银行业务

1651573 刘客

功能介绍

- 输入说明：输入为一行正整数，其中第一数字N（N<=1000）为顾客总数，后面跟着N位顾客的编号。编号为奇数的顾客需要到A窗口办理业务，为偶数的顾客则去B窗口。数字间以空格分隔。
- 输出说明：按照业务处理完成的顺序输出顾客的编号。数字键以空格分隔，但是最后一个编号不能有多余的空格。

编译说明

- 在windows平台下的.exe文件
- 在Linux平台下的out文件

一 . 设计

1. 数据结构设计

> 题目要求我们模拟两个银行柜台进行处理。由银行柜台的性质可知，客户的处理顺序应该是先入先出的，因此我们应该选用**队列**来模拟银行柜台。

2. 结构体设计

- ## myQuene 节点
  - 成员变量

成员名称	属性	类型	描述
first	public	int	队列头指针
last	public	int	队列尾指针

|num|public|int*|存储队列元素的数组

二 . 函数实现

1. 进行初始化

> 定义队列对象，并进行初始化

```
 myQuene queneA;
 myQuene queneB;
 //初始化A、B队列首尾指针
 queneA.first = 0;
 queneB.first = 0;
 queneA.last = 0;
 queneB.last = 0;
```

> 定义人数变量和计时变量(因为两柜台处理速度不同)

```
 int peopleNum;
int customerNum;
//将处理计数时间设置为1,则每当countTime为偶数时,对B队列进行处理
int countTime = 1;
cout << "请输入顾客总数: ";
cin >> peopleNum;
if (peopleNum != 0) {
    cout << "请输入顾客编号: ";
}
```

2. 存储客户编号

> 利用for循环存储客户编号，当编号为奇数时，存入队列A，并将其尾指针++，当编号为偶数时，存入队列B，并将其尾指针++

```
 //输入顾客编号
for (auto i = 1; i <= peopleNum; i++) {
    cin >> customerNum;
    if (customerNum % 2 != 0) {
        //将尾指针递增,存储数据
        queneA.num[queneA.last++] = customerNum;
    }
    else {
        queneB.num[queneB.last++] = customerNum;
    }
}
```

3. 处理客户逻辑

> 定义布尔变量，用来判断队列是否输出完毕
> 题目要求最后一位不可以输出空格

```
 //用来判断是否队列输出完毕,以此来判断是否需要输出" "(最后一位不应该输出空格)
bool judgeAend = (queneA.first == queneA.last);
bool judgeBend = (queneB.first == queneB.last);
```

> 对特殊情况进行判断

```
 if (peopleNum != 0) {
    cout << "处理顺序为: ";
}
else {
    cout << "无顾客,不需要进行处理!";
}
```

> 处理逻辑
> 利用循环进行处理，每处理一次将时间加一，并将处理队列的首指针++。
> 在每处理完之后还会进行首指针是否等于尾指针的判定，如果相等，则表明该队列已经处理结束，将布尔变量标签标为true。
> 如果两个布尔标签均为true，则跳出循环。

```
 while (!judgeAend || !judgeBend) {
    if (!judgeAend) {
        //当A队列未处理完毕时,进行处理
        cout << queneA.num[queneA.first++];
        //如果处理一个编号后,发现队列已经处理完毕
        if (queneA.first == queneA.last) {
            //将标志量设置为true
            judgeAend = true;
            if (judgeBend) {
                //如果B也已经处理完毕,则表明所有的顾客编号都已经处理完毕,此时不需要输出空格,直
接break;
                break;
            }
        }
        //如果不然,则输出空格
        cout << " ";
    }
    //因为A队列的速度是B队列的两倍,所以只有计时为偶数时(countTime初始值为1),才对B进行处理,其他
逻辑与A队列相同
    if (countTime % 2 == 0 && !judgeBend) {
        cout << queneB.num[queneB.first++];
        if (queneB.first == queneB.last) {
            judgeBend = true;
            if (judgeAend) {
                break;
            }
        }
        cout << " ";
    }
    countTime++;
}
```

三 . 程序运行效果

> 和样例数据保持一致,并且最后一位无空格