

撲克牌遊戲：九九

計算機程式設計 期末書面報告

余京儒

曾柏學

李冠緯

摘要

九九是一款有趣的撲克牌遊戲，規則很簡單，就算是小學生也很容易上手，更不見得會輸給大孩子或是大人，因此玩過的人很多。其實九九這個遊戲很奇妙，為了達到勝利不需要過人的智慧，操作也可以很機械化，可能或多或少需要一點心理戰術，但是如果手牌比別人差還是會輸，對，這是一個純粹的運氣遊戲！就憑這幾點，九九非常適合作為我們程式設計初學者的題目。以下這篇報告要說明我們是如何寫作撲克牌遊戲九九的程式。

目錄

一、 研究動機

二、 理論說明

三、 實作流程架構與研究過程

四、 實作結果與討論

五、 參考資料

六、 程式列表

一、研究動機

這份報告我們以組員某次作業為基礎做延伸，最一開始做這個題目的原因是幾年前這位同學很喜歡跟朋友玩這款遊戲，甚至偶爾還會拿真正的撲克牌自己跟自己玩。長大以後仔細想想這個遊戲雖然跟朋友玩還是很好玩，但是自己跟自己玩這個畫面實在是不忍直視。因此學習了程式設計之後，就想要寫一個可以不用知道對手手牌的九九遊戲以增加趣味性，為邊緣人帶來福音。

那這次報告延續這個發想，將原本未完成的作品寫成一個真正的遊戲。

二、課程相關與文獻回顧

九九這款遊戲的規則我們都計得很清楚，所以沒有參考相關的資料，而是直接抱著兩本課程講義，閉門造車的寫作程式碼，也沒有參考別人是如何寫的。

這學期課程教的東西我們運用的不少，像是最基本的變數、各種迴圈(流程控制)、陣列、函式還有類別，另外繼承也有派上用場，但是多型的技巧還不是很熟悉，因此沒有使用。此外值得一提的是撲克牌在我們的程式碼中也是使用老師上課所講 0 到 51 整數代表的資料結構，不過花色順序有因應需求稍作修改。

三、理論說明

此版本的九九撲克牌遊戲有以下規則：

1. 玩家可以自行設定與之對戰的電腦數
2. 玩家永遠手持五張牌，輪到自己出牌時，每次只需出一張牌。出牌後，當計算牌堆中牌的點數總和超過 99 時遊戲即結束（稱作爆掉）。
3. 排組中有些牌屬於功能牌，功能牌的花色數字與其對應功能關係如下：
 - a. 黑桃 A：歸零
 - b. 所有花色的 4：迴轉
 - c. 所有花色的 10 和 Q：分別自行決定加減 10 或 20、
 - d. 所有花色的 J：跳果這次出牌階段
 - e. 所有花色的 K：直接使牌堆值變成 99
4. 玩家在出完牌後可以指定下一位出牌者。

5. 牌堆總和不能為負數，因此倘若減掉 10 或 20 後牌堆會變成負數時，凡出現 10 或 Q 均會自動選擇加法，而不讓玩家選擇加或減。
6. 若牌堆抽完後遊戲還未結束，則會重新洗一副新的牌繼續遊戲
7. 在此版本中，電腦玩家會優先選擇與目前累積數加起來不超過 99 的牌出

四、實作流程架構

先介紹主程式的虛擬碼：

1. 確認遊玩人數
2. 宣告物件遊戲桌面、人類玩家、物件陣列玩家、建立新遊戲
3. do{
 - 3.1 if(輪到玩家出牌){
 - 3.1.1 顯示玩家手牌
 - }
 - 3.2 顯示排疊總和
 - 3.3 出牌 //包含出牌會發生的所有事情}while(排疊總和<=99)
4. 顯示遊戲結束

再介紹重要步驟 3.3 出牌：類別 Table 的成員函式 turn

- 3.3.1 if (輪到人類玩家出牌){
 - 3.3.1.1 確認選牌
 - }
- 3.3.2 for (i = 0 到總人數){
 - 3.3.2.1 if (輪到某位玩家出牌){
 - 3.3.2.1.1 電腦玩家判斷出牌
 - 3.3.2.1.2 顯示出牌 //類別 Card 的成員函式 show，

```

handcard                                     //這些牌是類別 Player 的成員物件
3.3.2.1.3      執行牌的功能                  //類別 Card 的成員函式 function
3.3.2.1.4      玩家抽牌                      //類別 Player 的成員函式 GetaCard
3.3.2.1.5      break;
    }
}

