《数据结构》课程实践报告

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 院、系 | 计算机学院 | | 年级专业 | 21计科 | 姓名 | 方浩楠 | 学号 | 2127405048 |
| 实验布置日期 | | 9.14 | | 提交  日期 | 10.5 | | 成绩 |  |

课程实践实验3：小猫钓鱼纸牌游戏

## 问题描述及要求

A和B两个同学玩简单的纸牌游戏，每人手里有n张牌，两人轮流出牌并依次排列在桌面上，每次出掉手里的第1张牌，出牌后如果发现桌面上有跟刚才打出的牌的数字相同，则把到相同的那张牌结束的全部牌按次序放在自己手里牌的末尾。当一人手中牌先出完时，游戏结束，对方获胜。

如n为5，A手里的牌依次为2 3 5 6 ，B手里的牌依次为1 5 4 2 ，

A出2，B出1；此时，桌子上从前往后依次为21，A手里是356，B手里是542；

A出3，B出5；此时，桌子上从前往后依次为2135，A手里是56，B手里是42；

接着A出5，发现前面有一张5，则把两个5都拿掉，这时他手里有6 5 5；桌子上的牌依次为2 1 3；接着B出4，桌子上的牌是2134，他手里的牌是2；

接着A出6， A手里剩55，B出2，发现前面有2，全部收走到自己手里，它手上的牌即是：264312桌子上没有牌；依次类推，直到某人先出完牌为止，则对方是胜者。

编写程序，模拟显示出桌子上和A、B两位同学手里牌的变化过程，并判断谁是胜者以及此时胜者手里的牌的状态。如上例中，每次A和B各出一张牌后，桌上和手里牌的变化状态如下所示：

the cards on the desk:2 1

the cards in A's hand:3 5 6

the cards in B's hand:5 4 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

the cards on the desk:2 1 3 5

the cards in A's hand:5 6

the cards in B's hand:4 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

the cards on the desk:2 1 3 4

the cards in A's hand:6 5 5

the cards in B's hand:2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

the cards on the desk:

the cards in A's hand:5 5

the cards in B's hand:2 6 4 3 1 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

the cards on the desk:5 2

the cards in A's hand:5

the cards in B's hand:6 4 3 1 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

the cards on the desk:6

the cards in A's hand:5 2 5

the cards in B's hand:4 3 1 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

the cards on the desk:6 5 4

the cards in A's hand:2 5

the cards in B's hand:3 1 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

the cards on the desk:6 5 4 2 3

the cards in A's hand:5

the cards in B's hand:1 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

the cards on the desk:6 1

the cards in A's hand:5 3 2 4 5

the cards in B's hand:2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

the cards on the desk:6 1 5 2

the cards in A's hand:3 2 4 5

the cards in B's hand:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

胜者为A，A剩下的牌从上到下依次为3 2 4 5

## 概要设计

本次实验主要是要运用不同的数据结构来储存玩家手中的牌以及桌面牌堆中的牌。我们可以发现，玩家手中的牌在出牌时永远是出最先进入玩家手中的牌，而摸牌时永远是将摸来的牌放置在最后面，属于First In First Out，因此应该使用queue来储存玩家手中的牌。而桌面牌堆中的牌永远是对牌堆顶部的一张或多张牌进行操作，属于Last In First Out，因此应该使用stack来储存牌堆顶部的牌

