一、 所有交易策略和邏輯

Note : 這邊寫的 strategy X 指的是在 trading 函式中的 case X，而不是 cout 裡面寫的策略 X。

1. buy and hold – strategy 1

   當天最一開始時買進，最後結尾時賣出。

2. buy and hold + 停損 – strategy 2

   會多傳入一個參數，並根據開盤價和參數決定要停損的價格。

3. buy and hold + 停損 + 停 利 – strategy 3

   在策略2的基礎上，多傳一個參數以決定停利的價格。

4. 簡化版葛蘭碧交易法則 – strategy 4

   先計算價格的移動平均線，再根據葛蘭碧交易法則的判斷賣出和買入的訊號，並針對手上有的期貨數量做出相對應的動作，而當時間到最後一分鐘時，結算手上的股票。

5. 簡化版葛蘭碧交易法則 + 最佳停損 – strategy 5

   前面跟 strategy 4 差不多，但多加了停損的部分，並寫了一個 beststop 函式(後面介紹)找出最佳停損參數。

6. 每天選最佳獲利策略 – strategy 7

   先看哪一種策略能在當天賺最多錢，再決定用哪個策略，如果全部策略都虧錢的話，當天直接不做投資。感覺這個有點作弊，我也不知道符不符合要求，但既然做了就放上來吧。

二、 功能(基本+額外)

1. 單日最大虧損 and 獲利 / 累計獲利

   我是用一個陣列把每天結算後的錢存下來，再用 algorithm 中的sort 函式將陣列中的值由小到大排序，最高獲利就會是最後一項，而最高虧損則是第一項。累計獲利則是用 for 迴圈跑每一天，將每一天的值加起來。

2. 最佳化參數

   在 strategy 2 和strategy 5 中，都會有一個 stoploss 參數，因此我寫了一個 beststop 函式，他主要是用一個 for 迴圈將i 從 0.005 跑到 0.06，每次都加0.0001，將 stoploss 參數帶 i 後，算出每種i 的獲利， 並回傳獲利最大的 i，又因為我有兩個策略都會用到這個函式，所以寫了一個 switch 來處理，當時為了保存老師原本的程式碼，所以我 strategy 2 的最佳化是直接寫在 case 6。

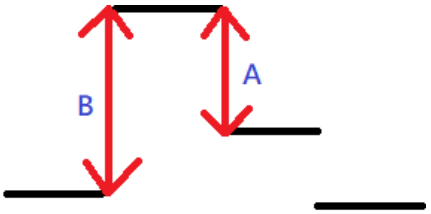
3. 極限獲利 and 虧損(不考慮做空) – strategy 8 and 9

   如果不考慮做空的話，每天極限的獲利就會是所有上漲的部分加起來，而極限的虧損則是下跌的部分全部加起來，所以就是用一個 for 迴圈算所有當日資料相差的值，判斷上漲或下跌後決定要不要加。

4. 極限獲利(考慮做空) – strategy 10

   上漲的部分買一口(1)，下跌的部分空一口(-1)，所以可以暫時把獲利弄成當日所有時間和前一段時間的價差(我寫了一個 abs 函式把它全部變成正的)*200，但因為

(賣一口跟空一口)或(買一口跟買回空的那口)不能同時執行，所以要再減掉轉折點前後比較小的價差，我以由漲轉跌作為例子。如圖，



這中間那三分鐘中，原本最一開始算的獲利是(A+B)*200， 但如果這樣的話，策略會是第一分鐘持一口，第二分鐘賣掉且同時空一口，第三分鐘買回，但因為第二分鐘那樣的操作是不被允許的，所以不可能有辦法賺到(A+B)*200 元。我畫的圖中，B>A，所以能有最大獲利的策略是賺了 B 後，把原本持有的賣掉，並在下一分鐘空一口。這樣操作的話就會比原本的(A+B)*200少 賺 A(較小的差距)*200，因此必須從原本的總金額扣除。而若 B<A 的話，策略就會改成轉折點前一分鐘賣掉，再轉折點時空一口，因此需要扣掉的是B(較小的差距)*200。以此類推，由跌轉漲時也是一樣要扣掉轉折點兩端較小的差距。(中間的價格-前面的價格)和(中間的價格-後面的價格)如果同號的話(++ or --) 代表遇到了轉折點，我用一個for迴圈尋找當日的轉折點，並再找到一個轉折點時，就依照上面的邏輯扣掉比較小的那個差距，最後就會得到當日極限的獲利。

5. UI

我用高中學的程式語言寫的，我只有把數字套進去而已，所以交易相關的程式碼都還是在 C++那份檔案裡，看的時候記得調螢幕大小~~~希望助教喜歡~~~

[final project !! - OpenProcessing]