

Computer Programming Final Project

Group 17 : B08502048 陳杰伸、B08502149 林千驊、B09901079 劉碩

Connect Four:

Readme

1.IDE: Dev-C++

2.Source File: Group_17 main.cpp

3.Compile Command: Unzip whole “Group_17.zip” folder. Then, open file “Group_17 main.cpp” in Dev-C++ and press “F11” to compile and run.

Notes: This is a gaming program called connect four (四連棋) that you can either play with another player or with AI written by ourselves. The winning condition is to connect four of your symbols on the 6x7 board with gravity applied.

1. Problem Description:

四連棋是以以前的回憶，為簡單有趣的益智遊戲，但在手遊盛行下逐漸被人遺忘。因此我們想利用所學找回最純粹的快樂。另外，最後我們完成的遊戲進行方法為下：首先，玩家輸入自己的名字及符號，若想和電腦對戰，則在 Player2 名字輸入「AI」，再來利用方向鍵左右決定要下哪一行，並按下 ENTER 放置棋子。過程中，兩玩家輪流動作，直到分出勝負。一局結束後，可選擇是否繼續遊戲，而下一場由下家開始。

2. Method:

程式碼分為四個部分，以下分別敘述，其中介面設計及 AI (p.4-6) 最有特色。

a. Player information class

```
1  #include <bits/stdc++.h>
2  #include <conio.h>
3  #define KEY_LEFT 75
4  #define KEY_RIGHT 77
5  #define KEY_ENTER 0x0d
6  using namespace std;
7  bool gamestatus,enable=false;
8  bool ailllose[7]={false},aiwin[7]={false},aihelp[7]={false};
9  int spa,turn=0;
10 char arr[6][7];
11 string winningname="";
12
13 //class that stores player information
14 class Player{
15     friend ostream& operator>>(ostream&,Player&);
16     friend void control();
17     private:
18         string name;
19         char symbol;
20     public:
21         Player():name(""),symbol(' '){}
22         ~Player(){name=""; symbol=' ';}
23         void checkWin(int,int);
24         char getSymbol(){
25             return symbol;
26         }
27         string getName(){
28             return name;
29         }
30     };
```

(line 7-11) 包含全域變數，gamestatus 為遊戲狀態、enable 為避免吃到多餘的 ENTER 鍵、三個陣列為後面 AI 會使用到的數據、spa 為輸入位置、turn 為計算回合、arr[6][7]為棋盤、winningname 為勝利者姓名。

(line 14-30) 宣告 class 的內容，包含是 data member 的名字、符號及 checkWin 和 get 函式。

```

32 //operator overloading
33 istream& operator>>(istream& input, Player& user){
34     static int x=1;
35     if(x>2) x=2;
36     cout<<"Input player "<<x<<"'s name: ";
37     input>>user.name;
38     cout<<"Input player "<<x<<"'s symbol (one character): ";
39     input>>user.symbol;
40     input.ignore(numeric_limits<streamsize>::max(), '\n');
41     x++;
42     return input;
43 }

```

(line 33-43) 重新定義符號「>>」以方便 main function 的撰寫，並有輸出文字作為輸入提示，同時忽略掉在輸入符號時可能多輸入的字。

```

45 //check winning condition
46 void Player::checkWin(int x,int y){
47     int coutx=0,couty=0,couts=0,coutw=0;
48
49     int X=x;
50     int Y=y;
51     if(x==-1) return;
52     while(x<6&&coutx!=4){
53         if(arr[x][y]==symbol){
54             coutx++;
55             x++;
56         }
57         else{
58             break;
59         }
60     }
61     x=X;
62     coutx--;
63     while(x>-1&&coutx!=4){
64         if(arr[x][y]==symbol){
65             coutx++;
66             x--;
67         }
68         else{
69             break;
70         }
71     }
72     x=X;
73     while(y<7&&couty!=4){
74         if(arr[x][y]==symbol){
75             couty++;
76             y++;
77         }
78         else{
79             break;
80         }
81     }
82     y=Y;
83     couty--;
84     while(y>-1&&couty!=4){
85         if(arr[x][y]==symbol){
86             couty++;
87             y--;
88         }
89         else{
90             break;
91         }
92     }
93     y=Y;

