資料結構

Data Structure

Lab 05

姓名：曾致嘉

學號：113AB0014

Lab05-Q1

Exercise: Blackjack (Use Queue)

1) Shuffle a standard 52-card deck (Without jokers).

2) Push the shuffled cards into a Queue.

3) Draw cards from the Queue and start the Blackjack.

Code

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <random>
#include <ctime>
using namespace std;
struct Player {// 玩家結構體，包含名稱和分數
    string name; // 玩家名稱
    int score;    // 玩家分數
};
class Card {//建立一個 Card 類別，用來表示單張撲克牌
public:
    string suit
        ; // 花色
    string rank; // 點數
    Card() {}
    Card(string s, string r) : suit(s), rank(r) {}
    void display() const {
        cout << rank << " of " << suit << endl;
    }
};
class Deck {// 牌組類別，負責生成和洗牌
private:
    Card cards[52]; // 52 張牌
    int index; // 目前發牌索引
public:
    Deck() : index(0) {// 初始化牌組
        string suits[] = { "Hearts", "Diamonds", "Clubs", "Spades" };// 四種花色
        string ranks[] = { "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A" };// 13
```

```
張牌
            int k = 0;
            for (int i = 0; i < 4; i++) {// 4 種花色
                for (int j = 0; j < 13; j++) {// 13 種點數
                    cards[k] = Card(suits[i], ranks[j]);
                    k++;
                }
            }
        }

        void shuffleDeck() {// 洗牌
            srand(time(0));
            for (int i = 51; i > 0; i--) {// 從最後一張牌開始，向前遍歷
                int j = rand() % (i + 1);// 生成隨機索引 j
                swap(cards[i], cards[j]);// 交換牌組中的兩張牌
            }
        }
        // 取得指定位置的牌
        Card getCard(int i) const {
            return cards[i];
        }
};
// Queue 類別
class Queue {
private:
        vector<Card> deck; // 用來存放卡牌的容器
        int Front;              // 指向牌堆的前端
        int Rear;               // 指向牌堆的尾端
        int capacity;          // 容器最大容量
public:
        // 建構函式
        Queue(int size) : Front(-1), Rear(-1), capacity(size) {}
        // 插入元素到佇列尾端
        void enqueue(Card card) {
            if(isFull()){
                cout<<"Queue is full"<<endl;
                return ;
            }
```

```
        if (isEmpty()) { // 如果佇列為空，則初始化 Front 和 Rear
            Front = 0;
            Rear = 0;
        }
        else {
            Rear++; // 移動 Rear 指標
        }
        deck.push_back(card);
        //提示:檢查是否滿，如果滿了請印出提示語；如果不是滿的，
        //則將 card 放入到 deck(使用 push_back)，然後將 Rear 指標移動
    }
    // 從佇列中移除最前端元素
    void dequeue() {
        if(isEmpty()){
            cout<<"Queue is Empty"<<endl;
            return ;
        }
        Front++; // 移動 Front 指標
        if (Front > Rear){ // 如果 Front 超過 Rear，則佇列已經清空
            Front = -1;
            Rear = -1;
            deck.clear(); // 清空佇列
        }
        return ;
    }
    Card front() {// 獲取佇列最前端元素
        Front++;//移動 Front 指標
        if (isEmpty()) { // 檢查是否空
            cout << "牌堆為空，無法獲取最前端的牌！" << endl;
            return Card("", "");
        }

        return deck[Front-1];//返回最前端的牌
    }
    bool isEmpty() const {// 判斷佇列是否為空
        if(Front==-1)return 1;//提示:front 和 rear 的關係
        return 0;
    }
```

