExcel.Checked then
    for j:=NomUzlaOt to NomUzlaDo do
      ExcelApplication.Cells[j+1,1].Value:=
        Trajects.Traject[0].t_arr[j]*d_small/epsilon/10000;

```

```

if CheckBoxDec.Checked=False then
begin
    Gauge1.MaxValue:=NomTrjDo-NomTrjOt+1;
for i:=NomTrjOt to NomTrjDo do
begin
        Series3:=Series.TLineSeries.Create(Chart1);
        Chart1.AddSeries(Series3);
        ColorsOfPoints:=clBlack;
for j:=NomUzlaOt to NomUzlaDo do
begin

            if YAxis='x' then y:=Trajects.Traject[i].x_arr[j]
            else if YAxis='v' then y:=Trajects.Traject[i].v_arr[j]
            else if YAxis='MA' then y:=Trajects.Traject[i].MA_arr[j]
            else if YAxis='MB' then y:=Trajects.Traject[i].MB_arr[j]
            else if YAxis='MC' then y:=Trajects.Traject[i].MC_arr[j]
            else if YAxis='En-eV' then y:=Energy_arr[i,j]
            else if YAxis='eV-Ang'
                then y:=Energy_arr[i,j]/(Trajects.Traject[i].tau*(j+1)
                    +Trajects.Traject[i].t_arr[0])*epsilon/d_small

            else y:=0;
            t:=Trajects.Traject[i].t_arr[j];
if CheckBoxAngstrem.Checked=False
then
begin
                Series3.AddXY(t,y,'',ColorsOfPoints);
                if CheckBoxExcel.Checked then ExcelApplication.Cells[j+1,i+2].Value:=y;
            end
else
begin
                Series3.AddXY(t*d_small/epsilon/1E8,y,'',ColorsOfPoints);
                if CheckBoxExcel.Checked then ExcelApplication.Cells[j+1,i+2].Value:=y;
            end;
end;
            Gauge1.Progress:=i+1-NomTrjOt;
end;

if (CheckBoxSelect.Checked=True) and (ListBox1.ItemIndex>=0) then
begin
        Series4:=Series.TLineSeries.Create(Chart1);
        Series4.LinePen.Width:=2;
        Chart1.AddSeries(Series4);
        ColorsOfPoints:=clBlue;
for j:=NomUzlaOt to NomUzlaDo do
begin
            if YAxis='x' then y:=Trajects.Traject[ListBox1.ItemIndex].x_arr[j]
            else if YAxis='v' then y:=Trajects.Traject[ListBox1.ItemIndex].v_arr[j]
            else if YAxis='MA' then y:=Trajects.Traject[ListBox1.ItemIndex].MA_arr[j]
            else if YAxis='MB' then y:=Trajects.Traject[ListBox1.ItemIndex].MB_arr[j]
            else if YAxis='MC' then y:=Trajects.Traject[ListBox1.ItemIndex].MC_arr[j]
            else if YAxis='En-eV' then y:=Energy_arr[ListBox1.ItemIndex,j]
            else if YAxis='eV-Ang'
                then y:=Energy_arr[ListBox1.ItemIndex,j]
                    /(Trajects.Traject[ListBox1.ItemIndex].tau*(j+1)
                    +Trajects.Traject[ListBox1.ItemIndex].t_arr[0])
                    *epsilon/d_small

            else y:=0;
            t:=Trajects.Traject[ListBox1.ItemIndex].t_arr[j];
if CheckBoxAngstrem.Checked=False
then Series4.AddXY(t,y,'',ColorsOfPoints)
            else Series4.AddXY(t*d_small/epsilon/1E8,y,'',ColorsOfPoints);
end;
end;
end;

if CheckBoxDec.Checked=True then
begin
        Gauge1.MaxValue:=2*(NomTrjDo-NomTrjOt+1);

for i:=NomTrjOt to NomTrjDo do

```



```

begin
    Series1:=Series.TPointSeries.Create(Chart1);
    Series1.Pointer.HorizSize:=1;
    Series1.Pointer.VertSize:=1;
    Series1.Pointer.Pen.Visible:=False;
    Series1.Pointer.Style:=psCircle;
    Chart1.AddSeries(Series1);
    ColorsOfPoints:=clBlack;

    for j:=NomUzlaOt to NomUzlaDo do
        begin
            if YAxis='x' then y:=Trajects.Traject[i].x_arr[j]
            else if YAxis='v' then y:=Trajects.Traject[i].v_arr[j]
            else if YAxis='MA' then y:=Trajects.Traject[i].MA_arr[j]
            else if YAxis='MB' then y:=Trajects.Traject[i].MB_arr[j]
            else if YAxis='MC' then y:=Trajects.Traject[i].MC_arr[j]
            else if YAxis='En-eV' then y:=Energy_arr[i,j]
            else if YAxis='eV-Ang'
                then y:=Energy_arr[i,j]/(Trajects.Traject[i].tau*(j+1)
                    +Trajects.Traject[i].t_arr[0])*epsilon/d_small
            else y:=0;
            t:=Trajects.Traject[i].t_arr[j];
            if CheckBoxAngstrem.Checked=False
            then Series1.AddXY(t,y,'',ColorsOfPoints)
            else Series1.AddXY(t*d_small/epsilon/1E8,y,'',ColorsOfPoints);

        end;
        Gauge1.Progress:=i+1-NomTrjOt;
    end;

    if (CheckBoxSelect.Checked=True) and (ListBox1.ItemIndex>=0) then
        begin
            Series1:=Series.TPointSeries.Create(Chart1);
            Series1.Pointer.HorizSize:=2;
            Series1.Pointer.VertSize:=2;
            Series1.Pointer.Pen.Visible:=False;
            Series1.Pointer.Style:=psCircle;
            Chart1.AddSeries(Series1);
            ColorsOfPoints:=clBlue;

            for j:=NomUzlaOt to NomUzlaDo do
                begin
                    if YAxis='x' then y:=Trajects.Traject[ListBox1.ItemIndex].x_arr[j]
                    else if YAxis='v' then y:=Trajects.Traject[ListBox1.ItemIndex].v_arr[j]
                    else if YAxis='MA' then y:=Trajects.Traject[ListBox1.ItemIndex].MA_arr[j]
                    else if YAxis='MB' then y:=Trajects.Traject[ListBox1.ItemIndex].MB_arr[j]
                    else if YAxis='MC' then y:=Trajects.Traject[ListBox1.ItemIndex].MC_arr[j]
                    else if YAxis='En-eV' then y:=Energy_arr[ListBox1.ItemIndex,j]
                    else if YAxis='eV-Ang'
                        then y:=Energy_arr[ListBox1.ItemIndex,j]
                            /(Trajects.Traject[ListBox1.ItemIndex].tau*(j+1)
                                +Trajects.Traject[ListBox1.ItemIndex].t_arr[0])
                                *epsilon/d_small
                    else y:=0;
                    t:=Trajects.Traject[ListBox1.ItemIndex].t_arr[j];
                    if CheckBoxAngstrem.Checked=False
                    then Series1.AddXY(t,y,'',ColorsOfPoints)
                    else Series1.AddXY(t*d_small/epsilon/1E8,y,'',ColorsOfPoints);

                end;
            end;

        {Проверка условия попадания траектории каждой частицы в
        интервал деканализирования:}
        For i:=NomTrjOt to NomTrjDo do
            begin
                Series2:=Series.TPointSeries.Create(Chart1);
                Series2.Pointer.HorizSize:=2;
                Series2.Pointer.VertSize:=2;
                Series2.Pointer.Style:=psCircle;
                Series2.Pointer.Pen.Visible:=False;
                Chart1.AddSeries(Series2);
                ColorsOfPoints:=clRed;
            end;
        end;
    end;

```

```

    Gauge1.Progress:=i+1-NomTrjOt+(NomTrjDo-NomTrjOt+1);
    for j:=NomUzlaOt to NomUzlaDo do
    begin
        x:=Trajects.Traject[i].x_arr[j];
        x:=x-Floor(x); {Определяем дробную часть значения x (от 0 до 1)}
        if ( (Ploskost='(100)') and
            ( (x<=(sigma_dec))
              or (x>(1-sigma_dec))
              or ( (x<=(0.25+sigma_dec)) and (x>(0.25-sigma_dec)) )
              or ( (x<=(0.5 +sigma_dec)) and (x>(0.5 -sigma_dec)) )
              or ( (x<=(0.75+sigma_dec)) and (x>(0.75-sigma_dec)) ) ) )
        or ( (Ploskost='(110)') and
            ( (x<=(sigma_dec))
              or (x>(1-sigma_dec))
              or ( (x<=(0.5+sigma_dec)) and (x>(0.5-sigma_dec)) ) ) )
        or ( (Ploskost='(111)') and
            ( (x<=(-0.0044+sigma_dec))
              or (x>(1-0.0044-sigma_dec))
              or ( (x<=(0.7556+sigma_dec)) and (x>(0.7556 -sigma_dec)) ) ) ) )
        then
        begin
            if YAxis='x' then y:=Trajects.Traject[i].x_arr[j]
            else if YAxis='v' then y:=Trajects.Traject[i].v_arr[j]
            else if YAxis='MA' then y:=Trajects.Traject[i].MA_arr[j]
            else if YAxis='MB' then y:=Trajects.Traject[i].MB_arr[j]
            else if YAxis='MC' then y:=Trajects.Traject[i].MC_arr[j]
            else if YAxis='En-eV' then y:=Energy_arr[i,j]
            else if YAxis='eV-Ang'
            then y:=Energy_arr[i,j]/(Trajects.Traject[i].tau*(j+1)
                                     +Trajects.Traject[i].t_arr[0])*epsilon/d_small
            else y:=0;
            if CheckBoxAngstrem.Checked=False
            then Series2.AddXY(Trajects.Traject[i].t_arr[j],y,'',ColorsOfPoints)
            else Series2.AddXY(Trajects.Traject[i].t_arr[j]
                               *d_small/epsilon/1E8,y,'',ColorsOfPoints);
        end;
    end;
end;

RepaintGraphChartXVT;

end;

procedure TFormChartXVT.ChartXtBtnClick(Sender: TObject);
begin
    DoChartXVT('x');
end;

procedure TFormChartXVT.ChartVtBtnClick(Sender: TObject);
begin
    DoChartXVT('v');
end;

procedure TFormChartXVT.ChartVXBtnClick(Sender: TObject);
var
    i, j: integer;
    x, y, t: double;
    ExcelApplication: Variant;
    ColorsOfPoints: TColor;

begin
    Chart1.Visible:=True;
    Chart1.FreeAllSeries;
    Chart1.LeftAxis.Title.Caption:='Vx (t)';
    Chart1.BottomAxis.Title.Caption:='X (t) / ax';

```

```

if CheckBoxExcel.Checked then
  begin
    ExcelApplication:=CreateOleObject ('Excel.Application');
    ExcelApplication.Visible:=True;
    ExcelApplication.WorkBooks.Add;
  end;

GetTrjUzel; if GetTrjUzelOk=False then exit;

if CheckBoxDec.Checked=False then
  begin
    Gauge1.MaxValue:=NomTrjDo-NomTrjOt+1;
    for i:=NomTrjOt to NomTrjDo do
      begin
        Series1:=Series.TPointSeries.Create(Chart1);
        Series1.Pointer.HorizSize:=1;
        Series1.Pointer.VertSize:=1;
        Series1.Pointer.Pen.Visible:=False;
        Series1.Pointer.Style:=psCircle;
        Chart1.AddSeries(Series1);
        ColorsOfPoints:=clBlack;
        for j:=NomUzlaOt to NomUzlaDo do
          begin
            y:=Trajects.Traject[i].v_arr[j];
            t:=Trajects.Traject[i].x_arr[j];
            Series1.AddXY(t,y,'',ColorsOfPoints);
            if CheckBoxExcel.Checked then begin
              ExcelApplication.Cells[j+1,2*i+1].Value:=t;
              ExcelApplication.Cells[j+1,2*i+2].Value:=y;
            end;
          end;
          Gauge1.Progress:=i+1-NomTrjOt;
        end;
      end;
    if (CheckBoxSelect.Checked=True) and (ListBox1.ItemIndex>=0) then
      begin
        Series1:=Series.TPointSeries.Create(Chart1);
        Series1.Pointer.HorizSize:=2;
        Series1.Pointer.VertSize:=2;
        Series1.Pointer.Pen.Visible:=False;
        Series1.Pointer.Style:=psCircle;
        Chart1.AddSeries(Series1);
        ColorsOfPoints:=clBlue;
        for j:=NomUzlaOt to NomUzlaDo do
          begin
            y:=Trajects.Traject[ListBox1.ItemIndex].v_arr[j];
            t:=Trajects.Traject[ListBox1.ItemIndex].x_arr[j];
            Series1.AddXY(t,y,'',ColorsOfPoints);
          end;
        end;
      end;
  end;

if CheckBoxDec.Checked=True then
  begin
    Gauge1.MaxValue:=2*(NomTrjDo-NomTrjOt+1);

    for i:=NomTrjOt to NomTrjDo do
      begin
        Series1:=Series.TPointSeries.Create(Chart1);
        Series1.Pointer.HorizSize:=1;
        Series1.Pointer.VertSize:=1;
        Series1.Pointer.Pen.Visible:=False;
        Series1.Pointer.Style:=psCircle;
        Chart1.AddSeries(Series1);
        ColorsOfPoints:=clBlack;
        for j:=NomUzlaOt to NomUzlaDo do
          begin
            y:=Trajects.Traject[i].v_arr[j];
            t:=Trajects.Traject[i].x_arr[j];
            Series1.AddXY(t,y,'',ColorsOfPoints);
          end;
        end;
        Gauge1.Progress:=i+1-NomTrjOt;
      end;
    end;
  end;

```

```

end;

if (CheckBoxSelect.Checked=True) and (ListBox1.ItemIndex>=0) then
begin
    Series1:=Series.TPointSeries.Create(Chart1);
    Series1.Pointer.HorizSize:=2;
    Series1.Pointer.VertSize:=2;
    Series1.Pointer.Pen.Visible:=False;
    Series1.Pointer.Style:=psCircle;
    Chart1.AddSeries(Series1);
    ColorsOfPoints:=clBlue;
    for j:=NomUzlaOt to NomUzlaDo do
    begin
        y:=Trajects.Traject[ListBox1.ItemIndex].v_arr[j];
        t:=Trajects.Traject[ListBox1.ItemIndex].x_arr[j];
        Series1.AddXY(t,y,'',ColorsOfPoints);
    end;
end;

{Проверка условия попадания траектории каждой частицы в
интервал деканализования:}
For i:=NomTrjOt to NomTrjDo do
begin
    Series2:=Series.TPointSeries.Create(Chart1);
    Series2.Pointer.HorizSize:=2;
    Series2.Pointer.VertSize:=2;
    Series2.Pointer.Style:=psCircle;
    Series2.Pointer.Pen.Visible:=False;
    Chart1.AddSeries(Series2);
    ColorsOfPoints:=clRed;
    Gauge1.Progress:=i+1-NomTrjOt+(NomTrjDo-NomTrjOt+1);
    for j:=NomUzlaOt to NomUzlaDo do
    begin
        x:=Trajects.Traject[i].x_arr[j];
        x:=x-Floor(x); {Определяем дробную часть значения x (от 0 до 1)}
        if ( ( Ploskost='(100)') and
            ( (x<=(sigma_dec)
              or (x>(1-sigma_dec))
              or ( (x<=(0.25+sigma_dec)) and (x>(0.25-sigma_dec)) )
              or ( (x<=(0.5 +sigma_dec)) and (x>(0.5 -sigma_dec)) )
              or ( (x<=(0.75+sigma_dec)) and (x>(0.75-sigma_dec)) ) ) )
            or ( (Ploskost='(110)') and
            ( (x<=(sigma_dec)
              or (x>(1-sigma_dec))
              or ( (x<=(0.5+sigma_dec)) and (x>(0.5-sigma_dec)) ) ) )
            or ( (Ploskost='(111)') and
            ( (x<=(-0.0044+sigma_dec)
              or (x>(1-0.0044-sigma_dec))
              or ( (x<=(0.7556+sigma_dec)) and (x>(0.7556 -sigma_dec)) ) ) ) )
            then Series2.AddXY(Trajects.Traject[i].x_arr[j],
                               Trajects.Traject[i].v_arr[j],'',ColorsOfPoints);
    end;
end;
end;

if CheckBoxExcel.Checked then ExcelApplication.Visible:=True;

RepaintGraphChartXVT;
end;

procedure TFormChartXVT.ChartMABtnClick(Sender: TObject);
begin
    DoChartXVT('MA');
end;

procedure TFormChartXVT.ChartMBBbtnClick(Sender: TObject);
begin
    DoChartXVT('MB');
end;

```

```

procedure TFormChartXVT.ChartMCBtnClick(Sender: TObject);
begin
    DoChartXVT('MC');
end;

procedure TFormChartXVT.ChartEnBtnClick(Sender: TObject);
begin
    DoChartXVT('En-eV');
end;

procedure TFormChartXVT.ChartEeVAngBtnClick(Sender: TObject);
begin
    DoChartXVT('eV-Ang');
end;

procedure TFormChartXVT.Zapolnenie; //процедура заполняет данными форму
begin
    if Settings[69]='CheckBoxDec=False'
    then CheckBoxDec.Checked:=False
    else CheckBoxDec.Checked:=True;
    if Settings[70]='CheckBoxAngstrem=False'
    then CheckBoxAngstrem.Checked:=False
    else CheckBoxAngstrem.Checked:=True;
end;

procedure TFormChartXVT.FormCreate(Sender: TObject);
begin
    Zapolnenie;
end;

procedure TFormChartXVT.RepaintGraphChartXVT;
begin
    Chart1.Repaint;
    LEGraphYmax.Text:=FloatToStrF(Chart1.LeftAxis.Maximum,ffGeneral,8,8);
    LEGraphYmin.Text:=FloatToStrF(Chart1.LeftAxis.Minimum,ffGeneral,8,8);
    LEGraphXmax.Text:=FloatToStrF(Chart1.BottomAxis.Maximum,ffGeneral,8,8);
    LEGraphXmin.Text:=FloatToStrF(Chart1.BottomAxis.Minimum,ffGeneral,8,8);
end;

procedure TFormChartXVT.SBChangeGraphClick(Sender: TObject);
begin

    if ChBoxGraphYAuto.Checked=True then
    begin
        Chart1.LeftAxis.Automatic:=True;
    end
    else
    begin
        Chart1.LeftAxis.Automatic:=False;
        try
            Chart1.LeftAxis.Minimum:=StrToFloat(LEGraphYmin.Text);
            Chart1.LeftAxis.Maximum:=StrToFloat(LEGraphYmax.Text);
        except
            exit;
        end;
    end;

    if ChBoxGraphXAuto.Checked=True then
        Chart1.BottomAxis.Automatic:=True
    else
    begin
        Chart1.BottomAxis.Automatic:=False;
        try
            Chart1.BottomAxis.Minimum:=StrToFloat(LEGraphXmin.Text);
            Chart1.BottomAxis.Maximum:=StrToFloat(LEGraphXmax.Text);
        except
            exit;
        end;
    end;
    SBChangeGraph.Enabled:=False;

```

```

RepaintGraphChartXVT;
end;

procedure TFormChartXVT.LEGraphYminChange(Sender: TObject);
begin
  if ChBoxGraphYAuto.Checked=False
  then SBChangeGraph.Enabled:=True;
end;

procedure TFormChartXVT.LEGraphXminChange(Sender: TObject);
begin
  if ChBoxGraphXAuto.Checked=False
  then SBChangeGraph.Enabled:=True;
end;

procedure TFormChartXVT.ChBoxGraphYAutoClick(Sender: TObject);
begin
  SBChangeGraph.Enabled:=True;
end;

procedure TFormChartXVT.GetTrjUzel;
begin
  GetTrjUzelOk:=False;

  if (LENomTrjOt.Font.Color=clRed)
  or (LENomTrjDo.Font.Color=clRed)
  or (LENomUzlaOt.Font.Color=clRed)
  or (LENomUzlaDo.Font.Color=clRed)
  then exit;

