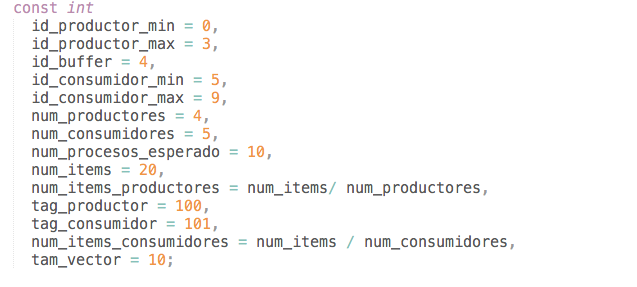
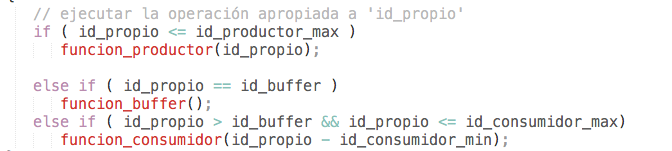
***PRODUCTOR CONSUMIDOR MULTIPLE***

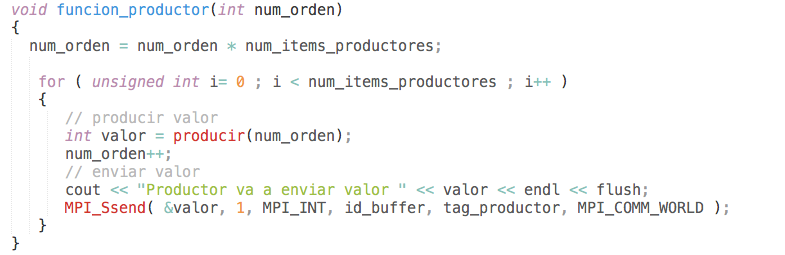
*Para el ejercicio del productor consumidor múltiple lo primero que voy a comentar en mi solución es son las constantes definidas globales.*

**

*He usado un rango de ids para los procesos productores y un rango de ids para los procesos consumidores, con lo que en el main bastará preguntar si el id\_propio esta dentro de ese rango o es exactamente 4, con lo que sería el id\_buffer. Dependiendo de que rango se trate llamaremos a una función u otra.*

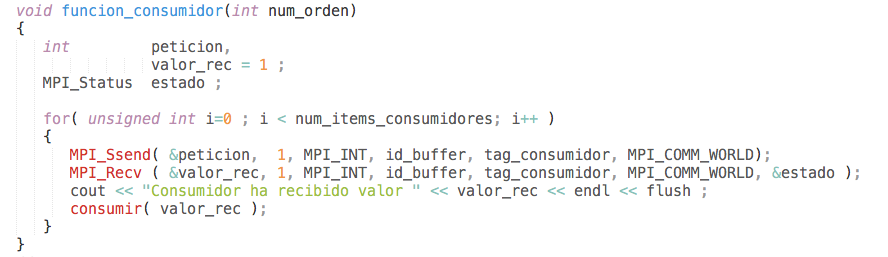
**

*Con respecto a las demás constantes definidas, observamos que tenemos dos tags, uno para productor y otro para consumidor, las cuales nos ayudaran en el proceso buffer para poder hacer la correcta operación.   
Primero, vamos a ver las funciones productor y consumidor.*

**

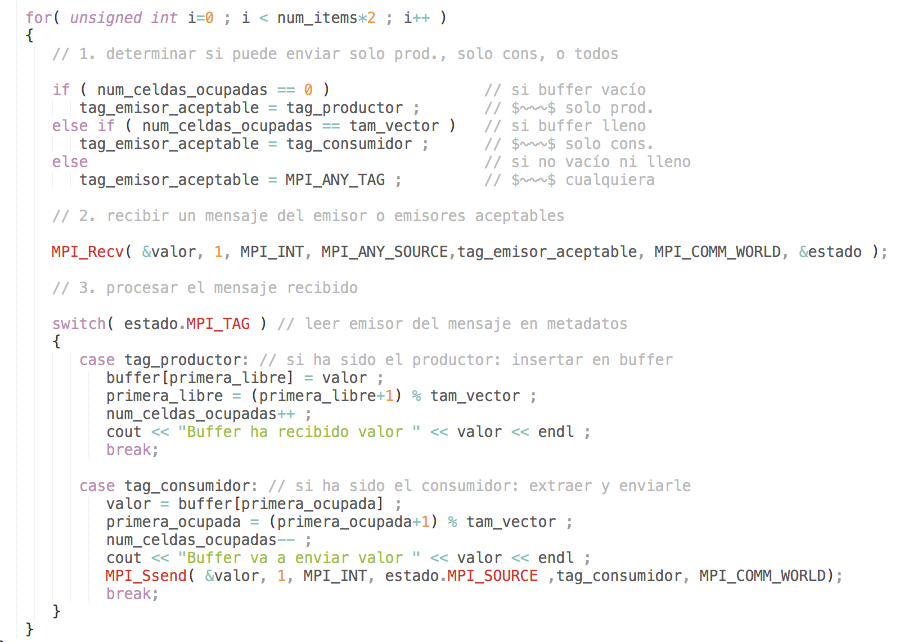
***PRODUCTOR CONSUMIDOR MULTIPLE***

*Lo primero que hacemos es calcular el ítems desde el que queremos que empiece a producir, para ello basta con multiplicar el num\_orden \* num\_items\_productores. Esto establecerá el primer elemento a producir, luego observamos que el for hará num\_items\_productores iteraciones, con lo que nos aseguramos el rango a producir.*