```
    bool isFull() const {// 判斷佇列是否已滿
        return Rear==(capacity-1);//提示:deck 的大小和 capacity 的關係
    }
};
void initializeDeck(Queue& cardDeck) {// 初始化並洗牌
    Deck deck;
    cout << "初始牌堆: \n";
    for (int i = 0; i < 52; i++) {
        deck.getCard(i).display(); //印出原始牌堆
    }
    deck.shuffleDeck();//洗牌
    cout << "\n 洗牌後的牌堆: \n";
    for (int i = 0; i < 52; i++) {
        deck.getCard(i).display();//印出洗牌後的牌堆
    }
    for (int i=0;i<52;i++){
        cardDeck.enqueue(deck.getCard(i));
    }
    return ;
    //提示:將洗好的牌放入牌堆
    //使用 for 迴圈，將洗好的牌(deck.getCard(i))放入牌堆
}
// 初始化玩家，發兩張牌
void initializePlayer(Player* player, string name, Queue& cardDeck) {
    player->name = name;// 玩家名稱
    player->score = 0;// 玩家分數
    cout << player->name << " 抽到的牌: ";

    for (int i = 0; i < 2; i++) {// 發兩張牌
        //提示:從牌堆中取出一張牌，然後從牌堆中移除這張牌
        //注意:卡牌變數的類型是 Card，並且使用 front 抽牌
        Card drawnCard=cardDeck.front();

        cout << drawnCard.rank << " of " << drawnCard.suit << "   ";//印出抽到的牌

        if (drawnCard.rank == "A") player->score += 1;//計算 A 點數
        else if (drawnCard.rank == "J" || drawnCard.rank == "Q" || drawnCard.rank == "K")
player->score += 10;//計算 JQK 點數
```

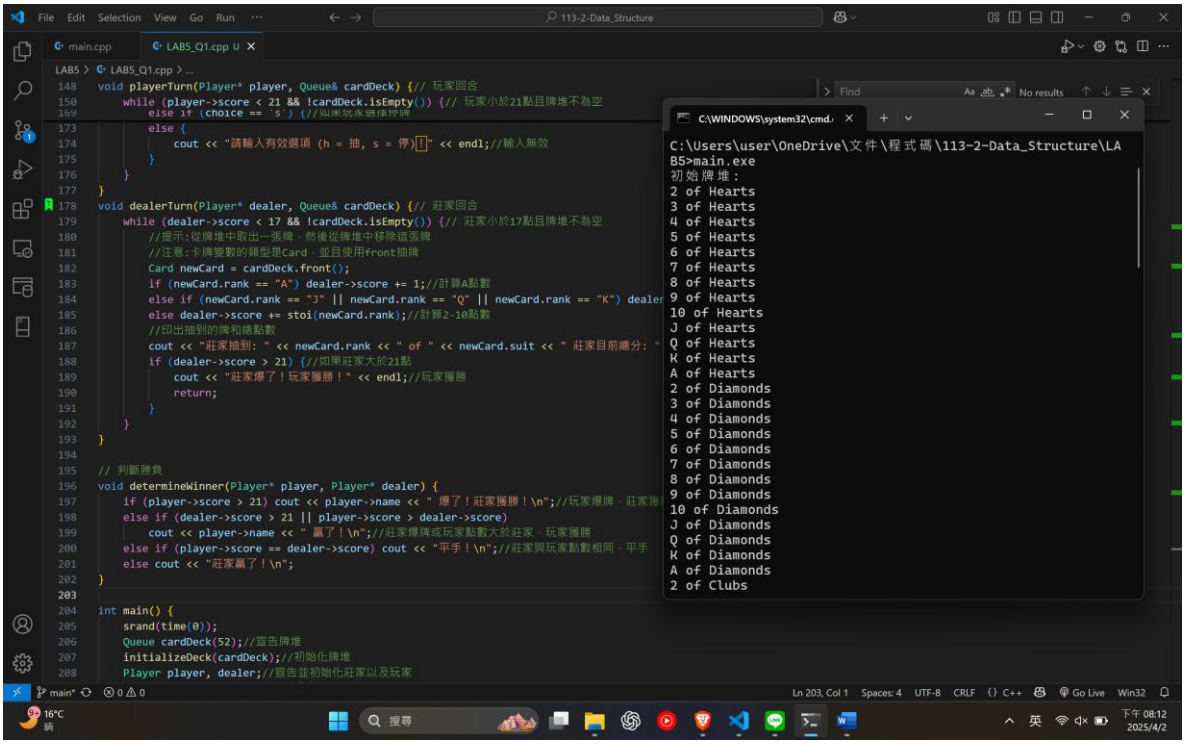
```
            else player->score += stoi(drawnCard.rank);//計算 2-10 點數
        }
        cout << "\n";
        cout << player->name << " 玩家的初始分數: " << player->score << endl;
        //印出玩家的初始點數
}

