Dining Philosophers Using a Monitor

```
enum threestates {thinking, hungry, eating};   // create enumerated type
threestates state [5];                         // array for 5 philosophers

class monitor {
   enum threestates {thinking, hungry, eating};
   threestates state [5];
   condition me [5];      // "condition" type for calling wait/signal

   void monitor :: pickup (int i) {
      state [i] = hungry;
      test (i);
      if (state [i] != eating)
         me[i].wait();
   }

   void monitor :: putdown (int i) {
      state [i] = thinking;
      test ((i+4)%5);
      test ((i+1)%5);
   }

   void monitor :: test (int k) {
      if (state [(k+4)%5] != eating &&
          state [(k+1)%5] != eating &&
          state [k] == hungry) {
          state [k] = eating;
          me [k].signal();
      }
   }

   monitor :: monitor () {
   // initialize the philosophers to thinking
      int i;
      for (i = 0; i < 5; i++)
         state [i] = thinking;
   }
```

Process P_i

```
repeat
    while (thinking);
    // get hungry - pick up both chopsticks
    dp.pickup (i);  // dp is global monitor
    // eat!
    while (! full)
       eat();
    // done - put down the chopsticks
    dp.putdown (i);
 until (time.eof());
```