```
/* program producerconsumer */
monitor boundedbuffer;
char buffer [N];                                        /* space for N items */
int nextin, nextout;                                      /* buffer pointers */
int count;                                      /* number of items in buffer */
cond notfull, notempty;         /* condition variables for synchronization */

void append (char x)
{
     if (count == N) cwait(notfull);      /* buffer is full; avoid overflow */
     buffer[nextin] = x;
     nextin = (nextin + 1) % N;
     count++;
     /* one more item in buffer */
     csignal(notempty);                       /* resume any waiting consumer */
}
void take (char x)
{
     if (count == 0) cwait(notempty);   /* buffer is empty; avoid underflow */
     x = buffer[nextout];
     nextout = (nextout + 1) % N;
     count--;                                   /* one fewer item in buffer */
     csignal(notfull);                        /* resume any waiting producer */
}
{                                                             /* monitor body */
     nextin = 0; nextout = 0; count = 0;          /* buffer initially empty */
}
```

```
void producer()
{
     char x;
     while (true) {
     produce(x);
     append(x);
     }
}
void consumer()
{
     char x;
     while (true) {
       take(x);
       consume(x);
     }
}
void main()
{
     parbegin (producer, consumer);
}
```

Figure 5.19 A Solution to the Bounded-Buffer Producer/Consumer Problem Using a Monitor