**Final Project**

**B0929034林芊妤**

Ｑ：做了什麼？

Ａ：我將題目分為以下五個步驟來做，而最主要的code在於生產者移動，後

面除了消費者死亡以外，皆是由主要的程式碼去做些微的修改。我是先放

生產者再放消費者，所以消費者的移動、複製都必須顧及生產者的存在。

* 生產者移動：這個的想法跟lab13中的很像，移動要顧慮在角落或是

在牆邊，若周圍九宮格都沒有其他東西的時候，最多有

8種移動方式。我分為x y方向來表示，左(x)或上(y)是

-1，右(x)、下(y)是1，而維持在中間是0。如此一來，就

可以判斷這個生產者的周圍有幾個可移動的方向。

* 生產者複製：複製的想法跟移動幾乎是一樣的，移動時是想總共有幾

個方向可選擇，並且隨機擇一，而複製則是想總共有幾

個空的位置可置放，並且隨機擇一。此兩個member

function唯二不同的在於置放的char是\*還是空白，還有紀錄此二維陣列成員是何種東西的宣告。

* 消費者移動＆吃生產者：這個與上面最不同的地方在於，他要記錄有

沒有吃到，若是移動到生產者的位置上表示有吃到，因

此就要將他的死亡時間變回3，若是移動到空的位置上

則代表沒有吃到，那新位置的死亡時間值就會等於原位

置的死亡時間值。

* 消費者複製：複製的前提建立在以吃到的情況下，因此要將死亡時間

變回3秒。其餘概念都跟生產者的複製一樣。

* 消費者死亡：這個member function的想法是裡面最簡單的，如果這

個二維陣列上是消費者、死亡時間已經歸零、並且是顯

示沒有吃到，若這三個條件同時成立，就把這個位置變

成空的。

Ｑ：遇到什麼困難？怎麼解決

Ａ：在生產者的移動跟複製上沒有什麼太大的問題，只是要注意是否在角落或

是否在牆邊，最複雜的地方主要在於消費者要考慮的東西有很多小細節。

在消費者移動＆吃生產者這個函式需要注意下一步是移動到空的位置還是

生產者身上，這點是關鍵性的影響，因為有沒有吃到決定了要不要複製，

同時也決定了死亡時間要怎麼處理，因此放了die跟eat兩個陣列上去在這個地方花了不少時間思考。在新位置和原位置的處理要必須要思考的很清楚，我de了最久的bug是在新舊位置的更改上，因為放反了就不會有任何的動作了。我在main裡面將生產者移動、消費者移動、消費者複製三個動作各間隔0.5秒，因為這樣可以比較清楚的看出現在進行到哪一步。

Ｑ：學到什麼？

Ａ：在輸入生產者和消費者的數量時，生產者不可以太少，因為總共的範圍有快2000個，若是生產者輸入的太少，消費者就會因為無法吃到生產者而死亡，因此就無法達到像生態圈平衡一樣的狀態。