計算機程式設計 HW7-2report

103061223 李俊穎

1. Introduction of the problem and your solution

The homework 7-2 is to complete the card game called Royal Casino. We need to write the rules like pair, combine, tailing, build and call, and try to use STL vectors to improve data structures. Although now we don't need to do it by vectors, I still use it because it is more efficient.

Because this time the instructor has given us some part of codes, what we need to complete is the judge equation and some function or loop, and call them correctly. I judge the condition by the switch-case and if-else, as well as some bool variables.

2. Implementation details ,Additional features of your program (data structure, flows and algorithms)

The rules of casino include pair, call, combine, build and tailing, the following is the part of pair：

```
case 3:
    cout<<"Choose card you want to pair"<<endl;
    cout<<"the card's sequence:";
    cin>>sequence;
    playerpoint.push_back(player[sequence-1]);
    for(int i=0;i<table.size();i++)
    {
        if(table[i].rank()==player[sequence-1].rank())//check if the numbers are same
        {
            playerpoint.push_back(table[i]);//store in the graveyard
        }
        if(table[i].rank()!=player[sequence-1].rank())
        {
            plate.push_back(table[i]);//the other cards stored to the temporary place
        }
    }
    player[sequence-1]=player[player.size()-1];
    player.pop_back();table.clear(); //do the same things as the case 2
    plate.swap(table);plate.clear();
    shownowcards();
    break;
```

How the pair function run is check the rank name from player to table, and pop_back they from their primary array.

The following is the part of combine：

```
case 1:
    cout<<"Choose card you want to use"<<endl;
    cout<<"the card's sequence:";
    cin>>sequence;    //sequence means the card's sequence on the screen
    playerpoint.push_back(player[sequence-1]);  //store the card in graveyard you throw becaus
    for(int j=1,length=table.size() ;j<table.size();j++,length--) //length is to make the i no
    {
        int i;
        for(i=0;i<length;i++)
        {
            if((table[i].rank()+table[i+j].rank())==player[sequence-1].rank()) //check the com
            {
                playerpoint.push_back(table[i]);   //store the cards in graveyard
                playerpoint.push_back(table[i+j]); //store the cards in graveyard
                Card change=table[i];    // change the combine two cards to the forward place
                table[i]=table[0];
                table[0]=change;
                Card change1=table[i+j];
                table[i+j]=table[1];
                table[1]=change1;


              if(playerpoint[playerpoint.size()-1].rank()==table[i+j].rank()) //jump out of this
                  break;
            }
        }
        if(playerpoint[playerpoint.size()-1].rank()==table[i+j].rank())   //jump out of this loop
            break;
    }
    for(int i=0,n=1;i<table.size();i++)
    {
        if(table[i].rank()==playerpoint[playerpoint.size()-1].rank()||(table[i].rank()==playerpoir
        {
            table[i]=table[table.size()-1];   //if yes,then change them to the end place and pop_b
            table.pop_back();
            n++;
        }
        if(n>2)  //move the first two cards out,then we can break this loop
            break;
    }
    player[sequence-1]=player[player.size()-1]; // move the card you throw to the handcard's end,t
    player.pop_back();
    shownowcards();
    break;
```

The combine is checking all the condition of sum of two cards on the table. If so, change the two cards to the front, and judge them again and change them to the end then pop_back it.

The following is the trail：

```
case 4:
    cout<<"Choose card you want to trail"<<endl;
    cout<<"the card's sequence:";
    cin>>sequence;
    table.push_back(player[sequence-1]); //add the handcard to the table
    player[sequence-1]=player[player.size()-1];
    player.pop_back();//move the card from handcard
    shownowcards();
    break;
default:
    break;
```

Tailing is just push_back the table and pop_back the hand cards.

The following is the call：

```
case 2:
    if(player1call==false)
    {
        cout<<"Choose card you want to call"<<endl;
        cout<<"the card's sequence:";
        cin>>sequence;
        cout<<"now the card you called is: "<<sequence<<endl;
        player1call=true; //to make me enter the else if loop in the next round
        table.push_back(player[sequence-1]);
        player[sequence-1]=player[player.size()-1];
        player.pop_back();
        shownowcards();
    }
    else if(player1call==true)
    {   player1call=false;  //revise it back to false then next time I can use it again
        cout<<"Choose card you want to call"<<endl;
        cout<<"the card's sequence:";
        cin>>sequence;
        playerpoint.push_back(player[sequence-1]);
        for(int i=0;i<table.size();i++)
        {
            if(table[i].rank()==player[sequence-1].rank())//find the last round card
            {
                playerpoint.push_back(table[i]);//store the cards in graveyard
            }
            if(table[i].rank()!=player[sequence-1].rank())
            {
                plate.push_back(table[i]);//the other cards stored to the temporary p
            }
        }
        player[sequence-1]=player[player.size()-1];
        player.pop_back();table.clear();//clean the primary table
        plate.swap(table);//store the things back from the temporary place
        plate.clear();//clean the temporary place
```

And this time I add my additional feature of my program, I create two function that one can detect if now the player can use pair and the other detect if now the player can use combine and I put then in the for loop to make them check every round when playing game. For example, the check function that check the pair：

```
void check(vector<Card>& player,vector<Card>& table)//check if player can use pair
{
    bool exam=false;
    for(int i=0;i<player.size();i++)
    {
        for(int j=0;j<table.size();j++)
        {
            if(player[i].rank_name()==table[j].rank_name())
                exam=true;
        }
    }
    if(exam)
        cout<<"you can use pair"<<endl;
}
```

I compare them one to one to know if there are the same number on the hand cards and table, then player can follow by the hint without check their hand cards.

This time I didn’t finish the build rules because my program will have lots of bugs. In other words, I don’t figure out how to judge this condition, so I don’t finish it in my program. I think I have to consider more time on it and discuss with classmates.