  NomTrjOt:=StrToInt(LENomTrjOt.Text);
  if NomTrjOt<0 then NomTrjOt:=0;
  if NomTrjOt>(Trajects.CntOfTraject-1) then NomTrjOt:=Trajects.CntOfTraject-1;
  LENomTrjOt.Text:=IntToStr(NomTrjOt);

  NomTrjDo:=StrToInt(LENomTrjDo.Text);
  if NomTrjDo<NomTrjDo then NomTrjDo:=NomTrjOt;
  if NomTrjDo>(Trajects.CntOfTraject-1) then NomTrjDo:=Trajects.CntOfTraject-1;
  LENomTrjDo.Text:=IntToStr(NomTrjDo);

  NomUzlaOt:=StrToInt(LENomUzlaOt.Text);
  if NomUzlaOt<0 then NomUzlaOt:=0;
  if NomUzlaOt>(Trajects.Traject[0].NodeCount-1) then
NomUzlaOt:=Trajects.Traject[0].NodeCount-1;
  LENomUzlaOt.Text:=IntToStr(NomUzlaOt);
  NomUzlaDo:=StrToInt(LENomUzlaDo.Text);
  if NomUzlaDo<NomUzlaOt then NomUzlaDo:=NomUzlaDo;
  if NomUzlaDo>(Trajects.Traject[0].NodeCount-1) then
NomUzlaDo:=Trajects.Traject[0].NodeCount-1;
  LENomUzlaDo.Text:=IntToStr(NomUzlaDo);
  GetTrjUzelOk:=True;
end;

procedure TFormChartXVT.LENomTrjOtChange(Sender: TObject);
begin
  RedErrors(FormChartXVT.LENomTrjOt, 'Integer');
end;

procedure TFormChartXVT.LENomTrjDoChange(Sender: TObject);
begin
  RedErrors(FormChartXVT.LENomTrjDo, 'Integer');
end;

procedure TFormChartXVT.LENomUzlaOtChange(Sender: TObject);
begin
  RedErrors(FormChartXVT.LENomUzlaOt, 'Integer');
end;

procedure TFormChartXVT.LENomUzlaDoChange(Sender: TObject);
begin
  RedErrors(FormChartXVT.LENomUzlaDo, 'Integer');

```

```

end;

procedure TFormChartXVT.LabelForAllTrjClick(Sender: TObject);
begin
    LENomTrjOt.Text:='0';
    LENomTrjDo.Text:=IntToStr(Trajects.CntOfTraject-1);
end;

procedure TFormChartXVT.LabelForAllUzelsClick(Sender: TObject);
begin
    LENomUzlaOt.Text:='0';
    LENomUzlaDo.Text:=IntToStr(Trajects.Traject[0].NodeCount-1);
end;

procedure TFormChartXVT.ShowGridClick(Sender: TObject);
begin
    if ShowGrid.Checked=False then
        begin
            Chart1.LeftAxis.Grid.Visible:=True;
            Chart1.BottomAxis.Grid.Visible:=True;
            ShowGrid.Checked:=True;
        end
    else
        begin
            Chart1.LeftAxis.Grid.Visible:=False;
            Chart1.BottomAxis.Grid.Visible:=False;
            ShowGrid.Checked:=False;
        end;
    end;

procedure TFormChartXVT.ShowTextClick(Sender: TObject);
begin
    if ShowText.Checked=False then
        begin
            Chart1.Foot.Text.Text:=
                FloatToStr(Trajects.CntOfTraject)+' tp. X='
                +FloatToStrF(Trajects.Traject[0].x_arr[0],ffGeneral,3,3)+'...'
                +FloatToStrF(Trajects.Traject[Trajects.CntOfTraject-1].x_arr[0],ffGeneral,3,3)
                +' V='+FloatToStrF(Trajects.Traject[0].v_arr[0],ffGeneral,3,3)+'...'
                +FloatToStrF(Trajects.Traject[Trajects.CntOfTraject-1].v_arr[0],ffGeneral,3,3)
                +' Zl='+FloatToStr(Zl)+' M='+FloatToStr(MassOfIon/m0c2)+'aem T='
                +FloatToStr(EnergyT/1000000)+'MэB'#13
                +' ypoж='+FloatToStr(DeltaTheta*180/Pi)+'град'+ ' k='+FloatToStr(k)
                +' l='+FloatToStr(l)+' '+Ploskost+' Curv='+FloatToStr(kt)+'*t+'
                +FloatToStr(Curv*Sqr(epsilon)/d_small)+'*d/eps^2';
            Chart1.Foot.Visible:=True; ShowText.Checked:=True;
        end
    else
        begin
            Chart1.Foot.Visible:=False; ShowText.Checked:=False;
        end;
    end;

procedure TFormChartXVT.PrintGraphClick(Sender: TObject);
begin
    if Form1.PrintDialog1.Execute then Chart1.Print;
end;

procedure TFormChartXVT.SaveGraphClick(Sender: TObject);
begin
    if Form1.SavePictureDialog1.Execute then
        Chart1.SaveToMetafileEnh(Form1.SavePictureDialog1.FileName);
end;

procedure TFormChartXVT.ToClipBrdClick(Sender: TObject);
begin
    Chart1.CopyToClipboardMetafile(true);
end;

end.

```

unit RaspredV;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, Buttons, Constants, Gauges, Trajectories, ComCtrls,
ExtCtrls, ExtDlgs, TeEngine, Series, TeeProcs, Chart, Menus, ClipBrd;

type

```
TFormRaspredV = class(TForm)
  TabSheet1: TTabSheet;
  TabSheet2: TTabSheet;
  TabSheet3: TTabSheet;
  PageRaspVXGl: TPageControl;
  VGistLegend: TLabel;
  LabelDelthaV: TLabel;
  GistVBtn: TBitBtn;
  XGistLegend: TLabel;
  LabelDelthaX: TLabel;
  GistXBtn: TBitBtn;
  RaspGlLegend: TLabel;
  ChBoxAngstrem: TCheckBox;
  RaspVglBtn: TBitBtn;
  GroupBoxCalcGl: TGroupBox;
  Label1: TLabel;
  LECalcGlOt: TLabeledEdit;
  LECalcGlDo: TLabeledEdit;
  CalcGlBtn: TBitBtn;
  CalcGlLegend: TLabel;
  GroupBoxCalcV: TGroupBox;
  LECalcVot: TLabeledEdit;
  LECalcVdo: TLabeledEdit;
  CalcVBtn: TBitBtn;
  CalcVLegend: TLabel;
  GroupBoxCalcX: TGroupBox;
  LECalcXot: TLabeledEdit;
  LECalcXdo: TLabeledEdit;
  CalcXBtn: TBitBtn;
  CalcXLegend: TLabel;
  Chart1: TChart;
  Series1: TLineSeries;
  Series2: TLineSeries;
  CloseFormBtn: TSpeedButton;
  TabSheet4: TTabSheet;
  FxvLegend: TLabel;
  Label3: TLabel;
  FxBtn: TBitBtn;
  LENomUzla: TLabeledEdit;
  LEXVot: TLabeledEdit;
  LEXVdo: TLabeledEdit;
  LEXVStep: TLabeledEdit;
  LEMamin: TLabeledEdit;
  LEMCmin: TLabeledEdit;
  LabelMamax: TLabel;
  LabelMCmax: TLabel;
  LabelFxvGlubina: TLabel;
  LabelGlubinaTau: TLabel;
  LabelGlubinaAngstrem: TLabel;
  FvBtn: TBitBtn;
  UpDownFxv: TUpDown;
  LabelVGlubina: TLabel;
  LEVNomUzla: TLabeledEdit;
  UpDownGistV: TUpDown;
  LabelVGlubinaTau: TLabel;
  LabelVGlubinaAngstrem: TLabel;
  LabelXGlubina: TLabel;
  LEXNomUzla: TLabeledEdit;
  UpDownGistX: TUpDown;
  LabelXGlubinaTau: TLabel;
  LabelXGlubinaAngstrem: TLabel;
```



```

LEStepUpDownFvx: TLabelEdit;
LEStepUpDownGistX: TLabelEdit;
LEStepUpDownGistV: TLabelEdit;
LEhV: TLabelEdit;
LEhX: TLabelEdit;
LabelVCalcRes: TLabel;
LabelXCalcRes: TLabel;
LabelG1CalcRes: TLabel;
LEVminG1: TLabelEdit;
LEVmaxG1: TLabelEdit;
Label7: TLabel;
ChBoxGraphYAuto: TCheckBox;
Label8: TLabel;
ChBoxGraphXAuto: TCheckBox;
SBChangeGraph: TSpeedButton;
Bevel3: TBevel;
LEGraphXmin: TLabelEdit;
LEGraphXmax: TLabelEdit;
LEGraphYmax: TLabelEdit;
LEGraphYmin: TLabelEdit;
Bevel2: TBevel;
LENomTrjDo: TLabelEdit;
LENomUzlaDo: TLabelEdit;
LENomUzlaOt: TLabelEdit;
LabelForAllUzels: TLabel;
Label10: TLabel;
Bevel6: TBevel;
LabelForAllTrj: TLabel;
LENomTrjOt: TLabelEdit;
Label6: TLabel;
Bevel5: TBevel;
LENormNo: TLabelEdit;
LENormNoGistV: TLabelEdit;
ChBoxNormGistV: TCheckBox;
LENormNoGistX: TLabelEdit;
ChBoxNormGistX: TCheckBox;
ChBoxNormG1V: TCheckBox;
LENormNoG1V: TLabelEdit;
Bevel1: TBevel;
MminLegend: TLabel;
Bevel4: TBevel;
Bevel7: TBevel;
PopupMenuChartXV: TPopupMenu;
ShowGrid: TMenuItem;
ShowText: TMenuItem;
N3: TMenuItem;
SaveGraph: TMenuItem;
PrintGraph: TMenuItem;
ToClipBrd: TMenuItem;
SBExcel: TSpeedButton;
SpeedButton1: TSpeedButton;
LENVResT: TLabelEdit;
SpeedButton2: TSpeedButton;
SpeedButton3: TSpeedButton;
procedure RaspVglBtnClick(Sender: TObject);
procedure GistVBtnClick(Sender: TObject);
procedure GistXBtnClick(Sender: TObject);
procedure CalcG1BtnClick(Sender: TObject);
procedure CalcVBtnClick(Sender: TObject);
procedure CalcXBtnClick(Sender: TObject);
procedure CloseFormBtnClick(Sender: TObject);
procedure FxBtnClick(Sender: TObject);
procedure FvBtnClick(Sender: TObject);
procedure LENomUzlaChange(Sender: TObject);
procedure UpDownFvxClick(Sender: TObject; Button: TUDBtnType);
procedure UpDownGistVClick(Sender: TObject; Button: TUDBtnType);
procedure LEVNomUzlaChange(Sender: TObject);
procedure UpDownGistXClick(Sender: TObject; Button: TUDBtnType);
procedure LEXNomUzlaChange(Sender: TObject);
procedure MminLegendClick(Sender: TObject);
procedure FormCreate(Sender: TObject);

```

```

procedure LEStepUpDownFvxChange(Sender: TObject);
procedure LEStepUpDownGistVChange(Sender: TObject);
procedure LEStepUpDownGistXChange(Sender: TObject);
procedure LEXVStepChange(Sender: TObject);
procedure LEMaminChange(Sender: TObject);
procedure LEMCminChange(Sender: TObject);
procedure LEXVotChange(Sender: TObject);
procedure LEXVdoChange(Sender: TObject);
procedure LECalcVotChange(Sender: TObject);
procedure LECalcVdoChange(Sender: TObject);
procedure LEhVChange(Sender: TObject);
procedure LECalcXotChange(Sender: TObject);
procedure LECalcXdoChange(Sender: TObject);
procedure LEhXChange(Sender: TObject);
procedure LEVminGlChange(Sender: TObject);
procedure LEVmaxGlChange(Sender: TObject);
procedure LECalcGlotChange(Sender: TObject);
procedure LECalcGlidoChange(Sender: TObject);
procedure SBChangeGraphClick(Sender: TObject);
procedure LEGraphYminChange(Sender: TObject);
procedure LEGraphXminChange(Sender: TObject);
procedure ChBoxGraphYAutoClick(Sender: TObject);
procedure LabelForAllTrjClick(Sender: TObject);
procedure LabelForAllUzelsClick(Sender: TObject);
procedure LENomTrjOtChange(Sender: TObject);
procedure LENomTrjDoChange(Sender: TObject);
procedure LENomUzlaOtChange(Sender: TObject);
procedure LENomUzlaDoChange(Sender: TObject);
procedure LENormNoChange(Sender: TObject);
procedure ShowGridClick(Sender: TObject);
procedure ShowTextClick(Sender: TObject);
procedure SaveGraphClick(Sender: TObject);
procedure PrintGraphClick(Sender: TObject);
procedure ToClipBrdClick(Sender: TObject);
procedure SBExcelClick(Sender: TObject);
procedure SpeedButton1Click(Sender: TObject);
procedure SpeedButton2Click(Sender: TObject);
private
  { Private declarations }
  NomTrjOt, NomTrjDo, NomUzlaOt, NomUzlaDo: integer;
  GetTrjUzelOk: Boolean;
procedure GetTrjUzel;
procedure GistogrammV;
procedure GistogrammX;
procedure FxGraph;
procedure FvGraph;
procedure RepaintGraphRaspredV;
public
  { Public declarations }
procedure Zapolnenie;
end;

var
  FormRaspredV: TFormRaspredV;

implementation

{$R *.dfm}

uses ComObj;

//Распределение по скоростям

procedure TFormRaspredV.GistogrammV;
{Процедура рассчитывает гистограмму по V на глубине,
 соответствующей узлу решения № NomUzla и строит график}
var
  V_Arr_Min, v_arr_max, hv, vj: double;
  i, j, NomUzla, NormNo: integer;
  nv: array of integer;
  S: String;

```

```

begin
    Series1.Clear;
    Series2.Clear;
    Chart1.BottomAxis.Title.Caption:='V(t)';

    if (LEVNomUzla.Font.Color=clRed)
    or (LEhV.Font.Color=clRed)
    or (LENormNoGistV.Font.Color=clRed)
    then exit;

    if ChBoxNormGistV.Checked=True then
    begin
        NormNo:=StrToInt(LENormNoGistV.Text);
        if NormNo<=0 then
        begin
            LENormNoGistV.Font.Color:=clRed;
            exit;
        end;
        Chart1.LeftAxis.Title.Caption:='N / No    ( No = '+IntToStr(NormNo)+' )';
    end
    else
    begin
        NormNo:=1;
        Chart1.LeftAxis.Title.Caption:='N, шт';
    end;

    hv:=StrToFloat(LEhV.Text);

    GetTrjUzel;
    if GetTrjUzelOk=False then exit;

    V_Arr_Min:=Trajects.Traject[0].v_arr[0];
    V_Arr_Max:=Trajects.Traject[0].v_arr[0];
    for i:=NomTrjOt to NomTrjDo do begin
        for j:=0 to Trajects.Traject[i].NodeCount-1 do begin
            if Trajects.Traject[i].v_arr[j]<V_Arr_Min
            then V_Arr_Min:=Trajects.Traject[i].v_arr[j];
            if Trajects.Traject[i].v_arr[j]>V_Arr_Max
            then V_Arr_Max:=Trajects.Traject[i].v_arr[j];
        end;
    end;
    V_Arr_Min:=V_Arr_Min-2;
    V_Arr_Max:=V_Arr_Max+2;

    SetLength(nv,round(abs(V_Arr_Max-V_Arr_Min)/hv)+4);

    NomUzla:=StrToInt(LEVNomUzla.Text);

    vj:=V_Arr_Min;
    For j:=1 to round((V_Arr_Max-V_Arr_Min)/hv)+1 do
    begin
        nv[j]:=0;
        for i:=NomTrjOt to NomTrjDo do
            if (Trajects.Traject[i].v_arr[NomUzla]<=vj+hv/2)
            and (Trajects.Traject[i].v_arr[NomUzla]>vj-hv/2)
            then nv[j]:=nv[j]+1;
        vj:=vj+hv;
    end;

    Str((NomUzla*Trajects.Traject[0].tau+dep_Start)*d_small/epsilon:2:2,S);
    Chart1.Title.Text:='Гистограмма распределения по V'#13
    +'на глубине '+S+' Анг';

    For j:=1 to round((V_Arr_Max-V_Arr_Min)/hv)+1 do
    begin
        vj:=V_Arr_Min+hv*(j-1.5);
        Series2.AddXY(vj,nv[j]/NormNo,'',clBlack);
    end;

    Finalize(nv);

```

```

RepaintGraphRaspredV;

end;

procedure TFormRaspredV.GistVBtnClick(Sender: TObject);
begin
    GistogrammV;
end;

procedure TFormRaspredV.UpDownGistVClick(Sender: TObject; Button: TUDBtnType);
begin
    GistogrammV;
end;

procedure TFormRaspredV.LEStepUpDownGistVChange(Sender: TObject);
var
    UpDownStep, code: integer;
    Text: String;
begin
    Text:=LEStepUpDownGistV.Text;
    Val(Text, UpDownStep, code);
    if code=0
        then UpDownGistV.Increment:=UpDownStep;
end;

procedure TFormRaspredV.LEVNomUzlaChange(Sender: TObject);
var
    NomUzla, Code: integer;
    S: String;
begin
    LEVNomUzla.Font.Color:=clBlack;
    S:=LEVNomUzla.Text;
    Val(S, NomUzla, Code);
    if (code<>0) or (NomUzla<0) or (NomUzla>=Trajects.Traject[0].NodeCount) then
        begin
            LEVNomUzla.Font.Color:=clRed;
            LabelVGlubinaTau.Caption:='??? tay';
            LabelVGlubinaAngstrem.Caption:='??? АНГ';
            exit;
        end;
    LabelVGlubinaTau.Caption:=FloatToStr(NomUzla*Trajects.Traject[0].tau+dep_Start)+'
tay';
    LabelVGlubinaAngstrem.Caption:=FloatToStr((NomUzla*Trajects.Traject[0].tau+dep_Start)
*d_small/epsilon)+' АНГ';
end;

procedure TFormRaspredV.CalcVBtnClick(Sender: TObject);
var
    vmin, vmax, v: double;
    i, nv, NomUzla: integer;
begin
    if (LEVNomUzla.Font.Color=clRed)
        or (LECalcVot.Font.Color=clRed)
        or (LECalcVdo.Font.Color=clRed)
        then exit;
    vmin:=StrToFloat(LECalcVot.Text);
    vmax:=StrToFloat(LECalcVdo.Text);
    NomUzla:=StrToInt(LEVNomUzla.Text);

    GetTrjUzel;
    if GetTrjUzelOk=False then exit;

    nv:=0;
    For i:=NomTrjOt to NomTrjDo do
        begin
            v:=Trajects.Traject[i].v_arr[NomUzla];
            if (v>=vmin) and (v<vmax) then nv:=nv+1;
        end;
    LabelVCalcRes.Caption:=IntToStr(nv);

```

```

end;

procedure TFormRaspredV.LECalcVotChange(Sender: TObject);
begin
  RedErrors(FormRaspredV.LECalcVot, '-');
end;

procedure TFormRaspredV.LECalcVdoChange(Sender: TObject);
begin
  RedErrors(FormRaspredV.LECalcVdo, '-');
end;

procedure TFormRaspredV.LEhVChange(Sender: TObject);
begin
  RedErrors(FormRaspredV.LEhV, 'No <=0');
end;

