《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 | 彭光 | 学号 | 2127405052 |
| 实验布置日期 | | 2022-09-14 | | 提交  日期 | 2022-09-30 | | 成绩 |  |

课程实践实验三：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

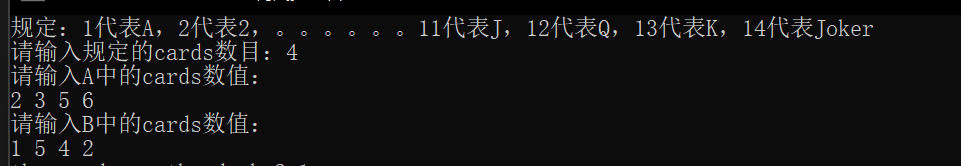
对实验内容的理解：

这个小猫钓鱼纸牌游戏其实就是对向量或者线性表等存储结构采用获取下标进行删除和添加操作，我们只需要利用while循环（条件是A卡牌没了或者B卡牌没了）到最后的时候来判断是谁赢就可以了。

系统功能列表：

自动进行小猫钓鱼纸牌游戏

程序运行的界面设计：



首先会出现屏幕提示规定数字代表什么纸牌，其次会出现：请输入规定的cards数目，接下来对A和B分两次进行输入，注意用空格隔开。

总体设计思路：

首先在看到这个问题及其描述的时候，我当时觉得先创建个desk向量将桌上的纸牌给收集起来，再用栈A和栈B进行存储A和B手中的卡牌。但是后来就发现了一个问题，我要对A上面的纸牌进行读取的时候只能把元素一一pop出来，就不能达到const的标准。后来我就都用了向量，发现更简单了一点。

首先我写了7个函数（有一个erase\_0的函数感觉可以去掉，我当时debug时发现有错误，就写了这个函数，其实不是它的问题）

首先来介绍一下Push函数，在这个函数里的接口时A或B的向量，desk向量和两个下标，主要进行的操作是：将desk向量到相同牌结束的全部牌都push进A或B向量中；

Erase函数主要进行的操作是：将desk向量里到相同牌结束的全部牌都给删除掉。

erase\_0函数主要进行的操作是：将A或B里的0号元素给pop出去。

get\_index函数主要进行的操作是:如果desk向量里出现和要push进去的元素一样，就返回其下标，否则就返回-1。

Print函数主要进行的操作是：将desk，A，B向量里的数据一一的给输出出来

IsWiner函数主要进行的操作是：如果A向量空，则返回B，否则返回B

接下来最重要的是process\_by\_cards函数，主要进行的操作是：建立一个while循环，如果出现A或B中有一个空了那就终止循环。

这个process\_by\_cards主要的设计思路为：

利用while(!(!X.size()||y.size()))来控制输出，我利用get\_index（desk,A[0]）是否等于-1来判断是否出现重复的元素，如果没有出现即返回的值是-1，我们就将A或B向量中的0号元素给push进desk里，如果是返回值不是-1就用Push函数将重复元素之间的元素都放到对应向量中，再将desk里的给删除掉；接下来用Print函数进行输出，循环之后调用IsWiner函数决定谁是获胜者。

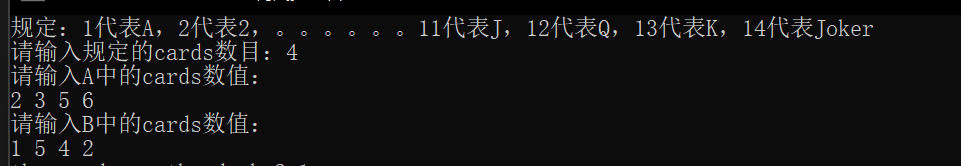
## 详细设计

## 微信图片_20220930195734

其中的难点部分主要是process\_by\_cards函数中while循环的编写部分：利用get\_index(desk,A[0])?=-1作为一个条件来判断是将元素push\_back进去还是将desk向量里的重复元素之间的元素给erase出来。如果不等于-1，那么我是没有将A[0]给先push进desk向量里，而是从找到重复元素的下标开始一直push进A向量中，然后将A【0】继续push进A中，再将A【0】erase出来。