```

在此程式中，我們一共規劃了四個類別，分別為 Card、Player、Human 以及 Table。

1. Card :

- a. 成員變數 card：建構是必須給不同物件對應的值，由 0-51 的編號代表全 52 張。
- b. 成員函式 show：將卡牌轉為其對應的花色加數字(用 if - else if 結構)。
- c. 成員函式 function：定義特殊功能牌的功能(用 if - else if 結構)。

2. Table：堆放幾個重要函式

- a. NewCardStack 洗牌(只是將 0 到 51 的數字亂排)。
- b. turn 如虛擬碼所示。
- c. next() 沒有指定、迴轉情況下，出牌順序順延，因此在 Card.function() 參考使用。

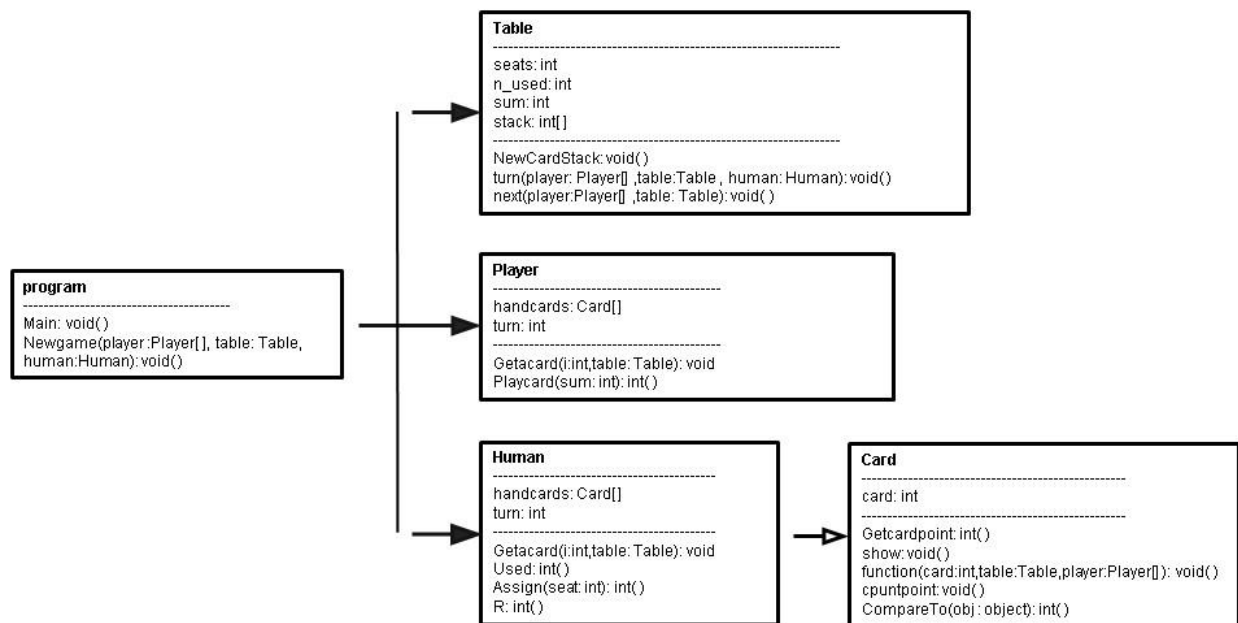
3. Player：

- a. 成員物件陣列 handcards：五個，代表手牌。
- b. 成員變數 turn：該回合的出牌順序，Table.next 改的就是這個。
- c. 成員函式 GetaCard：新宣告手牌物件，指派給 handcards[]。
- d. 成員函式 Playcard：考慮當下排堆總和，決定出哪張手牌。

4. Human：繼承 player, Used()，內有三個傳值函式

- a. Used 玩家選擇出哪一張牌，傳給 Table.turn()。
- b. Assign 玩家選擇指定哪位玩家，傳給 Card.function()。
- c. R 玩家選擇加減 10 或 20，傳給 Card.function()。

最後介紹函式 **Newgame**。這是為了建立新遊戲所寫，在主程式虛擬碼中有出現。裡面只是先洗牌，然後把所有玩家指派給已宣告的物件陣列，再來發牌，最後把牌疊總和歸零。