//Распределение по координатам

procedure TFormRaspredV.GistogrammX;
{Процедура рассчитывает гистограмму по X на глубине,
соответствующей узлу решения № NomUzla и строит график}
var
  X_Arr_Min, X_Arr_Max, hx, xj: double;
  i, j, NomUzla, NormNo: integer;
  nx: array of integer;
  S: String;

begin
  Series1.Clear;
  Series2.Clear;
  Chart1.BottomAxis.Title.Caption:='x/ax';

  if (LEXNomUzla.Font.Color=clRed)
  or (LEhX.Font.Color=clRed)
  or (LENormNoGistX.Font.Color=clRed)
  then exit;

  if ChBoxNormGistX.Checked=True then
  begin
    NormNo:=StrToInt(LENormNoGistX.Text);
    if NormNo<=0 then
    begin
      LENormNoGistX.Font.Color:=clRed;
      exit;
    end;
    Chart1.LeftAxis.Title.Caption:='N / No ( No = '+IntToStr(NormNo)+' )';
  end
  else
  begin
    NormNo:=1;
    Chart1.LeftAxis.Title.Caption:='N, шт';
  end;

  hx:=StrToFloat(LEhX.Text);

  GetTrjUzel;
  if GetTrjUzelOk=False then exit;

  X_Arr_Min:=Trajects.Traject[0].x_arr[0];
  X_Arr_Max:=Trajects.Traject[0].x_arr[0];
  for i:=NomTrjOt to NomTrjDo do begin
    for j:=0 to Trajects.Traject[i].NodeCount-1 do begin
      if Trajects.Traject[i].x_arr[j]<X_Arr_Min
      then X_Arr_Min:=Trajects.Traject[i].x_arr[j];
      if Trajects.Traject[i].v_arr[j]>X_Arr_Max
      then X_Arr_Max:=Trajects.Traject[i].x_arr[j];
    end;
  end;
  X_Arr_Min:=X_Arr_Min-1;
  X_Arr_Max:=X_Arr_Max+1;

```

```

SetLength(nx, round((X_Arr_Max-X_Arr_Min)/hx)+4);

NomUzla:=StrToInt(LEXNomUzla.Text);

xj:=X_Arr_Min;
For j:=1 to round((X_Arr_Max-X_Arr_Min)/hx)+1 do
begin
    nx[j]:=0;
    for i:=NomTrjOt to NomTrjDo do
        if (Trajects.Traject[i].x_arr[NomUzla]<=xj+hx/2)
            and (Trajects.Traject[i].x_arr[NomUzla]>xj-hx/2)
            then nx[j]:=nx[j]+1;
    xj:=xj+hx;
end;

Str((NomUzla*Trajects.Traject[0].tau+dep_Start)*d_small/epsilon:2:2,S);
Chart1.Title.Text.Text:='Гистограмма распределения по X'#13
    +'на глубине '+S+' Анг';

For j:=1 to round((X_Arr_Max-X_Arr_Min)/hx)+1 do
begin
    xj:=X_Arr_Min+hx*(j-1.5);
    Series2.AddXY(xj, nx[j]/NormNo, '', clBlack);
end;

Finalize(nx);

RepaintGraphRaspredV;

end;

procedure TFormRaspredV.GistXBtnClick(Sender: TObject);
begin
    GistogrammX;
end;

procedure TFormRaspredV.UpDownGistXClick(Sender: TObject; Button: TUDBtnType);
begin
    GistogrammX;
end;

procedure TFormRaspredV.LEStepUpDownGistXChange(Sender: TObject);
var
    UpDownStep, code: integer;
    Text: String;
begin
    Text:=LEStepUpDownGistX.Text;
    Val(Text, UpDownStep, code);
    if code=0 then UpDownGistX.Increment:=UpDownStep;
end;

procedure TFormRaspredV.LEXNomUzlaChange(Sender: TObject);
var
    NomUzla, Code: integer;
    S: String;
begin
    LEXNomUzla.Font.Color:=clBlack;
    S:=LEXNomUzla.Text;
    Val(S, NomUzla, Code);
    if (code<>0) or (NomUzla<0) or (NomUzla>=Trajects.Traject[0].NodeCount) then
        begin
            LEXNomUzla.Font.Color:=clRed;
            LabelXGlubinaTau.Caption:='??? tay';
            LabelXGlubinaAngstrem.Caption:='??? Анг';
            exit;
        end;
    LabelXGlubinaTau.Caption:=FloatToStr(NomUzla*Trajects.Traject[0].tau+dep_Start)+'
tay';
    LabelXGlubinaAngstrem.Caption:=FloatToStr((NomUzla*Trajects.Traject[0].tau+dep_Start)

```

```

end;

*d_small/epsilon)+' Анг';

procedure TFormRaspredV.CalcXBtnClick(Sender: TObject);
var
  xmin, xmax, x: double;
  i, nx, NomUzla: integer;
begin
  if (LEXNomUzla.Font.Color=clRed)
  or (LECalcXot.Font.Color=clRed)
  or (LECalcXdo.Font.Color=clRed)
  then exit;
  xmin:=StrToFloat(LECalcXot.Text);
  xmax:=StrToFloat(LECalcXdo.Text);
  NomUzla:=StrToInt(LEXNomUzla.Text);

  GetTrjUzel;
  if GetTrjUzelOk=False then exit;

  nx:=0;
  For i:=NomTrjOt to NomTrjDo do
    begin
      x:=Trajects.Traject[i].x_arr[NomUzla];
      if (x>=xmin) and (x<=xmax) then nx:=nx+1;
    end;
  LabelXCalcRes.Caption:=IntToStr(nx);
end;

procedure TFormRaspredV.LECalcXotChange(Sender: TObject);
begin
  RedErrors(FormRaspredV.LECalcXot, '-');
end;

procedure TFormRaspredV.LECalcXdoChange(Sender: TObject);
begin
  RedErrors(FormRaspredV.LECalcXdo, '-');
end;

procedure TFormRaspredV.LEhXChange(Sender: TObject);
begin
  RedErrors(FormRaspredV.LEhX, 'No <=0');
end;

//Распределение по глубине

procedure TFormRaspredV.RaspVglBtnClick(Sender: TObject);
  {Расчёт зависимости распределения поперечных скоростей
  от глубины проникновения}
var
  V_Arr_Min, V_Arr_Max, y: double;
  i, ti, NormNo: integer;
  ntv: array of integer;
begin
  Series1.Clear;
  Series2.Clear;
  Chart1.Title.Text.Text:='Угловое распределение частиц по глубине';
  if ChBoxAngstrem.Checked=False
  then Chart1.BottomAxis.Title.Caption:='т а у'
  else Chart1.BottomAxis.Title.Caption:='г л у б и н а,   А н г.';

  if (LEVminGl.Font.Color=clRed)
  or (LEVmaxGl.Font.Color=clRed)
  or (LENormNoGlV.Font.Color=clRed)
  then exit;

  if ChBoxNormGlV.Checked=True then
    begin
      NormNo:=StrToInt(LENormNoGlV.Text);
      if NormNo<=0 then
        begin
          LENormNoGlV.Font.Color:=clRed;

```

```

        exit;
    end;
    Chart1.LeftAxis.Title.Caption:='N / No    ( No = '+IntToStr(NormNo)+' )';
end
else
begin
    NormNo:=1;
    Chart1.LeftAxis.Title.Caption:='N, шт';
end;

V_Arr_Min:=StrToFloat(LEVminGl.Text);
V_Arr_Max:=StrToFloat(LEVmaxGl.Text);

GetTrjUzel;
if GetTrjUzelOk=False then exit;

SetLength(ntv,Trajects.Traject[0].NodeCount);
    {Массив ntv - кол-во частиц с данной скоростью на данной глубине}

{Расчёт распределения: Проверяются все частицы по всей глубине,
если частица № i имеет нужную поперечную скорость (от V_Arr_Min до V_Arr_Max)
на данной глубине (ti), то к значению массива ntv[ti] прибавляется 1}
    For i:=NomTrjOt to NomTrjDo do begin
        for ti:=NomUzlaOt to NomUzlaDo do
            begin
                y:=Trajects.Traject[i].v_arr[ti];
                if (y<=V_Arr_Max) and (y>V_Arr_Min)
                    then ntv[ti]:=ntv[ti]+1;
            end;
        end;
    end;

{Построение графика зависимости распределения скоростей от глубины}
    For ti:=NomUzlaOt to NomUzlaDo do
        begin
            if ChBoxAngstrem.Checked=False
                then Series1.AddXY(Trajects.Traject[0].t_arr[ti],ntv[ti]/NormNo,'',clBlack)
                else Series1.AddXY(Trajects.Traject[0].t_arr[ti]*d_small/epsilon,
                                     ntv[ti]/NormNo,'',clBlack);
        end;
    end;

Finalize(ntv);

RepaintGraphRaspredV;

end;

procedure TFormRaspredV.CalcGlBtnClick(Sender: TObject);
label L1;
var
    nv, i, ti, tiot, tido: integer;
    vvmax, vvmin, v: double;
begin
    if (LEVminGl.Font.Color=clRed)
        or (LEVmaxGl.Font.Color=clRed)
        or (LECalcGlot.Font.Color=clRed)
        or (LECalcGldo.Font.Color=clRed)
        then exit;

    vvmin:=StrToFloat(LEVminGl.Text);
    vvmax:=StrToFloat(LEVmaxGl.Text);

    GetTrjUzel;
    if GetTrjUzelOk=False then exit;

    if StrToFloat(LECalcGlot.Text)<=dep_Start
        then LECalcGlot.Text:=FloatToStrF(dep_Start,ffGeneral,10,10);
    if StrToFloat(LECalcGldo.Text)>=(dep_Start+glubina)
        then LECalcGldo.Text:=FloatToStrF((dep_Start+glubina),ffGeneral,10,10);
    tiot:=Round((StrToFloat(LECalcGlot.Text)-dep_Start)/Trajects.Traject[0].tau);
    tido:=Round((StrToFloat(LECalcGldo.Text)-dep_Start)/Trajects.Traject[0].tau);
    Label1.Caption:='Узлы от '+IntToStr(tiot)+' до '+IntToStr(tido);

```



```

    nv:=0;
For i:=NomTrjOt to NomTrjDo do begin
    for ti:=tiot to tido do
    begin
        v:=Trajects.Traject[i].v_arr[ti];
        if (v<=vvmax) and (v>vvmin)
        then begin nv:=nv+1; goto L1; end;
    end;
L1:
    end;
    LabelG1CalcRes.Caption:=IntToStr(nv);
    CalcG1Legend.Caption:='скоростью от '+
    FloatToStr(vvmin)+' до '+FloatToStr(vvmax)+' на заданной глубине (тау)';
end;

procedure TFormRaspredV.LEVminG1Change(Sender: TObject);
begin
    RedErrors(FormRaspredV.LEVminG1, '-');
end;

procedure TFormRaspredV.LEVmaxG1Change(Sender: TObject);
begin
    RedErrors(FormRaspredV.LEVmaxG1, '-');
end;

procedure TFormRaspredV.LECalcGlotChange(Sender: TObject);
begin
    RedErrors(FormRaspredV.LECalcGlot, '-');
end;

procedure TFormRaspredV.LECalcGldoChange(Sender: TObject);
begin
    RedErrors(FormRaspredV.LECalcGldo, '-');
end;

// f(x), f(v)

procedure TFormRaspredV.FxGraph; // Строит график распределения f(x)
var
    i, NomUzla, NormNo: integer;
    x, y, t, x_tr, MA, SumF, x_ot, x_do, x_step, MA_min: double;
    S: String;
begin
    Series1.Clear;
    Series2.Clear;
    Chart1.LeftAxis.Title.Caption:='f (x)';
    Chart1.BottomAxis.Title.Caption:='x';

    if (LENomUzla.Font.Color=clRed)
    or (LEXVot.Font.Color=clRed)
    or (LEXVdo.Font.Color=clRed)
    or (LEXVstep.Font.Color=clRed)
    or (LENormNo.Font.Color=clRed)
    then exit;

    x_ot:=StrToFloat(LEXVot.Text);
    x_do:=StrToFloat(LEXVdo.Text);
    x_step:=StrToFloat(LEXVstep.Text);
    NomUzla:=StrToInt(LENomUzla.Text);
    MA_min:=StrToFloat(LEMamin.Text);

    NormNo:=StrToInt(LENormNo.Text);
    if NormNo=0 then
    begin
        LENormNo.Font.Color:=clRed;
        exit;
    end;

    GetTrjUzel;
    if GetTrjUzelOk=False then exit;

```

```

Str((NomUzla*Trajects.Traject[0].tau+dep_Start)*d_small/epsilon:2:2,S);
Chart1.Title.Text.Text:='Пространственное распределение f(x) '#13
+'на глубине '+S+' Анг';;

x:=x_ot;
While x<=x_do do
begin
SumF:=0;
for i:=NomTrjOt to NomTrjDo do
begin
x_tr:=Trajects.Traject[i].x_arr[NomUzla];
if Trajects.Traject[i].MA_arr[NomUzla]<MA_min
then MA:=Trajects.Traject[i].MA_arr[NomUzla]+MA_min
else MA:=Trajects.Traject[i].MA_arr[NomUzla];

SumF:=SumF+exp(-Sqr(x-x_tr)/abs(2*MA))/Sqrt(abs(2*Pi*MA));
end;

y:=SumF/NormNo;
t:=x;
Series1.AddXY(t,y,'',clBlack);
x:=x+x_step;
end;
RepaintGraphRaspredV;
end;

procedure TFormRaspredV.FvGraph; // Строит график распределения f(v)
var
i, NomUzla, NormNo: integer;
v, y, t, v_tr, MC, SumF, v_ot, v_do, v_step, MC_min: double;
S: String;
begin
Series1.Clear;
Series2.Clear;
Chart1.LeftAxis.Title.Caption:='f (v)';
Chart1.BottomAxis.Title.Caption:='v';
if (LENomUzla.Font.Color=clRed) or (LEXVot.Font.Color=clRed)
or (LEXVdo.Font.Color=clRed) or (LEXVStep.Font.Color=clRed)
or (LENormNo.Font.Color=clRed) then exit;
v_ot:=StrToFloat(LEXVot.Text); v_do:=StrToFloat(LEXVdo.Text);
v_step:=StrToFloat(LEXVStep.Text); NomUzla:=StrToInt(LENomUzla.Text);
MC_min:=StrToFloat(LEMCmin.Text); NormNo:=StrToInt(LENormNo.Text);
if NormNo=0 then
begin
LENormNo.Font.Color:=clRed;
exit;
end;

GetTrjUzel;
if GetTrjUzelOk=False then exit;

Str((NomUzla*Trajects.Traject[0].tau+dep_Start)*d_small/epsilon:2:2,S);
FormRaspredV.Chart1.Title.Text:='Угловое распределение f(v) '#13
+'на глубине '+S+' Анг';;
v:=v_ot;
While v<=v_do do
begin
SumF:=0;
for i:=NomTrjOt to NomTrjDo do
begin
v_tr:=Trajects.Traject[i].v_arr[NomUzla];
if Trajects.Traject[i].MC_arr[NomUzla]<MC_min
then MC:=Trajects.Traject[i].MC_arr[NomUzla]+MC_min
else MC:=Trajects.Traject[i].MC_arr[NomUzla];
SumF:=SumF+exp(-Sqr(v-v_tr)/abs(2*MC))/Sqrt(abs(2*Pi*MC));
end;
y:=SumF/NormNo; t:=v;
Series1.AddXY(t,y,'',clBlack); v:=v+v_step;
end;
RepaintGraphRaspredV;

```

```

end;

procedure TFormRaspredV.FxBtnClick(Sender: TObject);
begin
    FxBtn.Font.Color:=clBlue;  FvBtn.Font.Color:=clBlack;
    LEXVot.EditLabel.Caption:='Xor =';  LEXVdo.EditLabel.Caption:='Xдо =';
    LEXVStep.EditLabel.Caption:='X =';  FvGraph;
end;

procedure TFormRaspredV.FvBtnClick(Sender: TObject);
begin
    FxBtn.Font.Color:=clBlack;  FvBtn.Font.Color:=clBlue;
    LEXVot.EditLabel.Caption:='Vor =';  LEXVdo.EditLabel.Caption:='Vдо =';
    LEXVStep.EditLabel.Caption:='V =';  FvGraph;
end;

procedure TFormRaspredV.UpDownFvxClick(Sender: TObject; Button: TUDBtnType);
begin
    if FvBtn.Font.Color=clBlue then FvGraph else FxGraph;
end;

procedure TFormRaspredV.LEStepUpDownFvxChange(Sender: TObject);
var
    UpDownStep, code: integer;
    Text: String;
begin
    Text:=FormRaspredV.LEStepUpDownFvxv.Text;
    Val(Text, UpDownStep, code);
    if code=0 then UpDownFvxv.Increment:=UpDownStep;
end;

procedure TFormRaspredV.LENomUzlaChange(Sender: TObject);
var
    NomUzla, Code: integer;
    S: String;
begin
    LENomUzla.Font.Color:=clBlack;
    S:=LENomUzla.Text;  Val(S, NomUzla, Code);
    if (code<>0) or (NomUzla<0) or (NomUzla>=Trajects.Traject[0].NodeCount) then
        begin
            LENomUzla.Font.Color:=clRed;  LabelGlubinaTau.Caption:='??? tay';
            LabelGlubinaAngstrem.Caption:='??? АНГ';  exit;
        end;
    LabelGlubinaTau.Caption:=FloatToStr(NomUzla*Trajects.Traject[0].tau+dep_Start)+'
tay';
    LabelGlubinaAngstrem.Caption:=FloatToStr((NomUzla*Trajects.Traject[0].tau+dep_Start)
*d_small/epsilon)+' АНГ';
end;

procedure TFormRaspredV.MminLegendClick(Sender: TObject);
begin
    if (MA_max>=0) and (MC_max>=0) then
        begin
            LEMamin.Text:=FloatToStr(MA_max/4);  LEMCmin.Text:=FloatToStr(MC_max/4);
        end;
end;

procedure TFormRaspredV.LEXVStepChange(Sender: TObject);
begin
    RedErrors(FormRaspredV.LEXVStep, 'No <=0');
end;

procedure TFormRaspredV.LEMAminChange(Sender: TObject);
begin
    RedErrors(FormRaspredV.LEMAmin, 'No <=0');
end;

procedure TFormRaspredV.LEMCminChange(Sender: TObject);
begin
    RedErrors(FormRaspredV.LEMCmin, 'No <=0');
end;

```

```

procedure TFormRaspredV.LEXVotChange(Sender: TObject);
begin
    RedErrors(FormRaspredV.LEXVot, '-');
end;

procedure TFormRaspredV.LEXVdoChange(Sender: TObject);
begin
    RedErrors(FormRaspredV.LEXVdo, '-');
end;

//Общие кнопки на форме FormRasperdV

procedure TFormRaspredV.Zapolnenie; //процедура заполняет данными форму
begin
    LEMamin.Text:=Settings[29];
    LEMCmin.Text:=Settings[30];
    LEXVot.Text:=Settings[31];
    LEXVdo.Text:=Settings[32];
    LEXVstep.Text:=Settings[33];
    LEhV.Text:=Settings[34];
    LECalcVot.Text:=Settings[35];
    LECalcVdo.Text:=Settings[36];
    LEhX.Text:=Settings[37];
    LECalcXot.Text:=Settings[38];
    LECalcXdo.Text:=Settings[39];
    LECalcGlot.Text:=Settings[42];
    LECalcGldo.Text:=Settings[43];
    LESTepUpDownFxv.Text:=Settings[66];
    LESTepUpDownGistX.Text:=Settings[67];
    LESTepUpDownGistV.Text:=Settings[68];
end;

procedure TFormRaspredV.FormCreate(Sender: TObject);
begin
    Zapolnenie; FxBtn.Font.Color:=clBlue;
end;

procedure TFormRaspredV.CloseFormBtnClick(Sender: TObject);
begin
    FormRaspredV.Close;
end;

procedure TFormRaspredV.RepaintGraphRaspredV;
begin
    Chart1.Repaint;
    LEGraphYmax.Text:=FloatToStrF(Chart1.LeftAxis.Maximum,ffGeneral,9,9);
    LEGraphYmin.Text:=FloatToStrF(Chart1.LeftAxis.Minimum,ffGeneral,9,9);
    LEGraphXmax.Text:=FloatToStrF(Chart1.BottomAxis.Maximum,ffGeneral,9,9);
    LEGraphXmin.Text:=FloatToStrF(Chart1.BottomAxis.Minimum,ffGeneral,9,9);
end;

procedure TFormRaspredV.SBChangeGraphClick(Sender: TObject);
begin
    if ChBoxGraphYAuto.Checked=True then
        Chart1.LeftAxis.Automatic:=True
    else
        begin
            Chart1.LeftAxis.Automatic:=False;
            try
                Chart1.LeftAxis.Minimum:=StrToFloat(LEGraphYmin.Text);
                Chart1.LeftAxis.Maximum:=StrToFloat(LEGraphYmax.Text);
            except
                exit;
            end;
        end;

    if ChBoxGraphXAuto.Checked=True then
        Chart1.BottomAxis.Automatic:=True
    else
        begin

```

