```
*** Two arrays must be declared:
!*** y (4) - an array containing the coordinates x = y (1),y=y(2) and projections of velocity vector
!*** Vx = y (3),Vy=y(4).The working array work (27) with a length to be calculated using the equation
!*** 3 + 6 * neqn (neqn- is number of equations)
!*** mu_atm and m_keha variables must also be declared as REAL. Here we take into  account
!*** the Fortran feature. If variables are not declared at the beginning of the program by the REAL,
!*** INTEGER, etc. operators, the fortran compiler can declare and create them by using the  default
!*** rule: if the first character in the variable name is i, j, k, l, m, n then the variable must be INTEGER
!*** if it is not so then the variable must be REAL.
!*** real y(4),work(27),mu_atm,m_keha
!*** Declaration of additional integer array iwork(5) with fixed length 5
integer iwork(5)
!*** Description the global variables which must be used in subroutine ″func″ to calculate
!*** derivatives of coordinate x,y and velocity projections  Vx,Vy with respect to time
common g_rask,ro_atm_0,ro_keha,param1,param2
!*** Next operator must be used do declare that ″func″ is an external subroutine
!*** not the name of ordinary variable
external func
!*** Now we need to open the file to save the calculations results
open(10,file=″dim2.dat″)
!*** On the next lines some important variables must be created and we must assign
!*** values for them
!*** Value of pi
pi=3.1415926536
!*** Radius of sphere (object of investigation) in meters
g_rask=9.814
!*** Radius of sphere (object of investigation) in meters
r_keha=1.8!*** Description the global variables which must be used in subroutine ″func″ to calculate
!*** derivatives of coordinate x,y and velocities projections  Vx,Vy with respect to time
!*** Sphere square of cross section
s_keha=pi*r_keha**2
!*** Aerodynamic parameter for sphere (taken from the internet)
c_keha=0.47
!*** volume of the sphere
v_keha=4.*pi*r_keha**3/3
!*** Mass of the sphere in kilograms
m_keha=5.
!*** Density of sphere
ro_keha=m_keha/v_keha
!*** Pressure of atmosphere  on the sea level in Pascale
p_atm=100000.
!*** Temperature of atmosphere on the sea level in Kelvin
t_atm=293.
!*** Mass of one kilomole of air  (average value) in kg/kmol
```

```
!*** Description the global variables which must be used in subroutine "func" to calculate
!*** derivatives of coordinate x,y and velocities projections  Vx,Vy with respect to timemu_atm=29.
```

!*** Universal gas constant in $\frac{J}{K \cdot kmol}$

```
r_gaas=8314.
!*** Calculation of the density of air on the sea level
ro_atm_0=p_atm*mu_atm/(r_gaas*t_atm)
!*** Two additional work variables used for convenience only
param1=g_rask*mu_atm/(r_gaas*t_atm)
param2=0.5*c_keha*s_keha/m_keha
!*** Number of differential equations to be integrated
neqn=4
!*** Total number of time steps
nt=400
!*** Time step in seconds
dt=0.1
!*** Initial value for time
t=0.
!*** Relative and absolute errors for calculation the coordinate and velocity
relerr=1.e-7
abserr=1.e-7
!*** Initial conditions
!*** y(1)=0 – x coordinate of sphere at start time moment
!*** y(2)=0 – y coordinate of sphere at start time moment
!*** y(3)=0 – velocity (x-projection) of sphere at start time moment
!*** y(4)=0 - velocity (y-projection)  of sphere at start time moment
y(1)=0.
y(2)=0.
y(3)=0.!*** virst derivative for coordinate y
Y(4)=0.
!*** iflag=1 is need to start the calculation, it means  that we are starting a new simulation
iflag=1

!*** Cycle operator to perform the integration of the system of differential equations and
!*** calculation of coordinate and velocity at different time moments with timestep dt=0.1 seconds.
!*** The total time of simulation can be calculated as a product of variables  nt and dt variables,
!*** in our  case it is equal to nt*dt=0.1*400=40 seconds.
do it=1,nt
!*** Caslculation of new value of time (should be done by hand)
tout=t+dt
!*** Calculation of coordinates y(1),y(2) and velocities y(3),y(4) at the next time moment tout=t+dt
call rkf45(func,neqn,y,t,tout,relerr,abserr,iflag,work,iwork)
!*** just in case the control of the iflag value.  If iflag=2 the calculation was success and
```

```
!*** we can can continue
if (iflag.ne.2) then
iflag=2
endif
!*** if-operator to control he position of sphere. If coordinate (height of trajectory)is less
!*** than zero we have to stop the calculations. In our case the height and x coordinate of
!*** ball are coincided.
if(y(2).le.0.) then
stop
endif
!*** Now we need to save the calculated results at the next format of data
!*** time   x  y   Vx   Vy
!*** 5 real numbers on each line and total number of lines is equal to value of variable “nt”
write (10,*)tout,y
!*** End of cycle operator
enddo
!*** stop program
stop
!*** End of source code for main program
end

!*** The most important part of the program. Here we must implement the differential equation.
!*** Now we need to calculate the derivatives with respect to the coordinate and velocity
!*** with respect to time.
!*** Now the subroutine ″func″ must be created. Number of input parameters is fixed.
!*** t-time, y(4)-array consist the coordinate y(1),y(2) and velocity projections y(3),y(4) of the ball,
!*** yp(4)-array with derivatives so that:
!*** yp(1)=dy(1)/dt=velocity x-projection=y(3)
!*** yp(2)=dy(2)/dt=velocity y-projection=y(4)
!*** yp(3)=dy(2)/dt=acceleration x-projection=-param2*roh*v*y(3)
!*** yp(4)=dy(2)/dt=acceleration y-projection=g_rask*(roh/ro_keha-1.)-param2*roh*v*y(4)
!*** (theoretical background you can find in concpects of lecture)
subroutine func(t,y,yp)
!*** Declaration of arrays
real y(4),yp(4)
!*** Globalization of some variables (same as in the main program)
common g_rask,ro_atm_0,ro_keha,param1,param2
!*** virst derivative for coordinate x
yp(1)=y(3)
!*** virst derivative for coordinate y
yp(2)=y(4)
!*** Calculation of the density of air on height y(2)
roh=ro_atm_0*exp(-param1*y(2))
!*** Absolute value of velocity
```

```
v=sqrt(y(3)**2+y(4)**2)
!*** Second derivative, calculation of x-projection of acceleration with the second Newton's low
yp(3)=-param2*roh*v*y(3)
!*** Second derivative, calculation of y-projection of acceleration with the second Newton's low
yp(4)=g_rask*(roh/ro_keha-1.)-param2*roh*v*y(4)
!*** Back to "call" operator in the main program
return
!*** End of source code for subroutine "func"
end
```