《数据结构》课程实践报告

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 院、系 | 计算机学院 | | 年级专业 | 计算机 | 姓名 |  | 学号 |  |
| 实验布置日期 | | 2019-9-27 | | 提交  日期 | 2019-10-11 | | 成绩 |  |
|  | |  | |  |  | |  |  |

课程实践实验3：小猫钓鱼纸牌游戏

## 一、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的牌，则把从相同的那张牌开始的全部牌按次序放在自己手里的牌的末尾。当一个人手中的牌先出完时，游戏结束，对方获胜。

如n为5，A手里的牌依次为2 3 5 6 1，B手里的牌依次为1 5 4 2 9；

A出2；

B出1；

A出3；

B出5；

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 1 5 5；

桌子上的牌依次为2 1 3；

B出4；

A出6；

B出2，发现前面有一张2，则把从2开始的牌全部拿掉，这时他手里有9 2 1 3 4 6 2；

桌子上没有牌了；

A出1；

B出9；

A出5；

B出2；

依次类推，直到某人先出完牌为止，则对方是胜者。

编写程序，利用栈和队列，判断谁是胜者。

## 二、概要设计

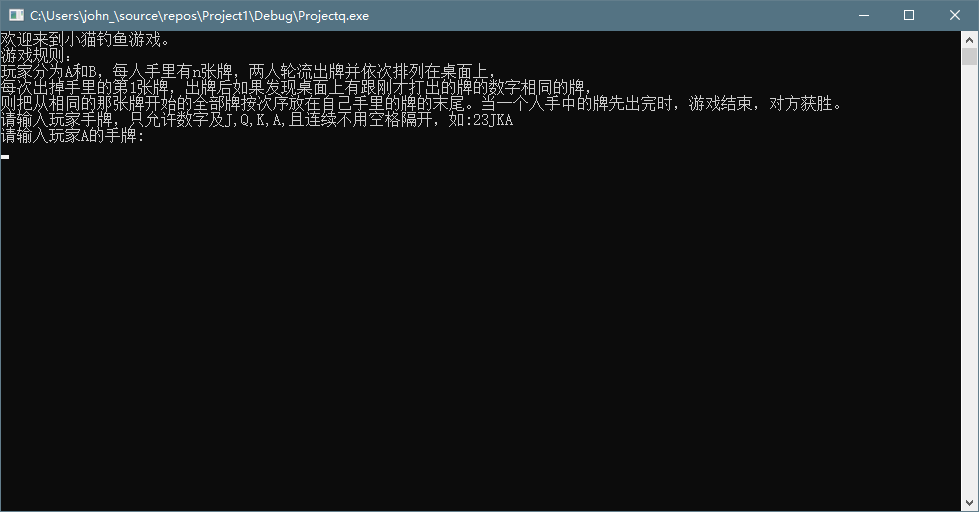
1. **对实验内容的理解和二次概括。**

**本实验可利用栈和队列的功能，由于玩家的手牌是从头部出牌，向尾部加牌，符合队列的定义，可将两个玩家的手牌定义为A，B两个队列，然后由于桌上的牌是从尾部加牌，也从尾部收牌，可将桌上的牌定义为一个table栈。通过队列和栈的功能实现本游戏。**

