1. 策略邏輯:
2. 資訊的使用:
3. Blokus Board Status.txt:

在遊戲一開始讀來更新棋盤上的狀況，將其儲存至14X14的char Board[14][14][20]陣列中，並根據color加以判斷自己跟對方的顏色以分析棋盤上的狀況。

1. LeftPattern\_Color.txt:

每次都讀取資料的最後一行並存至大小為int pattern[21]的陣列中，判斷自己剩下那些拼圖。

1. 策略的決定:

利用Blokus Board Status.txt讀入棋盤後，可以去分析棋盤中可擺放的角，並且利用LeftPattern\_Color.txt得知剩餘的拼圖可將之擺到角上面去觀察其附近的狀況，透過翻轉四次和鏡射兩次在不影響的前提下便可將之擺放上去，當判斷完其可以擺放上去後，透過讀入的棋盤資料與放置的拼圖，可以先去計算拼圖是幾乘幾(EX:3X3 = 9)以擺上長寬站平均又大的面積存起來，再來去計算拼圖實際的面積以利實際面積的比較並存起來，接著去判斷說放上去的拼圖有著那些角對棋盤造成影響，如放上去的拼圖創造自己多少角、消滅自己多少角、阻擋對方多少角、靠到對方多少邊存起來，再來進入策略判斷條件，情況為二：

1. 假如拼圖有阻擋角的話會比沒阻擋角的情況優先放(因為消滅對方的角對方就沒角可以繼續下了)，並接著判斷的條件為　（自己角創造的面積+阻擋對方角的面積-消滅自己角的面積)，較大著優先放(因為自己角發展機會較多又可以擋對方機會好的角)，再來才是判斷上述的自身面積比較，一樣大者優先放(佔地盤也是很重要)，最後是判斷靠對方的邊越多優先放(有機會去幾對方的空隙，並且消滅對方角)。
2. 假如今天沒有辦法阻擋對方的角，則次序則會換成，先算自身的面積，大者優先放(因為沒辦法消滅對方角時占地盤較重要)，在來是判斷靠對方多少邊優先放(因為才有機會下一步消滅對方的角)，最後才是判斷為　（自己角創造的面積+阻擋對方角的面積-消滅自己角的面積) ，較大著優先放。

2.策略架構:

(1) 我利用了 char Board[14][14][20]的陣列去存取14x14的棋盤資料，int pattern[21]存取剩餘的拼圖，char puzzle[21][5][5][20]去存取21塊拼圖的資料。剛開始先建立拼圖，再來讀取棋盤資料，跟剩餘拼圖，接著判斷是不是第一次放拼圖，如果是第一次放，會看棋盤(4,4)或(9,9)有無被佔，一個有則會放另一個位置，都沒有則下(4,4)，並且我會優先固定放16號拼圖到適當的位置，不是第一次放，先算出可放的角，再來將所有剩下的拼圖丟進去做旋轉、鏡射判斷可不可以放，可以的話就判斷我策略邏輯所說的步驟，並將最好的位置跟拼圖資訊都存起來，有更好的話便會取代掉，最後將判斷最好的情況擺上棋盤，並寫回資料。

(2) 如此設計的優點在於能夠限制對方角的發揮並且讓自己有機會擺更多東西，還有能夠在一開始有效放大的佔領地盤，缺點是無法清楚馬上畫出自己的領域，且放到最後不會去為下一步甚至到下下步著想，而導致最後沒有角可以讓小的拼圖補進去空的面積，總總情況會對方有機可乘。

(3) 因為我用了21個5x5的陣列去存放拼圖(因為拼圖長寬最大為5)，優點是翻轉跟鏡射可以用統一的function就將他搞定，而且清楚標明0-20(拼圖1-21)的陣列順序讓判斷擺放策略時易懂且容易統一處理(因為邏輯相同且知5x5為列乘行)，但缺點就是浪費空間做資料的擺放，拼圖實際面積不用那麼多空間，還有每次都要先計算其開始的列、行跟結束的列行，才不會讀到都是N的元素，除此之外我還有14x14的陣列存取棋盤資料，優點一樣是清楚易懂且好判斷，缺點也一樣就是佔空間。