```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <sys/types.h>
#define OVER (-1)           /*Elemento di buffer vuoto*/
#define max 40                  /*Numero di elementi da scrivere nel buffer*/
#define BUFFER_SIZE 20  /*Dimensione del buffer */

typedef struct //Struttura condivisa
{
    int buffer[BUFFER_SIZE];    //shared memory
    pthread_mutex_t lock;       //dichiarazione del mutex
    int readpos, writepos;
    int cont;
    pthread_cond_t notempty;    //dichiarazione delle variabili di condizione
    pthread_cond_t notfull;
} prodcons;


prodcons buffer;

/* Inizializza il buffer */
void init (prodcons *b){
    pthread_mutex_init(&b->lock, NULL); /*inizializza il mutex*/

    /*inizializza le condition variable*/
    pthread_cond_init(&b->notempty, NULL);
    pthread_cond_init(&b->notfull, NULL);
    b->cont=0;
    b->readpos = 0;
    b->writepos = 0;
}

/* Inserimento elementi nel buffer*/
void inserisci (prodcons *b, int data){
    pthread_mutex_lock (&b->lock); //mutex bloccato

    /* controlla che il buffer non sia pieno:*/
    while (b->cont==BUFFER_SIZE)
        pthread_cond_wait(&b->notfull, &b->lock);

    /* scrivi data e aggiorna lo stato del buffer */
    b->buffer[b->writepos] = data;
    b->cont++;
    b->writepos++;

    if (b->writepos >= BUFFER_SIZE)
```

```
48          b->writepos = 0;
49
50      /* risveglia eventuali thread (consumatori) sospesi */
51      pthread_cond_signal (&b->notempty);
52      pthread_mutex_unlock (&b->lock);//mutex sbloccato
53  }
54
55
56  /*Estrazione elementi dal buffer*/
57  int estrai (prodcons *b){
58      int data;
59      pthread_mutex_lock(&b->lock); //mutex bloccato
60
61      while (b->cont==0) /* il buffer e` vuoto? */
62          pthread_cond_wait (&b->notempty, &b->lock);
63
64      /* Leggi l'elemento e aggiorna lo stato del buffer*/
65      data = b->buffer[b->readpos];
66      b->cont--;
67      b->readpos++;
68
69      if (b->readpos >= BUFFER_SIZE)
70          b->readpos = 0;
71
72      /* Risveglia eventuali threads (produttori) sospesi*/
73      pthread_cond_signal (&b->notfull);
74      pthread_mutex_unlock (&b->lock); //mutex sbloccato
75      return data; /*Ritorna il dato letto dal buffer*/
76  }
77
78  void *producer (void *data){ /*Thread*/
79      int n;
80      printf("sono il thread produttore\n\n");
81
82      for (n = 0; n < max; n++){ /*Scrittura degli elementi nel buffer*/
83          printf ("Thread produttore scrivo %d --->\n", n);
84          inserisci (&buffer, n); /*Chiamata alla funzione inserisci*/
85      }
86
87      inserisci (&buffer, OVER); //Buffer pieno
88      return NULL;
89  }
90
91  void *consumer (void *data){ /*Thread*/
92      int d;
93      printf("sono il thread consumatore \n\n");
94
95      while (1){ /*Ciclo infinito*/
96          d = estrai (&buffer); /*Legge gli elementi da buffer*/
```

```c

        if (d == OVER) /*Esce dal ciclo quando il buffer è vuoto*/
            break;

        printf("Thread consumatore leggo: --> %d\n", d);
    }

    return NULL;
}

int main (){
    pthread_t th_a, th_b;
    init (&buffer); /*Inizializzazione struttura*/

    /* Creazione threads: */
    pthread_create (&th_a, NULL, producer,NULL);
    pthread_create (&th_b, NULL, consumer,NULL);

    /* Attesa teminazione threads creati: */
    pthread_join (th_a, NULL);
    pthread_join (th_b, NULL);
    return 0;
}

```