**

*En la función consumidor, básicamente lo que hacemos es un bucle que repetimos num\_items\_consumidores veces, en el cual enviamos una petición de 1 int del bufer de manera síncrona, a continuación lo recibimos y consumimos.*

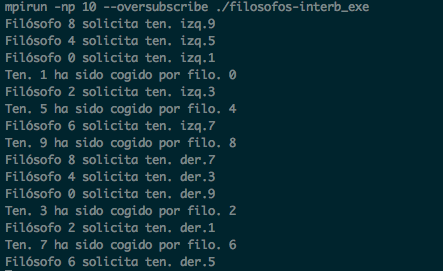
*Con respecto a la función buffer, la gran diferencia ahora radica en que filtramos según el tag del emisor, con lo que con esto controlamos la escritura y lectura del buffer.*

**

***FILOSOFOS CON INTERBLOQUEO***

*El problema que nos encontramos cuando rellenamos la plantilla facilitada de los filósofos, es que puede llegar a interbloqueo si todos piden su tenedor izquierdo a la vez, todos estarían esperando a su otro tenedor al mismo tiempo, con lo que nunca conseguirán su tenedor derecho.*

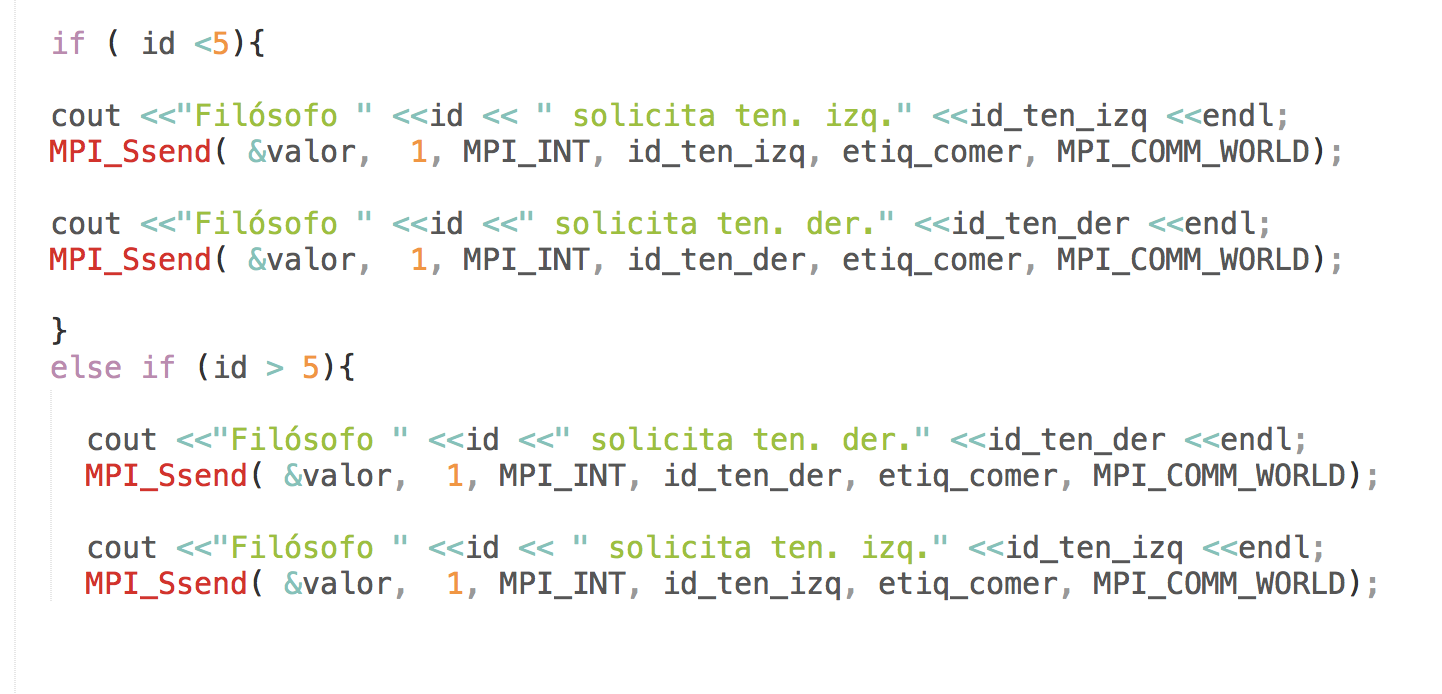
*En la siguiente imagen ofrezco una salida de este problema.*

