物件導向小專題報告

撲克牌-21點

目錄

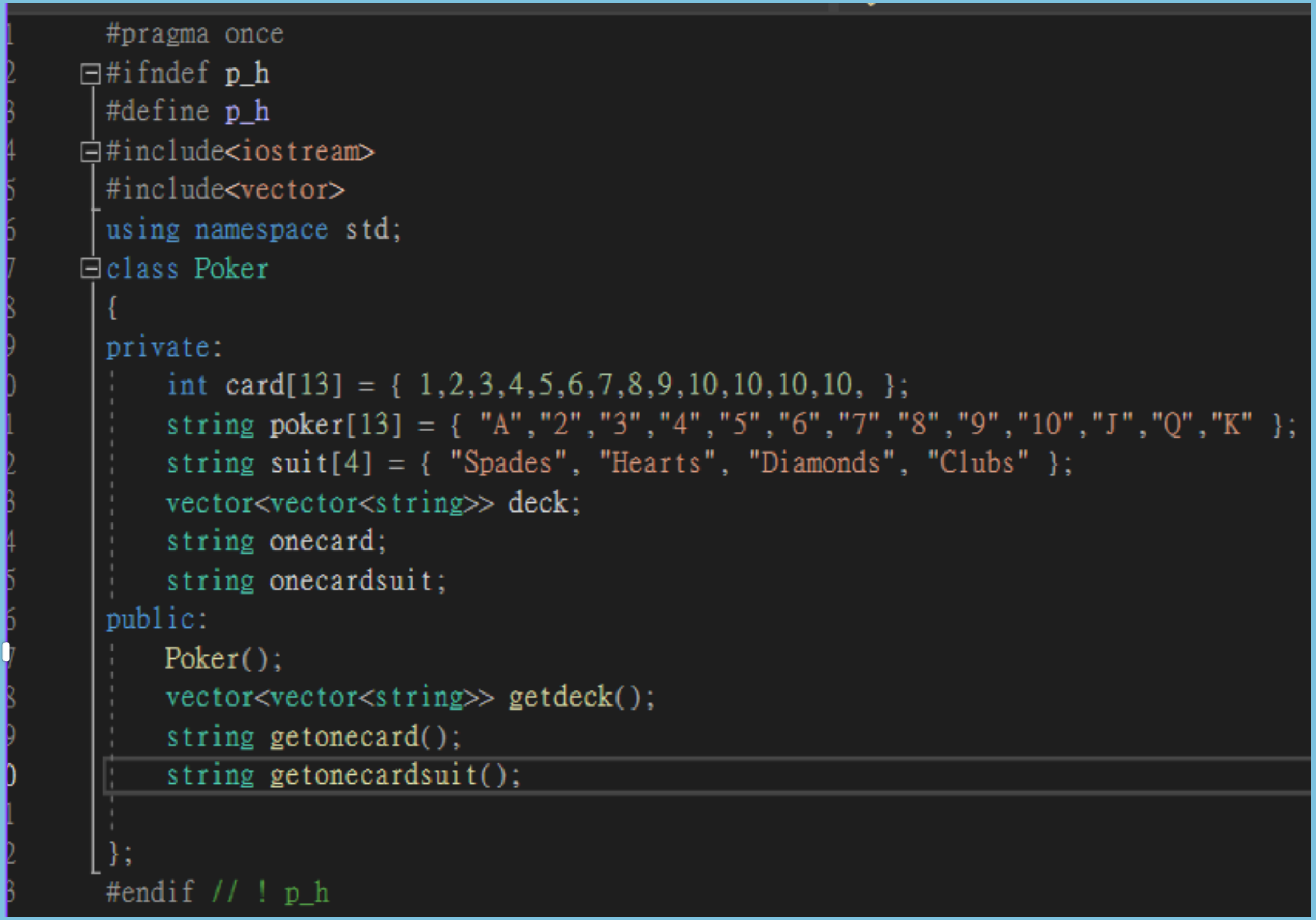
遊戲玩法

擁有最高點數的玩家獲勝，其點數必須等於或低於21點；超過21點的玩家稱為爆牌。2點至10點的牌以牌面的點數計算，J、Q、K 每張為10點，A 可當11或是1點。

遊戲流程

荷官開始先發兩張牌，按照順序玩家可以選要牌或不要，如果點數超過21點玩家輸錢，如若未超過21點玩家與荷官比大小，荷官自行判定自己要不要補牌，如超過21點則玩家贏，如未超過21點則荷關與玩家比大小。