五、研究過程

最一開始處理牌，先寫出 `Card.show` 並測試牌的物件有文字上的輸出。接下來想要讓玩家擁有手牌，寫了 `Player.Handcards`，那自然而然地想要把牌發給玩家，但是必須先有牌堆。所以開寫新類別 `Ttable`，想說牌是放在桌面上的，就寫入成員整數陣列 `Table.stack` 代表牌堆，然後順便寫出洗牌成員函式 `Table.NewCardStack`。既然寫出了牌堆，就可以發牌給玩家，所以 `Player.GetaCard` 也被寫出來了，於此同時，`Newgame` 也被寫出來用以測試當時的程式碼是否正確。接下來要處理玩家的出牌順序以及牌的功能，這兩者是互相影響的，不太好分開處理，不過 `Player.turn` 要先宣告，然後才好處理特殊牌的功能，於是 `Card.function` 完成，其中為了方便起見，也寫出了 `Table.next` 給 `Card.function` 參考。最後把這些步驟捆成函式 `Table.turn`，至此遊戲的架構大致完成。

但是還有許多不足的地方，主要有三個：

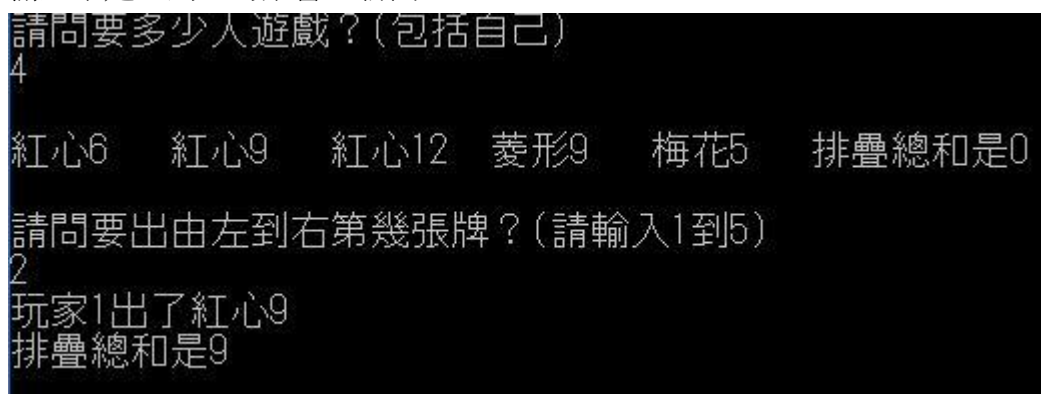
1. 當 52 張牌發完了還沒有玩家輸，程式會卡住。
2. 程式只有電腦玩家，根本稱不上是一個遊戲。

3. 電腦玩家隨機出牌，常常自爆沒有對戰的刺激。

關於第一點，我們在玩家抽牌時判斷牌疊是否空了，如果是，會發一副新的牌以及重計抽排數，順利解決。至於第二點，我們新寫了一個類別 Human 代表人類玩家，三個上述介紹的傳值函式傳給 Card.function 也順利解決。最後第三點，我們取得玩家手排的點數大小並對其排序，然後出不讓爆掉的最大那張牌，也順利達成。於是程式碼達到了我們的目標。

六、實作結果與討論

一開始會先詢問要多少人遊玩，基本上會讓玩家先出，所以先詢問玩家要出哪張，輸入不是 1 到 5 的話會重新問。



```
請問要多少人遊戲?(包括自己)
4
紅心6 紅心9 紅心12 菱形9 梅花5 排疊總和是0
請問要出由左到右第幾張牌?(請輸入1到5)
2
玩家1出了紅心9
排疊總和是9
```

接下來介紹各種功能牌，如下圖，輸入遊戲人數後，我們人類玩家就是玩家 1，之後會依照剛開始的出牌順序給予電腦玩家名，比如下一位電腦玩家叫做玩家 2，但是它被指定的作用跳過了。再觀察玩家 3 出的 Q，我們有刻意讓牌堆不會有負數出現，至於 K 的功能也在下圖展現。