1. **系统的功能列表**

**输入两个玩家的手牌，可自动进行游戏流程，给出每一轮玩家的手牌，出牌情况和桌上的牌，以及收牌的情况。**

1. **程序运行的界面设计**



1. **确定总体设计思路，采用何种数据结构，设计哪些类，各类的作用 ，类方法的介绍，类之间的关系描述**

**设计玩家A，B两个队列，代表他们手上的手牌，可用push收回手牌，用pop出牌，另外定义一个table栈，代表桌上的牌，可用push把玩家出的牌压入栈中，也可用pop把桌上的牌收回玩家手中。**

1. **程序结构设计，包括：对已有程序的使用，自己将设计哪些程序文件，各部分关系描述。**

**设计一个initialize函数用于初始化两个玩家的手牌，读入键盘输入的字符。**

**main\_process函数代表整个游戏流程。**

**check函数用于检查当前玩家是否符合收牌条件。**

**playcards是玩家出牌或手牌的过程。**

**showcards用于输出当前玩家的手牌。**

## 三、详细设计

开始

输入玩家初始手牌

否

输入是否合法

是

initialize函数

压入A,B栈中

出牌

check函数

把满足条件的牌收回

是否满足收牌条件

是

是

否

playcards函数

双方手上是否都有牌剩余

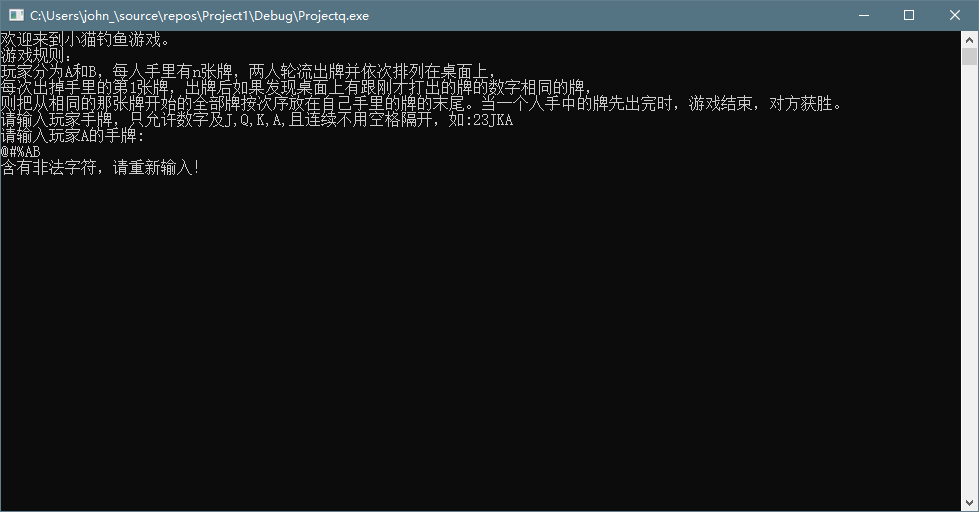
输出赢家

main\_process函数

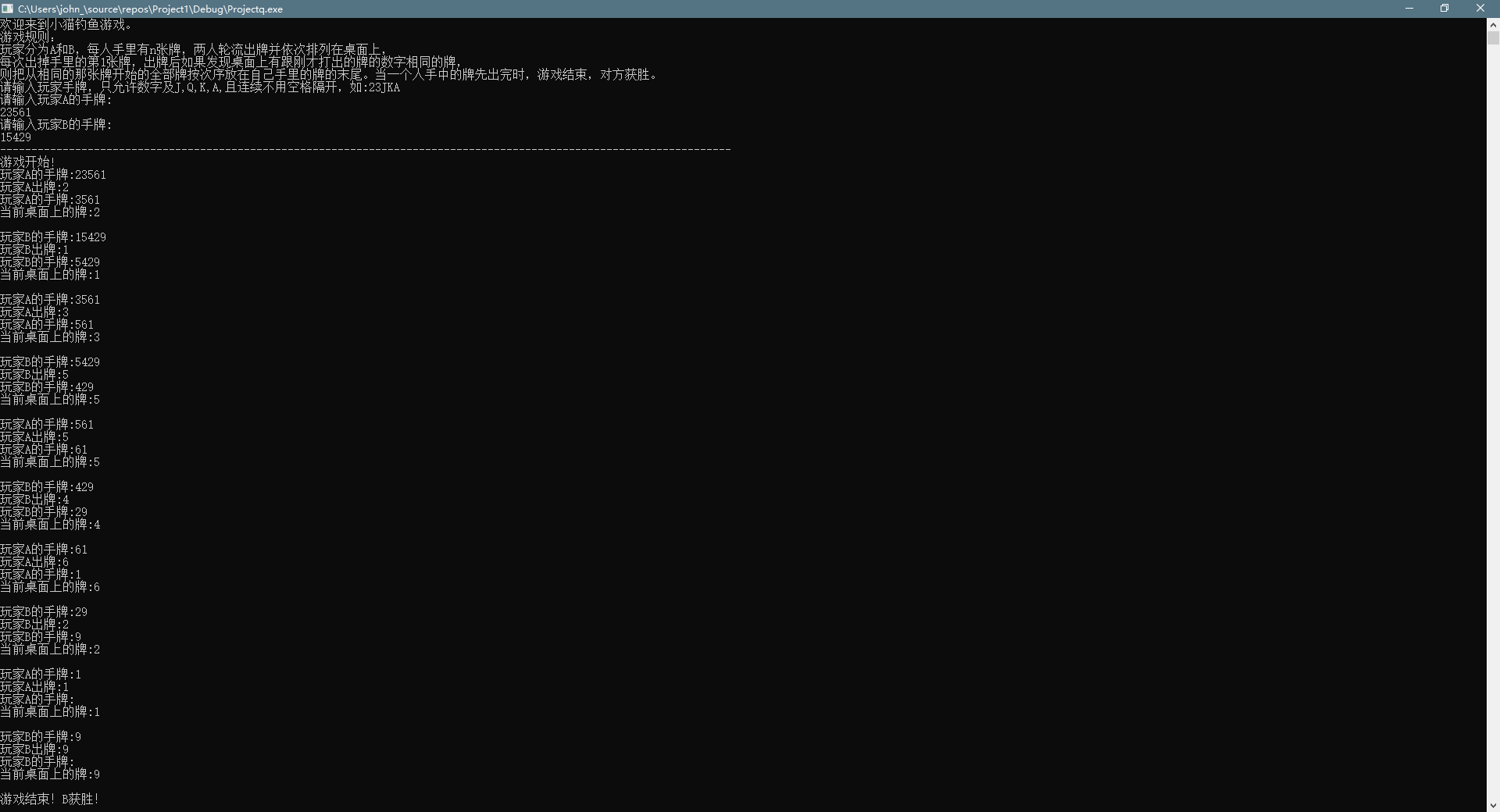
## 四、实验结果

紧扣题目要求设计提供相应的测试方法和结果。可以给出具体的测试用例，每个测试用例一般可列出：

* 测试输入：@#%AB
* 测试目的：测试是否能判断字符非法
* 正确输出：含有非法字符，请重新输入！
* 实际输出：



* 测试结论：通过
* 测试输入：23561 15429
* 测试目的：测试程序是否正常运行
* 正确输出：B获胜！
* 实际输出：



* 测试结论：通过

## 五、实验分析与探讨

从结果来看，本次实验主要使用了基本的栈和队列的功能，主要的基本方法是从一个栈或队列的顶端元素赋给另一个栈或队列，再将其弹出。利用该方法可实现出牌收牌的基本功能。本实验主要难点在于，如何在一个玩家出牌后，判断它是否满足收牌条件和收牌时的循环操作。涉及到的困难是出牌后再判断桌上牌面是否满足条件较难，必须在其出牌前将其存在暂时的一个参数内，再与桌上牌比对，同时也不能忘记要把牌放进桌上的栈。比较复杂的一个问题是需要注意收牌时的顺序，收牌是从桌上的牌的栈尾一个个弹出，但一个个弹出后需要倒序放进玩家手牌中，因此需要一个临时的栈来转换顺序。还有一个困难是无法直接输出每一轮当前各个人和桌上的牌，必须要运用一个循环来一个个弹出并输出。

时间复杂度：O(n2)

## 六、小结

本次实验实现了在键盘输入双方玩家手牌之后，自动运行小猫钓鱼游戏，并输出每轮玩家和桌上的牌，最后输出结果。

## 附录：源代码

#include <queue>

#include <iostream>

#include <cstdio>

#include <string>

#include <stack>

using namespace std;

void initialize(queue<char> &p)

{

string s;

int i = 0,flag=1;

while (flag)

{

getline(cin, s);

for (i = 0; i < s.size(); i++)

{

if (!(isdigit(s[i]) || s[i] == 'J' || s[i] == 'K' || s[i] == 'Q' || s[i] == 'A'))

{

cout << "含有非法字符，请重新输入！" << endl;

break;

}

if (i == s.size() - 1) flag = 0;

}

}

for (i = 0; i < s.size(); i++)

{

p.push(s[i]);

}

}

bool check(string s, char c)

{

int i = 0;

if (s == "") return false;

while (i<s.size())

{

if (s[i] == c) return true;

i++;

if (i == s.size()) return false;