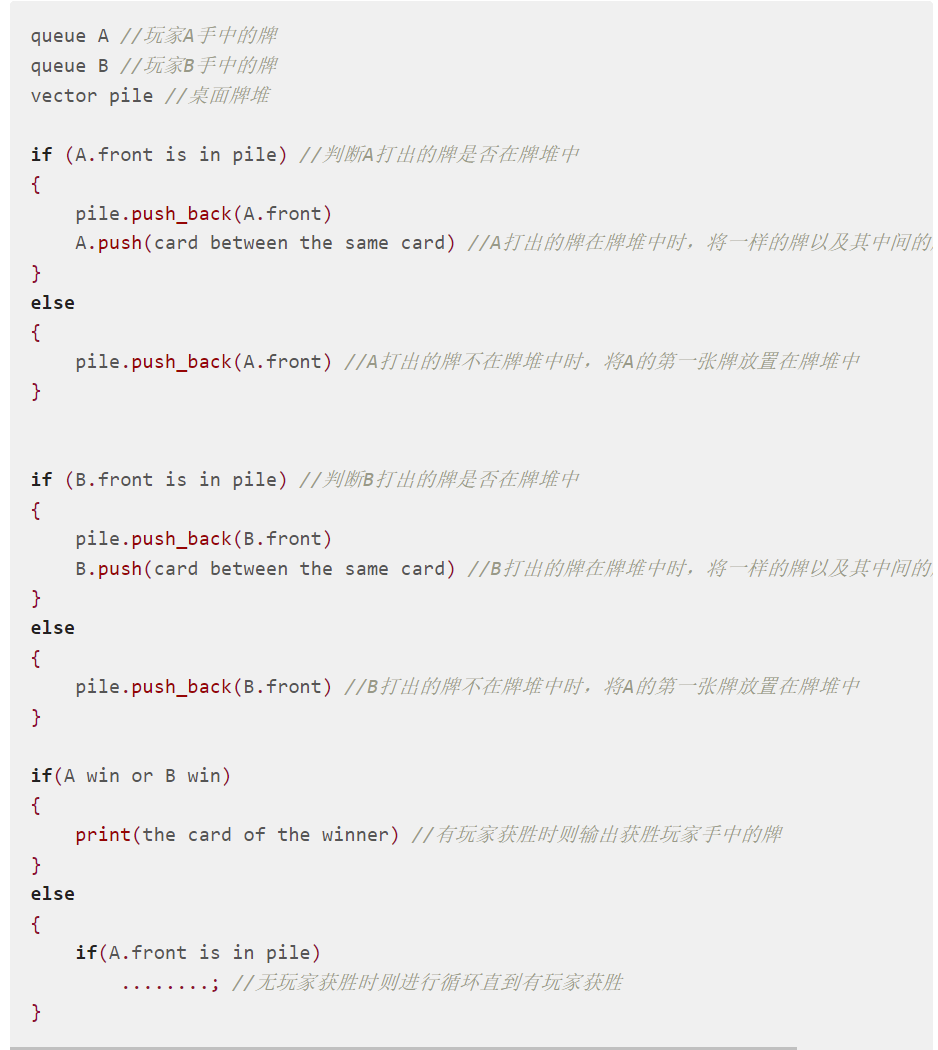
系统的功能为令用户在终端输入玩家A和玩家B手中的牌，然后运行，并且最终在终端输出结果。程序界面如下:



程序调用了标准模板库(STL)中的queue,stack以及vector。在本程序中为了易于编写，因此在储存桌面牌堆时运用的数据结构为vector而非stack。

程序中含有PlayerPile(),whoWin(),printPile(),printPileOfWinner()四个函数。其中PlayPile()函数用来模拟每次玩家出牌以及打出牌堆中有的牌时将牌堆中的牌放入玩家手中的过程。whoWin函数用来判断是否有人获胜。printPile函数用来输出两名玩家手中的牌以及桌面牌堆中的牌printPileOfWinner是在有玩家获胜时输出获胜玩家手中的牌。

