```
*** 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),m_maa,m_kuu
!*** 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,gk,rmk,omegak,phi0
!*** 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(10,file=″kepler.dat″)
!*** On the next lines some important variables should be created and we must assign
!*** values for them
!*** Value of pi
pi=3.1415926536
!*** Gravity constant
G=6.67408e-11
!*** Mass of he Earth in kilograms
m_maa=5.972e+24
!*** Mass of he Moon in kilograms
m_kuu=7.34767309e+22
!*** Radius of the Earth in kilometers
r_maa=6378.0
!*** Radius of the Moon in kilometers
r_kuu=1738.1
!*** Additional work parameter  (the number 1.e-9 arose due to conversion of meters into kilometers)
gm=G*m_maa*1.e-9
!*** Additional work parameter  (the number 1.e-9 arose due to conversion of meters into kilometers)
gk=G*m_kuu*1.e-9
!*** Distance between Earth and Moon n kilometers
rmk=384399.
!*** Orbital speed of The Moon in km/s
vk=1.022
!*** angle velocity (nurkkiirus) of the Moon
omegak=vk/rmk
```

```
!*** Number of differential equations
neqn=4
!*** Time step in seconds
dt=100.
!*** Total number of time steps
nt=5000
!*** Relative and absolute errors for calculation the coordinate and velocity this is a input parameter for
!*** subroutine rkf45
abserr=1.e-9
relerr=1.e-9
!*** Initial conditions for Moon module
!***
!*** Intitial x-coordinate of module in km
y(1)=-6471.0
!*** Intitial y-coordinate of module in km
y(2)=0.
!*** Initial x-projection of velocity
y(3)=0.
!***  Initial y-projection of velocity
y(4)=-11.01
!***  The next parameter is a angle in radians which give the initial position of the Moon
!phi0=-60.*pi/180.
!phi0=-52.7145*pi/180.
phi0=-52.435*pi/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 prod
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
!*** The next two operators used to calculate the position of Moon. We need to calculate two
!*** coordinate of the Moon x and y. We assume that Moon is moving on circle around the Earth with
velocity 1.022km/s
```

```
xk=rmk*cos(omegak*tout+phi0)
yk=rmk*sin(omegak*tout+phi0)
!***  Calculation of the distance from Moonmodule to  surface of the Earth
rr=sqrt(y(1)**2+y(2)**2)-r_maa
!*** Calculation the velocity of module
vv=sqrt(y(3)**2+y(4)**2)
!*** Calculation the distance between the modue and Moon
rkk=sqrt((y(1)-xk)**2+(y(2)-yk)**2)
!*** Data saving for visualization
write(10, '(10f15.4)')tout,y,xk,yk,rr,rkk-r_kuu,vv
!*** Check the position of the module relative to the moon. Did the module fall on the moon or not?
if (rkk-r_kuu.lt.0.) then
print *,"crash"
stop
endif
!***
enddo
stop
!*** end of 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 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/rr**3-gk*(y(1)-xk)/r**3
!*** dy(4)=dy(2)/dt=acceleration y-projection=-y(2)*gm/rr**3-gk*(y(2)-yk)/r**3
!*** (theoretical background you can find inlecture)
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,gk,rmk,omegak,phi0
!*** The next two operators used to calculate the position of Moon. We need to calculate two
!*** coordinate of the Moon x and y. We assume that Moon is moving on circle around the Earth with
velocity 1.022km/s
xk=rmk*cos(omegak*t+phi0)
yk=rmk*sin(omegak*t+phi0)
```

```
!***  Calculation of the distance from Moonmodule to  Earth
rr=sqrt(y(1)**2+y(2)**2)
!***  Calculation of the distance from Moonmodule to  Moon
r=sqrt((y(1)-xk)**2+(y(2)-yk)**2)
!*** 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/rr**3-gk*(y(1)-xk)/r**3
*** Second derivative, calculation of y-projection of acceleration with the second Newton's low
dy(4)=-y(2)*gm/rr**3-gk*(y(2)-yk)/r**3
!*** Back to call operator in the main program
return
!*** End of source code for subroutine "func"
end
```