```
module user_defined_types
   use, intrinsic :: iso_c_binding
   use fdps_vector
   use fdps_super_particle
   implicit none

   type, public, bind(c) :: full_ptcl !$fdps FP,EPI,EPJ,Force
      !$fdps copyFromForce full_ptcl (pot,pot) (acc,acc)
      !$fdps copyFromFP full_ptcl (mass,mass) (eps,eps) (pos,pos)
      !$fdps clear id=keep, mass=keep, eps=keep, pos=keep, vel=keep
      integer(kind=c_long_long) :: id
      real(kind=c_double)  mass !$fdps charge
      real(kind=c_double) :: eps
      type(fdps_f64vec) :: pos !$fdps position
      type(fdps_f64vec) :: vel
      real(kind=c_double) :: pot
      type(fdps_f64vec) :: acc
   end type full_ptcl

   contains

   subroutine calc_gravity_pp(ep_i,n_ip,ep_j,n_jp,f) bind(c)
      integer(c_int), intent(in), value :: n_ip,n_jp
      type(full_ptcl), dimension(n_ip), intent(in) :: ep_i
      type(full_ptcl), dimension(n_jp), intent(in) :: ep_j
      type(full_ptcl), dimension(n_ip), intent(inout) :: f
      integer(c_int) :: i,j
      real(c_double) :: eps2,poti,r3_inv,r_inv
      type(fdps_f64vec) :: xi,ai,rij

      do i=1,n_ip
         eps2 = ep_i(i)%eps * ep_i(i)%eps
         xi = ep_i(i)%pos
         ai = 0.0d0; poti = 0.0d0
         do j=1,n_jp
            rij%x  = xi%x - ep_j(j)%pos%x
            rij%y  = xi%y - ep_j(j)%pos%y
            rij%z  = xi%z - ep_j(j)%pos%z
            r3_inv = rij%x*rij%x + rij%y*rij%y + rij%z*rij%z &
                   + eps2
            r_inv  = 1.0d0/sqrt(r3_inv)
            r3_inv = r_inv * r_inv
            r_inv  = r_inv * ep_j(j)%mass
            r3_inv = r3_inv * r_inv
            ai%x   = ai%x - r3_inv * rij%x
            ai%y   = ai%y - r3_inv * rij%y
            ai%z   = ai%z - r3_inv * rij%z
            poti   = poti - r_inv
         end do
         f(i)%pot = f(i)%pot + poti
         f(i)%acc = f(i)%acc + ai
      end do

   end subroutine calc_gravity_pp

   subroutine calc_gravity_psp(ep_i,n_ip,ep_j,n_jp,f) bind(c)
      integer(c_int), intent(in), value :: n_ip,n_jp
      type(full_ptcl), dimension(n_ip), intent(in) :: ep_i
      type(fdps_spj_monopole), dimension(n_jp), intent(in) :: ep_j
      type(full_ptcl), dimension(n_ip), intent(inout) :: f
      integer(c_int) :: i,j
      real(c_double) :: eps2,poti,r3_inv,r_inv
      type(fdps_f64vec) :: xi,ai,rij

      do i=1,n_ip
         eps2 = ep_i(i)%eps * ep_i(i)%eps
         xi = ep_i(i)%pos
         ai = 0.0d0; poti = 0.0d0
         do j=1,n_jp
            rij%x  = xi%x - ep_j(j)%pos%x
            rij%y  = xi%y - ep_j(j)%pos%y
            rij%z  = xi%z - ep_j(j)%pos%z
            r3_inv = rij%x*rij%x + rij%y*rij%y + rij%z*rij%z &
                   + eps2
            r_inv  = 1.0d0/sqrt(r3_inv)
            r3_inv = r_inv * r_inv
            r_inv  = r_inv * ep_j(j)%mass
            r3_inv = r3_inv * r_inv
            ai%x   = ai%x - r3_inv * rij%x
            ai%y   = ai%y - r3_inv * rij%y
            ai%z   = ai%z - r3_inv * rij%z
            poti   = poti - r_inv
         end do
         f(i)%pot = f(i)%pot + poti
         f(i)%acc = f(i)%acc + ai
      end do

   end subroutine calc_gravity_psp

end module user_defined_types

subroutine f_main()
   use fdps_module
   use user_defined_types
   implicit none
   double precision, parameter :: time_end=10.0d0
   double precision, parameter :: dt=1.0d0/128.0d0
   integer :: i,j,k,ierr
   integer :: psys_num,dinfo_num,tree_num
   character(len=64) :: tree_type
   double precision :: time_sys=0.0d0
   type(fdps_controller) :: fdps_ctrl
   call fdps_ctrl%PS_Initialize()
   call fdps_ctrl%create_dinfo(dinfo_num)
   call fdps_ctrl%init_dinfo(dinfo_num)