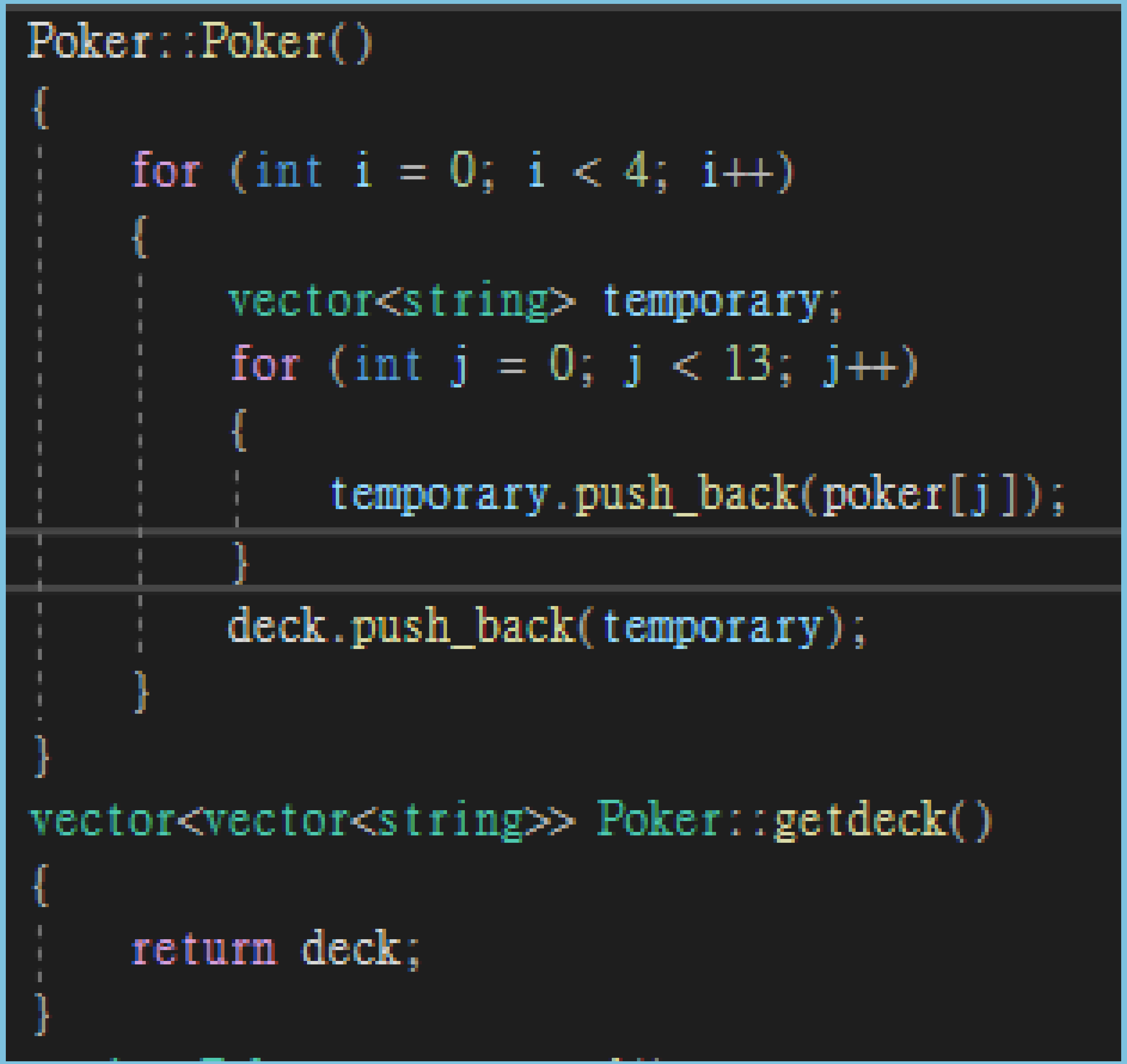
程式展示 撲克牌參數 撲克牌.h



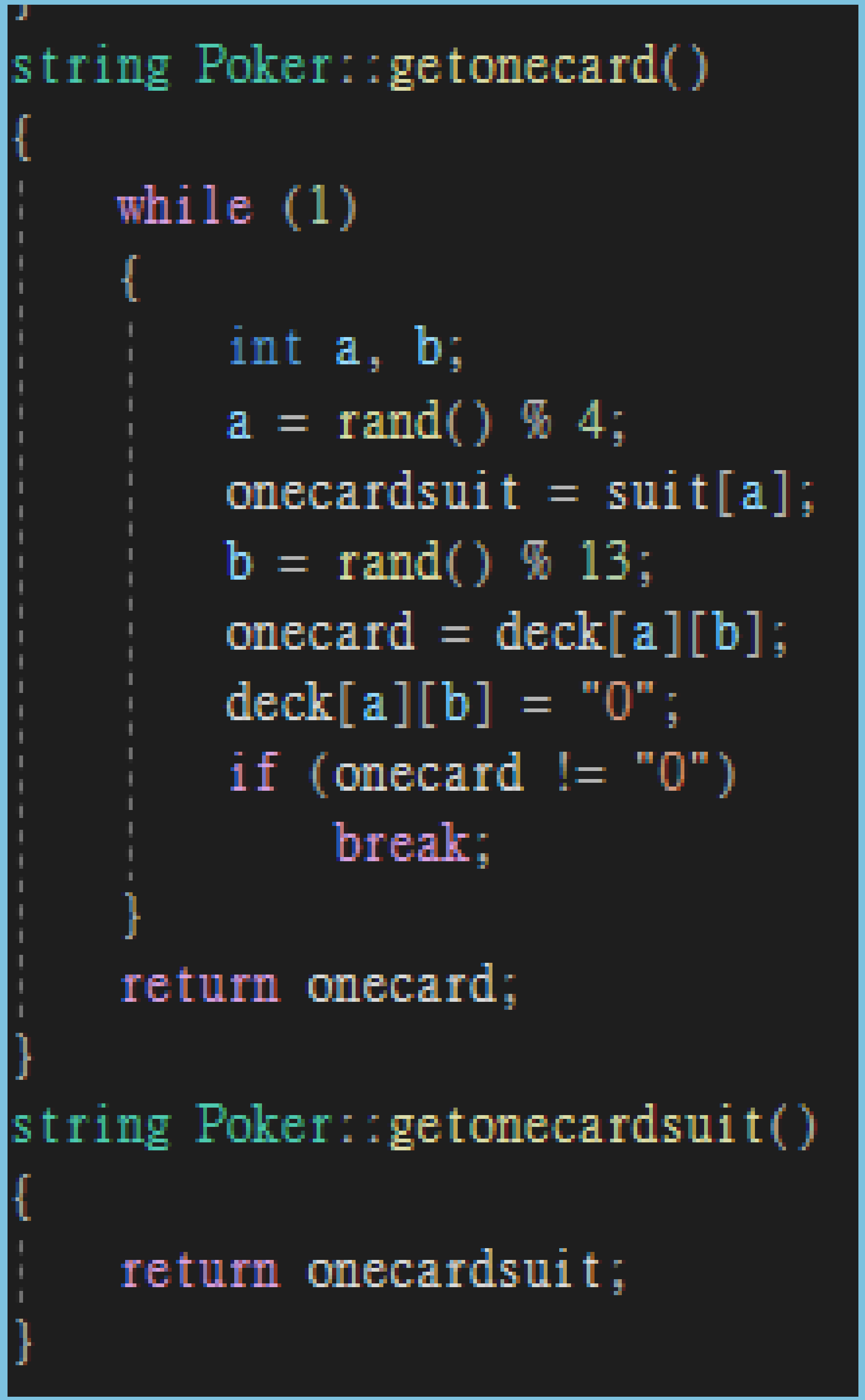
```
#pragma once
#ifndef p_h
#define p_h
#include<iostream>
#include<vector>
using namespace std;
class Poker
{
private:
    int card[13] = { 1,2,3,4,5,6,7,8,9,10,10,10,10, };
    string poker[13] = { "A","2","3","4","5","6","7","8","9","10","J","Q","K" };
    string suit[4] = { "Spades", "Hearts", "Diamonds", "Clubs" };
    vector<vector<string>> deck;
    string onecard;
    string onecardsuit;
public:
    Poker();
    vector<vector<string>> getdeck();
    string getonecard();
    string getonecardsuit();

};
#endif // ! p_h
```

程式展示 撲克牌參數 撲克牌.cpp

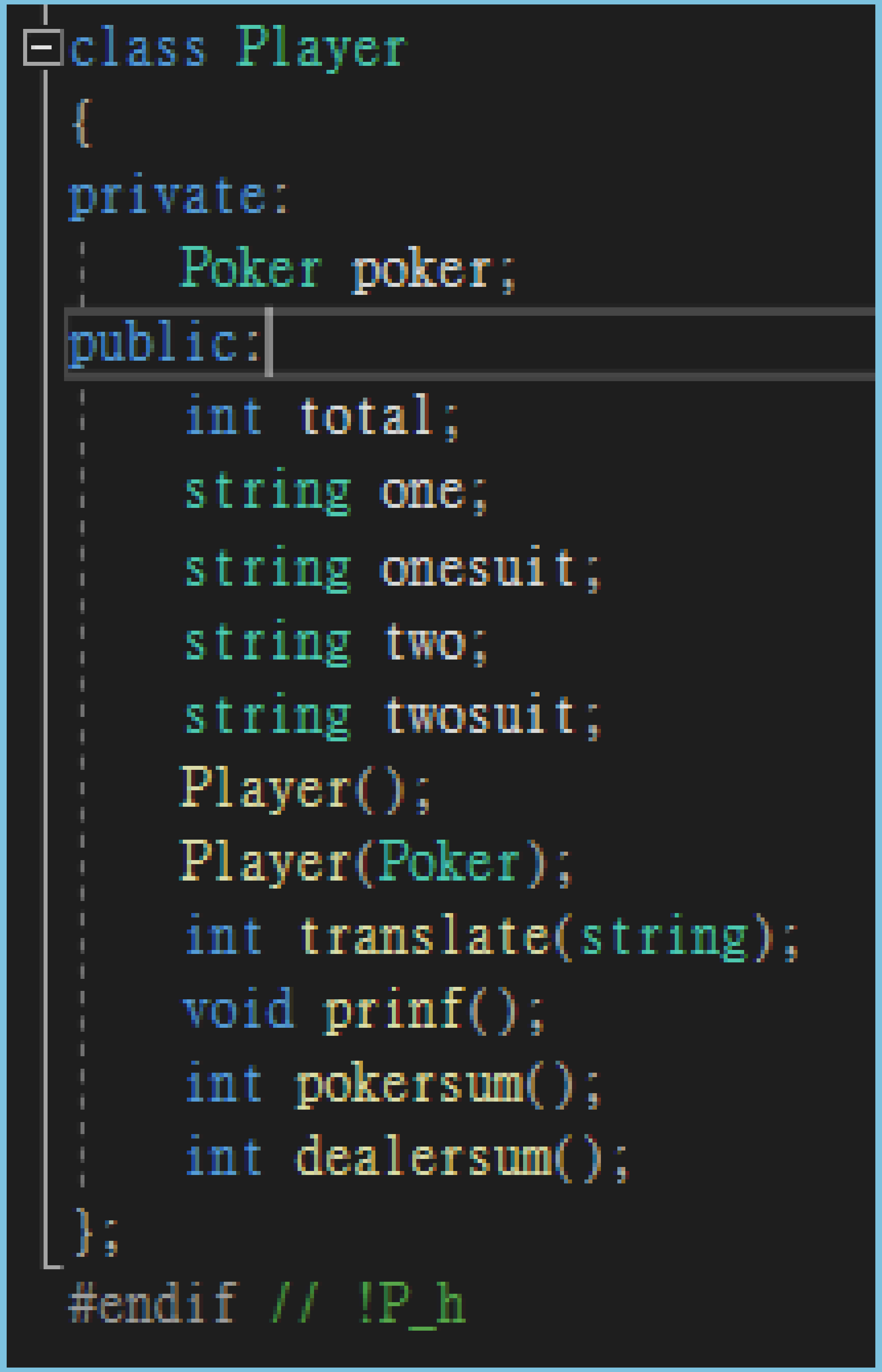


```cpp
Poker::Poker()
{
    for (int i = 0; i < 4; i++)
    {
        vector<string> temporary;
        for (int j = 0; j < 13; j++)
        {
            temporary.push_back(poker[j]);
        }
        deck.push_back(temporary);
    }
}
vector<vector<string>> Poker::getdeck()
{
    return deck;
}
```