```

(line 46-148) 判斷勝利條件。

(line 47-50) coutx 記錄縱向連續棋子數。couty 記錄橫向連續棋子數。couts 記錄左上至右下連續棋子數。coutw 記錄左下至右上連續棋子數。X 和 Y 分別記錄輸入棋子位置。

(line 51) 在 341 行中，為了讓程式順利運行，若無法回傳正確棋子位置則 return -1。因此若 checkWin 接收值為 -1，則直接 return 不判斷。

(line 52-61) 以輸入的棋子為起點，向下判斷連續相同棋子的數目。當判斷位置超過棋盤範圍亦或連續棋子數已達 4 個，則結束判斷並重置判斷位置。

(line 62-72) 以輸入的棋子為起點，向上判斷連續相同棋子的數目。當判斷位置超過棋盤範圍亦或連續棋子數已達 4 個，則結束判斷並重置判斷位置。由於在輸入棋子位置的點會重複計算到，因此在 62 行先將計數器減一。

(line 73-93) 依照上述步驟，判斷橫向的連續棋子數。

```

94 while(x<6&&y<7&&couts!=4){
95     if(arr[x][y]==symbol){
96         couts++;
97         x++;
98         y++;
99     }
100     else{
101         break;
102     }
103 }
104 x=X;
105 y=Y;
106 couts--;
107 while(x>-1&&y>-1&&couts!=4){
108     if(arr[x][y]==symbol){
109         couts++;
110         x--;
111         y--;
112     }
113     else{
114         break;
115     }
116 }
117 x=X;
118 y=Y;

119 while(x>-1&&y<7&&coutw!=4){
120     if(arr[x][y]==symbol){
121         coutw++;
122         x--;
123         y++;
124     }
125     else{
126         break;
127     }
128 }
129 x=X;
130 y=Y;
131 coutw--;
132 while(x<6&&y>-1&&coutw!=4){
133     if(arr[x][y]==symbol){
134         coutw++;
135         x++;
136         y--;
137     }
138     else{
139         break;
140     }
141 }
142
143 if(coutx==4||couty==4||coutw==4||coutv==4){
144     gamestatus=false;
145     winningname=name;
146 }
147
148 }

```

(line 94-118) 依照上述步驟，判斷左上至右下的連續棋子數。

(line 119-141) 依照上述步驟，判斷左下至右上的連續棋子數。

(line 143-146) 若達勝利條件(即任意方向連續棋子數為4)，則結束遊戲並記錄勝利者名字。

b. Gaming function

```

150 //initialize array
151 void Initialize(){
152     for(int i=0;i<6;i++){
153         for(int j=0;j<7;j++){
154             arr[i][j]=' ';
155         }
156     }
157 }
158 //restart the game
159 void Restart(Player player[]){
160     char willing;
161     cout<<"Do you want to restart (Y/N): ";
162     cin>>willing;
163     cin.ignore(numeric_limits<streamsize>::max(),'\n');
164     if(willing=='Y'){
165         Initialize();
166         winningname="";
167         gamestatus=true;
168         cout<<player[turn%2].getName()<<" first!\n";
169         cout<<"Press any key to restart the game...\n";
170         if(player[turn%2].getName()=="AI")
171             getch();
172     }
173 }

```

(line 151-156) 為初始化棋盤的函式，清空所有棋盤上的符號。

(line 161-162) 當一局結束後，詢問是否重新開始遊戲。

(line 163-172) 只讀取第一個字放入 willing，若為「Y」則重新開始遊戲，並且將遊戲所需的參數重新設置，同時顯示下家開始。

```

175 //check if it is full
176 void checkFull(){
177     bool isFull=true;
178     for(int h=0;h<7;h++){
179         if(arr[0][h]!=' '){
180             isFull=false;
181         }
182     }
183     if(isFull==true&&winningname==""){
184         cout<<"No player wins!\n";
185         cout<<"Press any key to continue the game...\n";
186         getch();
187         gamestatus=false;
188     }
189 }