## 实验结果

测试输入：



测试目的：测试给出的普通样例是否可以正确输出

正确输出：

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

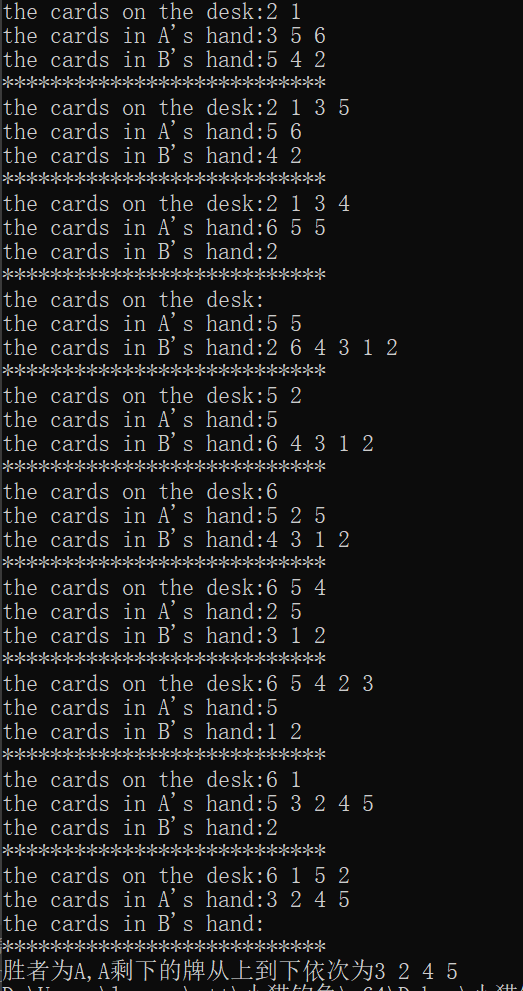
the cards in A's hand:3 2 4 5

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

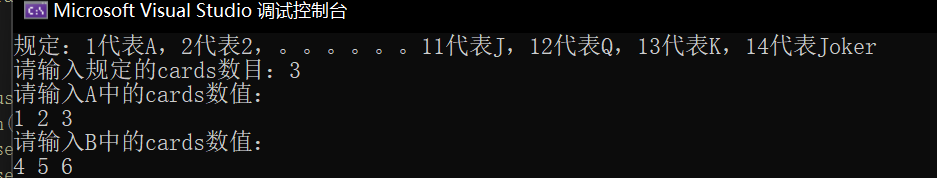
实际输出：



错误原因：无

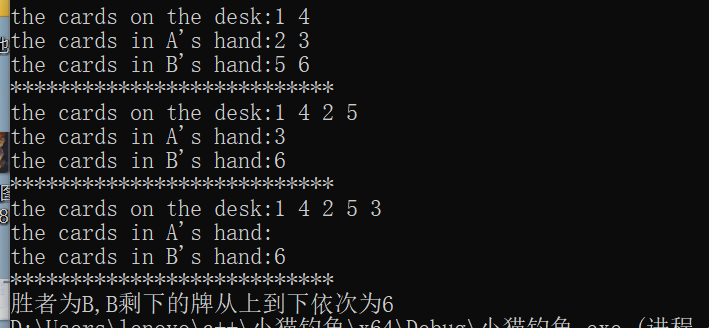
测试结论：通过

测试输入：

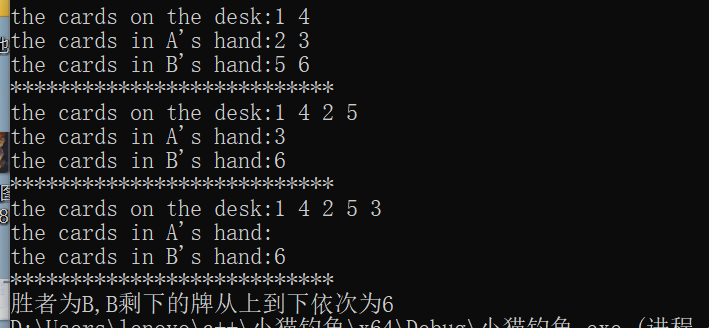


测试目的：测试一下特殊样例是否输出正确

正确输出：



实际输出：



错误原因：无

测试结论：通过

## 实验分析与探讨

时间复杂度：O（n^2）

空间复杂度：O（n）

测试结果的分析：针对A和B手中的卡牌有重叠部分还是没有重叠部分两方面进行测试，如果是重叠部分，那么必有一方卡牌消失，另一方卡牌数>0；但是如果不是重叠，那么B是肯定赢，且只有一张卡牌（建立在A和B卡牌数一致）。

实现过程中遇到的问题：

1. 首先是我没有先把A【0】给push进desk中，再进行重复部分的删减和添加，导致A[0]不能被随意的erase掉；
2. 发现如果一开始A和B卡牌都不一样（参考我的测试样例2），最后B中向量是空的。