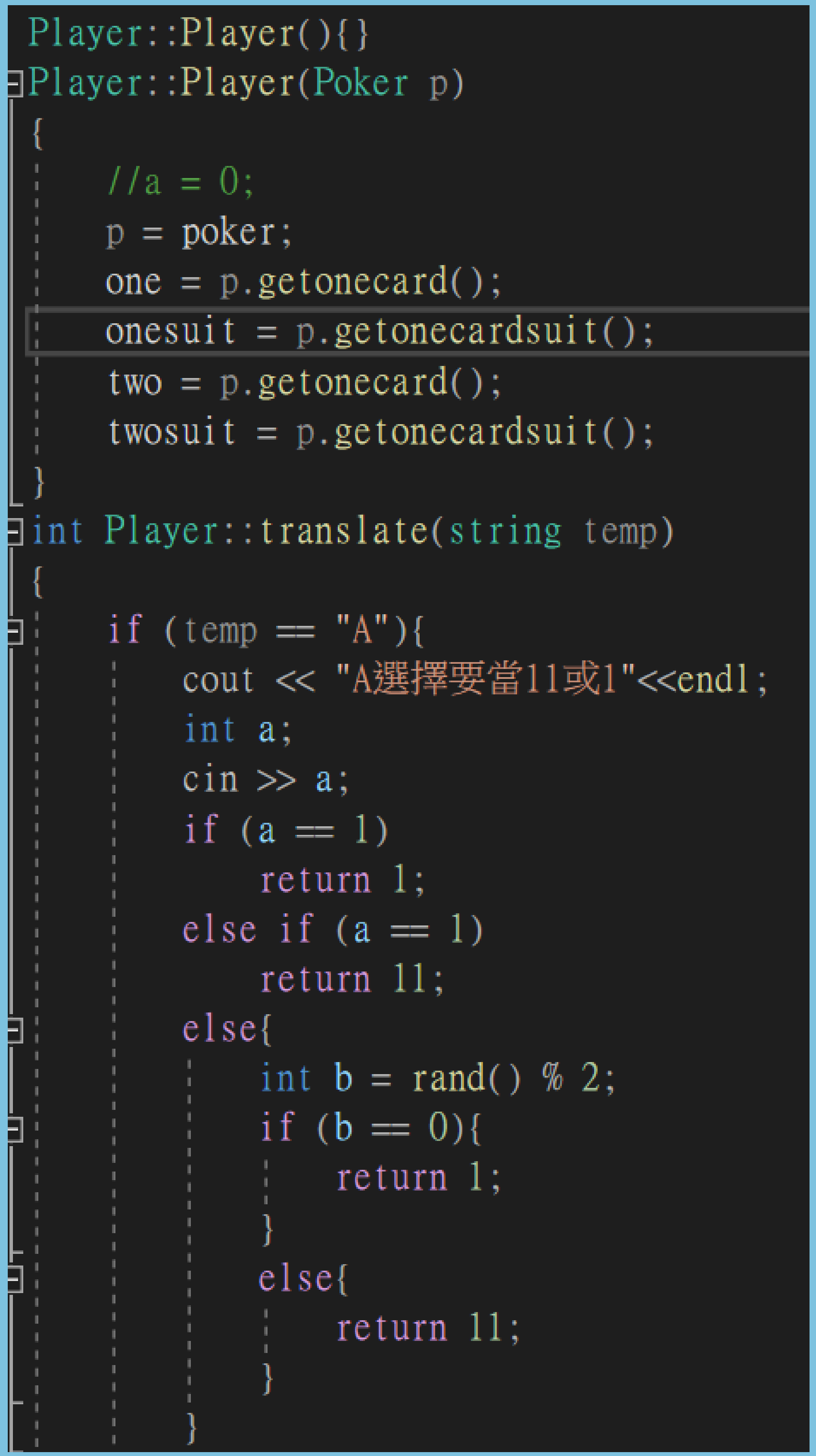
```cpp
string Poker::getonecard()
{
    while (1)
    {
        int a, b;
        a = rand() % 4;
        onecardsuit = suit[a];
        b = rand() % 13;
        onecard = deck[a][b];
        deck[a][b] = "0";
        if (onecard != "0")
            break;
    }
    return onecard;
}
string Poker::getonecardsuit()
{
    return onecardsuit;
}
```

程式展示玩家參數 玩家.h

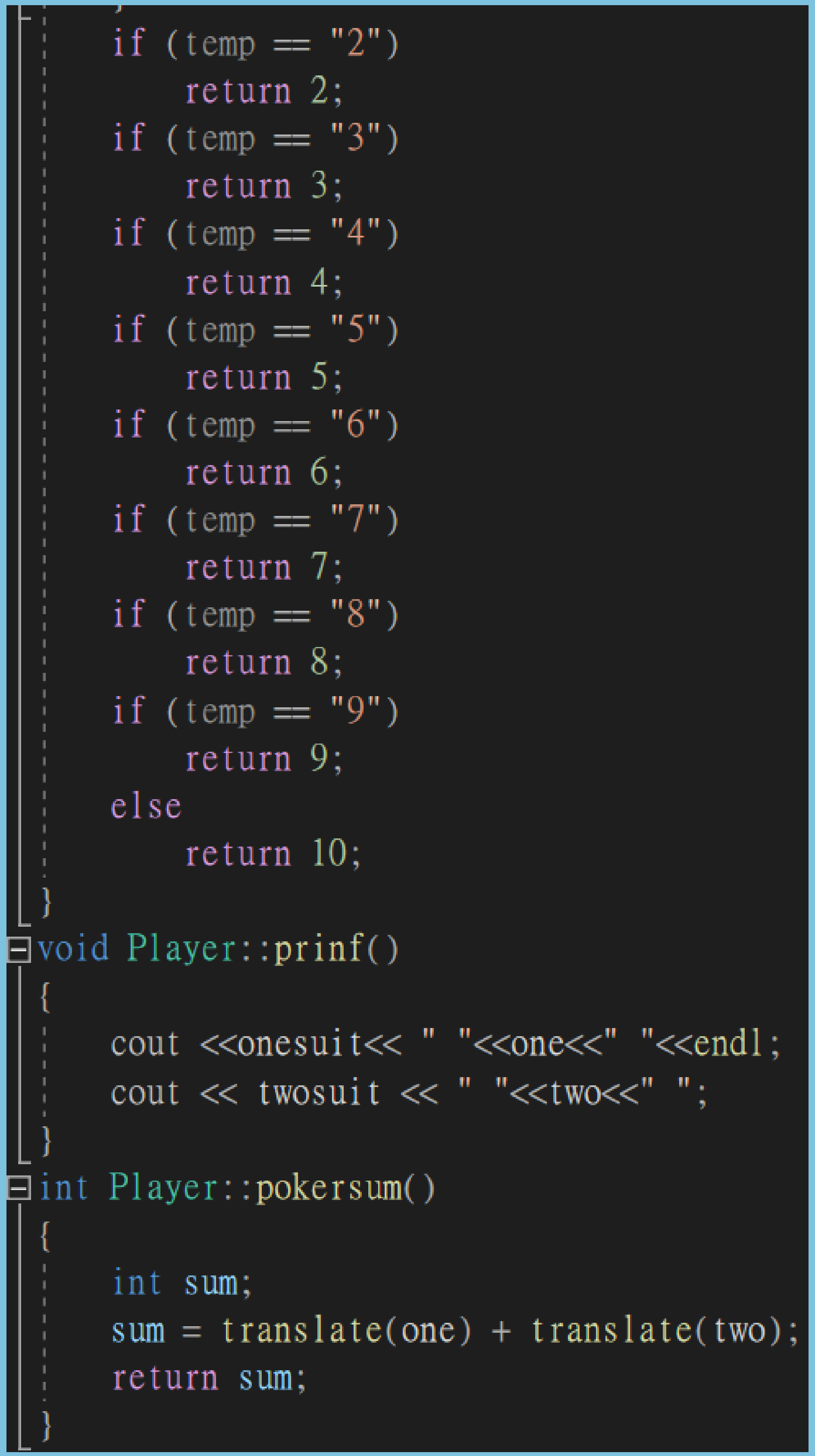


```
class Player
{
private:
	Poker poker;
public:
	int total;
	string one;
	string onesuit;
	string two;
	string twosuit;
	Player();
	Player(Poker);
	int translate(string);
	void prinf();
	int pokersum();
	int dealersum();
};
#endif // !P_h
```