   call fdps_ctrl%create_psys(psys_num,'full_ptcl')
   call fdps_ctrl%init_psys(psys_num)
   tree_type="Long,full_ptcl,full_ptcl,full_ptcl,Monopole"
   call fdps_ctrl%create_tree(tree_num,tree_type)
   call fdps_ctrl%init_tree(tree_num,0)
   call read_IC(fdps_ctrl,psys_num)
   call calc_gravity(fdps_ctrl,psys_num,dinfo_num,tree_num)
   do
      call kick(fdps_ctrl,psys_num,0.5d0*dt)
      time_sys=time_sys+dt
      call drift(fdps_ctrl,psys_num,dt)
      call calc_gravity(fdps_ctrl,psys_num,dinfo_num,tree_num)
      call kick(fdps_ctrl,psys_num,0.5d0*dt)
      if(time_sys >= time_end) exit
   end do
   call fdps_ctrl%PS_Finalize()
end subroutine f_main

subroutine calc_gravity(fdps_ctrl,psys_num,dinfo_num,tree_num)
   use fdps_module
   use user_defined_types
   implicit none
   type(fdps_controller), intent(IN) :: fdps_ctrl
   integer, intent(IN) :: psys_num,dinfo_num,tree_num
   type(c_funptr) :: pfunc_ep_ep,pfunc_ep_sp
   call fdps_ctrl%decompose_domain_all(dinfo_num,psys_num)
   call fdps_ctrl%exchange_particle(psys_num,dinfo_num)
   pfunc_ep_ep=c_funloc(calc_gravity_pp)
   pfunc_ep_sp=c_funloc(calc_gravity_psp)
   call fdps_ctrl%calc_force_all_and_write_back(tree_num,&
                                                pfunc_ep_ep,&
                                                pfunc_ep_sp,&
                                                psys_num,&
                                                dinfo_num)
end subroutine calc_gravity

subroutine kick(fdps_ctrl,psys_num,dt)
   use fdps_vector
   use fdps_module
   use user_defined_types
   implicit none
   type(fdps_controller), intent(IN) :: fdps_ctrl
   integer,intent(IN) :: psys_num
   double precision,intent(IN) :: dt
   integer :: i,nptcl_loc
   type(full_ptcl), dimension(:), pointer :: ptcl
   nptcl_loc = fdps_ctrl%get_nptcl_loc(psys_num)
   call fdps_ctrl%get_psys_fptr(psys_num,ptcl)
   do i=1,nptcl_loc
      ptcl(i)%vel = ptcl(i)%vel + ptcl(i)%acc*dt
   end do
   nullify(ptcl)
end subroutine kick

subroutine drift(fdps_ctrl,psys_num,dt)
   use fdps_vector
   use fdps_module
   use user_defined_types
   implicit none
   type(fdps_controller), intent(IN) :: fdps_ctrl
   integer, intent(IN) :: psys_num
   double precision, intent(IN) :: dt
   integer :: i,nptcl_loc
   type(full_ptcl), dimension(:), pointer :: ptcl
   nptcl_loc = fdps_ctrl%get_nptcl_loc(psys_num)
   call fdps_ctrl%get_psys_fptr(psys_num,ptcl)
   do i=1,nptcl_loc
      ptcl(i)%pos = ptcl(i)%pos + ptcl(i)%vel*dt
   end do
   nullify(ptcl)
end subroutine drift

subroutine read_IC(fdps_ctrl,psys_num)
   use fdps_module
   use user_defined_types
   implicit none
   type(fdps_controller), intent(IN) :: fdps_ctrl
   integer, intent(IN) :: psys_num
   character(len=16), parameter :: root_dir="input_data"
   character(len=16), parameter :: file_prefix="proc"
   integer :: i,myrank,nptcl_loc
   character(len=64) :: fname,proc_num
   type(full_ptcl), dimension(:), pointer::ptcl
   myrank = fdps_ctrl%get_rank()
   write(proc_num,"(i5.5)")myrank
   fname = trim(root_dir)//"/" &
         //trim(file_prefix)//proc_num//".dat"
   open(unit=9,file=trim(fname),action='read',form='unformatted',&
        access='stream',status='old')
      read(9)nptcl_loc
      call fdps_ctrl%set_nptcl_loc(psys_num,nptcl_loc)
      call fdps_ctrl%get_psys_fptr(psys_num,ptcl)
      do i=1,nptcl_loc
      read(9)ptcl(i)%id,ptcl(i)%mass,ptcl(i)%eps, &
             ptcl(i)%pos%x,ptcl(i)%pos%y,ptcl(i)%pos%z, &
             ptcl(i)%vel%x,ptcl(i)%vel%y,ptcl(i)%vel%z
   end do
   close(unit=9)
   nullify(ptcl)
end subroutine read_IC
```