```

    Chart1.BottomAxis.Automatic:=False;
    try
    Chart1.BottomAxis.Minimum:=StrToFloat(LEGraphXmin.Text);
    Chart1.BottomAxis.Maximum:=StrToFloat(LEGraphXmax.Text);
    except
    exit;
    end;
end;
SBChangeGraph.Enabled:=False; RepaintGraphRaspredV;
end;

procedure TFormRaspredV.LEGraphYminChange(Sender: TObject);
begin
    if ChBoxGraphYAuto.Checked=False then SBChangeGraph.Enabled:=True;
end;

procedure TFormRaspredV.LEGraphXminChange(Sender: TObject);
begin
    if ChBoxGraphXAuto.Checked=False then SBChangeGraph.Enabled:=True;
end;

procedure TFormRaspredV.ChBoxGraphYAutoClick(Sender: TObject);
begin
    SBChangeGraph.Enabled:=True;
end;

procedure TFormRaspredV.LabelForAllTrjClick(Sender: TObject);
begin
    LENomTrjOt.Text:='0';
    LENomTrjDo.Text:=IntToStr(Trajects.CntOfTraject-1);
    LENormNo.Text:=IntToStr(Trajects.CntOfTraject);
    LENormNoGlv.Text:=IntToStr(Trajects.CntOfTraject);
    LENormNoGistV.Text:=IntToStr(Trajects.CntOfTraject);
    LENormNoGistX.Text:=IntToStr(Trajects.CntOfTraject);
end;

procedure TFormRaspredV.LabelForAllUzelsClick(Sender: TObject);
begin
    LENomUzlaOt.Text:='0';
    LENomUzlaDo.Text:=IntToStr(Trajects.Traject[0].NodeCount-1);
end;

procedure TFormRaspredV.GetTrjUzel;
begin
    GetTrjUzelOk:=False;
    if (LENomTrjOt.Font.Color=clRed) or (LENomTrjDo.Font.Color=clRed)
        or (LENomUzlaOt.Font.Color=clRed) or (LENomUzlaDo.Font.Color=clRed)
    then exit;
    NomTrjOt:=StrToInt(LENomTrjOt.Text);
    if NomTrjOt<0 then NomTrjOt:=0;
    if NomTrjOt>(Trajects.CntOfTraject-1) then NomTrjOt:=Trajects.CntOfTraject-1;
    LENomTrjOt.Text:=IntToStr(NomTrjOt);
    NomTrjDo:=StrToInt(LENomTrjDo.Text);
    if NomTrjDo<NomTrjDo then NomTrjDo:=NomTrjOt;
    if NomTrjDo>(Trajects.CntOfTraject-1) then NomTrjDo:=Trajects.CntOfTraject-1;
    LENomTrjDo.Text:=IntToStr(NomTrjDo);
    NomUzlaOt:=StrToInt(LENomUzlaOt.Text);
    if NomUzlaOt<0 then NomUzlaOt:=0;
    if NomUzlaOt>(Trajects.Traject[0].NodeCount-1) then
        NomUzlaOt:=Trajects.Traject[0].NodeCount-1;
    LENomUzlaOt.Text:=IntToStr(NomUzlaOt);
    NomUzlaDo:=StrToInt(LENomUzlaDo.Text);
    if NomUzlaDo<NomUzlaOt then NomUzlaDo:=NomUzlaDo;
    if NomUzlaDo>(Trajects.Traject[0].NodeCount-1) then
        NomUzlaDo:=Trajects.Traject[0].NodeCount-1;
    LENomUzlaDo.Text:=IntToStr(NomUzlaDo);
    GetTrjUzelOk:=True;
end;

procedure TFormRaspredV.LENomTrjOtChange(Sender: TObject);
begin

```

```

    RedErrors (FormRaspredV.LENomTrjOt, 'Integer');
end;

procedure TFormRaspredV.LENomTrjDoChange(Sender: TObject);
begin
    RedErrors (FormRaspredV.LENomTrjDo, 'Integer');
end;

procedure TFormRaspredV.LENomUzlaOtChange(Sender: TObject);
begin
    RedErrors (FormRaspredV.LENomUzlaOt, 'Integer');
end;

procedure TFormRaspredV.LENomUzlaDoChange(Sender: TObject);
begin
    RedErrors (FormRaspredV.LENomUzlaDo, 'Integer');
end;

procedure TFormRaspredV.LENormNoChange(Sender: TObject);
begin
    RedErrors (FormRaspredV.LENormNo, 'Integer');
end;

procedure TFormRaspredV.ShowGridClick(Sender: TObject);
begin
    if ShowGrid.Checked=False then
        begin
            Chart1.LeftAxis.Grid.Visible:=True;    Chart1.BottomAxis.Grid.Visible:=True;
            ShowGrid.Checked:=True;
        end
    else
        begin
            Chart1.LeftAxis.Grid.Visible:=False;    Chart1.BottomAxis.Grid.Visible:=False;
            ShowGrid.Checked:=False;
        end;
    end;

procedure TFormRaspredV.ShowTextClick(Sender: TObject);
begin
    if ShowText.Checked=False then
        begin
            Chart1.Foot.Text.Text:=
                FloatToStr(Trajects.CntOfTraject)+' tp.  X='
                +FloatToStrF(Trajects.Traject[0].x_arr[0],ffGeneral,3,3)+'...'
                +FloatToStrF(Trajects.Traject[Trajects.CntOfTraject-1].x_arr[0],ffGeneral,3,3)
                +' V='+FloatToStrF(Trajects.Traject[0].v_arr[0],ffGeneral,3,3)+'...'
                +FloatToStrF(Trajects.Traject[Trajects.CntOfTraject-1].v_arr[0],ffGeneral,3,3)
                +' Z1='+FloatToStr(Z1)+' M='+FloatToStr(MassOfIon/m0c2)+'aem  T='
                +FloatToStr(EnergyT/1000000)+'MэB'#13
                +' ypoл='+FloatToStr(DeltaTheta*180/Pi)+'град'+ ' k='+FloatToStr(k)
                +' l='+FloatToStr(l)+' ' +Ploskost+' Curv='+FloatToStr(kt)+'*t+'
                +FloatToStr(Curv*Sqr(epsilon)/d_small)+'*d/eps^2';
            Chart1.Foot.Visible:=True;    Chart1.Title.Visible:=True;
            ShowText.Checked:=True;
        end
    else
        begin
            Chart1.Foot.Visible:=False;    Chart1.Title.Visible:=False;
            ShowText.Checked:=False;
        end;
    end;

procedure TFormRaspredV.SaveGraphClick(Sender: TObject);
begin
    if Form1.SavePictureDialog1.Execute then
        begin
            Chart1.SaveToMetafileEnh(Form1.SavePictureDialog1.FileName);
        end;
    end;

procedure TFormRaspredV.PrintGraphClick(Sender: TObject);

```

```

begin
  if Form1.PrintDialog1.Execute then Chart1.Print;
end;

procedure TFormRaspredV.ToClipBrdClick(Sender: TObject);
begin
  Chart1.CopyToClipboardMetafile(true);
end;

procedure TFormRaspredV.SBExcelClick(Sender: TObject);
var
  i: integer;
  ExcelApplication: Variant;
begin
  ExcelApplication:=CreateOleObject ('Excel.Application');
  ExcelApplication.Visible:=True;  ExcelApplication.WorkBooks.Add;

  for i:=1 to Series1.Count do
    begin
      ExcelApplication.Cells[i,1].Value:=Series1.XValue[i-1];
      ExcelApplication.Cells[i,2].Value:=Series1.YValue[i-1];
    end;
  end;

procedure TFormRaspredV.SpeedButton1Click(Sender: TObject);
var
  i: integer;
  ExcelApplication: Variant;
begin
  ExcelApplication:=CreateOleObject ('Excel.Application');
  ExcelApplication.Visible:=True;
  ExcelApplication.WorkBooks.Add;

  for i:=1 to Series1.Count do
    begin
      ExcelApplication.Cells[i,1].Value:=Series1.XValue[i-1];
      ExcelApplication.Cells[i,2].Value:=Series1.YValue[i-1];
    end;
  end;

procedure TFormRaspredV.SpeedButton2Click(Sender: TObject);
var
  i,n: integer;
  x,y: double;
  ExcelApplication: Variant;
begin
  ExcelApplication:=CreateOleObject ('Excel.Application');
  ExcelApplication.Visible:=True;
  ExcelApplication.WorkBooks.Add;

  i:=1;
  n:=0;
  While i<Series2.Count do
    begin
      inc(n);
      x:=Series2.XValue[i-1];
      y:=Series2.YValue[i-1];
      ExcelApplication.Cells[n,1].Value:=x;
      ExcelApplication.Cells[n,2].Value:=y;
      inc(i);
      inc(n);
      y:=Series2.YValue[i-1];
      ExcelApplication.Cells[n,1].Value:=x;
      ExcelApplication.Cells[n,2].Value:=y;
    end;

end;

end.

```

unit RaspredE;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, Buttons, Gauges, Constants, Trajectories, TeEngine,
Series, ExtCtrls, TeeProcs, Chart, ExtDlgs, ComCtrls, Menus;

type

```
TFormRaspredE = class(TForm)
    Chart1: TChart;
    Series1: TPointSeries;
    Series2: TPointSeries;
    Series3: TLineSeries;
    PageRaspVXGl: TPageControl;
    TabSheet1: TTabSheet;
    LabelDelthaV: TLabel;
    GistEBtn: TBitBtn;
    GroupBoxCalcE: TGroupBox;
    CalcVLegend: TLabel;
    TabSheet3: TTabSheet;
    RaspGlLegend: TLabel;
    RaspEglBtn: TBitBtn;
    GroupBoxCalcGl: TGroupBox;
    Label9: TLabel;
    CalcGlLegend: TLabel;
    LECalcGlot: TLabeledEdit;
    LECalcGlDo: TLabeledEdit;
    CalcGlBtn: TBitBtn;
    TabSheet4: TTabSheet;
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Label19: TLabel;
    RaspEBtn: TBitBtn;
    TabSheet2: TTabSheet;
    Gauge3: TGauge;
    EnergyArrayBtn: TBitBtn;
    ChBoxRoe07: TCheckBox;
    ChBoxGammaDeltha: TCheckBox;
    ChBGisteVAng: TCheckBox;
    Label15: TLabel;
    GraphETBtn: TBitBtn;
    Gauge2: TGauge;
    ChBAngstrem: TCheckBox;
    ChBeVAngstrem: TCheckBox;
    Series4: TLineSeries;
    Label1: TLabel;
    CloseFormBtn: TSpeedButton;
    LabelEGLubina: TLabel;
    LEENomUzla: TLabeledEdit;
    UpDownGistE: TUpDown;
    LabelEGLubinaTau: TLabel;
    LabelEGLubinaAngstrem: TLabel;
    LEStepUpDownGistE: TLabeledEdit;
    LECalcEot: TLabeledEdit;
    LECalcEdo: TLabeledEdit;
    DecEPerAngBtn: TBitBtn;
    CalcEBtn: TBitBtn;
    LabelResCalcE: TLabel;
    Label2: TLabel;
    ChBoxDecELGAng: TCheckBox;
    ChBoxNormGistEL: TCheckBox;
    LEhE: TLabeledEdit;
    LEEminGl: TLabeledEdit;
    LEEmaxGl: TLabeledEdit;
    LabelCalcGlRes: TLabel;
    LERazreshSpectra: TLabeledEdit;
    ChBoxWithhE: TCheckBox;
    ChBLayerEn: TCheckBox;
```



```

LELayerEn: TLabelEdit;
Label3: TLabel;
LabelSDec: TLabel;
LEGraphYmax: TLabelEdit;
LEGraphYmin: TLabelEdit;
LEGraphXmax: TLabelEdit;
LEGraphXmin: TLabelEdit;
ChBoxGraphYAuto: TCheckBox;
ChBoxGraphXAuto: TCheckBox;
SBChangeGraph: TSpeedButton;
ChBoxNormRandom: TCheckBox;
LERandom: TLabelEdit;
LabelAveDec: TLabel;
LENormNo: TLabelEdit;
Bevel4: TBevel;
Label4: TLabel;
Label6: TLabel;
LENomTrjOt: TLabelEdit;
LENomTrjDo: TLabelEdit;
LabelForAllTrj: TLabel;
Label10: TLabel;
LabelForAllUzels: TLabel;
LENomUzlaOt: TLabelEdit;
LENomUzlaDo: TLabelEdit;
Bevel1: TBevel;
Label7: TLabel;
Label8: TLabel;
Bevel2: TBevel;
Bevel3: TBevel;
Bevel5: TBevel;
Bevel6: TBevel;
BevelOrZav: TBevel;
LabelOrZav: TLabel;
Label11: TLabel;
Bevel7: TBevel;
SBExcel: TSpeedButton;
ChBoxAllExcel: TCheckBox;
LECenaKanala: TLabelEdit;
Label12: TLabel;
LabelGlubinaSpectraE: TLabel;
LabelTauSpectraE: TLabel;
LENomUzlaSpectraE: TLabelEdit;
LEStepSpectraE: TLabelEdit;
LabelAngSpectraE: TLabel;
UpDownSpectraE: TUpDown;
ChBeVAngSpectraE: TCheckBox;
SpeedButton2: TSpeedButton;
PopupMenuChartE: TPopupMenu;
ShowGrid: TMenuItem;
ShowText: TMenuItem;
N3: TMenuItem;
SaveGraph: TMenuItem;
PrintGraph: TMenuItem;
ToClipBrd: TMenuItem;
Excell: TMenuItem;
procedure GistEBtnClick(Sender: TObject);
procedure EnergyArrayBtnClick(Sender: TObject);
procedure RasPEGLBtnClick(Sender: TObject);
procedure GraphETBtnClick(Sender: TObject);
procedure RasPEBtnClick(Sender: TObject);
procedure ChBGisteVAngClick(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure ChBoxGammaDelthaClick(Sender: TObject);
procedure CalcGlBtnClick(Sender: TObject);
procedure CalcEBtnClick(Sender: TObject);
procedure CloseFormBtnClick(Sender: TObject);
procedure UpDownGistEClick(Sender: TObject; Button: TUDBtnType);
procedure LEENomUzlaChange(Sender: TObject);
procedure LEStepUpDownGistEChange(Sender: TObject);
procedure DecEPerAngBtnClick(Sender: TObject);
procedure LEhEChange(Sender: TObject);

```

```

procedure LECalcEotChange(Sender: TObject);
procedure LECalcEdoChange(Sender: TObject);
procedure LEEminGlChange(Sender: TObject);
procedure LEEmaxGlChange(Sender: TObject);
procedure LECalcGlotChange(Sender: TObject);
procedure LECalcGldoChange(Sender: TObject);
procedure LERazreshSpectraChange(Sender: TObject);
procedure LELayerEnChange(Sender: TObject);
procedure SBChangeGraphClick(Sender: TObject);
procedure LEGraphYminChange(Sender: TObject);
procedure LEGraphXminChange(Sender: TObject);
procedure ChBoxGraphYAutoClick(Sender: TObject);
procedure LabelForAllTrjClick(Sender: TObject);
procedure LabelForAllUzelsClick(Sender: TObject);
procedure LENomTrjOtChange(Sender: TObject);
procedure LENomTrjDoChange(Sender: TObject);
procedure LENomUzlaOtChange(Sender: TObject);
procedure LENomUzlaDoChange(Sender: TObject);
procedure LENormNoChange(Sender: TObject);
procedure LabelOrZavClick(Sender: TObject);
procedure SBExcelClick(Sender: TObject);
procedure LECenaKanalaChange(Sender: TObject);
procedure LESTepSpectraEChange(Sender: TObject);
procedure LENomUzlaSpectraEChange(Sender: TObject);
procedure UpDownSpectraEClick(Sender: TObject; Button: TUDBtnType);
procedure ChBeVAngSpectraEClick(Sender: TObject);
procedure SpeedButton2Click(Sender: TObject);
procedure ShowGridClick(Sender: TObject);
procedure ShowTextClick(Sender: TObject);
procedure SaveGraphClick(Sender: TObject);
procedure PrintGraphClick(Sender: TObject);
procedure ToClipBrdClick(Sender: TObject);
procedure ExcellClick(Sender: TObject);

private
  { Private declarations }
  NomTrjOt, NomTrjDo, NomUzlaOt, NomUzlaDo: integer;
  GetTrjUzelOk: Boolean;
  function roe(x:double):double;
  function RightPartE(x, Le_Energy:double):double;
  procedure GetTrjUzel;
  procedure GistogrammE;
  procedure SpectraE;
  procedure RepaintGraphEnergy;
public
  { Public declarations }
  procedure Zapolnenie;
  procedure EnergyReady;
  procedure SaveGrOut;
end;

var
  FormRaspredE: TFormRaspredE;

implementation

uses Dechanneling, OrientZav, ChartXVT, ComObj;

{$R *.dfm}

//Пачэт

function TFormRaspredE.roe(x:double):double;
var
  roe07: double;
begin
  if ChBoxRoe07.Checked=False then
    begin
      roe:=ro_e(x);
    end
  else

```

```

begin
    roe07:=NE*Z2;
    roe:=(ro_e(x)+roe07)/2;
end;
end;

function TFormRaspredE.RightPartE(x, Le_Energy:double):double;
begin
    RightPartE:=roe(x)*Le_Energy;
end;

procedure TFormRaspredE.EnergyArrayBtnClick(Sender: TObject);
var
    i,j: integer;
    Energy, Energy1, dt, x, Le_Energy: double;
    Progress: integer;
begin
    Gauge3.MaxValue:=Trajects.CntOfTraject*Trajects.Traject[0].NodeCount;
    Gauge3.ForeColor:=clBlue;
    Gauge3.Progress:=0;
    Form1.Gauge1.MaxValue:=Trajects.CntOfTraject*Trajects.Traject[0].NodeCount;
    Form1.Gauge1.ForeColor:=clInfoBk;
    Form1.Gauge1.Progress:=0;
    SetLength(Energy_arr, Trajects.CntOfTraject, Trajects.Traject[0].NodeCount);

    if ChBoxGammaDeltha.Checked=False
    then Le_Energy:=ln(2*mec2*betaE2/IE)
    else Le_Energy:=ln(2*mec2*betaE2*Sqr(gammaE)/IE)-betaE2-deltaE/2;
    Le_Energy:=Le_Energy*d_small*2*DvaPi*Sqr(Z1)*Sqr(e2)/(epsilon*mec2*betaE2);

    For i:=0 to Trajects.CntOfTraject-1 do
        begin
            dt:=Trajects.Traject[i].tau;
            Energy_arr[i,0]:=Energy_Start[i];
            Energy:=Energy_Start[i];
            for j:=1 to Trajects.Traject[i].NodeCount-1 do
                begin
                    x:=Trajects.Traject[i].x_arr[j];
                    Energy1:=Energy;
                    Energy:=Energy+RightPartE(x, Le_Energy)*dt;
                    Energy_arr[i,j]:=(Energy+Energy1)/2;

                    Progress:=i*Trajects.Traject[0].NodeCount+j+1;
                    Gauge3.Progress:=Progress;
                    Form1.Gauge1.Progress:=Progress;
                end;
            Energy_Last[i]:=Energy_arr[i, (Trajects.Traject[i].NodeCount-1)];
        end;