程式展示玩家參數 玩家.cpp

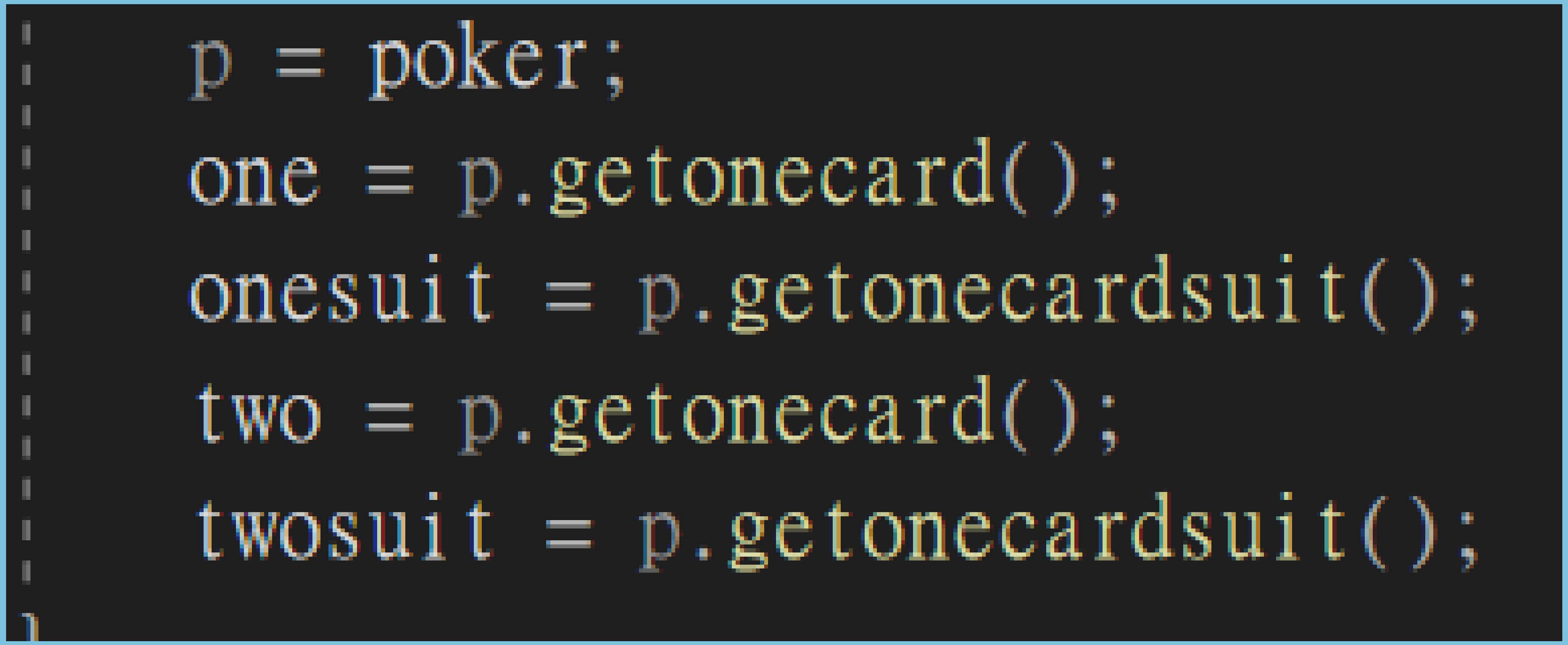


```cpp
Player::Player(){}
Player::Player(Poker p)
{
    //a = 0;
    p = poker;
    one = p.getonecard();
    onesuit = p.getonecardsuit();
    two = p.getonecard();
    twosuit = p.getonecardsuit();
}
int Player::translate(string temp)
{
    if (temp == "A"){
        cout << "A選擇要當11或1"<<endl;
        int a;
        cin >> a;
        if (a == 1)
            return 1;
        else if (a == 1)
            return 11;
        else{
            int b = rand() % 2;
            if (b == 0){
                return 1;
            }
            else{
                return 11;
            }
        }
```



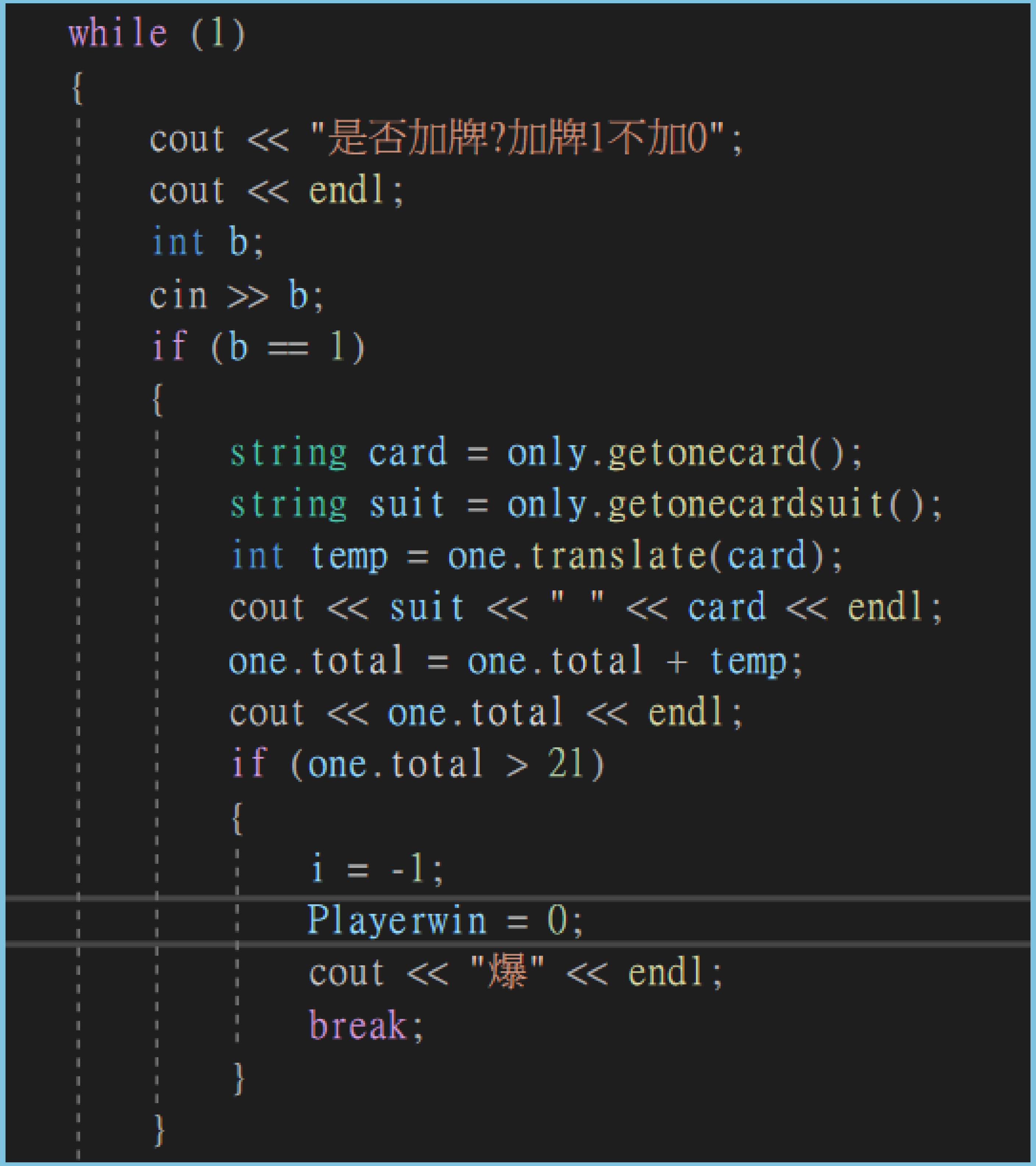
```cpp
    if (temp == "2")
        return 2;
    if (temp == "3")
        return 3;
    if (temp == "4")
        return 4;
    if (temp == "5")
        return 5;
    if (temp == "6")
        return 6;
    if (temp == "7")
        return 7;
    if (temp == "8")
        return 8;
    if (temp == "9")
        return 9;
    else
        return 10;
}
void Player::prinf()
{
    cout <<onesuit<< " "<<one<<" "<<endl;
    cout << twosuit << " "<<two<<" ";
}
int Player::pokersum()
{
    int sum;
    sum = translate(one) + translate(two);
    return sum;
}
```