## 三、详细设计



## 四.实验结果

测试输入: 2 3 5 6

1 4 5 7

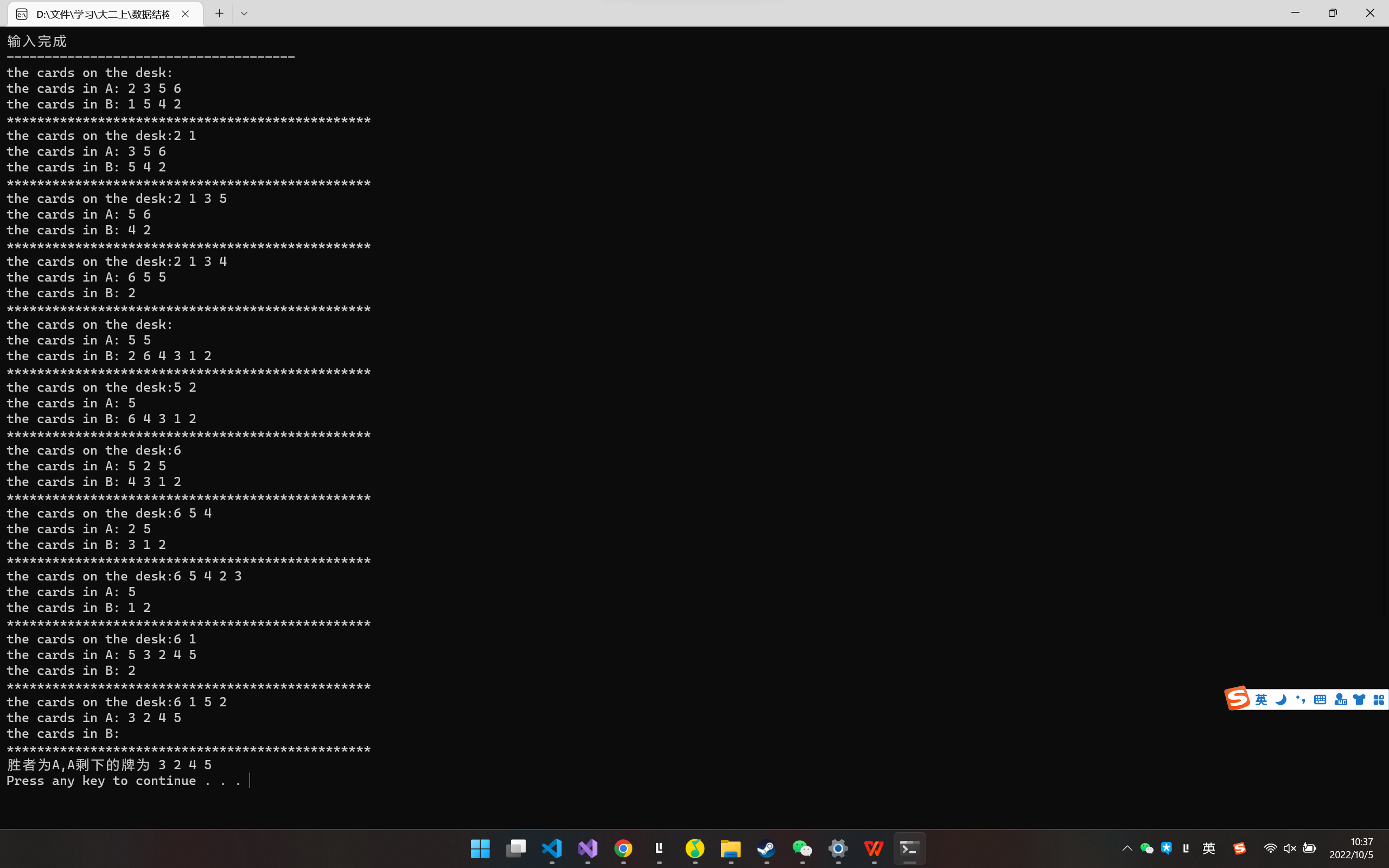
测试输出:



测试输入:2 3 5 6

1 5 4 2

测试输出:



## 实验分析与探讨

时间复杂度:

程序可以将桌面上的牌堆用链栈(Linked stack)来储存。因为桌面上的牌堆中的牌的大小一直在变化，使用vector或者sequence stack时，由于容器中的数量一直在变，导致插入元素的时间复杂度。而改用linked stack后，由于链不是一段连续的内存空间，使得在link stack中插入元素的时间复杂度。同时，由于桌上的牌堆不需要使用到随机访问，而只需要使用顺序访问，因此使用linked stack和sequence stack对stack中的元素进行顺序访问时的时间复杂度相同。所以使用链栈可以有效的节省访问桌面上的牌堆以及向桌面上的牌堆中放置牌的时间。

## 小结

本次实验让我了解了队列以及栈的使用，希望可以进一步学习更多关于数据结构的知识

输入的要求:只能输入int类型，并且数字之间都要用空格隔开。输入结束时需要输入空格来表示输入的结束。

## 附录：源代码

实验环境:Visual Studio 2020

（1）文件名 cat\_fishing.cpp

代码

/\*

A和B两个同学玩简单的纸牌游戏，每人手里有n张牌，两人轮流出牌并依次排列在桌面上，每次出掉手里的第1张牌，出牌后如果发现桌面上有跟刚才打出的牌的数字相同，

则把到相同的那张牌结束的全部牌按次序放在自己手里牌的末尾。当一人手中牌先出完时，游戏结束，对方获胜。

如n为5，A手里的牌依次为2 3 5 6 ，B手里的牌依次为1 5 4 2 ，

A出2，B出1；此时，桌子上从前往后依次为21，A手里是356，B手里是542；

A出3，B出5；此时，桌子上从前往后依次为2135，A手里是56，B手里是42；

接着A出5，发现前面有一张5，则把两个5都拿掉，这时他手里有6 5 5；桌子上的牌依次为2 1 3；接着B出4，桌子上的牌是2134，他手里的牌是2；

接着A出6， A手里剩55，B出2，发现前面有2，全部收走到自己手里，它手上的牌即是：264312桌子上没有牌；依次类推，直到某人先出完牌为止，则对方是胜者。

\*/

#include<iostream>

#include<queue>

#include<stack>

#include<vector>

#include<algorithm>

using namespace std;

void PlayerPile(queue<int>&pile\_in\_player, vector<int>&pile\_on\_table) //PlayerPile函数用来控制每一次A和B打出的牌