    EnergyReady;

end;

procedure TFormRaspredE.EnergyReady;
var
    Text: String;
begin
    GraphETBtn.Enabled:=True;
    FormOrientZav.ChBoxOriEL.Enabled:=True;
    FormChartXVT.ChartEnBtn.Enabled:=True;
    FormChartXVT.ChartEeVAngBtn.Enabled:=True;

    Text:=FormRaspredE.Caption;

    Chart1.Foot.Text.Text:=
        FloatToStr(Trajects.CntOfTraject)+' tp. X='
        +FloatToStrF(Trajects.Traject[0].x_arr[0], ffGeneral, 3, 3)+'...'
        +FloatToStrF(Trajects.Traject[Trajects.CntOfTraject-1].x_arr[0], ffGeneral, 3, 3)
        +' V='+FloatToStrF(Trajects.Traject[0].v_arr[0], ffGeneral, 3, 3)+'...'
        +FloatToStrF(Trajects.Traject[Trajects.CntOfTraject-1].v_arr[0], ffGeneral, 3, 3)
        +' Z1='+FloatToStr(Z1)+' M='+FloatToStr(MassOfIon/m0c2)+'aem T='

```

```

+FloatToStr(EnergyT/1000000)+'МэВ'#13
+' ypoл='+FloatToStr(DeltaTheta*180/Pi)+'град'+' k='+FloatToStr(k)
+' l='+FloatToStr(l)+' '+Ploskost+' Curv='+FloatToStr(kt)+'*t+'
+FloatToStr(Curv*Sqr(epsilon)/d_small)+'*d/eps^2'#13
+Text;
end;

procedure TFormRaspredE.GraphETBtnClick(Sender: TObject);
var
  y, t: double;
  i, j: integer;
begin
  Chart1.FreeAllSeries;
  if ChBAngstrem.Checked=False
  then Chart1.BottomAxis.Title.Caption:='г л у б и н а,   т а у'
  else Chart1.BottomAxis.Title.Caption:='г л у б и н а,   А н г.';
  if ChBeVAngstrem.Checked=False
  then Chart1.LeftAxis.Title.Caption:='Е - Ео (потерянная энергия),   эВ'
  else Chart1.LeftAxis.Title.Caption:='Е - Ео (потерянная энергия),   эВ/Анг.';

  GetTrjUzel; if GetTrjUzelOk=False then exit;

  Gauge2.MaxValue:=NomTrjDo-NomTrjOt+1;

  For i:=NomTrjOt to NomTrjDo do
    begin
      Series3:=Series.TLineSeries.Create(Chart1);
      Chart1.AddSeries(Series3);
      for j:=NomUzlaOt to NomUzlaDo do
        begin
          if ChBeVAngstrem.Checked=True
          then y:=Energy_arr[i,j]/(Trajects.Traject[i].tau*(j+1)
                                +Trajects.Traject[i].t_arr[0])*epsilon/d_small
          else y:=Energy_arr[i,j];
          t:=Trajects.Traject[i].t_arr[j];
          if ChBAngstrem.Checked=False
          then Series3.AddXY(t,y,'',clBlack)
          else Series3.AddXY(t*d_small/epsilon,y,'',clBlack);
        end;
      Gauge2.Progress:=i+1-NomTrjOt;
    end;

  Chart1.Title.Text.Text:='Потери энергии частиц с глубиной';
  RepaintGraphEnergy;
end;

procedure TFormRaspredE.ChBoxGammaDelthaClick(Sender: TObject);
begin
  Gauge3.Progress:=0;
  Gauge2.Progress:=0;

  // Подпись:

  if ChBoxGammaDeltha.Checked=False then
  begin
    if ChBoxRoe07.Checked=False
    then FormRaspredE.Caption:='Энергия: вариант № 1:'
    + ' Без gamma, delta; roe:=roe(x)'
    else FormRaspredE.Caption:='Энергия: вариант № 2:'
    + ' Без gamma, delta; roe:=(roe(x)+roe07)/2';
  end
  else
  begin
    if ChBoxRoe07.Checked=False
    then FormRaspredE.Caption:='Энергия: вариант № 3:'
    + ' Учёт gamma, delta; roe:=roe(x)'
    else FormRaspredE.Caption:='Энергия: вариант № 4:'
    + ' Учёт gamma, delta; roe:=(roe(x)+roe07)/2';
  end;
end;
end;

```

```

//Гистограмма по E

procedure TFormRaspredE.GistogrammE;
var
  Emin, Emax, hE, Ej: double; //hE - шаг разбиения отрезков по энергиям
  i, j, NomUzla: integer;      //Ej - среднее значение энергии на отрезке
  nE: array of integer;        //nE - кол-во частиц на отрезке
  EnForGist_arr: En_arr_Type;  //Массив энергии: задаётся в "эВ" или "эВ/Анг"
  S: String;
  NumUzelsInLayer, FirstUzel, LastUzel, NL: integer;
  NormNo: integer;

begin
  Chart1.FreeAllSeries;
  Chart1.LeftAxis.Title.Caption:='N (количество частиц), шт.';
  Series4:=Series.TLineSeries.Create(Chart1);
  Series4.Stairs:=True;
  Chart1.AddSeries(Series4);

  if (LEENomUzla.Font.Color=clRed)
    or (LEhE.Font.Color=clRed)
    or (LENormNo.Font.Color=clRed)
  then exit;

  NomUzla:=StrToInt(LEENomUzla.Text);
  hE:=StrToFloat(LEhE.Text);
  NormNo:=StrToInt(LENormNo.Text);

  GetTrjUzel;
  if GetTrjUzelOk=False then exit;

  SetLength(EnForGist_arr, Trajects.CntOfTraject, Trajects.Traject[0].NodeCount);

if ChBGisteVAng.Checked=False then // Гистограмма в эВ
begin
  Chart1.BottomAxis.Title.Caption:='E - Eo (потерянная энергия), эВ';
  EnForGist_arr:=Energy_arr;
end
else if ChBGisteVAng.Checked=True then // Гистограмма в эВ/Анг
begin
  if ChBLayerEn.Checked=False then
  begin
    Chart1.BottomAxis.Title.Caption:='E - Eo (потерянная энергия), эВ/Анг.';
    for i:=NomTrjOt to NomTrjDo do begin
      for j:=0 to Trajects.Traject[0].NodeCount-1 do begin
        EnForGist_arr[i,j]:=Energy_arr[i,j]/(Trajects.Traject[i].tau*(j+1)
          +Trajects.Traject[i].t_arr[0])*epsilon/d_small;
      end;
    end;
  end
else
  begin
    Chart1.BottomAxis.Title.Caption:='dE слоя [эВ] / dT слоя [Анг]';
    if (LELayerEn.Font.Color=clRed) then exit;
    NumUzelsInLayer:=StrToInt(LELayerEn.Text);

    if (NumUzelsInLayer<=0)
      or ((Trajects.Traject[0].NodeCount mod NumUzelsInLayer)<>0) then
      if Application.MessageBox(' Заданная длина слоёв в узлах'+#13+' '+#13+
        ' не годятся для расчёта гистограммы,'+#13+' '+#13+
        ' т.к. приводит к неравному делению на слои.'+#13+' '+#13+
        'При выполнении может произойти ошибка'+#13+' '+#13+
        ' ПРОДОЛЖИТЬ ВЫПОЛНЕНИЕ ?',
        'Предупреждение',
        MB_YesNo+MB_IconWarning) = mrNo then exit;

    for i:=NomTrjOt to NomTrjDo do begin
      NL:=1;
      FirstUzel:=0;
      LastUzel:=NumUzelsInLayer; // -1
      for j:=0 to Trajects.Traject[0].NodeCount-1 do begin

```

```

        EnForGist_arr[i,j] := (Energy_arr[i,LastUzel]-Energy_arr[i,FirstUzel])
                               / (Trajects.Traject[i].tau*NumUzelsInLayer)
                               *epsilon/d_small;

    if j>=LastUzel then
    begin
        inc (NL);
        FirstUzel := (NL-1)*NumUzelsInLayer;
        LastUzel := NL*NumUzelsInLayer;  //-1
    end;
end;
end;
end;
end;

{Автоматическое определение границ с помощью проверки всех значений EnForGist_arr}
Emin:=EnForGist_arr[0,0];
Emax:=EnForGist_arr[0,0];
for i:=NomTrjOt to NomTrjDo do begin
    for j:=0 to Trajects.Traject[0].NodeCount-1 do begin
        if EnForGist_arr[i,j]<Emin then Emin:=EnForGist_arr[i,j];
        if EnForGist_arr[i,j]>Emax then Emax:=EnForGist_arr[i,j];
    end;
end;
Emin:=Emin-hE;
Emax:=Emax+3*hE;

SetLength(nE, round((Emax-Emin)/hE)+4);

Ej:=Emin;
For j:=1 to round((Emax-Emin)/hE)+1 do
begin
    nE[j]:=0;
    for i:=NomTrjOt to NomTrjDo do
    begin
        if (EnForGist_arr[i,NomUzla]<=Ej+hE/2) and (EnForGist_arr[i,NomUzla]>Ej-hE/2)
            then nE[j]:=nE[j]+1;
    end;
    Ej:=Ej+hE;
end;

if ChBoxNormGistEL.Checked=True
then Chart1.LeftAxis.Title.Caption:='N / No  ( No = '+IntToStr(NormNo)+' )';

For j:=1 to round((Emax-Emin)/hE)+1 do
begin
    Ej:=Emin+hE*(j-1.5);
    if ChBoxNormGistEL.Checked=False
    then Series4.AddXY(Ej,nE[j],'',clBlack)
    else Series4.AddXY(Ej,nE[j]/NormNo,'',clBlack);
end;

Finalize(nE);
Finalize(EnForGist_arr);

Str((NomUzla*Trajects.Traject[0].tau+dep_Start)*d_small/epsilon:2:2,S);
Chart1.Title.Text.Text:='Гистограмма распределения частиц по '
                        +'скоростям потери энергии'#13
                        +'на глубине '+S+' Анг';
FormRaspredE.Caption:='Гистограмма распределения частиц по энергиям '
                        +'на глубине '+S+' Анг.';

RepaintGraphEnergy;
end;

procedure TFormRaspredE.GistEBtnClick(Sender: TObject);
begin
    GistogrammE;
end;

procedure TFormRaspredE.UpDownGistEClick(Sender: TObject;
    Button: TUDBtnType);

```

```

begin
    GistogrammE;
end;

procedure TFormRaspredE.LEStepUpDownGistEChange(Sender: TObject);
var
    UpDownStep, code: integer;
    Text: String;
begin
    Text:=FormRaspredE.LEStepUpDownGistE.Text;
    Val(Text, UpDownStep, code);
    if code=0
    then FormRaspredE.UpDownGistE.Increment:=UpDownStep;
end;

procedure TFormRaspredE.LEENomUzlaChange(Sender: TObject);
var
    NomUzla, Code: integer;
    S: String;
begin
    LEENomUzla.Font.Color:=clBlack;
    S:=LEENomUzla.Text;
    Val(S, NomUzla, Code);
    if (code<>0) or (NomUzla<0) or (NomUzla>=Trajects.Traject[0].NodeCount) then
    begin
        LEENomUzla.Font.Color:=clRed;
        LabelEGLubinaTau.Caption:='??? тay';
        LabelEGLubinaAngstrem.Caption:='??? АНГ';
        exit;
    end;
    LabelEGLubinaTau.Caption:=
        FloatToStr(NomUzla*Trajects.Traject[0].tau+dep_Start)+' тay';
    LabelEGLubinaAngstrem.Caption:=FloatToStr
        ((NomUzla*Trajects.Traject[0].tau+dep_Start)*d_small/epsilon)+' АНГ';
end;

procedure TFormRaspredE.LEhEChange(Sender: TObject);
begin
    RedErrors(FormRaspredE.LEhE, 'No <=0');
end;

procedure TFormRaspredE.ChBGisteVAngClick(Sender: TObject);
begin
    if ChBGisteVAng.Checked=True then
    begin
        LEhE.Text:='1';
        Label5.Caption:='эB/АНГ';
        Label2.Caption:='эB/АНГ';
        GroupBoxCalcE.Visible:=False;
        DecEPerAngBtn.Visible:=True;
        ChBoxDecELG1Ang.Visible:=True;
        ChBoxWithhE.Visible:=True;
        ChBLayerEn.Enabled:=True;
        LELayerEn.Enabled:=True;
        Label3.Enabled:=True;
        LabelSDec.Visible:=True;
        ChBoxNormRandom.Visible:=True;
        LERandom.Visible:=True;
        LabelAveDec.Visible:=True;
    end
    else //if ChBGisteVAng.Checked=False then
    begin
        LEhE.Text:='500';
        Label5.Caption:=' эB';
        Label2.Caption:=' эB';
        GroupBoxCalcE.Visible:=True;
        DecEPerAngBtn.Visible:=False;
        ChBoxDecELG1Ang.Visible:=False;
        ChBoxWithhE.Visible:=False;
        ChBLayerEn.Enabled:=False;
        LELayerEn.Enabled:=False;
    end
end;

```

```

    Label3.Enabled:=False;
    LabelSDec.Visible:=False;
    ChBoxNormRandom.Visible:=False;
    LERandom.Visible:=False;
    LabelAveDec.Visible:=False;
  end;
end;

procedure TFormRaspredE.CalcEBtnClick(Sender: TObject);
var
  Emin, Emax, Energy: double;
  i, nE, NomUzla: integer;
begin
  if (LEENomUzla.Font.Color=clRed)
    or (LECalcEot.Font.Color=clRed)
    or (LECalcEdo.Font.Color=clRed)
  then exit;
  Emin:=StrToFloat(LECalcEot.Text);
  Emax:=StrToFloat(LECalcEdo.Text);
  NomUzla:=StrToInt(LEENomUzla.Text);
  nE:=0;
  For i:=0 to Trajects.CntOfTraject-1 do
    begin
      Energy:=Energy_arr[i,NomUzla];
      if (Energy>=Emin) and (Energy<Emax) then nE:=nE+1;
    end;
  LabelResCalcE.Caption:=IntToStr(nE);
end;

procedure TFormRaspredE.DecEPerAngBtnClick(Sender: TObject);
var
  Emin, Emax, hE: double;
  i, j: integer;
  SD: integer;
  DecEPerAng_arr: array of integer;
  EnPerAng_arr: En_arr_Type;
  NumUzelsInLayer, FirstUzel, LastUzel, NL: integer;
  RandomDec, AveDec: double;
  NormNo: integer;

begin
  Chart1.FreeAllSeries;
  Series4:=Series.TLineSeries.Create(Chart1);
  Series4.Stairs:=True;
  Chart1.AddSeries(Series4);

  if (LEENomUzla.Font.Color=clRed)
    or (LECalcEot.Font.Color=clRed)
    or (LECalcEdo.Font.Color=clRed)
    or (LENormNo.Font.Color=clRed)
  then exit;

  Emin:=StrToFloat(LECalcEot.Text);
  Emax:=StrToFloat(LECalcEdo.Text);

  GetTrjUzel;
  if GetTrjUzelOk=False then exit;

  if ChBoxNormGistEL.Checked=True then
    begin
      NormNo:=StrToInt(LENormNo.Text);
      Chart1.LeftAxis.Title.Caption:='Ндек / No    ( No = '+IntToStr(NormNo)+ ')';
    end
  else
    begin
      NormNo:=1;
      Chart1.LeftAxis.Title.Caption:='Ндек, шт.';
    end;

  SetLength(EnPerAng_arr, Trajects.CntOfTraject, Trajects.Traject[0].NodeCount);

```



```

if ChBLayerEn.Checked=False then
  begin
    for i:=NomTrjOt to NomTrjDo do begin
      for j:=NomUzlaOt to NomUzlaDo do begin
        EnPerAng_arr[i,j]:=Energy_arr[i,j]/(Trajects.Traject[i].tau*(j+1)
          +Trajects.Traject[i].t_arr[0])*epsilon/d_small;
      end;
    end;
  end
else // Способ "по слоям"
  begin
    if (LELayerEn.Font.Color=clRed) then exit;
    NumUzelsInLayer:=StrToInt(LELayerEn.Text);

    if (NumUzelsInLayer<=0)
      or ((Trajects.Traject[0].NodeCount mod NumUzelsInLayer)<>0) then
      if Application.MessageBox('    Заданная длина слоёв в узлах'+#13+' '+#13+
        '    не годятся для расчёта деканализирования,'+#13+' '+#13+
        '    т.к. приводит к неравному делению на слои.'+#13+' '+#13+
        'При выполнении может произойти ошибка'+#13+' '+#13+
        '    ПРОДОЛЖИТЬ ВЫПОЛНЕНИЕ ?',
        'Предупреждение',
        MB_YesNo+MB_IconWarning) = mrNo then exit;

    for i:=NomTrjOt to NomTrjDo do begin
      NL:=1;
      FirstUzel:=0;
      LastUzel:=NumUzelsInLayer; // -1;
      for j:=NomUzlaOt to NomUzlaDo do begin
        EnPerAng_arr[i,j]:=(Energy_arr[i,LastUzel]-Energy_arr[i,FirstUzel])
          / (Trajects.Traject[i].tau*NumUzelsInLayer)
          *epsilon/d_small;

        if j>=LastUzel then
          begin
            inc(NL);
            FirstUzel:=(NL-1)*NumUzelsInLayer;
            LastUzel:=NL*NumUzelsInLayer; // -1;
          end;
        end;
      end;
    end;

if ChBoxWithhE.Checked=True then
  begin
    if (LEhE.Font.Color=clRed) then exit;
    hE:=StrToFloat(LEhE.Text);
    for i:=NomTrjOt to NomTrjDo do begin
      for j:=NomUzlaOt to NomUzlaDo do begin
        EnPerAng_arr[i,j]:=Round(EnPerAng_arr[i,j]/hE)*hE;
      end;
    end;
  end;

  SetLength(DecEPerAng_arr, Trajects.Traject[0].NodeCount);
  for j:=NomUzlaOt to NomUzlaDo do begin
    DecEPerAng_arr[j]:=0;
    for i:=NomTrjOt to NomTrjDo do begin
      if ((EnPerAng_arr[i,j]>=Emin) and (EnPerAng_arr[i,j]<Emax))
        then DecEPerAng_arr[j]:=DecEPerAng_arr[j]+1;
    end;
  end;

  if ChBoxNormRandom.Checked=True
  then RandomDec:=StrToFloat(LERandom.Text)
  else RandomDec:=1;

  AveDec:=0;

  for j:=NomUzlaOt to NomUzlaDo do
    begin
      if ChBoxDecELG1Ang.Checked=True then

```

```

begin
    Chart1.BottomAxis.Title.Caption:='г л у б и н а,   А н г.';
    Series4.AddXY(Trajects.Traject[0].t_arr[j]*d_small/epsilon,
        DecEPerAng_arr[j]/NormNo/RandomDec, '', clBlack);
    AveDec:=AveDec+DecEPerAng_arr[j]/NormNo/RandomDec;
end
else
begin
    Chart1.BottomAxis.Title.Caption:='г л у б и н а,   т а у';
    Series4.AddXY(Trajects.Traject[0].t_arr[j],
        DecEPerAng_arr[j]/NormNo/RandomDec, '', clBlack);
    AveDec:=AveDec+DecEPerAng_arr[j]/NormNo/RandomDec;
end;
end;

AveDec:=AveDec/(NomUzlaDo-NomUzlaOt+1);
LabelAveDec.Caption:='Среднее = '+FloatToStr(AveDec);

SD:=0;
for j:=NomUzlaOt to NomUzlaDo do SD:=SD+DecEPerAng_arr[j];
LabelSDDec.Caption:='Сдек = '+IntToStr(SD)+' ед.';

Finalize(DecEPerAng_arr);
Finalize(EnPerAng_arr);

Chart1.Title.Text.Text:='Деканалирование по dE/dL';
FormRaspredE.Caption:='Деканалирование по dE/dL';