程式顯示 玩家卡牌參數

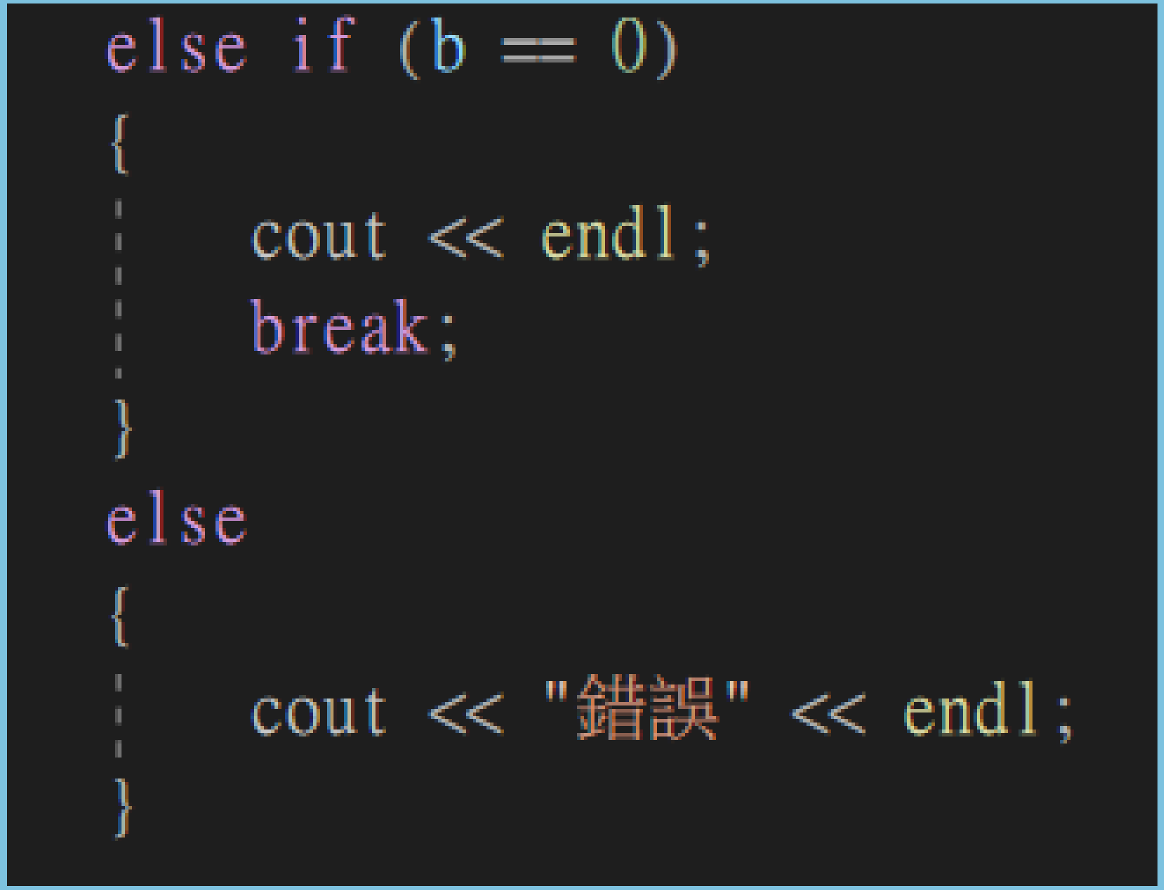


```
    p = poker;
    one = p.getonecard();
    onesuit = p.getonecardsuit();
    two = p.getonecard();
    twosuit = p.getonecardsuit();
```

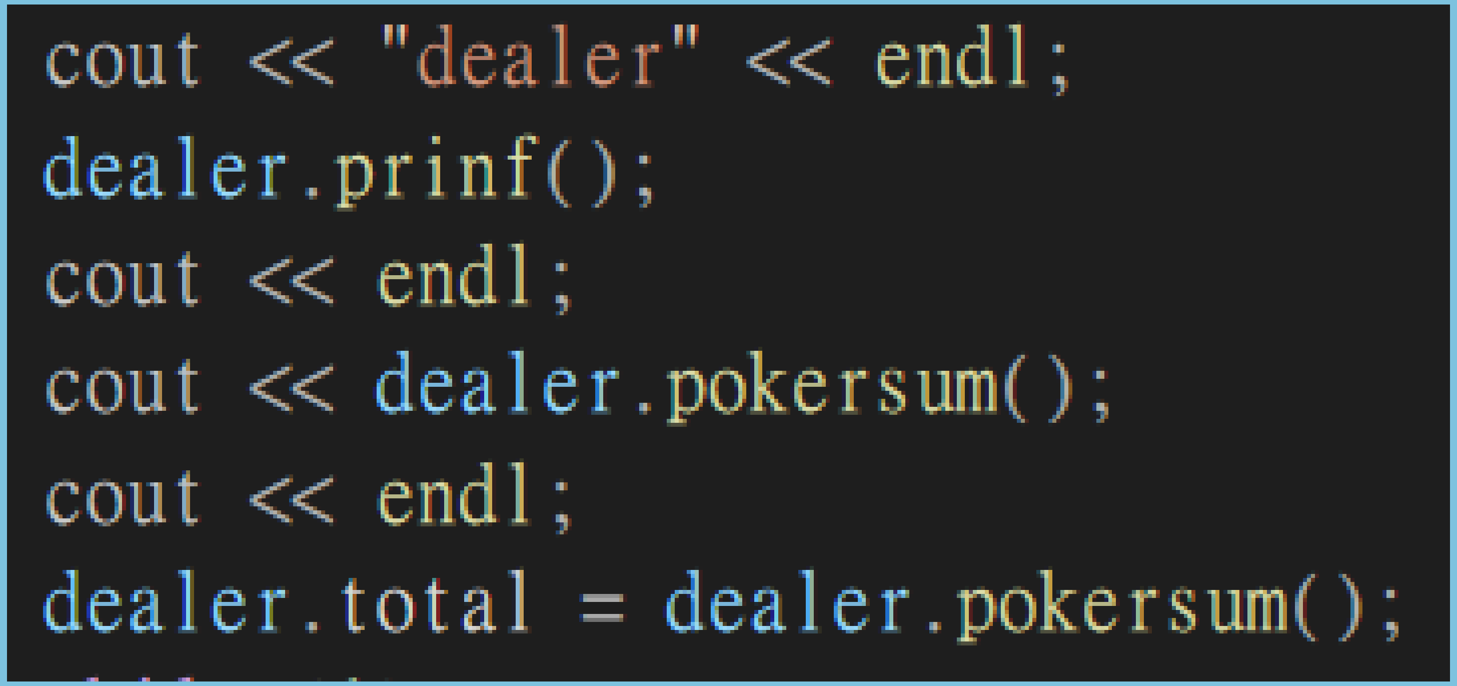
程式顯示 main.cpp



```
while (1)
{
    cout << "是否加牌?加牌1不加0";
    cout << endl;
    int b;
    cin >> b;
    if (b == 1)
    {
        string card = only.getonecard();
        string suit = only.getonecardsuit();
        int temp = one.translate(card);
        cout << suit << " " << card << endl;
        one.total = one.total + temp;
        cout << one.total << endl;
        if (one.total > 21)
        {
            i = -1;
            Playerwin = 0;
            cout << "爆" << endl;
            break;
        }
    }
```

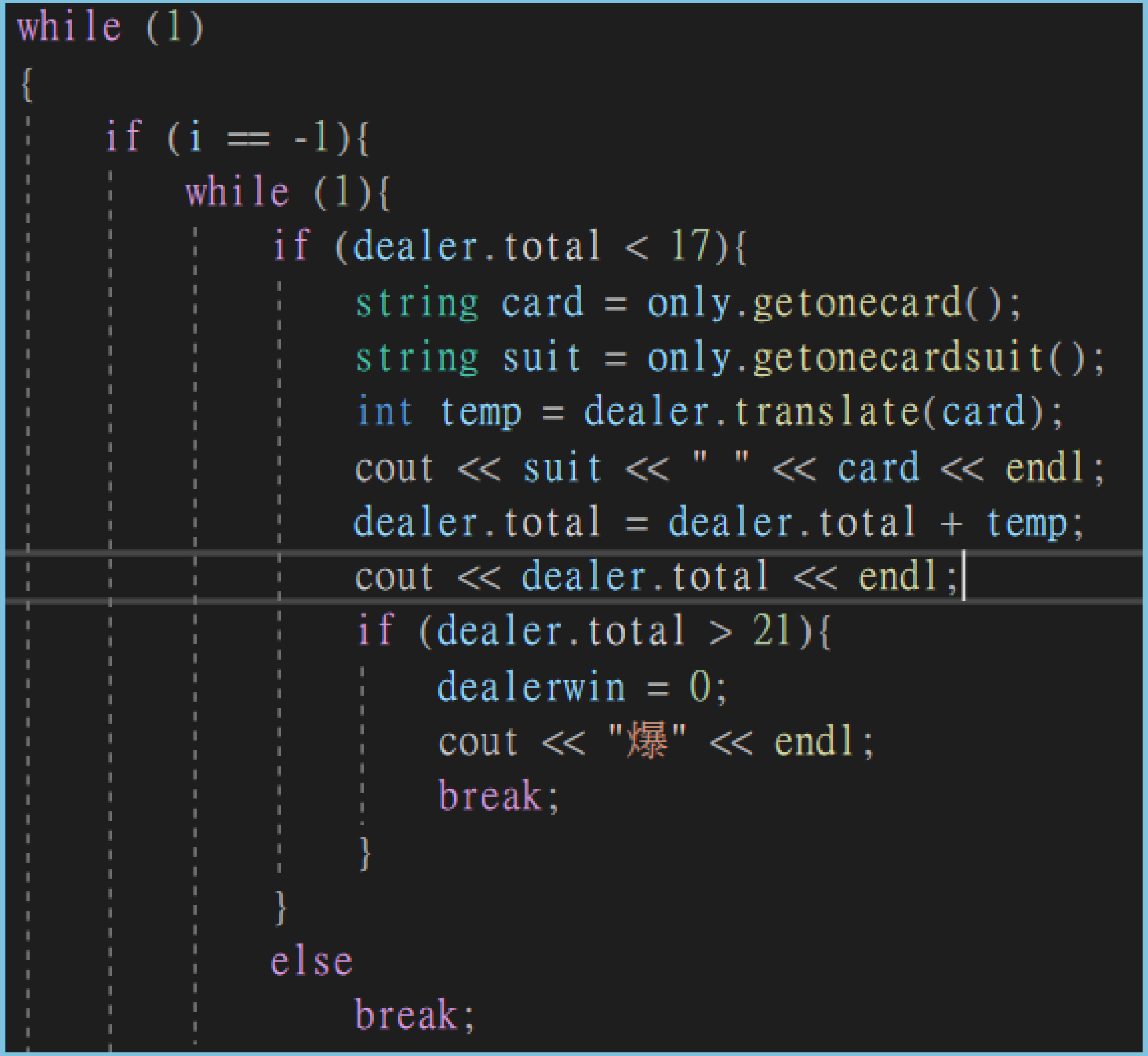


```
else if (b == 0)
{
    cout << endl;
    break;
}
else
{
    cout << "錯誤" << endl;
}
```