}

}

string showcards(queue<char> p)

{

string s;

while (!p.empty())

{

s += p.front();

p.pop();

}

return s;

}

void playcards(string player, queue<char> &A, stack<char> &table)

{

string s = "";

char tempcard;

stack<char> tempcards;

cout << "玩家"<<player<<"的手牌:" << showcards(A) << endl;

if (check(s, A.front()))

{

tempcard = A.front();

A.pop();

cout << "玩家"<<player<<"出牌:" << tempcard << endl;

tempcards.push(tempcard);

while (table.top() != tempcard)

{

tempcards.push(table.top());

table.pop();

s.pop\_back();

}

tempcards.push(table.top());

table.pop();

s.pop\_back();

cout << "玩家"<<player<<"收回:";

while (!tempcards.empty())

{

A.push(tempcards.top());

cout << tempcards.top();

tempcards.pop();

}

cout << endl;

}

else

{

cout << "玩家"<<player<<"出牌:" << A.front() << endl;

s += A.front();

table.push(A.front());

A.pop();

}

cout << "玩家"<<player<<"的手牌:" << showcards(A) << endl;

cout << "当前桌面上的牌:" << s << endl;

cout << endl;

}

void main\_process(queue<char> &A, queue<char> &B, stack<char> &table)

{

string s = "";

char tempcard;

stack<char> tempcards;

cout << "------------------------------------------------------------------------------------------------------------------------"<<endl;

cout << "游戏开始!" << endl;

while (!(A.empty() || B.empty()))

{

playcards("A", A, table);

playcards("B", B, table);

}

string winner;

if (A.empty()) winner = "B";

else if (B.empty()) winner = "A";

cout << "游戏结束！" << endl << winner << "获胜！" << endl;

}

void main()

{

queue<char> A, B;

stack<char> table;

cout << "欢迎来到小猫钓鱼游戏。\n游戏规则：\n玩家分为A和B，每人手里有n张牌，两人轮流出牌并依次排列在桌面上，\n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的牌，\n则把从相同的那张牌开始的全部牌按次序放在自己手里的牌的末尾。当一个人手中的牌先出完时，游戏结束，对方获胜。" << endl;

cout << "请输入玩家手牌，只允许数字及J,Q,K,A,且连续不用空格隔开，如:23JKA" << endl;

cout << "请输入玩家A的手牌:" << endl;

initialize(A);

cout << "请输入玩家B的手牌:" << endl;

initialize(B);

main\_process(A, B, table);

system("pause");

}