請問要多少人遊戲？(包括自己)
4
紅心2 菱形5 菱形3 黑桃8 黑桃12 排疊總和是0
請問要出由左到右第幾張牌？(請輸入1到5)
6
請問要出由左到右第幾張牌？(請輸入1到5)
2
玩家1出了菱形5 請問要指定?位玩家？(數字編號)
3
指定玩家3
排疊總和是0
玩家3出了菱形12
排疊總和是20
玩家4出了菱形13 九九
紅心2 梅花10 菱形3 黑桃8 黑桃12 排疊總和是99
請問要出由左到右第幾張牌？(請輸入1到5)

Handwritten annotations:
A red dashed arrow points from the prompt "請問要指定?位玩家？(數字編號)" to the input "3".
A red sequence "1 → 3 → 4 → 1" is written in the center.
A red arrow points from the input "九九" to the "排疊總和是99".

下圖是剩下尚未介紹的功能牌，像是 Q 從玩家手中出會詢問要加或減，還有歸零、pass、迴轉等。

玩家4出了菱形13 九九
紅心2 梅花10 菱形3 黑桃8 黑桃12 排疊總和是99
請問要出由左到右第幾張牌？(請輸入1到5)
5
玩家1出了黑桃12 請問要加(輸入p)？還是減(輸入m)
m
排疊總和是79
玩家2出了黑桃1 歸零發動
排疊總和是0
玩家3出了黑桃11 pass
排疊總和是0
玩家4出了紅心4 迴轉發動
排疊總和是0
玩家3出了菱形4 迴轉發動
排疊總和是0
玩家4出了黑桃13 九九
紅心2 梅花10 菱形3 黑桃8 黑桃6 排疊總和是99
請問要出由左到右第幾張牌？(請輸入1到5)

Handwritten annotations:
A red arrow points from the "99" in the first line to the "排疊總和是79".
A red arrow points from the "歸零發動" text to the "排疊總和是0".
A red arrow points from the "迴轉發動" text to the "排疊總和是0".
A red arrow points from the "迴轉發動" text to the "排疊總和是0".
A red sequence "4 → 3 → 4 → 1" is written at the bottom right.

最後是報調會結束遊戲。

```

黑桃7  菱形6  紅心4  紅心12  梅花6  排疊總和是89
請問要出由左到右第幾張牌？(請輸入1到5)
4
玩家1出了紅心12 請問要加(輸入p)？還是減(輸入m)
p
GG

```

盡管此版本的遊戲已經達成了撲克牌遊戲 99 的規則，我們認為此遊戲仍然還有幾個可以發展的方向，例如結合圖形化介面，新增可以讓人類玩家與人類玩家對戰的功能，對電腦出牌加時間延遲(模擬真人)，以目前的版本基本可以達成大部分遊戲的必須條件(最大化出牌效益)。

七、參考資料

課程講義：以 C#學程式設計基礎

八、程式列表

```

using System;

namespace final
{
    class Program
    {
        static void Main(string[] args) //玩家、洗牌、
電腦
    {
        Console.WriteLine("請問要多少人遊戲？(包括自己)");
        int seat = int.Parse(Console.ReadLine());
        Table table = new Table(seat);
        Player[] player = new Player[seat];

        Human human = new Human(); //改
        int i;
        Newgame(player, table, human); //改
    }
}

```

```

        if (player[0].Turn == 0)
        {
            //輪到玩家 0 出牌才顯示他的牌
            Console.WriteLine();
            for (i = 0; i < 5; i++)
            {
                player[0].Handcards[i].Show();
            }
        }

        Console.WriteLine("排疊總和是" + table.Sum);
        Console.WriteLine();
        table.turn(player, table, human);
        //Console.ReadLine();
    } while (table.Sum <= 99);
    Console.WriteLine("GG");
    Console.ReadLine();
}

static void Newgame(Player[] player, Table table, Human human)
{
    //新遊戲
    //改

    int s = table.Seats;
    int i, j;
    table.NewCardStack();
    for (i = 0; i < s; i++)
    {
        //改

        if (i == 0)
        {
            player[i] = human;
        }
        else
        {
            player[i] = new Player();
            //建構玩
            家
        }

        player[i].Turn = i;
        for (j = 0; j < 5; j++)
        {
            //發牌

```

```

        player[i].Handcards[j] = new Card(0);    //牌也要一張張
        player[i].Getacard(j, table);
    }
}

    table.Sum = 0;
}
}

}