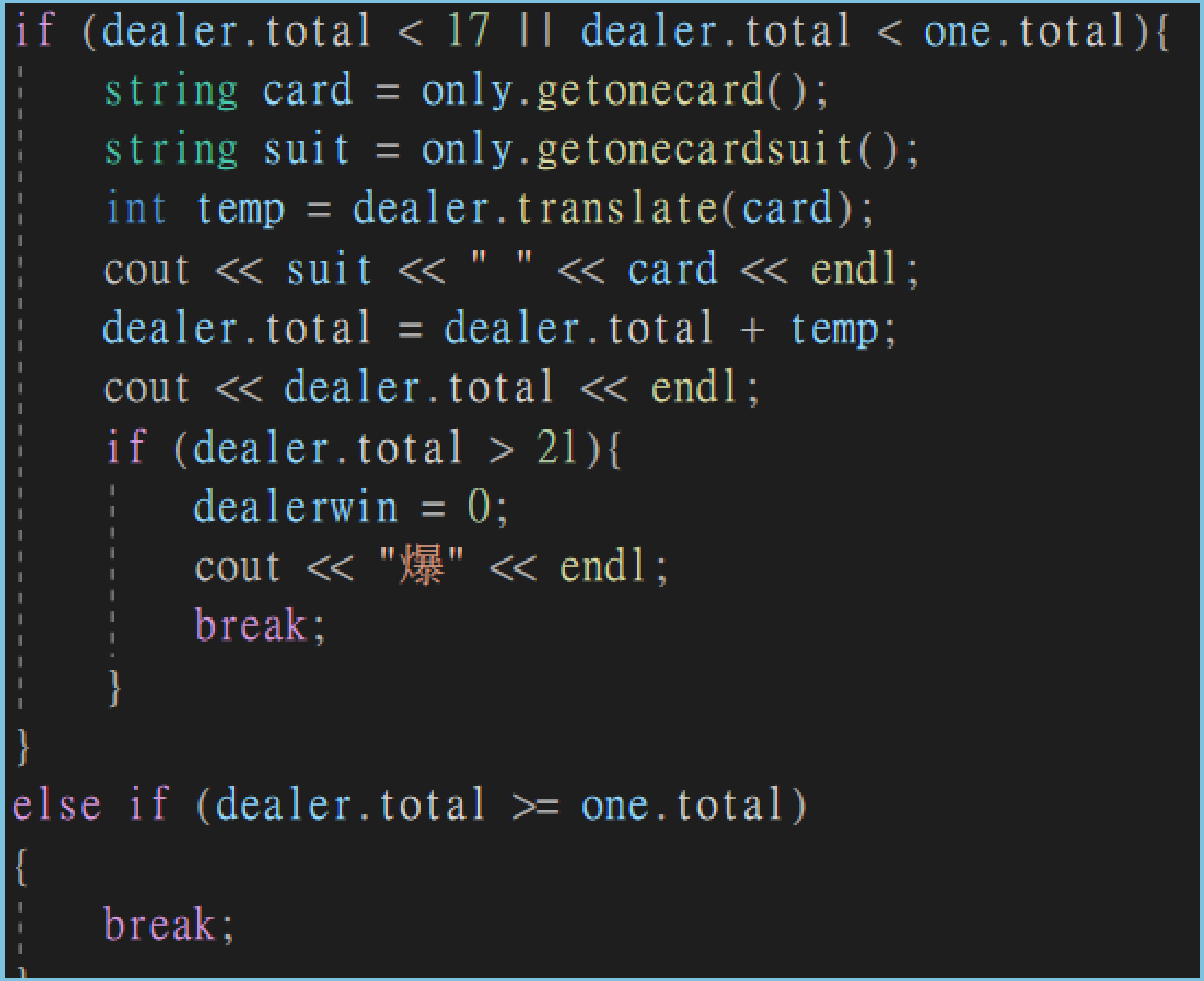


```
cout << "dealer" << endl;
dealer.prinf();
cout << endl;
cout << dealer.pokersum();
cout << endl;
dealer.total = dealer.pokersum();
```

程式顯示 main.cpp

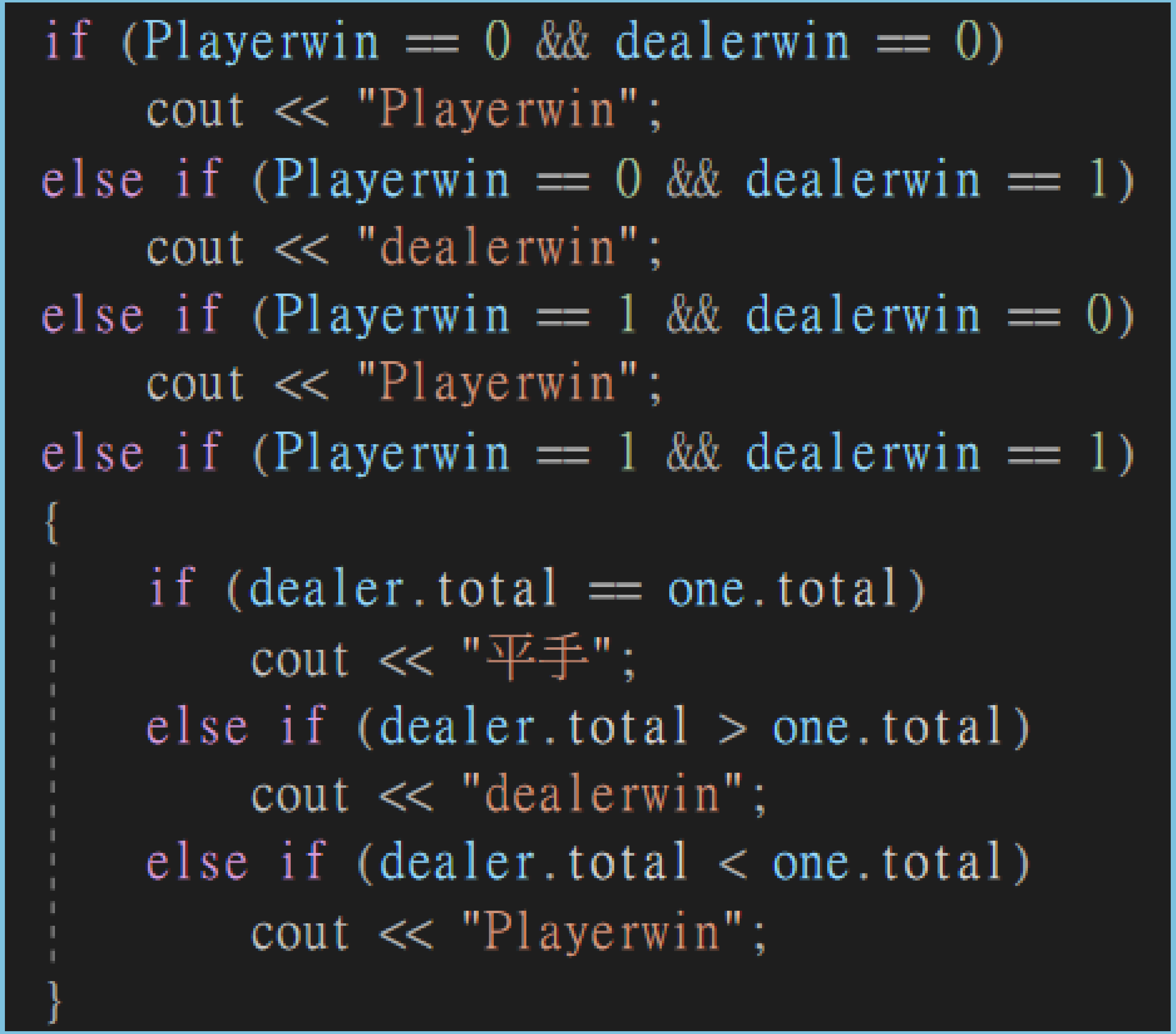


```
while (1)
{
    if (i == -1){
        while (1){
            if (dealer.total < 17){
                string card = only.getonecard();
                string suit = only.getonecardsuit();
                int temp = dealer.translate(card);
                cout << suit << " " << card << endl;
                dealer.total = dealer.total + temp;
                cout << dealer.total << endl;
                if (dealer.total > 21){
                    dealerwin = 0;
                    cout << "爆" << endl;
                    break;
                }
            }
            else
                break;
```