```

(line 177-181) 判斷是否全滿，如果是的話，則 isFull 為 true。

(line 183-188) 如果全滿且沒有玩家勝利的話，則輸出沒有玩家勝利並修改遊戲狀態。

```

305 //get input position from player
306 int Control(Player player[]){
307     int c=0;
308     switch(c=getch()) {
309     case KEY_LEFT:
310         if(spa<=0) spa=0;
311         else spa--;
312         break;
313     case KEY_RIGHT:
314         if(spa>=6) spa=6;
315         else spa++;
316         break;
317     case KEY_ENTER:
318         if(enable==true){
319             if(arr[0][spa]!=' '){
320                 cout<<"This column is Full!\n";
321                 cout<<"Press any key to continue the game...\n";
322                 getch();
323                 break;
324             }
325         }
326         for(int w=5;w>=0;w--){
327             if(arr[w][spa]!=' '){
328                 continue;
329             }
330             else{
331                 arr[w][spa]=player[turn%2].getSymbol();
332                 turn++;
333                 return w;
334                 break;
335             }
336         }
337         break;
338     default:
339         break;
340     }
341     return -1;
342 }

```

(line 306-342) 控制程序，讀取方向鍵與 ENTER 鍵，其餘按鍵則不會有動作。

(line 308-316) 讀取左右鍵，左鍵會減少游標前的空格，右鍵會增加，且避免超出範圍。

(line 317-324) 如果按下 ENTER 鍵，先判斷游標所在該行有沒有空格可擺(用游標前的空格判斷在哪一行)，若沒有的話中斷輸入，並重新回使用者介面再讀取按鍵。

(line 326-328) 將下的棋子用迴圈拉到最下面可放置的空間。

(line 329-334) 下棋子並加回合數，下的棋子是回合數%2 (0 代表先手方，1 代表後手方)。

```

344 //print and check array
345 void UserUI(Player player[]){
346     int x; int y;
347     if(player[1].getName()!="AI"||turn%2==0){
348         x=Control(player);
349         y=spa;
350     }
351     else if(turn%2==1){
352         y=AI(player);
353         for(int w=5;w>=0;w--){
354             if(arr[w][y]!=' '){
355                 continue;
356             }
357             else{
358                 arr[w][y]=player[1].getSymbol();
359                 turn++;
360                 x=w;
361                 break;
362             }
363         }
364     }
365     system("cls");
366     cout<<" ";
367     for(int h=0;h<spa;h++){
368         cout<<" ";
369     }
370     cout<<"Y"<<endl;
371     cout<<" 1 ";
372 }
373 for(int z=2;z<=7;z++){
374     cout<<" "<<z<<" ";
375 }
376 }
377 cout<<endl;
378 for(int i=0;i<6;i++){
379     for(int k=0;k<13;k++){
380         cout<<"-+<<" ";
381     }
382     cout<<"-+<<endl;
383     cout<<"|";
384 }
385 for(int g=0;g<7;g++){
386     cout<<" "<<arr[i][g]<<" "<<"|";
387 }
388 cout<<endl;
389 }
390 for(int k=0;k<13;k++){
391     cout<<"-+<<" ";
392 }
393 cout<<"-+<<endl;
394 cout<<player[turn%2].getName()<<"'s turn."<<endl;
395 }
396 player[(turn-1)%2].checkWin(x,y);
397 checkFull();
398 }

```

(line 345-398) 使用者介面，負責輸出畫面與 IO 控制。

(line 347-350) 如果該回合是人類下，那就把控制參數設為人類使用的參數(游標等)。

(line 351-363) 如果該回合是電腦下，把控制參數設為電腦專用的參數。

(line 364) 清除整個畫面，達到刷新螢幕的效果。

(line 365-394) 列印棋盤與游標，棋子等內容。棋盤固定不會動，游標因為前面的空格可調整左右的位置。顯示棋子的方式則是讀取該格子的資料，如果沒資料就輸出空格，以保持格式正確的效果，有資料則輸出對應的符號。

(line 396-397) 用對應的 Function 確認遊戲是否還需要進行下去。

c. AI

```

191 //AI will lose
192 void AIplayer1Win(Player player[]){
193     for(int u=0;u<7;u++) ailose[u]=false;
194     for(int v=0;v<7;v++){
195         for(int h=5;h>=0;h--){
196             if(arr[h][v]!=' ')
197                 continue;
198             else{
199                 arr[h][v]=player[0].getSymbol();
200                 player[0].checkWin(h,v);
201                 ailose[v]=(!gamestatus);
202                 gamestatus=true;
203                 arr[h][v]=' ';
204                 winningname="";
205                 break;
206             }
207         }
208     }
209 }
210
211 //AI can win
212 void AicanWin(Player player[]){
213     for(int u=0;u<7;u++) aiwin[u]=false;
214     for(int v=0;v<7;v++){
215         for(int h=5;h>=0;h--){
216             if(arr[h][v]!=' ')
217                 continue;
218             else{
219                 arr[h][v]=player[1].getSymbol();
220                 player[1].checkWin(h,v);
221                 aiwin[v]=(!gamestatus);
222                 gamestatus=true;
223                 arr[h][v]=' ';
224                 winningname="";
225                 break;
226             }
227         }
228     }
229 }