void playerTurn(Player* player, Queue& cardDeck) {// 玩家回合
        char choice;
        while (player->score < 21 && !cardDeck.isEmpty()) {// 玩家小於 21 點且牌堆不為空
            cout << player->name << " 您的手牌分數目前為: " << player->score << " 要抽牌嗎？(h = 抽, s = 停) : ";//顯示選擇
            cin >> choice;//輸入選擇

            if (choice == 'h') {//如果玩家選擇抽牌
                //提示:從牌堆中取出一張牌，然後從牌堆中移除這張牌
                //注意:卡牌變數的類型是 Card，並且使用 front 抽牌
                Card newCard = cardDeck.front();
                if (newCard.rank == "A") player->score += 1;//計算 A 點數
                else if (newCard.rank == "A" && player->score==10) player->score += 11;
                 //如果 A 和玩家的點數為 10，則 A 的點數計為 11
                else if (newCard.rank == "J" || newCard.rank == "Q" || newCard.rank == "K") player->score += 10; //計算 JQK 點數
                else player->score += stoi(newCard.rank);//計算 2-10 點數
                //印出抽到的牌和總點數
                cout << player->name << " 抽到: " << newCard.rank << " of " << newCard.suit << " 總分: " << player->score << endl;
                if (player->score > 21) {//如果玩家大於 21 點
                    cout << player->name << " 爆掉了！遊戲結束。\n";
                    //玩家爆牌，莊家獲勝
                    return;
                }
            }
            else if (choice == 's') {//如果玩家選擇停牌
                cout << player->name << " 選擇停牌，總分: " << player->score << endl;//印出總點數
                break;
            }
```

```
            else {
                cout << "請輸入有效選項 (h = 抽, s = 停)！" << endl;//輸入無效
            }
        }
}
void dealerTurn(Player* dealer, Queue& cardDeck) {// 莊家回合
        while (dealer->score < 17 && !cardDeck.isEmpty()) {// 莊家小於 17 點且牌堆不為空
            //提示:從牌堆中取出一張牌，然後從牌堆中移除這張牌
            //注意:卡牌變數的類型是 Card，並且使用 front 抽牌
            Card newCard = cardDeck.front();
            if (newCard.rank == "A") dealer->score += 1;//計算 A 點數
            else if (newCard.rank == "J" || newCard.rank == "Q" || newCard.rank == "K")
dealer->score += 10;//計算 JQK 點數
            else dealer->score += stoi(newCard.rank);//計算 2-10 點數
            //印出抽到的牌和總點數
            cout << "莊家抽到: " << newCard.rank << " of " << newCard.suit << " 莊家目前總
分: " << dealer->score << endl;
            if (dealer->score > 21) {//如果莊家大於 21 點
                cout << "莊家爆了！玩家獲勝！" << endl;//玩家獲勝
                return;
            }
        }
}