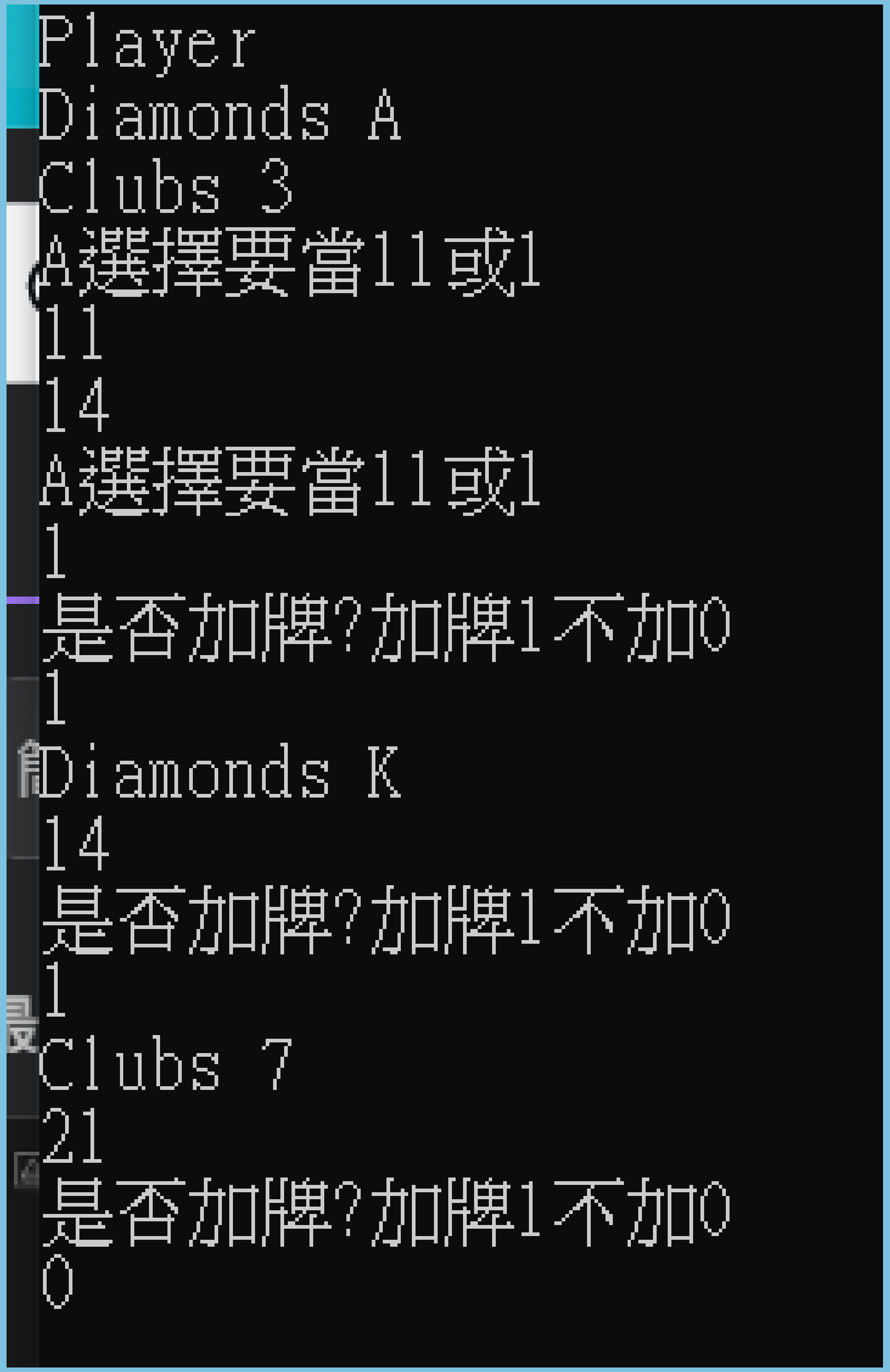
```
if (dealer.total < 17 || dealer.total < one.total){
    string card = only.getonecard();
    string suit = only.getonecardsuit();
    int temp = dealer.translate(card);
    cout << suit << " " << card << endl;
    dealer.total = dealer.total + temp;
    cout << dealer.total << endl;
    if (dealer.total > 21){
        dealerwin = 0;
        cout << "爆" << endl;
        break;
    }
}
else if (dealer.total >= one.total)
{
    break;
```

程式顯示 main.cpp

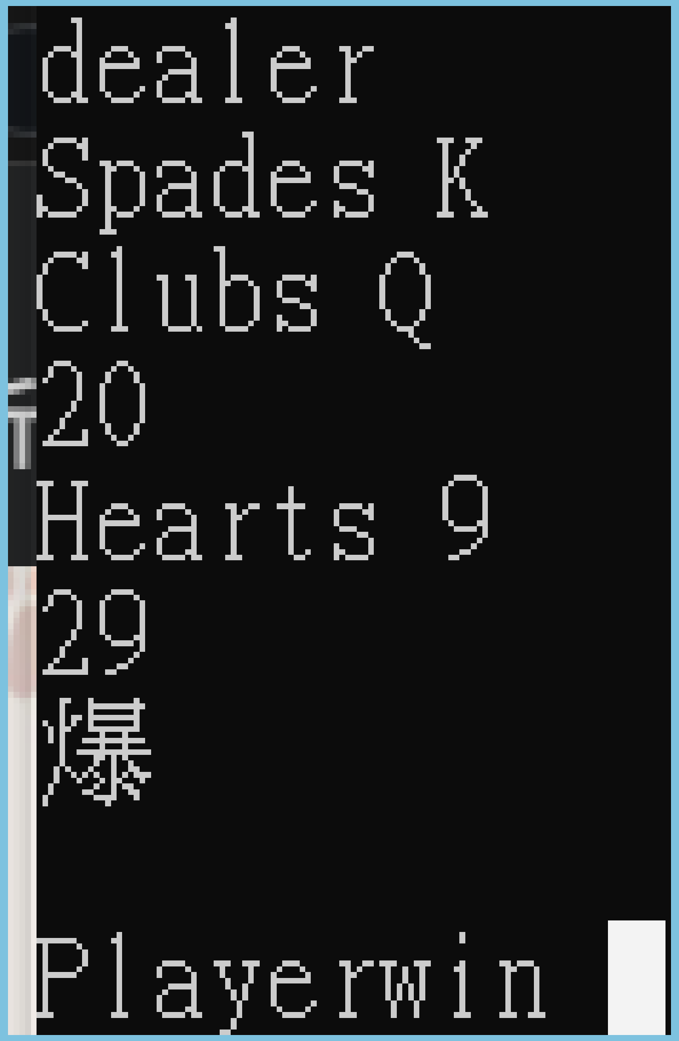


```
if (Playerwin == 0 && dealerwin == 0)
    cout << "Playerwin";
else if (Playerwin == 0 && dealerwin == 1)
    cout << "dealerwin";
else if (Playerwin == 1 && dealerwin == 0)
    cout << "Playerwin";
else if (Playerwin == 1 && dealerwin == 1)
{
    if (dealer.total == one.total)
        cout << "平手";
    else if (dealer.total > one.total)
        cout << "dealerwin";
    else if (dealer.total < one.total)
        cout << "Playerwin";
}
```

