*** 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)
!*** m_maa variable 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, m_maa
```

!*** Declaration of additional integer work array iwork(5) with fixed length 5 (defined in subroutine rkf45)

```
integer iwork(5)
```

!*** 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

```
common gm,r_maa,ro_atm_0,param1,param2
```

!*** external operator should be used to describe the variable “func” as a name of external subroutine

```
external func
```

!*** Now we need to open the file to save the calculated results

```
open(20,file=″fall.dat″)
```

!*** On the next lines some important variables should be created and we must assign
!*** values for them
!*** Value of pi

```
pi=3.1415926536
```

!*** Aerodynamic parameter for landing module (taken from the internet)

```
c_keha=0.47
```

!*** Radius of landing module (object of investigation) in meters

```
r_keha=2.
```

!*** Mass of the landing module in kilograms

```
m_keha=2800.
```

!*** 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!*** Pressure of atmolanding module on the sea level in Pascale

```
p_atm=101000.
```

!*** Mass of one kilomole of air (average value) in kg/kmol

```
mu_atm=29.
```

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

```
r_gaas=8314.
```

!*** Temperature of atmosphere on the sea level in Kelvin

```
t_atm=300.
```

!*** Gravity constant

```
G=6.67408e-11
```

!*** Mass of he Earth in kilograms

```
m_maa=5.9722e+24
!*** Additional work parameter  (the number 1.e-9 arose due to conversion of meters into kilometers)
gm=G*m_maa*1.e-9
!*** Radius of the Earth in kilometers
r_maa=6378.
!*** landing module square of cross section
s_keha=pi*r_keha**2
!*** Calculation of the density of air on a sea level
ro_atm_0=p_atm*mu_atm/(r_gaas*t_atm)
!*** Two additional work variables used for convenience only
param2=s_keha*c_keha*500./m_keha
param1=mu_atm*9814./(r_gaas*t_atm)
!*** Number of differential equations
neqn=4
!*** Time step in seconds
dt=1.
!*** Total number of time steps
nt=2000
!*** Relative and absolute errors for calculation the coordinate and velocity this is a input parameter for
!*** subroutine rkf45
abserr=1.e-8
relerr=1.e-8
!*** Intitial value of velocity (modulus)
v0=7.7884079
!*** Entering the angle between the initial velocity and the “y” axis
!*** (determination of the direction of the satellite’s velocity vector towards the Earth)
print ‘(“Enter alpha=”$)’
read *,alpha
!*** Initial conditions
!*** y(1)=6578 km      – x coordinate of landing module at start time moment
!*** y(2)=0 km         – y coordinate of landing module at start time moment
!*** y(3)=0. km/s      – velocity (x-projection) of landing module at start time moment
!*** y(4)=9.78337 km/s- velocity (y-projection)  of landing module at start time moment
y(1)=6578.0
y(2)=0.
y(3)=-v0*sin(pi*alpha/180.)
y(4)=v0*cos(pi*alpha/180.)
!*** Initial value of time
t=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 coordinates and velocities at different time moments with timestep dt=1 second.
```

```
!*** The total time of simulation can be calculated as a product of variables  nt and dt ,
!*** in our  case it is equal to nt*dt=1*2000=2000 seconds.
do i=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 to continue
if(iflag.ne.2) then
iflag=2
endif
!*** Calculation of the absolute value of velocity (   |v|=sqrt(v_x^2+v_y^2)   )
v=sqrt(y(3)**2+y(4)**2)
!*** Calculation of the distance between satellite and center of the Earth (   |r|=sqrt(x^2+y^2)   )
r=sqrt(y(1)**2+y(2)**2)
!*** Height of trajectory
h=r-r_maa
!*** Density of atmosphere on height h
ro_atm=ro_atm_0*exp(-param1*h)
!*** Calculation of the projections and modulus of acceleration vector
ax=-y(1)*gm/r**3-param2*ro_atm*v*y(3)
ay=-y(2)*gm/r**3-param2*ro_atm*v*y(4)
a=sqrt(ax**2+ay**2)
!*** Parachute modeling: if the height of the trajectory becomes less than 5 km,
!*** we increase the radius of the landing module by 5 times
if (h.le.5.)then
r_keha=20.
s_keha=pi*r_keha**2
param2=c_keha*s_keha*500./m_keha
endif
!*** Checking the landing on the Earth
if(h.le.0.)then
stop
endif
!***Saving data in the following format
!*** time x y Vx Vy height velocity(km/h) acceleration(m/s^2)
write(20, '(10f15.5)') tout,y,h,v*3600,a*1000
!***
enddo
!***
stop
!***
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 sphere,
!*** yp(4)-array with derivatives so that:
!*** dy(1)=dy(1)/dt=velocity=y(3) and
!*** dy(2)=dy(2)/dt=velocity=y(4) and
!*** r=sqrt(y(1)**2+y(2)**2)
!*** dy(3)=dy(2)/dt=acceleration x-projection=-y(1)*gm/r**3-param2*ro_atm*vv*y(3)
!*** dy(4)=dy(2)/dt=acceleration y-projection=-y(2)*gm/r**3-param2*ro_atm*vv*y(4)
!*** (theoretical background you can find in lecture)
subroutine func(t,y,dy)
!*** Declaration of arrays
real y(4),dy(4)
!*** 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
common gm,r_maa,ro_atm_0,param1,param2
!***
r=sqrt(y(1)**2+y(2)**2)
!***
vv=sqrt(y(3)**2+y(4)**2)
!***
h=r-r_maa
ro_atm=ro_atm_0*exp(-param1*h)
!*** first derivative for coordinate x with respect to time
dy(1)=y(3)
!*** first derivative for coordinate y with respect to time
dy(2)=y(4)
!*** Second derivative, calculation of x-projection of acceleration with the second Newton′s low
dy(3)=-y(1)*gm/r**3-param2*ro_atm*vv*y(3)
!*** Second derivative, calculation of y-projection of acceleration with the second Newton′s low
dy(4)=-y(2)*gm/r**3-param2*ro_atm*vv*y(4)
!*** Back to call operator in the main program
return
!*** End of source code for subroutine “func”
end
```