{

int temp\_card = pile\_in\_player.front(); //取出玩家手中的第一张牌

pile\_in\_player.pop();

int card\_position = -1;//card\_position用来表示玩家手中的第一张牌在牌堆中的位置。若玩家手中的第一张牌不在牌堆中，则card\_position为-1。在牌堆中则返回在牌堆中的位置

for (int i = 0; i < pile\_on\_table.size(); i++) //判断玩家手中的第一张牌在牌堆中的位置

{

if (pile\_on\_table[i] == temp\_card)

{

card\_position = i;

break;

}

}

if (card\_position != -1)//玩家打出的牌在牌堆中有一样的牌

{

pile\_on\_table.push\_back(temp\_card);//将玩家手中的牌放入牌堆中

for (int i = pile\_on\_table.size()-1; i >= card\_position; i--) //将牌堆中把直到相同牌的所有牌逆序放置在玩家手中

{

pile\_in\_player.push(pile\_on\_table[i]);

}

pile\_on\_table.erase(pile\_on\_table.begin() + card\_position, pile\_on\_table.end());

}

else if(card\_position == -1)//玩家打出的牌在牌堆中没有一样的牌

{

pile\_on\_table.push\_back(temp\_card);

}

}

const string whoWin(queue<int>pile\_A, queue<int>pile\_B)//用来判断是否有玩家获胜。若有玩家获胜，则返回玩家的名字。若没有玩家获胜则返回None

{

if (pile\_A.size() == 0) { return "B"; }

else if (pile\_B.size() == 0) { return "A"; }

else { return "None"; }

}

void printPile(queue<int>pile\_A, queue<int>pile\_B, vector<int>pile\_table) //输出两名玩家手中的牌以及牌堆中的牌

{

cout << "the cards on the desk:";

for (int i = 0; i < pile\_table.size(); i++)

{

cout << pile\_table[i] << " ";

}

cout << endl;

cout << "the cards in A:" << " ";

int pile\_A\_size = pile\_A.size();

for (int i = 0; i < pile\_A\_size; i++)

{

cout << pile\_A.front() << " ";

pile\_A.push(pile\_A.front());

pile\_A.pop();

}

cout << endl;

cout << "the cards in B:" << " ";

int pile\_B\_size = pile\_B.size();

for (int i = 0; i < pile\_B\_size; i++)

{

cout << pile\_B.front() << " ";

pile\_B.push(pile\_B.front());

pile\_B.pop();

}

cout << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

void PrintPileOfWinner(string winner,queue<int>pile\_winner) //有玩家获胜时输出获胜玩家手中的牌

{

cout << "胜者为" << winner << "," << winner << "剩下的牌为" << " ";

int pile\_winner\_size = pile\_winner.size();

for (int i = 0; i < pile\_winner\_size; i++)

{

cout << pile\_winner.front() << " ";

pile\_winner.push(pile\_winner.front());

pile\_winner.pop();

}

}

int main()

{

queue<int>pile\_A; //玩家A手中的牌

queue<int>pile\_B; //玩家B手中的牌

cout << "请输入A手中的牌,每张牌之间用空格隔开，输入完后请输入0" << endl;

while (true)

{

int a;

cin >> a;

if (a != 0)

{

pile\_A.push(a);

}

else

{

break;

}

}

cout << "请输入B手中的牌,每张牌之间用空格隔开，输入完后请输入0" << endl;

while (true)

{

int a;

cin >> a;

if (a != 0)

{

pile\_B.push(a);

}

else

{

break;

}

}

//test

//pile\_A.push(2); pile\_A.push(3); pile\_A.push(5); pile\_A.push(6);

//pile\_B.push(1); pile\_B.push(5); pile\_B.push(4); pile\_B.push(2);

vector<int>pile\_table; //牌堆中的牌

while (true)

{

if (whoWin(pile\_A, pile\_B) == "None")

{

printPile(pile\_A, pile\_B, pile\_table);

PlayerPile(pile\_A,pile\_table); //A出牌

if (whoWin(pile\_A, pile\_B) == "None")

{

PlayerPile(pile\_B, pile\_table);//B出牌

if (whoWin(pile\_A, pile\_B)=="None")

{

;

}

else

{

goto Label;

}

}

else

{

goto Label;

}

}

else

{

goto Label;

}

}

Label: //一旦有人获胜则跳出while循环

printPile(pile\_A, pile\_B,pile\_table);

string winner = whoWin(pile\_A, pile\_B);

if (winner == "A")

{

PrintPileOfWinner("A", pile\_A);

}

else if (winner == "B")

{

PrintPileOfWinner("B", pile\_B);

}

cout << endl;

system("pause");

return 0;

}