程式操作



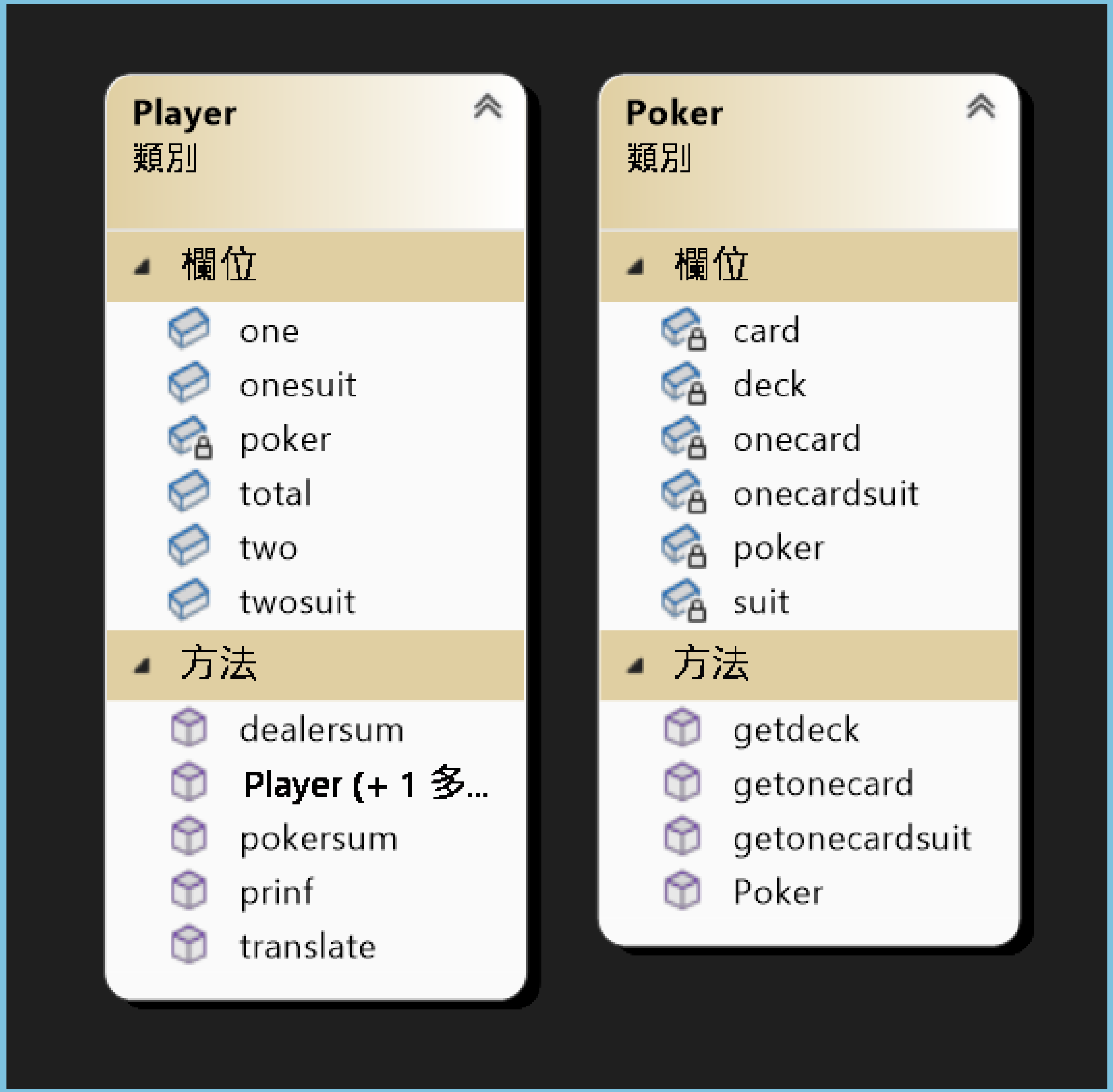
```
Player
Diamonds A
Clubs 3
A選擇要當11或1
11
14
A選擇要當11或1
1
是否加牌?加牌1不加0
1
Diamonds K
14
是否加牌?加牌1不加0
1
Clubs 7
21
是否加牌?加牌1不加0
0
```



```
dealer
Spades K
Clubs Q
20
Hearts 9
29
爆

Playerwin
```

撲克牌UML



Player
類別

欄位
- one
- onesuit
- poker
- total
- two
- twosuit

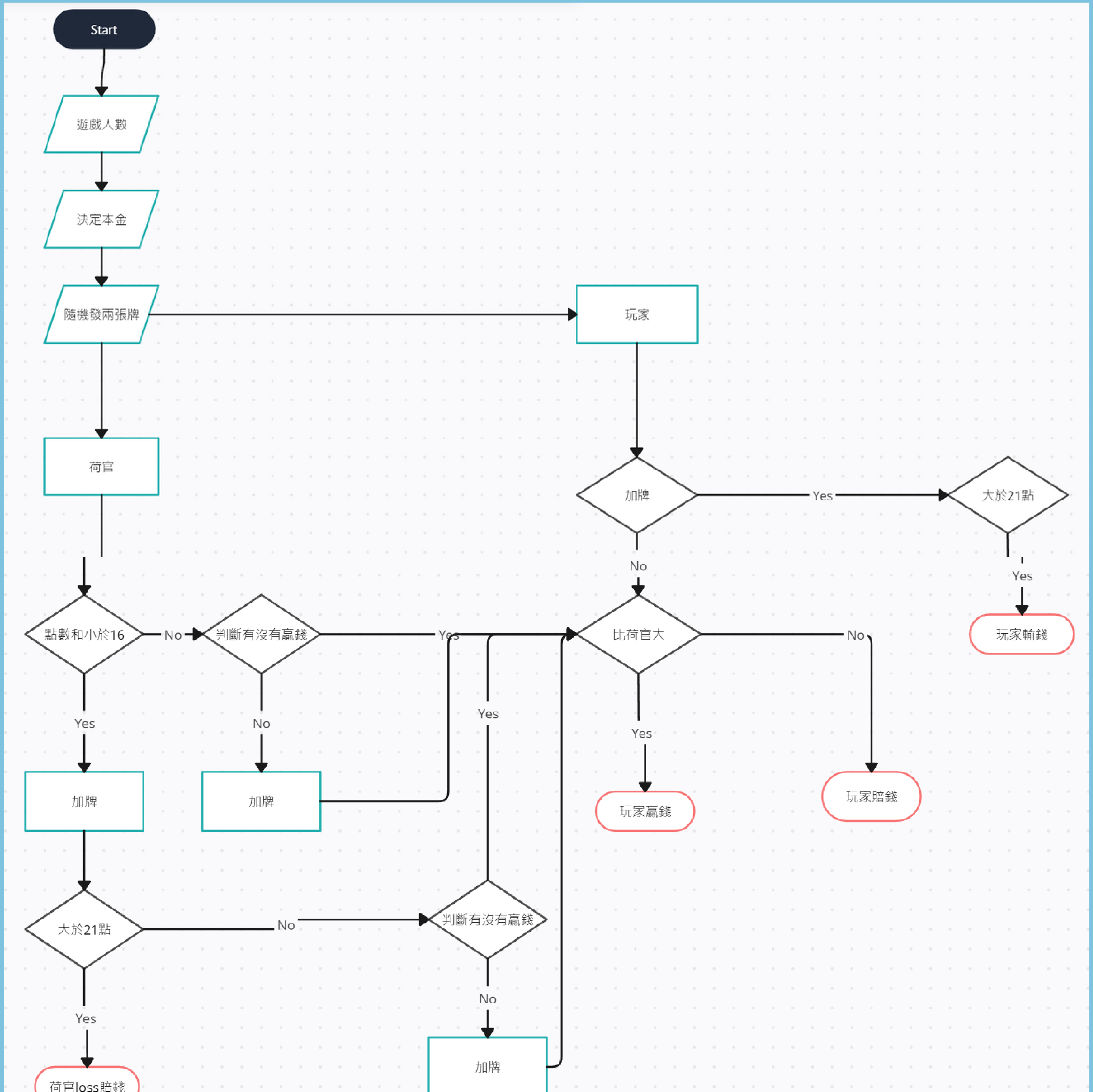
方法
- dealersum
- **Player (+ 1 多...**
- pokersum
- prinf
- translate

Poker
類別

欄位
- card
- deck
- onecard
- onecardsuit
- poker
- suit

方法
- getdeck
- getonecard
- getonecardsuit
- Poker

流程圖



Start
遊戲人數
決定本金
隨機發兩張牌
玩家
荷官
加牌
Yes
大於21點
No
Yes
玩家輸錢
點數和小於16
No
判斷有沒有贏錢
Yes
比荷官大
No
Yes
Yes
No
加牌
加牌
玩家贏錢
玩家賠錢
大於21點
No
判斷有沒有贏錢
Yes
No
荷官loss賠錢
加牌

程式改進

1. 隨機種子：在主程式的srand函式中使用time(NULL)作為隨機種子，這可以根據當前時間設置隨機種子，讓每次運行的隨機數序列更為隨機。不過，為了確保每次運行的結果一致，可以將srand的設置移到程式開頭，而不是每次遊戲迴圈中。
2. 使用函式封裝：在程式碼中，可以將一些重複性的程式片段封裝成函式，提高程式碼的可讀性和可維護性。例如，將獲取一張牌的邏輯封裝成一個函式，可以讓程式碼更簡潔。
3. 遊戲邏輯優化：目前的程式碼只實現了單一玩家和莊家的基本遊戲邏輯，但還有一些遊戲規則和玩家互動的部分可以優化。例如，可以添加遊戲開始和結束的提示，允許玩家決定是否再次遊戲，以及根據莊家的規則自動執行莊家的操作等。

分工表

高浩庭: 流程圖、程式設計、報告
林立:ＰＰＴ製作、程式設計
黃楷竣:程式設計、ＰＰＴ製作
林柏赫:ＰＰＴ製作、程式設計、UML

謝謝聆聽！