```

```

using System;

namespace final
{
    class Card:IComparable
    {
        private int card;    // 0 到 51
        private int point;
        public Card(int card)
        {
            this.card = card;
        }
        public int Getcard
        {
            set { card = value; }
            get { return card; }
        }
        public int Getcardpoint
        {
            get { return card; }
        }
    }
}

```

```

public void Show()
{
    //牌輸出
    if (card / 13 == 0)
    {
        Console.Write("黑桃" + (card % 13 + 1) + "\t");
        //0 到12
    }
    else if (card / 13 == 1)
    {
        Console.Write("紅心" + (card % 13 + 1) + "\t");
        //13 到25
    }
    else if (card / 13 == 2)
    {
        Console.Write("菱形" + (card % 13 + 1) + "\t");
        //26 到38
    }
    else
    {
        Console.Write("梅花" + (card % 13 + 1) + "\t");
        //39 到51
    }
}

public void function(int card, Table table, Player[]
player, Human human)
{
    //每張牌的作用
    Random rand = new Random();
    int r = (rand.Next() % 2) * 2 - 1;
    if (table.Sum < 20)
    {
        //+-10 或20 不能讓值出現負號
        r = 1;
    }
    if (table.Sum > 79)
    {
        r = -1;
    }
}

```

```

    }

    int i;
    if (card == 0)
    {
        //黑桃A 歸零
        table.Sum = 0;
        table.next(player, table);
        Console.WriteLine("歸零發動");
    }
    else if (card % 13 == 3)
    {
        //迴轉
        for (i = 0; i < table.Seats; i++)
        {
            player[i].Turn = player[i].Turn * (-1) +
table.Seats;
        }
        table.next(player, table);
        Console.WriteLine("迴轉發動");
    }
    else if (card % 13 == 4)
    {
        int assign;
        //
        if (player[0].Turn == 0)
        {
            assign = player[human.Assign(table.Seats) - 1].Turn;
        }
        else
        {
            assign = player[(rand.Next() % table.Seats)].Turn;
        }
        int p = 0;
        for (i = 0; i < table.Seats; i++)
        {
            player[i].Turn += -assign;
            if (player[i].Turn == 0)
            {

```

```

        p = i;
    }
    if (player[i].Turn < 0)
    {
        player[i].Turn += table.Seats;
    }
}

Console.WriteLine("指定玩家" + (p + 1));
}

else if (card % 13 == 9)
{
    //+-10
    if (player[0].Turn == 0 & table.Sum > 10)
    {
        //改
        r = human.R();
    }
    table.next(player, table);
    table.Sum += (10 * r);
    Console.WriteLine();
}

else if (card % 13 == 10)
{
    //pass
    table.next(player, table);
    Console.WriteLine("pass");
}

else if (card % 13 == 11)
{
    //+-20
    if (player[0].Turn == 0 & table.Sum > 20)
    {
        //改
        r = human.R();
    }
    table.next(player, table);
    table.Sum += (20 * r);
    Console.WriteLine();
}

else if (card % 13 == 12)
{
    //99
    table.Sum = 99;
}

```

```

        table.next(player, table);
        Console.WriteLine("九九");
    }
    else
    {
        //其他加法
        table.Sum += (card % 13 + 1);
        table.next(player, table);
        Console.WriteLine();
    }
}

public void cpuntpoint()
{
    if (this.card == 0 | this.card % 13 == 3 | this.card % 13
    == 4 | this.card % 13 == 9 | this.card % 13 == 10 | this.card % 13 ==
    11 | this.card % 13 == 12) this.point = 0;
    else this.point = (this.card % 13) + 1;

}

public int CompareTo(object obj)
{
    Card tobeCompared = (Card)obj;
    if (this.point > tobeCompared.point)
        return 1;
    else if (this.point == tobeCompared.point)
        return 0;
    else return -1;
}

}

}
}

