```
type point = rec(double x, y);
point p[1:n], v[1:n], f[1:n];   # position, velocity,
double m[1:n];          # force and mass for each body
double G = 6.67e-11;
initialize the positions, velocities, forces, and masses;

# calculate total force for every pair of bodies
procedure calculateForces() {
  double distance, magnitude; point direction;
  for [i = 1 to n-1, j = i+1 to n] {
    distance = sqrt( (p[i].x - p[j].x)**2 +
                     (p[i].y - p[j].y)**2 );
    magnitude = (G*m[i]*m[j]) / distance**2;
    direction = point(p[j].x-p[i].x, p[j].y-p[i].y);
    f[i].x = f[i].x + magnitude*direction.x/distance;
    f[j].x = f[j].x - magnitude*direction.x/distance;
    f[i].y = f[i].y + magnitude*direction.y/distance;
    f[j].y = f[j].y - magnitude*direction.y/distance;
  }
}

# calculate new velocity and position for each body
procedure moveBodies() {
  point deltav;   # dv = f/m * DT
  point deltap;   # dp = (v + dv/2) * DT
  for [i = 1 to n] {
    deltav = point(f[i].x/m[i] * DT, f[i].y/m[i] * DT);
    deltap = point( (v[i].x + deltav.x/2) * DT,
                    (v[i].y + deltav.y/2) * DT);
    v[i].x = v[i].x + deltav.x;
    v[i].y = v[i].y + deltav.y;
    p[i].x = p[i].x + deltap.x;
    p[i].y = p[i].y + deltap.y;
    f[i].x = f[i].y = 0.0;    # reset force vector
  }
}

# run the simulation with time steps of DT
for [time = start to finish by DT] {
  calculateForces();
  moveBodies();
}
```

Figure 11.9 Sequential program for the *n*-body problem.