RepaintGraphEnergy;
end;

procedure TFormRaspredE.SaveGrOut;
var
    Emin, Emax: double;
    i, j: integer;
    DecEPerAng_arr: array of integer;
    EnPerAng_arr: En_arr_Type;
    NumUzelsInLayer, FirstUzel, LastUzel, NL: integer;
    SaveF: TextFile;

begin
    Emin:=StrToFloat(FormRaspredE.LECalcEot.Text);
    Emax:=StrToFloat(FormRaspredE.LECalcEdo.Text);
    NumUzelsInLayer:=1000; //StrToInt(FormRaspredE.LELayerEn.Text);

    SetLength(EnPerAng_arr, Trajects.CntOfTraject, Trajects.Traject[0].NodeCount);

    if (NumUzelsInLayer<=0)
        or ((Trajects.Traject[0].NodeCount mod NumUzelsInLayer)<>0) then exit;

    for i:=0 to (Trajects.CntOfTraject-1) do begin
        NL:=1;
        FirstUzel:=0;
        LastUzel:=NumUzelsInLayer-1;
        for j:=0 to (Trajects.Traject[i].NodeCount-1) do begin
            EnPerAng_arr[i,j]:=(Energy_arr[i,LastUzel]-Energy_arr[i,FirstUzel])
                / (Trajects.Traject[i].tau*NumUzelsInLayer)
                *epsilon/d_small;

            if j>=LastUzel then
                begin
                    inc(NL);
                    FirstUzel:=(NL-1)*NumUzelsInLayer;
                    LastUzel:=NL*NumUzelsInLayer-1;
                end;
        end;
    end;

    SetLength(DecEPerAng_arr, Trajects.Traject[0].NodeCount);
    for j:=0 to (Trajects.Traject[0].NodeCount-1) do begin
        DecEPerAng_arr[j]:=0;
        for i:=0 to (Trajects.CntOfTraject-1) do begin

```



```

end
else if Trajects.Traject[0].NodeCount=1000 then begin
  WriteLn(SaveF, Round(Trajects.Traject[0].t_arr[0]), ';', DecEPerAng_arr[499]);
  WriteLn(SaveF, Round(Trajects.Traject[0].t_arr[999]), ';', DecEPerAng_arr[499]);
end;

CloseFile(SaveF);

Finalize(DecEPerAng_arr);
Finalize(EnPerAng_arr);
end;

procedure TFormRaspredE.LECalcEotChange(Sender: TObject);
begin
  RedErrors(FormRaspredE.LECalcEot, '-');
end;

procedure TFormRaspredE.LECalcEdoChange(Sender: TObject);
begin
  RedErrors(FormRaspredE.LECalcEdo, '-');
end;

procedure TFormRaspredE.LELayerEnChange(Sender: TObject);
begin
  RedErrors(FormRaspredE.LELayerEn, 'Integer');
end;

procedure TFormRaspredE.SBExcelClick(Sender: TObject);
var
  i, n: integer;
  ExcelApplication: Variant;
begin
  ExcelApplication:=CreateOleObject ('Excel.Application');
  ExcelApplication.Visible:=True;
  ExcelApplication.WorkBooks.Add;

  if ChBoxAllExcel.Checked=True then
  begin
    for i:=1 to Series4.Count do
    begin
      ExcelApplication.Cells[i,1].Value:=Series4.XValue[i-1];
      ExcelApplication.Cells[i,2].Value:=Series4.YValue[i-1];
    end;
  end
  else
  begin
    i:=1; n:=1;
    ExcelApplication.Cells[n,1].Value:=Series4.XValue[i-1];
    ExcelApplication.Cells[n,2].Value:=Series4.YValue[i-1];
    i:=i+StrToInt(LELayerEn.Text);
    n:=n+1;
    While i<=Series4.Count do
    begin
      ExcelApplication.Cells[n,1].Value:=Series4.XValue[i-1];
      ExcelApplication.Cells[n,2].Value:=Series4.YValue[i-2];
      n:=n+1;

      ExcelApplication.Cells[n,1].Value:=Series4.XValue[i-1];
      ExcelApplication.Cells[n,2].Value:=Series4.YValue[i-1];

      i:=i+StrToInt(LELayerEn.Text);
      n:=n+1;
    end;
    ExcelApplication.Cells[n,1].Value:=Series4.XValue[Series4.Count-1];
    ExcelApplication.Cells[n,2].Value:=Series4.YValue[Series4.Count-1];
  end;
end;

//Распределение по глубине

```

```

procedure TFormRaspredE.RaspeGLBtnClick(Sender: TObject);
var
    Emin, Emax, y: double;
    i, ti: integer;
    ntE: array of integer;
begin
    Chart1.FreeAllSeries;
    Chart1.BottomAxis.Title.Caption:='tay';
    Chart1.LeftAxis.Title.Caption:='N (количество частиц), шт';

    if (LEEminGl.Font.Color=clRed)
    or (LEEmaxGl.Font.Color=clRed)
    then exit;

    Emin:=StrToFloat(LEEminGl.Text);
    Emax:=StrToFloat(LEEmaxGl.Text);

    GetTrjUzel;
    if GetTrjUzelOk=False then exit;

    SetLength(ntE,Trajects.Traject[0].NodeCount);

    For i:=NomTrjOt to NomTrjDo do begin // Расчёт распределения:
        for ti:=NomUzlaOt to NomUzlaDo do // Проверяются все частицы
            begin // по всей глубине,
                y:=Energy_arr[i,ti]; // если частица № i имеет нужную энергию
                if (y<=Emax) and (y>Emin) // (от Emin до Emax) на данной глубине (ti),
                    then ntE[ti]:=ntE[ti]+1; // то к значению массива ntE[ti]
                end; // прибавляется +1
            end;

    {Построение графика}
    Series3:=Series.TLineSeries.Create(Chart1);
    Chart1.AddSeries(Series3);
    For ti:=NomUzlaOt to NomUzlaDo do
        Series3.AddXY(Trajects.Traject[0].t_arr[ti],ntE[ti],',',clBlack);

    Finalize(ntE); {Удаление динамического массива из памяти}

    Chart1.Title.Text.Text:='Распределение частиц с энергиями от '#13
        +FloatToStr(Emin)+' до '+FloatToStr(Emax)+' эВ по глубине';
    FormRaspredE.Caption:='Распределение частиц с энергиями от '
        +FloatToStr(Emin)+' до '+FloatToStr(Emax)+' эВ по глубине';
    RepaintGraphEnergy;
end;

procedure TFormRaspredE.CalcGLBtnClick(Sender: TObject);
label L1;
var
    nE, i, ti, tiot, tido: integer;
    EEmax, EEmin, Energy: double;
begin
    if (LEEminGl.Font.Color=clRed)
    or (LEEmaxGl.Font.Color=clRed)
    or (LECalcGlot.Font.Color=clRed)
    or (LECalcGldo.Font.Color=clRed)
    then exit;

    EEmin:=StrToFloat(LEEminGl.Text);
    EEmax:=StrToFloat(LEEmaxGl.Text);

    if StrToFloat(LECalcGlot.Text)<=dep_Start
    then LECalcGlot.Text:=FloatToStrF(dep_Start,ffGeneral,10,10);
    if StrToFloat(LECalcGldo.Text)>=(dep_Start+glubina)
    then LECalcGldo.Text:=FloatToStrF(dep_Start+glubina,ffGeneral,10,10);

    tiot:=Round((StrToFloat(LECalcGlot.Text)-dep_Start)/(Trajects.Traject[0].tau));
    tido:=Round((StrToFloat(LECalcGldo.Text)-dep_Start)/(Trajects.Traject[0].tau));
    Label9.Caption:='Узлы от '+IntToStr(tiot)+' до '+IntToStr(tido);

    nE:=0;

```

```

For i:=0 to Trajects.CntOfTraject-1 do begin
  for ti:=tiot to tido do
    begin
      Energy:=Energy_arr[i,ti];
      if (Energy<=EEmax) and (Energy>EEmin)
        then begin nE:=nE+1; goto L1; end;
      end;
    L1:
      end;
      LabelCalcGlRes.Caption:=IntToStr(nE);
      CalcGlLegend.Caption:='Кол-во частиц с E от '+'
      FloatToStr(EEmin)+' до '+'FloatToStr(EEmax)+' на данной глубине (тау)';
    end;

procedure TFormRaspredE.LEEminGlChange(Sender: TObject);
begin
  RedErrors(FormRaspredE.LEEminGl, '-');
end;

procedure TFormRaspredE.LEEmaxGlChange(Sender: TObject);
begin
  RedErrors(FormRaspredE.LEEmaxGl, '-');
end;

procedure TFormRaspredE.LECalcGlChange(Sender: TObject);
begin
  RedErrors(FormRaspredE.LECalcGl, '-');
end;

procedure TFormRaspredE.LECalcGldoChange(Sender: TObject);
begin
  RedErrors(FormRaspredE.LECalcGldo, '-');
end;

//Распределение по энергиям

procedure TFormRaspredE.SpectraE;
var
  i, j, NomUzla: integer;
  Emin, Emax, Ej, CenaKanala, Razreshenie: double;
  nE: array of integer;
  EnForSpectra_arr: En_arr_Type;
begin
  Chart1.FreeAllSeries;
  Chart1.LeftAxis.Title.Caption:='N (количество частиц), шт.';

  if (LECenaKanala.Font.Color=clRed)
    or (LERazreshSpectra.Font.Color=clRed)
    or (LENomUzlaSpectraE.Font.Color=clRed)
    then exit;

  CenaKanala:=StrToFloat(LECenaKanala.Text);
  Razreshenie:=StrToFloat(LERazreshSpectra.Text);
  NomUzla:=StrToInt(LENomUzlaSpectraE.Text);

  GetTrjUzel;
  if GetTrjUzelOk=False then exit;

  SetLength(EnForSpectra_arr, Trajects.CntOfTraject, Trajects.Traject[0].NodeCount);

  if ChBeVAngSpectraE.Checked=False then // Спектр в эВ
    begin
      Chart1.BottomAxis.Title.Caption:='E - Eo (потерянная энергия), эВ';
      EnForSpectra_arr:=Energy_arr;
    end
  else // Спектр в эВ/Анг
    begin
      Chart1.BottomAxis.Title.Caption:='E - Eo (потерянная энергия), эВ/Анг.';
      for i:=NomTrjOt to NomTrjDo do begin
        for j:=0 to Trajects.Traject[0].NodeCount-1 do begin
          EnForSpectra_arr[i,j]:=Energy_arr[i,j]/(Trajects.Traject[i].tau*(j+1)

```

```

+Trajects.Traject[i].t_arr[0])*epsilon/d_small;

    end;
  end;
end;

{Определение Emin и Emax на заданной глубине}
Emin:=EnForSpectra_arr[0,NomUzla];
Emax:=EnForSpectra_arr[0,NomUzla];
for i:=NomTrjOt to NomTrjDo do begin
  if EnForSpectra_arr[i,NomUzla]<Emin then Emin:=EnForSpectra_arr[i,NomUzla];
  if EnForSpectra_arr[i,NomUzla]>Emax then Emax:=EnForSpectra_arr[i,NomUzla];
end;
Emin:=Emin-Razreshenie;
Emax:=Emax+Razreshenie;

SetLength(nE, round((Emax-Emin)/CenaKanala)+4);

Ej:=Emin; {Средняя энергия на отрезке № j}
For j:=1 to round((Emax-Emin)/CenaKanala)+1 do begin
  nE[j]:=0; {Кол-во частиц с энергией от Ej-Razreshenie/2 до Ej+Razreshenie/2}
  for i:=NomTrjOt to NomTrjDo do
    begin
      if ((EnForSpectra_arr[i,NomUzla]<=Ej+Razreshenie/2)
        and (EnForSpectra_arr[i,NomUzla]>Ej-Razreshenie/2))
        then nE[j]:=nE[j]+1;
    end;
  Ej:=Ej+CenaKanala;
end;

Series4:=Series.TLineSeries.Create(Char1);
Series4.Stairs:=False;
Series4.Pointer.Visible:=True;
Series4.Pointer.HorizSize:=2;
Series4.Pointer.VertSize:=2;
Series4.Pointer.Pen.Visible:=True;
Series4.Pointer.Style:=psCircle;
Chart1.AddSeries(Series4);

for j:=1 to round((Emax-Emin)/CenaKanala)+1 do
  begin
    Ej:=Emin+CenaKanala*(j-1);
    Series4.AddXY(Ej,nE[j],',',clBlack);
  end;

Finalize(nE);

Chart1.Title.Text.Text:='Распределение частиц по потерям энергии';
FormRaspredE.Caption:='Распределение частиц по потерям энергии';

RepaintGraphEnergy;
end;

procedure TFormRaspredE.RaspeBtnClick(Sender: TObject);
begin
  SpectraE;
end;

procedure TFormRaspredE.LEStepSpectraEChange(Sender: TObject);
var
  UpDownStep, code: integer;
  Text: String;
begin
  Text:=FormRaspredE.LEStepSpectraE.Text;
  Val(Text,UpDownStep,code);
  if code=0
    then FormRaspredE.UpDownSpectraE.Increment:=UpDownStep;
end;

procedure TFormRaspredE.LENomUzlaSpectraEChange(Sender: TObject);
var
  NomUzla, Code: integer;

```

```

S: String;
begin
  LENomUzlaSpectraE.Font.Color:=clBlack;
  S:=LENomUzlaSpectraE.Text;
  Val(S, NomUzla, Code);
  if (code<>0) or (NomUzla<0) or (NomUzla>=Trajects.Traject[0].NodeCount) then
  begin
    LENomUzlaSpectraE.Font.Color:=clRed;
    LabelTauSpectraE.Caption:='??? tay';
    LabelAngSpectraE.Caption:='??? Анг';
    exit;
  end;
  LabelTauSpectraE.Caption:=
    FloatToStr(NomUzla*Trajects.Traject[0].tau+dep_Start)+' tay';
  LabelAngSpectraE.Caption:=FloatToStr
    ((NomUzla*Trajects.Traject[0].tau+dep_Start)*d_small/epsilon)+' Анг';
end;

procedure TFormRaspredE.UpDownSpectraEClick(Sender: TObject;
  Button: TUDBtnType);
begin
  SpectraE;
end;

procedure TFormRaspredE.ChBeVAngSpectraEClick(Sender: TObject);
begin
  if ChBeVAngSpectraE.Checked=True then
  begin
    Label12.Caption:='эВ/Анг';
    Label19.Caption:='эВ/Анг';
  end
  else
  begin
    Label12.Caption:=' эВ';
    Label19.Caption:=' эВ';
  end;
end;

procedure TFormRaspredE.SpeedButton2Click(Sender: TObject);
var
  i: integer;
  ExcelApplication: Variant;
begin
  ExcelApplication:=CreateOleObject ('Excel.Application');
  ExcelApplication.Visible:=True;
  ExcelApplication.WorkBooks.Add;

  for i:=1 to Series4.Count do
  begin
    ExcelApplication.Cells[i,1].Value:=Series4.XValue[i-1];
    ExcelApplication.Cells[i,2].Value:=Series4.YValue[i-1];
  end;
end;

procedure TFormRaspredE.LERazreshSpectraChange(Sender: TObject);
begin
  RedErrors(FormRaspredE.LERazreshSpectra, 'No <=0');
end;

procedure TFormRaspredE.LECenaKanalaChange(Sender: TObject);
begin
  RedErrors(FormRaspredE.LECenaKanala, 'No <=0');
end;

//Общие кнопки на форме FormRaspredE

procedure TFormRaspredE.Zapolnenie; //процедура заполняет данными форму
begin
  if Settings[45]='ChBoxRoe07=False'
  then ChBoxRoe07.Checked:=False
  else ChBoxRoe07.Checked:=True;

```



```

if Settings[46]='ChBoxGammaDeltha=False'
then ChBoxGammaDeltha.Checked:=False
else ChBoxGammaDeltha.Checked:=True;
ChBoxGammaDelthaClick(FormRaspredE);
if Settings[49]='ChBAngstrem=False'
then ChBAngstrem.Checked:=False
else ChBAngstrem.Checked:=True;
if Settings[50]='ChBeVAngstrem=False'
then ChBeVAngstrem.Checked:=False
else ChBeVAngstrem.Checked:=True;
LEhE.Text:=Settings[51];
LECalcEot.Text:=Settings[52];
LECalcEdo.Text:=Settings[53];
if Settings[54]='ChBGisteVAng=False'
then ChBGisteVAng.Checked:=False
else ChBGisteVAng.Checked:=True;
LEEminGl.Text:=Settings[55];
LEEmaxGl.Text:=Settings[56];
LECalcGlot.Text:=Settings[57];
LECalcGlDo.Text:=Settings[58];
LERazreshSpectra.Text:=Settings[59];
LEStepUpDownGistE.Text:=Settings[65];
if Settings[81]='ChBoxNormGistEL=False'
then ChBoxNormGistEL.Checked:=False
else ChBoxNormGistEL.Checked:=True;
if Settings[82]='ChBoxDecELGLAng=False'
then ChBoxDecELGLAng.Checked:=False
else ChBoxDecELGLAng.Checked:=True;
if Settings[89]='ChBoxWithhE=True'
then ChBoxWithhE.Checked:=True
else ChBoxWithhE.Checked:=False;
LECenaKanala.Text:=Settings[92];
LEStepSpectraE.Text:=Settings[93];
end;

procedure TFormRaspredE.FormCreate(Sender: TObject);
begin
Zapolnenie;

FormRaspredE.ChBoxRoe07.Hint:=
' '+#13+'OK = Учёт правила равнораспределения'+#13+' '+#13+
' ( ro_e(x) + roe07 )'+#13+
' ro_e --> -----'+#13+
' 2 ' ;

FormRaspredE.ChBoxGammaDeltha.Hint:=
' '+#13+' OK = Учёт Gamma и Deltha'+#13+' '+#13+
' / 2 mec2 beta^2 gamma^2 \ deltha'+#13+
' Le = ln ----- - beta^2 - -----'+#13+
' \ 11.5 Z2 / 2';

FormRaspredE.ChBLayerEn.Hint:=
' '+#13+'OK = Пересчёт энергии по слоям:'+#13+' '+#13+
' Слои по глубине разбиваются с заданным кол-вом. узлов в слое'+#13+
' Находится разность потерь энергии dE(эВ) в начале и конце каждого'+#13+
' слоя; Эта разность dE(эВ) делится на длину слоя в Ангстремах.'+#13+'';

end;

procedure TFormRaspredE.CloseFormBtnClick(Sender: TObject);
begin
FormRaspredE.Close;
end;

procedure TFormRaspredE.RepaintGraphEnergy;
begin
Chart1.Repaint;
LEGraphYmax.Text:=FloatToStrF(Chart1.LeftAxis.Maximum,ffGeneral,9,9);
LEGraphYmin.Text:=FloatToStrF(Chart1.LeftAxis.Minimum,ffGeneral,9,9);
LEGraphXmax.Text:=FloatToStrF(Chart1.BottomAxis.Maximum,ffGeneral,9,9);
LEGraphXmin.Text:=FloatToStrF(Chart1.BottomAxis.Minimum,ffGeneral,9,9);

```