```

(line 193) 清空每一行的狀態，此陣列存 AI 是否會輸的資料。

(line 194-208) 從第一行開始，判斷每一行可放置棋子的最低位置。運用 checkWin 判斷，若對手的棋子下在這個位置即可獲勝(即對手的棋子放置在此可達成連線)，則使 ailose 的狀態為 true，使 AI 下在該位置阻止對方獲勝，並重置因執行 checkWin 而改變的參數。

(line 213) 清空每一行的狀態，此陣列存 AI 是否會贏的資料。

(line 214-228) 從第一行開始，判斷每一行可放置棋子的最低位置。運用 checkWin 判斷，若棋子下在這個位置能夠獲勝(即棋子放置在此可達成連線)，則使 aiwin 的狀態為 true，使 AI 直接下在此位置，並重置因執行 checkWin 而改變的參數。

```

231 //AI help win
232 void AIhelpWin(Player player[]){
233     for(int u=0;u<7;u++) aihelp[u]=false;
234     for(int v=0;v<7;v++){
235         for(int h=5;h>=0;h--){
236             if(arr[h][v]!=' ')
237                 continue;
238             else{
239                 if(h-1>=0){
240                     arr[h-1][v]=player[0].getSymbol();
241                     player[0].checkWin(h-1,v);
242                     aihelp[v]=(!gamestatus);
243                     gamestatus=true;
244                     arr[h-1][v]=' ';
245                     winningname="";
246                     break;
247                 }
248             }
249         }
250     }
251 }

```

(line 233) 清空每一行的狀態，此陣列存 AI 是否會幫助對手贏的資料。

(line 234-250) 從第一行開始，判斷每一行可放置棋子的最低位置。運用 checkWin 判斷，若棋子下在這個位置會導致對手獲勝(即對手的棋子放置在其上可達成連線)，則使 aihelp 的狀態為 true，使 AI 避免下在該位置，並重置因執行 checkWin 而改變的參數。

```
253 //AI
254 int AI(Player player[]){
255     int y,g;
256     int flag=0;
257     srand(time(0));
258     AIcanWin(player);
259     AIplayer1Win(player);
260     AIhelpWin(player);
261
262     for(int k=0;k<=6;k++){
263         if(aiwin[k]==true){
264             return k;
265         }
266     }
267
268     for(int l=0;l<=6;l++){
269         if(ailose[l]==true){
270             return l;
271         }
272     }
273
274     int nomorestep;
275     do{
276         flag=0;
277         nomorestep=-1;
278         g=(rand()%30);
279         if(g==0&&aihelp[0]==false) y=0;
280         else if (g>=1&&g<=3&&aihelp[1]==false) y=1;
281         else if (g>=4&&g<=9&&aihelp[2]==false) y=2;
282         else if (g>=10&&g<=19&&aihelp[3]==false) y=3;
283         else if (g>=20&&g<=25&&aihelp[4]==false) y=4;
284         else if (g>=26&&g<=28&&aihelp[5]==false) y=5;
285         else if (g==29&&aihelp[6]==false) y=6;
286         else{
287             for(int h=0;h<7;h++){
288                 if(aihelp[h]==true||arr[0][h]!=' '){
289                     nomorestep=1;
290                 }
291                 else{
292                     nomorestep=0;
293                     break;
294                 }
295             }
296             if(nomorestep==1)
297                 y=rand()%7;
298             else
299                 flag=-1;
300         }
301     }while(arr[0][y]!=' '||flag==-1);
302     return y;
303 }
```

(line 255-260) 此行開始為 AI 主程式。定義 variable (y 為判斷完最後下的位置、g 為隨機取樣所得到的數字)，並執行 AIcanWin、AIplayer1Win、AIhelpWin。

(line 262-266) 首先判斷是否能獲勝。若執行完 AIcanWin 後 aiwin[k] 為 true (即下在 k 行能獲勝)，則直接 return k。

