

```

1 module user_defined_types
2   use, intrinsic :: iso_c_binding
3   use fdps_vector
4   use fdps_super_particle
5   implicit none
6
7   type, public, bind(c) :: full_particle ! $fdps FP,EPI,EPJ,Force
8     ! $fdps copyFromForce full_particle (pot,pot) (acc,acc)
9     ! $fdps copyFromFP full_particle (id,id) (mass,mass) (eps,eps) (pos,pos)
10    ! $fdps clear id=keep, mass=keep, eps=keep, pos=keep, vel=keep
11    integer(kind=c_long_long) :: id
12    real(kind=c_double) mass ! $fdps charge
13    real(kind=c_double) :: eps
14    type(fdps_f64vec) :: pos ! $fdps position
15    type(fdps_f64vec) :: vel ! $fdps velocity
16    real(kind=c_double) :: pot
17    type(fdps_f64vec) :: acc
18  end type full_particle
19
20  contains
21
22  subroutine calc_gravity_pp(ep_i,n_ip,ep_j,n_jp,f) bind(c)
23    integer(c_int), intent(in), value :: n_ip,n_jp
24    type(full_particle), dimension(n_ip), intent(in) :: ep_i
25    type(full_particle), dimension(n_jp), intent(in) :: ep_j
26    type(full_particle), dimension(n_ip), intent(inout) :: f
27    integer(c_int) :: i,j
28    real(c_double) :: eps2,poti,r3_inv,r_inv
29    type(fdps_f64vec) :: xi,ai,rij
30
31    do i=1,n_ip
32      eps2 = ep_i(i)%eps * ep_i(i)%eps
33      xi = ep_i(i)%pos
34      ai = 0.0d0
35      poti = 0.0d0
36      do j=1,n_jp
37        rij%x = xi%x - ep_j(j)%pos%x
38        rij%y = xi%y - ep_j(j)%pos%y
39        rij%z = xi%z - ep_j(j)%pos%z
40        r3_inv = rij%x*rij%x &
41          + rij%y*rij%y &
42          + rij%z*rij%z &
43          + eps2
44        r_inv = 1.0d0/sqrt(r3_inv)
45        r3_inv = r_inv * r_inv
46        r_inv = r_inv * ep_j(j)%mass
47        r3_inv = r3_inv * r_inv
48        ai%x = ai%x - r3_inv * rij%x
49        ai%y = ai%y - r3_inv * rij%y
50        ai%z = ai%z - r3_inv * rij%z
51        poti = poti - r_inv
52      end do
53      f(i)%pot = f(i)%pot + poti
54      f(i)%acc = f(i)%acc + ai
55    end do
56
57  end subroutine calc_gravity_pp
58
59  subroutine calc_gravity_psp(ep_i,n_ip,ep_j,n_jp,f) bind(c)
60    integer(c_int), intent(in), value :: n_ip,n_jp
61    type(full_particle), dimension(n_ip), intent(in) :: ep_i
62    type(fdps_spj_monopole), dimension(n_jp), intent(in) :: ep_j
63    type(full_particle), dimension(n_ip), intent(inout) :: f
64    integer(c_int) :: i,j
65    real(c_double) :: eps2,poti,r3_inv,r_inv
66    type(fdps_f64vec) :: xi,ai,rij
67
68    do i=1,n_ip
69      eps2 = ep_i(i)%eps * ep_i(i)%eps
70      xi = ep_i(i)%pos
71      ai = 0.0d0
72      poti = 0.0d0
73      do j=1,n_jp
74        rij%x = xi%x - ep_j(j)%pos%x
75        rij%y = xi%y - ep_j(j)%pos%y
76        rij%z = xi%z - ep_j(j)%pos%z
77        r3_inv = rij%x*rij%x &
78          + rij%y*rij%y &
79          + rij%z*rij%z &
80          + eps2
81        r_inv = 1.0d0/sqrt(r3_inv)
82        r3_inv = r_inv * r_inv
83        r_inv = r_inv * ep_j(j)%mass
84        r3_inv = r3_inv * r_inv
85        ai%x = ai%x - r3_inv * rij%x
86        ai%y = ai%y - r3_inv * rij%y
87        ai%z = ai%z - r3_inv * rij%z
88        poti = poti - r_inv
89      end do
90      f(i)%pot = f(i)%pot + poti
91      f(i)%acc = f(i)%acc + ai
92    end do
93
94  end subroutine calc_gravity_psp
95
96 end module user_defined_types

```