```

end;

procedure TFormRaspredE.SBChangeGraphClick(Sender: TObject);
begin
if ChBoxGraphYAuto.Checked=True then
    Chart1.LeftAxis.Automatic:=True
else
    begin
        Chart1.LeftAxis.Automatic:=False;
        try
            Chart1.LeftAxis.Minimum:=StrToFloat(LEGraphYmin.Text);
            Chart1.LeftAxis.Maximum:=StrToFloat(LEGraphYmax.Text);
        except
            exit;
        end;
    end;

if ChBoxGraphXAuto.Checked=True then
    Chart1.BottomAxis.Automatic:=True
else
    begin
        Chart1.BottomAxis.Automatic:=False;
        try
            Chart1.BottomAxis.Minimum:=StrToFloat(LEGraphXmin.Text);
            Chart1.BottomAxis.Maximum:=StrToFloat(LEGraphXmax.Text);
        except
            exit;
        end;
    end;
SBChangeGraph.Enabled:=False;
RepaintGraphEnergy;
end;

procedure TFormRaspredE.LEGraphYminChange(Sender: TObject);
begin
    if ChBoxGraphYAuto.Checked=False
    then SBChangeGraph.Enabled:=True;
end;

procedure TFormRaspredE.LEGraphXminChange(Sender: TObject);
begin
    if ChBoxGraphXAuto.Checked=False
    then SBChangeGraph.Enabled:=True;
end;

procedure TFormRaspredE.ChBoxGraphYAutoClick(Sender: TObject);
begin
    SBChangeGraph.Enabled:=True;
end;

procedure TFormRaspredE.LabelForAllTrjClick(Sender: TObject);
begin
    LENomTrjOt.Text:='0';
    LENomTrjDo.Text:=IntToStr(Trajects.CntOfTraject-1);
    LENormNo.Text:=IntToStr(Trajects.CntOfTraject);
end;

procedure TFormRaspredE.LabelForAllUzelsClick(Sender: TObject);
begin
    LENomUzlaOt.Text:='0';
    LENomUzlaDo.Text:=IntToStr(Trajects.Traject[0].NodeCount-1);
end;

procedure TFormRaspredE.LENomTrjOtChange(Sender: TObject);
begin
    RedErrors(FormRaspredE.LENomTrjOt, 'Integer');
end;

procedure TFormRaspredE.LENomTrjDoChange(Sender: TObject);
begin
    RedErrors(FormRaspredE.LENomTrjDo, 'Integer');
end;

```

```

end;

procedure TFormRaspredE.LENomUzlaOtChange(Sender: TObject);
begin
    RedErrors(FormRaspredE.LENomUzlaOt, 'Integer');
end;

procedure TFormRaspredE.LENomUzlaDoChange(Sender: TObject);
begin
    RedErrors(FormRaspredE.LENomUzlaDo, 'Integer');
end;

procedure TFormRaspredE.LENormNoChange(Sender: TObject);
begin
    RedErrors(FormRaspredE.LENormNo, 'Integer');
end;

procedure TFormRaspredE.LabelOrZavClick(Sender: TObject);
begin
    FormOrientZav.Visible:=True;
    FormOrientZav.Show;
    FormOrientZav.Gaugel.Progress:=0;
    FormOrientZav.Series1.Clear;
    FormOrientZav.ListBox1.Items.Clear;
    FormOrientZav.LENumTrPerBun.Text:=IntToStr(ntr);
    FormOrientZav.LEOrientZavNomUzla.Text:=
        IntToStr(Trajects.Traject[0].NodeCount-1);
    if Trajects.Traject[0].NodeCount<=32767
    then FormOrientZav.UpDownOrientZav.Max:=(Trajects.Traject[0].NodeCount-1)
    else FormOrientZav.UpDownOrientZav.Enabled:=False;
    FormOrientZav.ChBoxOriEL.Checked:=True;
    if FormRaspredE.ChBLayerEn.Checked=True
    then FormOrientZav.ChBoxDecLast.Checked:=False
    else FormOrientZav.ChBoxDecLast.Checked:=True;
end;

procedure TFormRaspredE.GetTrjUzel;
begin
    GetTrjUzelOk:=False;

    if (LENomTrjOt.Font.Color=clRed)
    or (LENomTrjDo.Font.Color=clRed)
    or (LENomUzlaOt.Font.Color=clRed)
    or (LENomUzlaDo.Font.Color=clRed)
    then exit;

    NomTrjOt:=StrToInt(LENomTrjOt.Text);
    if NomTrjOt<0 then NomTrjOt:=0;
    if NomTrjOt>(Trajects.CntOfTraject-1) then NomTrjOt:=Trajects.CntOfTraject-1;
    LENomTrjOt.Text:=IntToStr(NomTrjOt);

    NomTrjDo:=StrToInt(LENomTrjDo.Text);
    if NomTrjDo<NomTrjDo then NomTrjDo:=NomTrjOt;
    if NomTrjDo>(Trajects.CntOfTraject-1) then NomTrjDo:=Trajects.CntOfTraject-1;
    LENomTrjDo.Text:=IntToStr(NomTrjDo);

    NomUzlaOt:=StrToInt(LENomUzlaOt.Text);
    if NomUzlaOt<0 then NomUzlaOt:=0;
    if NomUzlaOt>(Trajects.Traject[0].NodeCount-1) then
NomUzlaOt:=Trajects.Traject[0].NodeCount-1;
    LENomUzlaOt.Text:=IntToStr(NomUzlaOt);

    NomUzlaDo:=StrToInt(LENomUzlaDo.Text);
    if NomUzlaDo<NomUzlaOt then NomUzlaDo:=NomUzlaDo;
    if NomUzlaDo>(Trajects.Traject[0].NodeCount-1) then
NomUzlaDo:=Trajects.Traject[0].NodeCount-1;
    LENomUzlaDo.Text:=IntToStr(NomUzlaDo);

    GetTrjUzelOk:=True;
end;

```

```

procedure TFormRaspredE.ShowGridClick(Sender: TObject);
begin
    if ShowGrid.Checked=False then
        begin
            Chart1.LeftAxis.Grid.Visible:=True;
            Chart1.BottomAxis.Grid.Visible:=True;
            ShowGrid.Checked:=True;
        end
    else
        begin
            Chart1.LeftAxis.Grid.Visible:=False;
            Chart1.BottomAxis.Grid.Visible:=False;
            ShowGrid.Checked:=False;
        end;
end;

procedure TFormRaspredE.ShowTextClick(Sender: TObject);
begin
    if ShowText.Checked=False then
        begin
            Chart1.Foot.Visible:=True;
            Chart1.Title.Visible:=True;
            ShowText.Checked:=True;
        end
    else
        begin
            Chart1.Foot.Visible:=False;
            Chart1.Title.Visible:=False;
            ShowText.Checked:=False;
        end;
end;

procedure TFormRaspredE.SaveGraphClick(Sender: TObject);
begin
    if Form1.SavePictureDialog1.Execute then
        begin
            Chart1.SaveToMetafileEnh(Form1.SavePictureDialog1.FileName);
        end;
end;

procedure TFormRaspredE.PrintGraphClick(Sender: TObject);
begin
    if Form1.PrintDialog1.Execute then
        Chart1.Print;
end;

procedure TFormRaspredE.ToClipBrdClick(Sender: TObject);
begin
    Chart1.CopyToClipboardMetafile(true);
end;

procedure TFormRaspredE.ExcelClick(Sender: TObject);
var
    i: integer;
    ExcelApplication: Variant;
begin
    ExcelApplication:=CreateOleObject ('Excel.Application');
    ExcelApplication.Visible:=True;
    ExcelApplication.WorkBooks.Add;

    for i:=1 to Series1.Count do
        begin
            ExcelApplication.Cells[i,1].Value:=Series1.XValue[i-1];
            ExcelApplication.Cells[i,2].Value:=Series1.YValue[i-1];
        end;
end;
end.

```

```

unit Dechanneling;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Buttons, StdCtrls, Gauges, TeEngine, Series, ExtCtrls, TeeProcs,
  Chart, ExtDlgs, Trajectories, Constants, Menus, Math;

type
  TFormDechanneling = class(TForm)
    Chart1: TChart;
    SolveDecXBtn: TBitBtn;
    Series1: TLineSeries;
    Label3: TLabel;
    CBLeaveDec: TCheckBox;
    Label7: TLabel;
    OrientZavBtn: TBitBtn;
    CBAngstrem: TCheckBox;
    Gauge1: TGauge;
    CloseFormBtn: TSpeedButton;
    LENomTrjOt: TLabeledEdit;
    LENomTrjDo: TLabeledEdit;
    ChBoxNormDec: TCheckBox;
    Bevel1: TBevel;
    PopupMenuChartDec: TPopupMenu;
    ShowGrid: TMenuItem;
    ShowText: TMenuItem;
    N3: TMenuItem;
    SaveGraph: TMenuItem;
    PrintGraph: TMenuItem;
    ToClipBrd: TMenuItem;
    BitBtn1: TBitBtn;
    ToExcel: TMenuItem;
    procedure SolveDecXBtnClick(Sender: TObject);
    procedure OrientZavBtnClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure CloseFormBtnClick(Sender: TObject);
    procedure LENomTrjOtChange(Sender: TObject);
    procedure LENomTrjDoChange(Sender: TObject);
    procedure ShowGridClick(Sender: TObject);
    procedure ShowTextClick(Sender: TObject);
    procedure SaveGraphClick(Sender: TObject);
    procedure PrintGraphClick(Sender: TObject);
    procedure ToClipBrdClick(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure ToExcelClick(Sender: TObject);
  private
    { Private declarations }
    procedure GraphTextDechanneling;
  public
    { Public declarations }
    procedure Zapolnenie;
  end;

var
  FormDechanneling: TFormDechanneling;

implementation

uses OrientZav, ComObj;

{$R *.dfm}

procedure TFormDechanneling.SolveDecXBtnClick(Sender: TObject);
// Деканалирование по глубине
Label L1;
var
  i, ti, nvibiv, imin, imax: integer;
  x: double;
  Dec_Vs: integer;

```

```

Dec: array of double;
Dec1: array of integer;
S: String;
begin
Series1.Clear;

if (LENomTrjOt.Font.Color=clRed)
or (LENomTrjDo.Font.Color=clRed)
then exit;
imin:=StrToInt(LENomTrjOt.Text);
imax:=StrToInt(LENomTrjDo.Text);
if imin<0 then imin:=0;
if imax<0 then imax:=0;
if imax>(Trajects.CntOfTraject-1) then imax:=(Trajects.CntOfTraject-1);
if imin>(Trajects.CntOfTraject-1) then imin:=(Trajects.CntOfTraject-1);
if imax<imin then imax:=imin;
LENomTrjOt.Text:=IntToStr(imin);
LENomTrjDo.Text:=IntToStr(imax);

if imin=imax then
FormDechanneling.Caption:='Деканализирование по глубине - одна частица № '
+IntToStr(imin)
+' (Xнач='+FloatToStrF(Trajects.Traject[imin].x_arr[0],ffGeneral,3,3)
+' Vнач='+FloatToStrF(Trajects.Traject[imin].v_arr[0],ffGeneral,3,3)+')'

else if (imin=0) and (imax=Trajects.CntOfTraject-1) then
FormDechanneling.Caption:='Деканализирование - все частицы'

else
FormDechanneling.Caption:='Деканализирование - '+IntToStr(imax-imin+1)
+' частиц с номерами от '+IntToStr(imin)+' до '+IntToStr(imax);

FormDechanneling.Gauge1.MaxValue:=(imax-imin+1);
SetLength(Dec,Trajects.Traject[0].NodeCount);
{Задаётся длина динамического массива ДЕКАНАЛИРОВАНИЯ:
длина массива := кол-ву узлов}
nvibiv:=0; {кол-во выбывших частиц}
Dec_Vs:=0; {кол-во отсчётов площади деканализирования}

{Проверка условия попадания траектории каждой частицы в
интервал деканализирования:}
For i:=imin to imax do begin
SetLength(Dec1,Trajects.Traject[i].NodeCount+1);
{Dec1 - массив деканализирования данной частицы № i}
Dec1[0]:=0;
for ti:=1 to Trajects.Traject[i].NodeCount do begin
x:=Trajects.Traject[i].x_arr[ti-1];
x:=x-Floor(x); {Определяем дробную часть значения x (от 0 до 1)}
if ( ( Ploskost='(100)') and
( (x<=( sigma_dec))
or (x>( 1-sigma_dec))
or ( (x<=(0.25+sigma_dec)) and (x>(0.25-sigma_dec)) )
or ( (x<=(0.5 +sigma_dec)) and (x>(0.5 -sigma_dec)) )
or ( (x<=(0.75+sigma_dec)) and (x>(0.75-sigma_dec)) ) ) ) )

or ( (Ploskost='(110)') and
( (x<=(sigma_dec))
or (x>(1-sigma_dec))
or ( (x<=(0.5+sigma_dec)) and (x>(0.5-sigma_dec)) ) ) )

or ( (Ploskost='(111)') and
( (x<=(-0.0044+sigma_dec))
or (x>(1-0.0044-sigma_dec))
or ( (x<=(0.7556+sigma_dec)) and (x>(0.7556 -sigma_dec)) ) ) ) )
then
begin
Dec1[ti]:=1;
Dec[ti-1]:=Dec[ti-1]+1;
Dec_Vs:=Dec_Vs+1;
end;
if (Dec1[ti]<Dec1[ti-1]) and (CBLeaveDec.Checked=True)

```

```

        {Частица прошла интервал деканализирования - при этом она выбывает и
        далее её не надо учитывать}
    then
    begin
        nvibiv:=nvibiv+1;
        goto L1; {Переход к метке L1 - выход из цикла проверки условия
        деканализирования для данной частицы}
    end;
end;
L1:
    Gauge1.Progress:=i+1;
    Finalize(Dec1);
end;

{Вывод сообщения о количестве деканализировавших частиц}
S:=IntToStr(nvibiv);
Label3.Caption:='По всей глубине выбывают '+S+' частиц';

{Нормировка на количество частиц}
if ChBoxNormDec.Checked=True then
begin
    Chart1.LeftAxis.Title.Caption:='Ндек/No [ No= '+IntToStr(imax-imin+1)+' шт ]';
    for ti:=0 to Trajects.Traject[0].NodeCount-1
    do Dec[ti]:=Dec[ti]/(imax-imin+1);
    end
else
begin
    Chart1.LeftAxis.Title.Caption:='Ндек, шт [Кол-во частиц с X = Xдек]';
end;

{Построение графика деканализирования}
if CBAngstrem.Checked=False then
begin
    Chart1.Title.Text.Text:='Функция деканализирования по глубине';
    Chart1.BottomAxis.Title.Caption:='т а y';
    for ti:=0 to Trajects.Traject[0].NodeCount-1
    do Series1.AddXY(Trajects.Traject[0].t_arr[ti],Dec[ti],',',clBlack);
    end
else // if CBAngstrem.Checked=True then
begin
    Chart1.Title.Text.Text:='Функция деканализирования по глубине';
    Chart1.BottomAxis.Title.Caption:='г л у б и н а, А н г.';
    for ti:=0 to Trajects.Traject[0].NodeCount-1
    do Series1.AddXY(Trajects.Traject[0].t_arr[ti]
        *d_small/epsilon,Dec[ti],',',clBlack);
    end;
Finalize(Dec);
Label7.Caption:='Площадь деканализирования = '+IntToStr(Dec_Vs)+' ед.';
GraphTextDechanneling; // Подпись графика
end; {конец процедуры}

procedure TFormDechanneling.LENomTrjOtChange(Sender: TObject);
begin
    RedErrors(FormDechanneling.LENomTrjOt, 'Integer');
end;

procedure TFormDechanneling.LENomTrjDoChange(Sender: TObject);
begin
    RedErrors(FormDechanneling.LENomTrjDo, 'Integer');
end;

procedure TFormDechanneling.OrientZavBtnClick(Sender: TObject);
begin
    FormOrientZav.Visible:=True;
    FormOrientZav.Show;
    FormOrientZav.Gauge1.Progress:=0;
    FormOrientZav.Series1.Clear;
    FormOrientZav.ListBox1.Items.Clear;
    FormOrientZav.LENumTrPerBun.Text:=IntToStr(ntr);
    FormOrientZav.LEOrientZavNomUzla.Text:=
        IntToStr(Trajects.Traject[0].NodeCount-1);

```

```

if Trajects.Traject[0].NodeCount<=32767
  then FormOrientZav.UpDownOrientZav.Max:=(Trajects.Traject[0].NodeCount-1)
  else FormOrientZav.UpDownOrientZav.Enabled:=False;
end;

procedure TFormDechanneling.Zapolnenie; //процедура заполняет данными форму
begin
  if Settings[60]='CBAngstrem=False' then CBAngstrem.Checked:=False
  else CBAngstrem.Checked:=True;
  if Settings[61]='CBLeaveDec=True' then CBLeaveDec.Checked:=True
  else CBLeaveDec.Checked:=False;
  if Settings[86]='ChBoxNormDec=False' then ChBoxNormDec.Checked:=False
  else ChBoxNormDec.Checked:=True;
end;

procedure TFormDechanneling.FormCreate(Sender: TObject);
begin
  Zapolnenie;
end;

procedure TFormDechanneling.CloseFormBtnClick(Sender: TObject);
begin
  FormDechanneling.Close;
end;

procedure TFormDechanneling.GraphTextDechanneling;
begin
  Chart1.Foot.Text.Text:=
    FloatToStr(Trajects.CntOfTraject)+' тп. X='
    +FloatToStrF(Trajects.Traject[0].x_arr[0],ffGeneral,3,3)+'...'
    +FloatToStrF(Trajects.Traject[Trajects.CntOfTraject-1].x_arr[0],ffGeneral,3,3)
    +' V='+FloatToStrF(Trajects.Traject[0].v_arr[0],ffGeneral,3,3)+'...'
    +FloatToStrF(Trajects.Traject[Trajects.CntOfTraject-1].v_arr[0],ffGeneral,3,3)
    +' Zl='+FloatToStr(Zl)+' M='+FloatToStr(MassOfIon/m0c2)+'aem T='
    +FloatToStr(EnergyT/1000000)+'МэВ'#13
    +' yрол='+FloatToStr(DeltaTheta*180/Pi)+'град'+ ' k='+FloatToStr(k)
    +' l='+FloatToStr(l)+' ' +Ploskost+' Curv='+FloatToStr(kt)+'*t+'
    +FloatToStr(Curv*Sqr(epsilon)/d_small)+'*d/eps^2'#13
    +FormDechanneling.Caption;
end;

procedure TFormDechanneling.ShowGridClick(Sender: TObject);
begin
  if ShowGrid.Checked=False then
    begin
      Chart1.LeftAxis.Grid.Visible:=True; Chart1.BottomAxis.Grid.Visible:=True;
      ShowGrid.Checked:=True;
    end
  else
    begin
      Chart1.LeftAxis.Grid.Visible:=False; Chart1.BottomAxis.Grid.Visible:=False;
      ShowGrid.Checked:=False;
    end;
end;

procedure TFormDechanneling.ShowTextClick(Sender: TObject);
begin
  if ShowText.Checked=False then
    begin
      Chart1.Foot.Visible:=True; ShowText.Checked:=True;
    end
  else
    begin
      Chart1.Foot.Visible:=False;
      ShowText.Checked:=False;
    end;
end;

procedure TFormDechanneling.SaveGraphClick(Sender: TObject);
begin
  if Form1.SavePictureDialog1.Execute then

```