**

*Observamos que todos los filósofos han cogido su tenedor izquierdo, con lo cual nunca van a poder coger su tenedor derecho.*

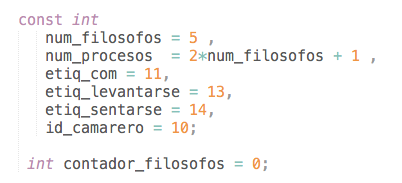
***SOLUCIÓN ENCONTRADA***

*Para solucionar este problema he optado por cambiar el orden en el que cogen los tenedores la mitad de los filósofos, si su id es mayor que 5. Con esto se consigue que nunca se llegue a esperar todos por el mismo tenedor.*

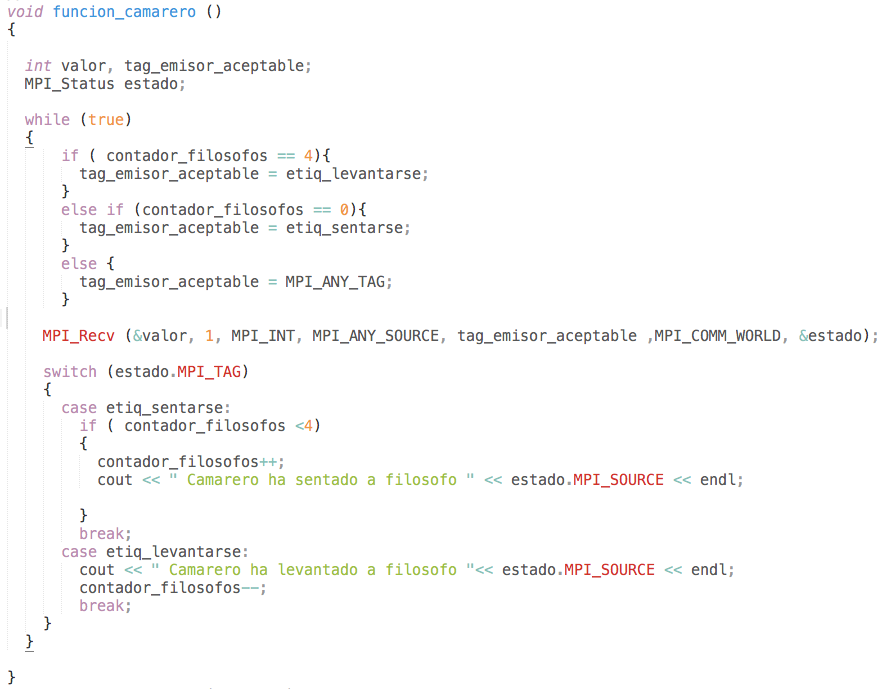
**

**SOLUCIÓN PROPUESTA**

*Comenzamos viendo las constantes definidas para la solución con el proceso camarero.*

**

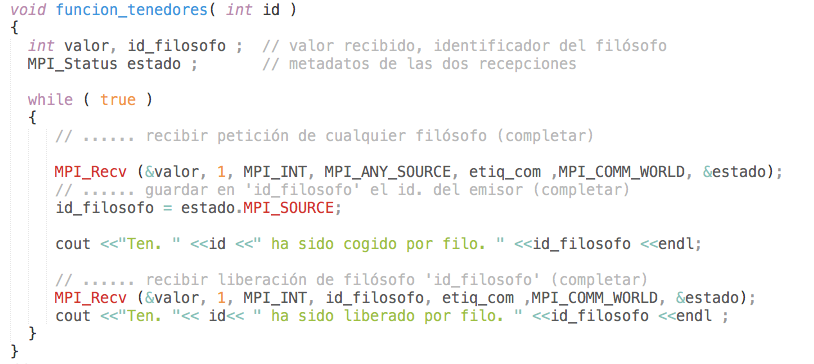
*Las dos primeras son las constantes definidas para el numero de filósofos y el numero de procesos mientras que las tres siguientes nos sirven para filtrar los procesos. El contador de filósofos nos sirve para que no haya mas de 4 filosofos sentados a la mesa en el mismo instante de tiempo, evitando asi que se produzca un posible interbloqueo.*

**

*El proceso camarero es la función clave de este problema, controla el numero de camareros sentados a la mesa y según ellos filtramos el tag aceptable, si son 4 solo levantarse, y si son 0 solo sentarse.*

*La función filósofos no merece gran mención, simplemente se limita a seguir el orden de acciones establecidas, enviando peticiones a los procesos correspondientes.*

*La función tenedores espera a recibir la llamada del proceso filosofo, y acto seguido espera su liberación*

**