// 判斷勝負
void determineWinner(Player* player, Player* dealer) {
        if (player->score > 21) cout << player->name << " 爆了！莊家獲勝！\n";
        //玩家爆牌，莊家獲勝
        else if (dealer->score > 21 || player->score > dealer->score)
            cout << player->name << " 贏了！\n";
            //莊家爆牌或玩家點數大於莊家，玩家獲勝
        else if (player->score == dealer->score) cout << "平手！\n";//莊家與玩家點數相同，
平手
        else cout << "莊家贏了！\n";
}

int main() {
        srand(time(0));
```

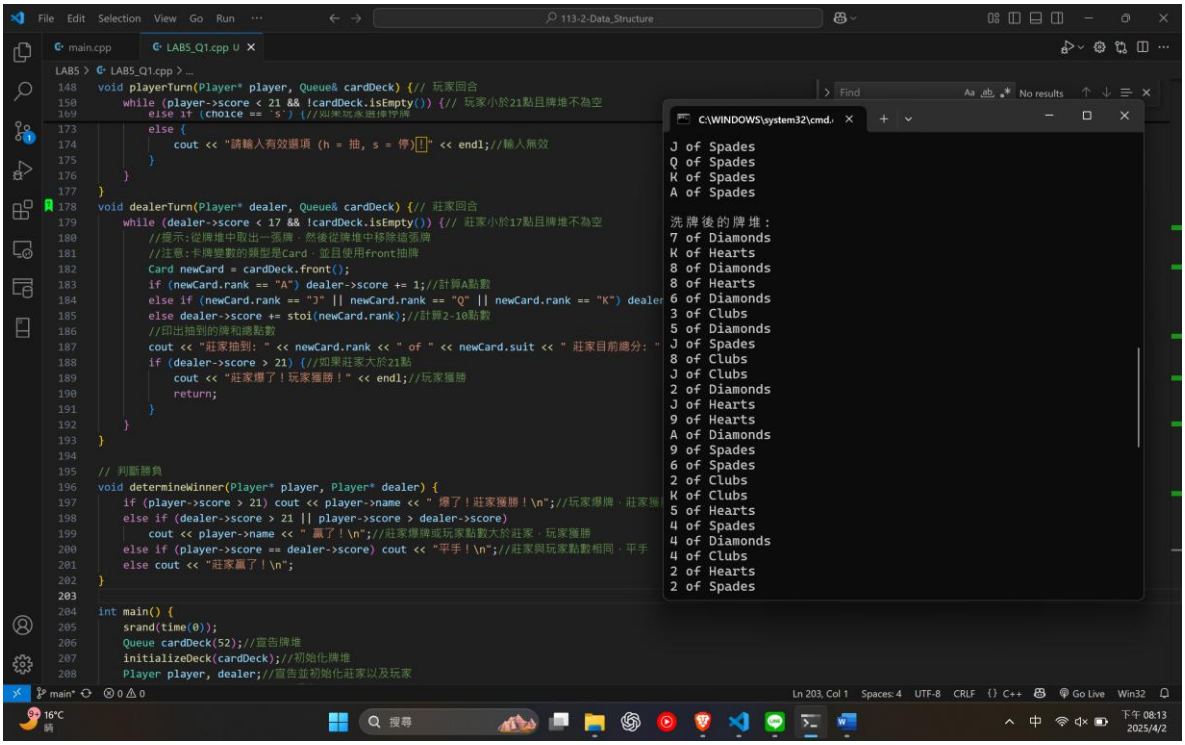
```
    Queue cardDeck(52);//宣告牌堆
    initializeDeck(cardDeck);//初始化牌堆
    Player player, dealer;//宣告並初始化莊家以及玩家
    initializePlayer(&player, "玩家", cardDeck);
    initializePlayer(&dealer, "莊家", cardDeck);

    playerTurn(&player, cardDeck);
    if (player.score <= 21) {
        cout << "\n 莊家回合...\n";
        dealerTurn(&dealer, cardDeck);
        determineWinner(&player, &dealer);
    }