```

    Chart1.SaveToMetafileEnh(Form1.SavePictureDialog1.FileName);
end;

procedure TFormDechanneling.PrintGraphClick(Sender: TObject);
begin
    if Form1.PrintDialog1.Execute then
        Chart1.Print;
end;

procedure TFormDechanneling.ToClipBrdClick(Sender: TObject);
begin
    Chart1.CopyToClipboardMetafile(true);
end;

procedure TFormDechanneling.BitBtn1Click(Sender: TObject);
var
    i, ti, imin, imax: integer;
    x, v: double;
    Dec: array of double;
begin
    Series1.Clear;
    imin:=0;
    imax:=(Trajects.CntOfTraject-1);
    Gauge1.MaxValue:=Trajects.Traject[0].NodeCount-1;
    SetLength(Dec, Trajects.Traject[0].NodeCount);
    for ti:=0 to Trajects.Traject[0].NodeCount-1 do begin
        Dec[ti]:=0;
        for i:=imin to imax do begin
            v:=Trajects.Traject[i].v_arr[ti];
            x:=Trajects.Traject[i].x_arr[ti];
            if 2*Sqr(v)*epsilon*d_small*Diffuz(x)+U(x)>=Vmax
                then Dec[ti]:=Dec[ti]+1;
            Gauge1.Progress:=ti;
        end;
    end;
    {Построение графика деканализирования}
    if CBAngstrem.Checked=False then
        begin
            Chart1.Title.Text.Text:='Функция деканализирования по глубине';
            Chart1.BottomAxis.Title.Caption:='т а у';
            for ti:=0 to Trajects.Traject[0].NodeCount-1
                do Series1.AddXY(Trajects.Traject[0].t_arr[ti], Dec[ti], '', clBlack);
        end
    else // if CBAngstrem.Checked=True then
        begin
            Chart1.Title.Text.Text:='Функция деканализирования по глубине';
            Chart1.BottomAxis.Title.Caption:='г л у б и н а, А н г.';
            for ti:=0 to Trajects.Traject[0].NodeCount-1
                do Series1.AddXY(Trajects.Traject[0].t_arr[ti]
                    *d_small/epsilon, Dec[ti], '', clBlack);
        end;
    Finalize(Dec);
end; {конец процедуры}

procedure TFormDechanneling.ToExcelClick(Sender: TObject);
var
    i: integer;
    ExcelApplication: Variant;
begin
    ExcelApplication:=CreateOleObject ('Excel.Application');
    ExcelApplication.Visible:=True; ExcelApplication.WorkBooks.Add;
    for i:=1 to Series1.Count do
        begin
            ExcelApplication.Cells[i,1].Value:=Series1.XValue[i-1];
            ExcelApplication.Cells[i,2].Value:=Series1.YValue[i-1];
        end;
end;

end.

```

```

unit OrientZav;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart, Buttons,
  Constants, Trajectories, ExtDlgs, ComCtrls, Gauges, Menus, Math;

type
  TFormOrientZav = class(TForm)
    Chart1: TChart;
    Series1: TLineSeries;
    ListBox1: TListBox;
    LEnumTrPerBun: TLabelEdit;
    SolveOrientZavBtn: TBitBtn;
    LabelOrientZavGlubina: TLabel;
    LOrientZavNomUzla: TLabelEdit;
    UpDownOrientZav: TUpDown;
    LabelOrientZavGlubinaTau: TLabel;
    LabelOrientZavGlubinaAngstrem: TLabel;
    Gauge1: TGauge;
    CloseFormBtn: TSpeedButton;
    LStepUpDownOrientZav: TLabelEdit;
    ChBoxOriEL: TCheckBox;
    ChBoxDecLast: TCheckBox;
    PopupMenuChartOrZav: TPopupMenu;
    ShowGrid: TMenuItem;
    ShowText: TMenuItem;
    N3: TMenuItem;
    SaveGraph: TMenuItem;
    PrintGraph: TMenuItem;
    ToClipBrd: TMenuItem;
    procedure SolveOrientZavBtnClick(Sender: TObject);
    procedure UpDownOrientZavClick(Sender: TObject; Button: TUDBtnType);
    procedure LOrientZavNomUzlaChange(Sender: TObject);
    procedure CloseFormBtnClick(Sender: TObject);
    procedure LStepUpDownOrientZavChange(Sender: TObject);
    procedure ShowGridClick(Sender: TObject);
    procedure ShowTextClick(Sender: TObject);
    procedure SaveGraphClick(Sender: TObject);
    procedure PrintGraphClick(Sender: TObject);
    procedure ToClipBrdClick(Sender: TObject);
    procedure FormDechannelingExcel1Click(Sender: TObject);
  private
    { Private declarations }
    procedure DecOrientZav;
  public
    { Public declarations }
  end;

var
  FormOrientZav: TFormOrientZav;

implementation

uses Dechanneling, ComObj, RaspredE;

{$R *.dfm}

procedure TFormOrientZav.DecOrientZav;
{Эта процедура рассчитывает ориентационную зависимость на глубине,
соответствующей номеру узла № NomUzla}
var
  i, ti, kk, kmax, code,
  NumBunch,           // NumBunch - количество пучков частиц
  NumTrPerBun,        // NumTrPerBun - количество частиц(траекторий) в одном пучке
  CountBunch,         // CountBunch - номер текущего(рассчитываемого) пучка
  NomUzla: integer; // NomUzla - номер узла решения (соответствует глубине)
  x, xmin, xmax, xn: double;
  SD: array of integer; // SD - массив значений площади деканализирования по пучкам

```

```

vsr: array of double; // vsr -среднее значение начальной скорости в одном пучке
Emin, Emax: double; // Границы по энергии (dE/dL)
hE: double; // Разрешение по энергии
EnPerAng_arr: En_arr_Type; // Массив значений dE/dL
NumUzelsInLayer, FirstUzel, LastUzel, NL: integer;
RandomDec: double;
S: String;

begin
    Series1.Clear;
    ListBox1.Items.Clear;

    if LEOrientZavNomUzla.Font.Color=clRed then exit;

    s:=LENumTrPerBun.Text;
    Val(s, NumTrPerBun, code);
    if code<>0 then
        begin
            S:='100';
            S:=InputBox('Заданы нечисловые данные',
                'Введите кол-во частиц в пучке (Пример: 100):', S);
            LENumTrPerBun.Text:=S;
            Val(s, NumTrPerBun, code);
        end;

    NomUzla:=StrToInt(LEOrientZavNomUzla.Text);
    Gauge1.MaxValue:=Trajects.CntOfTraject;
    NumBunch:=(Trajects.CntOfTraject) div (NumTrPerBun);
    SetLength(SD, NumBunch);
    SetLength(vsr, NumBunch);
    RandomDec:=1;

    if ChBoxOriEL.Checked=False then
        begin // Деканалирование по глубине

            {Проверка условия попадания траектории каждой частицы в
            интервал деканалирования: }
            For CountBunch:=0 to NumBunch-1 do begin
                SD[CountBunch]:=0;

                if ChBoxDecLast.Checked=False then
                    begin
                        For i:=CountBunch*NumTrPerBun to (CountBunch+1)*NumTrPerBun-1 do begin
                            for ti:=0 to NomUzla do begin
                                x:=Trajects.Traject[i].x_arr[ti];
                                x:=x-Floor(x); {Определяем дробную часть значения x (от 0 до 1)}
                                if ( ( Ploskost='(100)') and
                                    (
                                        (x<=( sigma_dec))
                                        or
                                        (x>( 1-sigma_dec))
                                        or ( (x<=(0.25+sigma_dec)) and (x>(0.25-sigma_dec)) )
                                        or ( (x<=(0.5 +sigma_dec)) and (x>(0.5 -sigma_dec)) )
                                        or ( (x<=(0.75+sigma_dec)) and (x>(0.75-sigma_dec)) ) ) ) )

                                    or ( ( Ploskost='(110)') and
                                        (
                                            (x<=(sigma_dec))
                                            or
                                            (x>(1-sigma_dec))
                                            or ( (x<=(0.5+sigma_dec)) and (x>(0.5-sigma_dec)) ) ) ) )

                                    or ( ( Ploskost='(111)') and
                                        (
                                            (x<=(-0.0044+sigma_dec))
                                            or
                                            (x>(1-0.0044-sigma_dec))
                                            or ( (x<=(0.7556+sigma_dec)) and (x>(0.7556 -sigma_dec)) ) ) ) )
                                    then SD[CountBunch]:=SD[CountBunch]+1;
                            end;
                            Gauge1.Progress:=i+1;
                        end;
                    end
                else
                    begin
                        ti:=NomUzla;
                        For i:=CountBunch*NumTrPerBun to (CountBunch+1)*NumTrPerBun-1 do begin

```

```

x:=Trajects.Traject[i].x_arr[ti];
x:=x-Floor(x); {Определяем дробную часть значения x (от 0 до 1)}
if ( ( Ploskost='(100)') and
    (
        (x<=(sigma_dec))
        or
        (x>(1-sigma_dec))
        or ( (x<=(0.25+sigma_dec)) and (x>(0.25-sigma_dec)) )
        or ( (x<=(0.5 +sigma_dec)) and (x>(0.5 -sigma_dec)) )
        or ( (x<=(0.75+sigma_dec)) and (x>(0.75-sigma_dec)) ) ) )

or ( (Ploskost='(110)') and
    (
        (x<=(sigma_dec))
        or
        (x>(1-sigma_dec))
        or ( (x<=(0.5+sigma_dec)) and (x>(0.5-sigma_dec)) ) ) )

or ( (Ploskost='(111)') and
    (
        (x<=(-0.0044+sigma_dec))
        or
        (x>(1-0.0044-sigma_dec))
        or ( (x<=(0.7556+sigma_dec)) and (x>(0.7556 -sigma_dec)) ) ) ) )
    then SD[CountBunch]:=SD[CountBunch]+1;
    Gauge1.Progress:=i+1;
end;
end;

{Определение среднего значения начальной скорости в этом пучке (№ CounBunch):
Vср = ( Впервой частицы + Впоследней частицы ) / 2 }
vsr[CountBunch]:=(Trajects.Traject[CountBunch*NumTrPerBun].v_arr[0]
+Trajects.Traject[(CountBunch+1)*NumTrPerBun-1].v_arr[0])/2;
end;
end

else // if FormOrientZav.ChBoxOriEL.Checked=True

begin // Деканализирование по dE/dL
if (FormRaspredE.LECalcEot.Font.Color=clRed)
or (FormRaspredE.LECalcEdo.Font.Color=clRed)
then exit;
Emin:=StrToFloat(FormRaspredE.LECalcEot.Text);
Emax:=StrToFloat(FormRaspredE.LECalcEdo.Text);

SetLength(EnPerAng_arr, Trajects.CntOfTraject, Trajects.Traject[0].NodeCount);

if FormRaspredE.ChBLayerEn.Checked=False then
begin
for i:=0 to Trajects.CntOfTraject-1 do begin
for ti:=0 to Trajects.Traject[i].NodeCount-1 do begin
EnPerAng_arr[i,ti]:=Energy_arr[i,ti]/(Trajects.Traject[i].tau*(ti+1)
+Trajects.Traject[i].t_arr[0])*epsilon/d_small;
end;
end;
end
else // Способ "по слоям"
begin
if (FormRaspredE.LELayerEn.Font.Color=clRed) then exit;
NumUzelsInLayer:=StrToInt(FormRaspredE.LELayerEn.Text);

if (NumUzelsInLayer<=0)
or ((Trajects.Traject[0].NodeCount mod NumUzelsInLayer)<>0) then
if Application.MessageBox(' Заданная длина слоёв в узлах'+#13+' '+#13+
' не годятся для расчёта деканализирования,'+#13+' '+#13+
' т.к. приводит к неравному делению на слои.'+#13+' '+#13+
'При выполнении может произойти ошибка'+#13+' '+#13+
' ПРОДОЛЖИТЬ ВЫПОЛНЕНИЕ ?',
'Предупреждение',
MB_YesNo+MB_IconWarning) = mrNo then exit;

for i:=0 to Trajects.CntOfTraject-1 do begin
NL:=1;
FirstUzel:=0;
LastUzel:=NumUzelsInLayer-1;
for ti:=0 to Trajects.Traject[i].NodeCount-1 do begin
EnPerAng_arr[i,ti]:=(Energy_arr[i,LastUzel]-Energy_arr[i,FirstUzel])

```

```

        / (Trajects.Traject[i].tau*NumUzelsInLayer)
        *epsilon/d_small;
    if ti>=LastUzel then
    begin
        inc(NL);
        FirstUzel:=(NL-1)*NumUzelsInLayer;
        LastUzel:=NL*NumUzelsInLayer-1;
    end;
end;
end;
end;

if FormRaspredE.ChBoxWithhE.Checked=True then
begin
    if (FormRaspredE.LhE.Font.Color=clRed) then exit;
    hE:=StrToFloat(FormRaspredE.LhE.Text);
    for i:=0 to Trajects.CntOfTraject-1 do begin
        for ti:=0 to Trajects.Traject[i].NodeCount-1 do begin
            EnPerAng_arr[i,ti]:=Round(EnPerAng_arr[i,ti]/hE)*hE;
        end;
    end;
end;

For CountBunch:=0 to NumBunch-1 do begin
    SD[CountBunch]:=0;
    if ChBoxDecLast.Checked=False then
    begin
        For i:=CountBunch*NumTrPerBun to (CountBunch+1)*NumTrPerBun-1 do begin
            for ti:=0 to NomUzla do begin
                if ((EnPerAng_arr[i,ti]>=Emin) and (EnPerAng_arr[i,ti]<Emax))
                then SD[CountBunch]:=SD[CountBunch]+1;
            end;
            Gauge1.Progress:=i+1;
        end;
    end
    else
    begin
        ti:=NomUzla;
        For i:=CountBunch*NumTrPerBun to (CountBunch+1)*NumTrPerBun-1 do begin
            if ((EnPerAng_arr[i,ti]>=Emin) and (EnPerAng_arr[i,ti]<Emax))
            then SD[CountBunch]:=SD[CountBunch]+1;
            Gauge1.Progress:=i+1;
        end;
    end;
    vsr[CountBunch]:=(Trajects.Traject[CountBunch*NumTrPerBun].v_arr[0]
        +Trajects.Traject[(CountBunch+1)*NumTrPerBun-1].v_arr[0])/2;
end;

if FormRaspredE.ChBoxNormRandom.Checked=True then
begin
    if ChBoxDecLast.Checked=False
    then RandomDec:=StrToFloat(FormRaspredE.LERandom.Text)*NumTrPerBun*(NomUzla+1)
    else RandomDec:=StrToFloat(FormRaspredE.LERandom.Text)*NumTrPerBun;
end
else RandomDec:=1;

end;

for CountBunch:=0 to NumBunch-1 do
begin
    Series1.AddXY(vsr[CountBunch]*epsilon*180/Pi,
        SD[CountBunch]/RandomDec,'',clBlack);
    Str((vsr[CountBunch]*epsilon*180/Pi):7:3,S);
    S:=S+' | '+FloatToStr(SD[CountBunch]/RandomDec);
    ListBox1.Items.Add(S);
end;

Str((NomUzla*Trajects.Traject[0].tau+dep_Start)*d_small/epsilon:2:2,S);
Chart1.Title.Text.Text:='Ориентационная зависимость'#13
    +'на глубине '+S+' Анг';

```

```

    Finalize(SD); Finalize(vsr); Finalize(EnPerAng_arr);

end;

procedure TFormOrientZav.SolveOrientZavBtnClick(Sender: TObject);
begin
    if (Trajects.CntOfTraject<=StrToFloat(FormOrientZav.LENumTrPerBun.Text)) and
    ((Trajects.CntOfTraject) < (Trajects.Traject[0].NodeCount)) then
        if Application.MessageBox('    Расчитанные траектории'+#13+' '+#13+
            '        не годятся для ориентационной'+#13+' '+#13+
            '        зависимости.'+#13+' '+#13+
            'При выполнении может произойти ошибка'+#13+' '+#13+
            '        ПРОДОЛЖИТЬ ВЫПОЛНЕНИЕ ?',
            'Предупреждение',
            MB_YesNo+MB_IconWarning) = mrNo then exit;

    DecOrientZav;

end;

procedure TFormOrientZav.UpDownOrientZavClick(Sender: TObject;
    Button: TUDBtnType);
begin
    DecOrientZav;
end;

procedure TFormOrientZav.LEStepUpDownOrientZavChange(Sender: TObject);
var
    UpDownStep, code: integer;
    Text: String;
begin
    Text:=LEStepUpDownOrientZav.Text;    Val(Text, UpDownStep, code);
    if code=0 then UpDownOrientZav.Increment:=UpDownStep;
end;

procedure TFormOrientZav.LEOrientZavNomUzlaChange(Sender: TObject);
var
    NomUzla, Code: integer;
    S: String;
begin
    LOrientZavNomUzla.Font.Color:=clBlack;
    S:=LEOrientZavNomUzla.Text;
    Val(S, NomUzla, Code);
    if (code<>0) or (NomUzla<0) or (NomUzla>=Trajects.Traject[0].NodeCount) then
        begin
            LOrientZavNomUzla.Font.Color:=clRed;
            LabelOrientZavGlubinaTau.Caption:='??? тая';
            LabelOrientZavGlubinaAngstrem.Caption:='??? Анг';
            exit;
        end;
        LabelOrientZavGlubinaTau.Caption:=
            FloatToStr(NomUzla*Trajects.Traject[0].tau+dep_Start)+' тая';
        LabelOrientZavGlubinaAngstrem.Caption:=FloatToStr
            ((NomUzla*Trajects.Traject[0].tau+dep_Start)*d_small/epsilon)+' Анг';
end;

procedure TFormOrientZav.CloseFormBtnClick(Sender: TObject);
begin
    FormOrientZav.Close;
end;

procedure TFormOrientZav.ShowGridClick(Sender: TObject);
begin
    if ShowGrid.Checked=False then
        begin
            Chart1.LeftAxis.Grid.Visible:=True;    Chart1.BottomAxis.Grid.Visible:=True;
            ShowGrid.Checked:=True;
        end
    else
        begin
            Chart1.LeftAxis.Grid.Visible:=False;    Chart1.BottomAxis.Grid.Visible:=False;
        end
    end;

```

```

        ShowGrid.Checked:=False;
    end;
end;

procedure TFormOrientZav.ShowTextClick(Sender: TObject);
begin
    if ShowText.Checked=False then
    begin
        Chart1.Foot.Text.Text:=
        IntToStr((Trajects.CntOfTraject) div (StrToInt(LEnumTrPerBun.Text)))
        + ' x '+LEnumTrPerBun.Text+ ' tp.= '
        +FloatToStr(Trajects.CntOfTraject)+' tp. X='
        +FloatToStrF(Trajects.Traject[0].x_arr[0],ffGeneral,3,3)+'...'
        +FloatToStrF(Trajects.Traject[Trajects.CntOfTraject-1].x_arr[0],ffGeneral,3,3)
        + ' V='+FloatToStrF(Trajects.Traject[0].v_arr[0],ffGeneral,3,3)+'...'
        +FloatToStrF(Trajects.Traject[Trajects.CntOfTraject-1].v_arr[0],ffGeneral,3,3)
        + ' Zl='+FloatToStr(Zl)+' M='+FloatToStr(MassOfIon/m0c2)+'aem T='
        +FloatToStr(EnergyT/1000000)+'МэВ'#13
        + ' yрол='+FloatToStr(DeltaTheta*180/Pi)+'град'+ ' k='+FloatToStr(k)
        + ' l='+FloatToStr(l)+' '+Ploskost+' Curv='+FloatToStr(kt)+'*t+'
        +FloatToStr(Curv*Sqr(epsilon)/d_small)+'*d/eps^2';
        if ChBoxOriEL.Checked=False
        then Chart1.Foot.Text.Text:=Chart1.Foot.Text.Text
            +'деканализирование по глубине'
        else Chart1.Foot.Text.Text:=Chart1.Foot.Text.Text
            +'деканализирование по dE/dL (Emin='+FormRaspredE.LECalcEot.Text
            +' Emax='+FormRaspredE.LECalcEdo.Text+' эВ/Анр)';
        Chart1.Foot.Visible:=True; Chart1.Title.Visible:=True;
        ShowText.Checked:=True;
    end
    else
    begin
        Chart1.Foot.Visible:=False; Chart1.Title.Visible:=False;
        ShowText.Checked:=False;
    end;
end;

procedure TFormOrientZav.SaveGraphClick(Sender: TObject);
begin
    if Form1.SavePictureDialog1.Execute then
        Chart1.SaveToMetafileEnh(Form1.SavePictureDialog1.FileName);
end;

procedure TFormOrientZav.PrintGraphClick(Sender: TObject);
begin
    if Form1.PrintDialog1.Execute then
    begin
        Chart1.Legend.Visible:=True; Chart1.Print; Chart1.Legend.Visible:=False;
    end;
end;

procedure TFormOrientZav.ToClipBrdClick(Sender: TObject);
begin
    Chart1.CopyToClipboardMetafile(true);
end;

procedure TFormOrientZav.FormDechannelingExcel1Click(Sender: TObject);
var
    i: integer;
    ExcelApplication: Variant;
begin
    ExcelApplication:=CreateOleObject ('Excel.Application');
    ExcelApplication.Visible:=True; ExcelApplication.WorkBooks.Add;
    for i:=1 to Series1.Count do
    begin
        ExcelApplication.Cells[i,1].Value:=Series1.XValue[i-1];
        ExcelApplication.Cells[i,2].Value:=Series1.YValue[i-1];
    end;
end;
end.

```