[Esercizio2 (P&C).pdf](https://drive.google.com/open?id=1DZ6tybUmlv9jvlfUe2bVUmDk1C_GL_iw)

Si consideri il classico problema dei produttori e consumatori, con il buffer implementato con un array di interi di dimensione 100.

Si assuma, per semplicita’, che gli interi prodotti e consumati siano tutti strettamente positivi oppure uguali a 0, vale a dire >=0.

Si considerino le seguenti condizioni aggiuntive:

1. Gli interi pari possono occupare solo le posizioni di indice pari (0, 2, 4, ...., 98);
2. Gli interi dispari divisibili per 3 possono occupare solo le posizioni di indice dispari divisibile per 3 (3, 9, 15, 21,..., 99).
3. Gli interi dispari che non sono divisibili per 3 possono occupare solo le posizioni di indice dispari non divisibili per 3 (1, 5, 7, 11, 13, 17.., 97).

Quando un processo tenta di effettuare un’operazione al momento non consentita (per esempio produrre un intero che il buffer non puo’ al momento ospitare), il processo deve essere sospeso.

Programmare il sistema sfruttando i semafori con la semantica tradizionale

Appunti iniziali:

* Buffer di 100 interi tutti >=0

## Semafori

full=0 //

emptyP=50 //

emptyD3=17 //

emptyND3=33 //

mutexP=1 //

mutexD3=1 //

mutexND3=1 //

Si assume che tutti gli elementi dell’array siano inizializzati a -1

# Produttore

while(true) {

…

item=...;

if(item%2==0) {

wait(mutexP);

buffer[i]=item;

i=(i+2)%100;

wait(emptyP);

signal(mutexP);

}

else {

if(item%3==0) {

wait(mutexD3);

buffer[j]=item;

if(j==99) // Non si può scrivere j=(j+3)%100 perché prenderebbe anche i numeri pari divisibili per 3

j=3; // Non si può nemmeno usare un while(j%2==0) (come nel caso di dispari non divisibile per 3) perché prenderebbe pure quello non divisibili per 3.

else

j=j+6;

wait(emptyD3);

signal(mutexD3);

}

else {

wait(mutexND3);

buffer[k]=item;

while(k%2==0 && k%3==0)

j=(j+1)%100;

wait(emptyND3);

signal(mutexND3);

}

}

signal(full);

...

}

# Consumatore

while(true) {

…

wait(mutexP);

wait(mutexD3);

wait(mutexND3);

while(buffer[h]==-1) // Vado avanti finché non trovo un elemento non nullo (in questo caso, non negativo)

h++;

item=buffer[h];

buffer[h]=-1;

h=(h+1)%100; // Faccio questo nonostante ci sia il while di prima che “fa la stessa funzione” perché così facendo, al posto di andare avanti all’infinito, torno a 0 quando arriva al valore 100

if(item%2==0)

signal(emptyP);

else {

if(item%3==0)

signal(emptyD3);

else

signal(emptyND3);

}

wait(full);

signal(mutexP);

signal(mutexD3);

signal(mutexND3);

…

}