解决方法：

1. 因为我的Push函数和Erase函数都是关于A【0】的，所以我只需要把eraseA【0】的函数放在这两者之后就可以了。

2.因为我是利用两个if条件，两者互不打扰，所以在A空的时候后面的if条件句还会继续运行，所以在第一个if条件句后加入了if (x.empty())

{

Print(a, x, y);

break;

}

## 小结

在做这个实验的时候，因为清楚的知道小猫钓鱼的玩法，所以我能很快的写出函数。我觉得完善的方向可能是自己控制卡牌数和在玩的过程中调用random函数，这样可能更公平一些。完成了要求的所有内容。

补充说明：输入的时候先输入卡牌数目，在以空格为间隔输入卡牌元素（先输完A再输B）

## 附录：源代码

**1.实验环境：Visual Studio 2022**

2、

1. cards.cpp

代码：

#include<iostream>

#include<stack>

#include<string>

#include<vector>

using namespace std;

//将desk上面相同卡牌及中间的卡牌存入对应的A或B的向量中

void Push(vector<int>& a,vector<int>&b, int x, int y)

{

for (int i = y; i >= x; i--)

{

a.push\_back(b[i]);

}

}

//删除desk向量中的重复之间的元素

void Erase(vector<int>& a, int x, int y)

{

for (int i = y; i >= x; i--)

{

a.erase(a.begin() + i);

}

}

//删除向量中的0号元素

void erase\_0(vector<int>& a)

{

if (!a.empty())

{

a.erase(a.begin());

}

}

//将desk所对应的向量中如果出现卡牌和要出的一样，就返回对应的下标

int get\_index(vector<int>& b,int x)

{

for (int i = 0; i < b.size(); i++)

{

if (b[i] == x)

return i;

}

return -1;

}

//输出cards的过程

void Print(vector<int>&a,const vector<int>&x,const vector<int>&y)

{

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

for (int i = 0; i < a.size(); i++)

{

cout << a[i] << " ";

}

cout << endl<<"the cards in A's hand:";

int count1 = 0, count2 = 0;

while (count1<x.size())

{

cout << x[count1] << " ";

count1++;

}

cout << endl << "the cards in B's hand:";

while (count2<y.size())

{

cout << y[count2] << " ";

count2++;

}

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

}

//判断是A赢还是B赢

string IsWiner(vector<int>&x,vector<int>&y)

{

if (!x.size())

return "B";

if (!y.size())

return "A";

}

//开始运行小猫钓鱼的全过程

void process\_by\_cards(vector<int> a,vector<int> x,vector<int> y)

{

while (!(x.empty()||y.empty()))

{

if (get\_index(a, x[0]) == -1)

{

a.push\_back(x[0]);

x.erase(x.begin() );

}

else

{

x.push\_back(x[0]);

Push(x, a, get\_index(a, x[0]), a.size() - 1);

Erase(a, get\_index(a, x[0]), a.size() - 1);

erase\_0(x);

}

if (x.empty())

{

Print(a, x, y);

break;

}

if (get\_index(a, y[0]) == -1)

{

a.push\_back(y[0]);

y.erase(y.begin());

}

else

{

y.push\_back(y[0]);

Push(y, a, get\_index(a, y[0]), a.size() - 1);

Erase(a, get\_index(a, y[0]), a.size() - 1);

erase\_0(y);

}

Print(a, x, y);

}

cout << "胜者为" << IsWiner(x, y) << "," << IsWiner(x, y) << "剩下的牌从上到下依次为";

if (IsWiner(x, y) == "A")

{

for (int i = 0; i < x.size(); i++)

cout << x[i] << " ";

}

else

{

for (int i = 0; i < y.size(); i++)

cout << y[i] << " ";

}

}

int main()

{

cout << "规定：1代表A，2代表2，。。。。。。11代表J，12代表Q，13代表K，14代表Joker"<<endl;

int n;

cout << "请输入规定的cards数目：";

cin >> n;

vector<int> A, B;

vector<int> desk;

desk.clear();

cout << "请输入A中的cards数值："<<endl;

int count = 0,s;

while (++count <= n)

{

cin >> s;

A.push\_back(s);

}

cout << "请输入B中的cards数值：" << endl;

int count1 = 0;

while (++count1 <= n)

{

cin >> s;

B.push\_back(s);

}

process\_by\_cards(desk, A, B);

return 0;

}