(line 268-272) 再來判斷對手是否能獲勝。若執行完 AIplayer1Win 後 ailose[l] 為 true (即對手下在 l 行能獲勝)，則直接 return l。

(line 274-302) 若上述兩項條件都不符合，則利用 do while 迴圈開始隨機取樣(g)，根據得到的數字決定要放置棋子的地方(y)。由於棋子置於棋盤中間較有機會獲勝，因此中間行數的機率較左右兩邊大。而經由 AIhelpWin 判斷，若下在選定的位置會幫對手獲勝(aihelp[y] 為 true)，則進入 else 並判斷是否有其他地方可以下，此即 nomorestep 之值。當其值為 1 時，即沒有地方可下，則隨機產生 y 值。另一方面，當 nomorestep 值為 0 時，即還有其他地方可以下，則將 flag 改為 -1 以重新進行迴圈。最後，在 while 條件有確保該行未滿，若滿則重新進行迴圈，而當跳出迴圈時 return y。

d. Main function

```

400 int main(){
401     cout<<"-----\n";
402     cout<<"| Welcome to Connect Four |\n";
403     cout<<"-----\n\n";
404     cout<<"Input AI in player 2 to start AI.\n\n";
405     Initialize();
406     gamestatus=true;
407     //initialize
408     Player player[2];
409     cin>>player[0]>>player[1];
410     //input player
411     while(player[0].getSymbol()==player[1].getSymbol()||player[0].getName()==player[1].getName()){
412         cout<<"Repeated. Input again!\n";
413         cin>>player[1];
414     }
415     //check validation
416     cout<<"Press any key to start the game...\n";
417
418     do{
419         UserUI(player);
420         enable=true;
421         if(gamestatus==false){
422             if(winningname!="")
423                 cout<<winningname<<" win!\n";
424
425             Restart(player);
426             enable=false;
427         }
428     }while(gamestatus==true);
429     //gaming while loop
430     cout<<"Thanks for playing!"<<endl;
431     return 0;
432 }

```

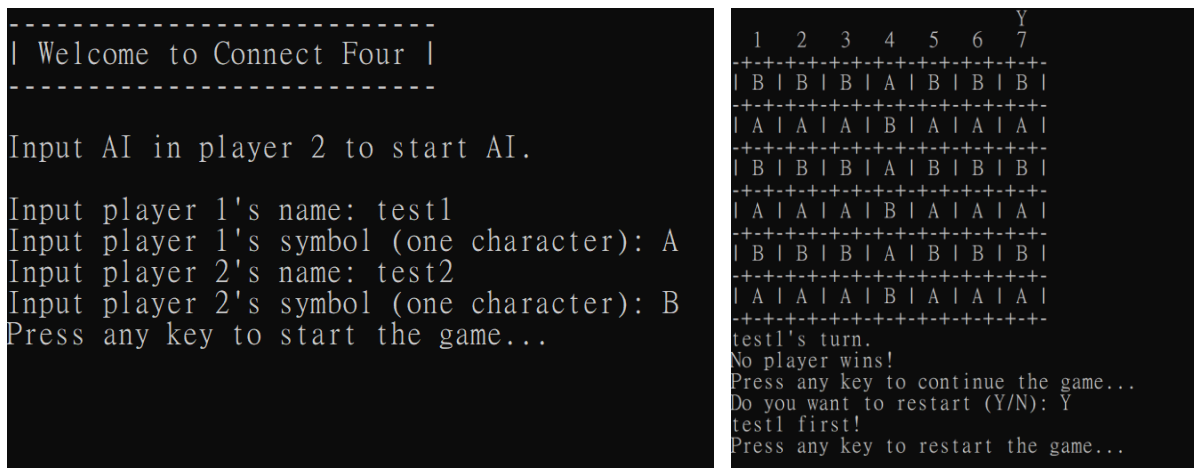
(line 405-406) 初始化遊戲。

(line 408-409) 利用重新定義過的「>>」輸入玩家資料。

(line 411-414) 避免玩家名稱或符號重複，導致不知道是誰的回合，或勝利判斷錯誤。

(line 418-428) 遊戲進行的 do while 迴圈，執行 UserUI，且當 gamestatus 為 false 時，若有人勝利則印出玩家姓名，並詢問是否重新開始，當 gamestatus 依然為 false 則結束遊戲。

3. Results:



The left screenshot displays the game's initial interface. It features a title bar with a dashed line, followed by the text "Welcome to Connect Four". Below this, it prompts the user to "Input AI in player 2 to start AI.". The main input section asks for "Input player 1's name: test1", "Input player 1's symbol (one character): A", "Input player 2's name: test2", and "Input player 2's symbol (one character): B". It concludes with "Press any key to start the game...".