```

```
using System;
```



```
namespace final
{

    class Human : Player
    {
        //
        public int Used()
        {
            int used;
            do
            {
                Console.WriteLine("請問要出由左到右第幾張牌？(請輸入 1 到 5)");
                used = int.Parse(Console.ReadLine());
            } while (used >= 6);
            return used;
        }

        public int Assign(int seat)
        {
            int assign;
            do
            {
                Console.WriteLine("請問要指定哪位玩家？(數字編號)");
                assign = int.Parse(Console.ReadLine());
            } while (assign > seat & assign != 1);
            return assign;
        }

        public int R()
        {
            char a;
            int r = 0;
            do
            {
                Console.WriteLine("請問要加(輸入 p)？還是減(輸入 m)");
                a = Console.ReadLine().ToCharArray()[0];
                switch (a)
                {
```

```

        case 'p':
            r = 1;
            break;
        case 'm':
            r = -1;
            break;
        default:
            Console.WriteLine("請輸入 p 或 m");
            r = 0;
            break;
    }
} while (r == 0);
return r;
}
}
}

```

```
using System;
```

```
namespace final
```

```

{
    class Player
    {
        private Card[] handcards = new Card[5]; //手牌
        public Card[] Handcards
        {
            set { handcards = value; }
            get { return handcards; }
        }

        private int turn; //出牌順
        public int Turn
        {
            set { turn = value; }
            get { return turn; }
        }
    }
}

```

```

    }

    public void Getacard(int i, Table table)
    {
        //玩家取牌

        Card h = new Card(table.Stack[table.N_Used]); //牌一張張建構//這行出事代表用完一副牌了，再洗牌還沒寫(牌疊的 index 超過 51)

        table.N_Used++;
        handcards[i] = h;
        if (table.N_Used == 52)
        {
            //如果卡組抽完則重新洗牌繼續抽

            table.NewCardStack();
            table.N_Used = 0;
        }

    }

    public int Playcard(int sum) //電腦出牌 輸入現在總點數
    {
        int end = 0; //預設出第一張
        for (int i = 0; i <= 4; ++i)
        {
            this.handcards[i].cpuntpoint();
        }
        Array.Sort(this.handcards);
        for (int i = 4; i >= 0; --i) //計算點數如果有個好的出法則
            換出最好那張
        {
            if ((this.handcards[i].Getcardpoint + sum) < 99) end
            = i; break;
        }
        return end;
    }

}

```

```
using System;

namespace final
{

    class Table
    {
        private int seats;           //玩家數
        private int n_used = 0;      //從取牌堆拿了多少牌
        private int sum;             //丟出的牌值總和
        public Table(int seats)
        {
            this.seats = seats;
        }
        public int Seats
        {
            get { return seats; }
        }
        public int Sum
        {
            set { sum = value; }
            get { return sum; }
        }
        public int N_Used
        {
            set { n_used = value; }
            get { return n_used; }
        }
        private int[] stack = new int[52]; //取牌堆
        public int[] Stack
        {
            get { return stack; }
        }
    }
}
```

```

public void NewCardStack()
{
    //洗牌
    int i, j, k;
    int l;

    int[] space = new int[52];
    bool unfound;
    Random rand = new Random();
    for (i = 0; i < 52; i++)
    {
        space[i] = i; //spaceu 有 0 到 51 ,
        準備指派給 stack
    }
    for (j = 0; j < 52; j++)
    {
        do
        {
            unfound = true;
            l = rand.Next() % 52;
            for (k = 0; k < 52; k++)
            {
                if (l == space[k])
                {
                    //從 space 找還沒指派的值(牌) , 取出
                    space[k] = 100;
                    unfound = false;
                    break;
                }
            }
        } while (unfound);
        stack[j] = l; //給取牌堆 stack
    }
}

public void turn(Player[] player, Table table, Human human)
{
    //處理玩家出牌會發生甚麼(牌功能+順序) //改
    int i;
    int n = 0;
    if (player[0].Turn == 0)

```

```

        { //改 n
            n = human.Used() - 1;
        }
        for (i = 0; i < seats; i++)
        {
            if (player[i].Turn == 0)
            {
                if (i > 0) n = player[i].Playcard(table.Sum);
                Console.Write("玩家" + (i + 1) + "出了");
                player[i].Handcards[n].Show();

                player[i].Handcards[n].function(player[i].Handcards[n].Getcard,
table, player, human);
                player[i].Getacard(n, table);
                break;
            }
        }
    }

    public void next(Player[] player, Table table)
    { //沒有指定、迴轉情況下，出牌順序順延
        int i;
        for (i = 0; i < table.Seats; i++)
        {
            player[i].Turn += -1;
            if (player[i].Turn == -1)
            { //剛出完，順序跑到最後
                player[i].Turn += table.Seats;
            }
        }
    }
}

```