    return 0;
}
```

Discussion Section

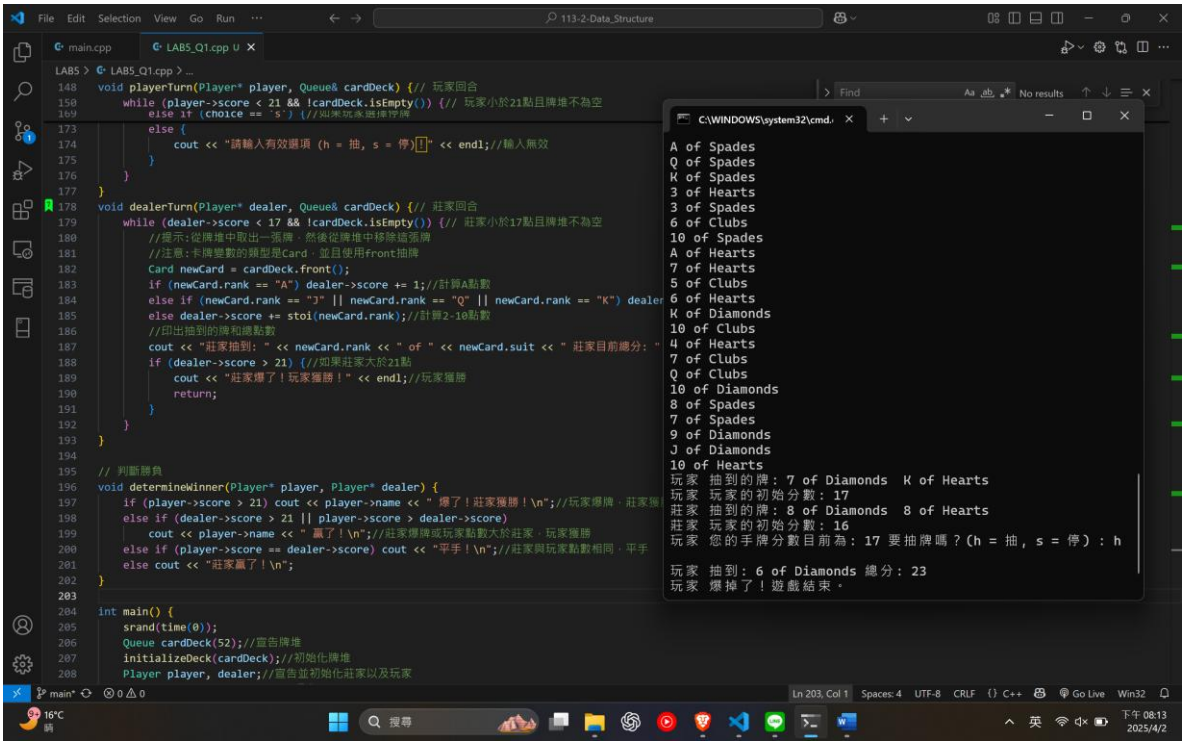
初始牌堆：



洗牌後牌堆：


J of Spades
Q of Spades
K of Spades
A of Spades
洗牌後的牌堆：
7 of Diamonds
K of Hearts
8 of Diamonds
8 of Hearts
6 of Diamonds
3 of Clubs
5 of Diamonds
J of Spades
8 of Clubs
J of Clubs
2 of Diamonds
J of Hearts
9 of Hearts
A of Diamonds
9 of Spades
6 of Spades
2 of Clubs
K of Clubs
5 of Hearts
4 of Spades
4 of Diamonds
4 of Clubs
2 of Hearts
2 of Spades

遊玩過程：


A of Spades
Q of Spades
K of Spades
3 of Hearts
3 of Spades
6 of Clubs
10 of Spades
A of Hearts
7 of Hearts
5 of Clubs
6 of Hearts
K of Diamonds
10 of Clubs
4 of Hearts
7 of Clubs
Q of Clubs
10 of Diamonds
8 of Spades
7 of Spades
9 of Diamonds
J of Diamonds
10 of Hearts
玩家 抽到的牌：7 of Diamonds K of Hearts
玩家 玩家的初始分數：17
莊家 抽到的牌：8 of Diamonds 8 of Hearts
莊家 玩家的初始分數：16
玩家 您的手牌分數目前為：17 要抽牌嗎？(h = 抽, s = 停) : h
玩家 抽到：6 of Diamonds 總分：23
玩家 爆掉了！遊戲結束。