The right screenshot shows a full game board. The board is a 7x7 grid with columns numbered 1 to 7 and a 'Y' header. The pieces are placed as follows: Column 1: A, B, A, B, A, B, A; Column 2: A, B, A, B, A, B, A; Column 3: A, B, A, B, A, B, A; Column 4: A, B, A, B, A, B, A; Column 5: A, B, A, B, A, B, A; Column 6: A, B, A, B, A, B, A; Column 7: A, B, A, B, A, B, A. The terminal window below the board shows the game's progress: "test1's turn.", "No player wins!", "Press any key to continue the game...", "Do you want to restart (Y/N): Y", "test1 first!", and "Press any key to restart the game...".

此為兩個玩家對局。

首先為輸入名字以及要使用的符號，並按任意鍵來開始遊戲。

用左右控制要下的位置，並按 ENTER 輸入，滿的地方沒辦法輸入。

此為全滿且沒人勝利的結果，並輸入「Y」由下家開始重新遊戲。

```

-----
| Welcome to Connect Four |
-----

Input AI in player 2 to start AI.

Input player 1's name: TEST
Input player 1's symbol (one character): 1
Input player 2's name: AI
Input player 2's symbol (one character): 1
Repeated. Input again!
Input player 2's name: AI
Input player 2's symbol (one character): 2
Press any key to start the game...

      Y
    1  2  3  4  5  6  7
  +--+--+--+--+--+--+--+
  |  |  |  |  |  |  |  |
  +--+--+--+--+--+--+--+
  |  | 2 | 2 | 1 |  |  |
  +--+--+--+--+--+--+--+
  |  | 1 | 2 | 2 | 2 |  |  |
  +--+--+--+--+--+--+--+
  |  | 1 | 1 | 2 | 1 |  |  |
  +--+--+--+--+--+--+--+
  |  | 1 | 1 | 1 | 2 |  |  |
  +--+--+--+--+--+--+--+
  |  | 2 | 1 | 2 | 2 | 1 |  |
  +--+--+--+--+--+--+--+
  TEST's turn.
  AI win!
  Do you want to restart (Y/N): N
  Thanks for playing!

```

此為在 player2 輸入「AI」來跟 AI 對局，若符號重複則請求重新輸入。
 首先為輸入名字以及要使用的符號，並按任意鍵來開始遊戲。
 用左右控制要下的位置，並按 ENTER 輸入，滿的地方沒辦法輸入。
 AI 會自動填上其想要下的位置。
 此為 AI 勝利的結果，並輸入「N」來結束遊戲。

4. Work division:

姓名	個人部分	共同部分
陳杰伸	主程式、類別及剩餘小函式	AI 的想法以及完成 做投影片以及報告
林千驊	判斷是否勝利	
劉 碩	使用者介面及方向鍵操作	

5. Discussion:

a. 遊戲介面的優化

這次由於時間緊迫，我們無法應用 SDL2 美化遊戲介面，只能以程式印出簡易的棋盤模樣。因此，改善我們的遊戲介面使其更加吸引人，是我們之後可以進步的方向。

b. AI 的改進

我們所設計的簡易 AI 僅能預判下一步可下或不可下的位置，且是以隨機變數決定要下的位置，導致勝率並非完美。因此若能整理出四連棋的戰術策略或者讓 AI 判斷更多步棋之後的結果，勢必能創造出更具挑戰性的 AI 和玩家對決。

c. 困難討論及心得

針對這次報告，我們覺得最困難的部份，是要將雙人對戰以及 AI 對戰的功能寫在同一程式，並讓使用者進行選擇，最後將兩個合併在函式 UserUI 中解決。另外，我們也藉由不斷測試，發現了原本程式不合理的地方，其中很容易將矩陣行列寫反，或者是搞錯矩陣位置編號。至於在 AI 主程式的設計，是經過很多次的修改才達到我們預期的結果。最後，多人合作最大的挑戰是要整合程式，因為每個人的想法存在些許差異。我們後來也發現程式有一點部分重疊，將來可以考慮另外寫成函式做精簡。