*** Two arrays must be declared:
!*** y (6) - an array containing the coordinates x = y (1), y=y(2),z=y(3) and projections of velocity vector
!*** Vx = y (4),Vy=y(5),Vz=y(6).
!*** The working array work (39) 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.
!*** here b(3) - is the magnetic induction vector, e (3) - is the electric field strength vector

```
real y(6),work(39),b(3),e(3),m
```

!*** 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 qm,b,e
```

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

```
external func
```

!*** New type for entering data by using of “namelist” operator
!*** http://jules-lsm.github.io/vn4.2/namelists/intro.html

```
namelist /input/q,m,nt,dt,abserr,relerr,b,e,y
```

!*** Opening files to reading and saving data

```
open(10,file="lorentz.in")
open(20,file="lorentz.dat")
```

!*** Reading data with namelist operator

```
read(10,input)
```

!*** Some additional parameters

```
qm=q/m
```

!*** Number of equations

```
neqn=6
```

!*** Initializing of the time and iflag

```
t=0
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
!*** The total time of simulation can be calculated as a product of variables nt and dt

```
do i=1,nt
```

!*** Calculation of new value of time (should be done by hand)

```
tout=t+dt
```

*** Calculation of coordinate y(1),y(2),y(3) and velocitie y(4),y(5),y(6) 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
```

```
!***Saving data in the following format
!*** time  x y z Vx Vy Vz
write(20,*) tout,y
enddo
!*** stop program
stop
!*** end of source code
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(6)-array consist the coordinate y(1),y(2),y(3) and velocity projections  y(4),y(5),y(6) of sphere,
!*** yp(6)-array with derivatives so that:
!*** dy(1)=dy(1)/dt=velocity_x=y(4)
!*** dy(2)=dy(2)/dt=velocity_y=y(5)
!*** dy(3)=dy(3)/dt=velocity_z=y(6)
!*** dy(4)=qm*(y(5)*b(3)-y(6)*b(2)+e(1))
!*** dy(5)=qm*(y(6)*b(1)-y(4)*b(3)+e(2))
!*** dy(6)=qm*(y(4)*b(2)-y(5)*b(1)+e(3))
!*** (theoretical background you can find in precis of lecture)
subroutine func(t,y,dy)
!*** Declaration of arrays
real y(6),dy(6),b(3),e(3)
!*** Description the global variables which must be used in subroutine ″func″ to calculate
!*** derivatives of coordinate x and velocities projections  Vx with respect to time
common qm,b,e
! *** Calculation of first derivative for coordinates  with respect to time
dy(1)=y(4)
dy(2)=y(5)
dy(3)=y(6)
! *** Calculation of first derivative for velocities  with respect to time
dy(4)=qm*(y(5)*b(3)-y(6)*b(2)+e(1))
dy(5)=qm*(y(6)*b(1)-y(4)*b(3)+e(2))
dy(6)=qm*(y(4)*b(2)-y(5)*b(1)+e(3))
!*** Back to call operator in the main program
return
!*** End of source code for subroutine “func”
end
```